ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики і інформатики

«До захисту допущено»
Завідувач кафедри ПМІ

_____ Ольга ДМИТРІЄВА
(підпис) (ініціали, прізвище)
«8» червня 2022 р.

Випускна кваліфікаційна робота

бакалавра
(освітньо–кваліфікаційний рівень)

на тему

«Захист конфіденційної інформації в месенджерах»
зі спецчастиною
«Розробка менеджеру паролів з підвищеним захистом на основі телеграм боту»

Виконав: студент 4 курсу, групи КІБ–18
(шифр групи)
спеціальності 125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)

Парубов В.А. _____
(прізвище та ініціали) (підпис)

Керівник проф. каф. ПМІ, д.т.н., проф. Євген БАШКОВ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Рецензент доц. каф. КІ, к.т.н., Юлія ДІКОВА _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Студент _____
(підпис)

Луцьк – 2022

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації

Кафедра прикладної математики і інформатики

Освітньо-кваліфікаційний рівень (освітній ступінь) бакалавр

Напрям підготовки (спеціальність) 125 «Кібербезпека»

ЗАТВЕРДЖУЮ:

Завідувач кафедри ПМІ

О.А. Дмитрієва

06.04.2022 р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ РОБОТУ СТУДЕНТУ

Парубов Владислав Андрійович

(прізвище, ім'я, по батькові)

1. Тема роботи «Захист конфіденційної інформації в месенджерах»

Спецчастина «Розробка менеджеру паролів з підвищеним захистом на основі телеграм боту»

Керівник роботи проф. каф. ПМІ, д.т.н., проф. Башков Є.О.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від " 06 " травня 2022 року № 180

2. Строк подання студентом роботи "17" червня 2022 року

3. Вихідні дані до роботи результати переддипломної практики, теоретичні матеріали, наукова література

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

1) аналіз предметної області та засобів розробки менеджеру паролів з підвищеним захистом на основі телеграм боту;

2) проєктування менеджеру паролів з підвищеним захистом на основі телеграм боту;

3) розробка менеджеру паролів з підвищеним захистом на основі телеграм боту;

4) тестування менеджеру паролів з підвищеним захистом на снові телеграм боту;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) робота містить 2 таблиці, 27 рисунків та інший матеріал, у кількості достатній для демонстрації результатів кваліфікаційної роботи.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 06 квітня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка задачі	01.04.22–06.04.22	
2	Проектування моделі додатку, вибір та обґрунтування засобів реалізації	11.04.22–24.04.22	
3	Вибір алгоритмів та проектування чат-боту менеджеру паролів	26.04.22–16.05.22	
4	Тестування розробленого чат-боту	17.05.22–22.05.22	
5	Оформлення пояснювальної записки	23.05.22–05.06.22	
6	Нормоконтроль пояснювальної записки, її перевірка антиплагіатною системою	06.06.22–12.06.22	
7	Оформлення супутніх матеріалів (розробка графічного матеріалу, написання доповіді, тощо) та рецензування роботи	12.06.2022	
8	Захист роботи	24.06.2022	

Студент _____ Парубов В.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Башков Є.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Парубов В. А. Захист конфіденційної інформації в месенджерах / Випускна кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 125 «Кібербезпека». – ДВНЗ ДонНТУ, Луцьк, 2022.

Об’єкт дослідження – процеси та процедури захисту інформації в соціальних мережах

Предмет дослідження – менеджер паролів для месенджеру з підвищеним захистом на основі телеграм боту.

Мета роботи – розробка дизайну і функціоналу менеджеру паролів з підвищеним захистом на основі телеграм боту.

Методи дослідження – аналіз додатків існуючої проблеми менеджерів паролів, виявлення переваг і недоліків; визначення вимог до структури, поведінки і функціоналу чат-боту; вибір інструментальних засобів розробки.

Розробка є оригінальною, бо це україномовний бот аналогів якому поки що немає, аналогічні чат-боти генерації паролів не мають можливості вибору типу паролю.

При розробці використовувалися спеціальні технології для компонування даних – Docker, опрацьовані методи шифрування, визначено їх типи та можливості для захисту даних, розроблено чат-бот менеджер паролів, та проведено різнопланове тестування, згідно прописаний кейсів тестування.

Практичне значення полягає у розробці чат-боту для генерації унікальних складних паролів з миттєвою можливістю доступу до даних.

Ключові слова: телеграм, чат-бот, TelegramAPI, генерація паролю.

ANNOTATION

Parubov VA Protection of confidential information in messengers / Graduation thesis for the degree of «bachelor» in the specialty 125 «Cybersecurity». - SHEI DonNTU, Lutsk, 2022.

The object of research is the processes and procedures of information protection in social networks

The subject of the research is a password manager for a messenger with increased protection based on bot telegrams.

The purpose of the work is to develop the design and functionality of a password manager with increased protection based on bot telegrams.

Research methods - analysis of applications of the existing problem of password managers, identification of advantages and disadvantages; defining requirements for the structure, behavior and functionality of the chatbot; choice of development tools.

The development is original, because it is a Ukrainian-language bot that has no analogues yet, similar chatbots for generating passwords do not have the ability to choose the type of password.

The development used special technologies for data composition - Docker, developed encryption methods, identified their types and capabilities for data protection, developed a chatbot password manager, and conducted a variety of testing, according to the prescribed test cases.

Of practical importance is the development of a chatbot to generate unique strong passwords with instant access to data.

Keywords: telegram, chatbot, TelegramAPI, password generation.

ЗМІСТ

ВСТУП

Чат-бот – це віртуальний співрозмовник або помічник, здатний взаємодіяти з користувачем із використанням природної мови. У сучасному світі були реалізовані такі чат-боти, які використовуються для освіти, пошуку інформації, обслуговування клієнтів, навігації по сайту, існують навіть боти, що надають розважальні послуги.

Однак припустимо, що в області генерування паролів з підвищеним захистом – робота буде першою спробою створення програми співрозмовника, який має можливість спілкуватися з клієнтами/користувачами у режимі 24/7 годин.

Багато дослідників та експертів у галузі безпеки настійно рекомендують усім використовувати менеджери паролів, тому що вони чудово справляються із завданням, пропонуючи набагато краще поєднання зручності та безпеки. Більшість людей або використовують паролі, що повторюються, або містять у своїх паролях біти інформації, такі як їх ім’я, дата народження і т.д.

В даний час здійснюються банківські операції онлайн, включаючи платежі за автомобіль, житло та кредити.

Також з’являється багато соціальних акаунтів з особистою інформацією та спілкуванням.

На основі існуючих розробок був представлений метод для процесу генерації паролів, але під час написання цієї роботи буде вдосконалено існуючи алгоритми та впроваджено у розробку.

Об’єкт дослідження – процеси та процедури захисту інформації в соціальних мережах

Предмет дослідження – менеджер паролів для месенджеру з підвищеним захистом на основі телеграм боту.

Мета роботи – розробка дизайну і функціоналу менеджеру паролів з підвищеним захистом на основі телеграм боту.

Методи дослідження – аналіз додатків існуючої проблеми менеджерів паролів, виявлення переваг і недоліків; визначення вимог до структури, поведінки і функціоналу чат-боту; вибір інструментальних засобів розробки.

