ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра комп'ютерної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Сергій КОВАЛЬОВ
(підпис) (ініціали, прізвище)

«___»_____2022 р.

Випускна кваліфікаційна робота

бакалавра
(освітньо-кваліфікаційний рівень)

на тему «Мобільний додаток "Електронний кабінет студента"»

Виконав студент 4 курсу, групи КІ-18
(шифр групи)

спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

Ільєнко М.О. _____
(прізвище та ініціали) (підпис)

Керівник Доц. каф. КІ, к.т.н., доц. Віталій ШАМАЄВ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент Доц. каі. ПМІ, к.т.н., доц. Ірина НАЗАРОВА _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Нормоконтроль:

_____ к.т.н., Юлія ДІКОВА

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

Студент _____
(підпис)

Луцьк – 2022 р.

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій і автоматизації
Кафедра комп'ютерної інженерії
Освітній ступінь бакалавр
Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____Сергій КОВАЛЬОВ
"____" _____20__ року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Ільєнку Миколі Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи студента "» «Мобільний додаток "Електронний кабінет

Керівник роботи **Шамаєв Віталій Віталієвич, к.т.н., доц.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від 06.05.2022 р. № 180

2. Строк подання студенткою роботи 07.06.2022 р.

3. Вихідні дані до роботи результати науково-дослідної роботи
результати переддипломної практики

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) виконати аналіз веб-сайтів декількох навчальних закладів України з метою виявлення корисної для студента функціональності; 2) виділити основні переваги та недоліки сучасних систем супроводу навчання; 3) сформувати функціональність мобільного додатку для відстеження успішності студента; 4) протестувати працездатність розробленого додатку та його вплив на рівень успішності студентів.

5. Консультанти розділів кваліфікаційної роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

6. Дата видачі завдання 22.04.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	*Назва етапів випускної кваліфікаційної роботи*	*Строк виконання етапів роботи*	*Примітка*
1	Аналіз предметної області	10.09.16-09.10.16	
2	Постановка завдання	10.10.16-10.11.16	
3	Огляд тематики роботи, актуальність роботи, формулювання мети на задач дослідження	11.11.16-20.01.17	
4	Опис та обґрунтування основних ідей роботи	21.01.17-04.02.17	
5	Розробка апаратних або програмних засобів	05.02.17-02.04.17	
6	Дослідження запропонованих ідеї, аналіз результатів дослідження, формулювання висновків та тенденцій	03.04.17-14.05.17	
7	Оформлення пояснювальної записки	15.05.17-30.05.17	
8	Нормоконтроль пояснювальної записки та додаткових матеріалів	01.06.17-07.06.17	
9	Рецензування роботи	08.06.17-12.06.17	
10	Захист роботи	згідно графіку	

Студент _____ Ільєнко М.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шамаєв В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Ільєнко М. О. Мобільний додаток “Електронний кабінет студента” / Випускна кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 123 Комп’ютерна інженерія. – ДВНЗ ДонНТУ, Луцьк, 2022.

Мета даної роботи полягає у створенні додатку для організації взаємодії студента між викладачами та навчальним закладом та покращення аналізу результатів навчання.

В кваліфікаційній роботі вирішено такі задачі:

1) виконання аналізу веб-сайтів декількох навчальних закладів України з метою виявлення корисної для студента функціональності;

2) виділиення основнив переваг та недоліків сучасних систем супроводу навчання;

3) формування функціональності мобільного додатку для відстеження успішності студента;

4) тестування працездатності розробленого додатку та його впливу на рівень успішності студентів.

Практичним результатом кваліфікаційної роботи є: мобільний додаток в якому реалізовано всі функціональні вимоги.

Ключові слова:

МОБІЛЬНИЙ ДОДАТОК, ОС ANDROID, УСПІШНІСТЬ СТУДЕНТА, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, БАЗА ДАНИХ, МОВА ПРОГРАМУВАННЯ.

ANNOTATION

ILIENKO M. O. Mobile application “Student’s electronic account” / Graduate qualification work for bachelor’s degree in 123 Computer Engineering of DonNTU, Lutsk, 2022.

The purpose of this work is to develop an application for organization of interaction between stundents and their lecturers or educational institution, improving the analysis of learning results.

The following tasks were solved in the graduate qualification work:

– analazing websites of several educational institutions of Ukraine to identify useful functionality for the student;

– identifying the main strengths and weaknesses of current training support;

– creating the functionality of mobile application for tracking student progress;

– testing application performance and its impact on student achievement.

The practical result of the qualification work is: the mobile application, in which all functional requirements are implemented

Keywords: mobile application, OS Android, student achievement, information technology, learning process, programming language, Google services, information education, database.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
SQL	Structured Query Language
БД	База Даних
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language
JSON	Javascript Object Notation
ASP.NET	Active Server Pages
AJAX	Asynchronous Javascript And XML
КПК	Кишеньковий Персональний Комп'ютер
PDA	Personal Digital Assistants
ЕОМ	Електронна Обчислювальна Машина
ІКТ	Інформаційно-Комунікаційні Технології
ТЗН	Технічні Засоби Навчання
SDK	Software Development Kit
IDE	Integrated Development Environment
MVVM	Model View Viewmodel
WPF	Windows Presentation Foundation
GCM	Google Cloud Messaging
API	Application Programming Interface

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 6

ВСТУП 9

1ЗАГАЛЬНА ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ, ЩО ВИКОРИСТОВУЮТЬСЯ У НАВЧАННІ СТУДЕНТІВ 11

1.1 Актуальність використання ІТ-технологій у навчанні студента 11

1.2 Сучасний стан та аналоги веб-сайтів, які використовуються у навчанні 13

1.3 Специфікація вимог до додатку 22

1.4 Опис предметної області 27

1.5 Постановка завдань для дослідження додатка відстеження успішності студента 28

2ДЕТАЛЬНИЙ АНАЛІЗ ЗАСОБІВ ЗА ДОПОМОГОЮ ЯКИХ БУДЕ ПРОВЕДЕНО ДОСЛІДЖЕННЯ 30

2.1 Інструментальні засоби для реалізації 30

2.1.1 ОС Android 31

2.1.2 Фреймворк Xamarin 32

2.1.3 Мова програмування - C# 34

2.1.4 Патерн MVVM 35

2.2 Аналіз Google додатків та сервісів, які будуть використані для створення програми 37

2.3 Висновки до розділу 40

3РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ 41

3.1 Розробка архітектури додатку 41

3.2 Проектування структури бази даних. 42

3.3 Програмна реалізація 47
3.4 Висновки до розділу 48
4ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ 49
4.1 Огляд мобільного додатка відстеження успішості студента 49
4.2 Тестування програми ..... **Error! Bookmark not defined.**
4.3 Можливі варіанти розвитку програмного додатка 57
4.4 Висновки до розділу 57
ВИСНОВКИ 58
ПЕРЕЛІК ПОСИЛАНЬ 59

ВСТУП

Актуальність теми.

Найважливішою із складових навчального процесу є контроль рівня знань і вмінь студентів. При навчанні та інших видах людської діяльності, перевірка і контроль за ними, завжди мають вплив на ставлення людини до виконання обов'язків, на якість та ефективність роботи людини. Останнім часом ми зіткнулися з проблемою організації дистанційного навчання, а в першу чергу з самостійним відстеженням успішності студентів, та можливості зручного зворотнього зв’язку. Навчальний процес повинен бути організований так, щоб студент самостійно прагнув до оволодіння знаннями. При цьому студент повинен оцінювати свій рівень підготовки, вибирати теми і визначати рівень засвоєння знань. Інформаційні технології дозволяють полегшити проблему з організацією дистанційного навчання та відстеженням студентами їх успішності при навчанні. Моя робота присвячена актуальній проблемі використання інформаційних технологій (у вигляді Android-додатку) для покращення аналізу успішності студента, та поліпшенням організації зовротнього зв’язку з викладачами та навчальним закладом.

Мета і задачі розробки.

Мета роботи – створення додатку для організації взаємодії студента між викладачами та навчальним закладом та покращення аналізу результатів навчання.

Для досягнення мети необхідно вирішити наступні **задачі**:

1) виконати аналіз веб-сайтів декількох навчальних закладів України з метою виявлення корисної для студента функціональності;
2) виділити основні переваги та недоліки сучасних систем супроводу навчання;
3) сформувати функціональність мобільного додатку для відстеження успішності студента;
4) протестувати працездатність розробленого додатку та його вплив на рівень успішності студентів.

Методи, засоби та технології розробки.

Для реалізації додатку була обрана клієнт-серверна архітектура, яка наддасть змогу одночасного доступу до системи багатьом користувачам. Технологією розробки для цього було обрано Visual Studio, а мову – C#.

Об'єкт дослідження – інформаційна підтримка процесу навчання учня (студента) з метою відстеження його успішності.

Предмет дослідження – мобільний додаток для відстеження успішності студента та зворотнього зв’язку з викладачами і вищим навчальним закладом.

Структура та обсяг кваліфікаційної роботи. КРБ складається зі вступу, чотирьох розділів, висновку і двох додатків, які викладені на 98 сторінок машинописного тексту, містить 40 рисунок та 10 таблиць. Список використаної літератури складається з 38 найменувань.

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ, ЩО ВИКОРИСТОВУЮТЬСЯ У НАВЧАННІ СТУДЕНТІВ

1.1 Актуальність використання ІТ-технологій у навчанні студента

В наш час мобільні пристрої (планшети, смартфони, КПК, та ін.) надають багато можливостей користувачам, наприклад: робота в мережах з виходом в Інтернет; підтримка переносних носіїв даних; використання різноманитних додатків з великим набором корисних функцій. Можна сказати, що можливостей які надають сучасні мобильні девайси майже повністю вистачає для дистанційної і зручної роботи в різних сферах праці, таких як: бізнес, освіта, наука та ін. Особливо перспективним є використання мобільних пристроїв у сфері дистанційного навчання. Проблема застосування нових технічних засобів у навчанні не є новою, а потреба у використанні мобільних пристроїв при навчанні виникла ще з поширенням переносних мобільних пристроїв в кінці ХХ-го століття. З розвитком мобільних технологій зросла можливість швидкого й зручного доступу до обміну і перегляду інформації. А в останні роки у зв’язку з епідемією Covid-19 потрібність у зручному дистанційному навчанні миттєво виросла до необхідності. За останні роки були різні думки про те, що саме є мобільним навчанням [1]. Різноманітність думок частково пов’язана з швидким розвитком мобільного навчання. Європейська гільдія з електронного навчання визначає його так: будь-яка діяльність, яка дозволяє людям бути більш продуктивними у споживанні та створенні інформації, а також взаємодії з нею компактними цифровими пристроями, якщо людина виконує ці дії на регулярній основі, має надійний зв’язок, і пристрій може бути розташованим в кишені або сумочці. Отже, термін «мобільне навчання» (м-навчання), або mobile learning (m-learning) [2], відноситься до використання у викладанні та навчанні за допомогою мобільних і портативних ІТ-пристроїв, таких, як кишенькові комп’ютери PDA (Personal Digital Assistants), мобільні телефони, ноутбуки, нетбуки, планшетні ПК, iPhone, iPad та інші. Засоби

мобільних інформаційно-комунікаційних технологій (ІКТ) навчання можна поділити на апаратні та програмні. Існують різні програмні мобільні засоби навчання, наприклад: MobileELDIT, MLEX, MLE-Moodle, AmadeusLMSMobile, LearnCast, Mobl21 та ін [3]. Сучасні апаратні пристрої, на які встановлені різні операційні системи (Android, BlackBerry, iOS, WindowsCE та ін.), безкоштовно підтримують різні мовні функціональні можливості без додаткового ПЗ. Під функціональними можливостями ми розуміємо різні сервіси, які пропонують операційній системі і самі пристрої своїм користувачам. На платформі цих систем можна встановити безкоштовні додатки, такі як браузер (Opera, GoogleChrome та ін.), різні програми мессенджери (Telegram), перекладачі (GoogleTranslate), програми для перегляду електроних книг та документів (Adobe Acrobat, Microsoft Word), пошту (Gmail) та ін. Наприклад браузер GoogleChrome, встановлений на платформі ОС Android або Windows, дозволяє зручно переглядати інформацію в мережі Інтернет, також має фунції як – автоматичний переклад сторінок сайтів, перегляд мобільної чи настільної версію сайтів, додавання закладок для спрощення пошуку улюблених сайтів. Комп'ютери, мобільні пристрої та Інтернет-ресурси стали невід'ємною частиною навчального процесу, і з'являються більш прості та ефективні пристрої, відкриваються нові можливості для розширення сфер використання інформаційно-комунікаційних технологій. Варто відзначити, що мобільні пристрої коштують дешевше, ніж настільні ПК, і є більш дешевим засобом доступу до Інтернет-ресурсів. Тому особливої актуальності набуває пошук нових підходів до організації навчального процесу і створення навчальних матеріалів і технологій, які б враховували можливості мобільних пристроїв.

1.2 Сучасний стан та аналоги веб-сайтів, які використовуються у навчанні

Перш ніж почати розробляти певну систему, потрібно спочатку здійснити пошук аналогів, дослідити, проаналізувати їх переваги та недоліки, щоб зрозуміти, яких помилок потрібно уникнути та які їх переваги доцільно реалізувати.

У межах роботи проаналізовано чотири програмні системи-аналоги, які реалізують схожі функції предметної області:

1) всеукраїнська безкоштовна освітня мережа «Щоденник.ua» [11];

2) система призначена для перегляду модульних оцінок з предметів що викладались в поточному семестрі [12];

3) система призначена для перегляду кількості пропущених занять та розкладу ТНЕУ [13];

4) система МуАТ [14] де вчителі та викладачі можуть повідомляти про відвідуваність онлайн, відслідковувати студентів, надсилати повідомлення їх батькам і багато іншого в масштабі реального часу.

Розглянемо основні можливості та недоліки зазначених систем-аналогів.

На рис. 1.1 зображено головну сторінку Всеукраїнської освітньої мережі «Щоденник.ua». Основними перевагами є доступ до інформації про будь яку школу України та надає перевагу використовувати розклад уроків, електронний журнал, електронний щоденник, шкільний сайт та шкільну медіатеку.

На сайті загальну інформацію може переглядати, як і зареєстрований та не зареєстрований користувач, але тільки для зареєстрованого є можливість побачити детальну інформацію учня. Також присутній захист приватності, оскільки реєстрація відбувається тільки за кодами-запрошенням та інші користувачі не мають доступу до вашої інформації.

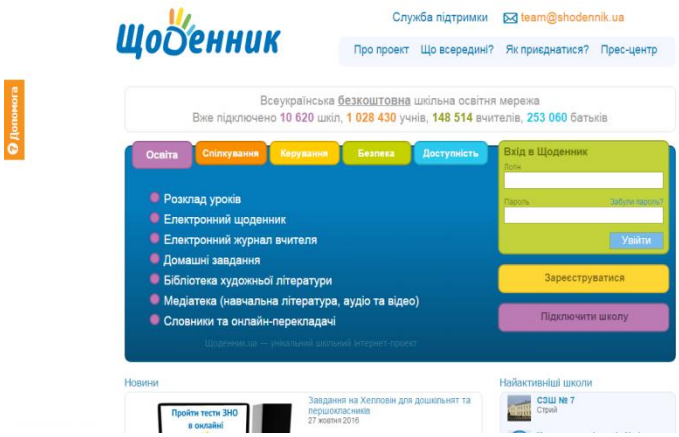

Рисунок 1.1 – Головне вікно сайту

Кожному учневі в «Щоденнику» доступні всі його оцінки з різних предметів. Крім того, доступна можливість перегляду оцінок з предметів і за певний період (тиждень, семестр). Загальний вигляд вкладки «Щоденник» подано на рис. 1.2.

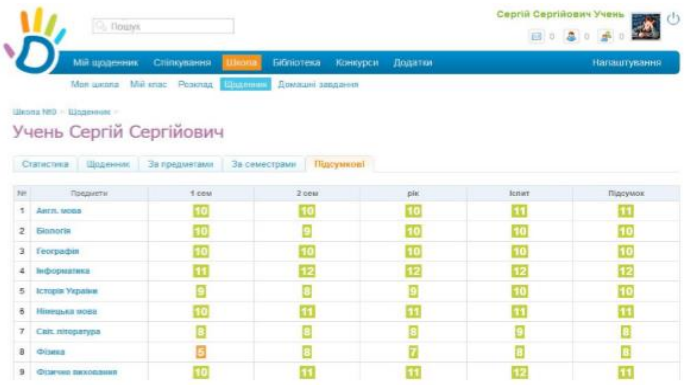

Рисунок 1.2 – Вкладка «Щоденник»

Вчителі в «Електронному журналі» можуть виставляти оцінки в тих класах, в яких вони викладають, а за наявності адміністративних прав і в будь-якому класі школи. На рис. 1.3 показано загальний вигляд вкладки «Електронний журнал».

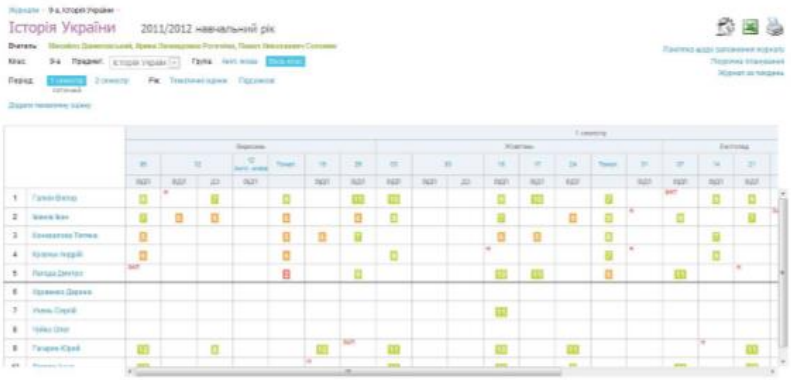

Рисунок 1.3 – Вкладка «Електронний журнал»

На сторінці конкретної школи міститься коротка інформація про школу, список керівництва та адміністраторів, останні новини, зміни в форумі, останні завантажені файли (рис. 1.4).

