ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
Факультет комп'ютерно-інформаційних технологій та автоматизації

Кафедра прикладної математики та інформатики

«До захисту допущено»
Завідувачка кафедри
Наталія МАСЛОВА
(підпис) (Ім'я та ПРІЗВИЩЕ)

«_____»_____2024 р.

Випускна кваліфікаційна робота

бакалавра
(освітній ступінь)

на тему: «Розробка програмного забезпечення для інтерактивного навчального комплексу з вивчення іноземних мов»

зі спецчастиною: «Розробка модулів та інтерфейсу інтерактивного навчального комплексу з використанням вебтехнологій»

Виконав: студент 4 курсу, групи ІПЗ-20
(шифр групи)
спеціальності 121 Інженерія програмного забезпечення
освітньої програми 121 Інженерія програмного забезпечення

Сергієнко Олександр Олександрович
(прізвище та ініціали) (підпис)

Керівник доц. каф. ПМІ, к.т.н., доц. Тетяна АЛТУХОВА
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Рецензент Доц. каф. ЕТКІ ДонНТУ, к.т.н., доц. Ганна ШЕЇНА
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент (підпис)

Луцьк – 2024р.

ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітній ступінь бакалавр
Напрям підготовки (спеціальність) 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувачка кафедри ПМІ
_____ Наталія МАСЛОВА
"____" _____2024 р.

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Сергієнку Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення для інтерактивного навчального комплексу з вивчення іноземних мов
зі спецчастиною

Розробка модулів та інтерфейсу інтерактивного навчального комплексу з використанням вебтехнологій

Керівник роботи Алтухова Т. В., к.т.н., доц. каф. ПМІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від " 25 " квітня 2024 року № 164
2. Строк подання студентом роботи "10" червня 2024 року
3. Вихідні дані до роботи результати науково-дослідної роботи, матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1) Аналіз предметої області та постановка задачі;
2) Аналіз інструментів розробки та реалізації програмного забезпечення;
3) Розробка програмного застосунку;
4) Тестування розробки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) робота містить 27 рисунків, 1 таблицю, інший матеріал, у кількості, достатній для демонстрації результатів кваліфікаційної роботи бакалавра

6. Консультанти розділів роботи

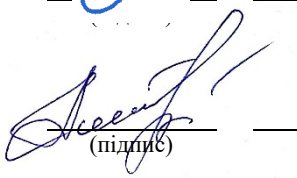
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	к.т.н., доц., доц. ПМІ Назарова І.А.		

7. Дата видачі завдання__25.04.2024__

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	15.04.2024 – 16.04.2024	
2	Постановка завдання	17.04.2024 – 18.04.2024	
3	Огляд тематики роботи, актуальність роботи, формулювання мети на задач дослідження	19.04.2024 – 21.04.2024	
4	Опис та обґрунтування основних ідей роботи	22.04.2024 – 24.04. 2024	
5	Розробка програмного забезпечення	25.04. 2024 – 26.04.2024	
6	Дослідження запропонованих ідеї, аналіз результатів дослідження, формулювання висновків та тенденцій	27.05. 2024 – 03.05. 2024	
7	Оформлення пояснювальної записки	04.05. 2024 – 10.06. 2024	
8	Нормоконтроль пояснювальної записки та додаткових матеріалів	11.06. 2024 – 16.06. 2024	
9	Рецензування роботи	17.06. 2024 – 24.06. 2024	
10	Захист роботи	26.06.2024	

Студент Сергієнко О.О.
(прізвище та ініціали)

Керівник роботи  Алтухова Т.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сергієнко О. О. Розробка програмного забезпечення для інтерактивного навчального комплексу з вивчення іноземних мов з використанням вебтехнології / Випускна кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення». ДВНЗ ДонНТУ, Луцьк, 2024.

Об’єктом дослідження є весь процес розробки від ідеї до реалізації та впровадження програмного забезпечення.

Предметом дослідження є специфічні аспекти розробки програмного забезпечення для навчального комплексу, зокрема, використання вебтехнологій для створення інтерактивного середовища для вивчення іноземних мов, що містить в собі аналіз потреб користувачів, методи взаємодії з програмним забезпеченням, психологічні аспекти ефективного навчання.

Наукова новизна: розробка нових алгоритмів взаємодії з користувачем для поліпшення засвоєння матеріалу, інтеграція інтерактивних елементів у вебзастосунок.

Практична цінність: розроблений застосунок дозволяє створити зручне та ефективне середовище для навчання, що може бути використане у навчальних закладах, самостійного навчання, підвищити доступність навчальних матеріалів та зробити процес навчання більш ефективним для широкого кола користувачів.

Для розробки програмного забезпечення було використано мови програмування JavaScript та Python, фреймворки Vue та Flask та редактор коду VSCodium. Використана ORM-система, що має назву SQLAlchemy, яка дозволяє використовувати більшість реляційних баз даних [1].

ІНТЕРАКТИВНИЙ НАВЧАЛЬНИЙ КОМПЛЕКС, ВИВЧЕННЯ ІНОЗЕМНИХ МОВ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ВЕБТЕХНОЛОГІЇ, ЕФЕКТИВНЕ НАВЧАННЯ, ПЕРСОНАЛІЗОВАНЕ НАВЧАННЯ, ПСИХОЛОГІЧНІ АСПЕКТИ НАВЧАННЯ.

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 9
1.1 Аналіз сучасних розробок в цій галузі 9
1.2 Аналіз методів вивчення іноземних мов 12
1.3 Постановка задач дослідження 13
1.4 Висновки за розділом 1 14
2 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ ТА РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15
2.1 Вибір мов програмування та фреймворків 15
2.1.1 Мова програмування Python та зв’язані програмні інструменти 15
2.1.2 Мова програмування JavaScript та его середовище виконання Node.js 18
2.2 Редактор коду VSCodium 20
2.3 Висновки за розділом 2 21
3 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ 22
3.1 Побудова архітектури програмного продукту 22
3.2 Розробка алгоритмів роботи додатка 24
3.3 Розробка бази даних 33
3.4 Розробка структури графічного інтерфейсу користувача 37
3.5 Висновки за розділом 3 40
4 ТЕСТУВАННЯ РОЗРОБКИ 42
4.1 Аналіз методів тестування вебдодатка 42
4.2 Тестування розробленого вебдодатка 44
4.3 Розробка інструкції користувача 48
4.4 Висновки до розділу 4 53
ЗАГАЛЬНІ ВИСНОВКИ 55
ПЕРЕЛІК ПОСИЛАНЬ 56
ДОДАТОК А ТЕКСТ ПРОГРАМНОГО ЗАСТОСУНКУ……58
ДОДАТОК Б ЗАУВАЖЕННЯ НОРМОКОНТРОЛЕРА 224

ВСТУП

Обґрунтування вибору теми дослідження. Розробка програмного забезпечення для інтерактивного навчального комплексу з вивчення іноземних мов з використанням вебтехнологій є актуальною темою оскільки є тенденція на зростання популярності онлайн-навчання та віддалених форматів навчання, що вимагає розробки нових технологій, які забезпечують ефективні методи навчання, особливо у галузі вивчення іноземних мов.

Оскільки, кожен учень має свої унікальні потреби та способи вивчення. розробка програмного забезпечення, яке може персоналізувати навчальний процес для кожного користувача, стає дедалі важливішою.

Використання саме вебтехнологій дозволяє створювати інтерактивні навчальні середовища, які можна запустити на будь-якому пристрої, що має доступ до інтернету.

Загалом**,** у сучасному світі зростає потреба вивчення іноземних мов для міжкультурного спілкування та співпраці. Програмне забезпечення для вивчення іноземних мов може допомогти зробити цей процес більш доступним та ефективним для широкого кола людей.

Мета роботи – обґрунтування та розробка ефективного програмного забезпечення для вивчення іноземних мов, яке спрямоване на здійснення навчання вдома або у будь-якому зручному місці та часі. Розробка застосунку має на меті підвищити доступність навчання іноземних мов, покращити якість освіти у цій сфері та сприяти розвитку міжкультурного спілкування.

Об’єктом дослідження є сам процес розробки від формулювання завдання до впровадження та використання програмного забезпечення для навчання та містить в себе аналіз вимог користувачів, розробку застосунку, тестування, впровадження та подальшу підтримку системи.

Предметом дослідження є конкретні аспекти такої розробки, як методи взаємодії з користувачем, використання вебтехнологій для створення інтерактивного середовища, адаптація змісту для ефективного вивчення мови, аналіз ефективності та результативності такого програмного забезпечення в контексті навчання іноземних мов.

Наукова новизна полягає у впровадженні нових ідеї та підходів у сфері розробки програмного забезпечення для навчання іноземних мов з використанням веб технологій, що може принести користь як науковій спільноті, так і практиці навчання.

Практична цінність полягає у забезпеченні доступності, ефективності та зручності навчання іноземних мов за допомогою спеціалізованого програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасних розробок в цій галузі

Існують багато програмних застосунків, що пропонують різні підходи та функціонал для вивчення іноземних мов з використанням вебтехнологій і дозволяють користувачам навчатися у власному темпі та у зручний час.

Duolingo – це один з найвідоміших і популярних інтерактивних додатків для вивчення іноземних мов (рис. 1.1). Він пропонує користувачам різні рівні складності, вправи на вивчення граматики, словниковий запас і вимову [2].

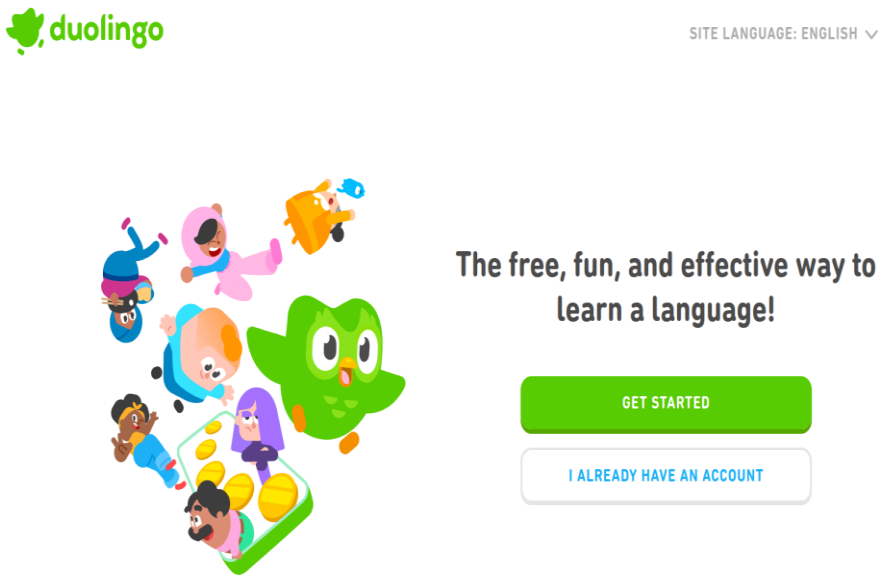


Рисунок 1.1 – Головна сторінка Duolingo за джерелом 2

LibreLingo – це відкрите програмне забезпечення, яке розвивається спільнотою, і має на меті створення вільного та відкритого середовища для навчання мови (рис. 1.2). Воно надає користувачам можливість вивчати мови за допомогою різноманітних вправ, тестів та інтерактивних уроків [3].

Рисунок 1.2 – Головна сторінка LibreLingo за джерелом 3

UneeBee – це також відкрите програмне забезпечення, яке надає змогу створювати короткі та зручні уроки, у тому числі з вивчення мов, адаптовані для початківців та тих, хто вже має деякий рівень володіння мовою (рис. 1.3) [4].

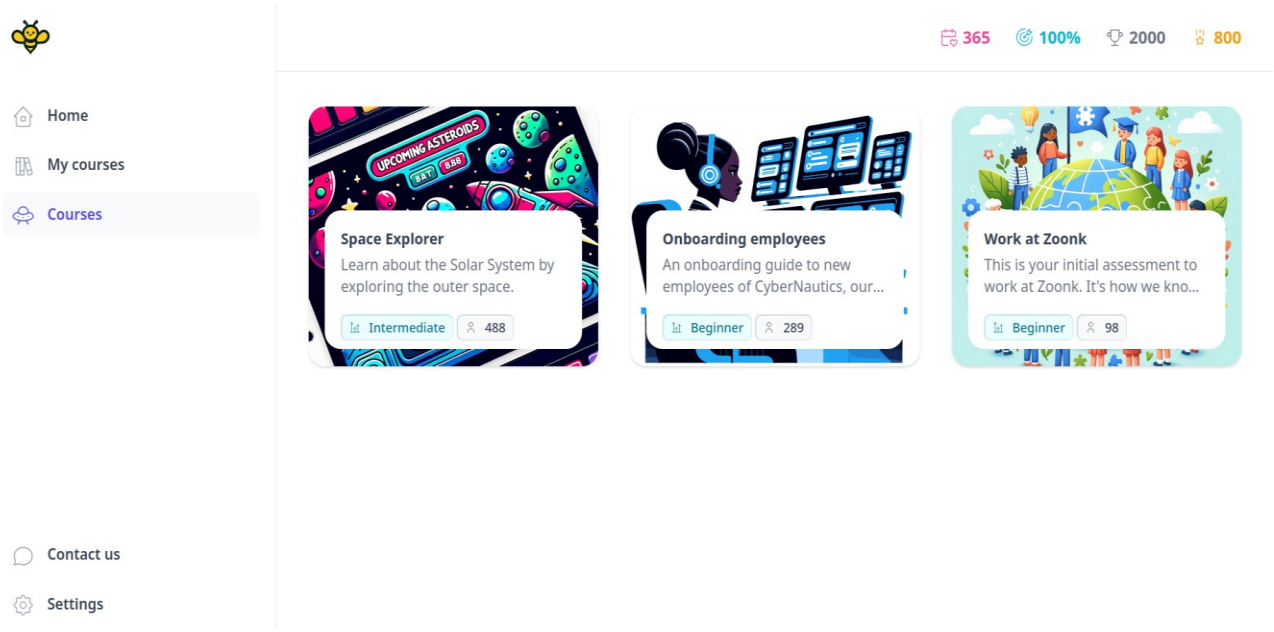


Рисунок 1.3 – Основна сторінка UneeBee за джерелом 4

Memrise – це додаток, який використовує методику навчання з використанням карток для полегшення запам'ятовування нових слів та виразів (рис. 1.4) [5].

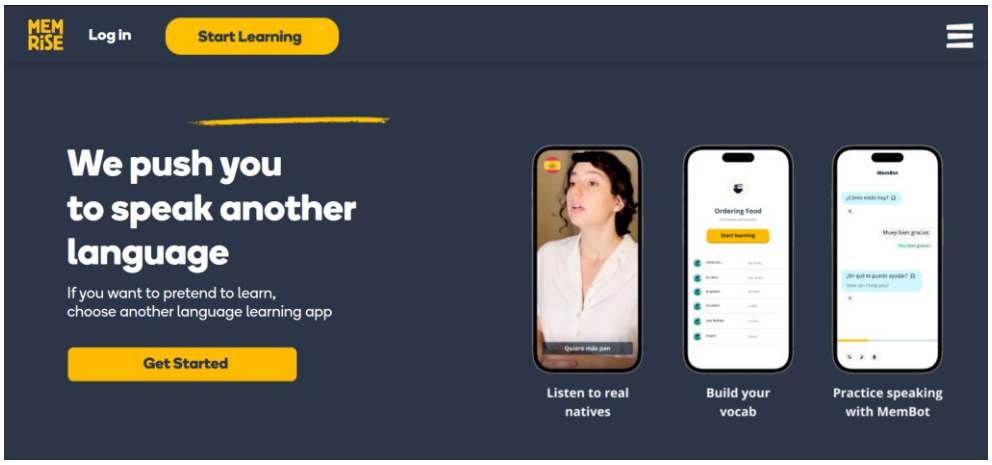


Рисунок 1.4 – Головна сторінка Memrise за джерелом [5]

Duolingo та Memrise є популярними додатками для вивчення іноземних мов, проте вони мають недоліки. Вони можуть стати монотонним через постійне повторення матеріалів та вправ, особливо на ранніх стадіях вивчення.

Зазвичай фокус зосереджується на вивченні словникового запасу та фраз, пропускаючи граматичні аспекти мови, що може призвести до недостатньої розуміння структури мови. Слова та фрази часто вивчаються окремо від контексту, що може ускладнювати їх розуміння та застосування в реальних ситуаціях.

Методи Duolingo та Memrise можуть бути недостатньо адаптовані до індивідуальних потреб та стилів вивчення користувачів, що може знижувати ефективність навчання.

Хоча LibreLingo акцентується на відкритості та співпраці, що робить його привабливим вибором для тих, хто шукає вільне та доступне середовище для вивчення мови, він має недолік у вигляді підтримки навчання лише іспанської мови для носіїв англійської мови.

Що стосується UneeBee, він має по досить маленьку спільноту, що сповільнює процес створення новий навчальних курсів, тому він поки що має не достатню кількість інформації для вивчення.

1.2 Аналіз методів вивчення іноземних мов

Проблеми, які потребують уваги при розробці програмного забезпечення для вивчення іноземних мов. Недостатній рівень інтерактивності може призвести до зниження зацікавленості користувачів та їхньої мотивації до навчання. А відсутність асоціацій з образами або ситуаціями також може ускладнити процес запам'ятовування та засвоєння нового матеріалу.

Для розв'язання цих проблем можна використовувати різноманітні підходи. Використання мультимедійних ресурсів способом додавання зображень, відео та аудіофайлів до навчального контенту може сприяти кращому засвоєнню матеріалу та створювати асоціації з реальними ситуаціями.

Інтерактивні вправи та ігри, а саме включення різноманітних інтерактивних елементів, таких як ігри, кросворди, вікторини тощо, може зробити процес навчання цікавішим.

Застосування методів асоціативного навчання, впровадження технік, які сприяють створенню асоціацій між словами або виразами та конкретними образами або ситуаціями, може полегшити процес запам'ятовування та зрозуміння матеріалу.

Залучення користувачів до активної участі способом надання можливостей для користувачів створювати власний контент або ділитися своїми асоціаціями та власними прикладами може сприяти покращенню їхнього розуміння та запам'ятовування матеріалу.

Розв'язання цих проблем може зробити програмне забезпечення для вивчення іноземних мов більш ефективним та привабливим для користувачів.

1.3 Постановка задач дослідження

При постановці задач дослідження для програмного забезпечення інтерактивного навчального комплексу з вивчення іноземних мов з використанням вебтехнологій потрібно врахувати деякі аспекти.

Аналіз потреб користувачів: дослідження потреб та очікувань цільової аудиторії щодо навчального процесу, їхніх уподобань щодо методів навчання та вимог до програмного забезпечення.

Ефективність методів навчання: оцінка ефективності різних методів навчання і їх вплив на результативність процесу вивчення мови, включаючи використання вебтехнологій та інтерактивних елементів.

Вивчення інтерфейсу користувача: дослідження ергономіки та зручності інтерфейсу користувача для забезпечення найкращого досвіду взаємодії з програмним забезпеченням.

Адаптивність та персоналізація: вивчення можливостей для адаптації програмного забезпечення до потреб кожного користувача та створення індивідуалізованих навчальних траєкторій.

Вивчення впливу інтерактивних елементів та асоціативних методів: оцінка впливу використання різних інтерактивних елементів та методів асоціативного навчання на ефективність навчання та засвоєння матеріалу.

Розробка та тестування нових функцій: постановка завдань щодо розробки та тестування нових функцій програмного забезпечення для поліпшення навчального процесу та користувацького досвіду.

Загальна задача дослідження - це створення програмного забезпечення, яке буде ефективно допомагати користувачам вивчати іноземні мови, використовуючи інтерактивні методи та вебтехнології.

1.4 Висновки за розділом 1

На основі постановки задач дослідження можна зробити наступні висновки:

– потреба в індивідуальному підході до користувачів, дослідження потреб користувачів підтверджує, що індивідуальні підходи до навчання, які враховують їхні уподобання та стилі вивчення, є важливими для успішного навчання мови;

– важливість дослідження ефективності різних методів навчання підкреслює значення використання науково обґрунтованих підходів та використання інтерактивних елементів для забезпечення ефективного вивчення мови;

– необхідність дослідження ергономіки та зручності інтерфейсу користувача показує, що простота та доступність інтерфейсу є ключовими факторами для покращення зручності користування;

– розгляд потреб адаптивності та персоналізації підтверджує, що програмне забезпечення повинне бути здатне адаптуватися до потреб кожного користувача для досягнення найкращих результатів;

– дослідження ефективності інтерактивних елементів та асоціативних методів навчання свідчить про їхню значущість для поліпшення процесу навчання та засвоєння матеріалу.

В цілому, висновки підкреслюють важливість розробки програмного забезпечення для навчання іноземних мов з використанням вебтехнологій, яке буде забезпечувати індивідуальні підходи, ефективні методи навчання та зручний інтерфейс користувача для досягнення успіху у вивченні мови.

2 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ ТА РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мов програмування та фреймворків

Вибір мов програмування та фреймворків для розробки програмного забезпечення для інтерактивного навчального комплексу з вивчення іноземних мов та використанням вебтехнологій є ключовим етапом, оскільки від цього залежить якість, швидкість та ефективність розробки.

Фреймворк (англ. Framework) – це набір інструментів та бібліотек, який призначений для спрощення розробки додатків, оскільки він надає готові рішення для поширених завдань, що у свою чергу дозволяє розробникам зосередитися на створенні логіки програми, а не витрачати час на написання коду для вирішення спільних завдань [1].

2.1.1 Мова програмування Python та зв’язані програмні інструменти

Програмний застосунок було розроблено мовою програмування Python, яка є потужною та високорівневою мовою з широким спектром застосувань. Python створений Гвідо ван Россумом у 1991 році, відомий своєю простотою використання та об'єктно-орієнтованим підходом до програмування, а також має динамічну типізацію. Управління пам'яттю у Python автоматизовано завдяки Garbage Collector, який ефективно керує звільненням пам'яті [6].

Порівняно з іншими мовами програмування, Python відрізняється тим, що не повертає всю пам'ять оперативній системі відразу ж після видалення об'єктів. Замість цього, він використовує спеціальний менеджер пам'яті, який

ефективно управляє виділенням блоків пам'яті для зберігання малих об'єктів [7].

Python також відзначається вбудованим набором бібліотек, що є однією з ключових переваг цієї мови. Ці бібліотеки включають різноманітні вбудовані модулі, такі як os, sys, datetime, які дозволяють виконувати різноманітні операції з файлами, операції з операційною системою та роботу з датою та часом. Крім того, Python має сторонні бібліотеки та фреймворки, такі як Flask, який використовується для побудови серверної частини цього програмного застосунку, що робить мову Python привабливою для розробки програмного забезпечення [8].

Фреймворк Flask для розробки серверної частини вебзастосунка.

Flask – це легкий вебфреймворк для створення серверної частини вебзастосунка мовою програмування Python. Він був створений у 2010 році Арміном Ронахером (англ. Armin Ronacher) і є відкритим програмним забезпеченням [6].

Flask дозволяє швидко створювати вебзастосунки та вебсервіси, використовуючи просте API на базі WSGI. Він має дружній та простий у використанні синтаксис, який дозволяє швидко створювати вебзастосунки з мінімальними зусиллями та має модульну архітектуру, що дозволяє легко розширювати його функціональність за допомогою різних розширень та плагінів [9].

Flask дає розробникам велику гнучкість у виборі структури проєкту та використання різних бібліотек та інструментів, має компактний розмір та мінімалістичний набір функцій, що дозволяє зосередитися на основних аспектах розробки та чудову документацію, яка допомагає швидко засвоїти основи та деталі роботи з фреймворком [8].

Фреймворк Flask-RESTful для реалізації REST API.

Фреймворк Flask-RESTful - це розширення для Flask, яке дозволяє легко створювати RESTful API (API з використанням архітектурного стилю REST) для вебдодатків мовою програмування Python. Він дозволяє організувати

програмні застосунки у вигляді ресурсів, які відповідають за певні об'єкти чи концепції, і використовувати HTTP методи (GET, POST, PUT, DELETE тощо) для взаємодії з ними [10].

Flask-RESTful автоматично виконує перетворення об'єктів Python в JSON та навпаки, що спрощує роботу з даними на рівні API. Він має широкий вибір розширень та додаткових функцій, які дозволяють додавати до вашого API різноманітні можливості, такі як аутентифікація, авторизація, логування тощо.

Flask-RESTful має зрозумілу та просту у використанні синтаксис, що дозволяє швидко створювати та підтримувати RESTful API без зайвого напруження та надає зручні засоби для тестування вашого API, що допомагає забезпечити якість та надійність додатка.

Загалом, Flask-RESTful є потужним та зручним інструментом для розробки RESTful API з використанням мови програмування Python та фреймворку Flask. Його простота використання та широкий функціонал роблять його відмінним вибором для будь-яких проєктів, які вимагають створення вебсервісів або інтерфейсів для взаємодії з додатками та сервісами [10].

Вибір SQLAlchemy як ORM-системи програмного застосунку.

SQLAlchemy - це ORM-система створена Майклом Байером у 2006 році, що підтримує різні СУБД, включаючи SQLite, PostgreSQL, MySQL та інші, що надає можливість працювати з різними типами баз даних без необхідності великих змін у коді додатка. Він дозволяє використовувати об'єктно-орієнтований підхід до роботи з базою даних. Замість написання складних SQL-запитів, ви можете працювати з об'єктами Python, що спрощує розробку та підтримку коду [11].

SQLAlchemy надає: високорівневий API для створення, читання, оновлення та видалення записів у базі даних, що дозволяє розробникам швидко та ефективно працювати з даними без прямого використання SQL, можливість використання транзакцій для групування змін у базі даних, а також

забезпечує захист від SQL-ін'єкцій та інших типів атак на безпеку даних та багато функцій для створення складних SQL-запитів, включаючи з'єднання, фільтрацію, сортування та групування даних [12].

Загалом, вибір SQLAlchemy як ORM-системи для програмного додатка дозволяє зробити роботу з базою даних більш ефективною, зручною та безпечною. Його потужність та гнучкість роблять його відмінним вибором для будь-якого проєкту, який вимагає роботи з базою даних у Python.

2.1.2 Мова програмування JavaScript та его середовище виконання Node.js

JavaScript є однією з найпопулярніших мов програмування, що створена Брендом Айком у 1961 році, яка застосовується як на клієнтській стороні (у браузерах), так і на серверній стороні (за допомогою Node.js), що дозволяє розробникам використовувати одну мову для написання як клієнтської, так і серверної частини своїх додатків. Він має велику кількість бібліотек та фреймворків для розробки різноманітних додатків та отримує значну перевагу завдяки своєму асинхронної моделі виконання, що дозволяє створювати швидкі та ефективні серверні застосунки [13].

Node.js – середовище виконання JavaScript створено Раян Далом у 2009 році. Він дозволяє розробникам легко масштабувати свої застосунки завдяки можливості обробки великої кількості одночасних з'єднань. JavaScript разом з Node.js дозволяє швидко створювати прототипи та розвивати застосунки завдяки простому синтаксису та багатим функціоналом.

Загалом, вибір JavaScript та Node.js як мови програмування та її середовища виконання дозволяє розробникам створювати широкий спектр додатків для різних платформ та пристроїв, забезпечуючи при цьому ефективність, масштабованість та швидкість розробки.

Фреймворк Vue для розробки клієнтської частини вебзастосунка. Фреймворк Vue.js – це прогресивний JavaScript-фреймворк для розробки клієнтської частини вебдодатків створений Еваном Ю у 2014 році. Він має простий та зрозумілий синтаксис, що робить його легким для вивчення та використання навіть для початківців у веброзробці та використовує реактивну систему, яка автоматично відстежувати зміни у стані додатка та автоматично оновлює відповідний інтерфейс користувача, що спрощує роботу з динамічними даними [14].

Vue.js побудований на базі компонентної архітектури, що дозволяє розбивати інтерфейс на невеликі компоненти, що полегшує управління складністю додатка та забезпечує швидку розробку, що надає розробникам гнучкість у виборі способу реалізації функціональності та легко розширюється за допомогою плагінів та розширень [14].

Загалом, Vue.js є потужним та гнучким інструментом для розробки клієнтської частини вебдодатків, який дозволяє розробникам швидко та ефективно створювати сучасні та інтерактивні інтерфейси користувача.

Типізація JavaScript за допомогою TypeScript. TypeScript – надбудова для мови програмування JavaScript, створена компанією Microsoft у 2012 році. Він дозволяє визначити типи для змінних, функцій та інших елементів програми, що дозволяє виявляти помилки типізації під час компіляції, а не під час виконання програми. Це полегшує виявлення та виправлення помилок під час розробки та підтримує сучасні функціональні можливості JavaScript, такі як стрілкові функції, асинхронні функції та інші, що робить його ідеальним інструментом для розробки сучасних вебдодатків[15].

Використання типів дозволяє зробити код більш зрозумілим та читабельним, оскільки типи вказують на очікуваний формат даних та взаємодію об'єктів. За допомогою статичній типізації TypeScript допомагає зменшити кількість помилок у коді, що виникають через неправильне використання типів даних та активно підтримує останні версії ECMAScript

стандартів, що дозволяє розробникам використовувати нові функціональні можливості JavaScript без обмежень. Він має велику та активну спільноту розробників, а також багату екосистему розширень та інструментів для підтримки розробки [15].

Загалом, використання TypeScript дозволяє підвищити якість, надійність та читабельність JavaScript коду, що робить його ідеальним вибором для серйозних та великих проєктів.

2.2 Редактор коду VSCodium

VSCodium – це вільний та відкритий редактор коду, що має велику кількість розширень та плагінів, які дозволяють розширити його функціональність та адаптувати до конкретних потреб розробника. Він надає розробникам потужні інструменти для редагування коду, включаючи підсвічування синтаксису, автодоповнення, вбудовані інструменти для роботи з Git та інші.

VSCodium має ряд інтеграцій з різними сервісами розробки, такими як GitLab, Docker, Kubernetes та інші, що полегшує роботу з ними безпосередньо з редактора.

Велика та активна спільнота користувачів та розробників, що активно підтримує та розвиває редактор коду, забезпечує стабільність та актуальність його функціональності.

Загалом, VSCodium – це потужний та універсальний редактор коду з великою кількістю функцій та можливостей, який дозволяє розробникам працювати ефективно та продуктивно над своїми проєктами.

2.3 Висновки за розділом 2

Обрані мови програмування, фреймворки та інструменти є добре збалансованими та оптимальними для розробки інтерактивного навчального комплексу з вивчення іноземних мов з використанням вебтехнологій.

Python є відмінним вибором для розробки серверної частини додатка, оскільки має елегантний синтаксис та легко масштабується. Крім того, наявність багатофункціональних бібліотек та фреймворків, таких як Flask, дозволяє розробникам швидко створювати вебзастосунки.

JavaScript разом з Node.js надає можливість створювати динамічні та ефективні застосунки. Його асинхронна модель дозволяє ефективно керувати великими навантаженнями та взаємодіяти з клієнтами в реальному часі.

Vue.js - це зручний та потужний фреймворк для розробки інтерфейсів користувача. Його компонентна архітектура спрощує розробку та підтримку клієнтської частини, а реактивний підхід дозволяє створювати динамічні інтерфейси.

SQLAlchemy – це потужний ORM інструмент, який дозволяє ефективно взаємодіяти з реляційними базами даних у Python. Використання ORM спрощує роботу з даними та дозволяє уникнути написання складних SQL-запитів.

VSCodium – це зручний та потужний редактор коду з великою кількістю функцій та розширень. Він надає зручний інтерфейс для редагування та організації коду, а також інтегрується з іншими інструментами розробки.

Загалом, обрана комбінація мов програмування, фреймворків та інструментів забезпечує швидку, ефективну та масштабовану розробку інтерактивного навчального комплексу з вивчення іноземних мов та використанням вебтехнологій.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Побудова архітектури програмного продукту

В першу чергу необхідно визначити основну архітектуру вебдодатка [16]. Для початку можемо визначити шар візуалізації даних (Presentation layer), який буде виконувати роль візуалізації потрібних даних на вебсторінці. Він є одним з основних, оскільки дозволяє користувачу швидко та ефективно засвоювати надану інформацію та організовувати взаємозв’язок користувача з програмною частиною [17].

Наступним шаром визначимо рівень бізнес-логіки (Business logic layer). Основною її функцією є перетворення введеної користувачем інформації в дані які є зрозумілими для системи. А, основною задачею своєю чергою буде валідація даних, методи та алгоритми виконання бізнес-правил (він англ. Business - Справа), обробка бізнес-процесів, інтеграція з іншими системами та сервісами. Наприклад, розрахунок податків, обробка замовлень, реєстрація та авторизація користувачів та інше [17].

В якості останнього шару об'єднаємо шар бази даних (Database layer) та шар доступу до даних (Data Access layer), тому що вони відповідають за збереження та надання інформації. Їх функцією є зберігання та керування даними, надання доступу до бази даних, створення та закриття транзакцій, виконання запитів до бази даних та відповідає за збереження даних додатка у впорядкованому вигляді. В цілому виконання CRUD операцій (Create, Read, Update, Delete) створення, збереження, оновлення та видалення даних, забезпечення цілісності та безпеки даних, оптимізація запитів для швидкого доступу [17].

На рис. 3.1 зображено зразкову архітектуру вебдодатка.

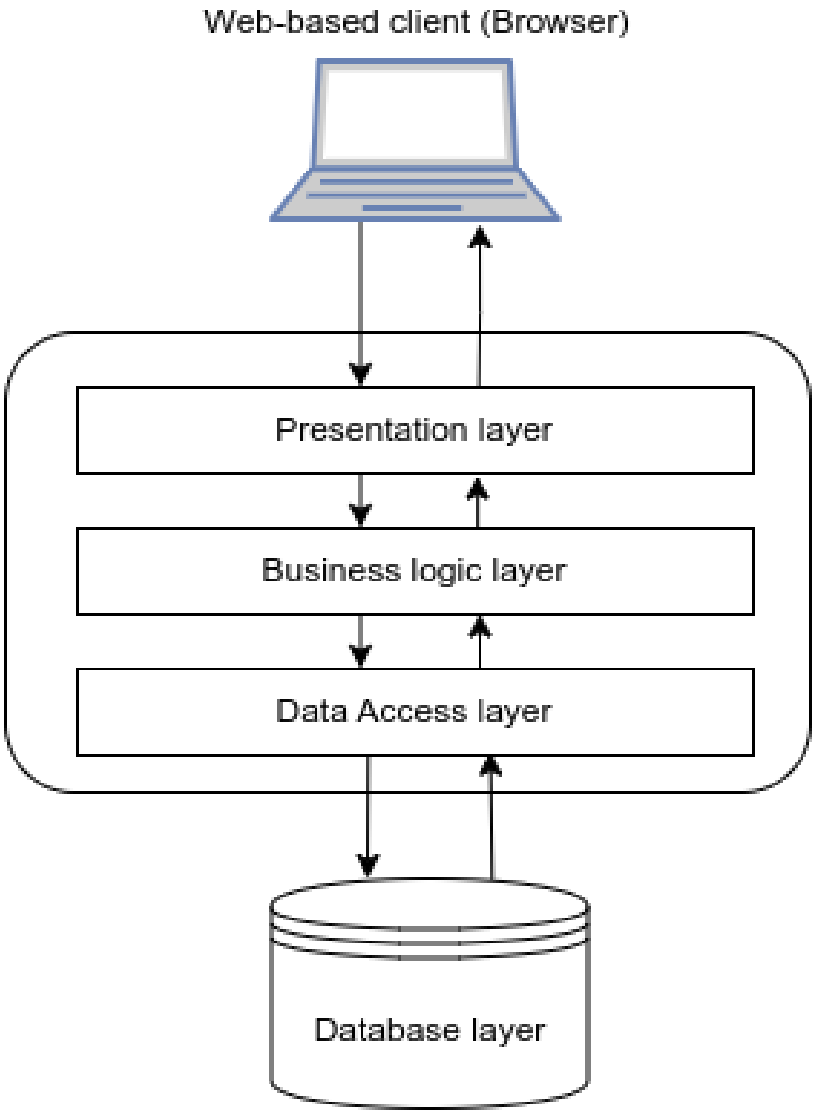


Рисунок 3.1 – Архітектура вебдодатка

Коли користувач через браузер встановлений на його комп'ютері працює з додатком, при спрацюванні деяких сценаріїв запити відправляються на сервер. Кожна дія викликає відповідну дію на рівні візуалізації даних.

Шар бізнес-логіки обробляє всі перетворення і маніпуляції даними в додатку. В цьому шарі визначається, яким чином і в якій таблиці повинні здійснюватися зміни, що запросив користувач. Після цього проводиться підключення до бази даних для запису змін. Якщо дія відбувається успішно, на вебсторінці повинен виводитися очікуваний результат. В разі помилки, шар візуалізації визначає її характер, та користувач отримує відповідне сповіщення.

Ця архітектура виконує різні функції різного рівня відповідності. Це полегшує роботу з системою, забезпечує надійність, підвищує ефективність та задає вимоги до концепцій її роботи. Шар візуалізації дає змогу користувачу взаємодіяти з додатком через інтуїтивно зрозумілий інтерфейс. Шар бази даних зберігає дані в структурованому вигляді та забезпечує їх цілісність та неушкодженість.

3.2 Розробка алгоритмів роботи додатка

Для розробки вебдодатка з вивчення іноземних мов, були розроблені алгоритми роботи та взаємодії з користувачем. До нього входять алгоритм, що перетворює число на його словесний еквівалент (рис. 3.2), алгоритм пошуку множини (рис 3.5) та алгоритми побудови речення англійською мовою (рис. 3.3, 3.4).

Алгоритм перетворення числа на його словесний еквівалент дозволяє користувачу краще зрозуміти побудову та промову чисел в англійській мові для його використання в реченні, надає можливості тренування мовлення.

Алгоритм побудови речення з "able to" демонструє як виглядає речення з дієсловом “уміти” та дозволяє користувачу краще зрозуміти тему.

Алгоритм побудови речення з "know how" показує як виглядає речення з дієсловом “знати як” та також дозволяє користувачу краще зрозуміти цю тему.

Алгоритм пошуку множини описує правила утворення множини для різних випадків англійських іменників, враховуючи різні винятки та спеціальні випадки.

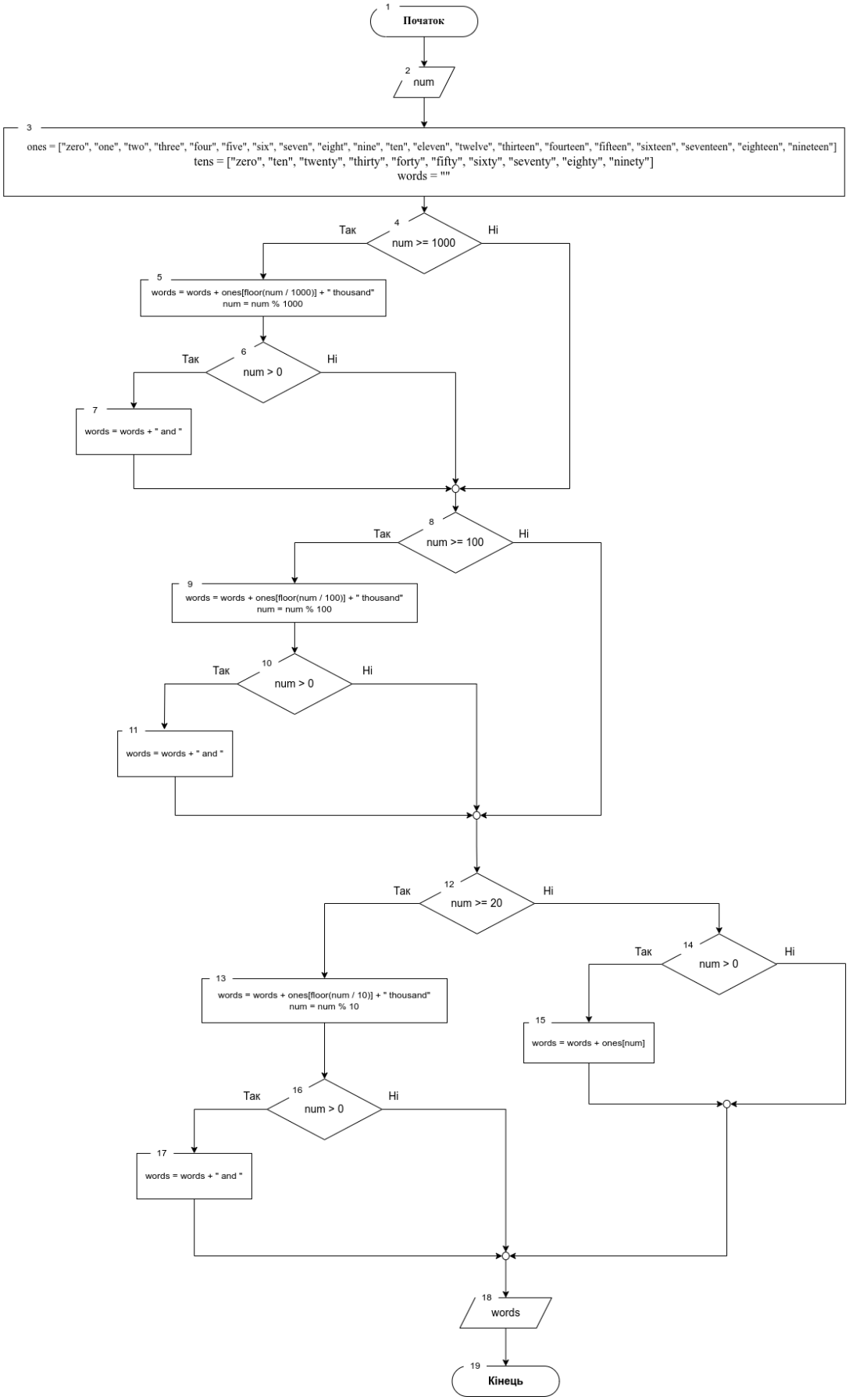


Рисунок 3.2 – Алгоритм перетворення числа на його словесний еквівалент

Для кращого розуміння роботи алгоритму, роздивимось його покроково:

1) початок виконання алгоритму;

2) введення числа num, яке необхідно перетворити на текстовий формат;

3) ініціалізація масивів ones і tens, які містять текстові представлення чисел від 0 до 19 та десятків відповідно. ініціалізація змінної words як порожнього рядка;

4) перевірка, чи більше або дорівнює num 1000;

5) додавання до words текстового представлення тисячної частини числа. зменшення num на цю тисячну частину (num = num % 1000);

6) перевірка, чи більше ніж 0 залишкове значення num;

7) додавання до words слова "and" у випадку, якщо залишкове значення num більше ніж 0;

8) перевірка, чи більше або дорівнює num 100;

9) додавання до words текстового представлення сотенної частини числа. зменшення num на цю сотенну частину (num = num % 100);

10) перевірка, чи більше ніж 0 залишкове значення num;

11) додавання до words слова "and" у випадку, якщо залишкове значення num більше ніж 0;

12) перевірка, чи більше або дорівнює num 20;

13) додавання до words текстового представлення десяткової частини числа. зменшення num на цю десяткову частину (num = num % 10);

14) перевірка, чи більше ніж 0 залишкове значення num;

15) додавання до words текстового представлення одиничної частини числа;

16) перевірка, чи більше ніж 0 залишкове значення num;

17) додавання до words слова "and" у випадку, якщо залишкове значення num більше ніж 0;

18) виведення змінної words, яка містить повний текстовий еквівалент числа;

19) завершення виконання алгоритму.