Задачі розробки:

– проаналізувати літературу та інтернет-джерела, з метою відбору та використання матеріалу по створенню чат-боту;

– визначити програмне забезпечення для розробки чат-боту;

– реалізувати менеджер паролів з підвищеним захистом на основі телеграм боту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЗАСОБІВ РОЗРОБКИ МЕНЕДЖЕРУ ПАРОЛІВ З ПІДВИЩЕНИМ ЗАХИСТОМ НА ОСНОВІ ТЕЛЕГРАМ БОТУ

1.1 Поняття захисту даних в месенджерах

Месенджери добре вписалися в сучасний процес цифровізації суспільства, і цілком об’єктивно очікувати від цифрового спілкування приблизно такої ж безпеки, як і при вербальному (особистому) спілкуванні співрозмовників.

Основні параметри безпеки:

– конфіденційність розмови (вербальну розмову між співрозмовниками можна вважати конфіденційною за умови, що ніхто не чує розмови учасників);

– ідентифікація та автентифікація учасника розмови (співрозмовника ми зазвичай впізнаємо (ідентифікуємо) за його персональними особливостями (голос, зовнішність та ін.), запам’ятовуємо їх разом з ім’ям учасника діалогу при першому знайомстві, а згодом переконуємось у тому, що перед нами саме та людина, за яку він себе видає);

– автентифікація джерела даних (перевірка та підтвердження того, що інформація створена саме заявленим учасником розмови).

Зазвичай у розмові можна визначити, хто і що сказав, однак за цифрового спілкування це не очевидно. У межах цієї властивості також виділяють поняття «цілісність» – відсутність зміни у інформації, що передається, тобто підтвердження того, що ми почули саме те, що сказано. Зазвичай співрозмовники при вербальній розмові не замислюються про цю властивість, проте при цифровому спілкуванні, коли дані легко зберігати і змінювати, воно стає дуже важливим.

За вербальної розмови двох осіб третя особа, за умови виконання якості конфіденційності, може дізнатися зміст розмови лише зі слів одного з

учасників. При цьому, за відсутності підслуховуючих осіб і записуючих пристроїв немає можливості довести, що конкретний учасник вербальної розмови говорив саме те, що затверджується третьою особою. Таким чином, для цифрового спілкування треба виключити можливість будь-кого довести авторство повідомлення і, більше, довести сам факт розмови.

1.1.1 Методи захисту даних в месенджерах

Незважаючи на різноманітність систем миттєвого обміну повідомленнями, у більшості з них використовуються схожі принципи побудови або базові криптографічні механізми: вироблення секретних ключів та узгодження ключа між користувачами, встановлення захищеного каналу зв'язку між користувачами, зміна ключових елементів, криптографічні методи забезпечення конфіденційності та цілісності передачі інформації.

Усі системи миттєвого обміну повідомленнями за своєю архітектурою можна розділити на дві групи: з симетричним (наприклад, Briar) та асиметричним каналом зв'язку (наприклад, WhatsApp). При використанні систем першої групи обмін повідомленнями між двома користувачами можливий лише коли обидва підключені до загального каналу передачі даних, а у разі застосування систем другої групи обмін повідомленнями між користувачами можливий за умови, коли хоча б один з них має доступ до загального каналу передачі даних . Побудова асиметричного каналу вимагає наявності сервісу доставки - пристрою, постійно підключеного до каналу зв'язку, зберігає повідомлення (якщо одержувач «не в мережі») і користувачу, що пересилає їх, коли він з'явиться в мережі.

Процес взаємодії двох учасників розмови A і B починається з вироблення загального секретного ключа та автентифікації абонентів. Вироблення загального секретного ключа здійснюється за допомогою протоколу Діффі – Хеллмана [1] з параметрами p і g групи Zp, де p - деяке просте число, а g - утворює елемент підгрупи групи Zp. Сторони A та B

виробляють секретні ключі *xA, xB* і передають одна одній відкриті ключі g_A^x та g_B^x,, а загальним секретним ключем є величина $g_A^x g_B^x$. Загальний секретний ключ використовується для вироблення секретних ключів алгоритмів шифрування та імітозахисту. По значенням g_A^x та g_B^x, що спостерігаються в каналі зв'язку, зловмиснику досить складно визначити загальний секрет $g_A^x g_B^x$, однак він може реалізувати атаку «людина посередині». Для її виключення при першому виконанні протоколу Діффі-Хеллмана відбувається автентифікація абонентів шляхом застосування схеми електронного підпису:

$$A \to B: Sing(g_A^x, k_A), K_A \quad (1.1)$$
$$B \to A: Sing(g_B^x, k_B), K_B \quad (1.2)$$

де k_A та K_A – відповідно секретний та відкритий ключі абонента *A*, k_B та K_B – відповідно секретний та відкритий ключі абонента *B*, а *Sign* – процедура обчислення електронного підпису.

Наступний етап процесу взаємодії двох учасників розмови *A* та *B* – вироблення ключів шифрування та імітозахисту. Ці ключі виробляються із загального секрету шляхом застосування односпрямованої функції – наприклад, хеш-функції, використання якої дозволяє в разі необхідності «розширити» загальний секрет до потрібного розміру, отримавши ключі шифрування та імітозахисту [1]. На ключі шифрування накладаються вимоги про короткий термін їх експлуатації: пропонується змінювати їх якнайчастіше, наприклад після кожного повідомлення. При цьому використані ключі повинні відразу видалятися з пам'яті пристрою, а для захисту від читання вперед також необхідно за протоколом Діффі-Хеллмана постійно виробляти нові спільні секретні ключі.

Далі виконується етап передачі повідомлень, що шифруються за допомогою блокового шифру, наприклад, працює в режимі CTR (Counter mode – режим лічильника) або GCM (Galois/Counter Mode). Перший режим

забезпечує лише конфіденційність даних, що передаються, і для забезпечення автентифікації джерела даних додатково використовують ключові хеш-функції - наприклад, HMAC (Hash-based Message Authentication Code). Вона виробляє імітівставку – «контрольну суму» повідомлення, за якою власник секретного ключа може перевірити, чи отримане повідомлення є неушкодженим і незміненим. Причому сформувати «правильну» імітівставку може лише власник секретного ключа імітозахисту. Режим GCM дозволяє шифрувати повідомлення і одночасно виробляти імітівставку за допомогою лише одного секретного ключа. Подібні режими називаються «режимами шифрування з імітозахистом» [2, 3].

Таким чином, конфіденційність повідомлень забезпечується за рахунок їх шифрування алгоритмами блокового шифрування, автентифікація джерела даних та контроль цілісності повідомлень – за рахунок використання ключової хеш-функції або режиму шифрування з імітозахистом, а відмова від авторства досягається або за рахунок розголошення використаних секретних ключів. у протоколі OTR (Off-the-Record Messaging), або з допомогою розголошення відкритих ключів з протоколу Діффі-Хеллмана, як і протоколі Signal.

У першому випадку зловмисник формально може підробити або змінити будь-яке старе повідомлення, обчисливши йому коректну імітівставку. Відповідно, якщо будь-хто може скласти «коректне» повідомлення, неможливо довести, що саме якесь конкретне заявлене повідомлення передавалося користувачами. Проте конфіденційність у своїй не порушується.

У другому випадку зловмисник може скласти листування, «не відмінне від справжнього». Ніхто, крім тих, хто володіє секретним ключем *xA* або *xB* за ключами g_A^x та g_B^x, не може перевірити, що якийсь заявлений загальний секрет *ga* дійсно дорівнює величині g_A^x та g_B^x. Відповідно, без розголошення секретних ключів користувача неможливо довести сам факт розмови між конкретними користувачами. Навіть у разі розголошення одним із

користувачів свого секретного ключа інший учасник розмови може відмовитися від авторства всіх переданих повідомлень. Справді, за ключами *xA* та *gxB* можна виробити загальний секрет $g_{\mathrm{A}}^{x} g_{\mathrm{B}}^{x}$ та «ідентичне справжнє» листування [4].