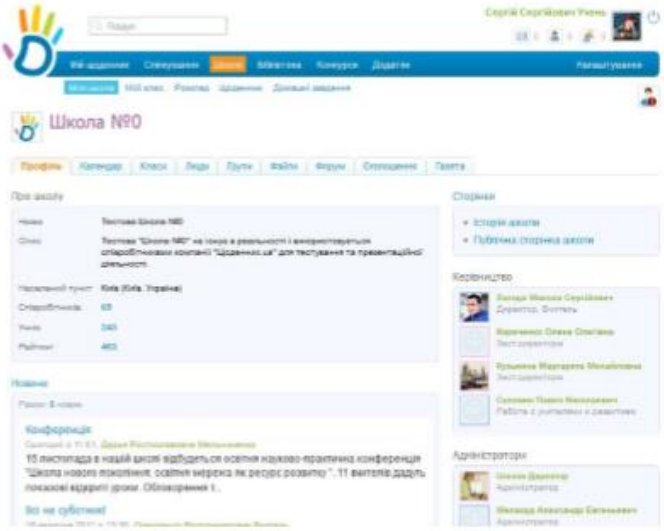

Рисунок 1.4 – Сторінка конкретної школи у системі «Щоденник»

Для кожного користувача «Щоденник» веде особистий календар, в якому відображується інформація про розклад уроків, дні народження та інші події (рис. 1.5).

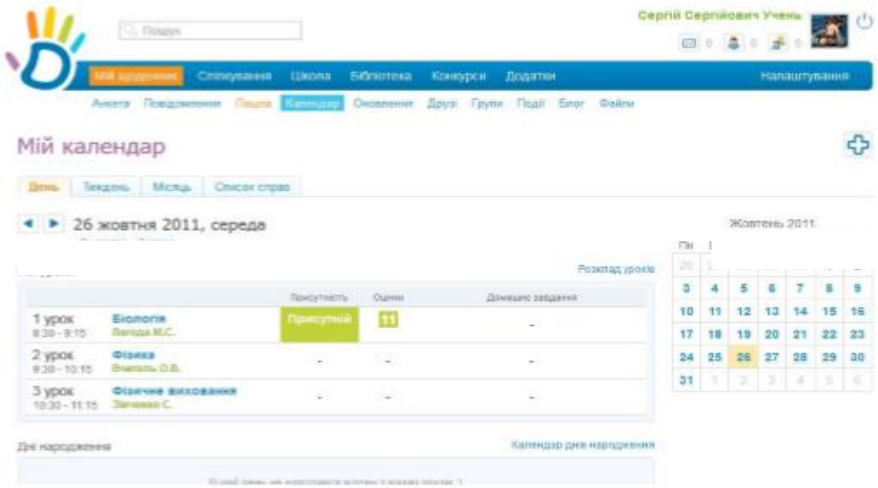

Рисунок 1.5 – Сторінка особистого календаря користувача

На сайті Rating system (ТНЕУ), головне вікно якого наведено на рис. 1.6, є вкладки авторизації та реєстрації, також є вкладка з наведеним переліком кроків які необхідно зробити для реєстрації та контактна інформація на випадок втрати даних для авторизації.

Рисунок 1.6 – Головне вікно сайту

Реєстрація в системі складається з двох частин:

1) На сторінці реєстрації потрібно заповнити всі поля (рис. 1.7).

2) Звернутися у відповідний деканат з проханням активації свого облікового запису в системі.

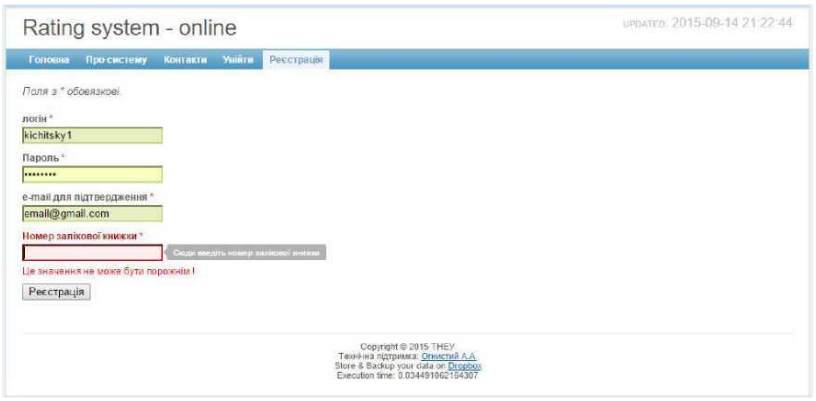

Рисунок 1.7 – Вікно реєстрації в системі

При успішній авторизації студент може переглянути свої модульні оцінки, та загальну оцінку за семестр. Вікно успішності показано на рис. 1.8.

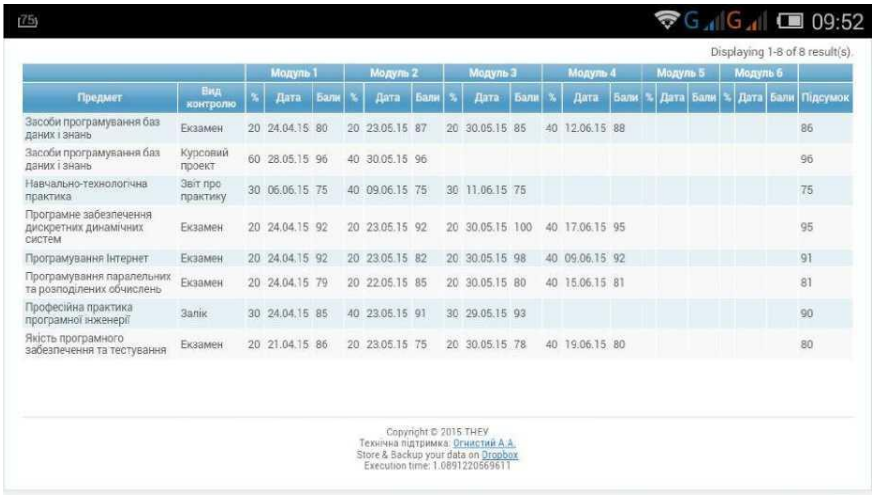

Рисунок 1.8 – Відображення «Таблиця успішності»

Система FCIT TEACH. На рис. 1.9 зображено головне вікно сайту FCIT TEACH (ТНЕУ). На головній сторінці сайту є форма авторизації, вкладки з

наведеним переліком критеріїв пошуку на сайті (розкладу студента, розкладу викладача, та пошуку вільної аудиторії).

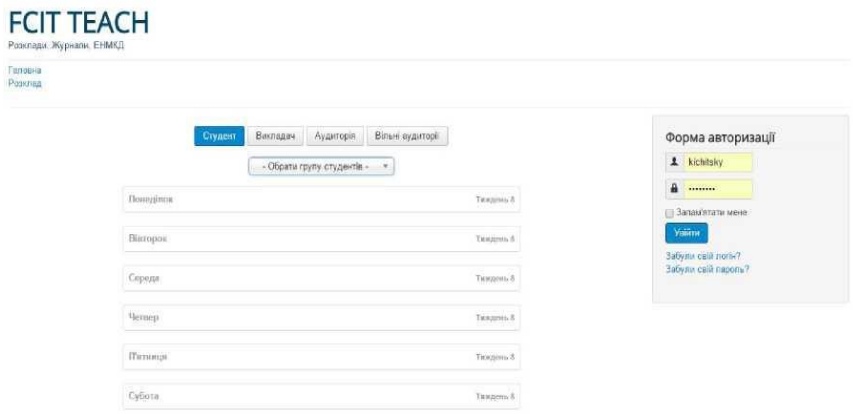

Рисунок 1.9 – Головне вікно сайту

При успішній авторизації студент може переглянути кількість пропущених занять (без ПП, не відпрацьованих), перелік довідок та заяв. Вікно пропущених занять показано на рис. 1.10.

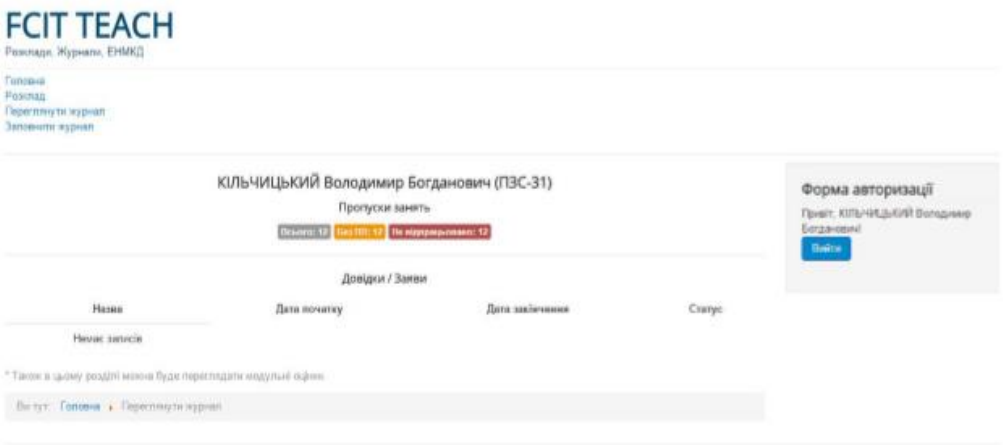

Рисунок 1.10 – Вікно пропущених занять

Також студент може переглянути розклад занять на поточний навчальний тиждень (як своєї групи так і всіх інших груп університету). Вікно розкладу занять показано на рис. 1.11.

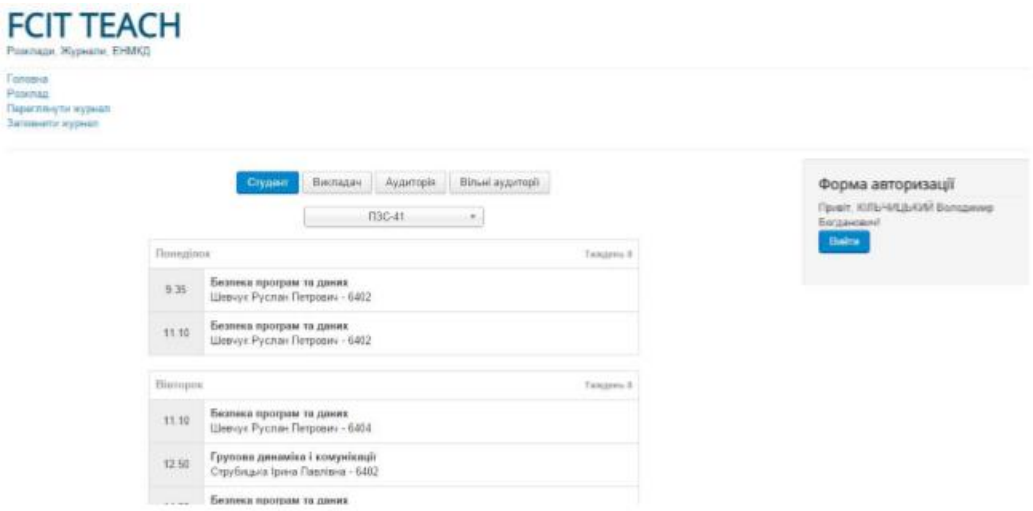

Рисунок 1.11 – Вікно розклад занять

Також ми можемо переглянути список вільних аудиторій на поточний навчальний тиждень, на потрібну годину, в потрібний. Вікно списку вільних аудиторій показано на рис. 1.12.

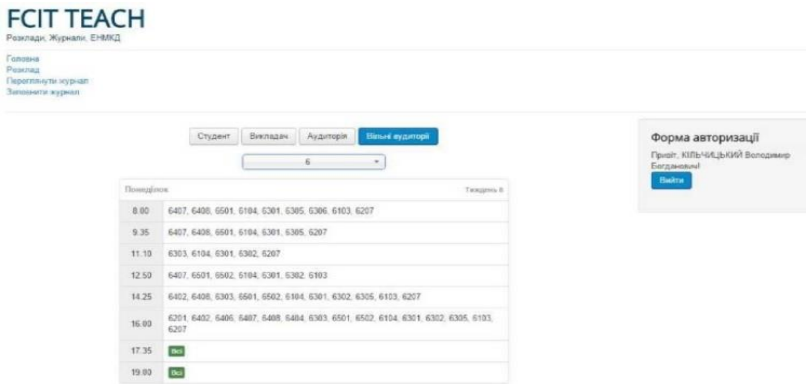

Рисунок 1.12 – Вікно пошуку вільних аудиторій

Наступною розглянемо систему MyAT. На рис. 1.13 зображено головну сторінку сайту відстежування відвідуваності MyAT (MyAttendanceTracker). На даній сторінці відображаються різні вкладки, можна переглянути інформацію про систему, її авторів, особливості системи; авторизації та реєстрації; також відображені новини освіти у світі.

Рисунок 1.13 – Головне вікно сайту

Також ведеться облік зареєстрованих навчальних закладів, зареєстрованих користувачів, та студентів які відслідковуються системою, як це зображено на рис. 1.14.

Рисунок 1.14 – Облік користувачів

Кожен користувач MyAT має свою особисту сторінку на котрій відображаються списки його студентів, предметів які він відвідує, його

особиста відвідуваність, також тут студент може розповісти про себе, про свої інтереси, завантажити фотографії, як це зображено на рис. 1.15.

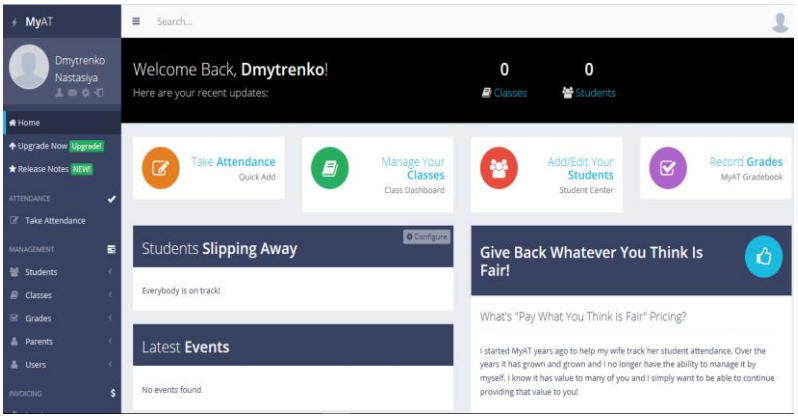

Рисунок 1.15 – Особиста сторінка користувача сайту

Вчителі в Електронному журналі можуть повідомляти про відвідуваність студентів в тих класах, в яких вони викладають, як це зображено на рис. 1.16.

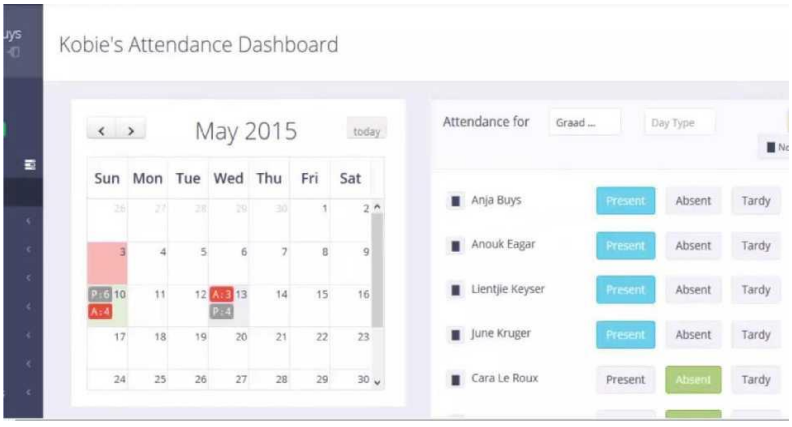

Рисунок 1.16 – Сторінка особистого календаря користувача

Порівняльна характеристика розглянутих систем наведена у табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика програмних продуктів

Назва програмного продукту	Щоденник	Rating system	FCIT TEACH	MyAT
Інтерфейс користувача	WEB-система	WEB-система	WEB-система	WEB-система
Функціональність	- Профіль користувача/ школи - Перегляд користувачів - Перегляд шкіл - Щоденник - Журнал	- Профіль - Про систему - Допомога	- Профіль - Журнал - Вільні аудиторії - Розклад студента/ викладача	- Профіль користувача/ школи - Перегляд користувачів - Перегляд шкіл - Журнал огляду відвідуваності

1.3 Специфікація вимог до додатку

Після огляду аналогів, детального аналізу їх функціонування та визначення основних переваг і недоліків кожного з них перейдемо до етапу опису специфікації вимог до програмного продукту – документу, котрий містить повний опис продукту який розроблюють. Специфікація вимог є необхідним документом для зворотного зв’язку з замовником системи на початку проектування, тому повинна бути написана в простій та зрозумілій формі. До основних етапів формування специфікації можна віднести: - створення глосарію розроблюваного продукту; - опис варіантів використання системи, як зі сторони користувача, так і з керуючої сторони; - опис основних функціональних та не функціональних вимог. Отже, першочергово при розробці специфікації вимог створюють словник вузьконаправлених

термінів, які будуть використані в процесі проектування, реалізації та експлуатації програмного продукту. Глосарій системи наведено в табл. 1.2.

Таблиця 1.2 – Глосарій системи

Термін	Опис терміну
Авторизований користувач	Має можливість використовувати усі функції, які надаються сайтом.
Android-додаток	Веб-додаток - на платформі Android.
Адресат	Той, кому адресується лист.
Авторизація	Це надане будь-якій особі право на вчинення певних дій в конкретній системі - на сайті або терміналі, наприклад, при використанні онлайн-банкінгу. Авторизація дозволяє виключити доступ до інформації небажаних лиць та надання певних повноважень на виконання деяких дій у системі.
Електронна пошта	Це технологія та сервіс для надсилання й отримання електронних повідомлень (під назвою "листи" або "Електронна пошта") по комп'ютерній мережі (в тому числі глобальній).
Реєстрація	Процес при якому користувач вказує ПТБ (логін), яким будуть підписуватись його пропозиції, також підтверджується поштова адреса та встановлюється пароль на акаунт.
Користувач	Це та особа, яка без авторизації має можливість переглядати сайт, але немає доступу до всіх його функцій.
Android	Операційна система для комунікаторів, планшетних комп'ютерів, цифрових програвачів, наручних годинників, нетбуків і смартфонів, заснована на ядрі Linux.