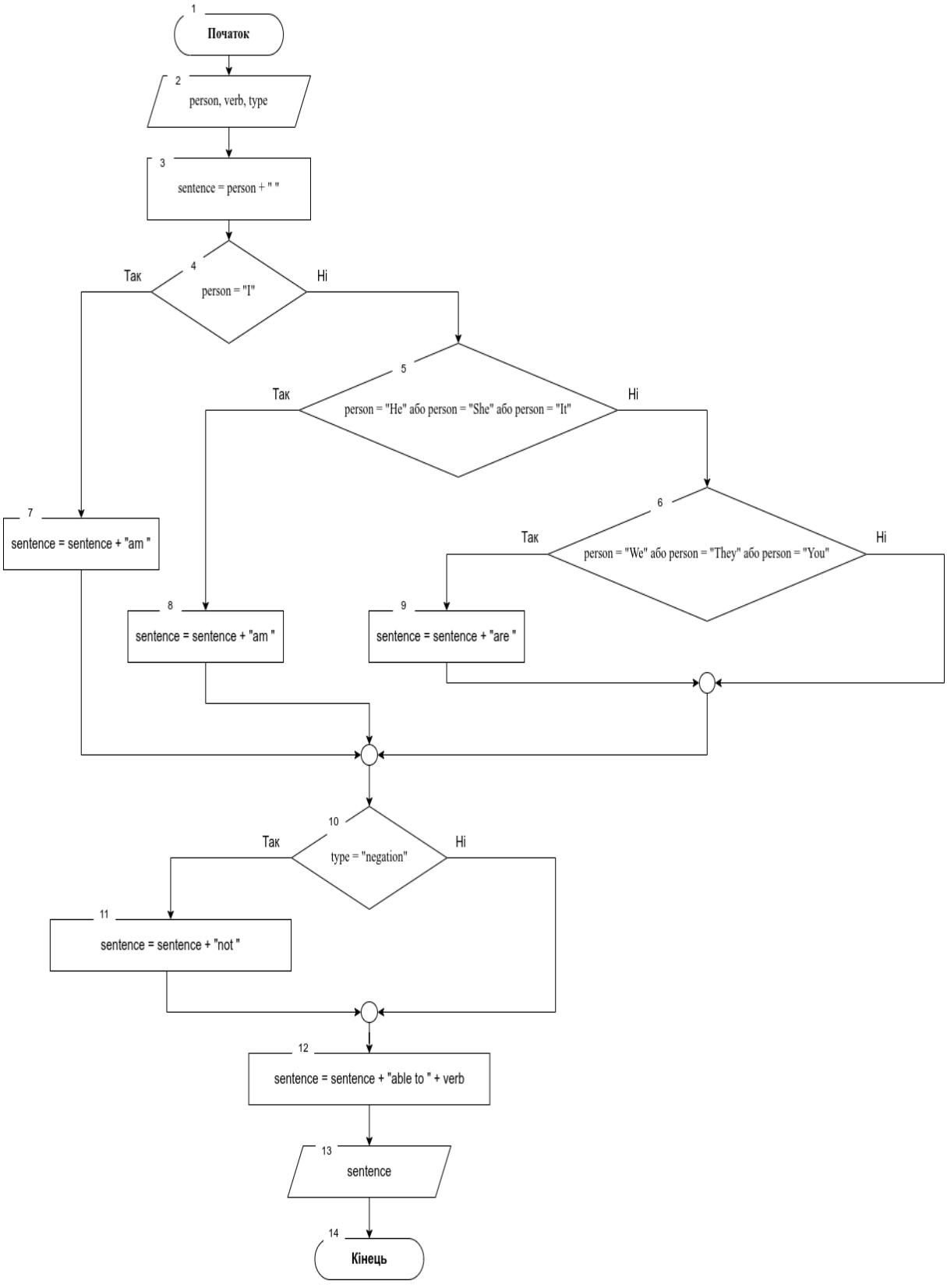


Рисунок 3.3 – Блок-схема алгоритм побудови речення з "able to"

Для кращого розуміння роботи алгоритму, роздивимось його покроково:

1) початок виконання алгоритму;

2) введення трьох змінних: person (особа), verb (дієслово), type (тип речення);

3) ініціалізація змінної sentence та додавання до неї значення person;

4) перевірка, чи дорівнює person значенню "I";

5) перевірка, чи дорівнює person значенням "He", "She" або "It";

6) перевірка, чи дорівнює person значенням "We", "They" або "You";

7) додавання до sentence слова "am", якщо person = "I";

8) додавання до sentence слова "am", у випадку успішної перевірки в блоці 4 (повторне додавання);

9) додавання до sentence слова "are", якщо person = "We", "They" або "You";

10) перевірка, чи дорівнює type значенню "negation" (заперечення);

11) додавання до sentence слова "not", якщо type = "negation";

12) додавання до sentence фрази "able to" та значення змінної verb;

13) завершення формування змінної sentence;

14) завершення виконання алгоритму, виведення готового речення;

Для кращого розуміння роботи алгоритму (рис. 3.4), роздивимось його покроково:

1) початок виконання алгоритму;

2) введення трьох змінних: person (особа), verb (дієслово), type (тип речення);

3) ініціалізація змінної sentence та додавання до неї значення person;

4) перевірка, чи дорівнює type значенню "negation" (заперечення);

5) додавання до sentence фрази "does not" у випадку, якщо type = "negation";

6) перевірка, чи дорівнює person значенню "I";

7) перевірка, чи дорівнює person значенням "He", "She" або "It";

8) перевірка, чи дорівнює person значенням "We", "They" або "You";

9) додавання до sentence слова "know", якщо person = "I";

10) додавання до sentence слова "knows", якщо person = "He", "She" або "It";

11) додавання до sentence слова "know", якщо person = "We", "They" або "You";

12) додавання до sentence фрази "how to" та значення змінної verb;

13) завершення формування змінної sentence;

14) завершення виконання алгоритму, виведення готового речення.

15) додавання до sentence слова "know", якщо person = "We", "They" або "You";

16) додавання до sentence фрази "how to" та значення змінної verb;

17) завершення формування змінної sentence;

18) завершення виконання алгоритму, виведення готового речення.

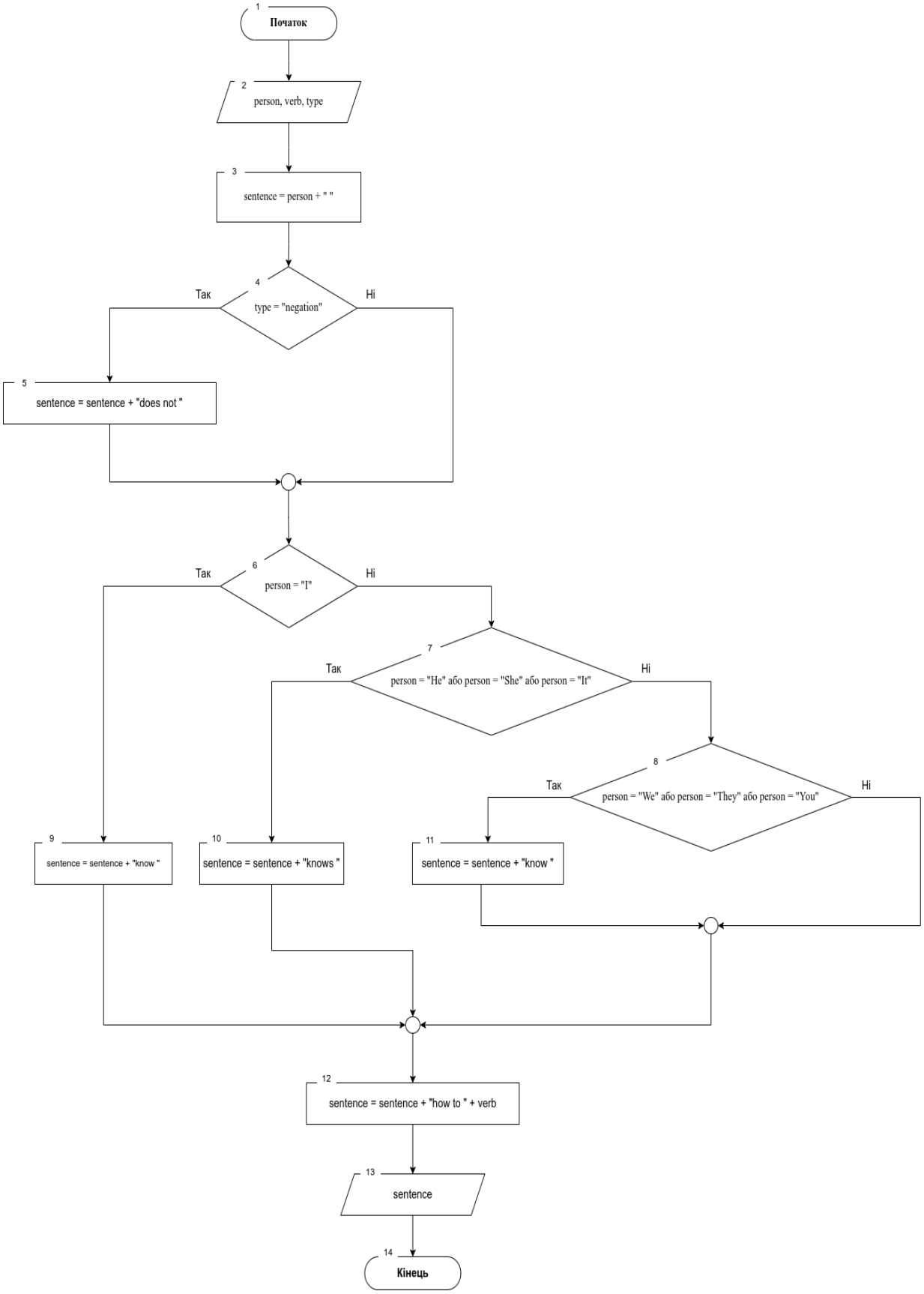


Рисунок 3.4 – Блок-схема алгоритм побудови речення з "know how"

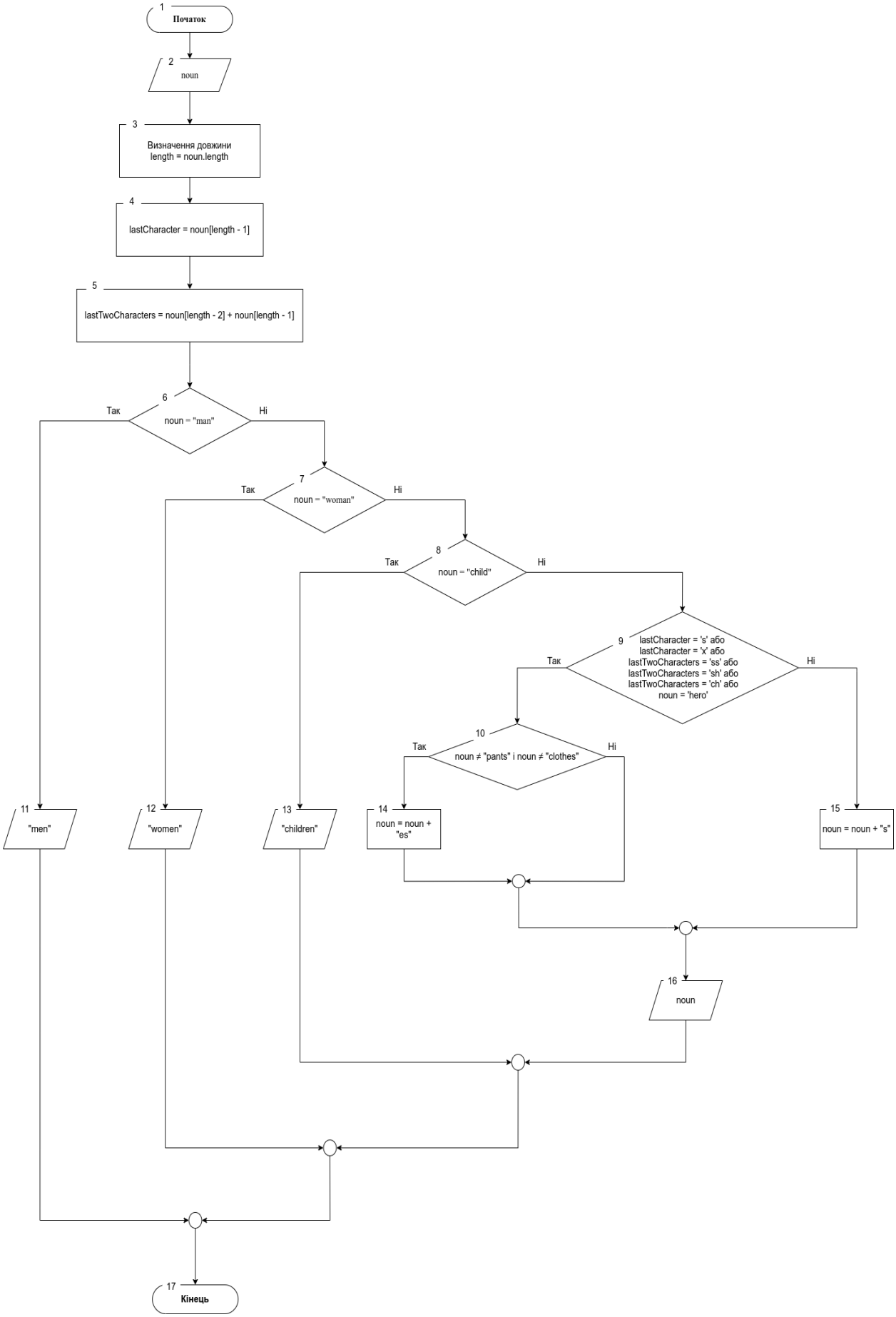


Рисунок 3.5 – Блок-схема алгоритм пошуку множини

Для кращого розуміння роботи алгоритму, роздивимось його покроково:

1) алгоритм починається;
2) вводиться іменник (noun);
3) визначається довжина рядка іменника: length = noun.length;
4) визначається останній символ іменника: lastCharacter = noun[length - 1];
5) визначаються останні два символи іменника: lastTwoCharacters = noun[length - 2] + noun[length - 1];
6) якщо іменник дорівнює "man", то перейти до кроку 11, інакше перейти до кроку 7;
7) якщо іменник дорівнює "woman", то: перейти до кроку 12, інакше перейти до кроку 8;
8) якщо іменник дорівнює "child", то перейти до кроку 13, інакше перейти до кроку 9;
9) перевірка закінчень і спеціальних випадків, якщо останній символ дорівнює "y" або "o" або "s" або останні два символи дорівнюють "sh" або "ch" або іменник дорівнює "hero", то перейти до кроку 10, інакше перейти до кроку 15;
10) якщо іменник дорівнює "pants" або "clothes", то перейти до кроку 16, інакше перейти до кроку 14;
11) вивести "men";
12) вивести "women";
13) вивести "children";
14) додати "es" до іменника;
15) додати "s" до іменника;
16) вивести перетворений іменник;
17) завершення алгоритму.

3.3 Розробка бази даних

Вибір бази даних та її проєктування є важливими етапами розробки додатка. У сучасному світі існує велика кількість баз даних як за реляційною системою управління даними, так і не реляційною. При цьому база даних визначає не тільки як дані зберігаються, а й спосіб доступу до них, тому її вибір є важливою частиною розробки.

Загалом, проєктування бази даних – процес створення її проєкту що призначений для підтримки функціонування економічного об'єкта та сприяє досягненню його цілей. Це трудомісткий процес, який вимагає спільних зусиль аналітиків, проєктувальників й користувачів [19].

В роботі з базами даних необхідно враховувати те, що вона має задовольняти ряд вимог.

- цілісність бази даних та всіх її записів;
- багаторазове використання даних;
- ефективний та швидкий пошук, оновлення даних й отримання інформації за запитами користувачів;
- захист даних від несанкціонованого доступу.

Також необхідно буде розуміти наступні поняття.

Таблиця – основна структура для зберігання даних у реляційній базі даних. Таблиця складається з рядків (записів) і стовпців (полів). Кожна таблиця має унікальну назву і представляє певну сутність, наприклад, користувачів або замовлення.

Запис – сукупність полів, що представляє одиничний екземпляр сутності в таблиці. Наприклад, один користувач у таблиці "Користувачі".

Поле – окремий стовпець у таблиці, що зберігає певний тип даних. Наприклад, поле "Ім'я" у таблиці "Користувачі".

Первинний ключ – унікальний ідентифікатор для кожного запису в таблиці. Первинний ключ не може містити null значень і забезпечує унікальність кожного запису.

Зовнішній ключ – поле або набір полів у одній таблиці, що посилається на первинний ключ в іншій таблиці. Зовнішні ключі використовуються для створення зв'язків між таблицями.

Індекс – структура даних, що покращує швидкість пошуку і доступу до даних у таблиці. Індекси створюються на основі одного або декількох полів.

SQL (Structured Query Language) – мова програмування, що використовується для управління і маніпулювання базами даних. SQL включає команди для створення, читання, оновлення і видалення даних.

Транзакція – сукупність однієї або декількох SQL команд, які виконуються як єдине ціле. Транзакції забезпечують атомарність, цілісність, ізольованість і довговічність.

Ізольованість – вплив однієї транзакції на інші мінімізується.

Довговічність – після завершення транзакції її результати зберігаються навіть у випадку збою системи.

Реплікація – процес копіювання і підтримки синхронізованих копій бази даних на декількох серверах для забезпечення доступності та відмовостійкості.

Сутність (англ. Entity) – реальний або уявний об'єкт, що має істотне значення для розглянутої предметної області, інформація про який підлягає зберіганню. Кожна сутність повинна мати унікальний ідентифікатор. Кожен екземпляр сутності повинен однозначно ідентифікуватися і відрізнятися від усіх інших екземплярів даного типу сутності [18].

Знання та вміння застосувати ці основні поняття допоможуть краще зрозуміти та ефективно працювати з базами даних.

Тепер розглянемо на рис. 3.6 діаграму класів, що показує сутності бази даних.

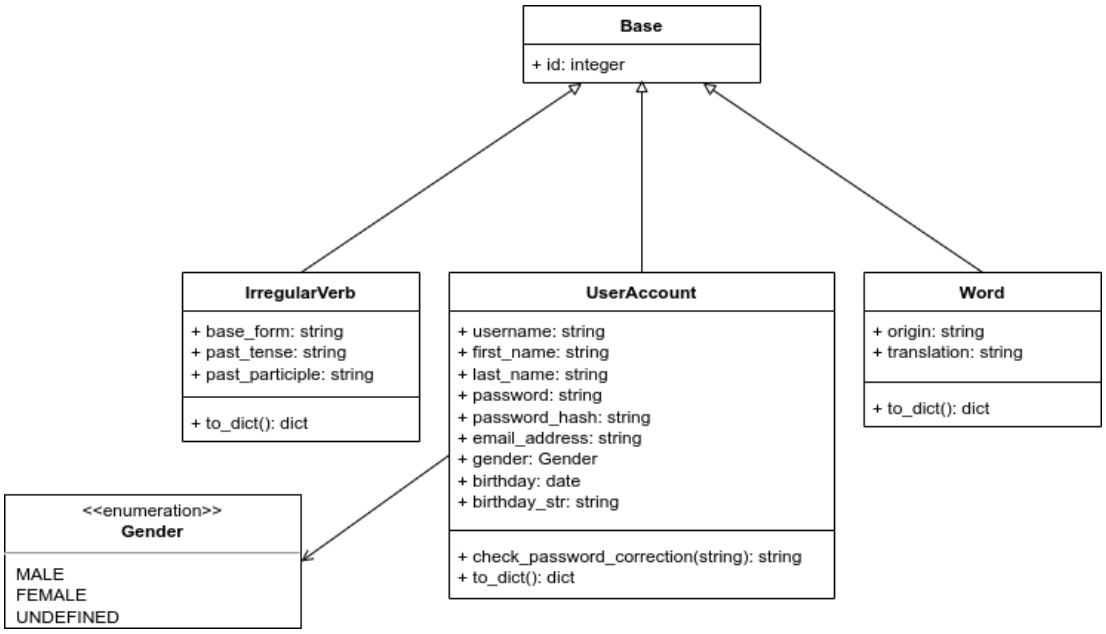


Рисунок 3.6 – Діаграма класів, що відображає сутності бази даних

Опишемо кожний клас та його властивості:

Клас Base (Це базовий клас, від якого наслідуються інші класи).

Атрибути:

– id: integer: унікальний ідентифікатор для кожного об'єкта.

Клас IrregularVerb (наслідується від Base).

Атрибути:

– base_form: string: базова форма неправильного дієслова.

– past_tense: string: форма минулого часу неправильного дієслова.

– past_participle: string: форма минулого причастя неправильного дієслова.

Методи:

– to_dict(): dict: перетворює об'єкт у словник.

Клас UserAccount (наслідується від Base).

Атрибути:

– username: string: ім'я користувача.

– first_name: string: ім'я користувача.

– last_name: string: прізвище користувача.

– password: string: пароль користувача.

– password_hash: string: хеш пароля користувача.

– email_address: string: електронна адреса користувача.

– gender: Gender: стать користувача, що використовується з переліком Gender.

– birthday: date: дата народження користувача.

– birthday_str: string: дата народження у форматі рядка.

Методи:

– check_password_correction(string): string: перевіряє правильність пароля.

– to_dict(): dict: перетворює об'єкт у словник.

Клас Word (наслідується від Base)

Атрибути:

– origin: string: Оригінальне слово.

– translation: string: Переклад слова.

Методи:

– to_dict(): dict: Перетворює об'єкт у словник.

Перелік (enumeration) Gender

Значення:

– MALE: Чоловіча стать.

– FEMALE: Жіноча стать.

– UNDEFINED: Невизначена стать.

Отже, дана діаграма показує, як різні класи пов'язані між собою через наслідування від базового класу Base. Клас IrregularVerb містить інформацію про неправильні дієслова, клас UserAccount містить інформацію про облікові записи користувачів, включаючи їх персональні дані та методи для перевірки пароля, а клас Word містить інформацію про слова і їх переклади. Перелік Gender визначає можливі значення для атрибута gender у класі UserAccount.

3.4 Розробка структури графічного інтерфейсу користувача

Графічний інтерфейс користувача (Graphical User Interface) потрібен програмному додатку для полегшення взаємодії користувача з програмою. Інтерфейс надає можливості формувати візуальні елементи, що дозволяють користувачу інтуїтивно керувати функціональністю та логікою програми без необхідності використовувати емулятор термінала та команд або мати знання внутрішньої структури програми [20].

До основних його задач можна віднести наступні пункти.

Задачі забезпечення зручності та інтуїтивності взаємодії. Таким чином GUI повинен залишатись зрозумілим та доступним у використанні навіть для недосвідчених у певній сфері користувачів, в свою чергу забезпечувати інтуїтивний доступ до функцій програми.

Задачі підвищення продуктивності користувачів. Безумовно, добре спроєктований графічний інтерфейс допомагає користувачам програми збільшити швидкість та ефективність виконання їх завдань, зменшуючи кількість кроків, необхідних для досягнення мети при цьому мінімізуючи можливість помилок.

GUI має бути таким, що забезпечує та підтримує узгодженість у використанні елементів інтерфейсу, кольорів, шрифтів і стилів у всій системі. Це допомагає користувачам швидше зрозуміти програму і зменшує навчальне навантаження.

Надання GUI візуального зворотного зв'язку на дії користувача, наприклад, підтверджуючи, що завдання виконано успішно, або попереджаючи про помилки. Це може бути реалізовано за допомогою повідомлень, змін кольору, анімацій тощо.

Не менш важливим чинником є дозволяти користувачам налаштовувати інтерфейс відповідно до їхніх потреб і уподобань підтримуючи

персоналізацію. Це може включати зміну кольорових схем, розміру шрифту, розташування елементів тощо.

Також інтерфейс повинен дозволяти користувачам легко перемикатися між різними вкладками або вікнами, забезпечуючи ефективне управління багатозадачністю.

Нарешті, GUI має забезпечувати захист деякої конфіденційної інформації, підтримуючи такі механізми як аутентифікації та авторизації для безпечної взаємодії з даними [21].

Виконання цих задач допомагає забезпечити ефективність, зручність і задоволення для користувачів від взаємодії з програмним застосунком, що є критично важливим для комерційного успіху додатка. Структурні схеми інтерфейсу додатка зображені на рис. 3.7 та 3.8.

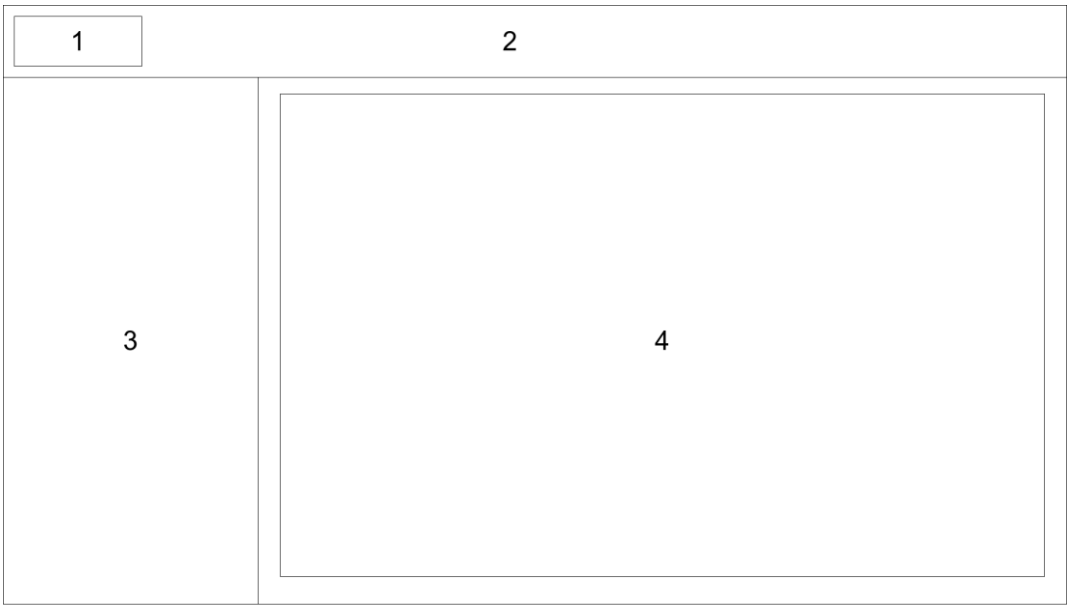


Рисунок 3.7 – Структурна схема інтерфейсу додатка

Основні елементи інтерфейсу додатка:

– логотип;
– заголовок додатка;
– бокове меню додатка;
– головний вміст додатка.

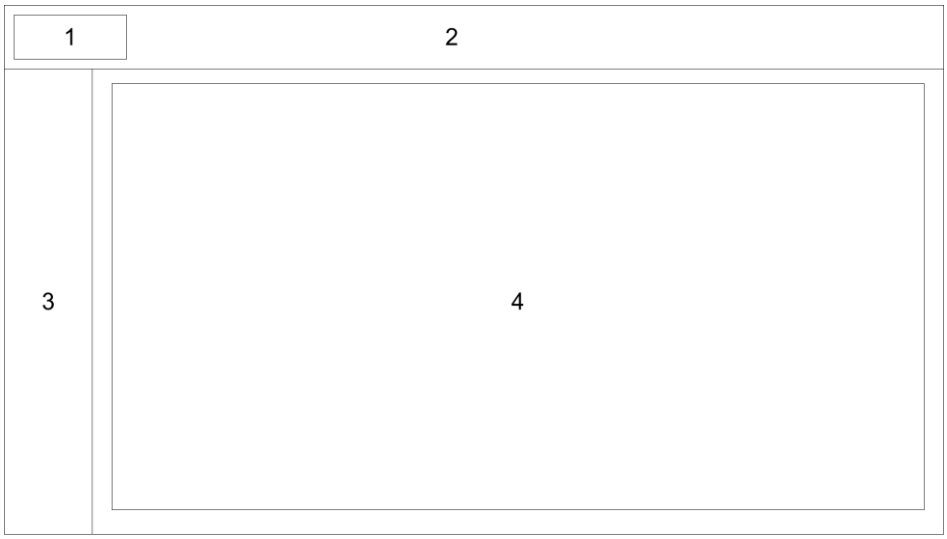


Рисунок 3.8 – Структурна схема інтерфейсу додатка зі згорнутим боковим меню додатка

В цьому випадку до основних елементів інтерфейсу, також належать: логотип, заголовок додатка, бокове меню додатка, головний вміст додатка. Але, основною відмінністю є те, що в даній схемі бокове меню додатка згорнуте, таким чином збільшується місце для основної частини, що спрощує сприймання навчального матеріалу. Розглянемо структурну схему заголовка додатка детальніше, вона зображена на рис. 3.9.

Рисунок 3.9 – Структурна схема заголовка додатка

Дана структурна схема демонструє заголовок додатка, який містить наступні елементи:

– кнопка згортання та розгортання бокового меню;
– логотип;
– розділи з додатковою інформацією;
– розділ перемикання станів бокового меню;
– кнопка для авторизації користувача;

– кнопка для переходу до реєстрації користувача.

3.5 Висновки за розділом 3

Отже, в третьому розділі було розглянуто архітектуру програмного застосунку, графічний інтерфейс користувача (GUI), моделі сутностей бази даних, деякі алгоритми роботи додатка. Таким чином, архітектура вебдодатка складається з трьох основних шарів.

Шар візуалізації даних (Presentation layer) - відповідає за відображення інформації на вебсторінці.

Рівень бізнес-логіки (Business logic layer) - перетворює введену користувачем інформацію в дані, зрозумілі для системи.

Шари бази даних (Database layer) та доступу до даних (Data Access layer) - відповідає за збереження та керування даними, забезпечення цілісності та безпеки даних, а також оптимізацію запитів для швидкого доступу до інформації.

Також це було відображено на діаграмі архітектури додатка.

Були розроблені структурні схеми інтерфейсу користувача що відображали які елементи де знаходяться та виділені основні вимоги до GUI, що включають:

– узгодженість у використанні елементів інтерфейсу;
– візуальний зворотний зв'язок;
– персоналізація інтерфейсу;
– ефективне управління багатозадачністю;
– безпека даних.

Розглянуті основні алгоритми та їх блок – схеми.

Алгоритм перетворення числа на його словесний еквівалент дозволяє користувачу краще зрозуміти побудову та промову чисел в англійській мові.

Алгоритм побудови речення з "able to" демонструє як виглядає речення з дієсловом “уміти” та дозволяє користувачу краще зрозуміти тему.

Алгоритм побудови речення з "know how" показує як виглядає речення з дієсловом “знати як”.

Алгоритм пошуку множини що описує правила утворення множини для різних випадків англійських іменників.

Нарешті, можна сказати що спроєктований графічний інтерфейс та чітко визначені алгоритми роботи додатка й архітектура програмного продукту є ключовими факторами успішності вебдодатка. Забезпечення узгодженості, зручності використання і захисту інформації сприяє підвищенню ефективності роботи користувачів та задоволення їх потреб, що як було сказано вище, є критично важливим для комерційного успіху додатка.

4 ТЕСТУВАННЯ РОЗРОБКИ

4.1 Аналіз методів тестування вебдодатка

Тестування означає фактичне тестування розробленого продукту на відповідність стандартам контролю якості. Це може включати використання, програмне тестування продукту, як ручне та й автоматизоване тестування або перевірку того, чи фактичні результати відповідають очікуваним результатам.

Цей процес виявляє проблеми в програмному продукті перед його запуском. Тестування забезпечує виявлення помилок, багів і дефектів програмного забезпечення. Тестування також орієнтоване на продукт [22].

Таким чином тестування програмного додатка є важливим етапом розробки ПЗ, що забезпечує виявлення і виправлення помилок до випуску продукту. До основних методів тестування можна віднести.

Модульне тестування (Unit Testing). Його мета перевірити окремі модулі або компоненти програми.

Інтеграційне тестування (Integration Testing). Це тестування з метою перевірки взаємодії між інтегрованими модулями. Зазвичай воно виконується після модульного тестування і використовує підходи "Big Bang", "Top-Down" або "Bottom-Up".

Системне тестування (System Testing). Воно ж в свою чергу комплексно перевіряє повний інтегрований програмний продукт і зазвичай виконується тестувальниками в середовищі, максимально наближеному до реального.

Функціональне тестування (Functional Testing). Перевіряє функціональність програми відповідно до специфікацій, може виконуватися як вручну, так і автоматично воно засноване на заздалегідь визначених специфікаціях і вимогах до системи.

Регресійне тестування (Regression Testing). Тестування яке має на увазі те, щоб нові зміни не порушували функціонал, зазвичай виконується після внесення якихось змін або виправлення помилок і часто автоматизується.

Альфа-тестування (Alpha Testing). Це внутрішнє тестування на ранніх етапах розробки яке може дозволити виявити критичні помилки на ранніх стадіях розробки проєкту.

Бета-тестування (Beta Testing). Тестування з залученням кінцевих користувачів в реальних умовах та дозволяє отримати реальні відгуки та зворотний зв'язок від користувачів. Також може виявити баги, які не були знайдені під час альфа-тестування.

Тестування чорного ящика (Black Box Testing). Тестування з метою перевірки функціональності програми, але без знання внутрішньої структури коду. Виконується тестувальниками з основною увагою на вхідних даних та очікуваних результатах.

Тестування білого ящика (White Box Testing). Воно являє собою перевірку внутрішньої структури та логіки коду і вимагає знання коду та внутрішньої логіки програми.

Тестування навантаженням (Load Testing). Тестування з метою перевірки роботи системи під різними навантаженнями. Створено для того, щоб виявити, як система реагує на збільшення кількості користувачів або оброблюваних даних і виконується для оцінки стабільності додатка.

Тестування безпеки (Security Testing). Це тестування з увагою на безпеку програми, що включає виявлення можливих вразливостей, тестування аутентифікації та авторизації, захист від атак. І виконується спеціалізованими командами безпеки або автоматизованими інструментами.

Загалом вище перераховані методи тестування забезпечують комплексний підхід до перевірки якості програмного забезпечення, що гарантує його стабільність, продуктивність та безпеку перед випуском у користування.

4.2 Тестування розробленого вебдодатка

Для тестування розробленого додатка було обрано метод чорного ящика (без знання внутрішньої структури коду), з основною увагою на вхідних даних та очікуваних результатах. Загалом були використані підходи ручного тестування, оскільки додаток має не великий обсяг та всі його функції можна перевірити вручну.

Розглянемо цей метод більш детально.

Тест-кейс №1 «Успішна реєстрація нового користувача»:

- відкрити додаток;
- натиснути «Register»;
- заповнити поля імені користувача, паролю та його повторення;
- натиснути кнопку «Register».

Очікуваним результатом є успішна реєстрація та авторизація користувача в додатку. Результат виконання тест-кейсу наведено на рис. 4.1.

Тест-кейс №2 «Пошук множини в англійській мові»:

- відкрити додаток;
- обрати пункт «Plural»;
- ввести іменник «book».

Очікуваним результатом є переведення слова в форму множини англійської мови з потрібним закінченням, а саме «books». Результат виконання тест-кейсу наведено на рис. 4.2.

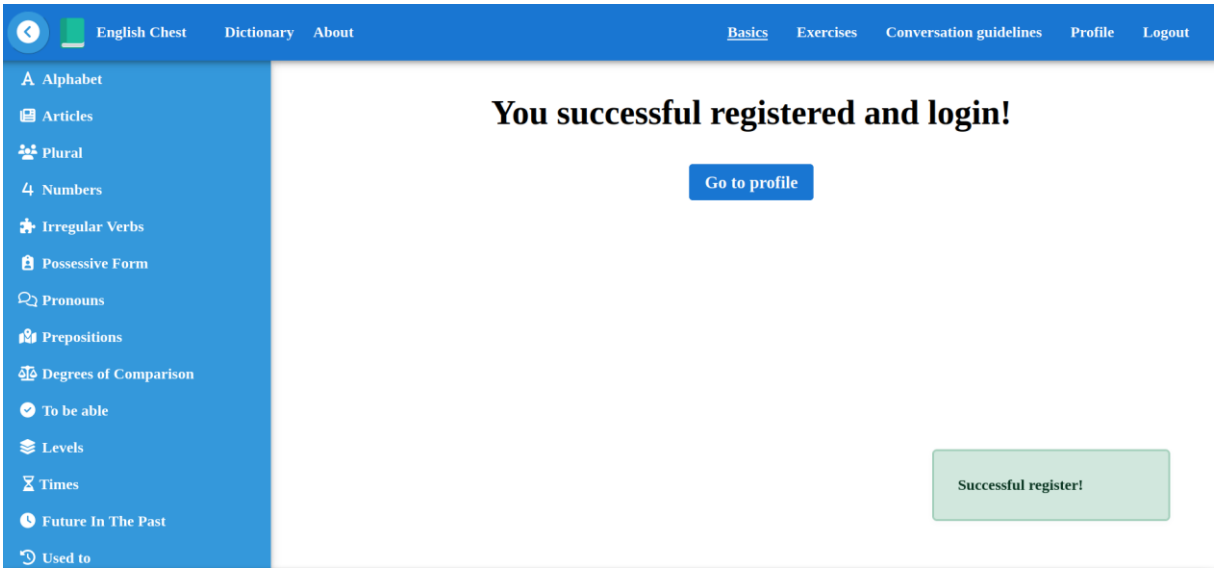


Рисунок 4.1 – Тест-кейс №1 «Успішна реєстрація нового користувача»

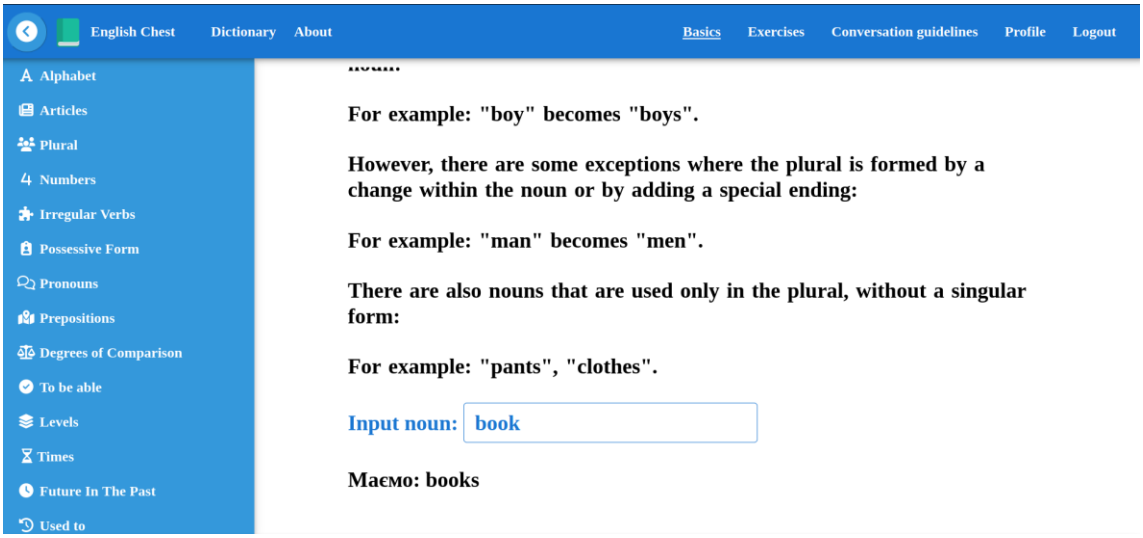


Рисунок 4.2 – Тест-кейс №2 «Пошук множини в англійській мові»

Тест-кейс №3 «Пошук словесного еквівалента числа в англійській мові»:

– відкрити додаток;
– обрати пункт «Number»;
– ввести число «42».

Очікуваним результатом є переведення введеного числа в його словесний еквівалент, а саме «forty-two». Результат виконання тест-кейсу наведено на рис. 4.3.

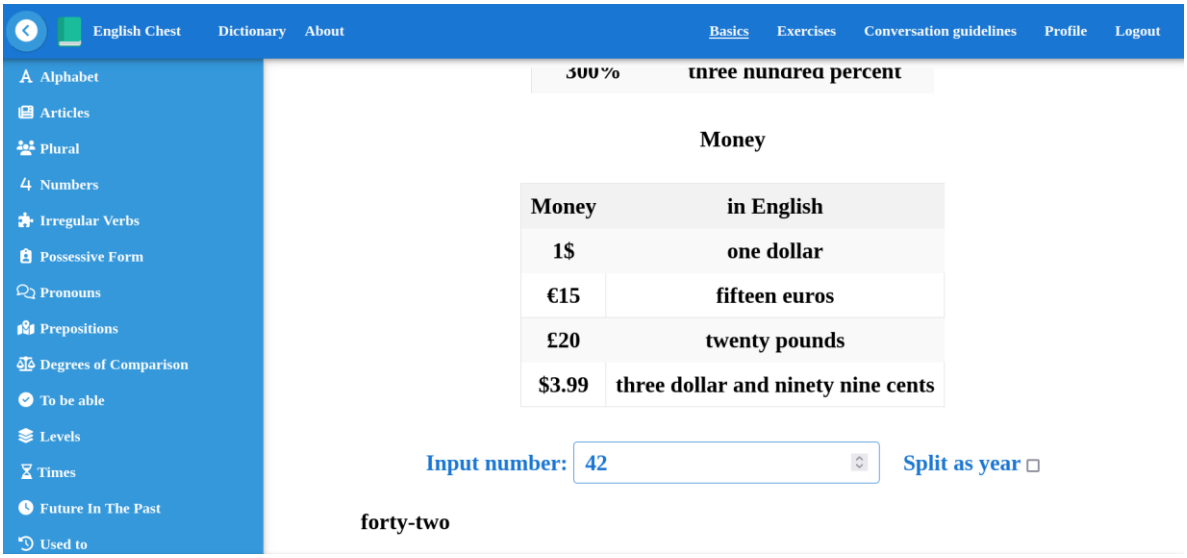


Рисунок 4.3 - Тест-кейс №3 «Пошук словесного еквівалента числа в англійській мові»

Тест-кейс №4 «Пошук ступенів порівняння прикметника»:

– відкрити додаток;
– обрати пункт «Degrees of Comparison»;
– ввести прикметник «chip».

Очікуваним результатом є переведення початкової форми прикметника в порівняльну форму та найвищу форму, а саме «chip - chiper - the chipest». Результат виконання тест-кейсу наведено на рис. 4.4.

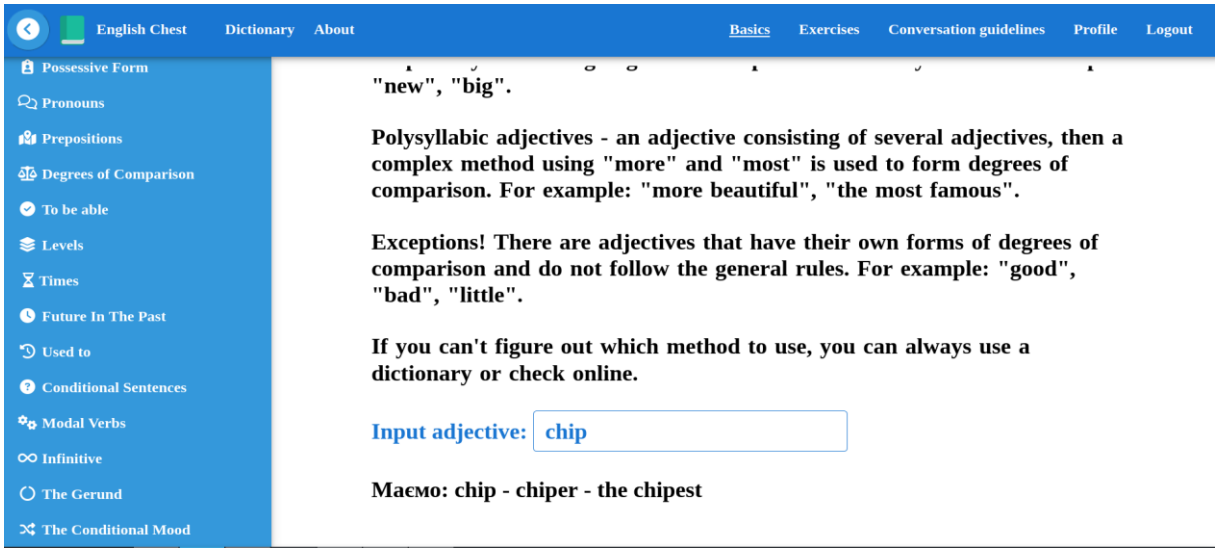


Рисунок 4.4 - Тест-кейс №4 «Пошук ступенів порівняння прикметника»

Тест-кейс №5 «Побудова речення з «To be able»:

– відкрити додаток;

– обрати пункт «To be able»;

– обрати негативну форму «Negation» (є можливість обрати позитивну форму);

– ввести дієслово «drive».

Очікуваним результатом є правильно побудовані речення, а саме «I am not able to drive» та «I does not know how to drive». Результат виконання тест-кейсу наведено на рис. 4.5.

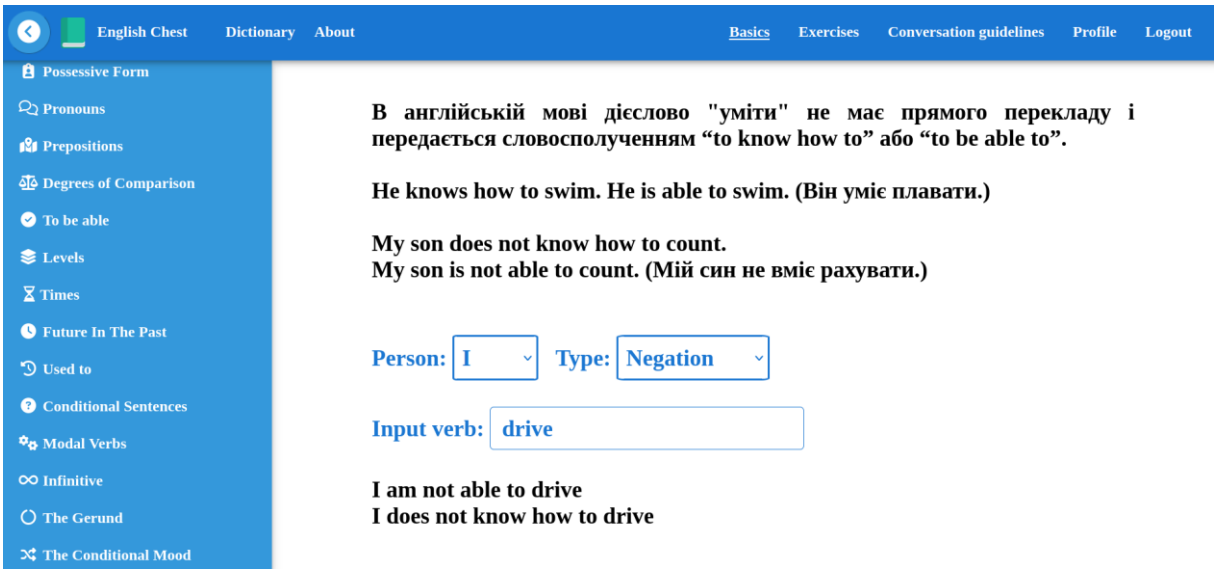


Рисунок 4.5 - Тест-кейс №5 «Побудова речення з «To be able»

Як можна було побачити, виконання тест-кейсів №1-5 відповідають очікуваним результати, усі тести пройдено успішно.

Й таким чином, у даному підрозділі було проведено ручні тести за методом «чорного ящика». Були надані пояснення щодо конкретного функціонала, описано розроблені тест-кейси. В цілому тестування додатка «English Chest» не виявило жодних проблем. Зокрема, кожен з тест-кейсів перевіряв різні аспекти функціональності додатка, що включало перевірку

коректності введення та виведення даних, відповідність результатів очікуванням, а також загальну стабільність роботи додатка.