Захист від читання забезпечується за рахунок використання односпрямованих функцій: нові ключі шифрування та імітозахисту формуються за допомогою застосування односпрямованої функції до старих ключів. Отже, якщо зловмисник знає нові ключі, йому обчислювально дуже важко визначити з них попередні ключі.

Захист від читання вперед забезпечується виробленням нового загального секрету за протоколом Діффі-Хеллмана. Цей загальний секрет «підмішується» в односпрямовану функцію, і якщо зловмиснику стали відомі старі ключі шифрування та імітозахисту, то вирахувати за ними нові ключі буде дуже трудомістким.

Загалом підхід із симетричним каналом схожий на ідею «криптоанархії»: ніхто один одному не довіряє, і система залежить лише від самих учасників та підтримується лише їхніми силами. До мінусів підходу слід віднести високе навантаження на пристрій користувача, якому потрібно постійно обчислювати нові ключі, виконувати шифрування та зберігати повідомлення до появи адресата в мережі. Також необхідно довіряти інфраструктурі відкритих ключів, яка зберігатиме значення $Sing(g_A^x, k_A), K_A$, або потрібна особиста зустріч для обміну значеннями $Sing(g_A^x, k_A), K_A$ [5].

Як приклад такої системи можна назвати проект Briar зі створення програми обміну повідомленнями, призначеного всім, кому потрібен безпечний, простий і надійний спосіб спілкування. На відміну від традиційних інструментів обміну повідомленнями, таких як електронна пошта, Twitter або Telegram, Briar не використовує центральний сервер: повідомлення передаються безпосередньо між користувачами. Структура Briar схожа на протокол OTR, але з істотною відмінністю: після вироблення загального секрету все нові ключі шифрування та імітозахисту виробляються

за допомогою односпрямованої функції зі старих ключів без «підмішування» нового загального секрету. Таким чином, якщо зловмисник у якийсь момент дізнався один ключ шифрування, він зможе читати все наступне листування користувачів.

У системах з асиметричним каналом зв’язку користувач *A* створює обліковий запис на сервісі автентифікації, отримує ідентифікатор та розміщує на сервісі свої відкриті ключі, підписані електронним підписом. Для обміну повідомленнями користувач запитує у сервісу автентифікації відкритий ключ користувача та обчислює загальний секрет. У першому повідомленні користувач *A* вказує свій підписаний відкритий ключ, використаний у протоколі, і відкритий ключ користувача *B*, використаний для вироблення загального секрету. Це повідомлення надсилається до сервісу доставки, який надалі переправить його користувачеві *B*.

Далі користувач *B* перевіряє підпис отриманих у повідомленні відкритих ключів користувача *A* та по них виробляє спільний секрет та розшифровує повідомлення. Для надсилання нового повідомлення користувач *B* виробляє новий відкритий ключ, за допомогою отриманого раніше відкритого ключа користувача *A* виробляє новий спільний секрет протоколу Діффі-Хеллмана і за його допомогою шифрує своє нове повідомлення. Користувач *B* разом із зашифрованим повідомленням також передає свій новий відкритий ключ користувачу *A*. Ситуація повторюється дзеркально: для відправки нового повідомлення користувач *A* виробляє новий відкритий ключ, за допомогою отриманого раніше відкритого ключа користувача B виробляє новий спільний секрет за протоколом Діффі - Хеллмана та з його підписом шифрує своє повідомлення. Користувач A разом із зашифрованим повідомленням також передає свій новий відкритий ключ користувачеві *B*. Тут задіяні два основні протоколи для побудови безпечного асиметричного каналу зв’язку: X3DH і Double Ratchet.

Протокол X3DH (Extended Triple Diffie-Hellman) – це асинхронний протокол узгодження ключів, що дозволяє двом учасникам виробити

спільний секрет на основі відкритих ключів та одночасно провести автентифікацію учасників протоколу. У даному протоколі користувач A запитує три підписані ключі користувача, які той заздалегідь (наприклад, при своїй реєстрації) розмістив на сервісі автентифікації, і для кожного підписаного ключа виконує протокол Діффі-Хеллмана. Новий спільний секрет виходить шляхом об’єднання (конкатенації) результатів роботи трьох протоколів Діффі-Хеллмана.

Double Ratchet – протокол, що дозволяє двом учасникам за допомогою попередньо розподіленого загального секрету, виробленого за протоколом X3DH, обмінюватися зашифрованими повідомленнями. У даному протоколі один учасник виробляє новий спільний секрет за протоколом Діффі-Хеллмана за допомогою старого відкритого ключа іншого користувача (отриманого, наприклад, попередньої ітерації даного протоколу, виконаного іншим учасником) і свого нового секретного ключа (виробляється в процесі виконання даного протоколу; цьому відповідний відкритий ключ пересилається у відкритому вигляді іншому учаснику разом із зашифрованим повідомленням).

Головний недолік систем з асиметричним каналом зв’язку – необхідність довіри до сервісу автентифікації, за яким може ховатися зловмисник, який видає себе за іншого користувача. Справді, відкриті ключі користувачі отримують саме від сервісу автентифікації, і не можна перевірити їхню справжність, тому що сервіс, взагалі кажучи, може сам згенерувати пари відкритих/секретних ключів і надсилати на запит користувачів тільки такі пари. Таким чином, сервіс може виступати «людиною посередині», розшифровуючи все листування між користувачами.

1.1.2 Вимоги до парольного захисту та складності паролю

Вимоги до парольного захисту розглянемо на основі положення з трьох стандартів: NIST Special Publication 800-63B, ISO/IEC 27002:2013, CIS Password Policy Guide’s.

Розглянемо вимоги по кожному стандарту.

1. NIST Special Publication 800-63B [6].

– мінімальна довжина 8 символів та максимальна довжина не менше 64 символів, доступна до вибору користувачем;

– дозволено використання символів ASCII (включаючи пробіл) та символів Unicode;

– перевірити передбачувані паролі зі списку, який містить значення, відомі як часто використовувані, очікувані або скомпрометовані;

– обмежити кількість послідовних невдалих спроб автентифікації для одного облікового запису не більше ніж 100;

– дозволено функцію «вставки» під час введення пароля;

– відсутність вимог до складності;

– нема терміну дії пароля;

– увімкнення багатофакторної автентифікації.

2. ISO/IEC 27002:2013 [7].

Зміна секретного коду автентифікаційної інформації постачальника за промовчанням після встановлення системи або програмного забезпечення.

Уникнення ведення запису (наприклад, на папері, у програмному файлі або на портативному пристрої) секретної автентифікаційної інформації, якщо тільки її не можна зберігати в безпечному місці та метод зберігання не затверджений (наприклад, сховище паролів).

Коли паролі використовуються як секретна автентифікаційна інформація, вибирайте якісні паролі достатньої мінімальної довжини, які:

1. легко згадати;

2. не ґрунтується на тому, що хтось інший міг би легко здогадатися або отримати, використовуючи інформацію, що стосується людини, наприклад, імена, номери телефонів, дати народження тощо;

3. невразливі для словникових атак (тобто не складаються зі слів, включених до словників);

4. без послідовних однакових, повністю цифрових або буквених символів.

Не використовувати однакову автентифікаційну інформацію для ділових та некомерційних цілей.