Варіанти використання представленні у таблицях 1.3 – 1.6

Таблиця 1.3 – Варіант використання реєстрація

Контекст використання	Реєстрація
Дійові особи	Користувач(студент)
Передумова	Доступ до мережі інтернет, користувач перейшов на сторінку «реєстрація»
Тригер	Відкрита сторінка реєстрації
Сценарій	Заповнюємо поле “Login” Заповнюємо поле “Password” Натиснути клавішу «Реєстрація»
Пост-умова	Користувач зареєстрований

Для того, щоб розпочати роботу з додатком, користувач(студент) повинен авторизуватися, варіант використання процесу «Авторизація» представлений у табл. 1.4.

Таблиця 1.4 – Варіант використання «Авторизація»

Контекст використання	Авторизація
Дійові особи	Користувач (студент)
Передумова	Повинен бути зареєстрований
Тригер	Відкрита сторінка авторизації
Сценарій	Заповнюємо поле “Login” Заповнюємо поле “Password”
Пост-умова	Користувач авторизований

Після успішної авторизації, система ідентифікує користувача, та виведе на екран головне меню додатку. Далі потрібно вибрати вкладку «Розклад занять» і система покаже йому повну інформацію про розклад поточного

семестру можна обрати верхній чи нижній тиждень, варіант використання процесу «Перегляд розкладу занять» представлений у табл. 1.5.

Таблиця 1.5 – Варіант використання «Перегляд розкладу занять»

Контекст використання	Перегляд успішності
Дійові особи	Користувач (студент)
Передумова	Користувач авторизований
Тригер	Відкрита вкладка Розклад занять
Сценарій	Розклад занять на поточний семестр відображається після натиску на вкладку «Розклад занять».
Пост-умова	Показано розклад занять на поточний семестр (можна обрати нижній чи верхній тиждень)

Також користувач може увійти до вкладки «Пошта», щоб мати можливість переглянути чи надіслати листа викладачеві або деканату, достатньо набрати адресатів, та натиснути кнопку «Відправити», варіант використання процесу «Відправка листа» представлений у табл. 1.6.

Таблиця 1.6 – Варіант використання «Пошта»

Контекст використання	Пошта
1	2
Дійові особи	Користувач (Студент)
Тригер	Відкрито додаток
Сценарій	Натискаємо на вкладку «Пошта» Набираємо у строці адресанта,. Натискаємо на кнопку «Відправити»
Пост-умова	Повідомлення було надіслано обраному адресатну.

Специфікація функціональних вимог наведена у табл. 1.7

Таблиця 1.7 – Специфікація функціональних вимог

Ідент. вимог	Назва вимоги	Атрибути вимог		Контакт
		Пріоритет	Складність	
1.	Перегляд функції «Реєстрація»	Високий	Середня	Користувач
2.	Перегляд функції «Авторизація»	Високий	Середня	Користувач
3.	Перегляд функції «Перегляд Розкладу занять»	Високий	Середня	Користувач
4.	Перегляд функції «Відправка листа»	Високий	Середня	Користувач
5.	Перегляд функції «Перегляд списку студентів»	Високий	Середня	Користувач

Специфікація нефункціональних вимог наведена у табл. 1.8.

Таблиця 1.8 – Специфікація нефункціональних вимог

№	Назва вимоги	Характеристики
1	Застосовність	
1.1	Час для навчання користувачів	Не більше 10 хв.
1.2	Вимірюваний час відгуку для типових завдань	Не повинно відрізнятися від аналогів
2	Надійність	
2.1	Середній час безвідмовної роботи	10 год.
2.2	Середнє напрацювання до ремонту	3 місяці
3	Проектні обмеження	
3.1	Мова програмування	C#

1.4 Опис предметної області

Метою діяльності освітньої організації є забезпечення сучасного якісної освіти. Контроль успішності - комплексний процес, в якому беруть участь навчальна частина вузу, кафедри і деканат.

– навчальна частина відповідає за формування освітніх програм, робочих і семестрових навчальних планів, ведення та облік контингенту, складання розкладу, складання переліку навчальних дисциплін;

– кафедри проводять навчальні заняття і приймають рішення про допуск або недопуск студентів, до проміжної атестації з дисципліни на підставі поточної успішності. Відомості про студентів, які мають заборгованість передаються в деканат відповідного факультету;

– деканати здійснюють контроль за проведенням проміжної атестації, прийом і аналіз атестаційних відомостей, які надійшли з кафедр, видачу атестаційних листів на перездачі, ведення навчальних карток студентів, формування наказів про рух контингенту.

На підставі навчальних планів для кожного факультету будуються навчальні графіки і розклади занять, визначається перелік дисциплін для кожного курсу і семестру навчання. Кожній дисципліні відповідає форма контролю. За підсумками здачі заліку/іспиту в атестаційну відомість виставляється оцінка (залік) і рейтинг студента з дисципліни.

Заповнені атестаційні відомості здаються в деканат відповідного факультету, де методисти розносять відомості по навчальній картці кожного студента. Ті студенти, які успішно пройшли проміжну атестацію наказом допускаються до навчання (перекладається на наступний курс). У разі не допуску до сесії, виставлення «незадовільно» або неявки на іспит/залік студент перездає дисципліну в призначений деканатом день поза групою. Результати перездач, кількість спроб фіксуються. Застосування інформаційних технологій спрощує і прискорює їх роботу, зменшує кількість помилок «людського фактора». Після проектування і розробки мобільного додатку успішності студентів, передбачається, що дані про результати сесій,

що надходять з кафедр повинні завантажуватися в систему, для того, щоб сформувати статистику успішності, заборгованості, рейтинг студента.

1.5 Постановка завдань для дослідження додатка відстеження успішності студента

Основним завданням КРБ є створення мобільного додатку, ща надасть можливість перегляду поточних оцінок студента, екзаменаційних оцінок, обліку відвідування студентами занять, а також інформацію про рейтинг студента у групі. Додаток повинен служити для оперативного перегляду даних про студентів з будь-якого мобільного пристрою на базі ОС Android, підключеного до мережі Інтернет.

У мобільному додатку будуть відбуватися наступні процеси:

1) реєстрація:
    а) верифікація заповнених полів;
    б) додання в БД;
    в) відображення звіту про успішність реєстрації.
2) реєстрація через google аккаунт:
    а) верифікація заповнених полів;
    б) додання в БД;
    в) відображення звіту про успішність реєстрації;
    г) відображення інформації з google аккаунта.
3) авторизація:
    а) ідентифікація користувача.
4) отримання необхідної інформації, щодо:
    а) розкладу занять;
    б) розкладу екзаменів;
    в) результатів зданих робіт та екзаменів/заліків;
    г) рейтингу студента у групі;
    ґ) обліку відвідуваності студентом кожної дисципліни.
5) відправка повідомлень студентам групи:

а) вибір поштового сервісу для відправки повідомлення;

б) вибір адресатів;

в) відправка повідомлення.

6) можливість налаштувати мобільний додаток:

а) підключення розсилки на електронну пошту листів с деканата та інших студентів;

б) кнопка для включення/відключення повідомлень на мобільний пристрій у якості прапорця на іконці додатка та в стрічці повідомлень;

в) кнопка для включення/відключення додавання до календаря розкладу екзаменів або заліків.

2 ДЕТАЛЬНИЙ АНАЛІЗ ЗАСОБІВ ЗА ДОПОМОГОЮ ЯКИХ БУДЕ ПРОВЕДЕНО ДОСЛІДЖЕННЯ

2.1 Інструментальні засоби для реалізації

Світ мобільних платформ сильно фрагментований, тут виділяються дві основні операційні системи - Android і iOS, а також платформу Windows Phone/Windows 10 Mobile. Є певні статистичні дані, що значна частина мобільних додатків створюється більш ніж для однієї платформи, наприклад, для Android і iOS [15]. Однак неминуче розробники стикаються з наступними труднощами:

– відмінність у підходах побудови графічного інтерфейсу так чи інакше впливає на розробку. Розробники змушені підлаштовувати додаток під вимоги до інтерфейсу на конкретній платформі;

– різні API - розбіжність у програмних інтерфейсів і реалізації тих чи інших функциональностей також вимагає від програміста облік цих специфічних особливостей;

– різні платформи для розробки. Наприклад, щоб створювати додатки для iOS нам необхідне відповідне середовище - Mac OS X і ряд спеціальних інструментів, типу XCode. А в якості мови програмування вибирається Objective-C або Swift. Для Androida ми можемо використовувати найрізноманітніший набір середовищ - Android Studio, Eclipse і т.д. Але тут для переважної більшості додатків застосовується Java [16].

Саме Xamarin дозволяє створювати одну єдину логіку додатка із застосуванням C # і .NET відразу для всіх трьох платформ - Android, iOS, Windows Mobile. Тобто Xamarin представляє технологію для кроссплатформенної розробки мобільних додатків.

В процесі розробки створюється єдиний код для всіх платформ. При створенні додатків ми можемо використовувати платформу .NET і мова програмування C #, яка є досить продуктивною, в той же час ясною і простою для освоєння та застосування. Ці субплатформи відіграють велику

роль - через них додатки можуть напрявляти запити до прикладних інтерфейсів на пристроях під управлінням ОС Android або iOS.

Завдяки цим платформам ми можемо створювати окремо додатки для Android, окремо для iOS, але найбільш важливою особливістю платформи є можливість створювати кросплатформені додатки - тобто одна логіка для всіх платформ. Одже ми один раз можемо визначити візуальний інтерфейс, один раз до нього прив'язати якусь логіку на C #, і все це буде працювати на Android, iOS і Windows Phone. Дана можливість представлена технологією Xamarin.Forms. І саме цю технологію ми і будемо використовувати у створенні додатка успішності студента [17].

2.1.1 ОС Android

Android це унікальна операційна система. Розробник додатків повинен знати її особливості та нюанси для отримання хорошого результату. Існують деякі труднощі, які потрібно враховувати при розробці. Перерахую їх коротко:

Додаток вимагає для установки в два рази (або навіть в чотири) більше місця, ніж оригінальний розмір програми;

Швидкість роботи з файлами на вбудованої флеш-карті падає в десятки разів при зменшенні вільного місця;

Кожен процес може використовувати до 16 Мб (іноді 24 Мб) оперативної пам’яті. Між додатком і ядром лежить шар API і шар бібліотек на нативному коді. Додаток виконується на віртуальній машині Java (Dalvik Virtual Machine).

В Android можна запускати багато додатків. Але одне з них є головним і займає екран. Від поточної програми можна перейти до попередньої або запустити нове. Це схоже на браузер з історією переглядів [18].

Кожен екран для користувача інтерфейсу представлений класом Activity в коді. Різні Activity містяться в процесах. Activity може навіть жити довше процесу, також бути припиненим і запущеним знову зі збереженням

всієї потрібної інформації. Використовує спеціальний механізм опису дій заснований на Intent. Коли потрібно виконати дію (зробити дзвінок, надіслати листа, показати вікно), викликається Intent.

Також Android містить сервіси подібні демонам в Linux для виконання потрібних дій у фоновому режимі (наприклад, програвання музики). Для обміну даними між додатками використовуються Content providers (провайдери вмісту).

Додатки для Android в своїй роботі використовують вікна (аналогічно Windows), проте в даній системі вищевказані вікна носять іншу назву - Activity. Як і в Windows, кожне вікно має свій життєвий цикл і свої особливості. При створенні нового вікна викликається метод onCreate (), при розробці даний метод переопределяется і в ньому відбувається ініціалізація програми та його компонентів. Далі викликаються методи onStart () і onResume (). Обидва методи викликаються перед відображенням вікна при його створенні, або відновленні (при перемиканні з іншої програми, при розгортанні згорнутого додатки і тп). При згортанні викликаються методи onPause () і onStop (). При закритті програми і вікна викликається onDestory (), в даному методі можна зберегти призначені для користувача дані і параметри.

2.1.2 Фреймворк Xamarin

Xamarin - це фреймворк для кроссплатформенної розробки мобільних додатків (iOS, Android, Windows Phone) з використанням мови C#.

Фреймворк складається з декількох основних частин:

– Xamarin.IOS - бібліотека класів для C#, що надає розробнику доступ до iOS SDK;

– Xamarin.Android - бібліотека класів для C#, що надає розробнику доступ до Android SDK; Компілятори для iOS і Android;

– IDE Xamarin Studio;

– Плагін для Visual Studio.

Xamarin заснований на open-source реалізації платформи .NET - Mono. Ця реалізація включає в себе власний компілятор C#, середу виконання, а так само основні .NET бібліотеки [19].

З точки зору виконання додатків між iOS і Android є одна ключова відмінність - спосіб їх попередньої компіляції. Як відомо, для виконання додатків в Android використовується віртуальна Java-машина Dalvik. Нативні додатки, які пишуться на Java, компілюються в якийсь проміжний байт-код, який інтерпретується Dalvik`ом в команди процесора в момент виконання програми (аналогічно тому, як працює CLR в .NET). Це так звана Just-in-time компіляція (компіляція на льоту). У iOS використовується інша модель компіляції - Ahead-of-Time (компіляція перед виконанням). Xamarin враховує цю різницю, надаючи окремі компілятори для кожної з цих платформ, які дозволяють на виході отримувати справжні, нативні додатки, які виконуються поза контекстом браузера і можуть використовувати всі апаратні і програмні ресурси платформи.

Для iOS ситуація проста - ніякої віртуальної машини немає і програмний код повинен бути просто заздалегідь скомпільованим в машинний. Для цієї мети використовується AOT компілятор Mono.

Для Android, при компіляції програми відбувається переклад коду на C# в проміжний байт-код, зрозумілий віртуальній машині Mono і сама ця віртуальна машина також додається. І Mono і Dalvik написані на Сі і працюють поверх ядра Linux. При запуску програми на Android обидві віртуальні машини починають працювати пліч-о-пліч і обмінюються даними через спеціальний механізм wrapper`ов.

Розробники Xamarin як середовище розробки пропонують використовувати або власну IDE - Xamarin Studio, або Visual.

Xamarin Studio - кроссплатформенна IDE, яка працює як на Mac OS X, так і на Windows [20]. На вигляд ця вона виглядає дуже простою, проте за зовнішньою простою ховається досить потужний інструмент, який включив в себе функції, звичні в Visual Studio і Resharper:

– підсвічування синтаксису;

– автодоповнення коду (включаючи можливість одночасного імпорту namespaces);

– зручний універсальний пошук за назвами файлів, типам, членам класів тощо;

– розвинені можливості навігації по проекту: Швидкий перехід до опису класу, перехід до базового класу, список місць використання класу і т.д.;

– різні механізми рефакторінга і швидка підказка (як alt + Enter в Resharper);

– досить розвинені механізми стеження, перегляду поточного значення змінної при наведенні, візуалізацію потоків і аналогу Immediate window в VS.

2.1.3 Мова програмування - C#

C# (Сі-Шарп) - об'єктно-орієнтована мова програмування для платформи .NET. Основним постулатом C# є вислів: "будь-яка сутність є об'єкт". Мова заснована на суворої компонентної архітектурі і реалізує передові механізми забезпечення безпеки коду. C# була створена спеціально для технології ASP.NET. У той же час на C# повністю написана і сама ASP.NET.[21].

C# - це повнофункціональна об'єктно-орієнтована мова, яка підтримує всі три «стовпи» об'єктно-орієнтованого програмування: інкапсуляцію, успадкування та поліморфізм. Вона має прекрасну підтримку компонентів, надійних і стійких завдяки використанню «збирання сміття», обробки виключень, безпеки типів.

Мова C# розроблялась "з нуля" і увібрала в себе багато корисних властивостей таких мов, як C ++, Java, Visual Basic, а також Pascal, Delphy і ін. При цьому необхідність забезпечення сумісності з попередніми версіями була відсутня, що дозволило мові C# уникнути багатьох негативних сторін своїх попередників.

Як і Java, C# розроблялась для Інтернет і приблизно 75 % його синтаксичних можливостей аналогічні мови програмування Java, його також називають «очищеної версією Java. 10 % подібні мови програмування C++, а 5 % запозичені з мови програмування Visual Basic. Обсяг нових концептуальних ідей в мові C # близько 10 % [22].

Виділення і об'єднання кращих ідей сучасних мов програмування робить мову C# не просто сумою їх достоїнств, а мовою програмування нового покоління.

2.1.4 Патерн MVVM

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку додатків від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми.

Даний патерн був представлений Джоном Госсманом (John Gossman) в 2005 році як модифікація шаблону Presentation Model і був спочатку націлений на розробку додатків в WPF. І хоча зараз цей патерн вийшов за межі WPF і застосовується в самих різних технологіях, в тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну.

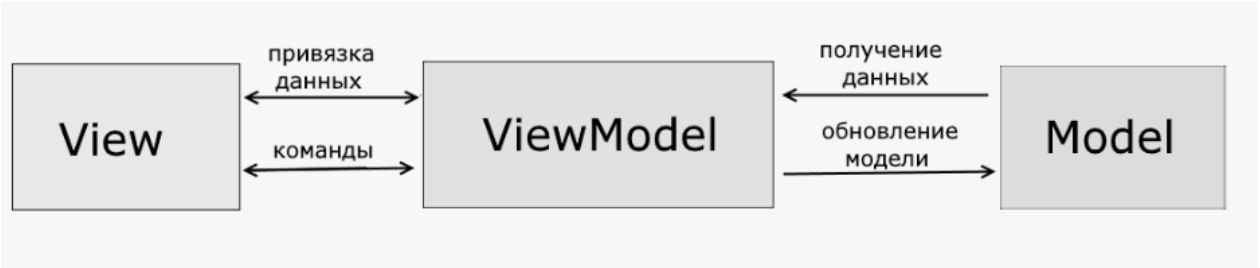

Рисунок 2.1 – Патерн MVVM в WPF

MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і уявлення (View) [23]:

1) Model:

Модель описує використовувані в додатку дані. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити ніякої логіки, пов'язаної з відображенням даних і взаємодією з візуальними елементами управління.

Нерідко модель реалізує інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, які дозволяють повідомляти систему про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до подання, хоча знову ж пряму взаємодію між моделлю і представленням відсутня.

2) View:

View або подання визначає візуальний інтерфейс, через який користувач взаємодіє з додатком. Стосовно до WPF уявлення - це код в xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) в WPF може містити як інтерфейс в xaml, так і прив'язаний до нього код C #, проте в ідеалі код C # не повинен містити якийсь логіки, крім хіба що конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна . Вся ж основна логіка додатки виноситься в компонент ViewModel.