4.3 Розробка інструкції користувача

Почнемо з того що користувачу необхідно завантажити всі необхідні компоненти у браузері (переглядач інтернету). Відразу після запуску додатка відкривається початкова сторінка додатка з якої можна почати використовувати додаток та всі його функції. Це зображено на рис. 4.1.

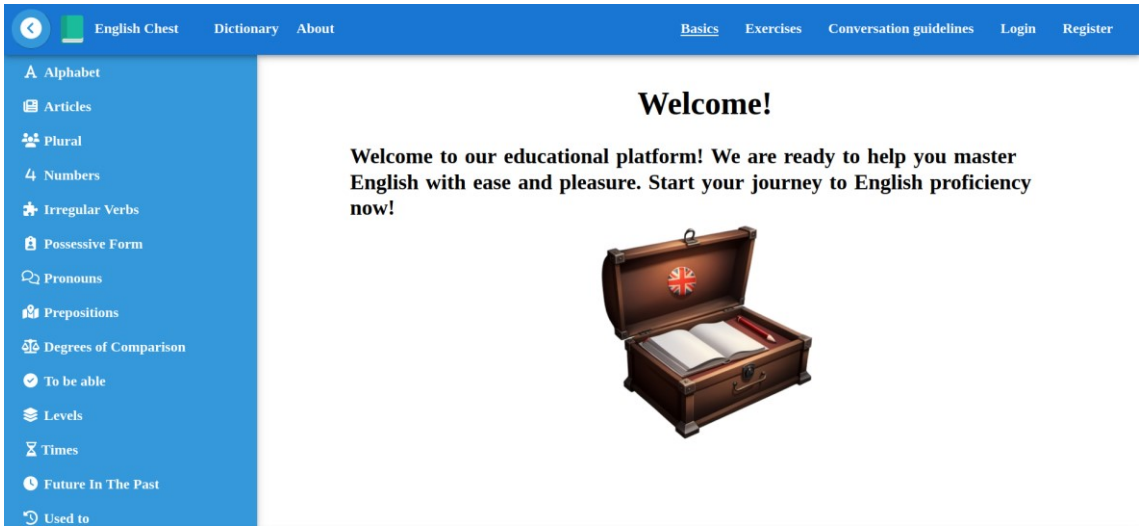


Рисунок 4.1 – Початкова сторінка

Для реєстрації та авторизації необхідно обрати пункт «Register» та «Login» відповідно (рис. 4.2, 4.3). Воно надає можливість зареєструватися або ввійти в обліковий запис.

Отже, при реєстрації нового облікового запису користувачеві необхідно ввести ім’я користувача, пароль та підтвердження пароля. Після натискання кнопки «Register!» автоматично створюється обліковий запис та користувач одразу авторизується.

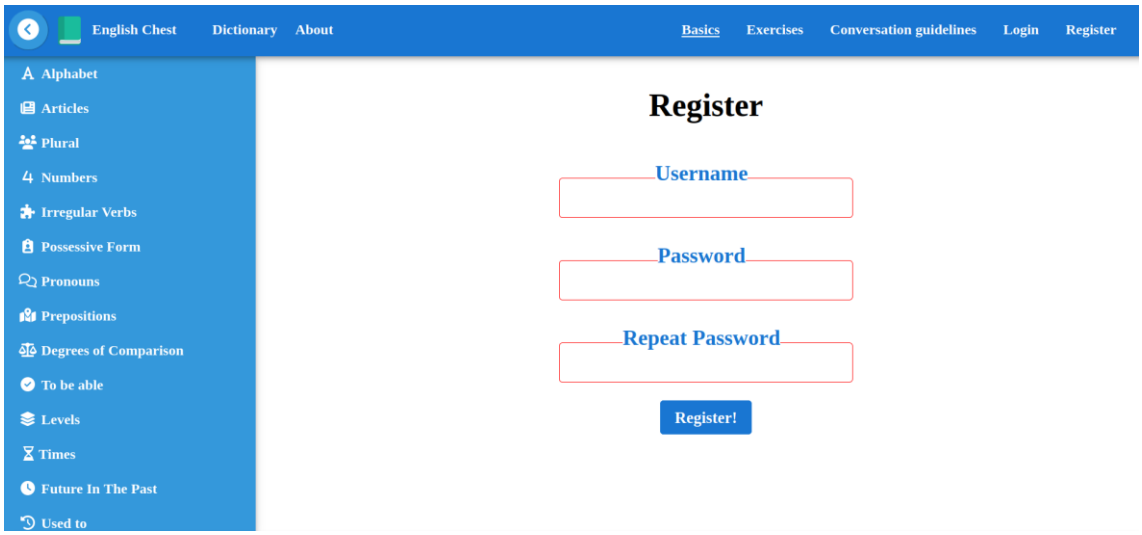


Рисунок 4.2 – Сторінка реєстрації облікового запису

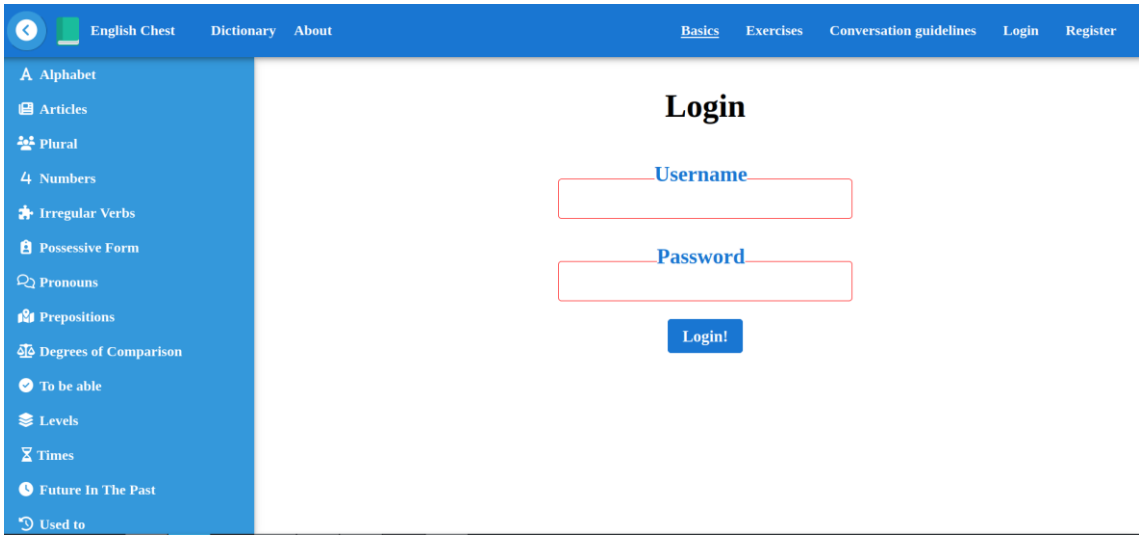


Рисунок 4.3 – Сторінка входу в обліковий запис

Головна частина сторінки надає доступ до основних функцій вебдодатка. Сторінка розділена на дві панелі: верхня містить кнопку згортання та розгортання бокового меню, логотип, розділи з додатковою інформацією «Dictionary» та «About», що надають інформацію про автора та доступ до словника (рис 4.4), розділи перемикання станів бокового меню «Basics», «Exercises» та «Conversation guidelines», кнопка для переходу до профілю облікового запису (рис. 4.5), кнопка для виходу з облікового запису.

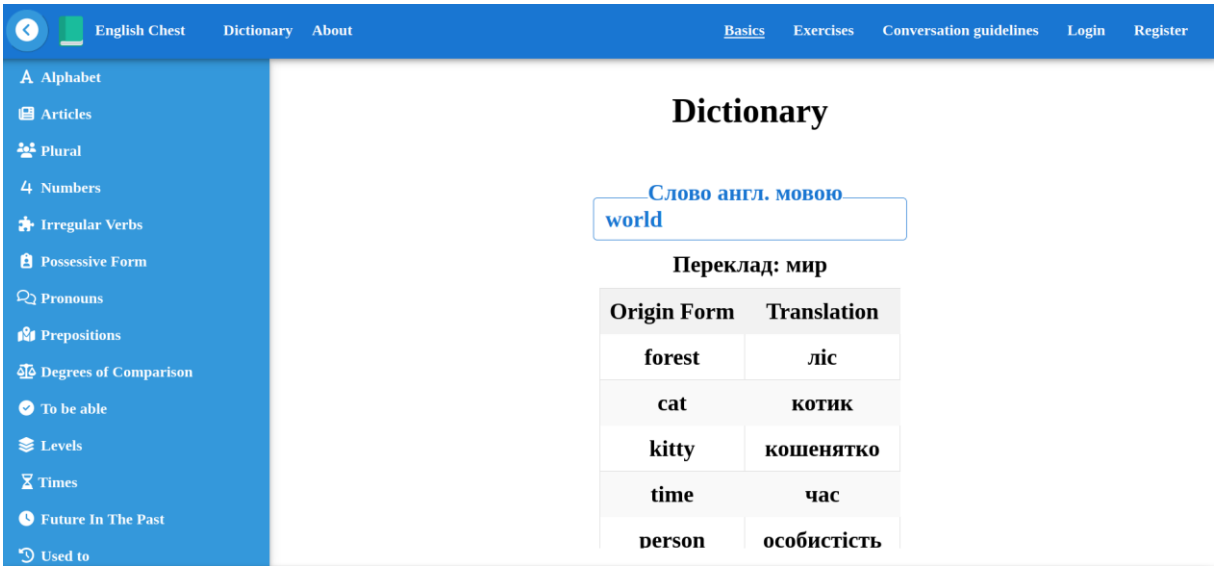


Рисунок 4.4 – Сторінка словник

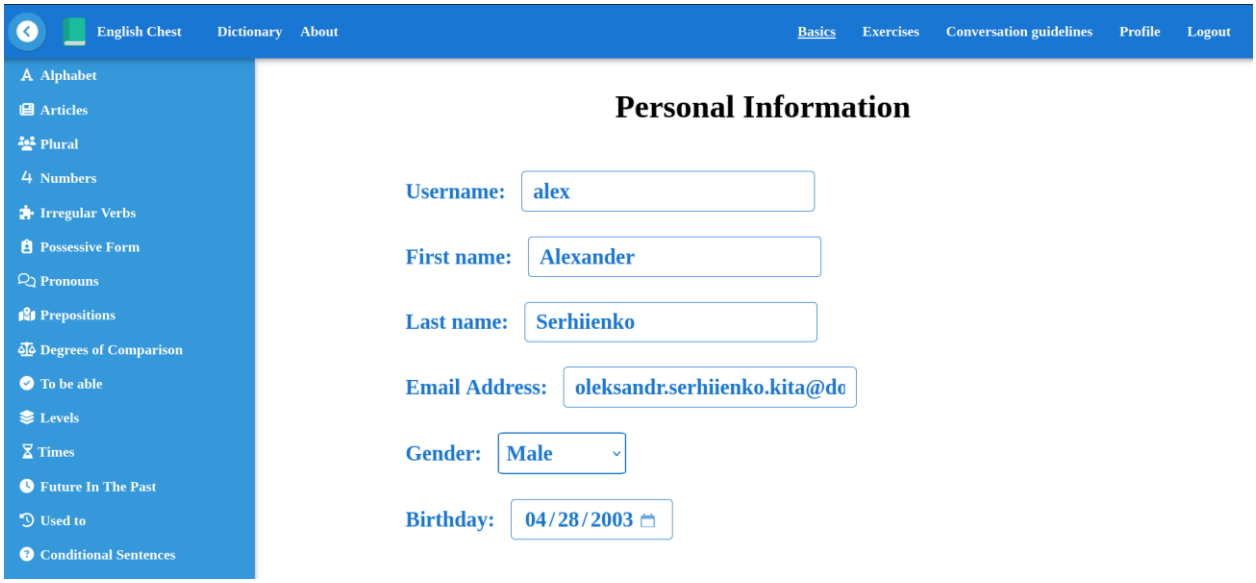


Рисунок 4.5 – Сторінка особистого профілю

Бокове меню відображає список доступних для навчання розділів. Деякі розділи списку описують довідкову інформацію з якою можна ознайомитися, а інші пропонують ще й додатковий функціонал для тренування навичків використання іноземної мови.

Розділ «Basics» містить базові відомості про іноземну мову з пунктами: «Alphabet», «Articles», «Plural», «Numbers», «Irregular Verbs», «Possessive Form», «Pronouns», «Prepositions», «Degrees of Comparison», «To be able»,

«Levels», «Times», «Future In The Past», «Used to», «Conditional Sentences», «Modal Verbs», «Infinitive», «The Gerund» та «The Conditional Mood». На рис. 4.6 зображено даний функціонал.

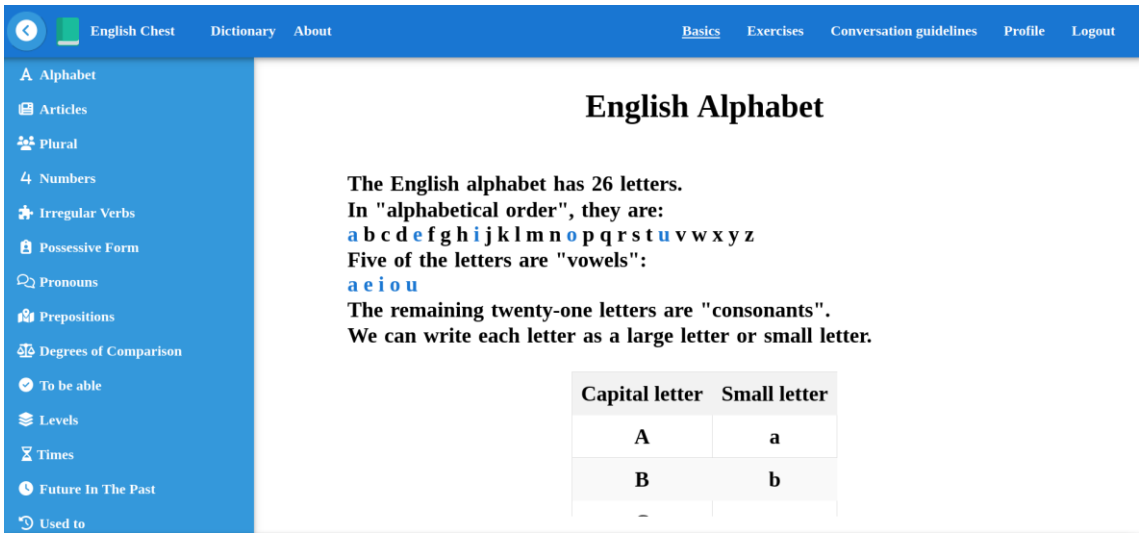


Рисунок 4.6 – Функціонал розділу «Basics»

Розділ «Exercises» містить вправи для тренувань та засвоєння матеріалу за темами: «Nouns articles», «Simple Tense», «Progressive Tense», «Perfect Tense», «Progressive Perfect Tense», «Future Perfect», «Conditional Sentences», «Gerund after prepositions» та «Gerund and infinitive». На рис. 4.7 зображено даний функціонал.

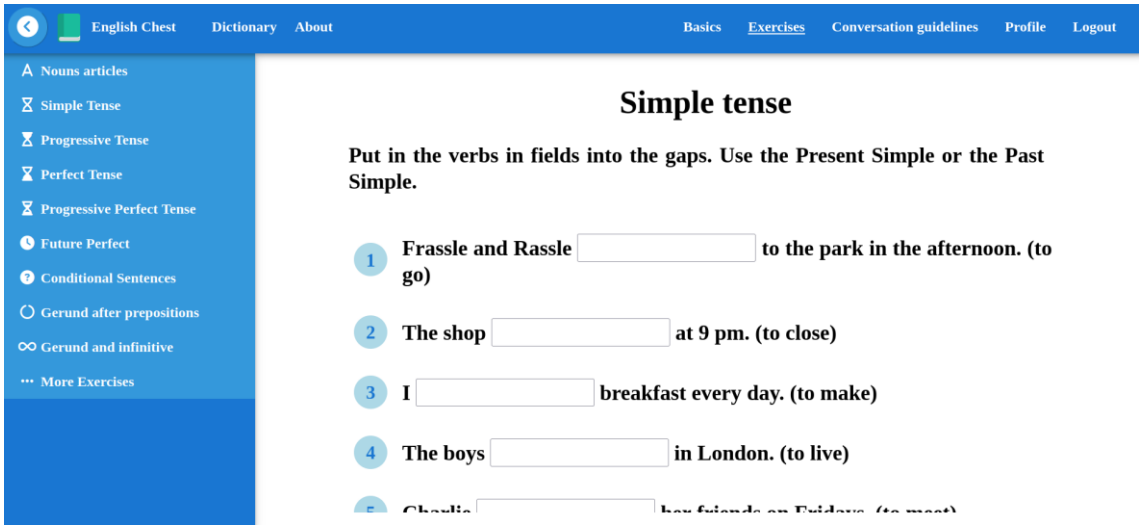


Рисунок 4.7 – Функціонал розділу «Exercises»

Розділ «Conversation guidelines» містить вказівки щодо ведення бесіди іноземною мовою за темами: «Greetings», «Goodbyes», «Introduction yourself» та «Socialite talk». На рис. 4.8 зображено даний функціонал.

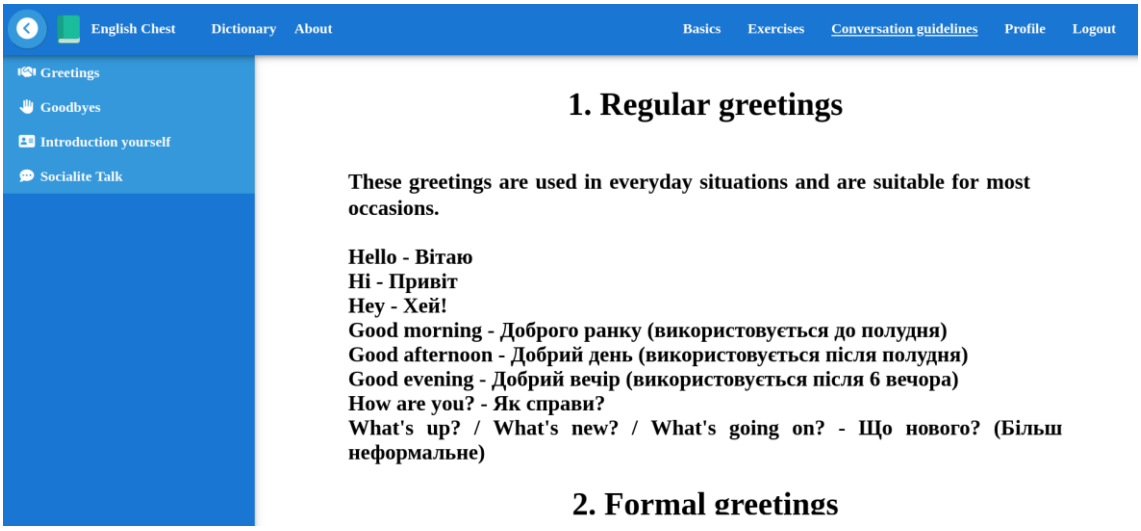


Рисунок 4.8 – Функціонал розділу «Conversation guidelines»

Також, для демонстрації всього функціоналу додатка було побудовано діаграму варіантів використання (рис. 4.9).

Діаграма варіантів використання (Use case diagram) демонструє набір ВВ (варіантів використання) і осіб-акторів (актантів) (які є спеціальним видом класів), а також їх зв’язки. Вони описують статичний вигляд ВВ системи й важливі для організації та моделювання поведінки системи [23]. Інакше кажучи, діаграма варіантів використання показує взаємодію між користувачами (та адміністраторами) й основними функціями системи. Діаграма наочно показує, які дії доступні кожному типу користувачів.

Користувачі можуть реєструватися, входити в систему, переглядати вміст, виконувати вправи, перевіряти свій прогрес та виходити з системи, тоді як адміністратори мають додаткову можливість управляти вмістом.

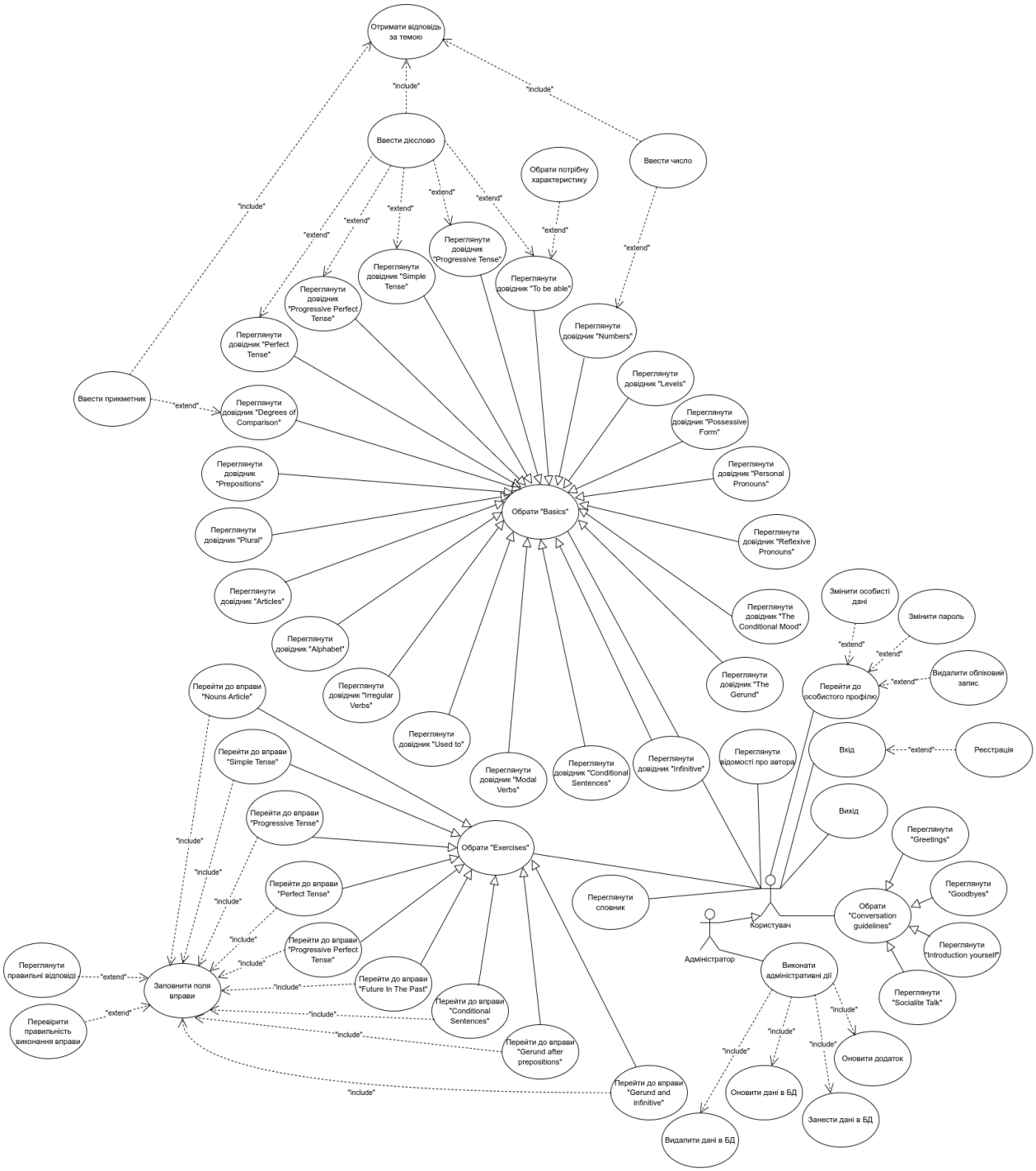


Рисунок 4.9 – Діаграма варіантів використання

4.4 Висновки до розділу 4

Як висновок можна сказати, що в даному розділі було проведено комплексне тестування вебдодатка з використанням методу «чорного ящика».

Виконання тест-кейсів продемонструвало коректність роботи основних функцій додатка, оскільки тести пройшли успішно та відповідали очікуваним результатам. Тестування було проведено вручну, що дозволило детально перевірити кожен аспект функціонала додатка. Метод «чорного ящика» виявився ефективним для перевірки додатка даного обсягу. Таким чином, вебдодаток відповідає встановленим вимогам та готовий до використання кінцевими користувачами.

На основі проведеного тестування можна зробити висновок, що додаток готовий до використання кінцевими користувачами, оскільки основні його функції працюють належним чином. Подальше тестування може бути сфокусовано на перевірці додаткових сценаріїв використання та навантажувальному тестуванні для впевненості у стабільній роботі додатка під різними умовами.

ЗАГАЛЬНІ ВИСНОВКИ

Отже, у бакалаврській дипломній роботі було проведено аналіз вибраної теми, визначено об’єкт і предмет дослідження, наведена мета робота. Розроблено програмний продукт для вивчення іноземних мов «English Check», що орієнтований на закріплення теоретичного матеріалу та придбання практичних навичок за основними розділами. Для розробки додатка були вибрані наступні інструменти, засоби та технології, а саме:

- Фреймворк для розробки клієнтської частини: Vue.js;
- Фреймворк для розробки серверної частини: Flask;
- Фреймворк для реалізації REST API: Flask-RESTful;
- ORM-система для взаємодії з базою даних: SQLAlchemy.

Основним напрямком стало розроблення інтерактивного навчального комплексу з вивчення іноземних мов з використанням вебтехнологій, зокрема, створення вебсервера для забезпечення взаємодії з користувачем та реалізація REST API для обміну даними з клієнтським додатком. Були розглянуті алгоритми роботи додатка, описані блок-схеми, розглянута багатошарова архітектура, проведені тести та наведена UML діаграма варіантів використання для демонстрації всього функціоналу, розглянута та розроблена структура бази даних та наведено схеми графічного інтерфейсу. Також була обрана мова програмування Python, JavaScript та редактор коду VSCodium.

Виконання бакалаврської дипломної роботи також сприяло розвитку аналітичних навичок і здатності до систематичного аналізу та проєктування програмного забезпечення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке фреймворк: пояснюємо простими словами. URL: https://brainlab.com.ua/uk/blog-uk/shho-take-frejmvork-poyasnyuyemo-prostymy-slovamy

2. Duolingo. The world's best way to learn a language. URL: https://www.duolingo.com/

3. LibreLingo - learn a language for free. URL: https://librelingo.app/

4. UneeBee | Open-source interactive courses URL: https://uneebee.com/

5. Memrise URL: https://app.memrise.com

6. Беррі Пол. Head First Python, друге видання / пер. з. англ. Г. Якубовська. Харків: ВД “Фабула”, 2021. 624 с.

7. Джейсон Р. Бріггс. PYTHON. Вступ до програмування [Текст] Джейсон Р Бріггс; перекл. З англ. Олександри Гординчук. Львів: Видавництво Старого Лева, 2023. 400с.

8. David Beazley, Brian Jones. Python Cookbook, Third Edition — O'Reilly Media, 2013. 704 p.

9. Luciano Ramalho. Fluent Python: Clear, Concise, and Effective Programming, 2015. 790 p.

10. Nitin Kumar Karma, RESTFul Web Services - The Python Flask Way: Build RESTful APIs using Python and Flask-Restful, Kindle Unlimited, 2018. 91p.

11. Jason Myers and Rick Copeland, Essential SQLAlchemy: Mapping Python to Databases, 2nd Edition. O'Reilly Media, 2016. 206 p.

12. Michael Bayer and Brandon Rhodes, Sqlalchemy: Database Access Using Python (Developer's Library), Addison-Wesley Professional; 1st edition (July 28, 2021). 504 p.

13. David Flanagan, JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language, 7nd Edition O'Reilly Media, 2020. 704 p.

14. Jiho Seok, The Complete Vue.js Developer's Guide: High Performance Web Applications with Vue.js, 1st Edition – Independently published (February 8, 2024). 162 p.

15. Josh Goldberg, Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript, 1st Edition, O'Reilly Media, (July 12, 2022). 317p.

16. Роберт Мартін: Чиста архітектура: мистецтво розробки програмного забезпечення: Фабула, 2019. 416с.

17. Paul Clements, Rick Kazman: Software Architecture in Practice: Addison- Wesley Professional, 2012. 322p.

18. Моделювання та аналіз програмного забезпечення: методичні вказівки до практичних та лабораторних занять. / Укладач: Л.В. Глазунова. Одеса: ДУІТЗ, 2021, 92с.

19. Martin Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, 1st Edition — O'Reilly Media, (May 2, 2017). 611 p.

20. Jenifer Tidwell, Charles Brewer, Aynne Valencia, Designing Interfaces: Patterns for Effective Interaction, 3rd Edition. O'Reilly Media, (February 18, 2020). 599 p.

21. Diana MacDonald, Practical UI Patterns for Design Systems: Fast-Track Interaction Design for a Seamless User Experience. Apress; 1st ed. edition (June 27, 2019). 315 p.

22. What Is QA in Software Testing? URL: https://www.3pillarglobal.com/insights/what-is-qa-in-software-testing/

23. Петрик М.Р. Моделювання програмного забезпечення: науково-методичний посібник / М.Р. Петрик, О.Ю. Петрик. Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2015. 200с.

24. Learn English Your Way. URL: https://www.englishclub.com/

25. English Tenses – Exercises. URL: https://www.englisch-hilfen.de/en/exercises_list/zeitformen.htm

ДОДАТОК А
ТЕКСТ ПРОГРАМНОГО ЗАСТОСУНКУ

Файл “src/backend/app.py”

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
from os import getenv
from flask import Flask
from flask_restful import Api
from flask_cors import CORS
from flask_bcrypt import Bcrypt
from dotenv import load_dotenv
from views.index_view import index_view
from resources.irregular_verbs_resource import IrregularVerbsResource
from resources.words_resource import WordsResource
from resources.user_accounts_resource import UserAccountResource
from resources.login_resource import LoginResource
from database.connector import DatabaseConnector
load_dotenv()
app = Flask(
    import_name=__name__,
    template_folder="templates",
    static_folder="static",
)
app.config["SECRET_KEY"] = getenv("SECRET_KEY")
api = Api(app=app)
bcrypt = Bcrypt(app=app)
cors = CORS(app=app)
app.add_url_rule("/", view_func=index_view)
api.add_resource(
    IrregularVerbsResource,
    "/api/irregular-verbs",
)
api.add_resource(
    WordsResource,
    "/api/words",
)
api.add_resource(
    UserAccountResource,
    "/api/users",
)
api.add_resource(
    LoginResource,
    "/api/login",
)
def main():
    app.run(debug=True, port=5000)
if __name__ == "__main__":
```

```
    main()
```

Файл “src/backend/getting_ready_app.py”

```
if __name__ == "__main__":
    import utils.getting_ready_database
```

Файл “src/backend/.env”

```
DB_URL="sqlite:///database/database.sqlite3"
SECRET_KEY="Your_secret_key"
```

Файл “src/backend/views/index_view.py”

```
#! /usr/bin/python3
# -*- coding: utf-8 -*-
from flask import render_template
def index_view():
    return render_template("index.html")
```

Файл “src/backend/utils/date_utils.py”

```
from typing import Optional
from datetime import date, datetime
DATE_ISO_FORMAT = "%Y-%m-%d"

def date_to_str(dt: Optional[date]) -> str:
    if dt:
        return dt.strftime(DATE_ISO_FORMAT)
    else:
        return ""
def str_to_date(date_str: str) -> Optional[date]:
    if date_str:
        return datetime.strptime(date_str, DATE_ISO_FORMAT).date()
```

Файл “src/backend/utils/getting_ready_database.py”

```
from utils.insert_data_from_sql import insert_data_from_sql_file
paths_to_sql_file = [
    "utils/sql/insert_irregular_verbs.sql",
    "utils/sql/insert_words.sql",
]
for path_to_sql_file in paths_to_sql_file:
    insert_data_from_sql_file(path_to_sql_file)
```

Файл “src/backend/utils/insert_data_from_sql.py”

```
from sqlalchemy import text
from database.connector import DatabaseConnector
connector = DatabaseConnector()
def insert_data_from_sql_file(path_to_sql_file: str) -> None:
    with open(path_to_sql_file, "r") as file:
        sql_query: str = file.read().lower()
    with connector.engine.connect() as connection:
        connection.execute(text(sql_query))
        connection.commit()
```

Файл “src/backend/utils/words_util.py”

```
from typing import Optional
def clear_origin(origin: str) -> Optional[str]:
    if origin:
        return (
            origin.lower()
            .replace("", "")
            .replace("'", "")
            .replace("(", "")
            .replace(")", "")
            .replace("{", "")
            .replace("}", "")
            .replace(",", "")
            .replace(".", "")
            .replace("[", "")
            .replace("]", "")
            .replace("$", "")
            .replace("#", "")
            .replace("@", "")
            .replace("!", "")
            .replace("?", "")
            .replace("an ", "")
            .replace("a ", "")
            .replace("the ", "")
            .replace("to ", "")
            .replace("of ", "")
        )
```

Файл “src/backend/utils/sql/insert_irregular_verbs.sql”

```
INSERT INTO irregular_verbs(`base_form`, `past_tense`, `past_participle`)
VALUES
    ('be', 'was/were', 'been'),
    ('begin', 'began', 'begun'),
    ('break', 'broke', 'broken'),
    ('bring', 'brought', 'brought'),
    ('buy', 'bought', 'bought'),
    ('build', 'built', 'built'),
    ('choose', 'chose', 'chosen'),
    ('come', 'came', 'come'),
```

```
  ('cost', 'cost', 'cost'),
  ('cut', 'cut', 'cut'),
  ('do', 'did', 'done'),
  ('draw', 'drew', 'drawn'),
  ('drive', 'drove', 'driven'),
  ('eat', 'ate', 'eaten'),
  ('feel', 'felt', 'felt'),
  ('find', 'found', 'found'),
  ('get', 'got', 'got'),
  ('give', 'gave', 'given'),
  ('go', 'went', 'gone'),
  ('have', 'had', 'had'),
  ('hear', 'heard', 'heard'),
  ('hold', 'held', 'held'),
  ('keep', 'kept', 'kept'),
  ('know', 'knew', 'known'),
  ('leave', 'left', 'left'),
  ('lead', 'led', 'led'),
  ('let', 'let', 'let'),
  ('lie', 'lay', 'lain'),
  ('lose', 'lost', 'lost'),
  ('make', 'made', 'made'),
  ('mean', 'meant', 'meant'),
  ('meet', 'met', 'met'),
  ('pay', 'paid', 'paid'),
  ('put', 'put', 'put'),
  ('run', 'ran', 'run'),
  ('say', 'said', 'said'),
  ('see', 'saw', 'seen'),
  ('sell', 'sold', 'sold'),
  ('send', 'sent', 'sent'),
  ('set', 'set', 'set'),
  ('sit', 'sat', 'sat'),
  ('speak', 'spoke', 'spoken'),
  ('spend', 'spent', 'spent'),
  ('stand', 'stood', 'stood'),
  ('take', 'took', 'taken'),
  ('teach', 'taught', 'taught'),
  ('tell', 'told', 'told'),
  ('think', 'thought', 'thought'),
  ('understand', 'understood', 'understood'),
  ('wear', 'wore', 'worn'),
  ('win', won', won'),
  ('write', 'wrote', 'written');
```

Файл “src/backend/utils/sql/insert_words.sql”