Забезпечити належний захист паролів, коли паролі використовуються як секретна автентифікаційна інформація в процедурах автоматичного входу в систему та зберігаються.

Системи керування паролями повинні бути інтерактивними та забезпечувати якісні паролі.

Якщо впроваджена система управління паролями, то вона повинна відповідати наступним вимогам:

1. забезпечити використання індивідуальних ідентифікаторів користувачів та паролів для забезпечення підзвітності;

2. дозволити користувачам вибирати та змінювати свої паролі та включати процедуру підтвердження, що дозволяє виключити помилки введення;

3. забезпечити вибір якісних паролів;

4. змушувати користувачів змінювати свої паролі при першому вході до системи;

5. забезпечити регулярну зміну пароля та при необхідності;

6. вести облік попередніх паролів користувачів та запобігати їх повторному використанню;

7. не відображати паролі на екрані під час введення;

8. зберігати файли паролів окремо від системних даних програм;

9. зберігати та передавати паролі у захищеному (зашифрованому або хешованому) вигляді.

3. CIS Password Policy Guide’s [8].

Використовувати MFA скрізь, де це можливо.

Потрібна мінімальна довжина 14 символів для облікових записів лише з паролем та 8 символів для облікових записів з підтримкою MFA.

Не обмежувати максимальну довжину паролів.

Дозволяти всі типи символів у паролі та вимагати принаймні один не літерний символ для облікових записів лише з паролем.

Негайно змінювати пароль у разі небезпеки.

Перевіряти створення нового пароля за внутрішнім списком заборонених відомих поганих, слабких або попередніх 5 паролів.

Встановлювати затримку зміни пароля щонайменше на один день.

Блокувати поточний сеанс через 15 хвилин (або менше) бездіяльності.

Тимчасове блокування облікового запису (15 хвилин і більше) після п’яти послідовних невдалих спроб.

Подвоєння часу регулювання (у хвилинах) між кожною повторною спробою (0, 1, 2, 4, 8 тощо) з постійним блокуванням облікового запису (потрібне скидання IT) після 12 повторних спроб.

Контролювання та оповіщення ключового персоналу, коли зазначені вище невдалі спроби входу досягають межі.

Автоматично зупиняти дію облікового запису через 45 днів без дійсного входу до системи.

Не дозволяти «підказки» до паролю при вході в систему.

1.2 Загальні поняття чат-ботів

Чат-боти – це спеціальні акаунти, за якими не закріплена будь-яка людина, а повідомлення, надіслані з них або на них, обробляються зовнішньою системою. Крім того, для користувача спілкування з ботом виглядає як звичайне листування з реальною людиною.

1.2.1 Призначення чат-ботів

Чат-бот – це розумна програма, яка живе у месенджерах та виконує різні функції.

Призначення чат-бота.

Підтримка клієнтів. Чат-бот допоможе замінити незручний FAQ на сайті, який іноді не одразу можна побачити, чи зможе відповісти на типові питання клієнта. Бот може працювати 24 години на добу та розвантажить співробітників.

Клієнтський сервіс. За допомогою чат-бота можна робити покупки та отримувати послуги. В роздрібній торгівлі, з постійним розширенням асортименту, складніше шукати конкретні товарні позиції. Після невеликого аналізу бот зрозуміє, що цікавить клієнта і відправить пряме посилання.

Маркетинг. Чат-бот – це ще один маркетинговий інструмент, який допоможе розповсюджувати контент, підтримувати лояльність клієнтів та збирати аналітику. За допомогою нього можна робити розсилки, інформувати клієнтів про акції, збирати коментарі про товари або послуги, якість обслуговування, навіть допомагати робити покупки.

Робота всередині компанії. Чат-боти допомагають оптимізувати такі процеси як: бронювання переговорів, інформування працівників про дати відпустки, розклад корпоративного транспорту, терміни зарплати та багато іншого.

Рекрутинг. Функціональність просунутих ботів – це первинний збір інформації про кандидатів. На основі співбесіди з чат-ботом менеджер з персоналу вирішує, яких кандидатів слід запросити на живу співбесіду, хто має пройти тестове завдання, а кому слід відмовити у спробі отримати роботу.

1.2.2 Класифікація чат-ботів

Існує кілька варіантів класифікації чат-ботів, але проаналізувавши їх усі, виділимо два види: бізнес-класифікація чат-бот додатків та класифікація чат-ботів за технічним типом.

Бізнес-класифікація чат-бот додатків.

Розглянемо кожен тип докладніше:

– розмовні – створені для спілкування, дуже схожі на спілкування зі звичайною людиною, які мають конкретну мету;

– асистенти – виходячи з конкретних цілей, з користувальницьких відповідей отримують необхідні дані;

– Q&A(питання-відповідь) – принцип роботи: одне питання – одна відповідь.

Технічна класифікація чат-ботів.

Розглянемо кожен тип докладніше.

Засновані на бізнес-правилах. У такому типі розмова людини та бота заздалегідь продумана розробником і має деревоподібну структуру. Завдяки великій кількості кнопок людина приходить певний шлях. Запитань з відповіддю у вільній формі у такому типі не існує.

Засновані на штучному інтелекті. Повністю відрізняються від першого типу, не мають визначеної структури. Шлях розмови визначено неявним чином на основі тестованих даних, які використовувалися для навчання моделі машинного навчання. Такі боти повинні мати великий обсяг даних для якісної роботи.

Гібридні. Цей тип чат-ботів використовує у собі взаємодію першого і другого типу, тобто розмова з користувачем ведеться по визначеному шляху, але використовується штучний інтелект для визначення намірів користувача, та вилучення даних їх листування.

1.2.3 Застосування чат-ботів у різних сферах діяльності

Розглянемо основний пул переваг, які отримає той або інший бізнес застосовуючи чат-ботів у роботі.

Конфіденційність. Соціальні мережі за своєю природою є відкритими платформами. Через них бізнесу легко транслювати свої повідомлення, спілкуватися з аудиторією, яка вже знайома з брендом, та залучати нову. Але соціальні мережі не можуть похвалитися надійним зберіганням конфіденційної інформації. Месенджери WhatsApp, Facebook Messenger,

Viber та Telegram – це канали, якими інформація передається безпосередньо одержувачу. Крім того, більшість програм обміну миттєвими повідомленнями мають додаткову наскрізну систему шифрування, що також робить цю платформу безпечною для банківського сектора.

Управління репутацією. Спілкування тет-а-тет через месенджери набагато краще. По перше, проблеми ваших клієнтів не стануть предметом обговорення для всієї аудиторії Facebook та ЗМІ. В особистому листуванні можна виконувати запит більш якісно та ймовірність, що клієнт залишиться задоволеним зростає. По-друге, при листуванні в месенджері можна виявити як позитивні якості компанії, так і негативні.

Зручне середовище для зв’язку. В основі месенджерів – асинхронність, швидкість, а головне всі діалоги в одному місці. Поєднання всіх цих властивостей робить їх основним цифровим засобом комунікації. Чат-боти Telegram збільшують продажі. Завдяки їм у короткий термін можна отримати цільову аудиторію. При правильному використанні бота він допоможе у продажах та надасть необхідну інформацію покупцеві. Як показує практика, використання чат-бота для підтримки клієнтів може зменшити до 40% часу консультантів в онлайн-чатах. Крім того, до половини звернень до робота відбувається у неробочий час. На відміну від живої людини, бот здатний одночасно давати відповіді кільком користувачам. Більше того, відповідь надається миттєво. Великою перевагою чат-ботів є кросплатформність. Готового робота не складно адаптувати до інших платформ.