Однак іноді в файлі пов'язаного коду все може знаходитися певна логіка, яку важко реалізувати в рамках паттерна MVVM у ViewModel.

Подання і не виконує жодних події за рідкісним винятком, а виконує дії в основному за допомогою команд [24].

3) ViewModel:

ViewModel або модель уявлення пов'язує модель і уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна відображуваних даних в поданні, хоча безпосередньо модель і уявлення не пов'язані.

ViewModel також містить логіку по отриманню даних з моделі, які потім передаються в уявлення. І також ViewModel визначає логіку по оновленню даних в моделі.

Оскільки елементи уявлення, тобто візуальні компоненти типу кнопок, не використовують події, то уявлення взаємодіє з ViewModel за допомогою команд.

Наприклад, користувач хоче зберегти введені в текстове поле дані. Він натискає на кнопку і тим самим відправляє команду під ViewModel. А ViewModel вже отримує передані дані і відповідно до них оновлює модель [25].

Підсумком застосування патерну MVVM є функціональний розподіл програми на три компонента, які простіше розробляти і тестувати, а також в подальшому модифікувати і підтримувати...

2.2 Аналіз Google додатків та сервісів, які будуть використані для створення програми

Сервіси Google сьогодні є популярним інструментом спілкування, обміну думками та отримання інформації. Останнім часом у світовій педагогічній спільноті обговорюються можливості використання мережних сервісів в освіті [26]. Так, за даними опитування, до сотні найкращих засобів, що застосовуються для створення і оприлюднення матеріалів навчального призначення або у якості інструментів для особистісного та професійного навчання, увійшли популярні базові сервіси Google, зокрема Диск (Документи) Google, YouTube, Веб-пошук Google та Google+ (відповідно 2, 3, 4 та 10 місце у рейтингу). Виділяють наступні переваги використання сервісів Google перед іншими видами мережних технологій [27]:

– висока функціональність та надійність, зручний україномовний інтерфейс. Способи комунікації та оприлюднення інформації в цьому середовищі вивчені більшістю користувачів. Цьому сприяє як зручність та зрозумілість системи, так і активний та тривалий досвід її використання.

Етап адаптації студентів до нового комунікативного простору значно скорочується;

– індивідуальний доступ до ресурсів і сервісів, значний обсяг дискового (хмарного) простору, який надається користувачеві;

– можливість формування груп і підрозділів користувачів. Форуми, опитування, голосування, коментарі, підписки, відправлення персональних повідомлень забезпечують широкі можливості для спільної роботи. Окрім формування навичок співпраці, це стимулює самостійну пізнавальну діяльність, прискорює отримання конкретного інтелектуального або творчого результату, розвиває критичність мислення студентів і дозволяє викладачу спостерігати за роботою студентів та координувати її;

– інтеграція з іншими програмними засобами вищого навчального закладу, можливість використання з мобільних пристроїв. Серед переваг навчання за допомогою сервісів Google студенти вищих навчальних закладів відзначають інтерактивність і безперервність навчального процесу, можливість виконання завдання в зручний для себе час і в зручному місці. Можливості використання базових сервісів Google у навчальному процесі з фізики показані на схемі, складеній нами за [28].

Одним з важливих елементів підготовки до занять є робота з навчальним матеріалом. Сучасні мережні сервіси надають багато можливостей для пошуку (Веб-пошук Google, Книги Google, Академія Google, YouTube) та зберігання файлів (хмарне сховище Диск Google). У цьому випадку файли (підручники, методичні посібники, навчальне відео) зберігаються у виділеному сховищі на сервері, а студенти отримують до них доступ і мають можливість працювати з ними через Інтернет.

Адекватно організована навчальна діяльність у віртуальному освітньому просторі є неможливою без самоактивності та відповідальності студента, інакше кажучи, у даному випадку йдеться про інтелектуальний саморозвиток як прямий продукт такої діяльності [29]. У якості проблемних

моментів при використанні сервісів Google у навчальному процесі слід відмітити [30]:

– значні зусилля та витрати часу, яких вимагають від викладача організація та підтримка навчального процесу в умовах безперервного навчання;

– велика кількість факторів, присутніх у соціальній мережі Google+ та відеохостингу YouTube, відволікають студентів від навчальної діяльності (активна комунікація, стрімкий інформаційний потік, розважальне наповнення).

Для реалізації мобільного додатку відстеження успішності студента будуть використовуватись наступні Google-сервіси:

– Google Cloud Messaging (зазвичай називають GCM) являє собою мобільний сервіс, розроблений Google, що дозволяє розробникам сторонніх додатків для передачі даних повідомлень або інформації від розробників запустити сервери для додатків, орієнтованих на Google Android операційну систему, а також додатки та розширення, розроблених для Google Chrome інтернет-браузерів;

– Google Calendar і API даних Google. Централізуя відображення і управління подіями, користувачі Google Calendar можуть спільно використовувати і підтримувати дані подій на одному сайті, що усуває один з безлічі можливих джерел помилок процесу організації подій. Мої друзі можуть відвідувати інтерактивний календар, щоб бути в курсі майбутніх подій і дій, не плутаючись в застарілої інформації на розрізнених Web-сторінках. Мабуть, інтерактивні додатки планування є ідеальним рішенням [33];

– Google Диск - це файловий хостинг, створений і підтримуваний компанією Google. Його функції включають зберігання файлів в Інтернеті, загальний доступ до них і спільне редагування. До складу Google Диску входять Google Документи, Таблиці та Презентації - набір офісних додатків для спільної роботи над текстовими документами, електронними таблицями,

презентаціями, кресленнями, веб-формами та іншими файлами. Загальнодоступні документи на Диску індексуються пошуковими системами [34].

2.3 Висновки до розділу

У даному розділі обрано інструменти, завдяки яким буде розроблено мобільний додаток. В рамках КРБ було обрано архітектурне рішення, назване як «MVVM-паттерн», що дозволяє відділити бізнес-логіку програми від її вигляду, краще розробити інтерфейс програми та підвищує пітрмику програми; розглянути різні мобільні платформи, що популярні зараз на ринку мобільних пристроїв, коротко описано їх основні дані, представлено основні їх моменти, переваги та недоліки кожного з них. В результаті було обрано мобільну систему Android, як основну мобільну платформу для розроблюваної системи та iOS, як альтернативну;

Також проаналізовано Google-сервіси, у якості помічника у навчанні студента та обрано декілька сервісів, які будуть використані у роботі мобільного додатка успішності студента.

3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1 Розробка архітектури додатку

На даний момент існує декілька типів архітектури розробки програмного забезпечення. В залежності від основного завдання, яке повинно виконувати ПЗ, потрібно вибрати відповідну архітектуру [35]. При виборі потрібної архітектури може залежати час розробки програмного забезпечення, вартість, тривалість та в подальшому можливість імплементувати чи модифікувати певні частини програмного забезпечення.

Додаток буде розроблено на основі архітектури «клієнт-сервер». Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Як сервери так і клієнти функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого.

Розглянемо детальніше схему роботи розроблюваного додатку. Спочатку потрібно зареєструватися в системі вказавши номер залікової книжки, електронну пошту, пароль та підтвердження паролю. Студенту ж непотрібно робити ніяких додаткових кроків щоб побачити свою успішність, достатньо просто відкрити додаток, ввести логін та пароль та натиснути клавішу «Вхід». Для того, щоб розпочати роботу з додатком, користувач повинен авторизуватися (на рис. 3.1 наведено діаграму станів процесу «Авторизація»).

Рисунок 3.1 – Діаграма станів процесу «Авторизація»

Після успішної авторизації система, якщо користувач ідентифікований виведе на екран новини сайту. Якщо потрібно побачити поточну інформацію успішності студента, необхідно у пункті меню обрати вкладку «Результати». Виведе на екран таблицю його успішності (оцінки за роботи, іспити та заліки, кожного з навчальних предметів). На рис. 3.2 зображено діаграма станів процесу «Перегляд успішності».

Рисунок 3.2 – Діаграма станів процесу «Перегляд успішності»

Після цього студент може вибрати вкладку «група» та натиснути на конверт, йому буде запропонована форма через, яку можна відправити повідомлення студентам, викладачам та ін. Достатньо просто відзначити адресатів напроти їхніх імен, та обрати спосіб відправки листів.

3.2 Проектування структури бази даних.

Під час розробки додатка необхідним етапом є проектування бази даних, яка забезпечить зберігання даних і надасть змогу проводити операції над ними. На цьому етапі потрібно визначити, які дані будуть зберігатися та які операції необхідно проводити над ними.

Для створення мобільного додатку були використані наступні таблиці:

– таблиця Course (рис. 3.3) зберігає в собі інформацію, щодо предмета та ПІБ викладача, а також яка форма контроля буде у саме цього предмета;

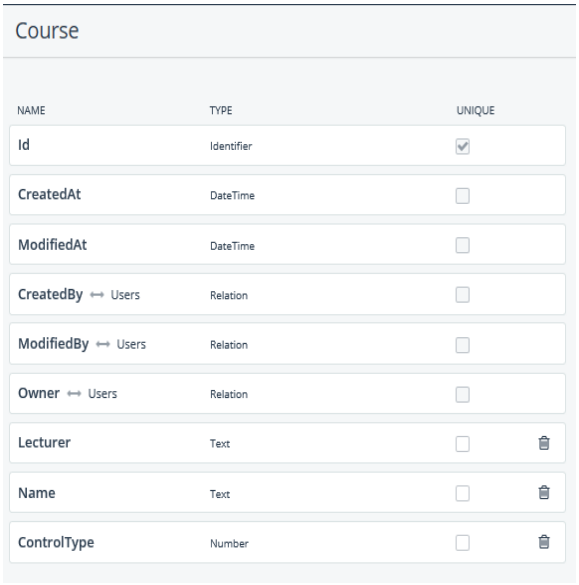

Рисунок 3.3 – Таблиця Course

– таблиця CourseAbsentStudents (рис. 3.4) зберігає в собі інформацію, щодо відсутності студента під час заняття, а саме назву ПІБ студета, начальну дисципліну день та час;

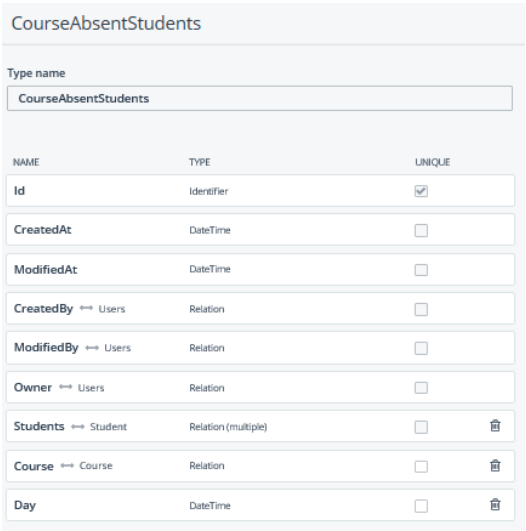

Рисунок 3.5 – Таблиця CourseAbsentStudents

– таблиця ExamsTimetable (рис. 3.6) зберігає в собі інформацію, щодо розкладу екзаменів в групі, а саме назву предмета, корпус та аудиторію, в якій відбудеться екзамен та дата проведення;

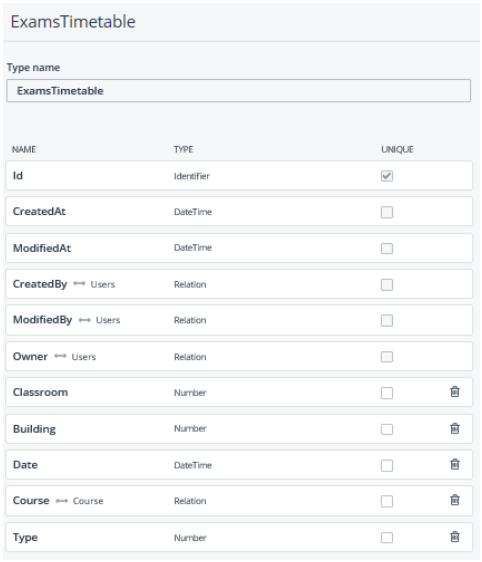

Рисунок 3.6 – Таблиця ExamsTimetable

– таблиця Group (рис. 3.7) зберігає в собі інформацію, щодо групи, а саме спеціальність, рік вступу до ВНЗ, ПІБ студента, який навчається, а також назва цієї групи;

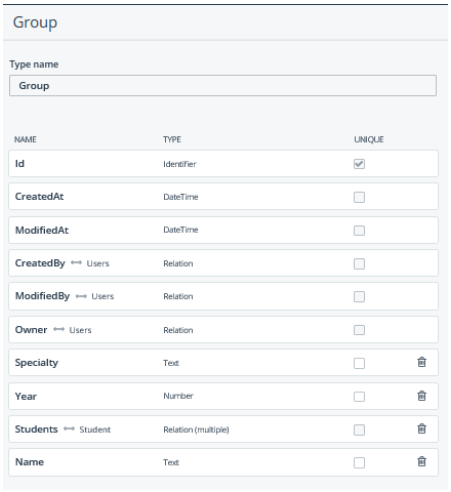

Рисунок 3.7 – Таблиця Group

– таблиця Student (рис. 3.8) зберігає в собі інформацію, щодо студента, а саме прізвище та ім’я та номер залікової книжки;

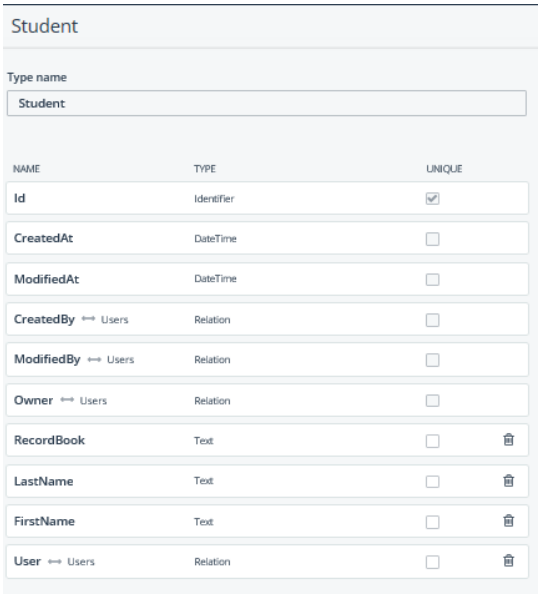

Рисунок 3.8 – Таблиця Student

– таблиця StudentControlResult (рис. 3.9) зберігає в собі інформацію, щодо студента та предмета, який він здав та оцінка, яку він отримав по цьому предмету;

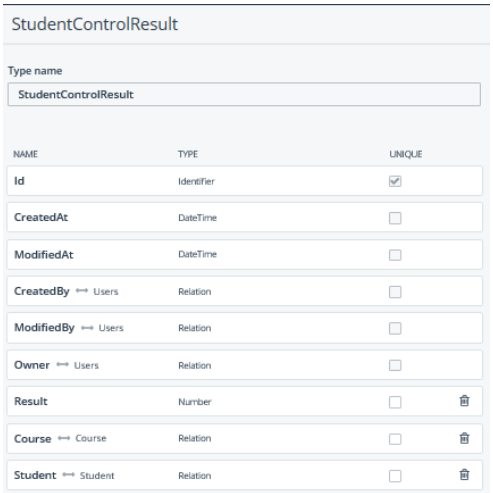

Рисунок 3.9 – Таблиця StudentControlResult

– таблиця StudentTaskResult (рис. 3.10) зберігає в собі інформацію, щодо студента та робіт, які необхідно здати під час начального семестру та результат, отриманий студентом за цю роботу;

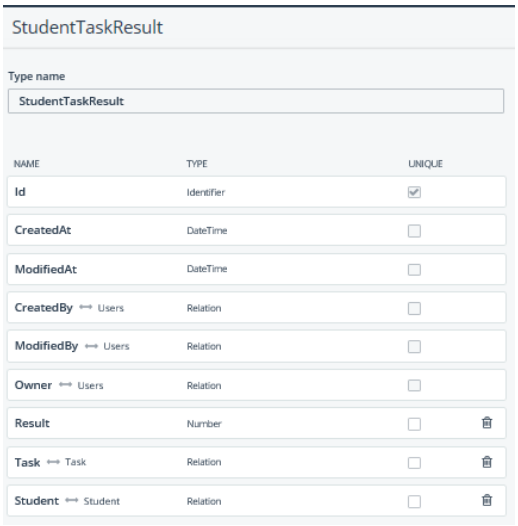

Рисунок 3.10 – Таблиця StudentTaskResult

– таблиця Task (рис. 3.11) зберігає в собі інформацію, щодо робіт, які необхідно здати під час начального семестру, а саме назву роботи та предмета, тип роботи, а також файли, які додаються до цього завдання;

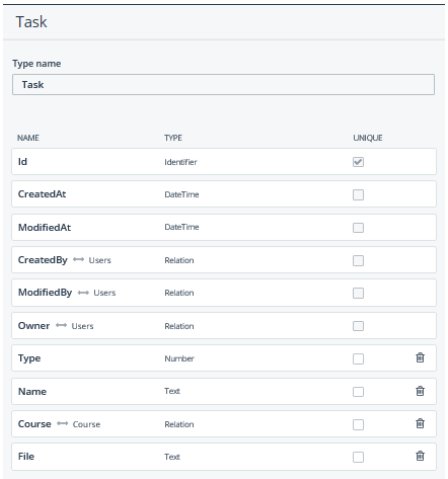

Рисунок 3.11 – Таблиця Task

– таблиця TimeTable (рис. 3.12) зберігає в собі інформацію, щодо розкладу занять в групі, а саме назву предмета, який тип пари (лекція або практичне заняття), корпус та аудиторію, в якій відбудеться пара, день тижня, вид тижня (верхній або нижній тиждень).