```
INSERT INTO words(`origin`, `translation`)
VALUES
  ('forest', 'ліс'),
  ('cat', 'Котик'),
  ('kitty', 'Кошенятко'),
```

('time', 'час'),
('person', 'особистість'),
('year', 'рік'),
('way', 'шлях, дорога'),
('day', 'день'),
('thing', 'річ'),
('man', 'чоловік'),
('world', 'мир'),
('life', 'життя'),
('hand', 'рука'),
('part', 'частина'),
('child', 'дитина'),
('eye', 'очі'),
('bee', 'Бджола'),
('woman', 'жінка'),
('place', 'місце'),
('week', 'тиждень'),
('case', 'справа'),
('point', 'сенс, точка'),
('government', 'держава'),
('company', 'компанія'),
('number', 'номер'),
('group', 'група'),
('problem', 'проблема'),
('fact', 'факт'),
('be', 'бути'),
('have', 'щось мати'),
('do', 'робити'),
('say', 'сказати'),
('get', 'отримати'),
('make', 'робити'),
('go', 'йти'),
('know', 'знати'),
('take', 'брати, взяти'),
('see', 'бачити'),
('come', 'приходити, приїжджати'),
('think', 'думати'),
('look', 'дивитися'),
('want', 'хотіти'),
('give', 'давати'),
('use', 'використовувати'),
('find', 'знаходити'),
('tell', 'розповідати'),
('ask', 'запитувати'),
('work', 'працювати'),
('seem', 'здаватися'),
('feel', 'відчувати'),
('try', 'намагатися'),
('leave', 'залишати'),
('call', 'дзвонити'),
('good', 'хороший'),
('important', 'важливий'),

('few', 'мало, кілька'),
('public', 'публічний'),
('bad', 'поганий'),
('same', 'одне і теж, так само'),
('able', 'здатний'),
('to', 'до, в'),
('of', 'коли ти частина чого-то'),
('in', 'в'),
('for', 'для'),
('on', 'на'),
('with', 'з'),
('at', 'в'),
('by', 'до, в, повз'),
('from', 'з'),
('up', 'вгору'),
('about', 'про'),
('into', 'в'),
('over', 'над'),
('after', 'після'),
('the', 'Цей'),
('and', 'і'),
('a', 'невизначений артикль'),
('that', 'той'),
('I', 'я'),
('it', 'це'),
('not', 'частка не'),
('he', 'він'),
('as', 'як'),
('you', 'ти'),
('this', 'це'),
('but', 'але'),
('his', 'його'),
('her', 'її'),
('they', 'вони'),
('she', 'вона'),
('or', 'або'),
('a', 'один'),
('an', 'один'),
('will', 'буду'),
('my', 'мій'),
('one', 'один'),
('all', 'всі'),
('would', 'буду, маю намір'),
('there', 'там'),
('their', 'їх'),
('apple', 'яблуко'),
('genetic engineering', 'генна інженерія'),
('nature', 'природа'),
('genes', 'гени'),
('gave birth to', 'народила'),
('birth', 'народження'),
('transfer', 'передача'),

('created', 'створено'),
('duck', 'качка'),
('beans', 'квасоля'),
('effect', 'ефект'),
('rats', 'щури'),
('moths', 'молі'),
('eyesight', 'зір'),
('prevent', 'запобігти'),
('starving', 'голодуючи'),
('gene', 'ген'),
('book', 'Книга'),
('believe', 'Вірити'),
('street', 'Вулиця'),
('among', 'Серед'),
('learn', 'Вчитися'),
('hurry', 'Поспішати'),
('you', 'Ви'),
('thirty', 'Тридцять'),
('often', 'Часто'),
('morning', 'Ранок'),
('he', 'Він'),
('happen', 'Відбудеться'),
('was', 'Був'),
('dog', 'Собака'),
('although', 'Хоча'),
('silent', 'Мовчання'),
('bring', 'Принести'),
('hardly', 'Навряд чи'),
('strange', 'Дивно'),
('kid', 'Дитина'),
('talk', 'Говорити'),
('mother', 'Мати'),
('mountain', 'Гора'),
('sister', 'Сестра'),
('ball', "М'яч"),
('kind', 'Вид'),
('his', 'Його'),
('black', 'Чорні'),
('each', 'Кожен'),
('young', 'Молодий'),
('thin', 'Тонкий'),
('nod', 'Кивок'),
('direction', 'Напрямок'),
('cause', 'Причина'),
('not', 'Не'),
('her', 'Її'),
('stood', 'Стояв'),
('half', 'Половина'),
('show', 'Показати'),
('red', 'Червоний'),
('month', 'Місяць'),
('office', 'Офіс'),

('keep', 'Тримати'),
('but', 'Але'),
('sun', 'Сонце'),
('it', 'Це'),
('pain', 'Біль'),
('in', 'В'),
('finally', 'Нарешті'),
('afraid', 'Боїться'),
('notice', 'Повідомлення'),
('for', 'Для'),
('ham', 'Шинка'),
('grab', 'Захоплення'),
('had', 'Був'),
('do', 'Робити'),
('feet', 'Ноги'),
('hell', 'Пекло'),
('slip', 'Прослизання'),
('knock', 'Стукати'),
('wolf', 'Вовк'),
('led', 'Привели'),
('receive', 'Отримати'),
('brown', 'Карі'),
('shut', 'Shut'),
('box', 'Коробка'),
('wonder', 'Чудо'),
('she', 'Вона'),
('eight', 'Вісім'),
('loud', 'Голосно'),
('feel', 'Відчувати'),
('during', 'Під час'),
('next', 'Поруч'),
('wheel', 'Колесо'),
('beautiful', 'Гарна'),
('beyond', 'За'),
('begin', 'Почати'),
('him', 'Йому'),
('dark', 'Темний'),
('became', 'Став'),
('hate', 'Ненависть'),
('as', 'Як'),
('along', 'Разом'),
('shadow', 'Тінь'),
('lying', 'Лежачи'),
('pass', 'Пройти'),
('slowly', 'Повільно'),
('retrieve', 'Отримати'),
('said', 'Сказав'),
('course', 'Звичайно'),
('instead', 'Замість'),
('secret', 'Секрет'),
('window', 'Вікно'),
('ear', 'Вухо'),

('wild', 'Дикий'),
('we', 'Ми'),
('dead', 'Мертвих'),
('picture', 'Картина'),
('color', 'Колір'),
('colour', 'Колір'),
('all', 'Все'),
('hour', 'Час'),
('interest', 'Інтерес'),
('fair', 'Справедливої'),
('this', 'Це'),
('wanted', 'Хотів'),
('king', 'Король'),
('east', 'Схід'),
('have', 'Мати <щось>'),
('matter', 'Має значення'),
('present', 'Справжнє'),
('wing', 'Крило'),
('there', 'Там'),
('since', 'з'),
('bit', 'біт'),
('main', 'Головний'),
('what', 'Що'),
('cry', 'Клич'),
('living', 'Живий'),
('visit', 'Візит'),
('out', 'На'),
('heart', 'Серце'),
('remain', 'Залишаються'),
('following', 'Наступне'),
('up', 'Вгору'),
('answer', 'Відповідь'),
('appear', "З'являються"),
('sharp', 'Гострий'),
('one', 'Один'),
('small', 'Маленький'),
('store', 'Магазин'),
('subject', "Суб'єкт"),
('from', 'Від'),
('gotten', 'Отримав'),
('marry', 'Одружилися'),
('understood', 'Розуміти'),
('me', 'мене'),
('doing', 'Робити'),
('rather', 'Досить'),
('lake', 'Озеро'),
('go', 'Йти'),
('four', 'Чотири'),
('doubt', 'Сумніви'),
('page', 'Сторінка'),
('five', "П'ять"),
('move', 'Рухатися'),

('teeth', 'Зуби'),
('skin', 'Шкіра'),
('finish', 'Закінчити'),
('live', 'Наживо'),
('force', 'Сила'),
('station', 'Станція'),
('like', 'Подобається, подібно'),
('suddenly', 'Раптово'),
('spoke', 'Говорив'),
('study', 'Дослідження'),
('when', 'Коли'),
('spring', 'Весна'),
('could', 'Може'),
('body', 'Тіло'),
('then', 'Тоді'),
('fire', 'Вогонь'),
('seat', 'Місце'),
('building', 'Будівництво'),
('be', 'Бути'),
('mouth', 'Рот'),
('not', 'Ні'),
('desk', 'Стіл'),
('them', 'їх'),
('leave', 'Залишити'),
('save', 'Рятувати'),
('hung', 'Висіли'),
('did', 'Зробив'),
('smile', 'Посмішка'),
('job', 'Робота'),
('fifty', "П'ятдесят"),
('been', 'Був'),
('state', 'Стан'),
('war', 'Війна'),
('gold', 'Золото'),
('golden', 'Золотий'),
('now', 'Тепер'),
('family', "Сім'я"),
('dress', 'Платье'),
('dressing', 'Одягання'),
('undressing', 'Роздягати'),
('neither', 'Ні'),
('nobody', 'Ніхто'),
('look', 'Погляд'),
('road', 'Дорога'),
('cover', 'Покриття'),
('send', 'Відправити'),
('back', 'Назад'),
('front', 'Фронт'),
('realize', 'Чи розуміють'),
('stick', 'Палка'),
('my', 'Мій'),
('laugh', 'Сміятися'),

('gun', 'Пістолет'),
('straight', 'Прямий'),
('no', 'Ні'),
('gave', 'Далі'),
('heavy', 'Важкий'),
('darkness', 'Темнота'),
('your', 'Ваш'),
('across', 'Через'),
('waiting', 'Очікування'),
('experience', 'Досвід'),
('which', 'Який'),
('step', 'Крок'),
('fellow', 'Побратим'),
('bar', 'Бар'),
('about', 'О <...>'),
('quite', 'Досить'),
('sign', 'Знак'),
('imagine', 'Уявіть собі'),
('change', 'Зміна'),
('stuff', 'Матеріал'),
('ship', 'Корабель'),
('down', 'Вниз'),
('later', 'Пізніше'),
('decide', 'Вирішити'),
('clear', 'Ясно'),
('together', 'Разом'),
('expect', 'Очікувати'),
('thus', 'Таким чином'),
('forget', 'Забувають'),
('who', 'Хто'),
('soon', 'Скоро'),
('hit', 'Хіт'),
('nine', "Дев'ять"),
('can', 'Може'),
('least', 'Мінімум'),
('corner', 'Кут'),
('agree', 'Згоден'),
('know', 'Знати'),
('second', 'Другий'),
('beaver', 'Бобр'),
('if', 'Якщо'),
('kill', 'Убити'),
('interior', "Інтер'єр"),
('just', 'Просто'),
('watch', 'Дивитися'),
('happy', 'Щасливий'),
('rain', 'Дощ'),
('wife', 'Дружина'),
('phone', 'Телефон'),
('block', 'Блок'),
('get', 'Отримати'),
('hard', 'Жорсткий'),

('several', 'Ряд'),
('fix', 'виправити'),
('over', 'над'),
('wait', 'чекати'),
('city', 'місто'),
('company', 'Компанія'),
('more', 'Більше'),
('sat', 'їх'),
('news', 'Новини'),
('warm', 'Теплий'),
('some', 'Деякі'),
('feeling', 'Відчуття'),
('summer', 'Літо'),
('business', 'Бізнес'),
('high', 'високий'),
('different', 'Різні'),
('dinner', 'Вечеря'),
('come', 'Приходь'),
('guy', 'парубок'),
('sit', 'сидіти'),
('somewhere', 'Десь'),
('board', 'Рада'),
('idea', 'Ідея'),
('actually', 'Насправді'),
('space', 'Простір'),
('so', 'Так'),
('line', 'Лінія'),
('caught', 'спійманий'),
('chest', 'Сундук'),
('other', 'Інше'),
('fear', 'Страх'),
('old', 'Старий'),
('write', 'Напишіть'),
('little', 'Маленька'),
('miss', 'Міс'),
('broken', 'зламаний'),
('rich', 'Багатий'),
('see', 'Див.'),
('bad', 'поганий'),
('early', 'рано'),
('pause', 'Пауза'),
('here', 'Тут'),
('hair', 'Волосся'),
('bright', 'Яскравий'),
('single', 'Одинокий'),
('things', 'речі'),
('minute', 'хвилина'),
('daughter', 'Дочка'),
('yard', 'Двір'),
('hand', 'Рука'),
('everything', 'Кожен'),
('enter', 'Введіть'),

('further', 'далі'),
('either', 'Або'),
('please', 'Будь ласка'),
('shot', 'постріл'),
('leaving', 'Залишити'),
('still', 'Ще'),
('gray', 'сірий'),
('carry', 'Нести'),
('soft', "м'який"),
('way', 'Шлях'),
('follow', 'Дотримуйтесь'),
('breath', 'Дихання'),
('allow', 'Дозволити'),
('again', 'знову'),
('hope', 'Надія'),
('scary', 'страшно'),
('onto', 'На'),
('right', 'Право'),
('blood', 'кров'),
('lady', 'Дама'),
('pale', 'Блідий'),
('only', 'ТІЛЬКИ'),
('won', 'виграв'),
('truth', 'Правда'),
('tie', "зв'язати"),
('am', 'I + (to be), -> I am'),
('human', 'Людина'),
('air', 'Повітря'),
('reply', 'Відповідь'),
('how', 'Як'),
('perhaps', 'Можливо'),
('after', 'Після'),
('ought', 'ТАК'),
('think', 'Подумайте'),
('set', 'Набір'),
('yourself', 'Сам'),
('or', 'або'),
('tree', 'Дерево'),
('form', 'Форма'),
('doctor', 'Доктор'),
('got', 'Отримано'),
('run', 'бігти'),
('worry', 'турбуватися'),
('sweet', 'Солодкий'),
('death', 'Смерть'),
('good', 'Добре'),
('fat', 'жир'),
('grin', 'Посмішка'),
('eye', 'Око'),
('exactly', 'Точно'),
('escape', 'Втеча'),
('play', 'Грати'),

('else', 'Інакше'),
('has', 'Має'),
('thought', 'думка'),
('explain', 'пояснити'),
('fill', 'Заповніть'),
('general', 'Загальний'),
('day', 'День'),
('bed', 'ліжко'),
('certain', 'Впевнений'),
('shout', 'Крик'),
('two', 'Два'),
('lot', 'Багато'),
('nose', 'Ніс'),
('aware', 'Знають'),
('than', 'Чим'),
('point', 'Точка'),
('brutal', 'Брутальний'),
('consider', 'Розглянути'),
('before', 'Перед'),
('money', 'Гроші'),
('continue', 'Продовжити'),
('grey', 'Сірі'),
('where', 'Де'),
('getting', 'Отримувати'),
('drink', 'Напій'),
('card', 'Карта'),
('very', 'Дуже'),
('off', 'Вимкнути'),
('suppose', 'Припустимо'),
('cloud', 'Хмара'),
('say', 'Кажуть'),
('whole', 'Ціле'),
('sky', 'Небо'),
('spot', 'Пляма'),
('came', 'Прийшов'),
('pull', 'Тягнути'),
('mark', 'Знак'),
('situation', 'Ситуація'),
('any', 'Будь'),
('park', 'Паркувати'),
('lower', 'Опустити'),
('bone', 'Кость'),
('rose', 'Троянди'),
('machine', 'Машина'),
('never', 'Ніколи'),
('reach', 'Досяжність'),
('whose', 'Якого'),
('through', 'Через'),
('power', 'Сила'),
('week', 'Тиждень'),
('figure', 'Фігура'),
('view', 'Вид'),

('even', 'Навіть'),
('thank', 'Спасибі'),
('thanks', 'Спасибі'),
('lie', 'Брехня'),
('project', 'Проект'),
('much', 'Багато'),
('wall', 'Стіна'),
('act', 'Акт'),
('bottle', 'Пляшка'),
('real', 'Реальний'),
('dollar', 'Долар'),
('broke', 'Ламати'),
('long', 'Довгий'),
('wish', 'Бажання'),
('cross', 'Хрест'),
('position', 'Позиція'),
('away', 'Геть'),
('best', 'Кращий'),
('master', 'Майстер'),
('sea', 'Море'),
('something', 'Щось'),
('read', 'Читати'),
('game', 'Гра'),
('field', 'Поле'),
('first', 'Перший'),
('thinking', 'Мислення'),
('pocket', 'Карман'),
('fresh', 'Свіжий'),
('make', 'Марка'),
('inside', 'Всередині'),
('stone', 'Камінь'),
('join', 'Приєднатися'),
('head', 'Голова'),
('stay', 'Перебування'),
('strong', 'Сильний'),
('worse', 'Гірше'),
('want', 'Хочу'),
('question', 'Питання'),
('russian', 'Русский'),
('ukrainian', 'Український'),
('fly', 'Літати'),
('turn', 'Включити'),
('horse', 'Кінь'),
('face', 'Особа'),
('cold', 'Холодний'),
('baby', 'Дитина'),
('offer', 'Пропозиція'),
('made', 'Зробив'),
('die', 'Помер'),
('knee', 'Коліно'),
('eat', 'їсти'),
('seem', 'Здається'),

('sir', 'Сер'),
('police', 'Поліція'),
('safe', 'Безпечний'),
('call', 'Виклик'),
('dream', 'Сон'),
('silence', 'Мовчання'),
('jump', 'Стрибає'),
('ask', 'спитати'),
('ran', 'RAN'),
('murder', 'Вбивство'),
('smiling', 'Посміхаючись'),
('should', 'Повинен'),
('sometimes', 'Іноді'),
('track', 'Трек'),
('afternoon', 'Днем'),
('put', 'Ставити'),
('bird', 'Птах'),
('creature', 'Істота'),
('take', 'Взяти'),
('tomorrow', 'Завтра'),
('bill', 'Законопроект'),
('stand', 'Стояти'),
('fast', 'Швидкий'),
('monster', 'Монстра'),
('saw', 'Пила'),
('son', 'Син'),
('add', 'Додати'),
('system', 'Система'),
('around', 'Навколо'),
('herself', 'Сама'),
('himself', 'Сам'),
('floor', 'Пол'),
('beat', 'Бити'),
('our', 'Наш'),
('pretty', 'вродлива'),
('none', 'Ні'),
('funny', 'Смішно'),
('door', 'Двері'),
('paper', 'Папір'),
('hospital', 'Лікарня'),
('last', 'Останній'),
('shoulder', 'Плече'),
('sonny', 'Синок'),
('raise', 'підіймати'),
('tell', 'Сказати'),
('care', 'Догляд'),
('cheek', 'Щека'),
('check', 'Перевірити'),
('service', 'Сервіс'),
('might', 'Міць'),
('alone', 'Поодинці'),
('meant', 'Чи означає'),

('common', 'Загальна'),
('own', 'Власний'),
('making', 'Рішень'),
('push', 'Поштовх'),
('lock', 'Замок'),
('night', 'Ніч'),
('near', 'Поруч'),
('nature', 'Природа'),
('sudden', 'Раптове'),
('year', 'Рік'),
('finger', 'Палець'),
('bear', 'Ведмідь'),
('usually', 'Зазвичай'),
('must', 'Повинен'),
('host', 'Господар'),
('struck', 'Ударив'),
('settle', 'Осідає'),
('because', 'Тому що'),
('return', 'Повернення'),
('tear', 'Сльоза'),
('slow', 'Повільно'),
('voice', 'Голос'),
('fact', 'Факт'),
('chapter', 'Глава'),
('threw', 'Кинув'),
('such', 'Такий'),
('understand', 'Розуміти'),
('middle', 'Середній'),
('train', 'Поїзд'),
('room', 'Номер'),
('guess', 'Вгадати'),
('note', 'Примітка'),
('unless', 'Якщо не'),
('place', 'Місце'),
('table', 'Стіл'),
('crazy', 'Божевільний'),
('ass', 'Осел'),
('those', 'Ті'),
('dear', 'Дорогий'),
('sent', 'Послав'),
('beneath', 'Під'),
('work', 'Робота'),
('trying', 'Намагається'),
('smoke', 'Дим'),
('nothing', 'Нічого'),
('mile', 'Милі'),
('promise', 'Обіцянка'),
('cool', 'Охолодитися'),
('most', 'Найбільш'),
('probably', 'Ймовірно'),
('thousand', 'Тисяча'),
('alive', 'Живим'),

('house', 'Будинок'),
('type', 'Типу'),
('hall', 'Зал'),
('crowd', 'Натовп'),
('why', 'Чому'),
('deep', 'Глибоко'),
('nice', 'Хороший'),
('quarter', 'Квартал'),
('may', 'Може'),
('glass', 'Скло'),
('moving', 'Переміщення'),
('written', 'Написати'),
('side', 'Сторона'),
('standing', 'Стоїть'),
('almost', 'Майже'),
('distance', 'Відстань'),
('great', 'Великий'),
('trouble', 'Біда'),
('whom', 'Кого'),
('laid', 'Поклав'),
('these', 'Це'),
('however', 'Однак'),
('law', 'Закон'),
('hat', 'Капелюх'),
('white', 'Білий'),
('taken', 'Прийнято'),
('meet', 'Зустріч'),
('trust', 'Довіра'),
('people', 'Люди'),
('hey', 'Гей!'),
('quiet', 'Тихий'),
('beauty', 'Краса'),
('upon', 'На'),
('sort', 'Рода'),
('sight', 'Зір'),
('expression', 'Вираз'),
('left', 'Лівий'),
('lay', 'Лежати'),
('throat', 'Горло'),
('arrive', 'Прибути'),
('open', 'Відкритим'),
('town', 'Місто'),
('age', 'Вік'),
('center', 'Центр'),
('sleep', 'Сон'),
('wide', 'Широкий'),
('give', 'Віддання'),
('friend', 'Друг'),
('school', 'Школа'),
('desire', 'Бажання'),
('manner', 'Манера'),
('clear', 'Ясно'),

('lost', 'втрачений'),
('hole', 'Діра'),
('merry', 'Весела'),
('three', 'Три'),
('green', 'Зелений'),
('certainly', 'Звичайно'),
('moon', 'Місяць'),
('knew', 'Знав'),
('gonna', 'Збирається'),
('terrible', 'Жахливий'),
('circle', 'Коло'),
('another', 'Інший'),
('lip', 'Губа'),
('earth', 'Земля'),
('leader', 'Лідер'),
('once', 'Коли'),
('round', 'Круглий'),
('kiss', 'Поцілунок'),
('conversation', 'Розмова'),
('guard', 'Охоронець'),
('mind', 'Розум'),
('scream', 'Кричати'),
('party', 'Партія'),
('truck', 'Вантажівка'),
('took', 'Взяв'),
('touch', 'Сенсорний'),
('movie', 'Кіно'),
('effort', 'Зусилля'),
('light', 'Світло'),
('blue', 'Синій'),
('possible', 'Можливо'),
('fight', 'Боротися'),
('yes', 'Так'),
('hundred', 'Сто'),
('kitchen', 'Кухня'),
('reading', 'Читати'),
('love', 'Любов'),
('know', 'Знати'),
('piece', 'Шматок'),
('group', 'Група'),
('end', 'Кінець'),
('running', 'Працює'),
('attention', 'Увага'),
('mention', 'Згадка'),
('boy', 'Хлопчик'),
('large', 'Великий'),
('silver', 'Срібло'),
('forth', 'Уперед'),
('new', 'Новий'),
('surprise', 'Сюрприз'),
('damn', 'Проклятий'),
('haven', 'Гавань'),

('leg', 'Нога'),
('beside', 'Поруч'),
('folk', 'Народний'),
('strength', 'Міцність'),
('sound', 'Звук'),
('talking', 'Говорити'),
('glance', 'Погляд'),
('moment', 'Момент'),
('ten', 'Десять'),
('themselves', 'Самі'),
('gate', 'Ворота'),
('less', 'Менше'),
('men', 'Чоловіки'),
('less', 'Менше'),
('laughing', 'Сміється'),
('radio', 'Радіо'),
('sick', 'Хворий'),
('ever', 'Коли-небудь'),
('letter', 'Лист'),
('pay', 'Платити'),
('mad', 'Божевільний'),
('under', 'Під'),
('twenty', 'Двадцять'),
('shirt', 'Сорочка'),
('writing', 'Написання'),
('evil', 'Зло'),
('snow', 'Сніг'),
('easy', 'Легко'),
('library', 'Бібліотека'),
('really', 'Дійсно'),
('kept', 'Тримали'),
('west', 'Захід'),
('music', 'Музика'),
('life', 'Життя'),
('brought', 'Приніс'),
('whisper', 'Шепіт'),
('paid', 'Платний'),
('world', 'Мир'),
('someone', 'Хтось'),
('today', 'Сегодня'),
('immediately', 'Негайно'),
('same', 'Ж'),
('past', 'Минуле'),
('rally', 'Ралі'),
('tall', 'Високий'),
('sure', 'Упевнений'),
('country', 'Країна'),
('telling', 'Красномовним'),
('action', 'Дія'),
('found', 'Знайдено'),
('pick', 'Вибрати'),
('ahead', 'Уперед'),

('wet', 'Мокрі'),
('being', 'Бути'),
('brother', 'Брате'),
('till', 'До'),
('especially', 'Особливо'),
('many', 'Багато'),
('wrong', 'Неправильно'),
('empty', 'Порожній'),
('purpose', 'Мета'),
('big', 'Великий'),
('sense', 'Почуття'),
('quickly', 'Швидко'),
('enemy', 'Ворог'),
('hostile', 'Вороже'),
('does', 'Робить'),
('hurt', 'Боляче'),
('north', 'Північ'),
('special', 'Спеціальний'),
('every', 'Кожен'),
('cut', 'Вирізати'),
('simply', 'Просто'),
('boss', 'Бос'),
('began', 'Почав'),
('drop', 'Падіння'),
('always', 'Завжди'),
('tire', 'Шина'),
('likely', 'Ймовірно'),
('girl', 'Дівчина'),
('ring', 'Кільце'),
('food', 'Їжа'),
('soul', 'Душа'),
('sitting', 'Сидить'),
('somehow', 'Якось'),
('speaking', 'Мовлення'),
('home', 'Домівка'),
('true', 'Істина'),
('clock', 'Годинники'),
('spirit', 'Дух'),
('without', 'Без'),
('poor', 'Бідних'),
('nearly', 'Майже'),
('boat', 'Човен'),
('heard', 'Чув'),
('order', 'Порядок'),
('bag', 'Мішок'),
('cigarette', 'сигарета'),
('toward', 'До (В напрямку),'),
('towards', 'в направлении'),
('fang', 'Ікло'),
('clothes', 'Одяг'),
('faint', 'Слабкий'),
('need', 'Потреба'),

('south', 'Південь'),
('perfect', 'Досконалий'),
('stop', 'Стоп'),
('ready', 'Готовий'),
('american', 'Американський'),
('enjoy', 'Насолоджуватися'),
('maybe', 'Може бути'),
('unite', "Об'єднати"),
('flat', 'Плоский'),
('part', 'Частина'),
('hot', 'Гарячої'),
('river', 'Річка'),
('forty', 'Сорок'),
('use', 'Використання'),
('speak', 'Говорити'),
('pleasure', 'Задоволення'),
('children', 'Діти'),
('use', 'Використання'),
('fine', 'Відмінно'),
('problem', 'Проблема'),
('simple', 'Простий'),
('though', 'Хоча'),
('given', 'Дано'),
('below', 'Нижче'),
('mostly', 'В основному'),
('far', 'Далеко'),
('able', 'Можливість'),
('break', 'Розбити'),
('hello', 'Привіт'),
('name', "Ім'я"),
('become', 'Стати'),
('couple', 'Пара'),
('self', 'Себе'),
('word', 'Слово'),
('low', 'Низький'),
('report', 'Доповідь'),
('everybody', 'Всі'),
('behind', 'За'),
('person', 'Людина'),
('key', 'Ключ'),
('flash', 'Спалах'),
('try', 'Спробувати'),
('evening', 'Вечір'),
('shape', 'Форма'),
('help', 'Допомога'),
('outside', 'Зовні'),
('within', 'В'),
('shock', 'Шок'),
('also', 'Також'),
('listen', 'Слухати'),
('third', 'Третій'),
('charge', 'Заряд'),

('better', 'Краще'),
('wind', 'Вітер'),
('anyone', 'Хто'),
('danger', 'Небезпека'),
('mean', 'Середній'),
('smell', 'Запах'),
('count', 'Розраховувати'),
('impossible', 'Неможливо'),
('possible', 'Можливо'),
('father', 'Батько'),
('chair', 'Стілець'),
('quick', 'Швидкий'),
('hi', 'Привіт'),
('against', 'Проти'),
('six', 'Шість'),
('seven', 'Сім'),
('million', 'Мільйон'),
('anything', 'Нічого'),
('forward', 'Уперед'),
('plan', 'План'),
('pack', 'Пакет'),
('start', 'Почати'),
('hill', 'Холм'),
('tonight', 'Сегодня'),
('thoracic', 'Грудні'),
('yet', 'Ще'),
('reason', 'Розум'),
('path', 'Шлях'),
('knowing', 'Знаючи'),
('walk', 'Прогулянка'),
('ground', 'Земля'),
('yellow', 'Жовтий'),
('public', 'Громадськість'),
('above', 'Вище'),
('drive', 'Диск'),
('stream', 'Потік'),
('seen', 'Бачив'),
('change', 'зміна'),
('huge', 'Величезний'),
('khan', 'Хан'),
('close', 'Закрити'),
('story', 'Історія'),
('bank', 'Банк'),
('remember', "Пам'ятати"),
('late', 'Пізно'),
('control', 'Контроль'),
('shoe', 'Башмак'),
('car', 'Автомобіль'),
('saying', 'Кажуть'),
('memory', "Пам'ять"),
('eating', 'Їсть'),
('between', 'Між'),

('star', 'Зірка'),
('working', 'Працює'),
('flower', 'Квітка'),
('until', 'До'),
('husband', 'Чоловік'),
('brain', 'Мозок'),
('grass', 'Трава'),
('both', 'Як'),
('size', 'Розмір'),
('history', 'Історія'),
('done', 'зроблений'),
('except', 'Крім'),
('met', 'Зустрілися'),
('prove', 'Довести'),
('god', 'Бог'),
('lord', 'Пан'),
('professor', 'Професор'),
('song', 'Пісня'),
('find', 'Знайти'),
('sorry', 'Вибачте'),
('mine', 'Моє'),
('affair', 'Дело'),
('water', 'Вода'),
('number', 'Кількість'),
('wearing', 'Носити'),
('carefully', 'Обережно'),
('arm', 'Рука'),
('roll', 'Ролл'),
('grew', 'Виріс'),
('natural', 'Природний'),
('few', 'Кілька'),
('longer', 'Більше'),
('animal', 'Тварина'),
('manage', 'Управляти'),
('thick', 'Товстий'),
('hear', 'Чути'),
('rock', 'Рок'),
('glad', 'Радий'),
('marriage', 'Шлюб'),
('joke', 'Жарт'),
('slam', 'Закритті'),
('require', 'Вимагають'),
('spent', 'Провів'),
('opening', 'Відкриття'),
('energy', 'Енергія'),
('thrust', 'Тяга'),
('art', 'Мистецтво'),
('sake', 'Саке'),
('feature', 'Особливість'),
('whip', 'Кнут'),
('easily', 'Легко'),
('sleeping', 'Спати'),

('happening', 'Відбувається'),
('approaching', 'Наближається'),
('listening', 'Прослуховування'),
('tight', 'Жорсткою'),
('horrible', 'Жахливий'),
('authority', 'Влада'),
('coffee', 'Кава'),
('fresh','Свіжий'),
('member', 'Член'),
('drown', 'Втопити'),
('condition', 'Стан'),
('fun', 'Весело'),
('swung', 'Замахнувся'),
('exclaim', 'Вигукнути'),
('meeting', 'Зустріч'),
('struggle', 'Боротьба'),
('christmas', 'Різдво'),
('xmas', 'Різдво'),
('influence', 'Вплив'),
('breakfast', 'Сніданок'),
('tone', 'Тон'),
('drawing', 'Малюнок'),
('insist', 'Наполягають'),
('frighten', 'Налякати'),
('grown', 'Дорослим'),
('favorite', 'Улюблений'),
('patient', 'Пацієнт'),
('officer', 'Офіцер'),
('garage', 'Гараж'),
('per', 'За'),
('beer', 'Пиво'),
('march', 'Марш'),
('meaning', 'Значення'),
('bullet', 'Куля'),
('ghost', 'Привид'),
('spread', 'Поширення'),
('quit', 'Кинути палити'),
('chicken', 'Курка'),
('grow', 'Рости'),
('stranger', 'Незнайомець'),
('series', 'Серія'),
('counter', 'Лічильник'),
('heat', 'Тепло'),
('bush', 'Кущ'),
('bower', 'Альтанка'),
('engage', 'Займатися'),
('serious', 'Серйозні'),
('load', 'Навантаження'),
('drunk', "П'яний"),
('hide', 'Приховати'),
('fallen', 'Упав'),
('odd', 'Непарний'),

('farmer', 'Фермер'),
('killing', 'Вбивство'),
('therefore', 'Тому'),
('recognize', 'Визнати'),
('happiness', 'Щастя'),
('twelve', 'Дванадцять'),
('shoot', 'Стріляти'),
('rear', 'Ззаду'),
('elbow', 'Локоть'),
('grip', 'Захоплення'),
('search', 'Пошук'),
('ourselves', 'Себе'),
('message', 'Повідомлення'),
('mirror', 'Дзеркало'),
('thrown', 'Викинутий'),
('pony', 'Конячка'),
('necessary', 'Необхідно'),
('horror', 'Жах'),
('closet', 'Шафа'),
('rage', 'Лють'),
('particular', 'Зокрема'),
('fifteen', "П'ятнадцять"),
('determine', 'Визначити'),
('tunnel', 'Тунель'),
('stupid', 'Дурний'),
('inch', 'Дюйм'),
('missing', 'Відсутня'),
('somewhat', 'Кілька'),
('fourth', 'Четверта'),
('sing', 'Співати'),
('oil', 'Масло'),
('stir', 'Переполох'),
('bent', 'Згинатися'),
('stage', 'Етап'),
('proper', 'Правильний'),
('test', 'Тест'),
('crying', 'Плачу'),
('stretch', 'Розтягнути'),
('charity', 'Благодійність'),
('thigh', 'Стегно'),
('final', 'Остаточний'),
('weight', 'Вага'),
('honey', 'Мед'),
('wedding', 'Весілля'),
('martian', 'Марсіанський'),
('broad', 'Широкий'),
('stair', 'Сходинка'),
('mental', 'Психічне'),
('nightmare', 'Кошмар'),
('avenue', 'Проспект'),
('area', 'Площа'),
('rate', 'Швидкість'),

('lamp', 'Лампа'),
('bike', 'Велосипед'),
('sigh', 'Зітхнути'),
('awhile', 'Деякий час'),
('mix', 'Суміш'),
('cylinder', 'Циліндр'),
('prison', 'Тюрма'),
('opposite', 'Навпаки'),
('bow', 'Лук'),
('puzzle', 'Головоломка'),
('peculiar', 'Своєрідний'),
('somebody', 'Хтось'),
('church', 'Церква'),
('ruin', 'Розруха'),
('sunset', 'Захід'),
('throw', 'Кинути'),
('court', 'Суд'),
('smooth', 'Згладжувати'),
('bread', 'Хліб'),
('castle', 'Замок'),
('destroy', 'Знищити'),
('excellent', 'Відмінно'),
('gut', 'Потрошити'),
('fade', 'Завмирання'),
('handler', 'обробник'),
('pity', 'Шкода'),
('storm', 'Буря'),
('lean', 'Пісне'),
('glow', 'Світіння'),
('prevent', 'Запобігти'),
('aside', 'В сторону'),
('orange', 'Помаранчевий'),
('hungry', 'Голодний'),
('sank', 'Потонув'),
('delight', 'Радість'),
('belong', 'Належить'),
('iron', 'Залізо'),
('succeed', 'Успіху'),
('lose', 'Програють'),
('spend', 'Провести'),
('particularly', 'Особливо'),
('temper', 'Характер'),
('match', 'Відповідає'),
('difficult', 'Важко'),
('print', 'Друк'),
('tough', 'Жорстким'),
('dust', 'Пил'),
('unable', 'Нездатний'),
('nobody', 'Ніхто'),
('quietly', 'Тихо'),
('fail', 'Провал'),
('shift', 'Зрушення'),

('clown', 'Клоун'),
('judge', 'Суддя'),
('advance', 'Просування'),
('dwarf', 'Карлик'),
('development', 'розробка'),
('developer', 'Розробник'),
('effect', 'Ефект'),
('shake', 'Трясти'),
('bother', 'Турбувати'),
('entrance', 'Вхід'),
('future', 'Майбутнє'),
('angle', 'Кут'),
('chamber', 'Камера'),
('evidently', 'Очевидно'),
('object', "Об'єкт"),
('buddy', 'Приятель'),
('habit', 'Звичка'),
('intend', 'Маємо намір'),
('respect', 'повага'),
('circumstance', 'Обставина'),
('bedroom', 'Спальня'),
('convince', 'переконати'),
('christ', 'Христос'),
('lap', 'коліна'),
('vanish', 'Зникнути'),
('fish', 'Риба'),
('patch', 'Патч'),
('willing', 'бажаю'),
('dry', 'сухий'),
('level', 'Рівень'),
('society', 'суспільство'),
('better', 'Краще'),
('ice', 'лід'),
('remove', 'видалити'),
('speech', 'мова'),
('eaten', 'Їжте'),
('dance', 'танець'),
('weak', 'Слабкі'),
('weakness', 'Слабкість'),
('weekness', 'тижні'),
('bite', 'Укус'),
('nurse', 'Медсестра'),
('forgotten', 'Забутий'),
('cop', 'Коп'),
('english', 'Англійська'),
('photograph', 'Фотографія'),
('lover', 'коханець'),
('jacket', 'Куртка'),
('language', 'Мова'),
('spell', 'Заклинання'),
('occur', 'відбуваються'),
('servant', 'слуга'),

```
('sixty', 'Шістдесят'),
('warning', 'Попередження'),
('result', 'Результат'),
('sheet', 'Лист'),
('support', 'Підтримка'),
('balloon', 'куля'),
('share', 'Частка'),
('shop', 'Магазин'),
('arrange', 'влаштувати'),
('suit', 'Костюм'),
('weather', 'Погода'),
('battle', 'битва'),
('beg', 'просити'),
('surface', 'поверхня'),
('afterwards', 'пізніше'),
('section', 'Розділ'),
('birthday', 'день народження'),
('attack', 'Напад'),
('powerful', 'Потужний'),
('tiny', 'крихітний'),
('blame', 'Брити'),
('ancient', 'Давній'),
('embrace', 'Обійми'),
('ill', 'хворі'),
('queen', 'Королева'),
('bridge', 'Міст'),
('stomach', 'шлунок'),
('beating', 'биття'),
('lack', 'відсутність'),
('movement', 'рух'),
('keeping', 'Утримання'),
('panic', 'Паніка'),
('leaning', 'нахиляючись'),
('rule', 'Правило'),
('lovely', 'чудовий'),
('passage', 'прохід '),
('wake', 'прокидатися'),
('trick', 'трюк'),
('waver', 'хвиля'),
('limit', 'обмеження'),
('record', 'рекорд'),
('using', 'використання'),
('snap', 'зачіпати'),
('yell', 'кричати'),
('compass', 'Компас'),
('banana', 'банан');
```

Файл “src/backend/templates/index.html”

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>API Service is available</title>
</head>
<body>
    API Service is available!
</body>
</html>
```

Файл “src/backend/services/irregular_verbs_service.py”

```
from typing import List
from sqlalchemy.orm import Session
from database.connector import DatabaseConnector
from models.irregular_verb import IrregularVerb

class IrregularVerbService:
    connector = DatabaseConnector()
    def get_irregular_verbs(self) -> List[IrregularVerb]:
        with Session(bind=self.connector.engine) as session:
            return session.query(IrregularVerb).all()
```

Файл “src/backend/services/user_accounts_service.py”

```
from typing import Optional
from sqlalchemy.orm import Session
from database.connector import DatabaseConnector
from models.user_account import UserAccount
class UserAccountService:
    connector = DatabaseConnector()
    def get_user_account(self, username: str, password: str) -> Optional[UserAccount]:
        with Session(bind=self.connector.engine) as session:
            user_account: Optional[UserAccount] = (
                session.query(UserAccount)
                .filter(UserAccount.username == username)
                .first()
            )
            if user_account and user_account.check_password_correction(password):
                return user_account
    def create_user_account(self, username: str, password: str) -> None:
        with Session(bind=self.connector.engine) as session:
            user_account = UserAccount(
                username=username,
                password=password,
            )
            session.add(user_account)
            session.commit()
    def update_user_account_personal_information(
        self,
        username: str,
        password: str,
        first_name: str,
```

```
        last_name: str,
        gender: str,
        email_address: str,
        birthday_str: str,
    ) -> bool:
        with Session(bind=self.connector.engine) as session:
            user_account: Optional[UserAccount] = self.get_user_account(
                username=username, password=password
            )
            if user_account:
                user_account.username = username
                user_account.first_name = first_name
                user_account.last_name = last_name
                user_account.gender = gender
                user_account.email_address = email_address
                user_account.birthday_str = birthday_str
                session.add(user_account)
                session.commit()
                return True
            return False

    def update_user_account_password(
        self, username: str, password: str, new_password: str
    ) -> bool:
        with Session(bind=self.connector.engine) as session:
            user_account: Optional[UserAccount] = self.get_user_account(
                username=username, password=password
            )
            if user_account:
                user_account.password = new_password
                session.add(user_account)
                session.commit()
                return True
            return False
    def delete_user_account(self, username: str, password: str) -> bool:
        with Session(bind=self.connector.engine) as session:
            user_account: Optional[UserAccount] = self.get_user_account(
                username=username, password=password
            )
            if user_account:
                session.delete(user_account)
                session.commit()
                return True
            return False
    def does_the_user_exists(self, username: str) -> bool:
        with Session(bind=self.connector.engine) as session:
            user = session.query(UserAccount).filter_by(username=username).first()
            return user is not None
```

Файл “src/backend/services/words_service.py”

```
from typing import List, Optional
```

```
from sqlalchemy.orm import Session
from database.connector import DatabaseConnector
from models.word import Word
class WordsService:
    connector = DatabaseConnector()
    def get_words(self, limit: int | str = None) -> List[Word]:
        with Session(bind=self.connector.engine) as session:
            if limit:
                return session.query(Word).limit(limit).all()
            else:
                return session.query(Word).all()

    def get_word_by_origin(self, origin: str) -> Optional[Word]:
        with Session(bind=self.connector.engine) as session:
            return session.query(Word).filter(Word.origin == origin).first()
```

Файл “src/backend/resources/irregular_verbs_resource.py”

```
from typing import List
from flask import jsonify, make_response, Response
from flask_restful import Resource
from services.irregular_verbs_service import IrregularVerbService
from models.irregular_verb import IrregularVerb
class IrregularVerbsResource(Resource):
    def __init__(self):
        self.irregular_verb_service = IrregularVerbService()
    def get(self) -> Response:
        irregular_verbs: List[IrregularVerb] = (
            self.irregular_verb_service.get_irregular_verbs()
        )
        irregular_verbs_dicts: List[dict] = [
            irregular_verb.to_dict() for irregular_verb in irregular_verbs
        ]
        return make_response(jsonify(irregular_verbs_dicts), 200)
```

Файл “src/backend/resources/login_resource.py”

```
from typing import Optional
from flask import jsonify, make_response, Response, request
from flask_restful import Resource
from services.user_accounts_service import UserAccountService
from models.user_account import UserAccount
class LoginResource(Resource):
    methods = ["GET", "POST"]
    def __init__(self):
        self.user_account_service = UserAccountService()
    def post(self) -> Response:
        username = request.json.get("username", None)
        password = request.json.get("password", None)
        return self.login_user(username, password)
    def login_user(self, username, password) -> Response:
        if not username or not password:
```

```
            return make_response(jsonify({"message": "Username and password is required!"}), 400)
        user_account: Optional[UserAccount] = (
            self.user_account_service.get_user_account(username=username, password=password)
        )
        if user_account:
            return make_response(jsonify(user_account.to_dict()), 200)
        else:
            return make_response(jsonify({"message": "User not found or password is incorrect!"}), 400)
```

Файл “src/backend/resources/user_accounts_resource.py”

```
from typing import Optional
from flask import jsonify, make_response, Response, request
from flask_restful import Resource
from services.user_accounts_service import UserAccountService
from models.user_account import UserAccount
from resources.login_resource import LoginResource
class UserAccountResource(Resource):
    methods = ["POST", "PATCH", "PUT", "DELETE"]
    def __init__(self):
        self.user_account_service = UserAccountService()
        self.login_resource = LoginResource()
    def post(self):
        username = request.json.get("username")
        password = request.json.get("password")
        # if username already use?!
        if self.user_account_service.does_the_user_exists(username=username):
            return make_response(
                jsonify({"message": "A user with the same username already exists!"}),
                401,
            )
        if username and password:
            self.user_account_service.create_user_account(
                username=username, password=password
            )
            return self.login_resource.login_user(
                username=username,
                password=password,
            )
        else:
            return make_response(jsonify({"message": "Not enough data!"}), 401)
    def patch(self):
        username = request.json.get("username")
        password = request.json.get("password")
        new_password = request.json.get("newPassword")
        if username and password and new_password:
            is_updated = self.user_account_service.update_user_account_password(
                username=username, password=password, new_password=new_password
            )
            if is_updated:
                return make_response(jsonify({"message": "Success!"}), 200)
```

```
            else:
                return make_response(jsonify({"message": "Wrong Password"}), 401)
        else:
            return make_response(jsonify({"message": "Not enough data!"}), 400)
    def put(self):
        username = request.json.get("username")
        password = request.json.get("password")
        first_name = request.json.get("firstName")
        last_name = request.json.get("lastName")
        gender = request.json.get("gender")
        email_address = request.json.get("emailAddress")
        birthday_str = request.json.get("birthday")
        if username and password:
            is_updated = (
                self.user_account_service.update_user_account_personal_information(
                    username=username,
                    password=password,
                    first_name=first_name,
                    last_name=last_name,
                    gender=gender,
                    email_address=email_address,
                    birthday_str=birthday_str,
                )
            )
            if is_updated:
                return make_response(jsonify({"message": "Success!"}), 200)
            else:
                return make_response(jsonify({"message": "Wrong Password"}), 400)
        else:
            return make_response(jsonify({"message": "Not enough data!"}), 400)
    def delete(self):
        username = request.json.get("username")
        password = request.json.get("password")

        if username and password:
            is_deleted = self.user_account_service.delete_user_account(
                username=username, password=password
            )
            if is_deleted:
                return make_response(jsonify({"message": "Success!"}), 200)
            else:
                return make_response(jsonify({"message": "Wrong Password"}), 401)
        else:
            return make_response(jsonify({"message": "Not enough data!"}), 400)
```

Файл “src/backend/resources/words_resource.py”

```
from typing import List, Optional
from flask import jsonify, make_response, Response, request
from flask_restful import Resource
from utils.words_util import clear_origin
```

```
from services.words_service import WordsService
from models.word import Word
class WordsResource(Resource):
    def __init__(self):
        self.words_service = WordsService()
    def get(self) -> Response:
        limit = int(request.args.get("limit", "0"))
        origin: Optional[str] = clear_origin(request.args.get("origin"))
        if origin:
            word: Optional[Word] = self.words_service.get_word_by_origin(origin=origin)
            if word:
                return make_response(jsonify(word.to_dict()), 200)
            else:
                return make_response(jsonify({"detail": "Sorry, word not found!"}))

        words: List[Word] = (
            self.words_service.get_words(limit)
        )
        words_dicts: List[dict] = [
            word.to_dict() for word in words
        ]
        return make_response(jsonify(words_dicts), 200)
```

Файл “src/backend/models/base.py”

```
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column
class Base(DeclarativeBase):
    id: Mapped[int] = mapped_column(primary_key=True, autoincrement=True)
```

Файл “src/backend/models/irregular_verb.py”

```
from sqlalchemy import String
from sqlalchemy.orm import Mapped, mapped_column
from models.base import Base
class IrregularVerb(Base):
    __tablename__ = "irregular_verbs"
    base_form: Mapped[str] = mapped_column(String(length=45), unique=True)
    past_tense: Mapped[str] = mapped_column(String(length=45))
    past_participle: Mapped[str] = mapped_column(String(length=45))
    def to_dict(self) -> dict:
        return {
            "_id": self.id,
            "baseForm": self.base_form,
            "pastTense": self.past_tense,
            "pastParticiple": self.past_participle,
        }
```

Файл “src/backend/models/user_account.py”

```
from typing import Optional
```

```
from datetime import date
from sqlalchemy.orm import Mapped, mapped_column
from sqlalchemy import String, Column, Enum
from utils.date_utils import date_to_str, str_to_date
from models.base import Base
from enums.gender import Gender
class UserAccount(Base):
    __tablename__ = "user_accounts"
    username: Mapped[str] = mapped_column(String(length=64), unique=True)
    first_name: Mapped[Optional[str]] = mapped_column(String(length=64))
    last_name: Mapped[Optional[str]] = mapped_column(String(length=64))
    password_hash: Mapped[str]
    email_address: Mapped[Optional[str]] = mapped_column(String(length=319))
    gender = Column(Enum(Gender), default=Gender.UNDEFINED)
    birthday: Mapped[Optional[date]]
    @property
    def password(self) -> str:
        return self.password_hash
    @password.setter
    def password(self, plain_text_password) -> None:
        from app import bcrypt
        self.password_hash = bcrypt.generate_password_hash(plain_text_password).decode(
            "utf-8"
        )
    def check_password_correction(self, attempted_password) -> bool:
        from app import bcrypt
        return bcrypt.check_password_hash(self.password_hash, attempted_password)
    @property
    def birthday_str(self):
        return date_to_str(self.birthday)
    @birthday_str.setter
    def birthday_str(self, value):
        self.birthday = str_to_date(value)
    def to_dict(self) -> dict:
        return {
            "_id": self.id,
            "username": self.username,
            "firstName": self.first_name,
            "lastName": self.last_name,
            "emailAddress": self.email_address,
            "gender": self.gender.name,
            "birthday": date_to_str(self.birthday),
        }
```

Файл “src/backend/models/word.py”

```
from sqlalchemy import String
from sqlalchemy.orm import Mapped, mapped_column
from models.base import Base
class Word(Base):
    __tablename__ = "words"
```

```
origin: Mapped[str] = mapped_column(String(length=45), unique=False)
translation: Mapped[str] = mapped_column(String(length=45))
def to_dict(self) -> dict:
    return {
        "_id": self.id,
        "origin": self.origin,
        "translation": self.translation,
    }
```

Файл “src/backend/enums/gender.py”

```
from enum import Enum
class Gender(Enum):
    MALE = "Male"
    FEMALE = "Female"
    UNDEFINED = "Undefined"
```

Файл “src/backend/database/connector.py”

```
from os import getenv
from dotenv import load_dotenv
from sqlalchemy import Engine, create_engine
from models.base import Base
from models.irregular_verb import IrregularVerb
from models.word import Word
from models.user_account import UserAccount
load_dotenv()
class DatabaseConnector:
    _engine: Engine = create_engine(getenv("DB_URL"), echo=True)
    def __init__(self):
        Base.metadata.create_all(self._engine)
    @property
    def engine(self):
        return self._engine
```

Файл “src/frontend/index.html”

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <link rel="icon" href="/favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css"
integrity="sha512-
SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFcV7oQPJkl9Q
evSCWr3W6A==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <title>English Chest</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.ts"></script>
  </body></html>
```

Файл “src/frontend/package.json”

```
{
  "name": "english-chest-frontend",
  "version": "0.0.0",
  "private": true,
  "type": "module",
```

```
  "scripts": {
    "dev": "vite",
    "build": "run-p type-check \"build-only {@}\" --",
    "preview": "vite preview",
    "test:unit": "vitest",
    "build-only": "vite build",
    "type-check": "vue-tsc --build --force",
    "lint": "eslint . --ext .vue,.js,.jsx,.cjs,.mjs,.ts,.tsx,.cts,.mts --fix --ignore-path .gitignore",
    "format": "prettier --write src/"
  },
  "dependencies": {
    "vue": "^3.4.21",
    "vue-router": "^4.3.0",
    "vuex": "^4.1.0"
  },
  "devDependencies": {
    "@rushstack/eslint-patch": "^1.8.0",
    "@tsconfig/node20": "^20.1.4",
    "@types/jsdom": "^21.1.6",
    "@types/node": "^20.12.5",
    "@vitejs/plugin-vue": "^5.0.4",
    "@vue/eslint-config-prettier": "^9.0.0",
    "@vue/eslint-config-typescript": "^13.0.0",
    "@vue/test-utils": "^2.4.5",
    "@vue/tsconfig": "^0.5.1",
    "eslint": "^8.57.0",
    "eslint-plugin-vue": "^9.23.0",
    "jsdom": "^24.0.0",
    "npm-run-all2": "^6.1.2",
    "prettier": "^3.2.5",
    "sass": "^1.75.0",
    "typescript": "~5.4.0",
    "vite": "^5.2.8",
    "vitest": "^1.4.0",
    "vue-tsc": "^2.0.11"
  }
}
```

Файл “src/frontend/src/main.ts”

```
import "./static/styles/css/styles.css"
import "./static/styles/css/fonts.css"
import { createApp } from "vue"
import App from "./App.vue"
import router from "./router"
import store from "./data/storage.ts"
const app = createApp(App)
app.use(router)
app.use(store)
app.mount("#app")
```

Файл “src/frontend/src/App.vue”

```
<script lang="ts" setup>
import { RouterView } from "vue-router"
import AppHeader from "@/components/AppHeader.vue"
```

```
import AppFooter from "@/components/AppFooter.vue"
import SideBar from "@/components/SideBar.vue"
import FlashMessage from './components/FlashMessage.vue'
</script>
<style scoped></style>
<template>
  <div class="grid-container">
    <app-header />
    <side-bar />
    <router-view />
    <flash-message v-if="getFlashMessage" :message="getFlashMessage.message"
:className="getFlashMessage.className"/>
    <!-- <app-footer /> -->
  </div>
</template>
<script lang="ts">
import { mapActions, mapGetters } from "vuex"
export default {
  name: "EnglishChest",
  computed: { ...mapGetters(["getDomain", "getFlashMessage"]) },
  methods: { ...mapActions(["fetchIrregularVerbs", "fetchWords"]) },
  async mounted() {
    await this.fetchIrregularVerbs()
    await this.fetchWords()
  }
}
</script>
```

Файл “src/frontend/src/router/index.ts”