Виходячи з перерахованого та розглянутого вище можна зробити висновок, що популярність чат-ботів набирає обертів та компанії, що прагнуть до збільшення ефективності свого бізнесу, все частіше замовляють розробникам нових чат-ботів.

1.3 Вибір програмного забезпечення для розробки чат-ботів

1.3.1 Вибір мови програмування

Python, по суті, є швейцарським армійським ножем для кодування завдяки своїй універсальності. Крім того, це одна з найпростіших мов для початківців, завдяки її послідовному синтаксису та мові, яка відображає людську мову [9].

Це означає, що коли Python був вперше випущений, він застосовувався до більш різноманітних випадків, ніж інші мови, такі як Ruby, який був обмежений веб-дизайном та розробкою. Тим часом Python розширився в області наукових обчислень, що спонукало до створення широкого спектру бібліотек з відкритим кодом, які отримали вигоду від років досліджень і розробок.

Найбільший недолік Python полягає в його документації, яка скудна в порівнянні з іншими усталеними мовами, такими як PHP, Java і C++. Пошук відповідей у Python схожий на пошук певного уривка в книзі, яку ви ніколи не читали. Крім того, у мові гостро не вистачає корисних і простих прикладів. Ясність також є проблемою, яка надзвичайно важлива під час створення чат-бота, оскільки навіть найменша двозначність на одному з кроків може призвести до його збою.

Якщо швидкість є головною проблемою при створенні чат-бота то Python може не вистачити в порівнянні з Java і C++. Однак виникає питання, коли насправді має значення час виконання коду? Більше значення має досвід кінцевого користувача, і вибір швидшої, але більш обмеженої мови для створення чат-ботів, як-от C++, є безпрограшним. З цієї причини жертвувати часом розробки та можливостями для бота, який міг би працювати на кілька мілісекунд швидше, не має сенсу.

Підбиваючи підсумки, приходимо до висновку, що для створення чат-боту мови Python буде достатньо та вона відповідає усім потребам розробки.

1.3.2 Огляд засобів для розгортання чат-ботів на сервері/локально

Розглянемо розгортання чат-боту на локальному сервері та за допомогою Docker.

Docker — це інструмент з відкритим вихідним кодом, який автоматизує розгортання програми всередині програмного контейнера. Найпростіший спосіб зрозуміти ідею Docker – це порівняти його зі стандартними контейнерами [10].

Коли розроблюємо програму, частіш за все потрібно надати свій код разом з усіма можливими залежностями, такими як бібліотеки, веб-сервер, бази даних тощо. Тому можемо опинитися в ситуації, коли програма працює на комп’ютері локально, але навіть не запуститься на проміжному сервері, або на машині розробника чи QA.

Цю проблему можна вирішити, ізолюючи додаток, щоб зробити його незалежним від системи.

Традиційно, щоб уникнути такої несподіваної поведінки використовувалися віртуальні машини. Основна проблема віртуальної машини полягає в тому, що «додаткова ОС» поверх операційної системи-хоста додає гігабайти місця до проекту. У більшості випадків на сервері буде розміщено кілька віртуальних машин, які будуть займати ще більше місця. І, до речі, на даний момент більшість постачальників хмарних серверів стягують з плату за додаткове місце. Ще один істотний недолік ВМ – повільне завантаження.

Docker усуває все вищезазначене, просто поширюючи ядро ОС у всіх контейнерах, що працюють як окремі процеси хост-ОС.

1.4 Огляд існуючих чат-ботів менеджерів паролів

Принципового поняття «менеджеру» паролів у Telegram ботів не існує. Вони не розгортаються, як професійне програмне забезпечення, не мають якогось специфічного інтерфейсу, та майже всі мають приблизно однаковий функціонал. Але ж такі деякі від’ємності маються.

Розглянемо два приклади: Random Password Generator та EasyStrongPasswordBot.

Random Password Generator. Має лише стандартний функціонал, який починається зі «старту» та однієї відповіді (рис. 1.1). Генерує пароль на 16 символів та на цьому все. Не має можливості більше ніякої взаємодії.

EasyStrongPasswordBot. Має також початковий «старт», але є можливість взаємодії: обрання мови та принцип генерування паролю (рис. 1.2). На цьому також все.

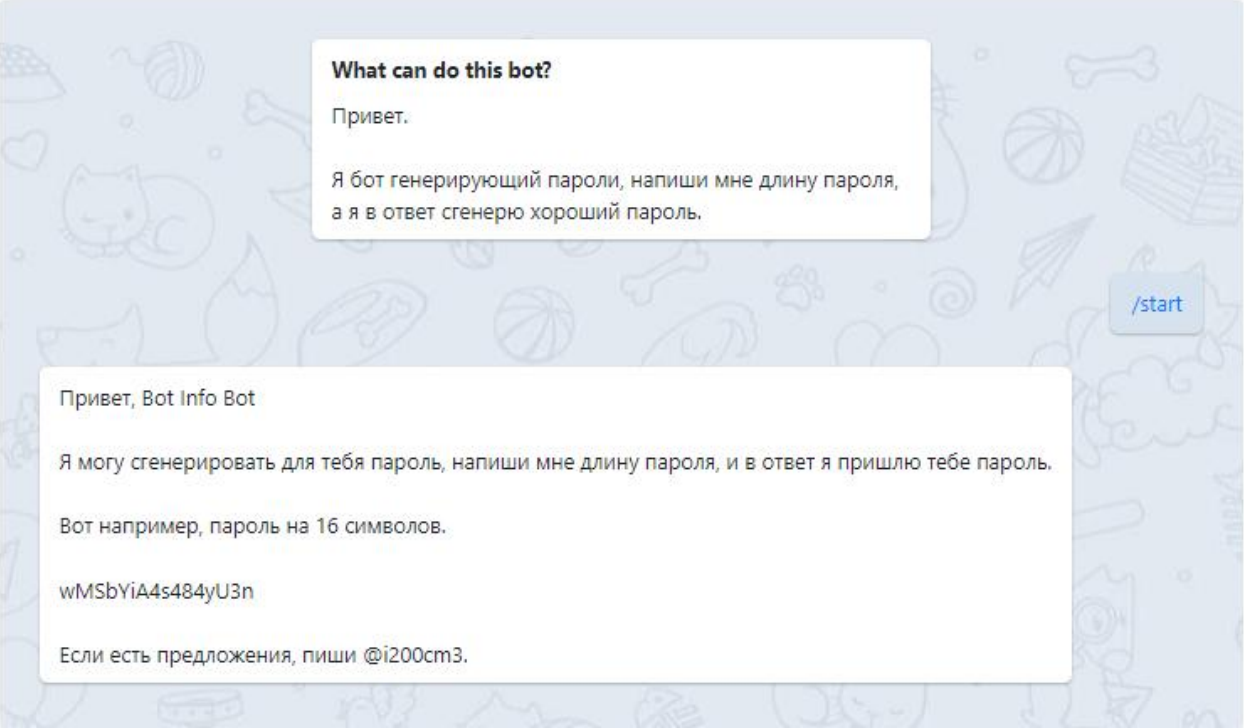

Рисунок 1.1 – Чат-бот «Random Password Generator»

Джерело: [10]

1.5 Постанова задачі на розробку менеджеру паролів з підвищеним захистом на основі телеграм боту

1.5.1 Об’єкт та предмет розробки

Об’єкт дослідження – процеси та процедури захисту інформації в соціальних мережах

Предмет дослідження – менеджер паролів для месенджеру з підвищеним захистом на основі телеграм боту.