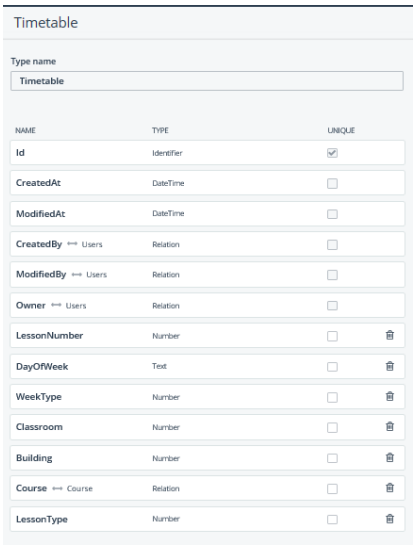

Рисунок 3.12 – Таблиця TimeTable

3.3 Програмна реалізація

Для того, щоб розпочати роботу в мобільному додатку, необхідно буде зареєструватися. Основою при реєстрації є номер залікової книжки студента який знаходиться у БД ВНЗ. Також для реєстрації необхідно заповнити всі поля, а саме номер залікової книжки, email, пароль та підтвердження пароля. Якщо ніяких помилок не має то користувач зареєстрован. Але існують перевірки, якщо введеного номера залікової книжки не має в базі даних, то буде помилка - ("Студент не найден"). Також при некорректно введених даних відображається помилка - ("Помилка реєстрації"):

Авторизація може відбуватися за допомогою Google-акаунта або завдяки email та паролю. Також існує перевірка на коректність введених даних або наявність пустих полей. У разі цього відкриваються діалогові вікна

з надписами "Помилка входу" та "Заповніть всі поля". Якщо користувача не має в базі з’являється помилка - "Користувач не знайден".

На сторінці з налаштуванням розташовано три кнопки, а саме: включення розсилки електронних листів на пошту, додавання до календаря дат іспитів з нагадуванням про них, а також кнопка включення повідомлень, які відображаються на іконці мобільного додатка та в панелі повідомлень.

Лістінг мобільного додатку можна подивитись у додатку Б.

3.4 Висновки до розділу

У цьому розділі розроблено архітектуру мобільного додатка та сформовані діаграми станів процесів. Додаток буде розроблено на основі архітектури «клієнт-сервер», оскільки вона є найпоширенішою концепцією у створенні розподілених мережевих застосунків і передбачає взаємодію та обмін даними між ними. Спроектовано структуру бази даних, яка дозволяє розробнику переглянути, які дані мають зберігатися в базі даних і, як вони повинні взаємодіяти між собою. Наведено приклади реєстрації, авторизації та меню «Налаштування».

4 ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1 Огляд мобільного додатка відстеження успішості студента

Для початку роботи з додатком потрібно завантажити файл формату .apk, та просто відкрити його на своєму смартфоні або планшеті, після чого Також невід'ємною умовою функціонування додатку є наявність підключення до мережі Інтернет. Для того щоб студент міг бачити всю необхідну інформацію йому потрібно бути зареєстрованим в системі. Скріншот вікна «Авторизації» в системі представлено на рис. 4.1.

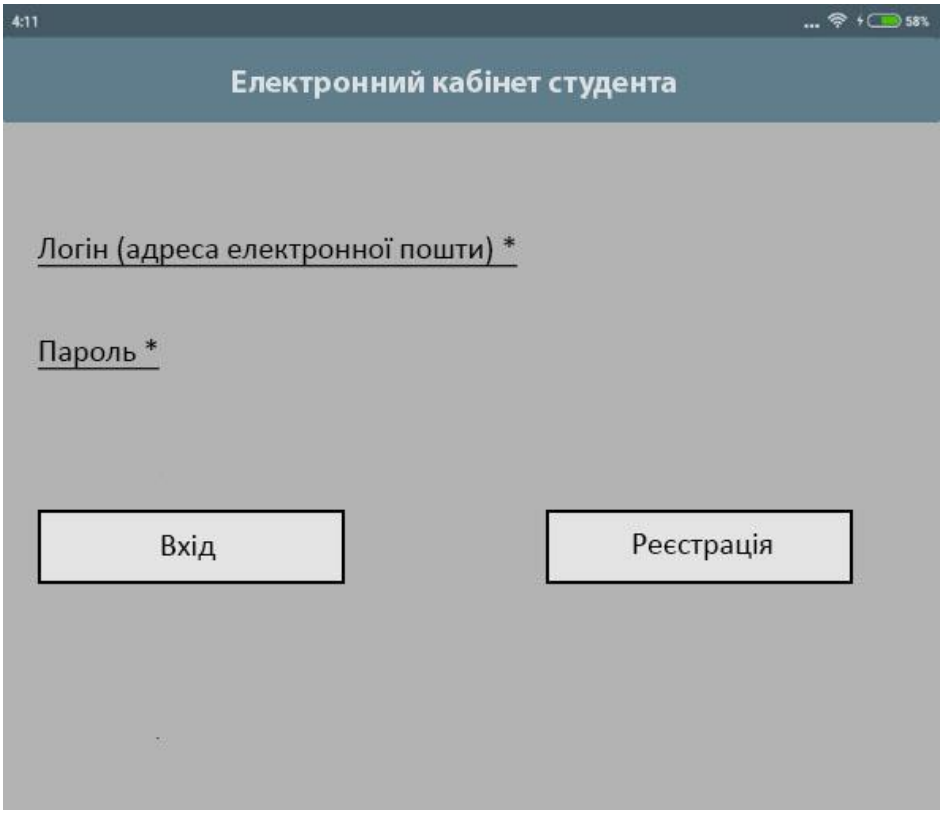

Рисунок 4.1 – Вікно «Авторизації»

Студенту ж непотрібно робити ніяких додаткових кроків щоб побачити свою успішність, достатньо просто відкрити додаток, ввести email та пароль, якщо він вже зареєстрован у додатку, потім натиснути кнопку

«Вхід». Якщо користувач ще не зареєстрован, то потрібно натиснути кнопку «Реєстрація». Скріншот вікна «Реєстрація» в системі представлено на рис. 4.2.

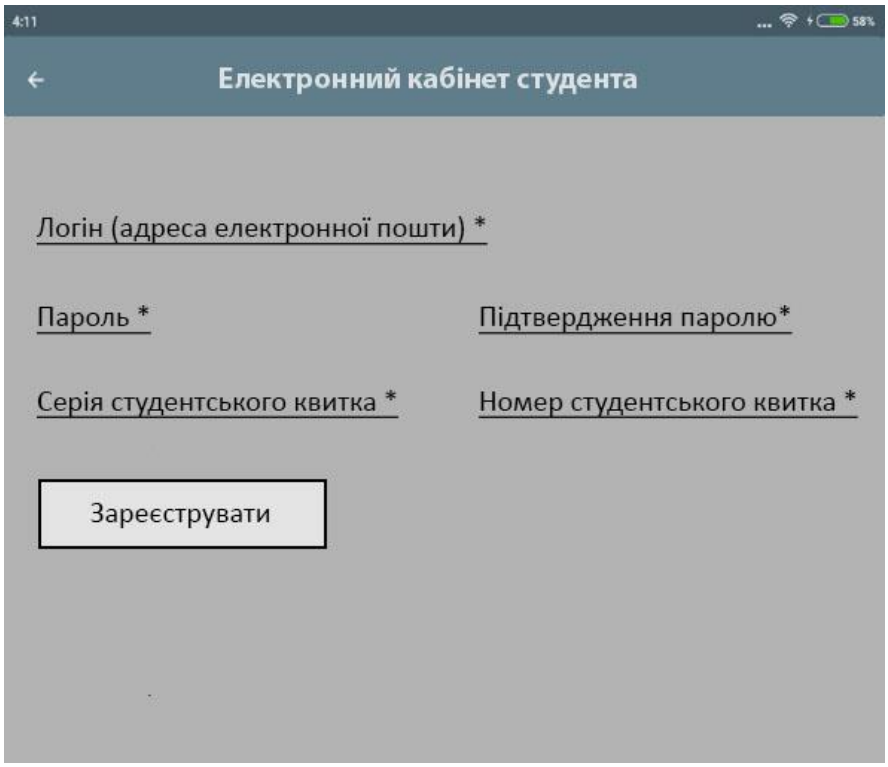

Рисунок 4.2 – Вікно «Реєстрація»

Для реєстрації необхідно заповнити всі поля, а саме номер залікової книжки, завдяки цієї інформації в програмі буде ідентифіковано який саме це студент зареєструвався. Далі вводиться email, пароль та підтвердження пароля. Потім потрібно натиснути кнопку «Зареєструвати». Також існує перевірка на заповненість всіх полей та на корректність введених даних. Якщо введений номер залікової книжки відсутній в базі, то буде виведено вікно повідомлення «Студента не знайдено».

Після входу до електронного кабінету користувач потрапляє до головного меню де він може вибрату вкладку яка його цікавить (рис. 4.3).

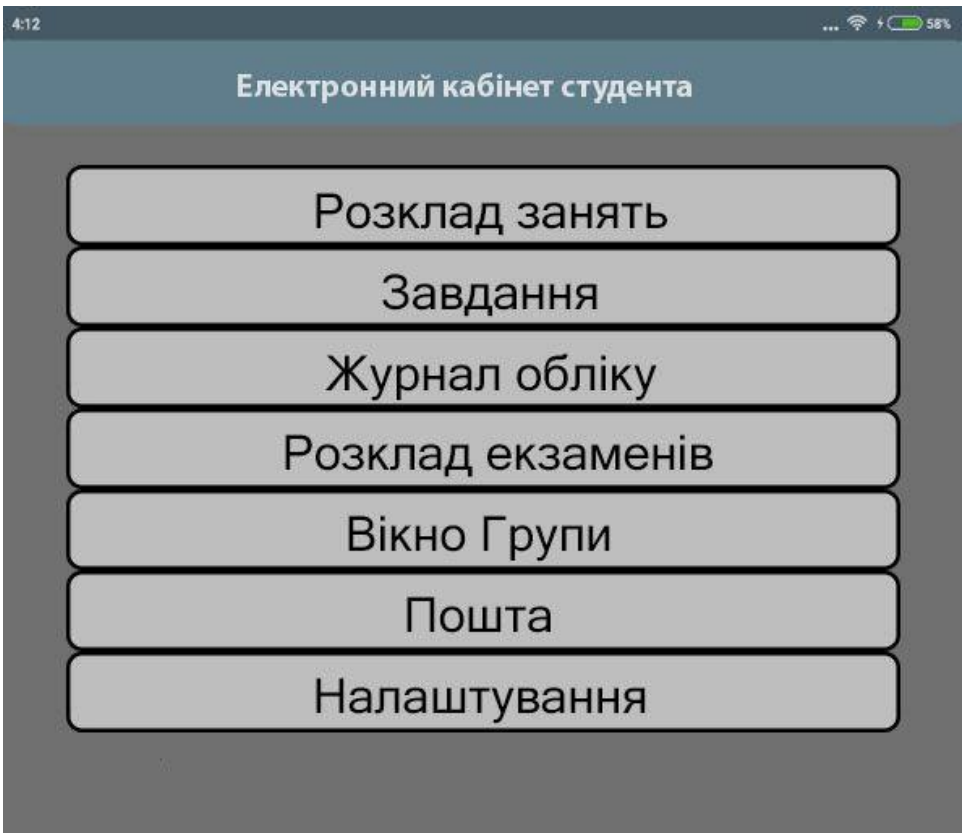

Рисунок 4.3 – Вікно «Меню мобільного додатка»

Перша вкладка - «Розклад занять». Розклад занять групи відображен у виді сторінок по дням тижня, листаючі вправо чи вліво можна подивитися розклад на весь тиждень. Розклад містить в собі інформацію, щодо навчальних дисциплін, місце, час та вид заняття (лекція або практичне заняття).

В свою чергу тижні поділяються на верхній та нижній, студент може обрати необхідний тиждень за допомогою кнопок розташованих зверху (рис. 4.4).

Третьою вкладкою є «Завдання». Після переходу, можна побачити завдання, які є по навчальній дисципліні на цей семестр, це можуть бути лабораторні работи, индивідуальні завдання та ін. Також можна завантажити необхідні матеріали, які додаються до роботи (рис. 4.5).

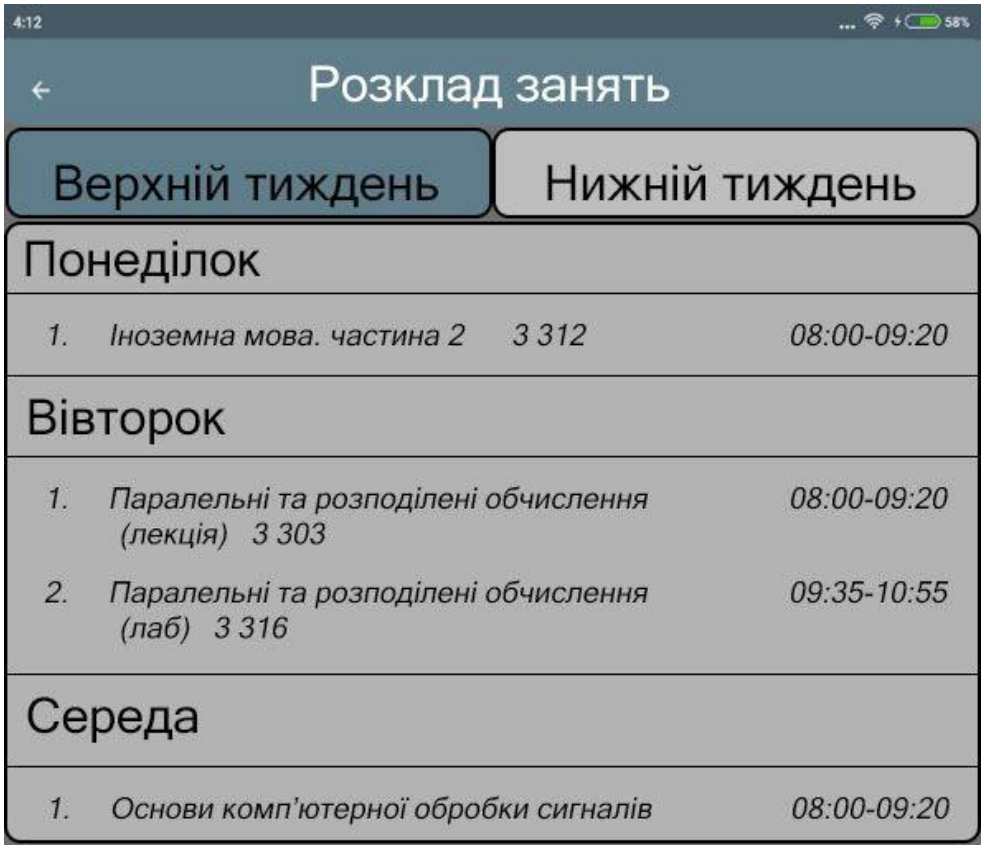

Рисунок 4.4 – Вікно «Розклад занять»

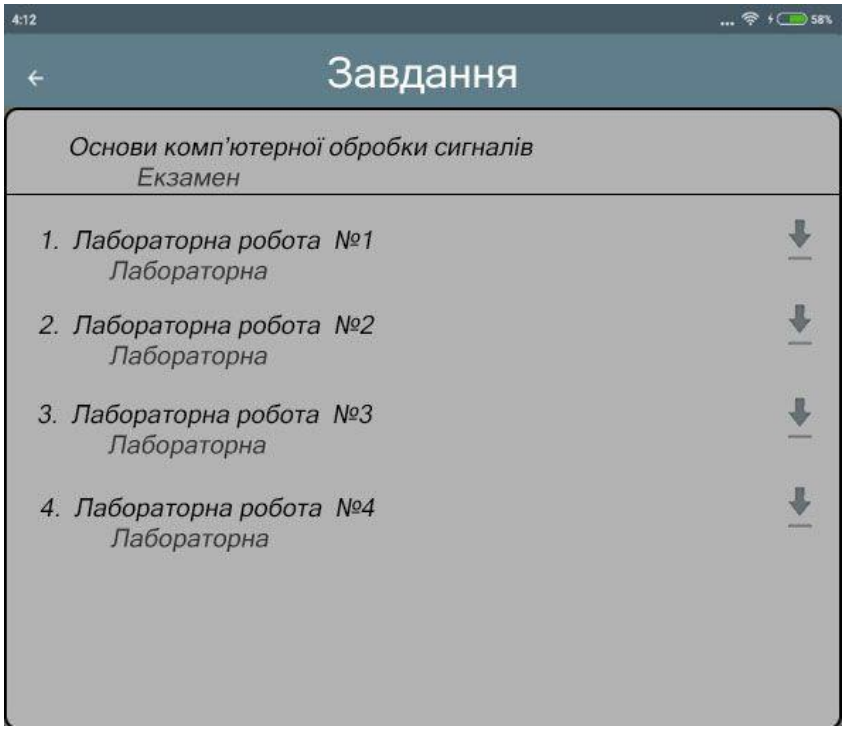

Рисунок 4.5 – Вікно «Завдання»

Кількість пропусків можна побачити у «Журналі обліку». На сторінці виводиться інформація, щодо начальної дисципліни та дату, коли були пропуски занять. Листаючі вліво чи враво, можна обрати необхідний предмет (рис. 4.6).

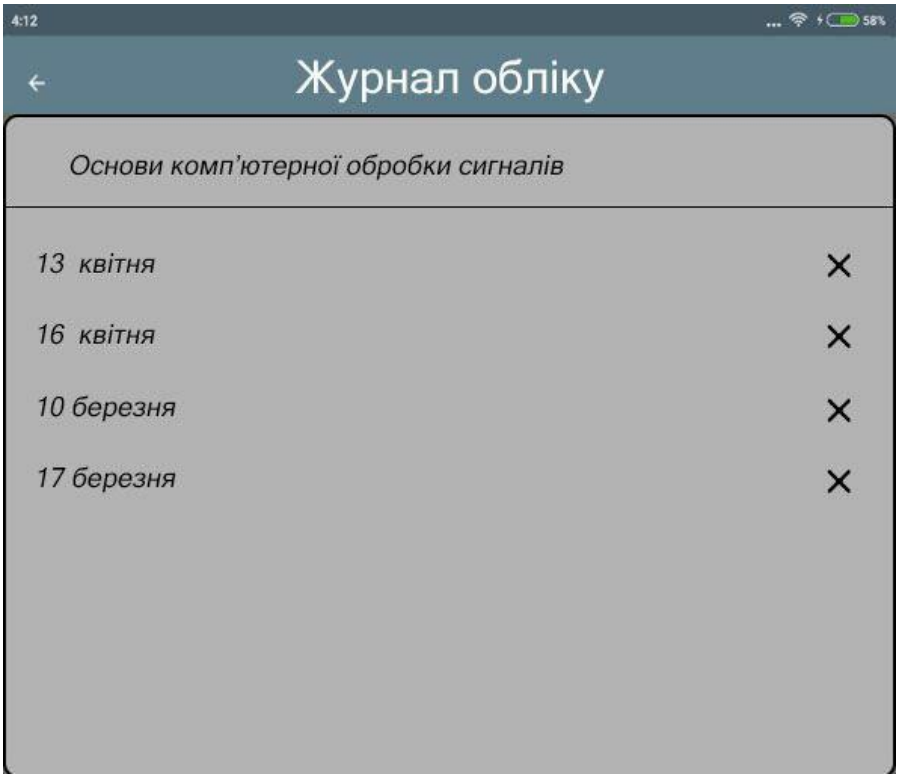

Рисунок 4.6 – Вікно «Журнал обліку»

«Розклад екзаменів» містить дані щодо дати, часу та предмета, з якого відбудеться іспит (рис. 4.7).