```
import { createRouter, createWebHistory } from "vue-router"
import HomeView from "../views/HomeView.vue"
import AboutView from "../views/AboutView.vue"
import PrivacyPolicyView from "../views/PrivacyPolicyView.vue"
import AlphabetView from "../views/AlphabetView.vue"
import ArticlesView from "../views/ArticlesView.vue"
import ComingSoonView from "../views/ComingSoonView.vue"
import ConditionalSentencesView from "../views/ConditionalSentencesView.vue"
import DegreesOfComparisonView from "../views/DegreesOfComparisonView.vue"
import DictionaryView from "../views/DictionaryView.vue"
import FutureInThePastView from "../views/FutureInThePastView.vue"
import InfinitiveView from "../views/InfinitiveView.vue"
import InOrderToView from "../views/InOrderToView.vue"
import LevelsView from "../views/LevelsView.vue"
import ModalVerbsView from "../views/ModalVerbsView.vue"
import NumbersView from "../views/NumbersView.vue"
import ProgressivePerfectTenseView from "../views/PerfectProgressiveTenseView.vue"
import PossessiveFormView from "../views/PossessiveFormView.vue"
import SimpleTenseView from "../views/SimpleTenseView.vue"
import ProgressiveTenseView from "../views/ProgressiveTenseView.vue"
import PerfectTenseView from "../views/PerfectTenseView.vue"
import PersonalPronounsView from "../views/PersonalPronounsView.vue"
import PluralView from "../views/PluralView.vue"
import ReflexivePronounsView from "../views/ReflexivePronounsView.vue"
import TheConditionalMoodView from "../views/TheConditionalMoodView.vue"
import TheGerundView from "../views/TheGerundView.vue"
import ToBeAbleView from "../views/ToBeAbleView.vue"
import UsedToView from "../views/UsedToView.vue"
import PrepositionsView from "../views/PrepositionsView.vue"
import IrregularVerbsView from "../views/IrregularVerbsView.vue"
import LoginView from '../views/LoginView.vue'
import RegisterView from '../views/RegisterView.vue'
import ProfileView from '../views/ProfileView.vue'
import GreetingsView from '../views/GreetingsView.vue'
import SocialiteTalkView from '../views/SocialiteTalkView.vue'
import GoodbyesView from '../views/GoodbyesView.vue'
import IntroductionYourselfView from '../views/IntroductionYourselfView.vue'
import GerundAfterPrepositionsExerciseView from
'../views/tasks/GerundAfterPrepositionsExerciseView.vue'
import GerundAndInfinitiveExerciseView from '../views/tasks/GerundAndInfinitiveExerciseView.vue'
import NounsArticlesExerciseView from '../views/tasks/NounsArticlesExerciseView.vue'
import ConditionalSentencesExerciseView from '../views/tasks/ConditionalSentencesExerciseView.vue'
import SimpleTenseExerciseView from '../views/tasks/SimpleTenseExerciseView.vue'
import ProgressiveTenseExerciseView from '../views/tasks/ProgressiveTenseExerciseView.vue'
import PerfectTenseExerciseView from '../views/tasks/PerfectTenseExerciseView.vue'
import PerfectProgressiveTenseExerciseView from
'../views/tasks/PerfectProgressiveTenseExerciseView.vue'
import FuturePerfectExerciseView from '../views/tasks/FuturePerfectExerciseView.vue'
const router = createRouter({
   history: createWebHistory(import.meta.env.BASE_URL),
   routes: [
      {
         path: "/",
         name: "home",
```

```
    component: HomeView,
  },
  {
    path: "/about",
    name: "about",
    component: AboutView,
  },
  {
    path: "/alphabet",
    name: "alphabet",
    component: AlphabetView,
  },
  {
    path: "/articles",
    name: "articles",
    component: ArticlesView,
  },
  {
    path: "/coming-soon",
    name: "coming-soon",
    component: ComingSoonView,
  },
  {
    path: "/conditional-sentences",
    name: "conditional-sentences",
    component: ConditionalSentencesView,
  },
  {
    path: "/degrees-of-comparison",
    name: "DegreesOfComparisonView",
    component: DegreesOfComparisonView,
  },
  {
    path: "/dictionary",
    name: "dictionary",
    component: DictionaryView,
  },
  {
    path: "/future-in-the-past",
    name: "future-in-the-past",
    component: FutureInThePastView,
  },
  {
    path: "/infinitive",
    name: "infinitive",
    component: InfinitiveView,
  },
  {
    path: "/in-order-to",
    name: "in-order-to",
    component: InOrderToView,
  },
  {
    path: "/levels",
    name: "levels",
    component: LevelsView,
```

```
},
{
  path: "/modal-verbs",
  name: "modal-verbs",
  component: ModalVerbsView,
},
{
  path: "/numbers",
  name: "numbers",
  component: NumbersView,
},
{
  path: "/progressive-perfect-tense",
  name: "progressive-perfect-tense",
  component: ProgressivePerfectTenseView,
},
{
  path: "/possessive-form",
  name: "possessive-form",
  component: PossessiveFormView,
},
{
  path: "/simple-tense",
  name: "simple-tense",
  component: SimpleTenseView,
},
{
  path: "/progressive-tense",
  name: "progressive-tense",
  component: ProgressiveTenseView,
},
{
  path: "/perfect-tense",
  name: "perfect-tense",
  component: PerfectTenseView,
},
{
  path: "/personal-pronouns",
  name: "personal-pronouns",
  component: PersonalPronounsView,
},
{
  path: "/plural",
  name: "plural",
  component: PluralView,
},
{
  path: "/reflexive-pronouns",
  name: "reflexive-pronouns",
  component: ReflexivePronounsView,
},
{
  path: "/the-conditional-mood",
  name: "the-conditional-mood",
  component: TheConditionalMoodView,
},
```

```
{
  path: "/the-gerund",
  name: "the-gerund",
  component: TheGerundView,
},
{
  path: "/to-be-able",
  name: "to-be-able",
  component: ToBeAbleView,
},
{
  path: "/used-to",
  name: "used-to",
  component: UsedToView,
},
{
  path: "/prepositions",
  name: "prepositions",
  component: PrepositionsView,
},
{
  path: "/privacy-policy",
  name: "privacy-policy",
  component: PrivacyPolicyView,
},
{
  path: "/irregular-verbs",
  name: "irregular-verbs",
  component: IrregularVerbsView,
},
{
  path: "/register",
  name: "register",
  component: RegisterView,
},
{
  path: "/login",
  name: "login",
  component: LoginView,
},
{
  path: "/profile",
  name: "profile",
  component: ProfileView,
},
{
  path: "/greetings",
  name: "greetings",
  component: GreetingsView,
},
{
  path: "/goodbyes",
  name: "goodbyes",
  component: GoodbyesView,
},
{
```

```
      path: "/introduction-yourself",
      name: "introduction-yourself",
      component: IntroductionYourselfView,
    },
    {
      path: "/socialite-talk",
      name: "socialite-talk",
      component: SocialiteTalkView,
    },
    {
      path: "/gerund-after-prepositions-exercise",
      name: "gerund-after-prepositions-exercise",
      component: GerundAfterPrepositionsExerciseView,
    },
    {
      path: "/gerund-and-infinitive-exercise",
      name: "gerund-and-infinitive-exercise",
      component: GerundAndInfinitiveExerciseView,
    },
    {
      path: "/nouns-articles-exercise",
      name: "nouns-articles-exercise",
      component: NounsArticlesExerciseView,
    },
    {
      path: "/conditional-sentences-exercise",
      name: "conditional-sentences-exercise",
      component: ConditionalSentencesExerciseView,
    },
    {
      path: "/simple-tense-exercise",
      name: "simple-tense-exercise",
      component: SimpleTenseExerciseView,
    },
    {
      path: "/progressive-tense-exercise",
      name: "progressive-tense-exercise",
      component: ProgressiveTenseExerciseView,
    },
    {
      path: "/perfect-tense-exercise",
      name: "perfect-tense-exercise",
      component: PerfectTenseExerciseView,
    },
    {
      path: "/progressive-perfect-tense-exercise",
      name: "progressive-perfect-tense-exercise",
      component: PerfectProgressiveTenseExerciseView,
    },
    {
      path: "/future-perfect-exercise",
      name: "future-perfect-exercise",
      component: FuturePerfectExerciseView,
    },
  ],
})
```

```
export default router
```

Файл “src/frontend/src/data/storage.ts”

```
import { createStore } from "vuex"
import irregularVerbs from "./modules/irregular-verbs.ts"
import State from "./modules/interfaces/State.ts"
import words from "./modules/words.ts"
import userAccount from './modules/userAccount.ts'
import SideBarStates from './enums/SideBarStates.ts'
import Context from './modules/interfaces/Context.ts'
import FlashMessage from './modules/interfaces/FlashMessage.ts'
export default createStore({
   state: {
      domain: "http://127.0.0.1:5000",
      sideBarState: SideBarStates.Basics,
      flashMessage: null,
      canCallFlashMessage: true
   },
   getters: {
      getDomain(state: State) {
         return state.domain
      },
      getSideBarState(state: State): SideBarStates {
         return state.sideBarState
      },
      getFlashMessage(state: State): FlashMessage | null {
         return state.flashMessage
      }
   },
   mutations: {
      updateSideBarState(state: State, newSideBarState: SideBarStates) {
         state.sideBarState = newSideBarState
      },
      updateFlashMessage(state: State, flashMessage: FlashMessage): void {
         if (state.canCallFlashMessage) {
            state.flashMessage = null
            setTimeout(() => {
               state.flashMessage = flashMessage
               // FlashMessage Reload
               state.canCallFlashMessage = false
               setTimeout(() => {
                  state.canCallFlashMessage = true
               }, 5000)
            }, 10);
         }
      }
   },
   modules: {
      irregularVerbs,
      words,
      userAccount,
   },
})
```

Файл “src/frontend/src/data/modules/irregular-verbs.ts”

```
import Context from "./interfaces/Context"
import State from "./interfaces/State"
interface IrregularVerb {
   _id: number
   baseForm: string
   pastTense: string
   pastParticiple: string
}
export default {
   state: {
      irregularVerbs: new Array<IrregularVerb>(),
   },
   getters: {
      getIrregularVerbs(state: State) {
         return state.irregularVerbs
      },
   },
   mutations: {
      updateIrregularVerbs(state: State, irregularVerbs: Array<IrregularVerb>) {
         state.irregularVerbs = irregularVerbs
      },
   },
   actions: {
      fetchIrregularVerbs(context: Context) {
         const domain = context.rootState.domain
         fetch(domain + "/api/irregular-verbs")
            .then((response) => response.json())
            .then((value) => {
               context.commit("updateIrregularVerbs", value)
            })
            .catch((error) => console.error(error))
      },
      getVerbInV2(context: Context, baseForm: string): string {
         const irregularVerbs: Array<IrregularVerb> =
            context.rootState.irregularVerbs.irregularVerbs
         const foundVerb = irregularVerbs.find((irregularVerb: IrregularVerb) => {
            return irregularVerb.baseForm === baseForm
         })
         if (foundVerb) {
            return foundVerb.pastTense
         }
         return baseForm + "ed"
      },
      getVerbInV3(context: Context, baseForm: string): string {
         const irregularVerbs: Array<IrregularVerb> =
            context.rootState.irregularVerbs.irregularVerbs
         const foundVerb = irregularVerbs.find((irregularVerb: IrregularVerb) => {
            return irregularVerb.baseForm === baseForm
         })
         if (foundVerb) {
            return foundVerb.pastParticiple
         }
         return baseForm + "ed"
      },
   },
}
```

Файл “src/frontend/src/data/modules/userAccount.ts”

```
import State from "./interfaces/State"
import Context from "./interfaces/Context"
import UserAccount from "./interfaces/UserAccount"
import AccountData from "./interfaces/AccountData"
import { mapMutations } from "vuex"
const defaultUserAccount: UserAccount = {
   _id: 0,
   username: "[your username]",
   firstName: "[your first name]",
   lastName: "[your last name]",
   emailAddress: "[your email address]",
   gender: "[your gender]",
   birthday: "[your birthday]",
}
export default {
   state: {
      userAccount: defaultUserAccount,
      isAuth: false,
   },
   getters: {
      getUserAccount(state: State): UserAccount {
         return state.userAccount
      },
      getIsAuth(state: State): boolean {
         return state.isAuth
      },
      getAge(state: State) {
         const birth = new Date(state.userAccount.birthday);
         const today = new Date();


         let age = today.getFullYear() - birth.getFullYear();


         const monthDifference = today.getMonth() - birth.getMonth();
         if (monthDifference < 0 || (monthDifference === 0 && today.getDate() < birth.getDate())) {
            age--;
         }

         return age;
      }
   },
   mutations: {
      ...mapMutations(["updateFlashMessage"]),
      updateUserAccount(state: State, userAccount: UserAccount): void {
         state.userAccount = userAccount
         if (!userAccount.message) {
            state.isAuth = true
         }
      },
      resetUserAccount(state: State): void {
         state.userAccount = defaultUserAccount
         state.isAuth = false
```

```
    },
},
actions: {
    async fetchLoginUser(context: Context, data: AccountData): Promise<void> {
        const domain = context.rootState.domain
        await fetch(domain + "/api/login", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify(data),
        })
            .then((response) => {
                return response.json()
            })
            .then((value) => {
                if (value.message) {
                    context.commit("updateFlashMessage", {
                        className: "red",
                        message: value.message,
                    })
                } else {
                    context.commit("updateUserAccount", value)
                    context.commit("updateFlashMessage", {
                        className: "green",
                        message: "Successful login!",
                    })
                }
            })
            .catch((error) => {
                console.error(error)
            })
    },
    async fetchRegisterUser(context: Context, data: AccountData): Promise<void> {
        const domain = context.rootState.domain
        await fetch(domain + "/api/users", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify(data),
        })
            .then((response) => {
                return response.json()
            })
            .then((value) => {
                if (value.message) {
                    context.commit("updateFlashMessage", {
                        className: "red",
                        message: value.message,
                    })
                } else {
                    context.commit("updateUserAccount", value)
                    context.commit("updateFlashMessage", {
                        className: "green",
                        message: "Successful register!",
```

```
            })
          }
        })
        .catch((error) => {
          console.error(error)
        })
  },
  async fetchUpdateUserPersonalInformation(
    context: Context,
    newUserAccountInformation: UserAccount,
  ): Promise<void> {
    const domain = context.rootState.domain
    await fetch(domain + "/api/users", {
      method: "PUT",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(newUserAccountInformation),
    })
      .then((response) => {
        return response.json()
      })
      .then((value) => {
        let className: string;
        if (value.message.includes("Success")) {
          className = "green"
        } else {
          className = "red"
        }
        context.commit("updateFlashMessage", {
          className,
          message: value.message,
        })
      })
      .catch((error) => {
        console.error(error)
      })
  },
  async fetchUpdateUserPassword(
    context: Context,
    accountDataWithNewPassword: AccountData,
  ): Promise<void> {
    const domain = context.rootState.domain
    await fetch(domain + "/api/users", {
      method: "PATCH",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(accountDataWithNewPassword),
    })
      .then((response) => {
        return response.json()
      })
      .then((value) => {
        let className: string;
        if (value.message.includes("Success")) {
```

```
                    className = "green"
                } else {
                    className = "red"
                }
                context.commit("updateFlashMessage", {
                    className,
                    message: value.message,
                })
            })
            .catch((error) => {
                console.error(error)
            })
    },
    async fetchDeleteUser(context: Context, accountData: AccountData): Promise<void> {
        const domain = context.rootState.domain
        await fetch(domain + "/api/users", {
            method: "DELETE",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify(accountData),
        })
            .then((response) => {
                return response.json()
            })
            .then((value) => {
                context.commit("resetUserAccount")
                context.commit("updateFlashMessage", {
                    className: "red",
                    message: value.message,
                })
            })
            .catch((error) => {
                console.error(error)
            })
    },
    passwordMismatchAlert(context: Content) {
        context.commit("updateFlashMessage", {
            className: "red",
            message: "Passwords mismatch!",
        })
    },
    logout(context: Context) {
        context.commit("resetUserAccount")
        context.commit("updateFlashMessage", {
            className: "blue",
            message: "Successful logout!",
        })
    },
  },
}
```

Файл “src/frontend/src/data/modules/words.ts”

```
import Context from "./interfaces/Context.ts"
import State from "./interfaces/State.ts"
```

```
import Word from "./interfaces/Word.ts"

export default {
   state: {
      words: new Array<Word>(),
   },
   getters: {
      getWords(state: State) {
         return state.words
      },
   },
   mutations: {
      updateWords(state: State, words: Array<Word>) {
         state.words = words
      },
   },
   actions: {
      async fetchWords(context: Context, limit: number = 10): Promise<void> {
         const domain = context.rootState.domain
         fetch(domain + "/api/words?limit=" + limit)
            .then((response) => response.json())
            .then((value) => {
               context.commit("updateWords", value)
            })
            .catch((error) => console.error(error))
      },
      async fetchWordByOrigin(context: Context, origin: string = ""): Promise<any> {
         const domain = context.rootState.domain
         console.log(domain + "/api/words?origin=" + origin);
         return fetch(domain + "/api/words?origin=" + origin)
            .then((response) => {
               if (!response.ok) {
                  throw new Error("Network response was not ok");
               }
               return response.json();
            })
            .catch((error) => {
               console.error("Error fetching word:", error);
               throw error;
            });
      },
   },
}
```

Файл “src/frontend/src/data/modules/interfaces/AccountData.ts”

```
interface AccountData {
   username: string
   password: string
   newPassword?: string
}
export default AccountData
```

Файл “src/frontend/src/data/modules/interfaces/Context.ts”

```
interface Context {
   commit: Function
   [contextProperty: string]: any
}
export default Context
```

Файл “src/frontend/src/data/modules/interfaces/FlashMessage.ts”

```
interface FlashMessage {
   message: string,
   className: string,
}
export default FlashMessage
```

Файл “src/frontend/src/data/modules/interfaces/State.ts”

```
interface State {
   [stateProperty: string]: any
}
export default State
```

Файл “src/frontend/src/data/modules/interfaces/UserAccount.ts”

```
interface UserAccount {
   _id: number
   username: string
   firstName: string
   lastName: string
   emailAddress: string
   gender: string
   birthday: string
   password?: string
}
export default UserAccount
```

Файл “src/frontend/src/data/modules/interfaces/Word.ts”

```
interface Word {
   _id: number
   origin: string
   translation: string
}
export default  Word
```

Файл “src/frontend/src/data/enums/SideBarStates.ts”

```
enum SideBarStates {
   Basics,
   ConversationGuidelines,
   Exercises,
```

```
}
export default SideBarStates
```

Файл “src/frontend/src/components/AppHeader.vue”

```
<script setup lang="ts">
import { RouterLink } from "vue-router"
import BookIcon from './icons/BookIcon.vue'
import SideBarStates from '../data/enums/SideBarStates'
</script>
<style lang="scss" scoped>
@mixin flex-vertical-center {
    display: flex;
    flex-direction: row;
    align-items: center;
}
header {
    box-shadow:
        rgba(0, 0, 0, 0.2) 0px 2px 4px -1px,
        rgba(0, 0, 0, 0.14) 0px 4px 5px 0px,
        rgba(0, 0, 0, 0.12) 0px 1px 10px 0px;
    width: 100%;
    height: 60px;
    background-color: var(--primary-color);
    color: rgb(255, 255, 255);
    z-index: 3;
}
.container {
    height: 100%;
    width: 90%;
    margin: auto auto;
    gap: 20px;
    @include flex-vertical-center;
}

.right-controls {
    position: absolute;
    right: 30px;
}
.right-controls button {
    margin-left: 30px;
}
.logotype {
    margin-right: 20px;
    @include flex-vertical-center;
}
.logotype-image {
    width: 30px;
    height: 30px;
    margin-right: 8px;
}
a,
button {
    font-size: 16px;
    color: white;
    text-decoration: none;
```

```
    transition: color 200ms ease-in-out;
    background: none;
    border: none;
}
a:hover,
button:hover {
    color: aquamarine;
}
.active {
    text-decoration: underline;
}
</style>
<template>
    <header>
        <div class="container">
            <router-link to="/" class="logotype">
                <book-icon class="logotype-image" />
                <span>English Chest</span>
            </router-link>
            <router-link to="/dictionary">Dictionary</router-link>
            <router-link to="/about">About</router-link>
            <span class="right-controls">
                <button :class="{active: getSideBarState === SideBarStates.Basics}"
@click="updateSideBarState(SideBarStates.Basics)">Basics</button>
                <button :class="{active: getSideBarState === SideBarStates.Exercises}"
@click="updateSideBarState(SideBarStates.Exercises)">Exercises</button>
                <button :class="{active: getSideBarState === SideBarStates.ConversationGuidelines}"
@click="updateSideBarState(SideBarStates.ConversationGuidelines)">Conversation
guidelines</button>
                <span v-if="getIsAuth">
                    <router-link to="/profile"><button>Profile</button></router-link>
                    <button @click="logout">Logout</button>
                </span>
                <span v-else>
                    <router-link to="/login"><button>Login</button></router-link>
                    <router-link to="/register"><button>Register</button></router-link>
                </span>
            </span>
        </div>
    </header>
</template>
<script lang="ts">
import { mapGetters, mapMutations, mapActions } from 'vuex';
export default {
    name: "app-header",
    computed: mapGetters(["getIsAuth", "getSideBarState"]),
    methods: {
        ...mapMutations(["updateSideBarState"]),
        ...mapActions(["logout"])
    }
}
</script>
```

Файл “src/frontend/src/components/AutotranslateText.vue”

```
<style lang="scss">
[data-translation] {
   display: inline-block;
   position: relative;
}
[data-translation]::after {
   content: " ";
   font-size: 70%;
   font-weight: span;
   vertical-align: super;
}
[data-translation]::before {
   display: block;
   width: max-content;
   max-width: 320px;
   padding: 8px;
   position: absolute;
   bottom: 100%;
   transform: translationX(-50%);
   font-size: small;
   z-index: 100;
   content: attr(data-translation);
   background-color: white;
   box-shadow: 0px 2px 5px 0px rgba(38, 51, 77, 0.03);
   visibility: hidden;
   border-radius: 5px;
   border-style: solid;
   border-color: #f5f6f7;
   border-width: 2px;
}
[data-translation]:hover {
   cursor: help;
   text-decoration: black wavy underline;
   text-decoration-skip-ink: none;
}
[data-translation]:hover::before {
   visibility: visible;
}
p>span {
   margin-right: 1%;
}
@media (orientation: portrait) {
   [data-translation]::before {
      background-color: black;
      color: white;
      display: block;
      position: fixed;
      width: 100%;
      max-width: unset;
      font-size: 1;
      /* reset to default */
      bottom: 0;
      text-align: center;
   }
```

```
}
</style>
<template>
    <p ref="elementParagraph">
        <slot></slot>
    </p>
</template>

<script lang="ts">
import { mapActions } from 'vuex'

export default {
    data: () => {
        return {
            origin: new String,
        }
    },
    methods: {
        ...mapActions(["fetchWordByOrigin"])
    },
    mounted() {
        const paragraph = this.$refs.elementParagraph
        const origins = paragraph.textContent.split(" ")
        paragraph.innerHTML = ""
        origins.forEach((origin) => {
            const span = document.createElement("span")
            span.textContent = origin
            this.fetchWordByOrigin(origin)
                .then((word: any) => {
                    if (word.translation) {
                        span.dataset.translation = word.translation
                    }
                    else {
                        span.dataset.translation = origin
                    }
                })
                .catch((error: Error) => {
                    console.error(error)
                })
            paragraph.appendChild(span)
        })

    }
}
</script>
```

Файл “src/frontend/src/components/CenteredContent.vue”

```
<style lang="scss" scoped>
div {
    text-align: center;
    display: flex;
    justify-content: center;
```

```
    align-items: center;
    flex-direction: column;
    gap: 10px;
}
</style>
<template>
    <div>
        <slot></slot>
    </div>
</template>
<script lang="ts">
export default {
    name: "centered-content"
}
</script>
```

Файл “src/frontend/src/components/Container.vue”

```
<style lang="scss" scoped>
div {
    width: 800px;
    margin: 0 auto;
    text-align: justify;
}
</style>
<template>
    <div>
        <slot></slot>
    </div>
</template>
<script lang="ts">
export default {
    // eslint-disable-next-line vue/multi-word-component-names
    name: "container",
}
</script>
```

Файл “src/frontend/src/components/FlashMessage.vue”

```
<style lang="scss" scoped>
div {
    width: 250px;
    height: 75px;
    background-color: white;
    position: fixed;
    right: 50px;
    bottom: 50px;
    z-index: 1000;
    display: flex;
    align-items: center;
    padding: 25px;
    box-shadow: 0px 2px 5px 0px rgba(38, 51, 77, 0.03);
    border-radius: 5px;
    border-style: solid;
```

```
    border-color: #f5f6f7;
    border-width: 2px;
    animation: appear 5s forwards;
}

@keyframes appear {
    from {
        transform: translateX(200%);
        opacity: 1;
    }
    10% {
        transform: translateX(0);
        opacity: 1;
    }
    90% {
        transform: translateX(0);
        opacity: 1;
    }
    to {
        opacity: 0;
        transform: translateX(200%);
    }
}
.red {
    color: #58151c;
    background-color: #f8d7da;
    border-color: #f1aeb4;
}
.green {
    color: #0a3622;
    background-color: #d1e7dd;
    border-color: #a3cfbb;
}

.blue {
    color: #052c65;
    background-color: #cfe2ff;
    border-color: #9ec5fe;
}
</style>
<template>
    <div :class="className" flash>
        {{ message }}
    </div>
</template>
<script lang="ts">
export default {
    name: "pretty-button",
    props: {
        message: {
            type: String,
            required: true,
        },
        className: {
            type: String,
            required: true,
```

```
        }
    },
}
</script>
```

Файл “src/frontend/src/components/InputLabel.vue”

```
<style lang="scss" scoped>
label {
    top: 18px;
    left: -5px;
    position: relative;
    color: var(--primary-color);
    background-color: white;
}
</style>
<template>
    <label>
        <slot></slot>
    </label>
</template>
<script lang="ts">
export default {
    name: "input-label"
}
</script>
```

Файл “src/frontend/src/components/PrettyButton.vue”

```
<style lang="scss" scoped>
button {
    display: inline-block;
    padding: 8px 16px;
    border: 1px solid var(--primary-color);
    background-color: var(--primary-color);
    color: #fff;
    font-size: 18px;
    text-align: center;
    text-decoration: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s ease, color 0.3s ease, border-color 0.3s ease;
}
button:hover {
    background-color: #fff;
    color: var(--primary-color);
    border-color: var(--primary-color);
}

</style>
<template>
    <button type="button">
```

```
        <slot></slot>
    </button>
</template>
<script lang="ts">
export default {
    name: "pretty-button"
}
</script>
```

Файл “src/frontend/src/components/PrettyLabel.vue”

```
<style lang="scss" scoped>
label {
    color: var(--primary-color);
}
</style>
<template>
    <label>
        <slot></slot>
    </label>
</template>

<script lang="ts">
export default {
    name: "pretty-label"
}
</script>
```

Файл “src/frontend/src/components/PrettyTable.vue”

```
<style lang="scss">
.pretty-table {
    margin: auto auto;
    border-collapse: collapse;
}
.pretty-table th,
.pretty-table td {
    border: 1px solid #ddd;
    text-align: center;
    padding: 10px;
}
.pretty-table th {
    background-color: #f2f2f2;
}
.pretty-table tbody tr:nth-child(even) {
    background-color: #f9f9f9;
}
.pretty-table tbody tr:hover {
    background-color: #f2f2f2;
}
</style>
<template>
    <table class="pretty-table">
        <slot></slot>
```

```
    </table>
</template>
<script lang="ts">
export default {
    name: "pretty-table"
}
</script>
```

Файл “src/frontend/src/components/SideBar.vue”

```
<script lang="ts" setup>
import { ref } from "vue"
import SidebarButton from './SidebarButton.vue'
import SidebarReference from './SidebarReference.vue'
import SideBarStates from '../data/enums/SideBarStates'
</script>
<style lang="scss">
:root {
    --transition-duration: 250ms;
}

summary {
    list-style: none;
    transition: all var(--transition-duration) ease-in-out;
}
.sidebar {
    box-shadow:
        rgba(0, 0, 0, 0.2) 0px 2px 4px -1px,
        rgba(0, 0, 0, 0.14) 0px 4px 5px 0px,
        rgba(0, 0, 0, 0.12) 0px 1px 10px 0px;
    background-color: var(--primary-color);
    z-index: 2;
    transition: width var(--transition-duration) ease-in-out;
    overflow-y: scroll;
    height: 90vh;
}
.expanded {
    width: 285px;
}
.collapsed {
    width: 60px;
}
.sidebar-button {
    text-wrap: nowrap;
    color: white;
    transition: width var(--transition-duration) ease-in-out;
    overflow: hidden;
    text-overflow: clip;
    span {
        text-align: center;
    }
}
details.sidebar-button:hover {
    text-decoration: none;
}
```

```
.sidebar-button {
    background-color: #3498db;
    width: 100%;
    max-height: 40px;
    white-space: nowrap;
    display: block;
    border: none;
    text-decoration: none;
    cursor: pointer;
    transition: all var(--transition-duration) ease;
    padding: 10px 20px;
    text-align: left;
    overflow: hidden;
    span {
        transition: all var(--transition-duration) ease-in-out;
    }
}

.sidebar-button:hover {
    background-color: var(--secondary-color);
    text-decoration: underline;
}
.expanded .sidebar-button {
    max-height: 100%;
    span {
        margin-left: 5px;
        opacity: 100%;
    }
}
.collapsed .sidebar-button {
    span {
        margin-left: 0;
        opacity: 0;
    }
}

.sidebar-control-button {
    position: fixed;
    top: 8px;
    left: 8px;
    z-index: 5;
    width: 45px;
    height: 45px;
    border-radius: 50%;
    background-color: #3498db;
    color: #fff;
    border-width: 1px;
    border-color: var(--primary-color);
    font-size: 16px;
    font-weight: bold;
    text-align: center;
    font-size: 24px;
}
.sidebar-control-button:hover {
    background-color: #2980b9;
}
```

```
.details-content {
   animation-name: slidedown;
   animation-duration: 200ms;
   animation-fill-mode: forwards;
}
@keyframes slidedown {
   from {
      transform: translateY(-200%);
   }
   to {
      transform: translateY(0%);
   }
}

</style>
<template>
   <nav ref="sidebarReference" class="sidebar expanded">
      <span v-if="getSideBarState === SideBarStates.Basics">
         <sidebar-button
            sign="Alphabet"
            route="/alphabet"
            icon="fa-solid fa-font"
         />
         <sidebar-button
            sign="Articles"
            route="/articles"
            icon="fa-solid fa-newspaper"
         />
         <sidebar-button
            sign="Plural"
            route="/plural"
            icon="fa-solid fa-users"
         />
         <sidebar-button
            sign="Numbers"
            route="/numbers"
            icon="fa-solid fa-4"
         />
         <sidebar-button
            sign="Irregular Verbs"
            route="/irregular-verbs"
            icon="fa-solid fa-puzzle-piece"
         />
         <sidebar-button
            sign="Possessive Form"
            route="/possessive-form"
            icon="fa-solid fa-clipboard-user"
         />
         <details class="sidebar-button">
            <summary @click="expandSidebar">
               <i class="fa-regular fa-comments"></i>
               <span>Pronouns</span>
            </summary>
            <br/>
            <div class="details-content">
               <sidebar-button
```

```
            sign="Personal Pronouns"
            route="/personal-pronouns"
            icon="fa-solid fa-user"
        />
        <sidebar-button
            sign="Reflexive Pronouns"
            route="/reflexive-pronouns"
            icon="fa-solid fa-reply"
        />
    </div>
</details>
<sidebar-button
    sign="Prepositions"
    route="/prepositions"
    icon="fa-solid fa-solid fa-map-location-dot"
/>
<sidebar-button
    sign="Degrees of Comparison"
    route="/degrees-of-comparison"
    icon="fa-solid fa-balance-scale"
/>
<sidebar-button
    sign="To be able"
    route="/to-be-able"
    icon="fa-solid fa-check-circle"
/>
<sidebar-button
    sign="Levels"
    route="/levels"
    icon="fa-solid fa-layer-group"
/>
<details class="sidebar-button">
    <summary @click="expandSidebar" style="margin-left: 5px;">
        <i class="fa-regular fa-hourglass-half"></i>
        <span>Times</span>
    </summary>
    <div class="details-content">
        <br />
        <sidebar-button
            sign="Simple Tense"
            route="/simple-tense"
            icon="fa-solid fa-hourglass"
        />
        <sidebar-button
            sign="Progressive Tense"
            route="/progressive-tense"
            icon="fa-solid fa-hourglass-start"
        />
        <sidebar-button
            sign="Perfect Tense"
            route="/perfect-tense"
            icon="fa-solid fa-hourglass-end"
        />
        <sidebar-button
            sign="Progressive Perfect Tense"
            route="/progressive-perfect-tense"
```

```
            icon="fa-solid fa-hourglass-half"
          />
        </div>
      </details>
      <sidebar-button
        sign="Future In The Past"
        route="/future-in-the-past"
        icon="fa-solid fa-clock"
      />
      <sidebar-button
        sign="Used to"
        route="/used-to"
        icon="fa-solid fa-history"
      />
      <sidebar-button
        sign="Conditional Sentences"
        route="/conditional-sentences"
        icon="fa-solid fa-question-circle"
      />
      <sidebar-button
        sign="Modal Verbs"
        route="/modal-verbs"
        icon="fa-solid fa-cogs"
      />
      <sidebar-button
        sign="Infinitive"
        route="/infinitive"
        icon="fa-solid fa-infinity"
      />
      <sidebar-button
        sign="The Gerund"
        route="/the-gerund"
        icon="fa-solid fa-circle-notch"
      />
      <sidebar-button
        sign="The Conditional Mood"
        route="/the-conditional-mood"
        icon="fa-solid fa-random"
      />
    </span>
    <span v-else-if="getSideBarState === SideBarStates.ConversationGuidelines">
      <sidebar-button
        sign="Greetings"
        route="/greetings"
        icon="fa-solid fa-handshake"
      />
      <sidebar-button
        sign="Goodbyes"
        route="/goodbyes"
        icon="fa-solid fa-hand-paper"
      />
      <sidebar-button
        sign="Introduction yourself"
        route="/introduction-yourself"
        icon="fa-solid fa-address-card"
      />
```

```
    <sidebar-button
      sign="Socialite Talk"
      route="/socialite-talk"
      icon="fa-solid fa-comment-dots"
    />
  </span>
  <span v-else-if="getSideBarState === SideBarStates.Exercises">
    <sidebar-button
      sign="Nouns articles"
      route="/nouns-articles-exercise"
      icon="fa-solid fa-a"
    />
    <sidebar-button
      sign="Simple Tense"
      route="/simple-tense-exercise"
      icon="fa-solid fa-hourglass"
    />
    <sidebar-button
      sign="Progressive Tense"
      route="/progressive-tense-exercise"
      icon="fa-solid fa-hourglass-start"
    />
    <sidebar-button
      sign="Perfect Tense"
      route="/perfect-tense-exercise"
      icon="fa-solid fa-hourglass-end"
    />
    <sidebar-button
      sign="Progressive Perfect Tense"
      route="/progressive-perfect-tense-exercise"
      icon="fa-solid fa-hourglass-half"
    />
    <sidebar-button
      sign="Future Perfect"
      route="/future-perfect-exercise"
      icon="fa-solid fa-clock"
    />
    <sidebar-button
      sign="Conditional Sentences"
      route="/conditional-sentences-exercise"
      icon="fa-solid fa-question-circle"
    />
    <sidebar-button
      sign="Gerund after prepositions"
      route="/gerund-after-prepositions-exercise"
      icon="fa-solid fa-circle-notch"
    />
    <sidebar-button
      sign="Gerund and infinitive"
      route="/gerund-and-infinitive-exercise"
      icon="fa-solid fa-infinity"
    />
    <sidebar-reference
      sign="More Exercises"
      href="https://www.englisch-hilfen.de/en/exercises_list/zeitformen.htm"
      icon="fa-solid fa-ellipsis"
```

```
            />
        </span>
    </nav>
    <button class="sidebar-control-button" @click="toggleCollapse">
        <i class="fa-solid fa-circle-chevron-right" v-if="isCollapsedReference"></i>
        <i class="fa-solid fa-circle-chevron-left" v-else></i>
    </button>
</template>
<script lang="ts">
import { mapGetters } from 'vuex'
const sidebarReference = ref<HTMLElement | null>(null)
const isCollapsedReference = ref<boolean>(false)
const closeAllDetails = function () {
    const openedDetails: NodeListOf<Element> = document.querySelectorAll("[open]")
    openedDetails.forEach((element: Element) => {
        element.removeAttribute("open")
    })
}
const collapseSidebar = function () {
    if (sidebarReference.value) {
        isCollapsedReference.value = true
        sidebarReference.value.classList.add("collapsed")
        sidebarReference.value.classList.remove("expanded")
    }
}
const expandSidebar = function () {
    if (sidebarReference.value) {
        isCollapsedReference.value = false
        sidebarReference.value.classList.add("expanded")
        sidebarReference.value.classList.remove("collapsed")
    }
}
const toggleCollapse = function () {
    if (sidebarReference.value) {
        isCollapsedReference.value = !isCollapsedReference.value
        if (isCollapsedReference.value) {
            collapseSidebar()
            closeAllDetails()
        } else {
            expandSidebar()
        }
    }
}
export default {
    name: 'side-bar',
    computed: {
        ...mapGetters(["getSideBarState"])
    },
}
</script>
```

Файл “src/frontend/src/components/SidebarButton.vue”

```
<script setup lang="ts">
import { RouterLink } from "vue-router"
</script>
<style lang="scss" scoped>
i {
   height: 15px;
   width: 20px;
   display: inline-block;
   text-align: center;
}
</style>
<template>
   <router-link :to="route" class="sidebar-button" @click="scrollToTop">
      <i :class="icon"></i>
      <span>{{ sign }}</span>
   </router-link>
</template>
<script lang="ts">
export default {
   props: {
      sign: {
         type: String,
         required: true,
      },
      route: {
         type: String,
         required: true,
      },
      icon: {
         type: String,
         required: true,
      },
   },
   methods: {
      scrollToTop() {
         window.scrollTo({ top: 0, behavior: "smooth" })
      },
   },
}
</script>
```

Файл “src/frontend/src/components/SidebarReference.vue”

```
<style lang="scss" scoped>
i {
   height: 15px;
   width: 20px;
   display: inline-block;
   text-align: center;
}
</style>

<template>
   <a :href="href" class="sidebar-button" target="_blank">
      <i :class="icon"></i>
```

```
      <span>{{ sign }}</span>
   </a>
</template>
<script lang="ts">
export default {
   props: {
      sign: {
         type: String,
         required: true,
      },
      href: {
         type: String,
         required: true,
      },
      icon: {
         type: String,
         required: true,
      },
   }
}
</script>
```

Файл “src/frontend/src/components/icons/BookIcon.vue”

```
<template>
   <span>
      <svg
         version="1.1"
         id="svg261"
         xml:space="preserve"
         width="25"
         height="32"
         xmlns="http://www.w3.org/2000/svg"
         xmlns:svg="http://www.w3.org/2000/svg"
      >
         <defs id="defs265" />
         <g id="g267" transform="translate(-3,-0.99994578)">
            <g transform="matrix(1.4124409,0,0,1.4608587,-1.2373227,-1502.808)" id="g14">
               <path
                  d="m 3,8 v 2 1 3 1 5 1 c 0,1.105 0.8954,2 2,2 h 14 c 1.105,0 2,-0.895 2,-2 V 20 15 11 8 Z"
                  fill="#16a085"
                  transform="translate(0,1028.4)"
                  id="path2"
               />
               <path
                  d="m 3,1035.4 v 2 1 3 1 5 1 c 0,1.1 0.8954,2 2,2 h 14 c 1.105,0 2,-0.9 2,-2 v -1 -5 -4 -3 z"
                  fill="#ecf0f1"
                  id="path4"
               />
               <path
                  d="m 3,1034.4 v 2 1 3 1 5 1 c 0,1.1 0.8954,2 2,2 h 14 c 1.105,0 2,-0.9 2,-2 v -1 -5 -4 -3 z"
                  fill="#bdc3c7"
                  id="path6"
```

```
                />
                <path
                   d="m 3,1033.4 v 2 1 3 1 5 1 c 0,1.1 0.8954,2 2,2 h 14 c 1.105,0 2,-0.9 2,-2 v -1 -5 -4 -3 z"
                   fill="#ecf0f1"
                   id="path8"
                />
                <path
                   d="M 5,1 C 3.8954,1 3,1.8954 3,3 v 1 4 2 1 3 1 5 1 c 0,1.105 0.8954,2 2,2 H 7 V 22 H 5.5
C 4.6716,22 4,21.328 4,20.5 4,19.672 4.6716,19 5.5,19 H 18 19 c 1.105,0 2,-0.895 2,-2 V 16 11 7 4 3 C
21,1.8954 20.105,1 19,1 h -4 z"
                   fill="#16a085"
                   transform="translate(0,1028.4)"
                   id="path10"
                />
                <path
                   d="m 8,1 v 18 h 1 9 1 c 1.105,0 2,-0.895 2,-2 V 16 11 7 4 3 C 21,1.8954 20.105,1 19,1 H
15 9 Z"
                   fill="#1abc9c"
                   transform="translate(0,1028.4)"
                   id="path12"
                />
             </g>
          </g>
       </svg>
    </span>
</template>
```

Файл “src/frontend/src/static/styles/css/styles.css”

```
@charset "utf-8";
:root {
    --primary-color: rgb(25, 118, 210);
    --secondary-color: #2980b9;
    --general-font-size: 24px
}
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
main * {
    font-size: var(--general-font-size);
    animation: 2s appear forwards;
}
@keyframes appear {
    0% {
        opacity: 0;
    }
    50% {
        opacity: 0;
    }
    100% {
        opacity: 1;
    }
```

```
}
body {
    overflow: hidden;
}
.grid-container {
    display: grid;
    grid-template-areas:
        "header header header"
        "sidebar main main"
        "sidebar main main";
    grid-template-rows: auto 1fr auto;
    /* header, main, footer */
    grid-template-columns: auto 1fr;
    /* sidebar, main */
    min-height: 100vh;
}
header {
    grid-area: header;
}
.sidebar {
    grid-area: sidebar;
    overflow-y: auto;
}
.algin-center {
    text-align: center;
}
main {
    grid-area: main;
    height: 85vh;
    overflow-x: hidden;
    overflow-y: scroll;
    margin-top: 10px;
    margin-left: 15px;
}
footer {
    grid-area: footer;
}
ul {
    list-style: none;
}
p,
li {
    font-size: var(--general-font-size);
}
h1,
h2,
h3,
h4,
h5,
h6 {
    text-align: center;
    margin-top: var(--general-font-size);
    margin-bottom: var(--general-font-size);
    font-size: calc(var(--general-font-size) * 1.5);
}
mark {
```

```
    background-color: white;
    color: var(--primary-color);
}
input.pretty {
    padding: 8px 12px;
    border: 1px solid var(--primary-color);
    color: rgb(25, 118, 210);
    border-radius: 4px;
    font-size: 24px;
    transition: border-color 0.3s ease;
}
input.pretty:focus {
    outline: none;
    border-color: var(--secondary-color);
}
input.pretty:invalid {
    border-color: red;
    color: red;
}
select.pretty {
    border: 2px solid var(--primary-color);
    color: var(--primary-color);
    border-radius: 5%;
    padding: 8px;
    background-color: white;
}
.placeholder {
    line-height: 30px;
    height: 30px;
}
label {
    z-index: 10;
}
```

Файл “src/frontend/src/static/styles/css/fonts.css”