Мета роботи – розробка дизайну і функціоналу менеджеру паролів з підвищеним захистом на основі телеграм боту.

Методи дослідження – аналіз додатків існуючої проблеми менеджерів паролів, виявлення переваг і недоліків; визначення вимог до структури, поведінки і функціоналу чат-боту; вибір інструментальних засобів розробки.

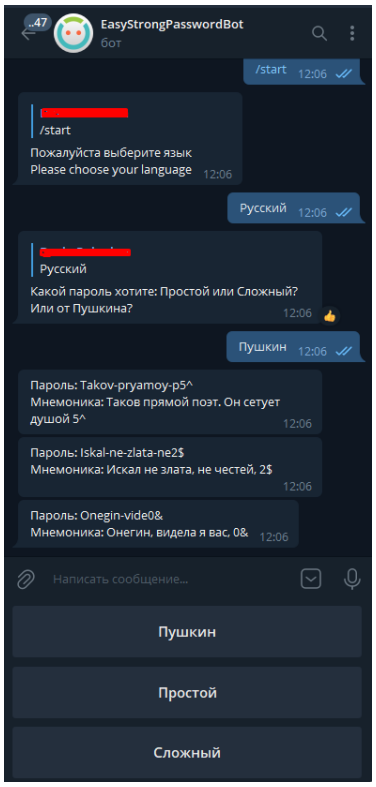

Рисунок 1.2 – Чат-бот «EasyStrongPasswordBot»

Джерело: [10]

1.5.2 Вимоги до інтерфейсу чат-боту

Інтерфейс користувача в додатку має бути діалоговим, при запуску виводиться привітання. Та після старту повинна бути інструкція, що вміє чат-бот та опис того як його використовувати. Слідкуючи за інструкцією користувач за допомогою додаткових кнопок або пояснень повинен дійти до своєї мети.

1.5.3 Вимоги до засобів реалізації чат-боту

Для засобів реалізації висунуті наступні вимоги:

– Telegram API;
– мова Python для кодування;
– PyCharm Community Edition (або Visual Code);
– бібліотеки для роботи з чат-ботами - pyTelegramBotAPI;
– мобільний пристрій для тестування.

1.5.4 Вимоги до функціоналу чат-боту

Менеджер паролів повинен мати наступний функціонал:

- генерація паролів різної складності;
- виведення паролів та можливість перегенерування;
- знищення паролю за згодою користувача;
- меню для взаємодії з користувачем.

1.6 Висновки за розділом

Під час написання першого розділу було розглянуто методи захисту даних в месенджерах, розглянуто вимоги до парольного захисту та складності. Визначені вимоги до менеджеру паролів з підвищеним захистом та поставлено завдання розробки. Обрано засоби для реалізації, а також розглянуто існуючи аналоги чат-ботів розроблених тими ж самими засобами.

2 ПРОЕКТУВАННЯ МЕНЕДЖЕРУ ПАРОЛІВ З ПІДВИЩЕНИМ ЗАХИСТОМ НА ОСНОВІ ТЕЛЕГРАМ БОТУ

2.1 Проєктування моделі чат-боту

2.1.1 Обґрунтування проєктування чат-боту на базі Telegram

Дослідження виявили що паролі які втекли у відкритий доступ, були найпростішими та найпоширенішими типу – «123456», він зустрічається приблизно в 0,722% випадків, далі йдуть – «123456789», «password», «qwerty», «12345678». Усього 8,83% із загальної бази паролів є унікальними, решта зустрічаються два і більше разів; середня довжина пароля становить 9,4822 символи; лише 12,04% паролів містять спеціальні символи; 8,79% паролів містять лише літери; 26,16% паролів містять символи лише у нижньому регістрі; 13,37% паролів містять лише цифри; 34,41% усіх паролів закінчуються цифрами, але лише 4,522% паролів починаються з цифр. З усього вищесказаного можна виділити, що тільки 9 відсотків створюють унікальні паролі. Інші ж нехтують цим правилом, оскільки запам'ятати багато паролів людина не зможе. Оскільки людина запам'ятати всі паролі не може, їй треба їх кудись записати, щоб не забути. Можна по-старому записати їх на папірець, а можна скористатися спеціальним додатком - менеджером паролів.

Менеджер паролів на базі Telegram. На сьогоднішній день Telegram мають більше 500 млн користувачів, тобто окремо не потрібно встановлювати програму, яка займає пам'ять пристрою. Telegram перевірена та надійна програма на відміну від інших виробників програмного забезпечення у маркетплейсах. При вході з іншого пристрою на обліковий запис потрібно пройти двоетапну автентифікацію, тобто крім повідомлення, яке приходить на головний пристрій необхідно ввести пароль який встановлює власник облікового запису. Також при вході з іншого пристрою на основний пристрій прийде повідомлення про те, що в аккаунт увійшли і

його місцезнаходження та айпі. При цьому є можливість з основного пристрою завершити інші сеанси. Якщо ж зловмисник захоче потрапити в Telegram саме з головного пристрою (украде його), то йому спершу треба буде, ввести пароль або за відбитком пальця зайти в сам смартфон. Крім того, як на вхід у Telegram додаток системно може бути встановлено пароль, так і в самому додатку може бути включено пароль навіть при вході з головного пристрою (див. рис. 2.1).

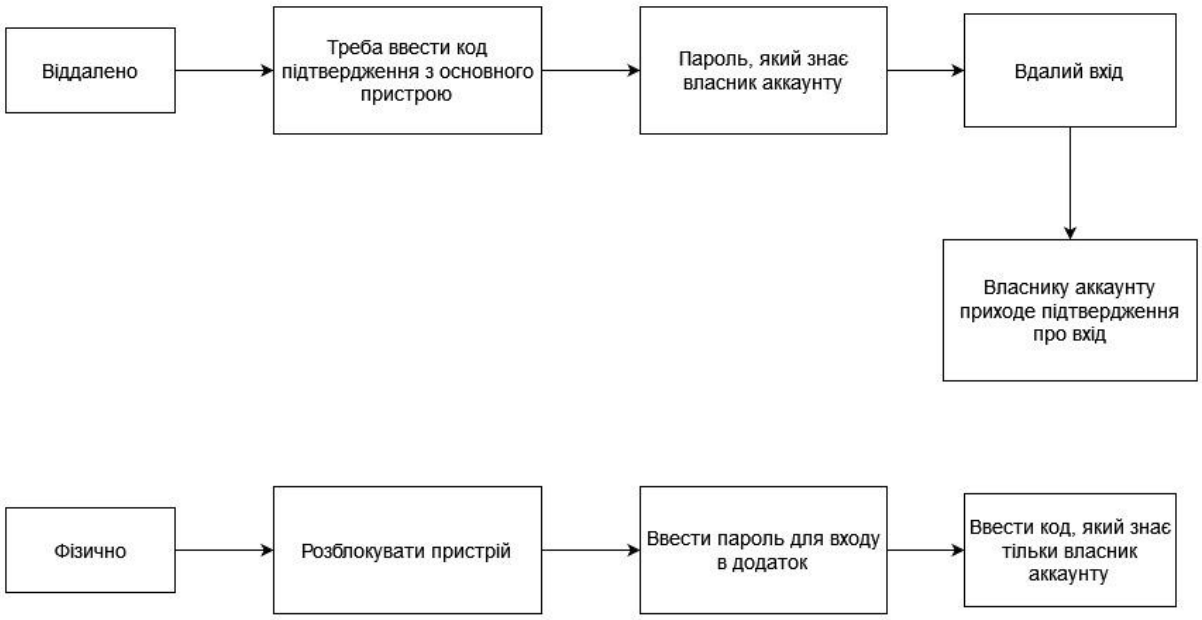

Рисунок 2.1 – Схема входу у Telegram

Джерело: авторська розробка

2.1.2 Модель взаємодії користувача з чат-ботом

Програма складається з веб-клієнта та чат-бота Telegram на Python. Основні функції програми:

– обробка повідомлень клієнтів у режимі реального часу;

– надання можливих варіантів відповідей;

– обробка рішення бота в режимі реального часу та надсилання відповіді клієнту.