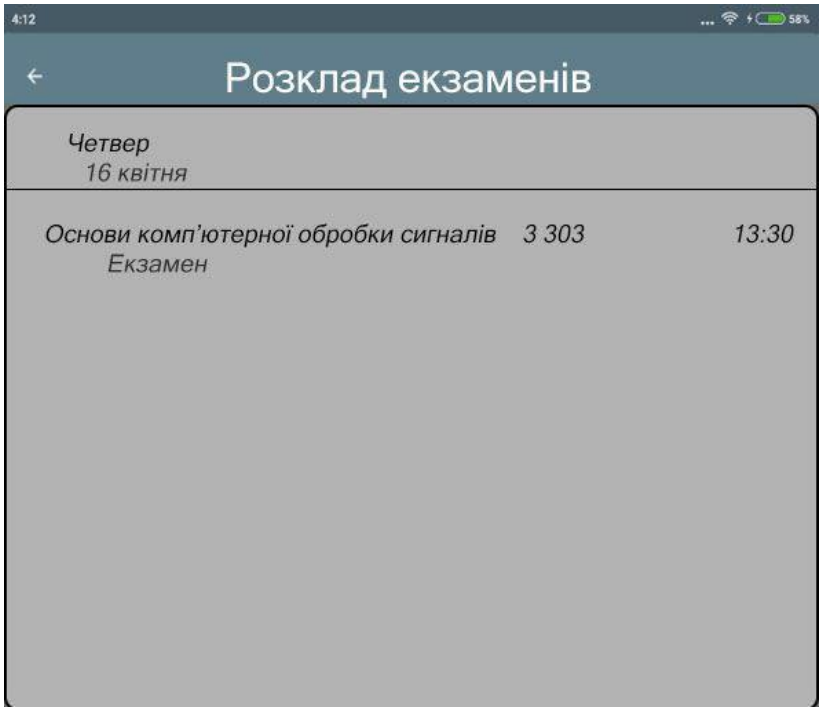

Рисунок 4.7 – Вікно «Розклад екзаменів»

Шоста вкладка – «Група». На сторінці відображено список групи, в якій навчається студент,викладачі які викладають у поточному семестрі та контакт деканату (рис. 4.8).

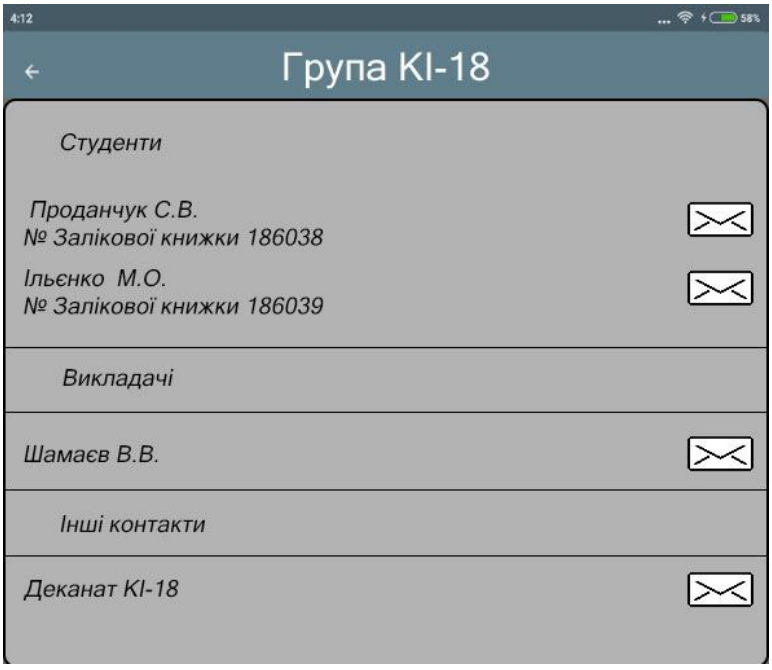

Рисунок 4.8 – Вікно «Група»

Є можливість відправити листа студенту,викладачеві або деканату. Після натискання на іконку конверта, користувача відсилає до вікна пошти і автоматично обирає адресанта на якого користувач натиснув у вікні групи.

У вікні пошта користувач має можливість самостійно обрати з якої адреси і кому він хоче надіслати листа. Зробити це можна заповнивши поля адресата і адресанта. Заповнивши поля Тема і Текст повідомлення, також користувач має змогу прикріпити до листа файл натиснувши на іконку скріпки, після цього нажати на іконку стрілки для відправки (рис. 4.9).

Рисунок 4.9 – Вікно «Пошта»

В останній вкладці «Налаштування» є можливість включити розсилку поштових листів, додати повідомлення, яке нагадує про іспит або залік (рис. 4.10).

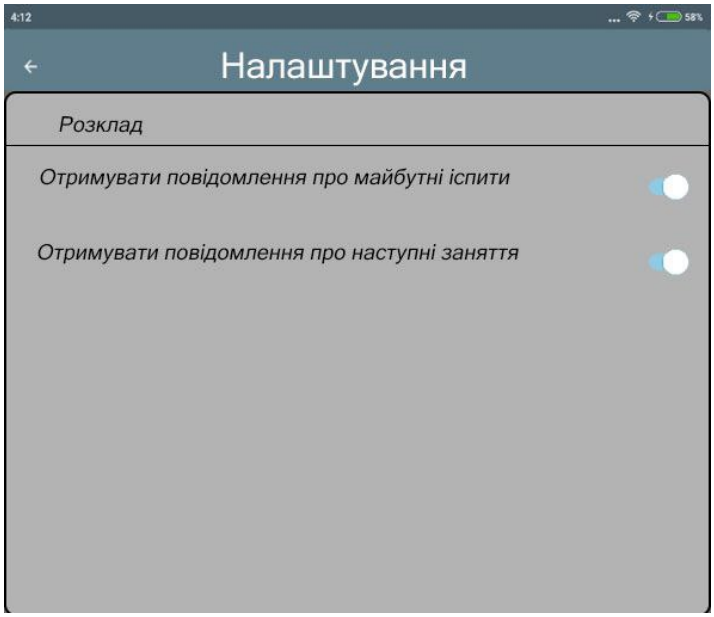

Рисунок 4.10 – Вікно «Налаштування»

Було проведено такі види тестування системи:

1) інсталяційне тестування. Під час проведення даного тестування додаток було успішно встановлено на смартфони з такими операційними системами:

– Android 4.4.4 (Kit Kat);
– Android 5.0 (Lollipop);
– Android 5.1 (Lollipop);
– Android 6.0 (Marshmallow).

2) тестування продуктивності. Під час проведення даного тестування смартфон продовжував успішно працювати з додатком при таких умовах:

– низькому рівні заряду акумулятора;
– поганому покритті мережею;
– низькій кількості доступної пам'яті;
– одночасному доступі до сервера кількома користувачами.

3) навантажувальне тестування:

– на смартфонах запускалося декілька сторонніх додатків які використовували підключення до мережі інтернет.

4) тестування перериванням:

– отримання повідомлень під час роботи з додатком;

– відповідь на дзвінок під час роботи з додатком;

– підключення та відключення USB кабелю;

– проведено тестування при різних типах підключення до мережі інтернет (4g, 3g, WiFi), яке успішно завершилося у всіх трьох випадках.

Всі тести додаток пройшов успішно, додаток готовий для використання та експлуатації, фатальних багів виявлено не було.

4.2 Можливі варіанти розвитку програмного додатка

Можливими шляхами розвитку додатки є:

– розробка додатка на інших операційних системах;

– використовування інших Google-сервісів;

– створення бази для всього університету;

– інтеграція додатка зі сторонніми базами даних;

– реалізація двухкомпонентной бази даних (локальної бази даних і бази даних в хмарі);

– реалізація інтерфейсу різними мовами.

4.3 Висновки до розділу

Проведено тестування програми на наявність помилок або дефектів, яке показало, що додаток повністю придатний для користування. Створено огляд інтерфейсу програми, у якій детально описано які кроки потрібно зробити щоб використовувати додаток на повну. Також проведено аналіз ефективності використання мобільного додатку студентами ВНЗ, який показав, що додаток позитивно впливає на якість навчання.

ВИСНОВКИ

Під час створення мобільного додатку у роботі було виконано:

1) досліджено предметну область. Проведено огляд та аналіз кількох веб-систем аналогів, що реалізують схожі функції предметної області.

2) було обрано необхідні інструменти, вибір яких базувався на більш універсальному методі створення кода та проаналізовано google-сервіси, які використовуються у мобільному додатку відстеження успішності студента.

3) реалізовано функції додатка, які написані на мові програмування c# з імплементацією запитів до серверу, також створено структуру бази даних.

4) проведено аналіз ефективності використання мобільного додатку студентами ЗВО. Проведено тестування програми на наявність помилок або дефектів, яке показало, що додаток повністю придатний для користування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Коваль, Т.І. Інтерактивні технології навчання іноземних мов у вищих навчальних закладах [Електронний ресурс] / Т.І. Коваль // Інформаційні технології і засоби навчання. — 2011. — № 6(26). - Режим доступу: http://journal.iitta.gov.ua/index.php/itlt/article/view/546/451
2. What is m-learning? [Електронний ресурс] // Tribal's Digital Learning Studio. Cambridge,United Kingdom. - Режим доступу: http://www.m-learning.org/knowledgecentre/whatismlearning
3. Mobile Assisted Language Learning (MALL) [Електронний ресурс] // Wikipedia. The FreeEncyclopedia. Wikimedia Foundation, Inc. - Режим доступу: https://en.wikipedia.org/wiki/Mobile_Assisted_Language
4. Інформаційні технології навчання [Електронний ресурс] - Режим доступу: http://npu.edu.ua/!e-book/book/html/D/ispu_kiovist_Ficyla_Pedagogika_VSh/770.html
5. Вірченко П., Геллер О. Запровадження інформаційних та мультимедійних технологій навчання // Зб. наук. праць. – К.: Обрії, 2001. – Вип. 4. – С. 347–349.
6. Жалдак М. І., Морзе Н. В., Олійник А. Г., Рамський В. С. Вплив нової інформаційної технології на зміст освіти // Сучасна інформаційна технологія в навчальному процесі: Зб. наук. праць. Наукові записки. Серія: Педагогіка. — 2009. — № 3.
7. Сучасний урок. Інтерактивні технології навчання: Наук.-метод. посібник / За ред. О. І. Пометун. – К., 2004. – 192 с.
8. Фіцула М .М. Педагогіка вищої школи: Навч. посібник. – К.: Академвидав, 2006. – 352 с.
9. Зинченко В. П. Аффект и интеллект в образовании. – М.: Тривола, 2005. – 64 с.
10. Тесленко І. Ф. Формування комп’ютерної грамотності учнів // Збірник статей. – К.: Радянська школа, 2007. – 157 с.

11. Щоденник. Всеукраїнська безкоштовна шкільна освітня мережа [Електронний ресурс] - Режим доступу: http://shodennik.ua
12. Rating system - online [Електронний ресурс] - Режим доступу: http://mod.tanet.edu.te.ua
13. FCIT TEACH Розклади, Журнали, ЕНМКД [Електронний ресурс] - Режим доступу: http://teach.tneu.org/
14. Attendance Tracking Online: MyAttendanceTracker [Електронний ресурс] - Режим доступу: https://www.myattendancetracker.com/
15. RevolverLab. Детальное сравнение iOS, Android и Windows Phone [Електронний ресурс] - Режим доступу: https://revolverlab.com/детальное-сравнение-ios-android-и-windows-phone-22f7cc17e6ba
16. Алешин, Л.И. Информационные технологии: Учебное пособие / Л.И. Алешин. - М.: Маркет ДС, 2011. - 384 с.
17. Reto Meier Professional Android 4 Application Development /R.Meier, Wrox, 2012. – 864 p.
18. Android [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Android 9
19. Хабрахабр. Особливості розробки під Xamarin.Forms. [Електронний ресурс]. - Режим доступу: https://habrahabr.ru/company/devexpress/blog/263645/
20. Хабрахабр. Детально про Xamarin. [Електронний ресурс]. - Режим доступу: https://habrahabr.ru/post/188130/
21. C Sharp. [Електронний ресурс]. - Режим доступу: http://lurkmore.to/C_Sharp; 14.
22. CyberForum. Плюси та мінуси C#. [Електронний ресурс]. - Режим доступу: http://www.cyberforum.ru/csharp-net/thread442516.html;
23. Model-View-ViewModel [Електронний ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/Model-View-ViewModel

24. Паттерны: MVC, MVP и MVVM [Електронний ресурс]. – Режим доступу: http://outcoldman.com/ru/ archive/2010/02/22/паттерны-mvc-mvp-и-mvvm.
25. Создание MVVM-приложений с помощью Xamarin и MvvmCross [Електронний ресурс]. – Режим доступу: https://msdn.microsoft.com/ru-ru/magazine/dn 759 442.aspx
26. Патаракин Е. Д. Социальные взаимодействия и сетевое обучение 2.0 / Патаракин Е. Д. – М. : Современные технологии в образовании и культуре, 2009. – 176 с.
27. Олексюк В. П. Досвід інтеграції хмарних сервісів Google APPS у інформаційно-освітній простір вищого навчального закладу [Електронний ре- сурс] / В. П. Олексюк // Інформаційні технології і засоби навчання. – 2013. – Т. 35, вип. 3. – С. 64-73. – Режим доступу : http://nbuv.gov.ua/jpdf/ITZN_2013_35_3_9.pdf
28. Патаракин Е. Д. Социальные взаимодействия и сетевое обучение 2.0 / Патаракин Е. Д. – М. : Современные технологии в образовании и культуре, 2009. – 176 с.
29. Смульсон М. Л. Інтелектуальній розвиток як мета дистанційного навчання [Електронний ресурс] / М. Л. Смульсон. // Технології розвитку інтелекту. – 2011. – № 2. – Режим доступу : http://www.nbuv.gov.ua/ujrn/e-journals/tri/2011_2/st07.pdf
30. Єчкало Ю. В. Модель персонального навчального середовища / Ю. В. Єчкало // Новітні комп'ютерні технології. – Кривий Ріг : ДВНЗ «Криворізький національний університет», 2013. – Випуск XI. – С. 51
31. Google Cloud Messaging [Електронний ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/Google_Cloud_Messaging
32. Персональні сервіси Google. Gmail (теорія) [Електронний ресурс] - Режим доступу: http://www.eduwiki.uran.net.ua/wiki/index.php?title=Міні-курс_Персональні_сервіси_Google._Gmail_(теорія)

33. Google API [Електронний ресурс] - Режим доступу: https://habrahabr.ru/hub/google_api/
34. Використання сервісів Google+ у навчальному процесі [Електронний ресурс] - Режим доступу: http://rekachuk18.blogspot.com/2016/12/4.html
35. Поняття архітектури програмного забезпечення. [Електронний ресурс]. — Режим доступу: http://www.lib.mdpu.org.ua/e-book/web/Lec10.html
36. Сравнение многоуровневых и клиент-серверных систем [Електронний ресурс] - Режим доступу: http://msdn.microsoft.com/ru-ru/library/ee895340.aspx
37. Distributed computing [Електронний ресурс] - Режим доступу: http://en.wikipedia.org/wiki/Distributed_computing
38. Modular programming [Електронний ресурс] - Режим доступу: http://en.wikipedia.org/wiki/Modular_programming

Додаток А – Заява на затвердження теми і керівника роботи

Завідувачу кафедри
комп’ютерної інженерії
Ковальову Сергію Олександровичу
студента 4 курсу, групи КІ-18,
спеціальності 123 «Комп`стерна
інженерія»
Ільєнка Миколи Олександровича

ЗАЯВА

Прошу призначити керівником моєї магістерської роботи к.т.н., доц., доц. кафедри комп’ютерної інженерії Шамаєва Віталія Віталієвича
(вчена ступінь, вчене звання, посада, П.І.Б викладача)

і закріпити за мною наступну тему: «Мобільний додаток “Електронний кабінет студента”»
(найменування теми)

«____» _____ 2022 р. _____
(підпис студента)

ПОГОДЖЕНО:

Науковий керівник _____ Шамаєв В. В.
(підпис) (прізвище та ініціали)