```
* {
    font-weight: bold;
    font-style: normal;
    font-family: Tinos, sans-serif;
}
```

Файл “src/frontend/src/views/AboutView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import CenteredContent from '../components/CenteredContent.vue'
</script>
<style scoped>
.references {
    counter-reset: li;
}
.references li {
    list-style: none;
```

```
    position: relative;
    padding: .75em .5em;
    margin: 0 0 0 2em;
}
.references li::before {
    background-color: lightblue;
    color: var(--primary-color);
    border-radius: 50%;
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    -ms-box-sizing: border-box;
    box-sizing: border-box;
    content: counter(li);
    counter-increment: li;
    font-size: .875em;
    font-weight: 700;
    height: 1.785em;
    left: -2em;
    margin-top: -0.9em;
    padding-top: 0;
    position: absolute;
    text-align: center;
    top: 50%;
    width: 1.785em;
    line-height: 1.8em;
}
</style>
<template>
    <main>
        <h1>About</h1>
        <container>
            <centered-content>
                <autotranslate-text>
                    English Chest is created by Alexander Serhiienko.
                </autotranslate-text>
            </centered-content>
        </container>
        <br>
        <h1>References</h1>
        <container>
            <centered-content>
                <ol class="references">
                    <li><a href="https://www.duolingo.com/" target="_blank">Edu. material of
Duolingo</a></li>
                    <li><a href="https://www.englishclub.com/" target="_blank">Learn English Your
Way</a></li>
                    <li><a href="https://www.englisch-hilfen.de/en/exercises_list/zeitformen.htm"
target="_blank">English Tenses - Exercise</a></li>
                </ol>
            </centered-content>
        </container>
    </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/AlphabetView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
</script>
<template>
  <main>
    <h1>English Alphabet</h1>
    <br>
    <container>
      <autotranslate-text>The English alphabet has 26 letters.</autotranslate-text>
      <autotranslate-text>In &amp;quot;alphabetical order&amp;quot;, they are:</autotranslate-text>
      <p>
        <mark>a</mark> b c d <mark>e</mark> f g h
        <mark>i</mark> j k l m n <mark>o</mark> p q r s t
        <mark>u</mark> v w x y z
      </p>
      <autotranslate-text>Five of the letters are "vowels": </autotranslate-text><p><mark>a e i o
u</mark></p>
      <autotranslate-text>The remaining twenty-one letters are "consonants".</autotranslate-text>
      <autotranslate-text>We can write each letter as a large letter or small letter.</autotranslate-text>
    </container>
    <br />
    <pretty-table>
      <thead>
        <tr>
          <th>Capital letter</th>
          <th>Small letter</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>A</td>
          <td>a</td>
        </tr>
        <tr>
          <td>B</td>
          <td>b</td>
        </tr>
        <tr>
          <td>C</td>
          <td>c</td>
        </tr>
        <tr>
          <td>D</td>
          <td>d</td>
        </tr>
        <tr>
          <td>E</td>
          <td>e</td>
        </tr>
        <tr>
```

```
    <td>F</td>
    <td>f</td>
</tr>
<tr>
    <td>G</td>
    <td>g</td>
</tr>
<tr>
    <td>H</td>
    <td>h</td>
</tr>
<tr>
    <td>I</td>
    <td>i</td>
</tr>
<tr>
    <td>J</td>
    <td>j</td>
</tr>
<tr>
    <td>K</td>
    <td>k</td>
</tr>
<tr>
    <td>L</td>
    <td>l</td>
</tr>
<tr>
    <td>M</td>
    <td>m</td>
</tr>
<tr>
    <td>N</td>
    <td>n</td>
</tr>
<tr>
    <td>O</td>
    <td>o</td>
</tr>
<tr>
    <td>P</td>
    <td>p</td>
</tr>
<tr>
    <td>Q</td>
    <td>q</td>
</tr>
<tr>
    <td>R</td>
    <td>r</td>
</tr>
<tr>
    <td>S</td>
    <td>s</td>
</tr>
<tr>
```

```
            <td>T</td>
            <td>t</td>
          </tr>
          <tr>
            <td>U</td>
            <td>u</td>
          </tr>
          <tr>
            <td>V</td>
            <td>v</td>
          </tr>
          <tr>
            <td>W</td>
            <td>w</td>
          </tr>
          <tr>
            <td>X</td>
            <td>x</td>
          </tr>
          <tr>
            <td>Y</td>
            <td>y</td>
          </tr>
          <tr>
            <td>Z</td>
            <td>z</td>
          </tr>
        </tbody>
      </pretty-table>
    </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/ArticlesView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
  <main>
    <h1>Articles</h1>
    <br />
    <container>
      <autotranslate-text>Articles in English language are words that indicate particularity or specify
        names.</autotranslate-text>
      <br>
      <autotranslate-text>"A" or "an" is an indefinite article. It is used when we are talking about
something
        general or when we mention a subject for the first time.
      </autotranslate-text>
      <autotranslate-text>Example: "I have a cat" (I have a cat).</autotranslate-text>
      <br>
```

```
        <autotranslate-text>"The" is a definite article. It is used when we are talking about something
specific
            that is already known or mentioned before.
        </autotranslate-text>
        <autotranslate-text>Example: "The cat is on the table" (Cat on the table).</autotranslate-text>
        <br>
        <autotranslate-text>Sometimes an article is not used at all, which is called a "null" article.
        </autotranslate-text>
        <autotranslate-text>It occurs before uncountable nouns or when we are talking about something
general or
            abstract.</autotranslate-text>
        <autotranslate-text>Example: "I like music" (I like music).</autotranslate-text>
      </container>
    </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/ConditionalSentencesView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
    <main>
      <h1>Conditional Sentences</h1>
      <br />
      <container>
        <autotranslate-text>There are three types of conditional sentences in English:</autotranslate-text>
        <br>
        <autotranslate-text>First conditional: expresses a possible future condition and its result if the
            condition is met. The structure is "if + present simple, will + base form verb".
            Example: If I have time, I will go to the cinema.
        </autotranslate-text>
        <br>
        <autotranslate-text>Second conditional: expresses an unreal or unlikely condition and its result.
The
            structure is "if + past simple, would/could/might + base form verb". Example: If I
            won the lottery, I would buy a house.
        </autotranslate-text>
        <br>
        <autotranslate-text>Third conditional: expresses a past condition and its result if the condition had
            been different. The structure is "if + past perfect, would/could/might + have + past
            participle". Example: If I had studied harder, I would have passed the exam.
        </autotranslate-text>
      </container>
    </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/DegreesOfComparisonView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyLabel from '../components/PrettyLabel.vue';
</script>
<style scoped>
.answer-place {
   height: 30px;
}
</style>
<template>
   <main>
      <h1>Degrees Of Comparison Of Adjectives</h1>
      <br />
      <container>
         <autotranslate-text>Degrees of comparative adjectives in English are divided into three types:
ordinary,
            superior and superlative.
         </autotranslate-text>
         <br>
         <autotranslate-text>Normal Degree (Positive Degree) is the normal form of an adjective, for
example: "new",
            "small", "old".</autotranslate-text>
         <br>
         <autotranslate-text>The Comparative Degree is a form that shows a comparison between two
things or people.
            To form a higher
            degree of an adjective, the ending "-er" is added, for example: "newer", "smaller", "older".
         </autotranslate-text>
         <br>
         <autotranslate-text>The Superlative Degree is a form that shows the highest quality among three
or more
            things or people. To
            form the highest degree of an adjective, the ending "-est" is added and the article "the" must be
placed
            before it, for example: "the newest", "the smallest", "the oldest".
         </autotranslate-text>
         <br>
         <autotranslate-text>There are three ways of forming degrees of comparison: simple, complex and
exceptions.
            The simple way
            requires the addition of "-er" and "-est", the complex way uses the adjectives "more" and
"most", and
            some
            adjectives have their own comparative forms, for example: "good" (good), "better", "best" (the
best).
         </autotranslate-text>
         <br>
         <autotranslate-text>To determine the method of formation of degrees of comparison of adjectives
in a
            specific case, it is
            first worth paying attention to the adjective itself:
         </autotranslate-text>
         <br>
```

```
        <autotranslate-text>Monosyllabic adjectives - an adjective consisting of only one adjective, a
simple way of
            forming degrees of
            comparison is usually used. For example: "new", "big".
        </autotranslate-text>
        <br>
        <autotranslate-text>Polysyllabic adjectives - an adjective consisting of several adjectives, then a
complex
            method using "more"
            and "most" is used to form degrees of comparison. For example: "more beautiful", "the most
famous".
        </autotranslate-text>
        <br>
        <autotranslate-text>Exceptions! There are adjectives that have their own forms of degrees of
comparison and
            do not follow
            the general rules. For example: "good", "bad", "little".
        </autotranslate-text>
        <br>
        <autotranslate-text>If you can't figure out which method to use, you can always use a dictionary
or check
            online.</autotranslate-text>
        <br>
        <pretty-label for="adjectiveInput">Input adjective: </pretty-label>
        <input class="pretty" name="adjectiveInput" id="adjectiveInput" type="text"
placeholder="Слово англ. мовою"
            v-model="adjective" />
        <br><br>
        <p class="answer-place">Маємо: {{ degreesOfComparison }}</p><br>
      </container>
  </main>
</template>
<script lang="ts">
export default {
  data: () => {
    return {
      adjective: "big",
      degreesOfComparison: "big – bigger – the biggest",
    }
  },
  watch: {
    adjective(newValue): void {
      if (newValue.length > 2)
        this.degreesOfComparison = this.getDegreesOfComparison(newValue)
    },
  },
  methods: {
    getDegreesOfComparison(adjective: string): string {
      adjective = adjective.toLowerCase()
      // Adjective-Exceptions
      switch (adjective) {
        case "many":
          return "many – more – the most"
        case "good":
          return "good – better – the best"
        case "bad":
```

```
                return "bad – worse – the worst"
            case "little":
                return "little – less – the least"
            case "old":
                return "old – older – the oldest"
            case "big":
                return "big – bigger – the biggest"
            case "far":
                return "far – farther – the farthest"
        }
        const originalAdjective: string = adjective
        const lengthOfAdjective: number = adjective.length
        const lastCharacter = adjective.slice(0, lengthOfAdjective - 1)
        if (lastCharacter == "y")
            // Remove last char and add 'i'
            adjective = adjective.slice(0, lengthOfAdjective - 1) + "i"
        else if (lastCharacter == "e")
            adjective = adjective.slice(0, lengthOfAdjective - 1)
        if (lengthOfAdjective < 8)
            return originalAdjective + " - " + adjective + "er - the " + adjective + "est"
        else
            return originalAdjective + " - more " + adjective + " - the most " + adjective
    },
  },
}
</script>
```

Файл “src/frontend/src/views/DictionaryView.vue”

```
<script lang="ts" setup>
import PrettyButton from '../components/PrettyButton.vue'
import PrettyTable from '../components/PrettyTable.vue'
import InputLabel from '../components/InputLabel.vue'
import Container from '../components/Container.vue'
import CenteredContent from "../components/CenteredContent.vue"
</script>
<template>
  <main>
    <container>
      <centered-content>
        <h1>Dictionary</h1>
        <input-label for="word">Слово англ. мовою</input-label>
        <input class="pretty" id="word" v-model="word" type="text" />
        <div class="placeholder" v-if="translation != "">
          Переклад: {{ translation }}
        </div>
        <div class="placeholder" v-else>{{ detail }}</div>
        <pretty-button @click="showDictionaryTable=true" v-if="!showDictionaryTable">Show
table</pretty-button>
        <div v-if="showDictionaryTable">
          <pretty-table>
            <thead>
              <tr>
                <th>Origin Form</th>
                <th>Translation</th>
```

```
                    </tr>
                </thead>
                <tbody>
                    <tr v-for="word in getWords" :key="word.id">
                        <td>{{ word.origin }}</td>
                        <td>{{ word.translation }}</td>
                    </tr>
                </tbody>
            </pretty-table>
            <br>
            <pretty-button type="button" id="browseMoreButton" @click="loadWords">
                Browse more
            </pretty-button>
        </div>
    </centered-content>
</container>
</main>
</template>
<script lang="ts">
import { mapActions, mapGetters } from "vuex"
export default {
    data: () => {
        return {
            limit: 20,
            word: new String,
            detail: new String,
            translation: new String,
            showDictionaryTable: false,
        }
    },
    computed: { ...mapGetters(["getDomain", "getWords"]) },
    methods: {
        ...mapActions(["fetchWords", "fetchWordByOrigin"]),
        async loadWords() {
            await this.fetchWords(this.limit).then(() => {
                this.limit += 10
            })
        }
    },
    watch: {
        async word(newValue) {
            if (newValue.length <= 2) {
                return
            }
            this.detail = this.translation = new String
            await this.fetchWordByOrigin(newValue)
                .then((value) => {
                    if (value.translation) {
                        this.translation = value.translation
                    }
                    else if (value.detail) {
                        this.detail = value.detail
                    }
                })
                .catch((error) => {
                    console.error(error)
```

```
            })
        },
    },
}
</script>
```

Файл “src/frontend/src/views/FutureInThePastView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
  <main>
    <h1>Future In The Past</h1>
    <br />
    <container>
      <autotranslate-text>"Future in the Past" is a construction used to express future time in the past
context.
        It allows you to
        describe an action that should have happened in the past, but at that moment was considered as
future.
      </autotranslate-text>
      <br>
      <autotranslate-text>It is formed with the help of the modal verb "would" or "was/were going to"
in the past
        tense + the
        infinitive of the semantic verb.
      </autotranslate-text>
      <br>
      For example:
      <autotranslate-text>She said she would call me later. (She said she would call me later.)
      </autotranslate-text>
      <autotranslate-text>He knew he was going to travel to Paris. (He knew that he was going to go to
        Paris.)</autotranslate-text>
      <br>
      <autotranslate-text>This construction is used when we describe plans, intentions or predictions
that were
        relevant in the past,
        but at the moment they are already in the past.
      </autotranslate-text>
    </container>
  </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/GoodbyesView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
  <main>
    <h1>Goodbyes</h1>
    <container>
      <autotranslate-text>Goodbyes are an important part of everyday communication. They can vary
from
        informal to formal
        depending on the situation. Here are some common ways to say goodbye in English.
```

```
        </autotranslate-text>
        <h1>Informal farewells</h1>
        <autotranslate-text>These goodbyes are used in everyday life with friends, family and colleagues
in
           informal setting.</autotranslate-text>
        <br>
        <p>Bye! - Бувай!</p>
        <p>See you! - Побачимося
!</p>
        <p>See you later! - До зустрічі!</p>
        <p>Catch you later! - Побачимося пізніше!</p>
        <p>Take care! - Бережи себе!</p>
        <p>Have a good day! - Гарного дня!</p>
        <p>I'm off! - Я пішов(ла)!</p>
        <h1>Semi-formal farewells</h1>
        <autotranslate-text>These goodbyes are suitable for use with colleagues, acquaintances or in a
less formal
           business environment.</autotranslate-text>
        <br>
        <p>Goodbye! - До побачення!</p>
        <p>Talk to you soon! - Скоро поговоримо!</p>
        <p>Have a great day! - Гарного дня!</p>
        <p>See you tomorrow! - Побачимося завтра!</p>
        <p>See you next time! - До наступної зустрічі!</p>
        <h1>Formal farewells</h1>
        <autotranslate-text>These farewells are used in formal situations such as business meetings,
communication
           with
           superiors or with
           people you don't know well.
        </autotranslate-text>
        <p>Goodbye! - До побачення!</p>
        <p>Farewell! - Прощавайте!</p>
        <p>It was nice meeting you! - Було приємно познайомитися!</p>
        <p>I look forward to our next meeting! - З нетерпінням чекаю на нашу наступну зустріч!</p>
        <p>Take care! - Бережіть себе!</p>
        <h1>Farewells by email</h1>
        <autotranslate-text>In business correspondence, goodbyes can also range from informal to
           formal.</autotranslate-text>
        <p>Best regards, - З найкращими побажаннями,</p>
        <p>Kind regards, - З повагою,</p>
        <p>Sincerely, - Щиро,</p>
        <p>Yours truly, - З повагою,</p>
        <p>Take care, - Бережіть себе,</p>
        <h1>Examples in context:</h1>
        <p>Informal context:</p>
        <autotranslate-text>A: "Alright, I'm heading out. See you later!"</autotranslate-text>
        <autotranslate-text>B: "Later!"</autotranslate-text>
        <p>Semiformal context:</p>
        <autotranslate-text>A: "It was great working with you today. Talk to you soon!"</autotranslate-
text>
        <autotranslate-text>B: "You too! Have a great day!"</autotranslate-text>
        <p>Formal context:</p>
        <autotranslate-text>A: "Thank you for the meeting. I look forward to our next
           discussion."</autotranslate-text>
        <autotranslate-text>B: "Goodbye! Have a good day!"</autotranslate-text>
```

```
        <autotranslate-text>Goodbyes are an important part of any language, and knowing how to say
goodbye correctly
            will help you
            sound polite and
            respectful in various situations.
        </autotranslate-text>
      </container>
   </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/GreetingsView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
   <main>
      <container>
         <h1>1. Regular greetings</h1>
         <br>
         <autotranslate-text>These greetings are used in everyday situations and are suitable for most
            occasions.</autotranslate-text>
         <br>
         <p>Hello - Вітаю</p>
         <p>Hi - Привіт</p>
         <p>Hey - Хей!</p>
         <p>Good morning - Доброго ранку (використовується до полудня)</p>
         <p>Good afternoon - Добрий день (використовується після полудня)</p>
         <p>Good evening - Добрий вечір (використовується після 6 вечора)</p>
         <p>How are you? - Як справи?</p>
         <p>What's up? / What's new? / What's going on? - Що нового? (Більш неформальне)</p>
         <h2>2. Formal greetings</h2>
         <autotranslate-text>These greetings are used in formal or professional situations.</autotranslate-
text>
         <br>
         <p>Good morning/afternoon/evening, mr./ms. - Доброе утро/день/вечер, містер/міс(іс).</p>
         <p>How do you do? - Як поживаєте? (дуже формально, часто використовується у відповідь
на таке ж вітання)</p>
         <p>
            It's a pleasure to meet you. - Приємно познайомитися.
            Nice to meet you. - Приємно познайомитися.
         </p>
         <p> Greetings. - Вітання.</p>
         <p>I hope you are doing well. - Надіюсь у вас все добре.</p>
         <h2>3. Informal greetings</h2>
         <autotranslate-text>These greetings are used with friends, family and in informal.</autotranslate-
text>
         <br>
         <p>Hey there! - Хей, привіт!</p>
         <p>Hiya! / Hiya there! - Привіт!</p>
         <p>Yo! - Йо!</p>
```

```
        <p>How's it going? - Як справи?</p>
        <p>How have you been? - Як ти?</p>
        <p>Long time no see! - Давно не бачилися!</p>
        <p>What's up? / Sup? - Що нового?</p>
        <p>Howdy! - Вітання! (зазвичай в Америці, особливо на півдні)</p>
        <h2>4. Greetings by time of day</h2>
        <autotranslate-text>Using greetings based on the time of day helps make your communications
more polite and
          appropriate.
        </autotranslate-text>
        <br>
        <p>Good morning (до 12:00)</p>
        <p>Good afternoon (з 12:00 до 17:00)</p>
        <p>Good evening (після 17:00)</p>
        <h2>5. Greetings in a business environment</h2>
        <autotranslate-text>Greetings in a business environment should be polite and
          professional.</autotranslate-text>
        <br>
        <p>Good morning/afternoon/evening, {{ name }}. - Доброго ранку/день/вечір, {{ name }}.</p>
        <p>Hello, {{ name }}. How are you today? - Привіт, {{ name }}. Як ви сьогодні?</p>
        <p>
          It's nice to see you again. - Приємно вас знову бачити.
        </p>
        <p>How have you been? - Як справи?</p>
        <p>I hope all is well with you. - Надіюсь у вас все добре.</p>
        <h2>6. Replies to greetings</h2>
        <autotranslate-text>You can respond to greetings in different ways depending on the situation.
        </autotranslate-text>
        <br>
        <p>Hello! / Hi! - Привіт!</p>
        <p>I'm good, thank you. How about you? - У мене все добре, дякую. А у вас?</p>
        <p>Doing well, thank you. - Все добре спасибі.</p>
        <p>Not bad, how about you? - Непогано, а у вас?</p>
        <p>Long time no see! How have you been? - Давно не бачилися! Як ти?</p>
        <h2>7. Examples of dialogues</h2>
        <p>Example 1: Regular greeting</p>
        <br>
        <p>A: Hi, how are you?</p>
        <p>B: I'm good, thanks. How about you?</p>
        <br>
        <p>Example 2: Formal greeting</p>
        <br>
        <p>A: Good morning, Mr. Smith. How do you do?</p>
        <p>B: Good morning. I'm well, thank you. And you?</p>
        <br>
        <p>Example 3: Informal greeting</p>
        <br>
        <p>A: Hey, what's up?</p>
        <p>B: Not much, just hanging out. You?</p>
      </container>
    </main>
</template>
<script lang="ts">
import { mapGetters } from 'vuex'
export default {
  computed: {
```

```
        ...mapGetters(["getUserAccount"]),
        name(): string {
            const userAccount = this.getUserAccount
            if (userAccount.firstName)
                return this.capitalize(userAccount.firstName)
            else
                return this.capitalize(userAccount.username)
        }
    },
    methods: {
        capitalize(str: string): string {
            return str[0].toUpperCase() + str.slice(1)
        },
    }
}
</script>
```

Файл “src/frontend/src/views/HomeView.vue”

```
<script lang="ts" setup>
import AutotranslateText from '../components/AutotranslateText.vue';
import Container from '../components/Container.vue';
</script>
<style lang="scss" scoped></style>
<template>
    <main>
        <h1>Welcome!</h1>
        <container>
            <autotranslate-text>Welcome to our educational platform! We are ready to help you master
English with ease and pleasure. Start your journey to English proficiency now!
            </autotranslate-text>
            <center>
                <img src="@/static/images/english-chest.png" alt="english-chest" width="250px"
height="250px">
            </center>
        </container>
    </main>
</template>
```

Файл “src/frontend/src/views/InfinitiveView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
  <main>
    <h1>Use Of The Infinitive</h1>
    <br />
    <container>
      <autotranslate-text>An infinitive in English is a form of a verb that is used unchanged, that is,
without ending -ed, -ing,
        etc.
        In English, infinitives often begin with the particle "to", they are called "to-infinitive".
      </autotranslate-text>
      <br>
      <autotranslate-text>Examples:</autotranslate-text>
      <autotranslate-text>to play (play)</autotranslate-text>
      <autotranslate-text>to eat (to eat)</autotranslate-text>
      <autotranslate-text>to study (to study)</autotranslate-text>
      <br>
      <autotranslate-text>Infinitives can be used after other verbs, after nouns, after adjectives, and in
other
        cases.</autotranslate-text>
      <br>
      <autotranslate-text>For example:</autotranslate-text>
      <autotranslate-text>She wants to sing a song. (She wants to sing a song.)
        He doesn't like to swim. (He likes to swim.)
        It's important to learn English. (It is important to learn English.)
      </autotranslate-text>
      <br>
      <autotranslate-text>Infinitives can also be used without "to". They are called "bare infinitive" and
are
        usually used after
        modal verbs (can, will, should, etc.) and after definite verbs (let, make, etc.).
      </autotranslate-text>
      <br>
      <autotranslate-text>Examples:</autotranslate-text>
      <autotranslate-text>She can sing. (She can sing.)</autotranslate-text>
      <autotranslate-text>They will go to the party. (They will go to the party.)</autotranslate-text>
      <br>
      <autotranslate-text>Infinitives play an important role in the English language as they are used to
form
        tenses, to express
        purposes, intentions, agreement, etc.
      </autotranslate-text>
    </container>
  </main>
</template>


<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/InOrderToView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
  <main>
    <h1>Sentences With The Conjunction "In Order To"</h1>
    <br />
    <container>
      <p>
        Хоча в англійській мові й існує еквівалент українського сполучника &laquo;щоб&raquo;
        - &ldquo;in order to&rdquo;, він не часто використовується в англійських реченнях.
        Замість того просто використовується дієслово у формі інфінітива.
      </p>
      <p>
        I went to the airport <strong>to </strong><strong>meet </strong> my father.
        <em>(Я поїхав у аеропорт, </em><strong>щоб </strong><strong>зустріти </strong
        ><em> мого батька.)</em>
      </p>
      <p>
        I need to see <strong>to </strong><strong>believe</strong>.
        <em>(Мені треба побачити, </em><strong>щоб </strong><strong>повірити </strong
        ><em>.)</em>
      </p>
      <br />
      <h5>
        <strong
          >Про інфінітив у реченнях типу: &laquo;Я хочу, щоб ви &hellip;&raquo; (Складне
          доповнення)
        </strong>
      </h5>
      <p>
        Речення зі складним доповненням (Complex Object) дуже характерні для англійської
        мови.
      </p>
      <p>I want you all to read this. <em>(Я хочу, щоб ви всі це прочитали.)</em></p>
      <p>
        Director does not want them to be late.
        <em>(Директор не хоче, щоб вони запізнилися.)</em>
      </p>
      <p>
        Do you want us to watch the film? <em>(Ви хочете, щоб ми подивилися цей фільм?)</em>
      </p>
      <p>
        Як ви бачите, у цих реченнях англійський інфінітив перекладається реченням зі
        сполучником &laquo;щоб&raquo;:
      </p>
      <p>you all to read this - щоб ви всі це прочитали</p>
      <p>them to be late - щоб вони запізнилися</p>
      <p>us to watch the film - щоб ми подивилися цей фільм</p>
      <p>Ось ще декілька прикладів речень зі складним доповненням (Complex Object)</p>
      <p>
        I want <strong>him </strong><strong>to </strong><strong>understand </strong> the
        problem. <em>(Я хочу, </em><strong>щоб </strong><strong>він </strong
```

```
            ><strong>зрозумів </strong><em> цю проблему.)</em>
          </p>
          <p>
            Do you want <strong>me </strong><strong>to </strong><strong>answer </strong> the
            question? <em>(Ви хочете, </em><strong>щоб </strong><strong>я </strong
            ><strong>відповів </strong><em> на це питання?)</em>
          </p>
          <p>
            I want <strong>you </strong><strong>to </strong><strong>repeat </strong> the
            question. <em>(Я хочу, </em><strong>щоб </strong><strong>ви </strong
            ><strong>повторили </strong><em> це питання.)</em>
          </p>
        </container>
    </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/IntroductionYourselfView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
    <main>
        <container>
            <h1>1. Casual Introductions</h1>

            <autotranslate-text>In casual settings, you can keep your introduction brief and friendly. Here are
a few
                examples:
            </autotranslate-text>
            <br>
            <autotranslate-text>Example 1: Meeting someone new at a social event</autotranslate-text>
            <br>
            <autotranslate-text>Hi, I’m [Your Name]. Nice to meet you!</autotranslate-text>
            <autotranslate-text>Hey, I’m [Your Name]. How’s it going?</autotranslate-text>
            <br>
            <autotranslate-text>Example 2: Introducing yourself to a group of friends</autotranslate-text>
            <br>
            <autotranslate-text>Hey everyone, I’m [Your Name].</autotranslate-text>
            <autotranslate-text>Hi guys, I’m [Your Name].</autotranslate-text>

            <h2>2. Professional Introductions</h2>

            <autotranslate-text>In professional settings, it's important to provide more context about yourself,
such as
                your job title,
                company, and possibly a brief description of your role.
            </autotranslate-text>
            <br>
            <autotranslate-text>Example 1: At a networking event</autotranslate-text>
            <br>
```

<autotranslate-text>Hello, I’m [Your Name], and I work as a [Your Job Title] at [Your Company].</autotranslate-text>
<autotranslate-text>Hi, I’m [Your Name]. I’m the [Your Job Title] at [Your Company], where I [briefly describe what you do].</autotranslate-text>

<autotranslate-text>Example 2: During a meeting</autotranslate-text>

<autotranslate-text>Good morning/afternoon, everyone. My name is [Your Name], and I’m the [Your Job Title] at [Your Company].
Hello, I’m [Your Name], and I lead the [Your Department] at [Your Company].
</autotranslate-text>

<h2>3. Written Introductions</h2>

<autotranslate-text>When introducing yourself in writing, such as in emails or letters, it's important to be clear and concise.
Here are some examples:
</autotranslate-text>

<autotranslate-text>Example 1: Email introduction</autotranslate-text>

<autotranslate-text>Subject: Introduction – [Your Name]</autotranslate-text>

<autotranslate-text>Dear [Recipient’s Name],</autotranslate-text>

<autotranslate-text>My name is [Your Name], and I am [Your Job Title] at [Your Company]. I am writing to introduce myself and to discuss [the purpose of your email].
</autotranslate-text>

<autotranslate-text>Looking forward to connecting with you.</autotranslate-text>

<autotranslate-text>Best regards,</autotranslate-text>

<autotranslate-text>[Your Name]</autotranslate-text>

<autotranslate-text>Example 2: Social media or forum introduction</autotranslate-text>

<autotranslate-text>Hi everyone,</autotranslate-text>

<autotranslate-text>I’m [Your Name], and I work as a [Your Job Title] at [Your Company]. I’m excited to join this group and look forward to sharing insights and learning from all of you.
</autotranslate-text>

<autotranslate-text>Best,</autotranslate-text>

<autotranslate-text>[Your Name]</autotranslate-text>

<h2>4. Self-Introduction in an Interview</h2>

```
        <autotranslate-text>In an interview, you need to provide a bit more detail about your background
and why you
            are interested in the position.
        </autotranslate-text>
        <br>
        <autotranslate-text>Example: Job interview</autotranslate-text>
        <br>
        <autotranslate-text>Thank you for having me today. My name is [Your Name], and I have a
background in [Your
            Field]. I have been
            working as a [Your Job Title] for [X years], where I [briefly describe your responsibilities and
            achievements]. I’m excited about this opportunity because [explain why you’re interested in the
            position/company].
        </autotranslate-text>
        <br>
        <autotranslate-text>Tips for Introducing Yourself</autotranslate-text>
        <br>
        <autotranslate-text>Be Confident: Speak clearly and make eye contact.
            Smile: A friendly smile goes a long way in making a positive impression.
            Keep It Brief: Provide enough information to be informative, but don’t overdo it.
            Adapt to the Context: Tailor your introduction based on the setting and the people you’re
meeting.
            Practice: The more you practice, the more natural your introduction will become.
        </autotranslate-text>
        <br>
        <autotranslate-text>By following these guidelines, you can introduce yourself effectively in
various
            situations.</autotranslate-text>
            <br><br>
      </container>
   </main>
</template>

<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/IrregularVerbsView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
</script>
<template>
   <main>
      <h1>Irregular Verbs</h1>
      <br />
      <container>
         <pretty-table>
            <thead>
               <tr>
                  <th>Base Form</th>
                  <th>Past Tense</th>
                  <th>Past Participle</th>
```

```
            </tr>
          </thead>
          <tbody>
            <tr v-for="irregularVerb in getIrregularVerbs" :key="irregularVerb.id">
              <td>{{ irregularVerb.baseForm }}</td>
              <td>{{ irregularVerb.pastTense }}</td>
              <td>{{ irregularVerb.pastParticiple }}</td>
            </tr>
          </tbody>
        </pretty-table>
      </container>
    </main>
</template>
<script lang="ts">
import { mapGetters } from "vuex"
export default {
  computed: mapGetters(["getIrregularVerbs"]),
}
</script>
```

Файл “src/frontend/src/views/LevelsView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
</script>
<template>
  <main>
    <container>
      <h1>Levels</h1>
      <br />
      <autotranslate-text>Language proficiency levels are displayed using generally recognized
international
        scales such as the CEFR (Common European Framework of Reference for Languages). This
scale describes
        speaking skills on a scale from A1 (beginning level) to C2 (advanced level).</autotranslate-
text>
      <br>
      <pretty-table>
        <tbody>
          <tr>
            <td>English levels</td>
            <td>About</td>
          </tr>
          <tr>
            <td>A1</td>
            <td>
              <autotranslate-text>Understanding and using basic phrases and
                expressions.</autotranslate-text>
              <autotranslate-text>Ability to introduce yourself and tell about your
                profile.</autotranslate-text>
              <autotranslate-text>Ability to understand simple sentences and expressions related to
                specific situations (for example, in a cafe or in a store).
              </autotranslate-text>
            </td>
          </tr>
          <tr>
            <td>A2</td>
            <td>
              <autotranslate-text>
                Increased vocabulary and the ability to express simple thoughts.
              </autotranslate-text>
              <autotranslate-text>Ability to understand basic instructions and expressions in everyday
                situations.</autotranslate-text>
              <autotranslate-text>Ability to form simple sentences and short texts.</autotranslate-
text>
            </td>
          </tr>
          <tr>
            <td>B1</td>
            <td>
              <autotranslate-text>Ability to express personal thoughts and ideas on simple
                topics.</autotranslate-text>
              <autotranslate-text>Understanding the general gist of messages and
                conversations.</autotranslate-text>
```

```
                    <autotranslate-text>Ability to interact with native speakers in everyday
                      situations.</autotranslate-text>
                  </td>
                </tr>
                <tr>
                  <td>B2</td>
                  <td>
                    <autotranslate-text>The ability to express thoughts clearly and
                      developed.</autotranslate-text>
                    <autotranslate-text>Understanding a large volume of text on various
                      topics.</autotranslate-text>
                    <autotranslate-text>Ability to conduct discussions and argue one's point of
                      view.</autotranslate-text>
                  </td>
                </tr>
                <tr>
                  <td>C1</td>
                  <td>
                    <autotranslate-text>High level of understanding and use of language in different
                      contexts.</autotranslate-text>
                    <autotranslate-text>Ability to read and analyze complex texts, including scientific
                      articles.</autotranslate-text>
                    <autotranslate-text>Ability to conduct professional conversations and
                      presentations.</autotranslate-text>
                  </td>
                </tr>
                <tr>
                  <td>C2</td>
                  <td>
                    <autotranslate-text>Perfect command of the language, both in conversation and in
writing.</autotranslate-text>
                    <autotranslate-text>Ability to understand complex academic and professional
content.</autotranslate-text>
                    <autotranslate-text>A high level of language use to express complex
ideas.</autotranslate-text>
                  </td>
                </tr>
              </tbody>
            </pretty-table>
          </container>
        </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/LoginView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import CenteredContent from '../components/CenteredContent.vue'
import InputLabel from '../components/InputLabel.vue'
import PrettyButton from '../components/PrettyButton.vue'
import { RouterLink } from "vue-router"
</script>
<template>
   <main>
      <container>
         <centered-content v-if="getIsAuth">
            <h1>You successful login!</h1>
            <router-link to="/profile">
               <pretty-button>Go to profile</pretty-button>
            </router-link>
         </centered-content>
         <form @submit="onSubmit" v-else>
            <h1>Login</h1>
            <centered-content>
               <input-label for="username">Username</input-label>
               <input class="pretty" type="text" v-model="username" maxlength="64" minlength="3"
required>
               <input-label for="password">Password</input-label>
               <input class="pretty" type="password" v-model="password" maxlength="64" minlength="8"
required>
               <br><br>
               <pretty-button type="submit">Login!</pretty-button>
            </centered-content>
         </form>
      </container>
   </main>
</template>

<script lang="ts">
import { mapActions, mapGetters, mapMutations } from 'vuex'

export default {
   data: () => {
      return {
         username: new String,
         password: new String
      }
   },
   computed: {
      ...mapGetters(["getIsAuth"])
   },
   methods: {
      ...mapActions(["fetchLoginUser"]),
      onSubmit(event: Event) {
         event.preventDefault()
         const data = {
            username: this.username,
            password: this.password,
```

```
        }
        this.fetchLoginUser(data).then(() => {
          this.clearFields()
        })
      },
      clearFields() {
        this.username = ""
        this.password = ""
      }
    }
}
</script>
```

Файл “src/frontend/src/views/ModalVerbsView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
</script>
<template>
  <main>
    <h1>Modal Verbs</h1>
    <br />
    <container>
      <autotranslate-text>Modal verbs in English essentially express possibility, not
        the possibility of some action, probability or improbability, obligation, necessity,
        desire, doubt, etc. Most modal verbs in English have more
        than one value. Each of the meanings is characterized by the specifics of the use of this word
        another modal verb.
      </autotranslate-text>
      <br>
      <pretty-table cellspacing="0" cellpadding="2">
        <thead>
          <tr>
            <th>
              <p>Модальне дієслово</p>
            </th>
            <th>
              <p>можливий переклад</p>
            </th>
            <th>
              <p>загальне значення</p>
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>
              <p>can</p>
            </td>
            <td>
              <p>могти</p>
            </td>
            <td>
```

```
        <p>
          Передає значення можливості або дозволу виконання тієї чи іншої дії.
        </p>
      </td>
    </tr>
    <tr>
      <td>
        <p>may</p>
      </td>
      <td>
        <p>могти</p>
      </td>
      <td>
        <p>
          Використовується у значенні ймовірності або дозволу. Дієслово may
          має більший ступінь ймовірності ніж can і є більш формальним.
        </p>
      </td>
    </tr>
    <tr>
      <td>
        <p>could</p>
      </td>
      <td>
        <p>міг, могла, могли</p>
      </td>
      <td>
        <p>
          Виражає упевненість в чомусь, але може також висловлювати
          необхідність, пораду або наполегливу рекомендацію.
        </p>
      </td>
    </tr>
    <tr>
      <td>
        <p>must</p>
      </td>
      <td>
        <p>повинен, зобов&rsquo;язаний, мусиш</p>
      </td>
      <td>
        <p>
          Виражає упевненість в чомусь, але може також висловлювати
          необхідність, пораду або наполегливу рекомендацію.
        </p>
      </td>
    </tr>
    <tr>
      <td>
        <p>would</p>
      </td>
      <td>
        <p>&ldquo;не могли б ви &hellip;&rdquo;</p>
      </td>
      <td>
        <p>Має значення ввічливого спонукання.</p>
```

```
                </td>
              </tr>
              <tr>
                <td>
                  <p>should</p>
                </td>
                <td>
                  <p>&amp;ldquo;слід щось зробити&amp;ldquo;</p>
                </td>
                <td>
                  <p>
                    Має загальне значення необов&amp;rsquo;язкової для виконання
                    рекомендації.
                  </p>
                </td>
              </tr>
            </tbody>
          </pretty-table>
          <br />
          <autotranslate-text>Unlike other verbs, modal verbs do not indicate action or state; they
              indicate the speaker's relationship to the action referred to in the sentence.
          </autotranslate-text>
          <br>
          <autotranslate-text>You <strong>can</strong> find it here. <em>(Ви
</em><strong>можете</strong
            ><em> знайти це тут.)</em>
          </autotranslate-text>
          <autotranslate-text><strong>Can</strong> I come in?
<em>(</em><strong>Можна</strong><em> увійти?)</em>
          </autotranslate-text>
          <autotranslate-text>I <strong>cannot</strong> live without water. <em>(Я не
</em><strong>можу</strong
            ><em> жити без води.)</em>
          </autotranslate-text>
          <autotranslate-text><strong>May</strong> I speak to Pedro?
<em>(</em><strong>Можна</strong
            ><em> поговорити з Педро?)</em>
          </autotranslate-text>
          <autotranslate-text><strong>Could</strong> you help me? <em>(Чи </em><strong>не</strong
            ><strong>могли</strong><strong>б</strong><em> ви мені допомогти?)</em>
          </autotranslate-text>
          <autotranslate-text>Children <strong>must</strong> eat. <em>(Діти
</em><strong>повинні</strong
            ><em> їсти.)</em>
          </autotranslate-text>
          <autotranslate-text>I <strong>would</strong><strong>be</strong> there if I could. <em>(Я </em
            ><strong>був</strong><strong>би</strong><em> там, якби міг.)</em>
          </autotranslate-text>
          <autotranslate-text>We <strong>should</strong> drink more water. <em>(Нам
</em><strong>слід</strong
            ><em> пити більше води.)</em>
          </autotranslate-text>
        </container>
      </main>
</template>
```

```
<script lang="ts">
export default {

}
</script>
```

Файл “src/frontend/src/views/NumbersView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
import CenteredContent from '../components/CenteredContent.vue'
import PrettyLabel from '../components/PrettyLabel.vue'
</script>
<style scoped>
.answerPlace {
   height: 30px;
}
.form-base {
   display: flex;
   justify-content: space-between;
   width: 660px;
   margin: 0 auto;
   align-items: center;
}
</style>
<template>
   <main>
      <container>
         <centered-content>
            <h1>Numbers</h1>
            <br />
            <p>The numbers from 0 to 100 are:</p>
            <pretty-table>
               <thead>
                  <tr>
                     <th>Number</th>
                     <th>Word</th>
                  </tr>
               </thead>
               <tbody>
                  <tr v-for="i in createRange(0, 101)" :key="i">
                     <td>{{ i }}</td>
                     <td>{{ numToWords(i) }}</td>
                  </tr>
               </tbody>
            </pretty-table>
            <h2><autotranslate-text>Fractions</autotranslate-text></h2>
            <br>
            <pretty-table>
               <tbody>
                  <tr>
                     <th>Fraction</th>
                     <th>in English</th>
```

```
        </tr>
        <tr>
          <td>1/2</td>
          <td>
            <div>a half o half</div>
          </td>
        </tr>
        <tr>
          <td>5/2</td>
          <td>
            <div>five halves</div>
          </td>
        </tr>
        <tr>
          <td>1/3</td>
          <td>
            <div>one third</div>
          </td>
        </tr>
        <tr>
          <td>1/5</td>
          <td>
            <div>one fifth</div>
          </td>
        </tr>
        <tr>
          <td>3/7</td>
          <td>
            <div>three sevenths</div>
          </td>
        </tr>
        <tr>
          <td>45/9</td>
          <td>
            <div>forty-five ninths</div>
          </td>
        </tr>
      </tbody>
    </pretty-table>
    <h2><autotranslate-text>Percentages</autotranslate-text></h2>
    <pretty-table>
      <tbody>
        <tr>
          <th>Percentage</th>
          <th>in English</th>
        </tr>
        <tr>
          <td>0.05%</td>
          <td>
            <div>zero point zero five percent</div>
          </td>
        </tr>
        <tr>
          <td>1.7%</td>
          <td>
            <div>one point seven percent</div>
```

```
            </td>
         </tr>
         <tr>
            <td>5.4%</td>
            <td>
               <div>five point four percent</div>
            </td>
         </tr>
         <tr>
            <td>25%</td>
            <td>
               <div>twenty five percent</div>
            </td>
         </tr>
         <tr>
            <td>300%</td>
            <td>
               <div>three hundred percent</div>
            </td>
         </tr>
      </tbody>
   </pretty-table>
   <h2><autotranslate-text>Money</autotranslate-text></h2>
   <pretty-table>
      <tbody>
         <tr>
            <th>Money</th>
            <th>in English</th>
         </tr>
         <tr>
            <td>1$</td>
            <td>
               <div>one dollar</div>
            </td>
         </tr>
         <tr>
            <td>€15</td>
            <td>
               <div>fifteen euros</div>
            </td>
         </tr>
         <tr>
            <td>£20</td>
            <td>
               <div>twenty pounds</div>
            </td>
         </tr>
         <tr>
            <td>$3.99</td>
            <td>
               <div>three dollar and ninety nine cents</div>
            </td>
         </tr>
      </tbody>
   </pretty-table>
   <br />
```

```
        </centered-content>
        <br>
        <div class="form-base">
          <div>
            <pretty-label for="numInput">Input number: </pretty-label>
            <input class="pretty" name="numInput" id="numInput" v-model="num" type="number"
max="999999"
              min="-999999" maxlength="6" placeholder="Input number" />
          </div>
          <div>
            <pretty-label for="splitAsYear">Split as year </pretty-label>
            <input class="pretty" type="checkbox" id="splitAsYear" name="splitAsYear" v-
model="splitAsYear" />
          </div>
        </div>
        <br />
        <div class="answerPlace"></div>
      </container>
  </main>
</template>
<script lang="ts">
export default {
  data: () => {
    return {
      num: new Number,
      splitAsYear: false,
    }
  },
  watch: {
    num(newValue: number) {
      this.updateAnswer(newValue)
    },
    splitAsYear() {
      this.updateAnswer(this.num)
    }
  },
  methods: {
    updateAnswer(newValue) {
      newValue = Math.abs(newValue)
      let result: string
      if (this.splitAsYear) {
        const value1: number = Number.parseInt(newValue.toString().slice(0, 2))
        const value2: number = Number.parseInt(newValue.toString().slice(-2))
        const result1: string = this.numToWords(value1)
        const result2: string = this.numToWords(value2)
        result = result1 + " " + result2
      } else
        result = this.numToWords(newValue)
      const answerPlace: any = document.querySelector(".answerPlace")
      answerPlace.innerText = result
    },
    numToWords(num: number): string {
      const ones: string[] = ["zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine",
"ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"]
      const tens: string[] = ["", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty",
"ninety"]
```

```
        let words: string = ""
        if (num >= 1000) {
          words += ones[Math.floor(num / 1000)] + " thousand"
          num %= 1000
          if (num > 0)
            words += " and "
        }
        if (num >= 100) {
          words += ones[Math.floor(num / 100)] + " hundred"
          num %= 100
          if (num > 0)
            words += " and "
        }
        if (num >= 20) {
          words += tens[Math.floor(num / 10)]
          num %= 10
          if (num > 0)
            words += "-" + ones[num]
        } else if (num > 0) {
          words += ones[num]
        }
        return words
      },
      *createRange(start = 0, end = Infinity, step = 1) {
        let iterationCount = 0
        for (let i = start; i < end; i += step) {
          iterationCount++
          yield i
        }
        return iterationCount
      }
    }
}
</script>
```

Файл “src/frontend/src/views/PerfectProgressiveTenseView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
import PrettyLabel from '../components/PrettyLabel.vue'
</script>
<template>
  <main>
    <h1>Perfect Progressive Tense</h1>
    <br />
    <container>
      <autotranslate-text>The perfect progressive tense is a category of verb tense used to mark the end
of an
        ongoing action. It covers the past perfect progressive tense, the present prefect
        progressive tense, and the future perfect progressive tense. Of note, the past
        perfect progressive tense and the future perfect progressive tense are used to mark
        the end of an ongoing action. However, present perfect progressive tense is used for
        actions that began in the past and continue into the present. The perfect
```

```
    progressive tenses are recognizable by the word 'have' (in one of its forms),
    'been,' and a present participle (i.e., the word that ends '-ing'). Here are some
    examples of verbs in the perfect progressive tenses.
</autotranslate-text>
<br>
<pretty-table>
   <tbody>
      <tr>
         <td>
            <p>Question(?)</p>
         </td>
         <td>
            <p>Affirmation(+)</p>
         </td>
         <td>
            <p>Negation(-)</p>
         </td>
         <td>
            <p>Times</p>
         </td>
      </tr>
      <tr>
         <td>
            <p>Will I have been {{ verb }}ing?</p>
            <p>Will You have been {{ verb }}ing?</p>
            <p>Will We have been {{ verb }}ing?</p>
            <p>Will They have been {{ verb }}ing?</p>
            <p>Will He/She/it have been {{ verb }}ing?</p>
         </td>
         <td>
            <p>I will have been {{ verb }}ing</p>
            <p>You will have been {{ verb }}ing</p>
            <p>We will have been {{ verb }}ing</p>
            <p>They will have been {{ verb }}ing</p>
            <p>He/She/it will have been {{ verb }}ing</p>
         </td>
         <td>
            <p>I will have not been {{ verb }}ing</p>
            <p>You will have not been {{ verb }}ing</p>
            <p>We will have not been {{ verb }}ing</p>
            <p>They will have not been {{ verb }}ing</p>
            <p>He/She/it will have not been {{ verb }}ing</p>
         </td>
         <td>Future</td>
      </tr>
      <tr>
         <td>
            <p>Have I been {{ verb }}ing?</p>
            <p>Have You been {{ verb }}ing?</p>
            <p>Have We been {{ verb }}ing?</p>
            <p>Have They been {{ verb }}ing?</p>
            <p>Has He/She/it been {{ verb }}ing?</p>
         </td>
         <td>
            <p>I have been {{ verb }}ing</p>
            <p>You have been {{ verb }}ing</p>
```

```
            <p>We have been {{ verb }}ing</p>
            <p>They have been {{ verb }}ing</p>
            <p>He/She/it has been {{ verb }}ing</p>
          </td>
          <td>
            <p>I have not been {{ verb }}ing</p>
            <p>You have not been {{ verb }}ing</p>
            <p>We have not been {{ verb }}ing</p>
            <p>They have not been {{ verb }}ing</p>
            <p>He/She/it has not been {{ verb }}ing</p>
          </td>
          <td>Present</td>
        </tr>
        <tr>
          <td>
            <p>Had I been {{ verb }}ing?</p>
            <p>Had You been {{ verb }}ing?</p>
            <p>Had We been {{ verb }}ing?</p>
            <p>Had They been {{ verb }}ing?</p>
            <p>Had He/She/it been {{ verb }}ing?</p>
          </td>
          <td>
            <p>I had been {{ verb }}ing</p>
            <p>You had been {{ verb }}ing</p>
            <p>We had been {{ verb }}ing</p>
            <p>They had been {{ verb }}ing</p>
            <p>He/She/it had been {{ verb }}ing</p>
          </td>
          <td>
            <p>I had not been {{ verb }}ing</p>
            <p>You had not been {{ verb }}ing</p>
            <p>We had not been {{ verb }}ing</p>
            <p>They had not been {{ verb }}ing</p>
            <p>He/She/it had not been {{ verb }}ing</p>
          </td>
          <td>Past</td>
        </tr>
      </tbody>
    </pretty-table>
    <br />
    <pretty-label for="verbInput">Input Word: </pretty-label>
    <input
      name="verbInput"
      id="verbInput"
      type="text"
      class="pretty"
      placeholder="Слово англ. мовою"
      v-model="verb"
    />
    <br><br>
    <p>
      Example: <br />I have been working for 2 hours. <br />I had been working for 2 hours
      by 9 pm yesterday. <br />I will have been working for 2 hours by 9 pm tomorrow.
      <br />I have already been learning English for 5 years. <br />I had already been
      learning English for 5 years before I finished school. <br />I will have already
      been learning English for 5 years by 2030.
```

```
            </p>
        </container>
    </main>
</template>


<script lang="ts">
export default {
    data: () => {
        return {
            verb: "work",
        }
    }
}
</script>
```