Веб-клієнт складається з інтерфейсу для роботи діалогового графа та клавіатури для надсилання повідомлень. Клієнтська взаємодія з ботом – класичний варіант роботи з ботом Telegram. Запуск програми також запускає два потоки. Один із них – забезпечення роботи Telegram-бота, а другий – роботу програми. Обробка повідомлень клієнта здійснюється за допомогою інструментів «Обробник повідомлень» бібліотеки TeleBot.

2.1.3 Розробка діаграми варіантів використання (Use-case)

Діаграма варіантів використання UML – це основна форма вимог до системи/програмного забезпечення при розробці нової, недостатньо розробленої програми. Варіанти використання визначають очікувану поведінку та приблизний спосіб її реалізації. Коли варіанти використання визначені, їх можна позначити як текстовим, так і візуальним уявленням (тобто діаграмою варіантів використання). Ключовою концепцією моделювання діаграми варіантів використання є те, що вона допомагає проектувати систему з погляду кінцевого користувача. Це ефективна техніка для повідомлення про поведінку системи з погляду користувача шляхом вказівки всієї поведінки системи, видимої ззовні [11].

Діаграма варіантів використання зазвичай проста. Вона не показує деталі варіантів використання:

– вона лише узагальнює деякі відносини між варіантами використання, дійовими особами та системами;

– вона показує порядок, у якому виконуються кроки задля досягнення цілей кожного варіанта використання.

Діаграма взаємодії показана на рисунку 2.2.

Рисунок 2.2 – Діаграма варіантів використання

Джерело: авторська розробка

2.1.4 Розробка діаграми станів

Діаграма станів складається із станів, переходів, подій та дій. Діаграми станів використовуються для ілюстрації динамічного уявлення системи. Вони особливо важливі при моделюванні поведінки інтерфейсу, класу чи спільної роботи. Діаграми станів наголошують на поведінці об’єкта залежно від подій, що особливо корисно при моделюванні реактивних систем [11].

Використовується для обліку стану об’єкта.

Наприклад, станів користувачів:

- новий;
- активний;
- неактивний.

Діаграма станів надає можливості:

- обліку станів об’єктів;
- зберігання даних про об’єкти у вузлах;
- немає обмежень на кількість заявок у процесі.

На початку графіка розташовано питання користувача та можливі варіанти відповіді. При натисканні на одну з відповідей текст автоматично відображається в просторі відповіді.

Потім система чекає на відповідь клієнта на відправлене повідомлення. При отриманні граф реорганізується шляхом додавання відповіді користувача у гілку з вибраними повідомленнями. Нові відповіді готуються програмою (див. рис. 2.3).

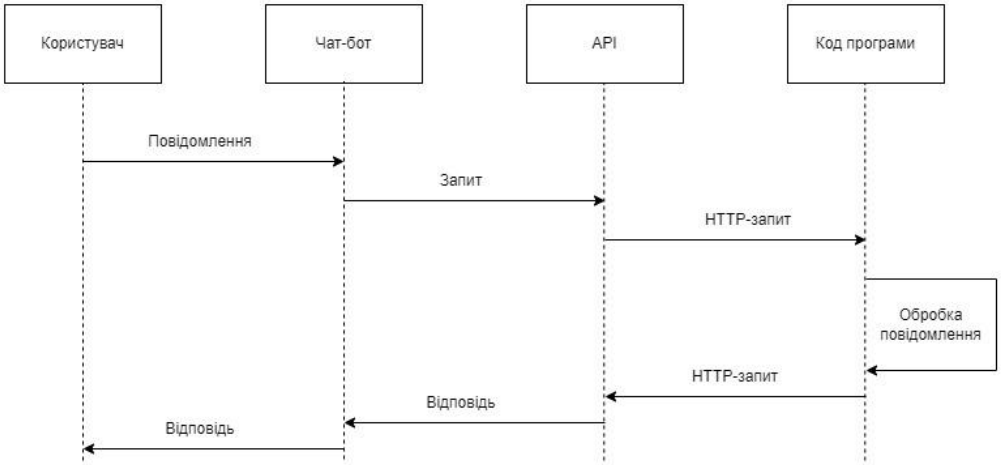

Рисунок 2.3 – Діаграма станів

Джерело: авторська розробка

2.2 Робота з сервером

2.2.1 Терміни та основні поняття

Для компактної та зручної роботи будемо використовувати Docker. Розглянемо основні терміни та поняття для роботи з ним:

Контейнер – поточний екземпляр, який інкапсулює необхідне ПЗ. Контейнери створюються з образів. Можуть відкривати порти та томи для взаємодії з іншими контейнерами або зовнішнім світом, а також легко видаляються та перетворюються заново за короткий проміжок часу.

Образ – це основний елемент для кожного контейнера. Після створення кожного кроку кешується і може бути використаний повторно (модель

копіювання при записі). Залежно від образу може знадобитися деякий час для його побудови. З них можуть бути одразу запущені контейнери.

Порт – TCP/UDP-порт у своєму звичайному поданні. Порти можуть бути відкриті для зовнішнього світу (доступи через основну ОС) або приєднані до інших контейнерів - доступні тільки з контейнерів і невидимі для зовнішнього світу.

Том може бути описаний як спільна папка. Тому ініціалізуються при створенні контейнера. Вони спроектовані для зберігання даних, незалежно від життєвого циклу контейнера.

Реєстр – сервер, який зберігає образи Docker. Функціонує за аналогією з Github – можна витягнути образ, щоб розгорнути його локально, а потім закинути у реєстр.

Docker Hub – реєстр із веб-інтерфейсом, створений Docker Inc. Він зберігає велику кількість Docker-образів із різним ПЗ. Docker Hub є джерелом офіційних Docker-образів, створених командою Docker. Офіційні образи містять списки своїх потенційних уразливостей. Ця інформація доступна для кожного авторизованого користувача. Доступні два типи облікових записів: безкоштовні та платні. На безкоштовному обліковому записі може бути один приватний образ і безліч загальнодоступних образів [12].

2.2.2 Представлення даних при розміщенні локально

Усі данні представляються та компонуються згідно з системними налаштуваннями, та не враховують компактність зберігання та мобільність розгортання на інших машинах.