Додаток Б – Лістінг програми

LoginActivity.cs

```
using System;
using Android.App;
using Android.Content;
using Android.Database;
using Android.Gms.Auth.Api;
using Android.Gms.Auth.Api.SignIn;
using Android.Gms.Common.Apis;
using Android.OS;
using Android.Provider;
using Android.Widget;
using MvvmCross.Extensions.Core.Util;
using MvvmCross.Extensions.Droid.V7.App;
using StudyApp.Core.Mvx.Platform;
using StudyApp.Core.Mvx.ViewModels;
using StudyApp.Core.Service.Interface;

namespace StudyApp.Droid.Activities
{
    [Activity(MainLauncher = true)]
    public class LoginActivity : AppCompatActivity<LoginViewModel>
    {
        private const int RcSignIn = 9910;
        private GoogleApiClient _googleApiClient;
        protected override int ContentID =>
Resource.Layout.LaunchActivity;

        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            ViewModel.Email = "";
            ViewModel.Password = "";

            var gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DefaultSignIn)
```

```
                .RequestIdToken("894599004816-
qib2di7pj8ggrtmvbtgngmi31f4cmmme.apps.googleusercontent.com")
                .RequestEmail()
                .RequestProfile()
                .Build();

            _googleApiClient = new GoogleApiClient.Builder(this)
                .EnableAutoManage(this, obj => { Toast.MakeText(this,
"Помилка входу", ToastLength.Long); })
                .AddApi(Auth.GOOGLE_SIGN_IN_API, gso)
                .Build();

            FindViewById(Resource.Id.GoogleLoginButton).Click        +=
OnClick;
        }

        private void OnClick(object sender, EventArgs eventArgs)
        {
            var                      signInIntent                      =
Auth.GoogleSignInApi.GetSignInIntent(_googleApiClient);
            StartActivityForResult(signInIntent, RcSignIn);
        }

        protected override void OnActivityResult(int requestCode,
Result resultCode, Intent data)
        {
            if (requestCode != RcSignIn) return;
            var                          result                        =
Auth.GoogleSignInApi.GetSignInResultFromIntent(data);
            HandleSignInResult(result);
        }

        private void HandleSignInResult(GoogleSignInResult result)
        {
            if (!result.IsSuccess) return;
            var email = result.SignInAccount.Email;
            var calendarsUri = CalendarContract.Calendars.ContentUri;

            string[] calendarsProjection =
```

```
            {
                CalendarContract.Calendars.InterfaceConsts.Id,

CalendarContract.Calendars.InterfaceConsts.CalendarDisplayName,
                CalendarContract.Calendars.InterfaceConsts.AccountName
            };

            var loader = new CursorLoader(this, calendarsUri,
calendarsProjection, null, null, null);
            var cursor = (ICursor) loader.LoadInBackground();

            while (cursor.MoveToNext())
            {
                var id = cursor.GetInt(0);
                var displayName = cursor.GetString(1);
                var account = cursor.GetString(2);

                if (displayName == email && account == email)
                    IocUtility.Resolve<ICalendarManager>().ID = id;
            }
            ViewModel.GoogleLogin(result.SignInAccount.Id,
result.SignInAccount.DisplayName,
                result.SignInAccount.Email);
            if (result.SignInAccount.PhotoUrl != null)
                IocUtility.Resolve<IAuthenticationService>().ImageUrl
= result.SignInAccount.PhotoUrl.ToString();
        }
    }
}
```

NotificationsService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using StudyApp.Core.Service.Interface;
using UAirshipSender;
using UAirshipSender.Model;
using UrbanAirship.Portable;

namespace StudyApp.Core.Service
{
```

```
    public class NotificationsService : INotificationsService
    {
        private readonly Dictionary<string, ScheduledPush>
_notificationByName = new Dictionary<string, ScheduledPush>();

        public string NamedUser
        {
            get { return Airship.Instance.NamedUser; }
            set { Airship.Instance.NamedUser = value; }
        }

        public Task<IList<ScheduledPush>> GetSchedules() =>
UASender.GetSchedules();

        public async Task AddScheduledPush(ScheduledPush push)
        {
            if (_notificationByName.Count == 0)
            {
                foreach (var item in await UASender.GetSchedules())
                {
                    _notificationByName[item.Name] = item;
                }
            }
            if (_notificationByName.ContainsKey(push.Name)) return;
            var schedule = await UASender.CreateSchedule(push);
            if (schedule != null) _notificationByName[schedule.Name] =
schedule;
        }

        public async Task RemoveScheduledPush(ScheduledPush push)
        {
            if (_notificationByName.Count == 0)
            {
                foreach (var item in await UASender.GetSchedules())
                {
                    _notificationByName[item.Name] = item;
                }
            }
            await UASender.DeleteSchedule(push);
```

```
            _notificationByName.Remove(push.Name);
        }
    }
}
```

AbsenceViewModel.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Model.Entities;
using StudyApp.Core.Mvx.Items;
using StudyApp.Core.Service.Interface;

namespace StudyApp.Core.Mvx.ViewModels
{
    [NavigationItem(Title = "Журнал обліку", Description = "Журнал обліку", Icon = "ic_chrome_reader_mode_black_24dp")]
    public class AbsenceViewModel : CollectionViewModel<KeyValuePair<string, List<DayAbsence>>>
    {
        private readonly IDataService _dataService;
        private readonly IAuthenticationService _authenticationService;

        public AbsenceViewModel(IDataService dataService, IAuthenticationService authenticationService)
        {
            _dataService = dataService;
            _authenticationService = authenticationService;
        }

        protected override async Task<IEnumerable<KeyValuePair<string, List<DayAbsence>>>> OnLoad()
        {
            var items = await _dataService.Get<CourseAbsentStudents>();
            var result = new Dictionary<string, List<DayAbsence>>();
            foreach (var item in items)
```

```
            {
                var course = await
_dataService.Get<Course>(item.Course);
                if (!result.ContainsKey(course.Name))
result[course.Name] = new List<DayAbsence>();
                result[course.Name].Add(new DayAbsence
                {
                    Day = item.Day,
                    IsAbcent = item.Students != null &&
item.Students.Contains(_authenticationService.Profile.Id)
                });
            }
            return result;
        }
    }
}
```

ExamsTimetableViewModel.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MvvmCross.Core.ViewModels;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.Util;
using MvvmCross.Extensions.Core.ViewModels;
using MvvmCross.Extensions.Core.ViewModels.Parameters;
using StudyApp.Core.Extensions;
using StudyApp.Core.Model.Entities;
using StudyApp.Core.Mvx.Items;
using StudyApp.Core.Mvx.Platform;
using StudyApp.Core.Service.Interface;
using UAirshipSender;
using UAirshipSender.Model;
using UrbanAirship.Portable;
using static MvvmCross.Extensions.Core.Util.SerializationUtility;
using System;

namespace StudyApp.Core.Mvx.ViewModels
{
```

```
    [NavigationItem(Title = "Розклад екзаменів", Description =
"Розклад екзаменів",
        Icon = "ic_date_range_black_24dp")]
    public class ExamsTimetableViewModel :
CollectionViewModel<TimetableDay>
    {
        private readonly IDataService _dataService;
        private readonly INotificationsService _notificationsService;
        private bool _notificationsEnabled;
        private IList<ExamsTimetable> _timetable;

        public ExamsTimetableViewModel(IDataService dataService,
INotificationsService notificationsService)
        {
            _dataService = dataService;
            _notificationsService = notificationsService;
        }

        public bool NotificationsEnabled
        {
            get { return _notificationsEnabled; }
            set { SetProperty(ref _notificationsEnabled, value); }
        }

        protected override async Task<IEnumerable<TimetableDay>>
OnLoad()
        {
            _timetable = await _dataService.GetExamsTimetable();

            var sc = await _notificationsService.GetSchedules();
            var notifications = sc.Where(n =>
n.Name.Contains(Airship.Instance.ChannelId))
                .ToList();

            if (notifications != null) NotificationsEnabled =
notifications.Count != 0;

            return _timetable.GroupBy(i => i.Date.Date).Select(items
=> new TimetableDay
```

```
            {
                Day = items.Key,
                Items = items
                    .Select(i => new Exam
                    {
                        Building = i.Building,
                        Classroom = i.Classroom,
                        Course = i.Course.Name,
                        Lecturer = i.Course.Lecturer,
                        Start = i.Date.TimeOfDay,
                        Type = i.Type
                    }).ToList(),
                ItemSelected = new
MvxCommand<TimetableItem>(OnItemSelected)
            }).ToList();
        }

        protected virtual void OnItemSelected(TimetableItem item)
        {
            ShowViewModel<ExamViewModel>(new
ViewModelParameter(SerializeObject(item)));
        }

        public async void DisableNotifications()
        {
            var notifications =
                (await _notificationsService.GetSchedules())
                    .Where(n =>
n.Name.Contains(Airship.Instance.ChannelId))
                    .ToList();

            if (notifications != null)
                foreach (var push in notifications)
                    await
_notificationsService.RemoveScheduledPush(push);

            NotificationsEnabled = false;
```

```
IocUtility.Resolve<IDialogPresenter>().ShowToast("Повідомлення
вимкнені");
        }

        public async void EnableNotifications()
        {
            foreach (var push in _timetable.Select(item => new
ScheduledPush
            {
                Name = $"{Airship.Instance.ChannelId}
{item.Course.Id}",
                Schedule = new Schedule {ScheduledTime =
item.Date.AddDays(-1)},
                Push = new Push
                {
                    Audience = new Audience {AndroidChannel =
Airship.Instance.ChannelId},
                    DeviceTypes = new[]
{Constants.UAirshipDeviceType.Android},
                    Notification = new Notification
                    {
                        Android = new PlatformSpecificNotification
                        {
                            Title = item.Type.DescriptionString(),
                            Message =
$"{item.Course.Name}\n{item.Date.Date.ToString("d")}"
                        }
                    }
                }
            }))
            {
                await _notificationsService.AddScheduledPush(push);
            }

            NotificationsEnabled = true;

IocUtility.Resolve<IDialogPresenter>().ShowToast("Повідомлення
увімкнені");
```

```
        }

        public void AddToCalendar()
        {
            foreach (var item in _timetable)
            {
                IocUtility.Resolve<ICalendarManager>()
                    .AddEvent(item.Course.Name,
item.Type.DescriptionString(), item.Date, item.Date.AddMinutes(80));
            }
            IocUtility.Resolve<IDialogPresenter>().ShowToast("Дату
екзамену було додано до календаря");
        }
    }
}
```

ExamViewModel.cs

```
using StudyApp.Core.Extensions;
using StudyApp.Core.Mvx.Items;

namespace StudyApp.Core.Mvx.ViewModels
{
    public class ExamViewModel : ObjectViewModel<Exam>
    {
        private Exam _exam;

        protected override void Init(Exam obj)
        {
            if (obj == null) return;
            Title = obj.Course;
            Subtitle = obj.Type.DescriptionString();
            Exam = obj;
        }

        public Exam Exam
        {
            get { return _exam; }
            set { SetProperty(ref _exam, value); }
        }
```

```
    }
}
```

GoogleSignUpViewModel.cs

```
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Mvx.Platform;
using StudyApp.Core.Service.Interface;
using static MvvmCross.Extensions.Core.Util.IocUtility;

namespace StudyApp.Core.Mvx.ViewModels
{
    public class GoogleSignUpViewModel : ViewModel
    {
        private readonly IDataService _dataService;
        private readonly IAuthenticationService
_authenticationService;
        private string _recordBook;
        private bool _loading;

        public string RecordBook
        {
            get { return _recordBook; }
            set { SetProperty(ref _recordBook, value); }
        }

        public bool Loading
        {
            get { return _loading; }
            set { SetProperty(ref _loading, value); }
        }

        public GoogleSignUpViewModel(IDataService dataService,
IAuthenticationService authenticationService)
        {
            _dataService = dataService;
            _authenticationService = authenticationService;
            Title = "Реєстрація";
        }
```

```
        public async void SignUp(string id, string displayName, string
email)
        {
            if (string.IsNullOrEmpty(RecordBook))
            {
                Resolve<IDialogPresenter>().ShowDialog("Введіть   номер
залікової книжки");
                return;
            }
            Loading = true;
            var student = await _dataService.FindStudent(RecordBook);
            if (student == null)
            {
                Resolve<IDialogPresenter>().ShowDialog("Студент      не
знайдений");
                Loading = false;
                return;
            }
            var              response              =              await
_authenticationService.SignUp(RecordBook, id, displayName, email);
            if (response.IsError)
            {
                Resolve<IDialogPresenter>().ShowDialog("Помилка
реєстрації");
                Loading = false;
                return;
            }
            response = await _authenticationService.Login(email, id);
            if (response.IsError)
            {
                Resolve<IDialogPresenter>().ShowDialog("Помилка
входу");
                Loading = false;
            }
            ShowViewModel<MainViewModel>();
        }
    }
}
```

GroupViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.Util;
using MvvmCross.Extensions.Core.ViewModels;
using MvvmCross.Plugins.WebBrowser;
using StudyApp.Core.Model.Entities;
using StudyApp.Core.Mvx.Items;
using StudyApp.Core.Mvx.Messages;
using StudyApp.Core.Mvx.Platform;
using StudyApp.Core.Service.Interface;

namespace StudyApp.Core.Mvx.ViewModels
{
    [NavigationItem(Title = "Група", Description = "Група", Icon =
"ic_group_black_24dp")]
    public class GroupViewModel : CollectionViewModel<Profile>
    {
        private readonly IAuthenticationService
_authenticationService;
        private readonly IDataService _dataService;
        private Group _group;

        public GroupViewModel(IAuthenticationService
authenticationService, IDataService dataService)
        {
            _authenticationService = authenticationService;
            _dataService = dataService;
        }

        protected override async Task<IEnumerable<Profile>> OnLoad()
        {
            if (_group == null)
                _group = await
_dataService.FindGroup(_authenticationService.Profile);
            var title = _group.Name;
            var subtitle = _group.Specialty;
```

```
            MessengerUtility.Publish(new SetTitleMessage(this, typeof
(MainViewModel), title, subtitle));

            var items = new List<Profile>();

            foreach (var student in _group.StudentsProfiles)
            {
                var results = await
_dataService.GetCourseResults(student);
                items.Add(new Profile
                {
                    FullName = $"{student.LastName}
{student.FirstName}",
                    RecordBook = student.RecordBook,
                    Average = results.Count != 0 ? results.Sum(r =>
r.Result)/results.Count : 0
                });
            }

            return items.OrderByDescending(i => i.Average).ToList();
        }

        public void SendEmail()
        {
            IocUtility
                .Resolve<IDialogPresenter>()
                .ShowDialog("Відправити пошту",
                    _group.StudentsProfiles.Select(c =>
c.LastName).ToArray(),
                    new int[0], Send);
        }

        private async void Send(IList<string> obj)
        {
            var emails = new List<string>();
            foreach (
                var profile in
                    _group.StudentsProfiles
                        .Where(p => obj.Contains(p.LastName))
```

```
                        .Where(profile => profile.User != Guid.Empty))
            {
                var user = await _dataService.GetUser(profile.User);
                if (user != null && user.Email != _authenticationService.User.Email) emails.Add(user.Email);
            }

            if (emails.Count <= 0) return;
            var link = $"mailto:{string.Join(",", emails)}";

IocUtility.Resolve<IMvxWebBrowserTask>()?.ShowWebPage(link);
        }
    }
}
```

LessonViewModel.cs

```
using StudyApp.Core.Extensions;
using StudyApp.Core.Mvx.Items;

namespace StudyApp.Core.Mvx.ViewModels
{
    public class LessonViewModel : ObjectViewModel<Lesson>
    {
        private Lesson _lesson;

        protected override void Init(Lesson obj)
        {
            if (obj == null) return;
            Title = obj.Course;
            Subtitle = obj.Type.DescriptionString();
            Lesson = obj;
        }

        public Lesson Lesson
        {
            get { return _lesson; }
            set { SetProperty(ref _lesson, value); }
        }
    }
```

```
}
```

LoginViewModel.cs

```
using MvvmCross.Core.ViewModels;
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Mvx.Platform;
using StudyApp.Core.Service.Interface;
using static MvvmCross.Extensions.Core.Util.IocUtility;

namespace StudyApp.Core.Mvx.ViewModels
{
    public class LoginViewModel : ViewModel
    {
        private readonly IAuthenticationService
_authenticationService;
        private bool _loading;
        private string _email;
        private string _password;

        public IMvxCommand SignUp { get; }
        public IMvxCommand Login { get; }
        public IMvxCommand GoogleSignUp { get; }

        public bool Loading
        {
            get { return _loading; }
            set { SetProperty(ref _loading, value); }
        }

        public string Email
        {
            get { return _email; }
            set { SetProperty(ref _email, value); }
        }

        public string Password
        {
            get { return _password; }
            set { SetProperty(ref _password, value); }
```

```
        }

        public LoginViewModel(IAuthenticationService
authenticationService)
        {
            _authenticationService = authenticationService;

            Title = "Study App";

            SignUp = new MvxCommand(() =>
ShowViewModel<SignUpViewModel>());
            GoogleSignUp = new MvxCommand(() =>
ShowViewModel<GoogleSignUpViewModel>());
            Login = new MvxCommand(OnLogin);
        }

        public async void OnLogin()
        {
            if (Loading) return;
            if (string.IsNullOrEmpty(Email) ||
                string.IsNullOrEmpty(Password))
            {
                Resolve<IDialogPresenter>().ShowDialog("Заповніть всі
поля");
                return;
            }
            Loading = true;
            var response = await
_authenticationService.Login(Email.Replace(" ", ""), Password);
            if (response.IsError)
                Resolve<IDialogPresenter>().ShowDialog("Користувач не
знайден");
            else
                ShowViewModel<MainViewModel>();
            Loading = false;
        }

        public async void GoogleLogin(string id, string displayName,
string email)
```

```
        {
            if (Loading) return;
            Loading = true;
            var response = await _authenticationService.Login(email,
id);
            if (response.IsError)
            {
                Resolve<IDialogPresenter>().ShowDialog("Помилка
входу");
                Loading = false;
                return;
            }
            ShowViewModel<MainViewModel>();
            Loading = false;
        }
    }
}
```

MainViewModel.cs

```
using System;
using System.Collections.Generic;
using MvvmCross.Extensions.Core.Util;
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Mvx.Messages;
using StudyApp.Core.Service.Interface;

namespace StudyApp.Core.Mvx.ViewModels
{
    public class MainViewModel : NavigationViewModel
    {
        private readonly IAuthenticationService
_authenticationService;
        private readonly IDataService _dataService;
        private string _name;
        private string _group;

        public override IList<Type> ViewModels { get; } = new[]
        {
            typeof (NewsViewModel),
```

```
            typeof (TimetableViewModel),
            typeof (TasksViewModel),
            typeof (AbsenceViewModel),
            typeof (ExamsTimetableViewModel),
            typeof (GroupViewModel),
            typeof (ResultsViewModel),
            typeof (SettingsViewModel),
        };

        public string Name
        {
            get { return _name; }
            set { SetProperty(ref _name, value); }
        }

        public string Group
        {
            get { return _group; }
            set { SetProperty(ref _group, value); }
        }

        public                    MainViewModel(IAuthenticationService
authenticationService, IDataService dataService)
        {
            _authenticationService = authenticationService;
            _dataService = dataService;
            MessengerUtility.Subscribe<SetTitleMessage>(this,
SetTitle);
        }

        private void SetTitle(SetTitleMessage obj)
        {
            if (obj.TargetViewModel != GetType()) return;
            Title = obj.Title;
            Subtitle = obj.Subtitle;
        }

        public override async void Start()
        {
```

```
            base.Start();
            var student = _authenticationService.Profile;
            var group = await _dataService.FindGroup(student);
            Name = $"{student.FirstName} {student.LastName}";
            Group = group.Name;
        }
    }
}
```