Файл “src/frontend/src/views/PerfectTenseView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
import PrettyLabel from '../components/PrettyLabel.vue'
</script>
<template>
    <main>
        <h1>Perfect Tense</h1>
        <br />
        <container>
            <autotranslate-text>Perfect tense is a category of verb tense used to describe completed actions. It
                covers the past perfect tense, the present perfect tense, and the future perfect
                tense. It is sometimes called the complete tense. The perfect tenses are formed
                using a form of the auxiliary verb 'to have' and the past participle.
            </autotranslate-text>
            <br>
            <pretty-table>
                <tbody>
                    <tr>
                        <td>
                            <p>Question(?)</p>
                        </td>
                        <td>
                            <p>Affirmation(+)</p>
                        </td>
                        <td>
                            <p>Negation(-)</p>
                        </td>
                        <td>
                            <p>Times</p>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <p>Will I have {{ verbV3 }}(V3)?</p>
                            <p>Will You have {{ verbV3 }}(V3)?</p>
```

```
        <p>Will We have {{ verbV3 }}(V3)?</p>
        <p>Will They have {{ verbV3 }}(V3)?</p>
        <p>Will He/She/it have {{ verbV3 }}(V3)?</p>
      </td>
      <td>
        <p>I will have {{ verbV3 }}(V3)</p>
        <p>You will have {{ verbV3 }}(V3)</p>
        <p>We will have {{ verbV3 }}(V3)</p>
        <p>They will have {{ verbV3 }}(V3)</p>
        <p>He/She/it will have {{ verbV3 }}(V3)</p>
      </td>
      <td>
        <p>I will not have {{ verbV3 }}(V3)</p>
        <p>You will not have {{ verbV3 }}(V3)</p>
        <p>We will not have {{ verbV3 }}(V3)</p>
        <p>They will not have {{ verbV3 }}(V3)</p>
        <p>He/She/it will not have {{ verbV3 }}(V3)</p>
      </td>
      <td>Future</td>
    </tr>
    <tr>
      <td>
        <p>Have I {{ verbV3 }}(V3)?</p>
        <p>Have You {{ verbV3 }}(V3)?</p>
        <p>Have We {{ verbV3 }}(V3)?</p>
        <p>Have They {{ verbV3 }}(V3)?</p>
        <p>Has He/She/it {{ verbV3 }}(V3)?</p>
      </td>
      <td>
        <p>I have {{ verbV3 }}(V3)</p>
        <p>You have {{ verbV3 }}(V3)</p>
        <p>We have {{ verbV3 }}(V3)</p>
        <p>They have {{ verbV3 }}(V3)</p>
        <p>He/She/it has {{ verbV3 }}(V3)</p>
      </td>
      <td>
        <p>I have not {{ verbV3 }}(V3)</p>
        <p>You have not {{ verbV3 }}(V3)</p>
        <p>We have not {{ verbV3 }}(V3)</p>
        <p>They have not {{ verbV3 }}(V3)</p>
        <p>He/She/it has not {{ verbV3 }}(V3)</p>
      </td>
      <td>Present</td>
    </tr>
    <tr>
      <td>
        <p>Had I {{ verbV3 }}(V3)?</p>
        <p>Had You {{ verbV3 }}(V3)?</p>
        <p>Had We {{ verbV3 }}(V3)?</p>
        <p>Had They {{ verbV3 }}(V3)?</p>
        <p>Had He/She/it {{ verbV3 }}(V3)?</p>
      </td>
      <td>
        <p>I had {{ verbV3 }}(V3)</p>
        <p>You had {{ verbV3 }}(V3)</p>
        <p>We had {{ verbV3 }}(V3)</p>
```

```
                    <p>They had {{ verbV3 }}(V3)</p>
                    <p>He/She/it had {{ verbV3 }}(V3)</p>
                  </td>
                  <td>
                    <p>I had not {{ verbV3 }}(V3)</p>
                    <p>You had not {{ verbV3 }}(V3)</p>
                    <p>We had not {{ verbV3 }}(V3)</p>
                    <p>They had not {{ verbV3 }}(V3)</p>
                    <p>He/She/it had not {{ verbV3 }}(V3)</p>
                  </td>
                  <td>Past</td>
                </tr>
              </tbody>
            </pretty-table>
            <br />
            <pretty-label for="verbInput">Input Word: </pretty-label>
            <input
              name="verbInput"
              id="verbInput"
              type="text"
              class="pretty"
              placeholder="Слово англ. мовою"
              v-model="verb"
            />
            <br><br>
            <p>
              Example: <br />
              I have already working.
              <br />I had already working by 9 pm yesterday. <br />I will have already working by 9
              pm tomorrow. <br />What had you written before I called you? <br />What will you
              have written by 5 p.m. tomorrow? <br />I have already (learnt) English. <br />I had
              already (learnt) English before / when I finished school. <br />I will have already
              (learnt) English by 2021. <br />I have not finished my project yet.
            </p>
          </container>
        </main>
      </template>
      <script lang="ts">
      import { mapActions } from 'vuex'
      export default {
        data: () => {
          return {
            verb: "work",
            verbV3: "worked",
          }
        },
        methods: {
          ...mapActions(["getVerbInV3"]),
        },
        watch: {
          async verb(newValue) {
            await this.getVerbInV3(newValue).then((verbV3: string) => {
              this.verbV3 = verbV3
            })
          }
        },
```

```
}
</script>
```

Файл “src/frontend/src/views/PerfectTenseView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
import PrettyLabel from '../components/PrettyLabel.vue'
</script>
<template>
  <main>
    <h1>Perfect Tense</h1>
    <br />
    <container>
      <autotranslate-text>Perfect tense is a category of verb tense used to describe completed actions. It
        covers the past perfect tense, the present perfect tense, and the future perfect
        tense. It is sometimes called the complete tense. The perfect tenses are formed
        using a form of the auxiliary verb 'to have' and the past participle.
      </autotranslate-text>
      <br>
      <pretty-table>
        <tbody>
          <tr>
            <td>
              <p>Question(?)</p>
            </td>
            <td>
              <p>Affirmation(+)</p>
            </td>
            <td>
              <p>Negation(-)</p>
            </td>
            <td>
              <p>Times</p>
            </td>
          </tr>
          <tr>
            <td>
              <p>Will I have {{ verbV3 }}(V3)?</p>
              <p>Will You have {{ verbV3 }}(V3)?</p>
              <p>Will We have {{ verbV3 }}(V3)?</p>
              <p>Will They have {{ verbV3 }}(V3)?</p>
              <p>Will He/She/it have {{ verbV3 }}(V3)?</p>
            </td>
            <td>
              <p>I will have {{ verbV3 }}(V3)</p>
              <p>You will have {{ verbV3 }}(V3)</p>
              <p>We will have {{ verbV3 }}(V3)</p>
              <p>They will have {{ verbV3 }}(V3)</p>
              <p>He/She/it will have {{ verbV3 }}(V3)</p>
            </td>
            <td>
              <p>I will not have {{ verbV3 }}(V3)</p>
```

```
            <p>You will not have {{ verbV3 }}(V3)</p>
            <p>We will not have {{ verbV3 }}(V3)</p>
            <p>They will not have {{ verbV3 }}(V3)</p>
            <p>He/She/it will not have {{ verbV3 }}(V3)</p>
        </td>
        <td>Future</td>
    </tr>
    <tr>
        <td>
            <p>Have I {{ verbV3 }}(V3)?</p>
            <p>Have You {{ verbV3 }}(V3)?</p>
            <p>Have We {{ verbV3 }}(V3)?</p>
            <p>Have They {{ verbV3 }}(V3)?</p>
            <p>Has He/She/it {{ verbV3 }}(V3)?</p>
        </td>
        <td>
            <p>I have {{ verbV3 }}(V3)</p>
            <p>You have {{ verbV3 }}(V3)</p>
            <p>We have {{ verbV3 }}(V3)</p>
            <p>They have {{ verbV3 }}(V3)</p>
            <p>He/She/it has {{ verbV3 }}(V3)</p>
        </td>
        <td>
            <p>I have not {{ verbV3 }}(V3)</p>
            <p>You have not {{ verbV3 }}(V3)</p>
            <p>We have not {{ verbV3 }}(V3)</p>
            <p>They have not {{ verbV3 }}(V3)</p>
            <p>He/She/it has not {{ verbV3 }}(V3)</p>
        </td>
        <td>Present</td>
    </tr>
    <tr>
        <td>
            <p>Had I {{ verbV3 }}(V3)?</p>
            <p>Had You {{ verbV3 }}(V3)?</p>
            <p>Had We {{ verbV3 }}(V3)?</p>
            <p>Had They {{ verbV3 }}(V3)?</p>
            <p>Had He/She/it {{ verbV3 }}(V3)?</p>
        </td>
        <td>
            <p>I had {{ verbV3 }}(V3)</p>
            <p>You had {{ verbV3 }}(V3)</p>
            <p>We had {{ verbV3 }}(V3)</p>
            <p>They had {{ verbV3 }}(V3)</p>
            <p>He/She/it had {{ verbV3 }}(V3)</p>
        </td>
        <td>
            <p>I had not {{ verbV3 }}(V3)</p>
            <p>You had not {{ verbV3 }}(V3)</p>
            <p>We had not {{ verbV3 }}(V3)</p>
            <p>They had not {{ verbV3 }}(V3)</p>
            <p>He/She/it had not {{ verbV3 }}(V3)</p>
        </td>
        <td>Past</td>
    </tr>
</tbody>
```

```
            </pretty-table>
            <br />
            <pretty-label for="verbInput">Input Word: </pretty-label>
            <input
               name="verbInput"
               id="verbInput"
               type="text"
               class="pretty"
               placeholder="Слово англ. мовою"
               v-model="verb"
            />
            <br><br>
            <p>
               Example: <br />
               I have already working.
               <br />I had already working by 9 pm yesterday. <br />I will have already working by 9
               pm tomorrow. <br />What had you written before I called you? <br />What will you
               have written by 5 p.m. tomorrow? <br />I have already (learnt) English. <br />I had
               already (learnt) English before / when I finished school. <br />I will have already
               (learnt) English by 2021. <br />I have not finished my project yet.
            </p>
         </container>
      </main>
   </template>
   <script lang="ts">
   import { mapActions } from 'vuex'
   export default {
      data: () => {
         return {
            verb: "work",
            verbV3: "worked",
         }
      },
      methods: {
         ...mapActions(["getVerbInV3"]),
      },
      watch: {
         async verb(newValue) {
            await this.getVerbInV3(newValue).then((verbV3: string) => {
               this.verbV3 = verbV3
            })
         }
      },
   }
   </script>
```

Файл “src/frontend/src/views/PersonalPronounsView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
</script>
<template>
   <main>
```

```
<h1>Personal Pronouns</h1>
<br />
<container>
  <autotranslate-text>Personal pronouns in English refer to a person or an object in a sentence. They have two case forms:</autotranslate-text>
  <br>
  <autotranslate-text>Nominative Case indicates the doer of the action.</autotranslate-text>
  <autotranslate-text>For example, "I", "he", "she".</autotranslate-text>
  <br>
  <autotranslate-text>Object case shows the recipient of the action.
  </autotranslate-text>
  <autotranslate-text>For example, "me", "him", "her".</autotranslate-text>
  <br>
  <autotranslate-text>Do not confuse the pronoun "her" (her) with the word "her" in the possessive form.</autotranslate-text>
  <autotranslate-text>For example, "Is it her?"</autotranslate-text>
  <autotranslate-text>(Is she?) different from "It is her!" (It's hers!).</autotranslate-text>
  <br>
  <autotranslate-text>"You" has the same form in both the nominative and the object case.</autotranslate-text>
  <autotranslate-text>For example, "You eat cheese", "He reads you the menu".</autotranslate-text><br><br>
  <pretty-table>
    <thead>
      <tr>
        <th>
          <p>Називний відмінок</p>
        </th>
        <th>
          <p>Називає виконувача дії у реченні</p>
        </th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>
          <p><strong>I</strong> (<em>я</em>)</p>
        </td>
        <td>
          <p><strong>I eat an egg</strong> (<em>Я їм яйце</em>)</p>
        </td>
      </tr>
      <tr>
        <td>
          <p><strong>we</strong> (<em>ми</em>)</p>
        </td>
        <td>
          <p><strong>We drink tea</strong> (<em>Ми п'ємо чай</em>)</p>
        </td>
      </tr>
      <tr>
        <td>
          <p><strong>you</strong> (<em>ти/ви</em>)</p>
        </td>
        <td>
          <p><strong>You don't cook</strong> (<em>Ти не готуєш</em>)</p>
```

```
            </td>
          </tr>
          <tr>
            <td>
              <p><strong>he</strong> (<em>він</em>)</p>
            </td>
            <td>
              <p><strong>He sleeps here</strong> (<em>Він спить тут</em>)</p>
            </td>
          </tr>
          <tr>
            <td>
              <p><strong>she</strong> (<em>вона</em>)</p>
            </td>
            <td>
              <p><strong>She knows</strong> (<em>Вона знає</em>)</p>
            </td>
          </tr>
          <tr>
            <td>
              <p><strong>it</strong> (<em>воно</em>)</p>
            </td>
            <td>
              <p><strong>It doesn't work</strong> (<em>Воно/це не працює</em>)</p>
            </td>
          </tr>
          <tr>
            <td>
              <p><strong>they</strong> (<em>вони</em>)</p>
            </td>
            <td>
              <p><strong>They have time</strong> (<em>Вони мають час</em>)</p>
            </td>
          </tr>
        </tbody>
      </pretty-table>
      <br /><br /><br />
      <pretty-table>
        <thead>
          <tr>
            <th>
              <p>Об&rsquo;єктний відмінок</p>
            </th>
            <th>
              <p>Називає отримувача дії у реченні</p>
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>
              <p><strong>me</strong></p>
            </td>
            <td>
              <p><strong>Give me a pen</strong> (<em>Дай мені ручку</em>)</p>
            </td>
```

```
            </tr>
            <tr>
              <td>
                <p><strong>him</strong></p>
              </td>
              <td>
                <p><strong>I read him a book</strong> (<em>Я читаю йому книжку</em>)</p>
              </td>
            </tr>
            <tr>
              <td>
                <p><strong>her</strong></p>
              </td>
              <td>
                <p><strong>I know her</strong> (<em>Я знаю її</em>)</p>
              </td>
            </tr>
            <tr>
              <td>
                <p><strong>it</strong></p>
              </td>
              <td>
                <p><strong>I see it</strong> (<em>Я бачу його/це</em>)</p>
              </td>
            </tr>
            <tr>
              <td>
                <p><strong>us</strong></p>
              </td>
              <td>
                <p>
                  <strong>He reads us the menu</strong> (<em>Він читає нам меню</em>)
                </p>
              </td>
            </tr>
            <tr>
              <td>
                <p><strong>them</strong></p>
              </td>
              <td>
                <p><strong>I like them</strong> (<em>Вони мені подобаються</em>)</p>
              </td>
            </tr>
          </tbody>
        </pretty-table>
      </container>
    </main>
</template>

<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/PluralView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyLabel from '../components/PrettyLabel.vue';
</script>
<style scoped>
.answer-place {
   height: 30px;
}
</style>
<template>
   <main>
      <h1>Plural</h1>
      <br />
      <container>
         <autotranslate-text>The plural in English is usually formed by adding the ending -s (-es) to the
            noun:</autotranslate-text>
         <br>
         <autotranslate-text>For example: "boy" becomes "boys".</autotranslate-text>
         <br>
         <autotranslate-text>However, there are some exceptions where the plural is formed by a change
within the
            noun or by adding a
            special ending:
         </autotranslate-text>
         <br>
         <autotranslate-text>For example: "man" becomes "men".</autotranslate-text>
         <br>
         <autotranslate-text>There are also nouns that are used only in the plural, without a singular
            form:</autotranslate-text>
         <br>
         <autotranslate-text>For example: "pants", "clothes".</autotranslate-text>
         <br>
         <pretty-label for="nounInput">Input noun: </pretty-label>
         <input class="pretty" name="nounInput" id="nounInput" type="text" placeholder="Слово англ.
мовою" v-model="noun" />
         <br><br>
         <p class="answer-place">Маємо: {{ plural }}</p><br>
      </container>
   </main>
</template>
<script lang="ts">
export default {
   data: () => {
      return {
         noun: "child",
         plural: "children",
      }
   },
   watch: {
      noun(newValue) {
         if (newValue.length > 2)
            this.plural = this.getPlural(newValue)
      }
   },
   methods: {
```

```
        getPlural(noun: string) {
            noun = noun.toLowerCase()
            noun = noun.replace("a ", "").replace("an ", "")
            const lastCharacter: string = noun.slice(-1)
            const lastTwoCharacters: string = noun.slice(-2)
            // Nouns-Exceptions!
            switch (noun) {
                case "man":
                    return "men"
                case "woman":
                    return "women"
                case "child":
                    return "children"
            }
            if (lastCharacter === 's' || lastCharacter === 'x' || lastTwoCharacters === 'ss' || lastTwoCharacters
=== 'sh' || lastTwoCharacters === 'ch' || noun === 'hero') {
                // Remember! "pants" and "clothes" Always plural!
                if (noun !== "pants" && noun !== "clothes")
                    noun = noun + "es"
            }
            else
                noun = noun + "s"
            return noun
        }
    }
}
</script>
```

Файл “src/frontend/src/views/PossessiveFormView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
    <main>
        <h1>Possessive Form</h1>
        <br />
        <container>
            <autotranslate-text>The possessive form in English shows that something belongs to someone or
something.
            </autotranslate-text>
            <autotranslate-text>There are two forms:</autotranslate-text>
            <br>
            <autotranslate-text>Normal form: used with a noun before it.
            </autotranslate-text>
            <autotranslate-text>For example, "my book", "his cat".</autotranslate-text>
            <br>
            <autotranslate-text>Absolute Form: This is a pronoun without a noun that can be the subject,
object, or
                predicate in a sentence.
            </autotranslate-text>
            <autotranslate-text>For example, "The book is mine", "The cat is hers".</autotranslate-text>
            <br>
```

```
        <autotranslate-text>To form the possessive case, add an apostrophe and the letter "s" at the end.
For
            example, "the student's book", "the girl's cat".
        </autotranslate-text>
      </container>
   </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
</script>
<template>
   <main>
      <h1>Prepositions</h1>
      <br />
      <container>
         <autotranslate-text>Prepositions in English indicate the location of something or the characteristic
of an event:</autotranslate-text>
         <br>
         <autotranslate-text>IN - indicates the location inside a room, city or country. For example, "They
are in the kitchen" (They are in the kitchen).
         </autotranslate-text>
         <br>
         <autotranslate-text>ON - shows location on something. For example, "Apples are on the
plate".</autotranslate-text>
         <br>
         <autotranslate-text>AT - is used when talking about an event or place of stay. For example, "He is
at a meeting" (He is at a meeting).
         </autotranslate-text>
         <br>
         <autotranslate-text>FOR - means "for" or indicates a reward. For example, "I have something for
you" (I have something for you).</autotranslate-text>
         <br>
         <autotranslate-text>EXCEPT (FOR) - used for an exception. For example, "Except for the
weather nothing changed".</autotranslate-text>
         <br>
         <autotranslate-text>DURING and AS - show the duration or simultaneity of the event. For
example, "During lunch", "As I write".</autotranslate-text>
         <br>
         <autotranslate-text>BY and NEAR mean "near". For example, "The book is by the newspaper",
"She walks by me".</autotranslate-text>
         <br>
         <pretty-table>
            <tbody>
               <tr>
                  <td>Time</td>
                  <td>Preposition</td>
                  <td>Place</td>
```

```
            </tr>
            <tr>
              <td>
                in XX cent<br />
                in The 90's<br />
                in 2003<br />
                in April<br />
                in summer<br />
                in The morning<br />
              </td>
              <td>IN</td>
              <td>
                in Germany<br />
                in London<br />
                in Brooklyn<br />
                in my room/car<br />
              </td>
            </tr>
            <tr>
              <td>
                on New Years Day<br />
                on Thursday<br />
                on The 1st of June<br />
                on my Birthday<br />
                on Friday morning<br />
              </td>
              <td>ON</td>
              <td>
                on Green Street / Fifth Avenue<br />
                on The table<br />
                on a train/plane<br />
                on my website<br />
                on YouTube<br />
              </td>
            </tr>
            <tr>
              <td>
                at 7 am<br />
                at night<br />
                at Christmas<br />
              </td>
              <td>AT</td>
              <td>
                at 10 Green Street<br />
                at cafe/home/work<br />
              </td>
            </tr>
          </tbody>
        </pretty-table>
      </container>
    </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/ProfileView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyLabel from '../components/PrettyLabel.vue'
import PrettyButton from '../components/PrettyButton.vue'
import CenteredContent from '../components/CenteredContent.vue'
</script>

<template>
  <main>
    <container v-if="getIsAuth">
      <h1>Personal Information</h1>
      <form @submit="onSubmitPersonalInformation">
        <br>
        <div>
          <pretty-label for="username">Username:  </pretty-label>
          <input type="text" class="pretty" id="username" v-model="username" maxlength="64"
minlength="3"
            required>
        </div>
        <br>
        <div>
          <pretty-label for="firstName">First name:  </pretty-label>
          <input type="text" class="pretty" id="firstName" v-model="firstName" maxlength="64"
minlength="3">
        </div>
        <br>
        <div>
          <pretty-label for="lastName">Last name:  </pretty-label>
          <input type="text" class="pretty" id="lastName" v-model="lastName" maxlength="64"
minlength="3">
        </div>
        <br>
        <div>
          <pretty-label for="emailAddress">Email Address:  </pretty-label>
          <input type="email" class="pretty" id="emailAddress" v-model="emailAddress"
minlength="3"
            maxlength="319">
        </div>
        <br>
        <div>
          <pretty-label for="gender">Gender:  </pretty-label>
          <select name="gender" id="gender" class="pretty" v-model="gender">
            <option value="UNDEFINED">Undefined</option>
            <option value="MALE">Male</option>
            <option value="FEMALE">Female</option>
          </select>
        </div>
        <br>
        <div>
          <pretty-label for="birthday">Birthday:  </pretty-label>
          <input type="date" class="pretty" id="birthday" v-model="birthday">
        </div>
        <br>
```

```
            <div>
              <pretty-label for="password">Password:  </pretty-label>
              <input type="password" class="pretty" id="password" v-model="password"
placeholder="Confirm changes"
                maxlength="64" minlength="3" required>
            </div>
            <br>
            <pretty-button type="submit">Save</pretty-button>
          </form>
          <h1>Update Password</h1>
          <form @submit="onSubmitUpdatePasswordForm">
            <div>
              <pretty-label for="password">Current Password:  </pretty-label>
              <input type="password" class="pretty" id="password" v-model="oldPassword"
placeholder=""
                maxlength="64" minlength="3" required>
            </div>
            <br>
            <div>
              <pretty-label for="password">New Password:  </pretty-label>
              <input type="password" class="pretty" id="newPassword" v-model="newPassword"
placeholder=""
                maxlength="64" minlength="3" required>
            </div>
            <br>
            <pretty-button type="submit">Save</pretty-button>
          </form>
          <h1>Delete Account</h1>
          <form @submit="onSubmitDeleteUserAccountForm">
            <div>
              <pretty-label for="password">Password:  </pretty-label>
              <input type="password" class="pretty" id="password" v-model="password"
placeholder="Confirm changes"
                maxlength="64" minlength="3" required>
            </div>
            <br>
            <pretty-button type="submit">Delete</pretty-button>
          </form>
        </container>
        <container v-else>
          <centered-content>
            <h1>Sorry, you must be authenticate to see this page!</h1>
            <span>
              <router-link to="/login" style="margin-right: 5px;">
                <pretty-button>Login</pretty-button>
              </router-link>
              <router-link to="/register">
                <pretty-button>Register</pretty-button>
              </router-link>
            </span>
          </centered-content>
        </container>
      </main>
    </template>
    <script lang="ts">
    import { mapMutations, mapGetters, mapActions } from 'vuex'
```

```
import UserAccount from '../data/modules/interfaces/UserAccount'
import AccountData from '../data/modules/interfaces/AccountData'
export default {
  data: () => {
    return {
      id: new Number,
      username: new String,
      firstName: new String,
      lastName: new String,
      emailAddress: new String,
      gender: new String,
      birthday: new String,
      password: new String,
      oldPassword: new String,
      newPassword: new String
    }
  },
  computed: {
    ...mapGetters(["getIsAuth", "getUserAccount"])
  },
  methods: {
    ...mapMutations(["updateUserAccount", "logout"]),
    ...mapActions(
      [
        "fetchUpdateUserPersonalInformation",
        "fetchUpdateUserPassword",
        "fetchDeleteUser",
      ]
    ),
    async onSubmitPersonalInformation(event: Event): Promise<void> {
      event.preventDefault()
      const userAccount: UserAccount = {
        _id: this.id,
        username: this.username,
        firstName: this.firstName,
        lastName: this.lastName,
        emailAddress: this.emailAddress,
        gender: this.gender,
        birthday: this.birthday,
        password: this.password
      }
      await this.fetchUpdateUserPersonalInformation(userAccount).then(() => {
        this.updateUserAccount(userAccount)
        this.clearPassword()
      })
    },
    async onSubmitUpdatePasswordForm(event: Event): Promise<void> {
      event.preventDefault()
      const accountData: AccountData = {
        username: this.getUserAccount.username,
        password: this.oldPassword,
        newPassword: this.newPassword
      }
      await this.fetchUpdateUserPassword(accountData).then(() => {
        this.clearPassword()
      })
```

```
    },
    async onSubmitDeleteUserAccountForm(event: Event): Promise<void> {
      event.preventDefault()
      const accountData: AccountData = {
        username: this.getUserAccount.username,
        password: this.password,
      }
      await this.fetchDeleteUser(accountData).then(() => {
        this.clearPassword()
        // this.logout()
      })
    },
    clearPassword(): void {
      this.password = this.oldPassword = this.newPassword = new String
    }
  },
  mounted() {
    const userAccount: UserAccount = this.getUserAccount
    this.id = userAccount._id
    this.username = userAccount.username
    this.firstName = userAccount.firstName
    this.lastName = userAccount.lastName
    this.emailAddress = userAccount.emailAddress
    this.gender = userAccount.gender
    this.birthday = userAccount.birthday
  }
}
</script>
```

Файл “src/frontend/src/views/ProgressiveTenseView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
import PrettyLabel from '../components/PrettyLabel.vue'
</script>
<template>
  <main>
    <h1>Progressive Tenses</h1>
    <br />
    <container>
      <autotranslate-text>Progressive tense is a category of verb tense used to describe ongoing actions.
The progressive tenses are sometimes called the
        'continuing' or 'continuous' tenses. The progressive tenses are recognizable by the
        present participle (i.e., the word that ends '-ing'). Here are some examples of
        verbs in the progressive tenses.
      </autotranslate-text>
      <br>
      <pretty-table>
        <tbody>
          <tr>
            <td>
              <p>Question(?)</p>
            </td>
```

```
    <td>
      <p>Affirmation(+)</p>
    </td>
    <td>
      <p>Negation(-)</p>
    </td>
    <td>
      <p>Times</p>
    </td>
  </tr>
  <tr>
    <td>
      <p>Will I be {{ verb }}ing?</p>
      <p>Will You be {{ verb }}ing?</p>
      <p>Will We be {{ verb }}ing?</p>
      <p>Will They be {{ verb }}ing?</p>
      <p>Will He/She/it be {{ verb }}ing?</p>
    </td>
    <td>
      <p>I will be {{ verb }}ing</p>
      <p>You will be {{ verb }}ing</p>
      <p>We will be {{ verb }}ing</p>
      <p>They will be {{ verb }}ing</p>
      <p>He/She/it will be {{ verb }}ing</p>
    </td>
    <td>
      <p>I will not be {{ verb }}ing</p>
      <p>You will not be {{ verb }}ing</p>
      <p>We will not be {{ verb }}ing</p>
      <p>They will not be {{ verb }}ing</p>
      <p>He/She/it will not be {{ verb }}ing</p>
    </td>
    <td>Future</td>
  </tr>
  <tr>
    <td>
      <p>Am I {{ verb }}ing?</p>
      <p>Are You {{ verb }}ing?</p>
      <p>Are We {{ verb }}ing?</p>
      <p>Are They {{ verb }}ing?</p>
      <p>Is He/She/it {{ verb }}ing?</p>
    </td>
    <td>
      <p>I am {{ verb }}ing</p>
      <p>You are {{ verb }}ing</p>
      <p>We are {{ verb }}ing</p>
      <p>They are {{ verb }}ing</p>
      <p>He/She/it is {{ verb }}ing</p>
    </td>
    <td>
      <p>I am not {{ verb }}ing</p>
      <p>You are not {{ verb }}ing</p>
      <p>We are not {{ verb }}ing</p>
      <p>They are not {{ verb }}ing</p>
      <p>He/She/it is not {{ verb }}ing</p>
    </td>
```

```
                <td>Present</td>
              </tr>
              <tr>
                <td>
                  <p>Was I {{ verb }}ing?</p>
                  <p>Were You {{ verb }}ing?</p>
                  <p>Were We {{ verb }}ing?</p>
                  <p>Were They {{ verb }}ing?</p>
                  <p>Was He/She/it {{ verb }}ing?</p>
                </td>
                <td>
                  <p>I was {{ verb }}ing</p>
                  <p>You were {{ verb }}ing</p>
                  <p>We were {{ verb }}ing</p>
                  <p>They were {{ verb }}ing</p>
                  <p>He/She/it was {{ verb }}ing</p>
                </td>
                <td>
                  <p>I was not {{ verb }}ing</p>
                  <p>You were not {{ verb }}ing</p>
                  <p>We were not {{ verb }}ing</p>
                  <p>They were not {{ verb }}ing</p>
                  <p>He/She/it was not {{ verb }}ing</p>
                </td>
                <td>Past</td>
              </tr>
            </tbody>
          </pretty-table>
          <br>
          <pretty-label for="verbInput">Input Word: </pretty-label>
          <input
            name="verbInput"
            id="verbInput"
            type="text"
            class="pretty"
            placeholder="Слово англ. мовою"
            v-model="verb"
          />
          <br><br>
          <p>Key Words: now</p>
          <p>
            Example: <br />I am working at a cafe. <br />I am working now. <br />I was working
            at 8 pm yesterday. <br />I will be working at 8 pm tomorrow. <br />I am learning
            English now. <br />I was learning English at 2 pm yesterday. <br />I will be learning
            English at 2 pm tomorrow.
          </p>
        </container>
      </main>
    </template>
    <script lang="ts">
    export default {
      data: () => {
        return {
          verb: "work",
        }
      }
```

```
}
</script>
```

Файл “src/frontend/src/views/ReflexivePronounsView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
</script>

<template>
  <main>
    <h1>Reflexive Pronouns</h1>
    <br />
    <container>
      <autotranslate-text>Reflexive pronouns are used when a person or object performs an action
directed at
        itself.</autotranslate-text>
      <autotranslate-text>For example, "He talked about himself", "She does not love
        herself".</autotranslate-text>
      <autotranslate-text>They can also serve for emotional reinforcement, as in the expression "She
saw it
        herself!" (She saw it herself!).</autotranslate-text>
      <autotranslate-text>In such cases, they are called reinforcing pronouns.</autotranslate-text>
      <autotranslate-text>They can also be used to strengthen the independence of an action, as in the
expression
        "I can do it by myself".</autotranslate-text>
      <br>
      <pretty-table cellspacing="0" cellpadding="2">
        <thead>
          <tr>
            <th width="159">
              <p>ОСОБА</p>
            </th>
            <th width="304">
              <p>ЗВОРОТНИЙ ЗАЙМЕННИК</p>
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td width="159">
              <p>I</p>
            </td>
            <td width="304">
              <p>MYSELF</p>
            </td>
          </tr>
          <tr>
            <td width="159">
              <p>YOU</p>
            </td>
            <td width="304">
              <p>YOURSELF/YOURSELVES</p>
```

```
                </td>
              </tr>
              <tr>
                <td width="159">
                  <p>HE</p>
                </td>
                <td width="304">
                  <p>HIMSELF</p>
                </td>
              </tr>
              <tr>
                <td width="159">
                  <p>SHE</p>
                </td>
                <td width="304">
                  <p>HERSELF</p>
                </td>
              </tr>
              <tr>
                <td width="159">
                  <p>IT</p>
                </td>
                <td width="304">
                  <p>ITSELF</p>
                </td>
              </tr>
              <tr>
                <td width="159">
                  <p>WE</p>
                </td>
                <td width="304">
                  <p>OURSELVES</p>
                </td>
              </tr>
              <tr>
                <td width="159">
                  <p>THEY</p>
                </td>
                <td width="304">
                  <p>THEMSELVES</p>
                </td>
              </tr>
            </tbody>
          </pretty-table>
        </container>
    </main>
</template>
<script lang="ts">
export default {}
</script>
```

Файл “src/frontend/src/views/RegisterView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
```

```
import AutotranslateText from '../components/AutotranslateText.vue'
import CenteredContent from '../components/CenteredContent.vue'
import InputLabel from '../components/InputLabel.vue'
import PrettyButton from '../components/PrettyButton.vue'
import { RouterLink } from "vue-router"
</script>


<template>
  <main>
    <container>
      <div v-if="getIsAuth">
      </div>
      <centered-content v-if="getIsAuth">
        <h1>You successful registered and login!</h1>
        <router-link to="/profile">
          <pretty-button>Go to profile</pretty-button>
        </router-link>
      </centered-content>
      <form @submit="onSubmit" v-else>
        <h1>Register</h1>
        <centered-content>
          <input-label for="username">Username</input-label>
          <input class="pretty" type="text" v-model="username" maxlength="64" minlength="3"
required>
          <input-label for="password">Password</input-label>
          <input class="pretty" type="password" v-model="password" maxlength="64" minlength="8"
required>
          <input-label for="password">Repeat Password</input-label>
          <input class="pretty" type="password" v-model="repeatPassword" maxlength="64"
minlength="8"
            required>
          <br><br>
          <pretty-button type="submit">Register!</pretty-button>
        </centered-content>
      </form>
    </container>
  </main>
</template>
<script lang="ts">
import { mapActions, mapGetters } from 'vuex'
export default {
  data: () => {
    return {
      username: new String,
      password: new String,
      repeatPassword: new String,
    }
  },
  computed: {
    ...mapGetters(["getIsAuth"])
  },
  methods: {
    ...mapActions(["fetchRegisterUser", "fetchLoginUser", "passwordMismatchAlert"]),
    async onSubmit(event: Event) {
      event.preventDefault()
```

```
        if (this.password === this.repeatPassword) {
          const data = {
            username: this.username,
            password: this.password,
          }
          await this.fetchRegisterUser(data).then(() => {
            this.clearFields()
          })
        }
        else {
          this.passwordMismatchAlert()
        }
      },
      clearFields() {
        this.username = ""
        this.password = ""
        this.repeatPassword = ""
      }
    }
}
</script>
```

Файл “src/frontend/src/views/SimpleTenseView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyTable from '../components/PrettyTable.vue'
import PrettyLabel from '../components/PrettyLabel.vue'
</script>
<template>
  <main>
    <h1>Simple Tense</h1>
    <br />
    <container>
      <autotranslate-text>The three simple tenses express facts or habitual activities. Unlike the other
        tenses, the simple tenses describe actions without specifically stating whether the
        actions are completed or ongoing.
      </autotranslate-text>
      <autotranslate-text>Be aware that the 'simple present tense' is an
        oddity. Despite its name, it is not always about activities in the present.
      </autotranslate-text>
      <autotranslate-text>The simple present tense can be used for activities that are not currently
happening,
        and it can be used for future events.</autotranslate-text>
      <br>
      <pretty-table>
        <tbody>
          <tr>
            <td colspan="2">
              <p>Question(?)</p>
            </td>
            <td>
              <p>Affirmation(+)</p>
            </td>
```

```
    <td>
      <p>Negation(-)</p>
    </td>
    <td>
      <p>Times</p>
    </td>
  </tr>
  <tr>
    <td>
      <p>Will</p>
    </td>
    <td>
      <p>I {{ verb }}?</p>
      <p>You {{ verb }}?</p>
      <p>We {{ verb }}?</p>
      <p>They {{ verb }}?</p>
      <p>He/She/it {{ verb }}?</p>
    </td>
    <td>
      <p>I will {{ verb }}</p>
      <p>You will {{ verb }}</p>
      <p>We will {{ verb }}</p>
      <p>They will {{ verb }}</p>
      <p>He/She/it will {{ verb }}</p>
    </td>
    <td>
      <p>I will not {{ verb }}</p>
      <p>You will not {{ verb }}</p>
      <p>We will not {{ verb }}</p>
      <p>They will not {{ verb }}</p>
      <p>He/She/it will not {{ verb }}</p>
    </td>
    <td>Future</td>
  </tr>
  <tr>
    <td>
      <p>Do</p>
    </td>
    <td>
      <p>I {{ verb }}?</p>
      <p>You {{ verb }}?</p>
      <p>We {{ verb }}?</p>
      <p>They {{ verb }}?</p>
    </td>
    <td>
      <p>I {{ verb }}</p>
      <p>You {{ verb }}</p>
      <p>We {{ verb }}</p>
      <p>They {{ verb }}</p>
    </td>
    <td>
      <p>I do not {{ verb }}</p>
      <p>You do not {{ verb }}</p>
      <p>We do not {{ verb }}</p>
      <p>They do not {{ verb }}</p>
    </td>
```

```
                <td rowspan="2">Present</td>
              </tr>
              <tr>
                <td>
                  <p>Does</p>
                </td>
                <td>
                  <p>He/She/it {{ verb }}?</p>
                </td>
                <td>
                  <p>He/She/it {{ verb }}s</p>
                </td>
                <td>
                  <p>He/She/it does not {{ verb }}</p>
                </td>
              </tr>
              <tr>
                <td>
                  <p>Did</p>
                </td>
                <td>
                  <p>I {{ verb }}?</p>
                  <p>You {{ verb }}?</p>
                  <p>We {{ verb }}?</p>
                  <p>They {{ verb }}?</p>
                  <p>He/She/it {{ verb }}?</p>
                </td>
                <td>
                  <p>I {{ verbV2 }}</p>
                  <p>You {{ verbV2 }}</p>
                  <p>We {{ verbV2 }}</p>
                  <p>They {{ verbV2 }}</p>
                  <p>He/She/it {{ verbV2 }}</p>
                </td>
                <td>
                  <p>I did not {{ verb }}</p>
                  <p>You did not {{ verb }}</p>
                  <p>We did not {{ verb }}</p>
                  <p>They did not {{ verb }}</p>
                  <p>He/She/it did not {{ verb }}</p>
                </td>
                <td>Past</td>
              </tr>
            </tbody>
          </pretty-table>
          <br />
          <pretty-label for="verbInput">Input Verb: </pretty-label>
          <input class="pretty" name="verbInput" id="verbInput" type="text" placeholder="Дієслово англ.
мовою" v-model="verb" />
          <br><br>
          <p>Key Words: Always, Usually, Sometimes, Rarely, Never</p>
          <p>
            Example: <br />I work at a cafe. <br />I work every day. <br />I worked yesterday.
            <br />I will work tomorrow. <br />I learn English every day. <br />I (learnt)
            English yesterday. <br />I will English learn tomorrow.
          </p>
```

```
      </container>
   </main>
</template>

<script lang="ts">
import { mapActions } from 'vuex'

export default {
   data: () => {
      return {
         verb: "work",
         verbV2: "worked",
      }
   },
   methods: {
      ...mapActions(["getVerbInV2"]),
   },
   watch: {
      async verb(newValue) {
         await this.getVerbInV2(newValue).then((verbV2: string) => {
            this.verbV2 = verbV2
         })
      }
   },
}
</script>
```

Файл “src/frontend/src/views/SocialiteTalkView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
   <main>
      <container>
         <h1>1. Common Topics for Socialite Talk</h1>
         <autotranslate-text>When engaging in small talk, it's helpful to have a few go-to topics. Here are
some
            commonly used ones:</autotranslate-text>
         <br>
         <autotranslate-text>Weather:</autotranslate-text>
         <autotranslate-text>"It’s a beautiful day, isn’t it?"</autotranslate-text>
         <autotranslate-text>"Can you believe how cold it is today?"</autotranslate-text>
         <br>
         <autotranslate-text>Current Events:</autotranslate-text>
         <autotranslate-text>"Did you hear about the new restaurant opening downtown?"</autotranslate-
text>
         <autotranslate-text>"What do you think about the latest news on [topic]?"</autotranslate-text>
         <br>
         <autotranslate-text>Hobbies and Interests:</autotranslate-text>
         <autotranslate-text>"Do you have any plans for the weekend?"</autotranslate-text>
         <autotranslate-text>"What do you like to do in your free time?"</autotranslate-text>
         <br>
         <autotranslate-text>Work and Career:</autotranslate-text>
```

```
<autotranslate-text>"How’s work going for you?"</autotranslate-text>
<autotranslate-text>"What kind of projects are you working on these days?"</autotranslate-text>
<br>
<autotranslate-text>Travel:</autotranslate-text>
<autotranslate-text>"Have you traveled anywhere interesting recently?"</autotranslate-text>
<autotranslate-text>"What’s your favorite place to visit?"</autotranslate-text>
<br>
<autotranslate-text>Entertainment:</autotranslate-text>
<autotranslate-text>"Have you seen any good movies lately?"</autotranslate-text>
<autotranslate-text>"What shows are you currently watching?"</autotranslate-text>
<h2>2. Starting a Conversation</h2>
<autotranslate-text>Starting a conversation can sometimes be the hardest part. Here are some phrases to help
   you get started:</autotranslate-text>
<br>
<autotranslate-text>"Hi, I don’t think we’ve met before. I’m [Your Name]."</autotranslate-text>
<autotranslate-text>"Hello! How are you doing today?"</autotranslate-text>
<autotranslate-text>"Hi, do you mind if I join you?"</autotranslate-text>
<autotranslate-text>"Excuse me, is this seat taken?"</autotranslate-text>
<h2>3. Keeping the Conversation Going</h2>
<autotranslate-text>Once you’ve started the conversation, you’ll want to keep it going. Here are some tips
   and phrases to help
   you maintain the flow:
</autotranslate-text>
<br>
<autotranslate-text>Show Interest:</autotranslate-text>
<br>
<autotranslate-text>"That sounds fascinating! Tell me more about it."</autotranslate-text>
<autotranslate-text>
   "How did you get into that hobby?"
</autotranslate-text>
<br>
<autotranslate-text>Ask Open-Ended Questions:</autotranslate-text>
<br>
<autotranslate-text>"What do you enjoy most about your job?"</autotranslate-text>
<autotranslate-text>"What was the highlight of your weekend?"</autotranslate-text>
<br>
<autotranslate-text>Share Something About Yourself:</autotranslate-text>
<br>
<autotranslate-text>"I love hiking too! Last weekend, I went to [place]."</autotranslate-text>
<autotranslate-text>"I’ve been meaning to try that restaurant. How was it?"</autotranslate-text>
<br>
<autotranslate-text>Give Compliments:</autotranslate-text>
<br>
<autotranslate-text>"I love your dress. Where did you get it?"</autotranslate-text>
<autotranslate-text>"You have a great sense of humor."
</autotranslate-text>
<h2>4. Ending the Conversation</h2>
<autotranslate-text>Knowing how to gracefully exit a conversation is just as important as starting one. Here
   are some polite
   ways to wrap things up:
</autotranslate-text>
<br>
<autotranslate-text>"It was great talking with you. I should get going now."</autotranslate-text>
```

<autotranslate-text>"I’ve really enjoyed our conversation. Let’s chat again soon."</autotranslate-text>
<autotranslate-text>"I need to say hello to someone over there, but it was lovely meeting you."</autotranslate-text>
<autotranslate-text>"Thank you for the chat. Have a wonderful day!"</autotranslate-text>
<h2>5. Tips for Effective Socialite Talk</h2>
<autotranslate-text>Be a Good Listener: Show genuine interest in what the other person is saying.</autotranslate-text>
<autotranslate-text>Be Mindful of Body Language: Maintain eye contact, nod, and smile to show engagement.</autotranslate-text>
<autotranslate-text>Avoid Controversial Topics: Steer clear of politics, religion, and other potentially divisive subjects.</autotranslate-text>
<autotranslate-text>Practice Active Listening: Reflect back what the other person is saying to show you are paying attention.</autotranslate-text>
<autotranslate-text>Stay Positive: Keep the conversation light and upbeat.</autotranslate-text>
<autotranslate-text>Be Respectful of Boundaries: If the other person seems uncomfortable or uninterested, gracefully change the subject or end the conversation.</autotranslate-text>

<autotranslate-text>Sample Socialite Talk Conversations</autotranslate-text>

<autotranslate-text>Example 1: At a Party</autotranslate-text>

<autotranslate-text>You: "Hi, I’m [Your Name]. I don’t think we’ve met before."</autotranslate-text>
<autotranslate-text>Other Person: "Hi, I’m [Their Name]. Nice to meet you."</autotranslate-text>
<autotranslate-text>You: "Nice to meet you too! How do you know the host?"</autotranslate-text>
<autotranslate-text>Other Person: "We work together. How about you?"</autotranslate-text>
<autotranslate-text>You: "We went to college together. Have you been working there long?"</autotranslate-text>

<autotranslate-text>Example 2: In the Office</autotranslate-text>

<autotranslate-text>You: "Good morning! How was your weekend?"</autotranslate-text>
<autotranslate-text>Other Person: "It was great, thanks! I went hiking. How about yours?"</autotranslate-text>
<autotranslate-text>You: "That sounds fun! Mine was good too. I went to a new restaurant in town. Have you been to [Restaurant Name]?"</autotranslate-text>

<autotranslate-text>Example 3: At a Networking Event</autotranslate-text>

<autotranslate-text>You: "Hello, I’m [Your Name]. I work in [Your Field]."</autotranslate-text>
<autotranslate-text>Other Person: "Hi, I’m [Their Name]. I’m in [Their Field]."</autotranslate-text>
<autotranslate-text>You: "Nice to meet you. What do you enjoy most about your job?"</autotranslate-text>


```
        <autotranslate-text>By following this guide, you should feel more confident in engaging in
socialite talk in
            various situations.</autotranslate-text>
        <autotranslate-text>Remember to be yourself and enjoy the conversation!</autotranslate-text>
      </container>
      <br><br>
   </main>
</template>
<script lang="ts">
export default {
}
</script>
```

Файл “src/frontend/src/views/TheConditionalMoodView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
   <main>
      <h1>The Conditional Mood</h1>
      <br />
      <container>
        <autotranslate-text>The conditional mood in English is used to talk about events or situations that
are not guaranteed to
            happen. It often involves hypothetical or imaginary scenarios.
        </autotranslate-text>
        <br>
        <autotranslate-text>There are four main types of conditional sentences:</autotranslate-text>
        <autotranslate-text>Zero Conditional: Used to talk about general truths or facts.
            If you heat ice, it melts.
            Present simple in both clauses.
        </autotranslate-text>
        <br>
        <autotranslate-text>First Conditional: Used to talk about real and possible future
            situations.</autotranslate-text>
        <autotranslate-text>If it rains, we will stay home.</autotranslate-text>
        <autotranslate-text>Present simple in the if-clause, and will + base form of the verb in the main
            clause.</autotranslate-text>
        <br>
        <autotranslate-text>Second Conditional: Used to talk about imaginary or unlikely situations in the
present
            or future.</autotranslate-text>
        <autotranslate-text>If I won the lottery, I would buy a big house.</autotranslate-text>
        <autotranslate-text>Past simple in the if-clause, and would + base form of the verb in the main
            clause.</autotranslate-text>
        <br>
        <autotranslate-text>Third Conditional: Used to talk about unreal situations in the
            past.</autotranslate-text>
        <autotranslate-text>If she had studied harder, she would have passed the exam.</autotranslate-
text>
        <autotranslate-text>Past perfect in the if-clause, and would have + past participle in the main
            clause.</autotranslate-text>
```

```
        <br>
        <autotranslate-text>Conditional sentences can be mixed and matched to express different degrees
of
            possibility and reality.</autotranslate-text>
        <autotranslate-text>They're useful for discussing hypothetical outcomes and their potential
            consequences.</autotranslate-text>
      </container>
   </main>
</template>
<script lang="ts">
export default {
}
</script>
```

Файл “src/frontend/src/views/TheGerundView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
   <main>
      <h1>The Gerund</h1>
      <br />
      <container>
         <autotranslate-text>The gerund in English is a form of a verb that ends in "-ing" and functions as
a noun.
            It's created by
            adding "-ing" to the base form of the verb.
         </autotranslate-text>
         <br>
         <autotranslate-text>Examples:</autotranslate-text>
         <autotranslate-text>Reading is fun. (Here, "reading" is the subject of the sentence.)
            I enjoy swimming. (Here, "swimming" is the object of the verb "enjoy".)
         </autotranslate-text>
         <br>
         <autotranslate-text>Gerunds can be used as subjects, objects, after prepositions, and in other
situations
            where a noun would
            typically be used. They can also follow certain verbs that are specifically followed by gerunds,
such as
            "enjoy," "admit," "avoid," etc.
         </autotranslate-text>
         <br>
         <autotranslate-text>Examples:</autotranslate-text>
         <autotranslate-text>He admitted eating all the cookies. (Gerund used as the object of the verb
"admitted.")
            She is good at cooking. (Gerund used after the preposition "at.")
         </autotranslate-text>
         <br>
         <autotranslate-text>Gerunds are versatile and commonly used in English for various purposes,
including
            expressing actions,
            activities, and states.
         </autotranslate-text>
```

```
      </container>
   </main>
</template>
<script lang="ts">
export default {
}
</script>
```

Файл “src/frontend/src/views/ToBeAbleView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
import PrettyLabel from '../components/PrettyLabel.vue';
</script>


<template>
   <main>
      <h1>About The Verb TO BE ABLE</h1>
      <br />
      <container>
         <p>
            В англійській мові дієслово "уміти" не має прямого перекладу і передається
            словосполученням &ldquo;to know how to&rdquo; або &ldquo;to be able to&rdquo;.
         </p>
         <br>
         <p>
            He <strong>knows </strong><strong>how </strong><strong>to </strong> swim. He
            <strong>is </strong><strong>able </strong><strong>to </strong> swim. <em>(Він
</em><strong>уміє
            </strong><em> плавати.)</em>
         </p>
         <br>
         <p>
            My son <strong>does </strong><strong>not </strong><strong>know </strong><strong>how
</strong><strong>to
            </strong> count. <br>My son <strong>is </strong><strong>not </strong><strong>able
</strong><strong>to
            </strong> count. (Мій син
            <strong>не </strong><strong>вміє </strong> рахувати.)
         </p>
         <br><br>
         <pretty-label class="SpanLabel">Person: </pretty-label>
         <select class="pretty" id="person" v-model="person" required>
            <option value="I">I</option>
            <option value="He">He</option>
            <option value="She">She</option>
            <option value="It">It</option>
            <option value="We">We</option>
            <option value="They">They</option>
         </select>
          
         <pretty-label class="SpanLabel">Type: </pretty-label>
```

```
        <select class="pretty" id="type" v-model="type" required>
          <option value="Affirmation">Affirmation</option>
          <option value="Negation">Negation</option>
          <option value="Question" disabled>Question</option>
        </select>
        <br /><br>
        <pretty-label for="verbInput">Input verb: </pretty-label>
        <input class="pretty" name="verbInput" id="verbInput" type="text" placeholder="Слово англ.
мовою" v-model="verb" /><br><br>
        <p>{{ sentenceAble }}</p>
        <p>{{ sentenceKnow }}</p>
        <br>
      </container>
    </main>
</template>
<script lang="ts">
export default {
    data: () => {
      return {
        verb: "manage",
        person: "I",
        type: "Affirmation",
        sentenceAble: "I am able to manage",
        sentenceKnow: "I know how to manage ",
      }
    },
    watch: {
      person() {
        this.updateSentences()
      },
      verb() {
        this.updateSentences()
      },
      type() {
        this.updateSentences()
      },
    },
    methods: {
      clearVerb(verb: string): string {
        const clearedVerb = verb
          .replace("a ", "")
          .replace("an ", "")
          .replace("the ", "")
          .replace("to ", "")
        return clearedVerb
      },
      updateSentences() {
        if (this.person && this.verb && this.type) {
          this.sentenceAble = this.buildAbleSentence(this.person, this.verb, this.type)
          this.sentenceKnow = this.buildKnowSentence(this.person, this.verb, this.type)
        }
      },
      buildAbleSentence(person: string, verb: string, type: string): string {
        verb = this.clearVerb(verb)
        let sentence = person + " "
        if (person === "I")
```

```
            sentence = sentence + "am "
        else if (person === "He" || person === "She" || person === "It")
            sentence = sentence + "is "
        else if (person === "We" || person === "They" || person === "You")
            sentence = sentence + "are "
        if (type === "Negation")
            sentence = sentence + "not "
        sentence = sentence + "able to " + verb
        return sentence
    },
    buildKnowSentence(person: string, verb: string, type: string): string {
        verb = this.clearVerb(verb)
        let sentence = person + " "
        if (type === "Negation")
            sentence = sentence + "does not "
        if (person === "I")
            sentence = sentence + "know "
        else if (person === "He" || person === "She" || person === "It")
            sentence = sentence + "knows "
        else if (person === "We" || person === "They" || person === "You")
            sentence = sentence + "know "
        sentence = sentence + "how to " + verb
        return sentence
    },
  }
}
</script>
```

Файл “src/frontend/src/views/UsedToView.vue”

```
<script lang="ts" setup>
import Container from '../components/Container.vue'
import AutotranslateText from '../components/AutotranslateText.vue'
</script>
<template>
  <main>
    <h1>Used to</h1>
    <br />
    <container>
        <autotranslate-text>"Used to" is a phrase that is used for expressing the idea that someone
regularly did
            something in the past but they no longer do it now.
        </autotranslate-text>
        <autotranslate-text>For example, "I used to play
            tennis every weekend, but I don't anymore." It can also be used for expressing a
            past habit or behavior that has changed over time, such as "She used to be shy, but
            she's much more outgoing now."
        </autotranslate-text>
    </container>
  </main>
</template>
<script lang="ts">
export default {
}
```

```
</script>
```

Файл “src/frontend/src/views/tasks/ConditionalSentencesExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
   counter-reset: li;
}
.exercise .tasks li {
   list-style: none;
   position: relative;
   padding: .75em .5em;
   margin: 0 0 0 2em;
}
.exercise .tasks li::before {
   background-color: lightblue;
   color: var(--primary-color);
   border-radius: 50%;
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   -ms-box-sizing: border-box;
   box-sizing: border-box;
   content: counter(li);
   counter-increment: li;
   font-size: .875em;
   font-weight: 700;
   height: 1.785em;
   left: -2em;
   margin-top: -0.9em;
   padding-top: 0;
   position: absolute;
   text-align: center;
   top: 50%;
   width: 1.785em;
   line-height: 1.8em;
}
.exercise-textfield {
   width: 200px;
}
.correct {
   border-color: green;
   color: green;
}
.incorrect {
   border-color: red;
   color: red;
}
button {
   margin-right: 15px;
}
```

```
</style>
<template>
  <main>
    <h1>Conditional Sentences</h1>
    <container>
      <autotranslate-text>Put in the correct phrases and form a conditional sentence (type I, II, III).
Watch the underlined verbs.
      </autotranslate-text>
      <br>
      <form class="exercise" @submit="onSubmitExercise">
        <ol class="tasks">
          <li>
            If we meet at 11:40, we <input type="text"
              class="exercise-textfield"> (to have) plenty of time.
          </li>
          <li>
            Alice would find the milk if she <input type="text"
              class="exercise-textfield"> (to look) in the fridge.
          </li>
          <li>
            The zookeeper would have punished her with a fine if she <input type="text"
              class="exercise-textfield"> (to feed) the animals.
          </li>
          <li>
            If you spoke louder, your classmates <input type="text"
              class="exercise-textfield"> (to understand) you.
          </li>
          <li>
            Denis <input type="text"
              class="exercise-textfield"> (to arrive) safe if he drove slowly.
          </li>
          <li>
            You <input type="text"
              class="exercise-textfield"> (to have) no trouble at school if you had done your
homework.
          </li>
          <li>
            If you <input type="text"
              class="exercise-textfield"> (to swim) in this lake, you'll shiver from cold.
          </li>
          <li>
            The door will unlock if you <input type="text"
              class="exercise-textfield"> (to press) the green button.
          </li>
          <li>
            If Rodger <input type="text"
              class="exercise-textfield"> (to ask) her teacher, he'd have answered her questions.
          </li>
          <li>
            I <input type="text"
              class="exercise-textfield"> (to call) the office if I was/were you.
          </li>
        </ol>
       <pretty-button type="submit">Check</pretty-button>
        <pretty-button @click="showAnswer">Show Answer</pretty-button>
      </form>
```

```
      </container>
   </main>
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
   data: () => {
      return {
         answers: ["will have", "looked","had fed","would understand","would arrive","would have
had","swim","press","had asked","would call"]
      }
   },
   computed: mapGetters(["getIsAuth"]),
   methods: {
      ...mapMutations(["updateFlashMessage"]),
      onSubmitExercise(event: Event) {
         event.preventDefault()
         const form: HTMLFormElement = document.querySelector('.exercise')
         const formElements: HTMLFormControlsCollection | any = form.elements
         for (let i = 0; i < formElements.length - 2; i++) {
            if (formElements[i].value === this.answers[i]) {
               formElements[i].classList.add("correct")
               formElements[i].classList.remove("incorrect")
            } else {
               formElements[i].classList.add("incorrect")
               formElements[i].classList.remove("correct")
            }
         }
      },
      isUserAccountAuth(): boolean {
         if (this.getIsAuth) {
            return true
         } else {
            this.updateFlashMessage({
               message: "Please login to view answers!",
               className: "red",
            })
            return false
         }
      },
      showAnswer() {
         if (this.isUserAccountAuth()) {
            const form: HTMLFormElement = document.querySelector('.exercise')
            const formElements: HTMLFormControlsCollection | any = form.elements
            for (let i = 0; i < formElements.length - 2; i++) {
               formElements[i].value = this.answers[i]
            }
         }
      }
   }
}
</script>
```

Файл “src/frontend/src/views/tasks/FuturePerfectExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
   counter-reset: li;
}
.exercise .tasks li {
   list-style: none;
   position: relative;
   padding: .75em .5em;
   margin: 0 0 0 2em;
}
.exercise .tasks li::before {
   background-color: lightblue;
   color: var(--primary-color);
   border-radius: 50%;
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   -ms-box-sizing: border-box;
   box-sizing: border-box;
   content: counter(li);
   counter-increment: li;
   font-size: .875em;
   font-weight: 700;
   height: 1.785em;
   left: -2em;
   margin-top: -0.9em;
   padding-top: 0;
   position: absolute;
   text-align: center;
   top: 50%;
   width: 1.785em;
   line-height: 1.8em;
}
.exercise-textfield {
   width: 200px;
}
.correct {
   border-color: green;
   color: green;
}
.incorrect {
   border-color: red;
   color: red;
}
button {
   margin-right: 15px;
}
</style>
```

```
<template>
  <main>
    <h1>Future Perfect</h1>
    <container>
      <autotranslate-text>Put in the verbs in brackets in Future Perfect.
      </autotranslate-text>
      <br>
      <form class="exercise" @submit="onSubmitExercise">
        <ol class="tasks">
          <li>
            Helena <input type="text" class="exercise-textfield"> her bike next week. (to repair)
          </li>
          <li>
            We <input type="text" class="exercise-textfield"> the washing by 8 o'clock. (to do)
          </li>
          <li>
            She <input type="text" class="exercise-textfield"> Paris by the end of next year. (to visit)
          </li>
          <li>
            I <input type="text" class="exercise-textfield"> this by 7 o'clock. (to finish)
          </li>
          <li>
            Henry <input type="text" class="exercise-textfield"> by next week. (to leave)
          </li>
          <li>
            She <input type="text" class="exercise-textfield"> this with her mother tonight. (to
discuss)
          </li>
          <li>
            The police <input type="text" class="exercise-textfield"> the driver. (to arrest)
          </li>
          <li>
            They <input type="text" class="exercise-textfield"> their essay by tomorrow. (to write)
          </li>
          <li>
            John <input type="text" class="exercise-textfield"> the teams. (to manage)
          </li>
          <li>
            If we can do that - then we <input type="text" class="exercise-textfield"> our mission. (to
fulfil)
          </li>
        </ol>
        <pretty-button type="submit">Check</pretty-button>
        <pretty-button @click="showAnswer">Show Answer</pretty-button>
      </form>
    </container>
  </main>
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'


export default {
  data: () => {
    return {
```

```
        answers: ["will have repaired", "shall have done", "will have visited", "shall have finished", "will
have left", "will have discussed", "will have arrested", "will have written", "will have managed", "shall
have fulfilled"]
      }
  },
  computed: mapGetters(["getIsAuth"]),
  methods: {
    ...mapMutations(["updateFlashMessage"]),
    onSubmitExercise(event: Event) {
      event.preventDefault()
      const form: HTMLFormElement = document.querySelector('.exercise')
      const formElements: HTMLFormControlsCollection | any = form.elements
      for (let i = 0; i < formElements.length - 2; i++) {
        if (formElements[i].value === this.answers[i]) {
          formElements[i].classList.add("correct")
          formElements[i].classList.remove("incorrect")
        } else {
          formElements[i].classList.add("incorrect")
          formElements[i].classList.remove("correct")
        }
      }
    },
    isUserAccountAuth(): boolean {
      if (this.getIsAuth) {
        return true
      } else {
        this.updateFlashMessage({
          message: "Please login to view answers!",
          className: "red",
        })
        return false
      }
    },
    showAnswer() {
      if (this.isUserAccountAuth()) {
        const form: HTMLFormElement = document.querySelector('.exercise')
        const formElements: HTMLFormControlsCollection | any = form.elements
        for (let i = 0; i < formElements.length - 2; i++) {
          formElements[i].value = this.answers[i]
        }
      }
    }
  }
}
</script>
```

Файл “src/frontend/src/views/tasks/GerundAfterPrepositionsExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
```

```
    counter-reset: li;
}
.exercise .tasks li {
    list-style: none;
    position: relative;
    padding: .75em .5em;
    margin: 0 0 0 2em;
}
.exercise .tasks li::before {
    background-color: lightblue;
    color: var(--primary-color);
    border-radius: 50%;
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    -ms-box-sizing: border-box;
    box-sizing: border-box;
    content: counter(li);
    counter-increment: li;
    font-size: .875em;
    font-weight: 700;
    height: 1.785em;
    left: -2em;
    margin-top: -0.9em;
    padding-top: 0;
    position: absolute;
    text-align: center;
    top: 50%;
    width: 1.785em;
    line-height: 1.8em;
}
.exercise-textfield {
    width: 100px;
}
.correct {
    border-color: green;
    color: green;
}
.incorrect {
    border-color: red;
    color: red;
}
button {
    margin-right: 15px;
}
</style>
<template>
    <main>
        <h1>Gerund after prepositions</h1>
        <container>
            <autotranslate-text>Put in the correct preposition.
            </autotranslate-text>
            <br>
            <form class="exercise" @submit="onSubmitExercise">
                <ol class="tasks">
                    <li>
                        My brother is good <input type="text"
```

```
                    class="exercise-textfield"> playing football.
                </li>
                <li>
                    He complains <input type="text"
                    class="exercise-textfield"> bullying.
                </li>
                <li>
                    We are afraid <input type="text"
                    class="exercise-textfield"> losing the match.
                </li>
                <li>
                    He doesn't feel <input type="text"
                    class="exercise-textfield"> working on the computer.
                </li>
                <li>
                    They are looking forward <input type="text"
                    class="exercise-textfield"> going out at the holiday.
                </li>
                <li>
                    Charlie dreams <input type="text"
                    class="exercise-textfield"> living on a small island.
                </li>
                <li>
                    Denis apologized <input type="text"
                    class="exercise-textfield"> being late.
                </li>
                <li>
                    I don't agree <input type="text"
                    class="exercise-textfield"> hunting.
                </li>
                <li>
                    The girls insisted <input type="text"
                    class="exercise-textfield"> going out with Charlie.
                </li>
                <li>
                    John often thinks <input type="text"
                    class="exercise-textfield"> climbing trees.
                </li>
            </ol>
            <pretty-button type="submit">Check</pretty-button>
            <pretty-button @click="showAnswer">Show Answer</pretty-button>
        </form>
    </container>
  </main>
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
  data: () => {
    return {
      answers: ["at","about","of","like","to","of","for","with","on","of"]
    }
  },
  computed: mapGetters(["getIsAuth"]),
  methods: {
    ...mapMutations(["updateFlashMessage"]),
```

```
        onSubmitExercise(event: Event) {
            event.preventDefault()
            const form: HTMLFormElement = document.querySelector('.exercise')
            const formElements: HTMLFormControlsCollection | any = form.elements
            for (let i = 0; i < formElements.length - 2; i++) {
                if (formElements[i].value === this.answers[i]) {
                    formElements[i].classList.add("correct")
                    formElements[i].classList.remove("incorrect")
                } else {
                    formElements[i].classList.add("incorrect")
                    formElements[i].classList.remove("correct")
                }
            }
        },
        isUserAccountAuth(): boolean {
            if (this.getIsAuth) {
                return true
            } else {
                this.updateFlashMessage({
                    message: "Please login to view answers!",
                    className: "red",
                })
                return false
            }
        },
        showAnswer() {
            if (this.isUserAccountAuth()) {
                const form: HTMLFormElement = document.querySelector('.exercise')
                const formElements: HTMLFormControlsCollection | any = form.elements
                for (let i = 0; i < formElements.length - 2; i++) {
                    formElements[i].value = this.answers[i]
                }
            }
        }
    }
}
</script>
```

Файл “src/frontend/src/views/tasks/GerundAndInfinitiveExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
    counter-reset: li;
}
.exercise .tasks li {
    list-style: none;
    position: relative;
    padding: .75em .5em;
    margin: 0 0 0 2em;
}
```

```
.exercise .tasks li::before {
   background-color: lightblue;
   color: var(--primary-color);
   border-radius: 50%;
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   -ms-box-sizing: border-box;
   box-sizing: border-box;
   content: counter(li);
   counter-increment: li;
   font-size: .875em;
   font-weight: 700;
   height: 1.785em;
   left: -2em;
   margin-top: -0.9em;
   padding-top: 0;
   position: absolute;
   text-align: center;
   top: 50%;
   width: 1.785em;
   line-height: 1.8em;
}


.exercise-textfield {
   width: 100px;
}
.correct {
   border-color: green;
   color: green;
}
.incorrect {
   border-color: red;
   color: red;
}
button {
   margin-right: 15px;
}
</style>
<template>
   <main>
      <h1>Gerund and Infinitive</h1>
      <container>
         <autotranslate-text>Put in the verbs in brackets in the Gerund or the to-infinitive.
         </autotranslate-text>
         <br>
         <form class="exercise" @submit="onSubmitExercise">
            <ol class="tasks">
               <li>
                  I can't imagine Bill
                  <input type="text" class="exercise-textfield">
                  (go) by bike.
               </li>
               <li>
                  He agreed
                  <input type="text" class="exercise-textfield">
```

```
                (buy) a new car.
              </li>
              <li>
                The question is easy<input type="text" class="exercise-textfield"> (answer).
              </li>
              <li>
                The man asked me how
                <input type="text" class="exercise-textfield">
                (get) to the airport.
              </li>
              <li>
                I look forward to
                <input type="text" class="exercise-textfield">
                (see) you at the weekend.
              </li>
              <li>
                We decided <input type="text" class="exercise-textfield"> (run) through the forest.
              </li>
              <li>
                The teacher expected Sarah <input type="text" class="exercise-textfield"> (study) hard.
              </li>
              <li>
                She doesn't mind <input type="text" class="exercise-textfield"> (work) the night shift.
              </li>
              <li>
                The girls insisted <input type="text" class="exercise-textfield"> going out with Charlie.
              </li>
              <li>
                I learned <input type="text" class="exercise-textfield"> (ride) the bike at the age of 5.
              </li>
            </ol>
            <pretty-button type="submit">Check</pretty-button>
            <pretty-button @click="showAnswer">Show Answer</pretty-button>
          </form>
        </container>
    </main>
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
    data: () => {
        return {
            answers: ["going", "to buy", "to answer", "to get", "seeing", "visiting", "to run", "to study",
"working", "to ride"]
        }
    },
    computed: mapGetters(["getIsAuth"]),
    methods: {
        ...mapMutations(["updateFlashMessage"]),
        onSubmitExercise(event: Event) {
            event.preventDefault()
            const form: HTMLFormElement = document.querySelector('.exercise')
            const formElements: HTMLFormControlsCollection | any = form.elements
            for (let i = 0; i < formElements.length - 2; i++) {
                if (formElements[i].value === this.answers[i]) {
                    formElements[i].classList.add("correct")
```

```
                formElements[i].classList.remove("incorrect")
            } else {
                formElements[i].classList.add("incorrect")
                formElements[i].classList.remove("correct")
            }
        }
    },
    isUserAccountAuth(): boolean {
        if (this.getIsAuth) {
            return true
        } else {
            this.updateFlashMessage({
                message: "Please login to view answers!",
                className: "red",
            })
            return false
        }
    },
    showAnswer() {
        if (this.isUserAccountAuth()) {
            const form: HTMLFormElement = document.querySelector('.exercise')
            const formElements: HTMLFormControlsCollection | any = form.elements
            for (let i = 0; i < formElements.length - 2; i++) {
                formElements[i].value = this.answers[i]
            }
        }
    }
  }
}
</script>
```

Файл “src/frontend/src/views/tasks/NounsArticlesExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
  counter-reset: li;
}
.exercise .tasks li {
  list-style: none;
  position: relative;
  padding: .75em .5em;
  margin: 0 0 0 2em;
}
.exercise .tasks li::before {
  background-color: lightblue;
  color: var(--primary-color);
  border-radius: 50%;
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  -ms-box-sizing: border-box;
```

```
    box-sizing: border-box;
    content: counter(li);
    counter-increment: li;
    font-size: .875em;
    font-weight: 700;
    height: 1.785em;
    left: -2em;
    margin-top: -0.9em;
    padding-top: 0;
    position: absolute;
    text-align: center;
    top: 50%;
    width: 1.785em;
    line-height: 1.8em;
}
.exercise-textfield {
    width: 100px;
}
.correct {
    border-color: green;
    color: green;
}
.incorrect {
    border-color: red;
    color: red;
}
button {
    margin-right: 15px;
}
</style>
<template>
    <main>
        <h1>Nouns Articles</h1>
        <container>
            <autotranslate-text>Put in the article a, an or the where necessary.
            </autotranslate-text>
            <br>
            <form class="exercise" @submit="onSubmitExercise">
                <ol class="tasks">
                    <li>
                        I like <input type="text" class="exercise-textfield"> purple T-shirt over there better <input
                            type="text" class="exercise-textfield"> than blue one.
                    </li>
                    <li>
                        Their car does 160 miles <input type="text" class="exercise-textfield"> hour.
                    </li>
                    <li>
                        Where's <input type="text" class="exercise-textfield"> USB drive I lent you last week?
                    </li>
                    <li>
                        Do you still live in <input type="text" class="exercise-textfield"> Bristol?
                    </li>
                    <li>
                        Is your father working in <input type="text" class="exercise-textfield"> old office
building?
                    </li>
```

```
                <li>
                    Bill's mother works as <input type="text" class="exercise-textfield"> electrician.
                </li>
                <li>
                    The potatoes are 98 pence <input type="text" class="exercise-textfield"> kilo.
                </li>
                <li>
                    What do you usually have for <input type="text" class="exercise-textfield"> breakfast?
                </li>
                <li>
                    Henry has <input type="text" class="exercise-textfield"> terrible headache.
                </li>
                <li>
                    After this tour you have <input type="text" class="exercise-textfield"> whole afternoon
free to
                    explore the city.
                </li>
            </ol>
            <pretty-button type="submit">Check</pretty-button>
            <pretty-button @click="showAnswer">Show Answer</pretty-button>
        </form>
    </container>
  </main>
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
  data: () => {
    return {
      answers: ["the", "the", "an", "the", "", "an", "an", "a", "", "a", "the"]
    }
  },
  computed: mapGetters(["getIsAuth"]),
  methods: {
    ...mapMutations(["updateFlashMessage"]),
    onSubmitExercise(event: Event) {
      event.preventDefault()
      const form: HTMLFormElement = document.querySelector('.exercise')
      const formElements: HTMLFormControlsCollection | any = form.elements
      for (let i = 0; i < formElements.length - 2; i++) {
        if (formElements[i].value === this.answers[i]) {
          formElements[i].classList.add("correct")
          formElements[i].classList.remove("incorrect")
        } else {
          formElements[i].classList.add("incorrect")
          formElements[i].classList.remove("correct")
        }
      }
    },
    isUserAccountAuth(): boolean {
      if (this.getIsAuth) {
        return true
      } else {
        this.updateFlashMessage({
          message: "Please login to view answers!",
          className: "red",
```

```
            })
            return false
        }
    },
    showAnswer() {
        if (this.isUserAccountAuth()) {
            const form: HTMLFormElement = document.querySelector('.exercise')
            const formElements: HTMLFormControlsCollection | any = form.elements
            for (let i = 0; i < formElements.length - 2; i++) {
                formElements[i].value = this.answers[i]
            }
        }
    }
  }
}
</script>
```

Файл “src/frontend/src/views/tasks/PerfectProgressiveTenseExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
  counter-reset: li;
}
.exercise .tasks li {
  list-style: none;
  position: relative;
  padding: .75em .5em;
  margin: 0 0 0 2em;
}
.exercise .tasks li::before {
  background-color: lightblue;
  color: var(--primary-color);
  border-radius: 50%;
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  -ms-box-sizing: border-box;
  box-sizing: border-box;
  content: counter(li);
  counter-increment: li;
  font-size: .875em;
  font-weight: 700;
  height: 1.785em;
  left: -2em;
  margin-top: -0.9em;
  padding-top: 0;
  position: absolute;
  text-align: center;
  top: 50%;
  width: 1.785em;
  line-height: 1.8em;
```

```
}
.exercise-textfield {
   width: 200px;
}
.correct {
   border-color: green;
   color: green;
}
.incorrect {
   border-color: red;
   color: red;
}
button {
   margin-right: 15px;
}
</style>
<template>
   <main>
      <h1>Perfect Progressive tense</h1>
      <container>
         <autotranslate-text>Use the verbs in fields in the Present Perfect Progressive in the sentences.
         </autotranslate-text>
         <br>
         <form class="exercise" @submit="onSubmitExercise">
            <ol class="tasks">
               <li>
                  I <input type="text" class="exercise-textfield"> this website since 1998. (to run)
               </li>
               <li>
                  My parents <input type="text" class="exercise-textfield"> for years. (to smoke)
               </li>
               <li>
                  She <input type="text" class="exercise-textfield"> for her friend since 7 o'clock. (to wait)
               </li>
               <li>
                  We <input type="text" class="exercise-textfield"> in Greece since last August. (to live)
               </li>
               <li>
                  Frassle and Rassle <input type="text" class="exercise-textfield"> their bikes all day. (to
ride)
               </li>
               <li>
                  They <input type="text" class="exercise-textfield"> in the sun for hours. (to lie)
               </li>
               <li>
                  Helena <input type="text" class="exercise-textfield"> on the computer for too long. (to
work)
               </li>
               <li>
                  It <input type="text" class="exercise-textfield"> since I came home. (to rain)
               </li>
               <li>
                  You <input type="text" class="exercise-textfield"> to music since 12:30. (to listen)
               </li>
               <li>
                  John <input type="text" class="exercise-textfield"> for the last 25 minutes. (to dream)
```

```
            </li>
          </ol>


          <pretty-button type="submit">Check</pretty-button>
          <pretty-button @click="showAnswer">Show Answer</pretty-button>
        </form>
      </container>
    </main>
</template>


<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
    data: () => {
        return {
            answers: ["have been running", "have been smoking", "has been waiting", "have been living",
"have been riding", "have been lying", "has been working", "has been raining", "have been listening",
"has been dreaming"]
        }
    },
    computed: mapGetters(["getIsAuth"]),
    methods: {
        ...mapMutations(["updateFlashMessage"]),
        onSubmitExercise(event: Event) {
            event.preventDefault()
            const form: HTMLFormElement = document.querySelector('.exercise')
            const formElements: HTMLFormControlsCollection | any = form.elements
            for (let i = 0; i < formElements.length - 2; i++) {
                if (formElements[i].value === this.answers[i]) {
                    formElements[i].classList.add("correct")
                    formElements[i].classList.remove("incorrect")
                } else {
                    formElements[i].classList.add("incorrect")
                    formElements[i].classList.remove("correct")
                }
            }
        },
        isUserAccountAuth(): boolean {
            if (this.getIsAuth) {
                return true
            } else {
                this.updateFlashMessage({
                    message: "Please login to view answers!",
                    className: "red",
                })
                return false
            }
        },
        showAnswer() {
            if (this.isUserAccountAuth()) {
                const form: HTMLFormElement = document.querySelector('.exercise')
                const formElements: HTMLFormControlsCollection | any = form.elements
                for (let i = 0; i < formElements.length - 2; i++) {
                    formElements[i].value = this.answers[i]
```

```
            }
        }
    }
  }
}
</script>
```

Файл “src/frontend/src/views/tasks/PerfectTenseExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
  counter-reset: li;
}
.exercise .tasks li {
  list-style: none;
  position: relative;
  padding: .75em .5em;
  margin: 0 0 0 2em;
}
.exercise .tasks li::before {
  background-color: lightblue;
  color: var(--primary-color);
  border-radius: 50%;
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  -ms-box-sizing: border-box;
  box-sizing: border-box;
  content: counter(li);
  counter-increment: li;
  font-size: .875em;
  font-weight: 700;
  height: 1.785em;
  left: -2em;
  margin-top: -0.9em;
  padding-top: 0;
  position: absolute;
  text-align: center;
  top: 50%;
  width: 1.785em;
  line-height: 1.8em;
}
.exercise-textfield {
  width: 200px;
}
.correct {
  border-color: green;
  color: green;
}
.incorrect {
```

```
    border-color: red;
    color: red;
}
button {
    margin-right: 15px;
}
</style>
<template>
    <main>
        <h1>Perfect tense</h1>
        <container>
            <autotranslate-text>Put in the verbs in fields into the gaps. Use Present Perfect.
            </autotranslate-text>
            <br>
            <form class="exercise" @submit="onSubmitExercise">
                <ol class="tasks">
                    <li>
                        Helena <input type="text" class="exercise-textfield"> me an e-mail. (to send)
                    </li>
                    <li>
                        Frassle and Rassle <input type="text" class="exercise-textfield"> the museum. (to visit)
                    </li>
                    <li>
                        I <input type="text" class="exercise-textfield"> at the pet shop. (to be)
                    </li>
                    <li>
                        They <input type="text" class="exercise-textfield"> already <input type="text"
                            class="exercise-textfield"> their backpacks. (to pack)
                    </li>
                    <li>
                        Alex <input type="text" class="exercise-textfield"> an accident. (to have)
                    </li>
                    <li>
                        We <input type="text" class="exercise-textfield"> the shopping for our grandmother. (to
do)
                    </li>
                    <li>
                        I <input type="text" class="exercise-textfield"> just <input type="text"
                            class="exercise-textfield"> my bike. (to clean)
                    </li>
                    <li>
                        Emily <input type="text" class="exercise-textfield"> her room. (to paint)
                    </li>
                    <li>
                        Alice and Alicia <input type="text" class="exercise-textfield"> to a concert. (to go)
                    </li>
                    <li>
                        My friends <input type="text" class="exercise-textfield"> smoking. (to give up)
                    </li>
                </ol>
                <pretty-button type="submit">Check</pretty-button>
                <pretty-button @click="showAnswer">Show Answer</pretty-button>
            </form>
        </container>
    </main>
```

```
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
   data: () => {
      return {
         answers: ["has sent", "have visited", "have been", "have", "packed", "has had", "have done",
"have", "cleaned", "has painted", "have gone", "have given up"]
      }
   },
   computed: mapGetters(["getIsAuth"]),
   methods: {
      ...mapMutations(["updateFlashMessage"]),
      onSubmitExercise(event: Event) {
         event.preventDefault()
         const form: HTMLFormElement = document.querySelector('.exercise')
         const formElements: HTMLFormControlsCollection | any = form.elements
         for (let i = 0; i < formElements.length - 2; i++) {
            if (formElements[i].value === this.answers[i]) {
               formElements[i].classList.add("correct")
               formElements[i].classList.remove("incorrect")
            } else {
               formElements[i].classList.add("incorrect")
               formElements[i].classList.remove("correct")
            }
         }
      },
      isUserAccountAuth(): boolean {
         if (this.getIsAuth) {
            return true
         } else {
            this.updateFlashMessage({
               message: "Please login to view answers!",
               className: "red",
            })
            return false
         }
      },
      showAnswer() {
         if (this.isUserAccountAuth()) {
            const form: HTMLFormElement = document.querySelector('.exercise')
            const formElements: HTMLFormControlsCollection | any = form.elements
            for (let i = 0; i < formElements.length - 2; i++) {
               formElements[i].value = this.answers[i]
            }
         }
      }
   }
}
</script>
```

Файл “src/frontend/src/views/tasks/ProgressiveTenseExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
```

```
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
   counter-reset: li;
}
.exercise .tasks li {
   list-style: none;
   position: relative;
   padding: .75em .5em;
   margin: 0 0 0 2em;
}
.exercise .tasks li::before {
   background-color: lightblue;
   color: var(--primary-color);
   border-radius: 50%;
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   -ms-box-sizing: border-box;
   box-sizing: border-box;
   content: counter(li);
   counter-increment: li;
   font-size: .875em;
   font-weight: 700;
   height: 1.785em;
   left: -2em;
   margin-top: -0.9em;
   padding-top: 0;
   position: absolute;
   text-align: center;
   top: 50%;
   width: 1.785em;
   line-height: 1.8em;
}
.exercise-textfield {
   width: 200px;
}
.correct {
   border-color: green;
   color: green;
}
.incorrect {
   border-color: red;
   color: red;
}
button {
   margin-right: 15px;
}
</style>
<template>
   <main>
      <h1>Progressive tense</h1>
      <container>
         <autotranslate-text>Put in the verbs in fields into the gaps. Use Present Progressive/Present
Continuous. Watch the punctuation and form sentences or questions.
         </autotranslate-text>
```

```
        <br>
        <form class="exercise" @submit="onSubmitExercise">
          <ol class="tasks">
            <li>
              The teacher <input type="text" class="exercise-textfield"> the door. (not/to close)
            </li>
            <li>
              <input type="text" class="exercise-textfield"> you <input type="text" class="exercise-
textfield"> the washing-up? (to do)
            </li>
            <li>
              They <input type="text" class="exercise-textfield"> the printers. (to check)
            </li>
            <li>
              <input type="text" class="exercise-textfield"> your grandfather <input type="text"
class="exercise-textfield"> at birds? (to look)
            </li>
            <li>
              We <input type="text" class="exercise-textfield"> the checkpoint. (not/to pass)
            </li>
            <li>
              <input type="text" class="exercise-textfield"> they <input type="text" class="exercise-
textfield"> to help? (to try)
            </li>
            <li>
              He <input type="text" class="exercise-textfield"> to the centre of the town. (not/to walk)
            </li>
            <li>
              <input type="text" class="exercise-textfield"> the policemen <input type="text"
class="exercise-textfield"> into the bank? (to run)
            </li>
            <li>
              Look! Steve's mother <input type="text" class="exercise-textfield"> a cake. (to make)
            </li>
            <li>
              You <input type="text" class="exercise-textfield"> the words correctly. (not/to link)
            </li>
          </ol>
          <pretty-button type="submit">Check</pretty-button>
          <pretty-button @click="showAnswer">Show Answer</pretty-button>
        </form>
      </container>
    </main>
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
    data: () => {
        return {
            answers: ["is not closing", "Are", "doing", "checking", "Is", "looking", "are not passing", "Are",
"trying", "is not walking", "Are", "running", "is making ", "are not linking"]
        }
    },
    computed: mapGetters(["getIsAuth"]),
    methods: {
        ...mapMutations(["updateFlashMessage"]),
```

```
    onSubmitExercise(event: Event) {
      event.preventDefault()
      const form: HTMLFormElement = document.querySelector('.exercise')
      const formElements: HTMLFormControlsCollection | any = form.elements
      for (let i = 0; i < formElements.length - 2; i++) {
        if (formElements[i].value === this.answers[i]) {
          formElements[i].classList.add("correct")
          formElements[i].classList.remove("incorrect")
        } else {
          formElements[i].classList.add("incorrect")
          formElements[i].classList.remove("correct")
        }
      }
    },
    isUserAccountAuth(): boolean {
      if (this.getIsAuth) {
        return true
      } else {
        this.updateFlashMessage({
          message: "Please login to view answers!",
          className: "red",
        })
        return false
      }
    },
    showAnswer() {
      if (this.isUserAccountAuth()) {
        const form: HTMLFormElement = document.querySelector('.exercise')
        const formElements: HTMLFormControlsCollection | any = form.elements
        for (let i = 0; i < formElements.length - 2; i++) {
          formElements[i].value = this.answers[i]
        }
      }
    }
  }
}
</script>
```

Файл “src/frontend/src/views/tasks/SimpleTenseExerciseView.vue”

```
<script lang="ts" setup>
import Container from '../../components/Container.vue'
import AutotranslateText from '../../components/AutotranslateText.vue'
import PrettyButton from '../../components/PrettyButton.vue'
</script>
<style lang="scss" scoped>
.exercise .tasks {
  counter-reset: li;
}
.exercise .tasks li {
  list-style: none;
  position: relative;
  padding: .75em .5em;
  margin: 0 0 0 2em;
```

```
}
.exercise .tasks li::before {
   background-color: lightblue;
   color: var(--primary-color);
   border-radius: 50%;
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   -ms-box-sizing: border-box;
   box-sizing: border-box;
   content: counter(li);
   counter-increment: li;
   font-size: .875em;
   font-weight: 700;
   height: 1.785em;
   left: -2em;
   margin-top: -0.9em;
   padding-top: 0;
   position: absolute;
   text-align: center;
   top: 50%;
   width: 1.785em;
   line-height: 1.8em;
}
.exercise-textfield {
   width: 200px;
}
.correct {
   border-color: green;
   color: green;
}
.incorrect {
   border-color: red;
   color: red;
}
button {
   margin-right: 15px;
}
</style>
<template>
   <main>
      <h1>Simple tense</h1>
      <container>
         <autotranslate-text>Put in the verbs in fields into the gaps. Use the Present Simple or the Past
Simple.
         </autotranslate-text>
         <br>
         <form class="exercise" @submit="onSubmitExercise">
            <ol class="tasks">
               <li>
                  Frassle and Rassle <input type="text" class="exercise-textfield"> to the park in the
afternoon. (to
                  go)
               </li>
               <li>
                  The shop <input type="text" class="exercise-textfield"> at 9 pm. (to close)
               </li>
```

```
            <li>
              I <input type="text" class="exercise-textfield"> breakfast every day. (to make)
            </li>
            <li>
              The boys <input type="text" class="exercise-textfield"> in London. (to live)
            </li>
            <li>
              Charlie <input type="text" class="exercise-textfield"> her friends on Fridays. (to meet)
            </li>
            <li>
              Roger <input type="text" class="exercise-textfield"> a nice dog. (to have)
            </li>
            <li>
              They <input type="text" class="exercise-textfield"> ice cream. (to like)
            </li>
            <li>
              Karen often <input type="text" class="exercise-textfield"> funny messages. (to send)
            </li>
            <li>
              We always <input type="text" class="exercise-textfield"> to the radio in the morning. (to
              listen)
            </li>
            <li>
              My friend <input type="text" class="exercise-textfield"> this song. (to love)
            </li>
          </ol>
          <pretty-button type="submit">Check</pretty-button>
          <pretty-button @click="showAnswer">Show Answer</pretty-button>
        </form>
      </container>
    </main>
</template>
<script lang="ts">
import { mapGetters, mapMutations } from 'vuex'
export default {
    data: () => {
      return {
        answers: ["go", "closes", "make", "live", "meets", "has", "like", "sends", "listen", "loves"]
      }
    },
    computed: mapGetters(["getIsAuth"]),
    methods: {
      ...mapMutations(["updateFlashMessage"]),
      onSubmitExercise(event: Event) {
        event.preventDefault()
        const form: HTMLFormElement = document.querySelector('.exercise')
        const formElements: HTMLFormControlsCollection | any = form.elements
        for (let i = 0; i < formElements.length - 2; i++) {
          if (formElements[i].value === this.answers[i]) {
            formElements[i].classList.add("correct")
            formElements[i].classList.remove("incorrect")
          } else {
            formElements[i].classList.add("incorrect")
            formElements[i].classList.remove("correct")
          }
        }
```

```
        },
        isUserAccountAuth(): boolean {
            if (this.getIsAuth) {
                return true
            } else {
                this.updateFlashMessage({
                    message: "Please login to view answers!",
                    className: "red",
                })
                return false
            }
        },
        showAnswer() {
            if (this.isUserAccountAuth()) {
                const form: HTMLFormElement = document.querySelector('.exercise')
                const formElements: HTMLFormControlsCollection | any = form.elements
                for (let i = 0; i < formElements.length - 2; i++) {
                    formElements[i].value = this.answers[i]
                }
            }
        }
    }
}
</script>
```

ДОДАТОК Б
ЗАУВАЖЕННЯ НОРМОКОНТРОЛЕРА

Таблиця Б.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
ПЗ	с. 29	Неплотне заповнення сторінок – подвійні пусті рядки перед пунктами, підпунктами тощо.
Додаток А		Додаток А має великий обсяг, внаслідок цього і вся ПЗ має великий обсяг -224 сторінки.