NewsViewModel.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Model.Entities;
using StudyApp.Core.Service.Interface;

namespace StudyApp.Core.Mvx.ViewModels
{
    [NavigationItem(Title = "Головна", Description = "Новини", Icon =
"ic_home_black_24dp")]
    public class NewsViewModel : CollectionViewModel<News>
    {
        private readonly IDataService _dataService;

        public NewsViewModel(IDataService dataService)
        {
            _dataService = dataService;
        }

        protected override async Task<IEnumerable<News>> OnLoad()
        {
            return await _dataService.Get<News>();
        }
    }
}
```

ObjectViewModel.cs

```
using MvvmCross.Extensions.Core.ViewModels;
using MvvmCross.Extensions.Core.ViewModels.Parameters;
using static MvvmCross.Extensions.Core.Util.SerializationUtility;

namespace StudyApp.Core.Mvx.ViewModels
{
    public abstract class ObjectViewModel<TObj> : ViewModel<ViewModelParameter>
    {
        public override void Init(ViewModelParameter parameters) => Init(DeserializeObject<TObj>(parameters?.Value));

        protected abstract void Init(TObj obj);
    }
}
```

ResultsViewModel.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Model.Entities;
using StudyApp.Core.Mvx.Items;
using StudyApp.Core.Service.Interface;

namespace StudyApp.Core.Mvx.ViewModels
{
    [NavigationItem(Title = "Результати", Description = "Результати", Icon = "ic_assessment_black_24dp")]
    public class ResultsViewModel : CollectionViewModel<CourseResult>
    {
        private readonly IDataService _dataService;
        private readonly IAuthenticationService _authenticationService;

        public ResultsViewModel(IDataService dataService, IAuthenticationService authenticationService)
        {
            _dataService = dataService;
            _authenticationService = authenticationService;
```

```
        }

        protected override async Task<IEnumerable<CourseResult>>
OnLoad()
        {
            var student = _authenticationService.Profile;
            var courses = await _dataService.Get<Course>();
            var results = new List<CourseResult>();
            foreach (var course in courses)
            {
                var controlResult = await
_dataService.GetCourseResult(course, student);
                var result = new CourseResult
                {
                    Name = course.Name,
                    Result = controlResult?.Result ?? -1,
                    Tasks = new List<TaskResult>(),
                    Type = course.Type
                };

                results.Add(result);

                course.Tasks = await _dataService.GetTasks(course);
                foreach (var task in course.Tasks)
                {
                    var studentTaskResult = await
_dataService.GetTaskResult(task, student);
                    var taskResult = new TaskResult
                    {
                        Name = task.Name,
                        Result = studentTaskResult?.Result ?? -1,
                        Type = task.Type
                    };

                    result.Tasks.Add(taskResult);
                }

            }
            return results;
```

```
        }

    }

}
```

SettingsViewModel.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.Util;
using MvvmCross.Extensions.Core.ViewModels;
using MvvmCross.Plugins.WebBrowser;
using StudyApp.Core.Extensions;
using StudyApp.Core.Model.Entities;
using StudyApp.Core.Mvx.Platform;
using StudyApp.Core.Service.Interface;
using StudyApp.Core.Settings;
using UAirshipSender;
using UAirshipSender.Model;
using UrbanAirship.Portable;

namespace StudyApp.Core.Mvx.ViewModels
{
    [NavigationItem(Description = "Налаштування", Title =
"Налаштування", Icon = "ic_settings_black_24dp")]
    public class SettingsViewModel :
CollectionViewModel<SettingsGroup>
    {
        private readonly IDataService _dataService;
        private readonly INotificationsService _notificationsService;
        private IList<ExamsTimetable> _timetable;

        public SettingsViewModel(IDataService dataService,
INotificationsService notificationsService)
        {
            _dataService = dataService;
            _notificationsService = notificationsService;
```

```
        }

        protected override async Task<IEnumerable<SettingsGroup>>
OnLoad()
        {
            _timetable = await _dataService.GetExamsTimetable();
            return new List<SettingsGroup>
            {
                new SettingsGroup("Пошта")
                {
                    new SwitchSetting("Розсилка", "Отримувати
повідомлення і новини на пошту", true, null)
                },
                new SettingsGroup("Розклад")
                {
                    new SwitchSetting("Повідомлення", "Отримувати
повідомлення про майбутні іспити", false,
                        OnNotificationsClick),
                    new SwitchSetting("Календар", "Додавати розклад в
календар", false, OnCalendarClick)
                },
                new SettingsGroup("Контакти")
                {
                    new Setting("Адреса", "85300, Україна, Донецька
область, м. Покровськ, пл. Шибанкова, 2"),
                    new Setting("Офіційна пошта",
"mail@donntu.edu.ua", OnEmailClick)
                }
            };
        }

        private static void OnEmailClick(Setting obj)
        {

IocUtility.Resolve<IMvxWebBrowserTask>()?.ShowWebPage($"mailto:{obj.Su
btitle}");
        }

        private void OnCalendarClick(SwitchSetting obj)
```

```
        {
            if (!obj.Checked) return;
            foreach (var item in _timetable)
            {
                IocUtility.Resolve<ICalendarManager>()
                    .AddEvent(item.Course.Name,
item.Type.DescriptionString(), item.Date, item.Date.AddMinutes(80));
            }
        }

        private async void OnNotificationsClick(SwitchSetting obj)
        {
            if (obj.Checked)
            {
                foreach (var push in _timetable.Select(item => new
ScheduledPush
                {
                    Name = $"{Airship.Instance.ChannelId}
{item.Course.Id}",
                    Schedule = new Schedule {ScheduledTime =
item.Date.AddDays(-1)},
                    Push = new Push
                    {
                        Audience = new Audience {AndroidChannel =
Airship.Instance.ChannelId},
                        DeviceTypes = new[]
{Constants.UAirshipDeviceType.Android},
                        Notification = new Notification
                        {
                            Android = new PlatformSpecificNotification
                            {
                                Title = item.Type.DescriptionString(),
                                Message =
$"{item.Course.Name}\n{item.Date.Date.ToString("d")}"
                            }
                        }
                    }
                }))
                {
```

```
                    await
_notificationsService.AddScheduledPush(push);
                }
            }
            else
            {
                var notifications =
                    (await _notificationsService.GetSchedules())
                        .Where(n                                    =>
n.Name.Contains(Airship.Instance.ChannelId))
                        .ToList();

                if (notifications != null)
                    foreach (var push in notifications)
                        await
_notificationsService.RemoveScheduledPush(push);
            }
        }
    }
}
```

SignUpViewModel.cs

```
using MvvmCross.Core.ViewModels;
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Mvx.Platform;
using StudyApp.Core.Service.Interface;
using static MvvmCross.Extensions.Core.Util.IocUtility;

namespace StudyApp.Core.Mvx.ViewModels
{
    public class SignUpViewModel : ViewModel
    {
        private readonly IDataService _dataService;
        private             readonly             IAuthenticationService
_authenticationService;
        private string _recordBook;
        private bool _loading;
        private string _email;
        private string _password;
```

```
        private string _confirmPassword;

        public string RecordBook
        {
            get { return _recordBook; }
            set { SetProperty(ref _recordBook, value); }
        }

        public string Email
        {
            get { return _email; }
            set { SetProperty(ref _email, value); }
        }

        public string Password
        {
            get { return _password; }
            set { SetProperty(ref _password, value); }
        }

        public string ConfirmPassword
        {
            get { return _confirmPassword; }
            set { SetProperty(ref _confirmPassword, value); }
        }

        public bool Loading
        {
            get { return _loading; }
            set { SetProperty(ref _loading, value); }
        }

        public IMvxCommand SignUp { get; }


        public SignUpViewModel(IDataService dataService,
IAuthenticationService authenticationService)
        {
            _dataService = dataService;
```

```
            _authenticationService = authenticationService;

            SignUp = new MvxCommand(OnSignUp);

            Title = "Реєстрація";
        }

        public async void OnSignUp()
        {
            if (Loading) return;

            var email = Email.Replace(" ", "");

            if (string.IsNullOrEmpty(RecordBook) ||
                string.IsNullOrEmpty(email) ||
                string.IsNullOrEmpty(Password) ||
                string.IsNullOrEmpty(ConfirmPassword))
            {
                Resolve<IDialogPresenter>().ShowDialog("Заповніть всі
поля");
                return;
            }

            Loading = true;

            var student = await _dataService.FindStudent(RecordBook);
            if (student == null)
            {
                Resolve<IDialogPresenter>().ShowDialog("Студент не
найден");
                Loading = false;
                return;
            }
            var response = await
_authenticationService.SignUp(RecordBook, email, Password);
            if (response.IsError)
            {
                Resolve<IDialogPresenter>().ShowDialog("Помилка
реєстрації");
```

```
                Loading = false;
                return;
            }
            response = await _authenticationService.Login(email,
Password);
            if (response.IsError)
            {
                Resolve<IDialogPresenter>().ShowDialog("Помилка
входу");
                Loading = false;
                return;
            }
            ShowViewModel<MainViewModel>();
        }
    }
}
```

TasksViewModel.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.ViewModels;
using StudyApp.Core.Model.Entities;
using StudyApp.Core.Service.Interface;

namespace StudyApp.Core.Mvx.ViewModels
{
    [NavigationItem(Title = "Завдання", Description = "Завдання", Icon
= "ic_work_black_24dp")]
    public class TasksViewModel : CollectionViewModel<Course>
    {
        private readonly IDataService _dataService;

        public TasksViewModel(IDataService dataService)
        {
            _dataService = dataService;
        }
```

```
        protected override async Task<IEnumerable<Course>> OnLoad()
        {
            var courses = await _dataService.Get<Course>();
            foreach (var course in courses)
                course.Tasks = await _dataService.GetTasks(course);
            return courses.Where(c => c.Tasks.Any()).ToList();
        }
    }
}
```

TimetableViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MvvmCross.Core.ViewModels;
using MvvmCross.Extensions.Core.Attributes;
using MvvmCross.Extensions.Core.ViewModels;
using MvvmCross.Extensions.Core.ViewModels.Parameters;
using StudyApp.Core.Model.Enum;
using StudyApp.Core.Mvx.Items;
using StudyApp.Core.Service.Interface;
using static MvvmCross.Extensions.Core.Util.SerializationUtility;
using static StudyApp.Core.Model.Enum.WeekType;

namespace StudyApp.Core.Mvx.ViewModels
{
    [NavigationItem(Title = "Розклад занять", Description = "Розклад
занять", Icon = "ic_date_range_black_24dp")]
    public class TimetableViewModel :
CollectionViewModel<TimetableDay>
    {
        private readonly IDictionary<WeekType, IList<TimetableDay>>
_itemsByWeekType =
            new Dictionary<WeekType, IList<TimetableDay>>
            {
                [Upper] = new List<TimetableDay>(),
                [Lower] = new List<TimetableDay>()
            };
```

```
        private static readonly Dictionary<int, TimeSpan> StartTime =
new Dictionary<int, TimeSpan>
        {
            [1] = new TimeSpan(8, 0, 0),
            [2] = new TimeSpan(9, 35, 0),
            [3] = new TimeSpan(11, 20, 0),
            [4] = new TimeSpan(12, 50, 0),
            [5] = new TimeSpan(14, 20, 0),
            [6] = new TimeSpan(15, 50, 0),
        };

        private static readonly Dictionary<int, TimeSpan> EndTime =
new Dictionary<int, TimeSpan>
        {
            [1] = new TimeSpan(9, 20, 0),
            [2] = new TimeSpan(10, 55, 0),
            [3] = new TimeSpan(12, 40, 0),
            [4] = new TimeSpan(14, 10, 0),
            [5] = new TimeSpan(15, 40, 0),
            [6] = new TimeSpan(17, 10, 0),
        };

        private readonly IDataService _dataService;
        private WeekType _selectedType = Upper;

        public IList<WeekType> WeekTypes { get; } = new[] {Upper,
Lower};

        public IMvxCommand WeekTypeChanged { get; private set; }

        public TimetableViewModel(IDataService dataService)
        {
            _dataService = dataService;
            WeekTypeChanged                    =                    new
MvxCommand<WeekType>(OnWeekTypeChanged);
        }

        private void OnWeekTypeChanged(WeekType weekType)
```

```
        {
            _selectedType = weekType;
            Items = _itemsByWeekType[_selectedType];
        }

        protected override async Task<IEnumerable<TimetableDay>>
OnLoad()
        {
            var timetable = await _dataService.GetTimetable();
            foreach (var items in timetable.GroupBy(t => t.DayOfWeek))
            {
                var lower = items.Where(t => t.WeekType ==
Lower).ToList();
                var upper = items.Where(t => t.WeekType ==
Upper).ToList();

                _itemsByWeekType[Lower].Add(new TimetableDay
                {
                    Day = DateTime.Now.AddDays((int) items.Key - (int)
DateTime.Now.DayOfWeek),
                    Items = lower
                        .Select(i => new Lesson
                        {
                            Building = i.Building,
                            Classroom = i.Classroom,
                            Course = i.Course.Name,
                            Lecturer = i.Course.Lecturer,
                            Type = i.LessonType,
                            Number = i.LessonNumber,
                            Start = StartTime[i.LessonNumber],
                            End = EndTime[i.LessonNumber],
                        }).OrderBy(i => i.Number).ToList(),
                    ItemSelected = new
MvxCommand<TimetableItem>(OnItemSelected)
                });

                _itemsByWeekType[Upper].Add(new TimetableDay
                {
```

```
                    Day = DateTime.Now.AddDays((int) items.Key - (int)
DateTime.Now.DayOfWeek),
                    Items = upper
                        .Select(i => new Lesson
                        {
                            Building = i.Building,
                            Classroom = i.Classroom,
                            Course = i.Course.Name,
                            Lecturer = i.Course.Lecturer,
                            Type = i.LessonType,
                            Number = i.LessonNumber,
                            Start = StartTime[i.LessonNumber],
                            End = EndTime[i.LessonNumber],
                        }).OrderBy(i => i.Number).ToList(),
                    ItemSelected                    =                    new
MvxCommand<TimetableItem>(OnItemSelected)
                });
            }

            return _itemsByWeekType[_selectedType];
        }

        protected virtual void OnItemSelected(TimetableItem item)
        {
            ShowViewModel<LessonViewModel>(new
ViewModelParameter(SerializeObject(item)));
        }
    }
}
```

Додаток В – Структура бази даних

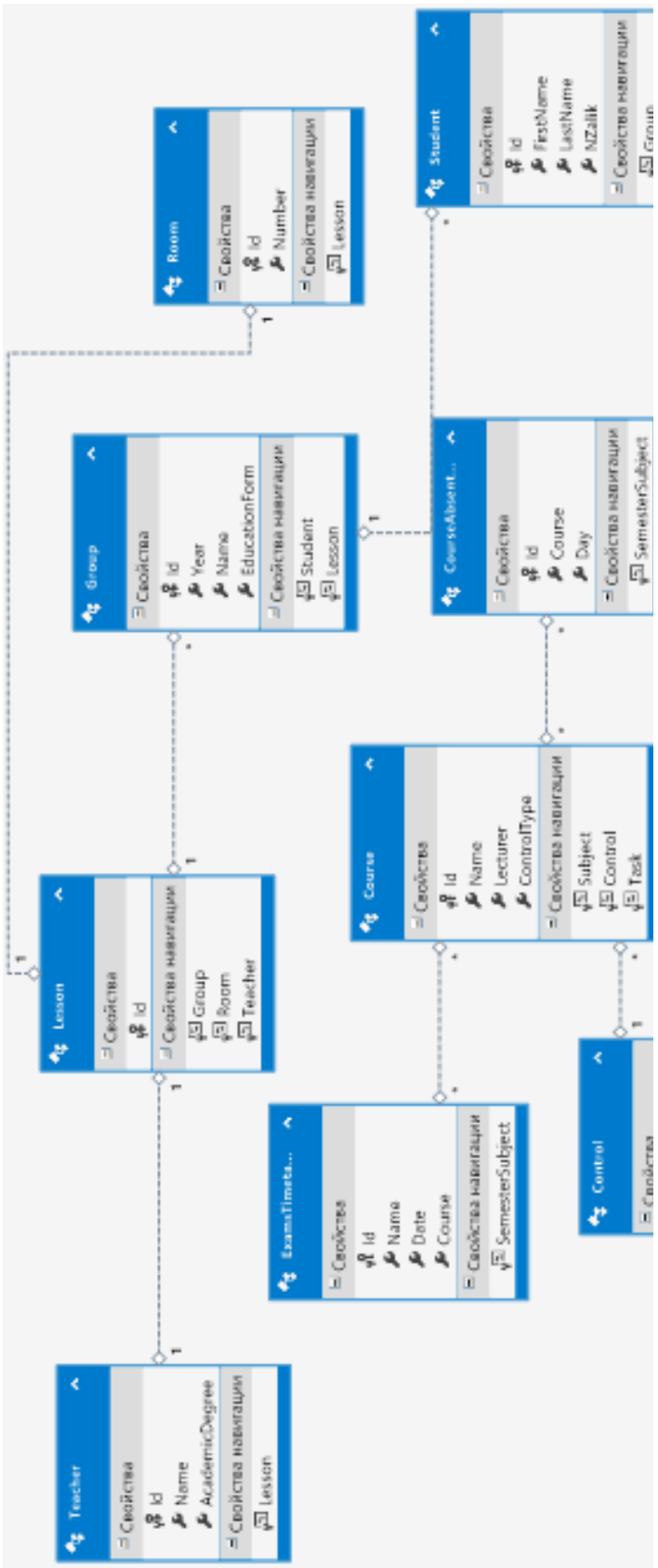

Рисунок 1 – Структура бази даних

Додаток Г – Перелік зауважень нормоконтролера

Позначення документа	Сторінка	Зміст зауваження