Представлення даних при розміщенні локально (див. рис. 2.4):

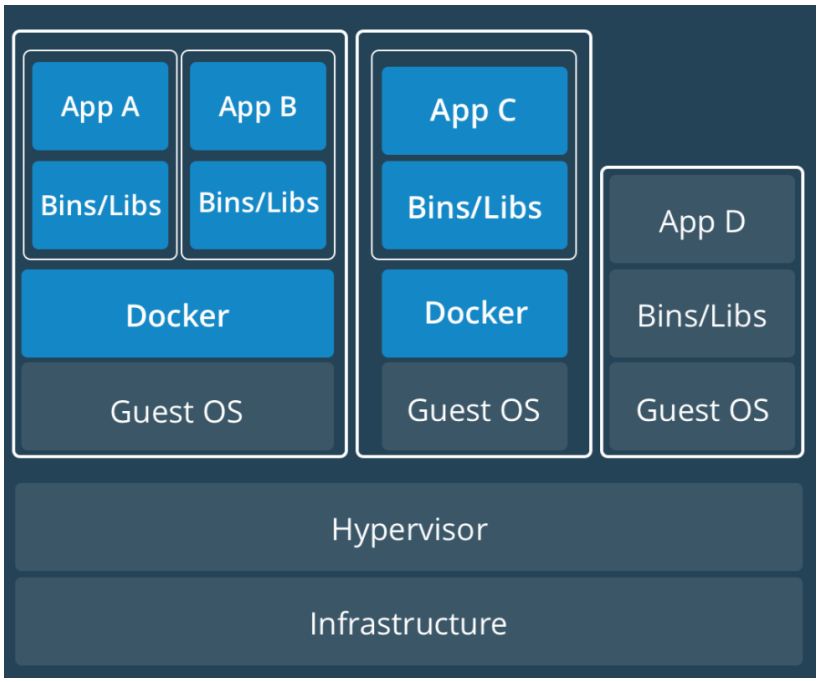

Рисунок 2.4 – Представлення локально даних у системі

Джерело: [12]

2.2.3 Представлення даних при розміщення засобами Docker

Docker використовує клієнт-серверну архітектуру. Клієнт Docker взаємодіє з «демоном Docker», який виконує всю важку роботу щодо участі, запуску та розповсюдження існуючих контейнерів Docker. Клієнт Docker та «демон» можуть працювати в одній системі або є можливість підключити клієнт Docker до віддаленого демона Docker. Клієнт Docker та демон взаємодіють за допомогою REST API, через сокети UNIX або мережевий інтерфейс. Також є ще один клієнт Docker - Docker Compose, який дозволяє працювати з одними програмами, що складаються з набору контейнерів (див. рис. 2.5).

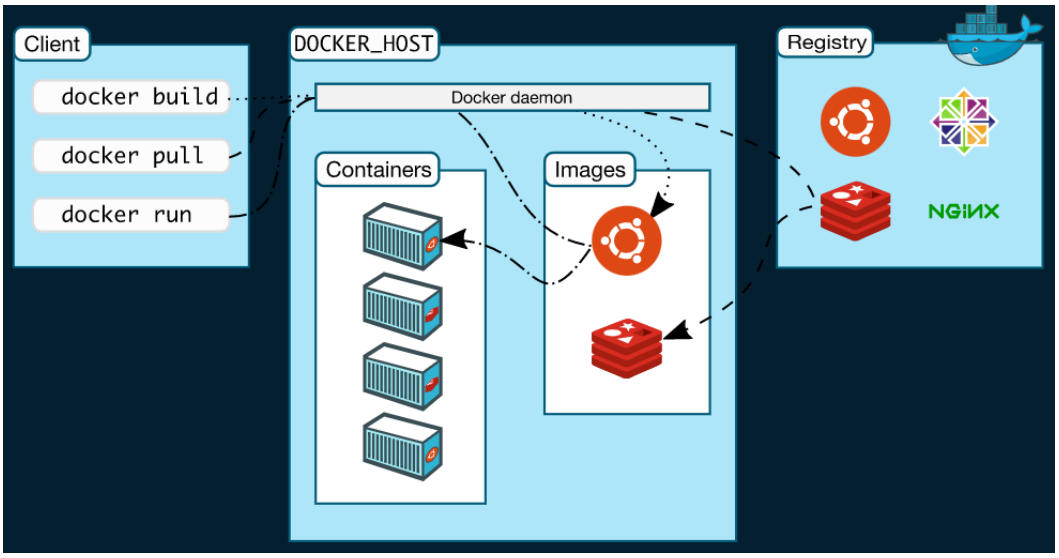

Рисунок 2.5 – Представлення даних засобами Docker

Джерело: [12]

2.3 Користувальницький графічний інтерфейс (GUI)

2.3.1 Основні принципи роботи GUI у Telegram

Клавіатури.

Одна з незвичайних можливостей Bot API – кастомізовані клавіатури. При передачі сервером відповіді можна передати команду на відображення спеціальної клавіатури з встановленими варіантами відповіді. Клієнту Telegram, отримавши повідомлення, відобразиться кастомна клавіатура. Натискання клавіші відразу відправить на сервер відповідну команду. Таким чином, можна значно спростити взаємодію робота з користувачем. На даний момент для відображення на клавіші можуть використовуватися емодзі та текст. Ось кілька прикладів таких клавіатур:

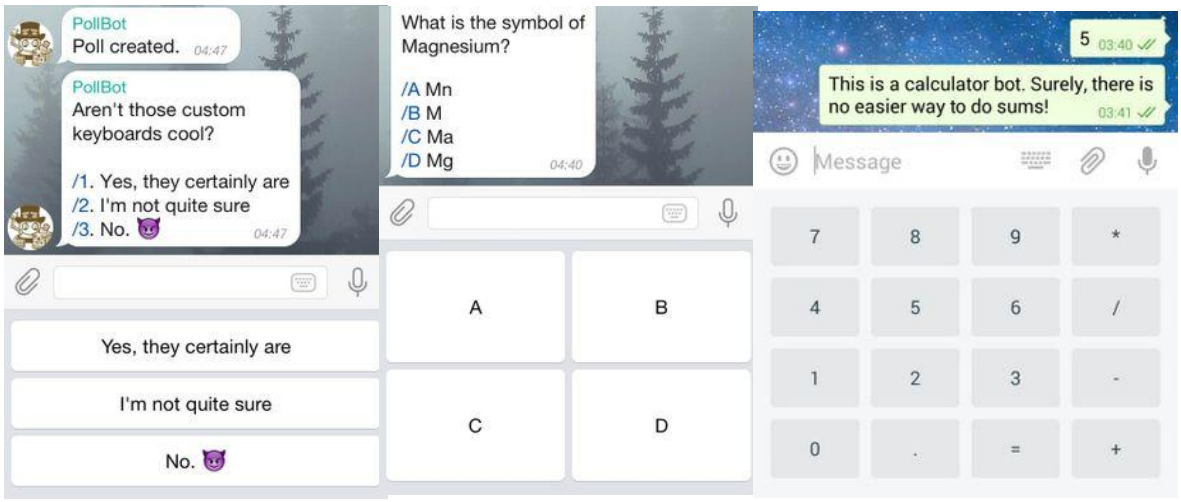

Рисунок 2.6 – Приклад відображення кастомних клавіатур

Джерело: авторська розробка

Команди.

Команди є більш гнучким способом спілкування з ботом. Рекомендується наступний синтаксис:

/команда [необов’язковий] [аргумент]

Команда повинна починатися з символу косої межі «/» і не може бути довшою за 32 символи. Команди можуть складатися з букв латинського алфавіту, цифр та підкреслення. Декілька прикладів:

/get_messages_stats

/set_timer 10min Alarm!

/get_timezone London, UK

Повідомлення, що починаються з косою риси, завжди доставлятимуться боту (так само, як і при відповіді на його повідомлення і на згадки бота в чаті) (див. рис. 2.7).

Пропонувати список команд, що підтримуються, з їх описом, коли користувач введе символ косої риси «/» (щоб цей пункт працював, необхідно задати опис команд у @BotFather). Натискання на опис призведе до надсилання цієї команди.

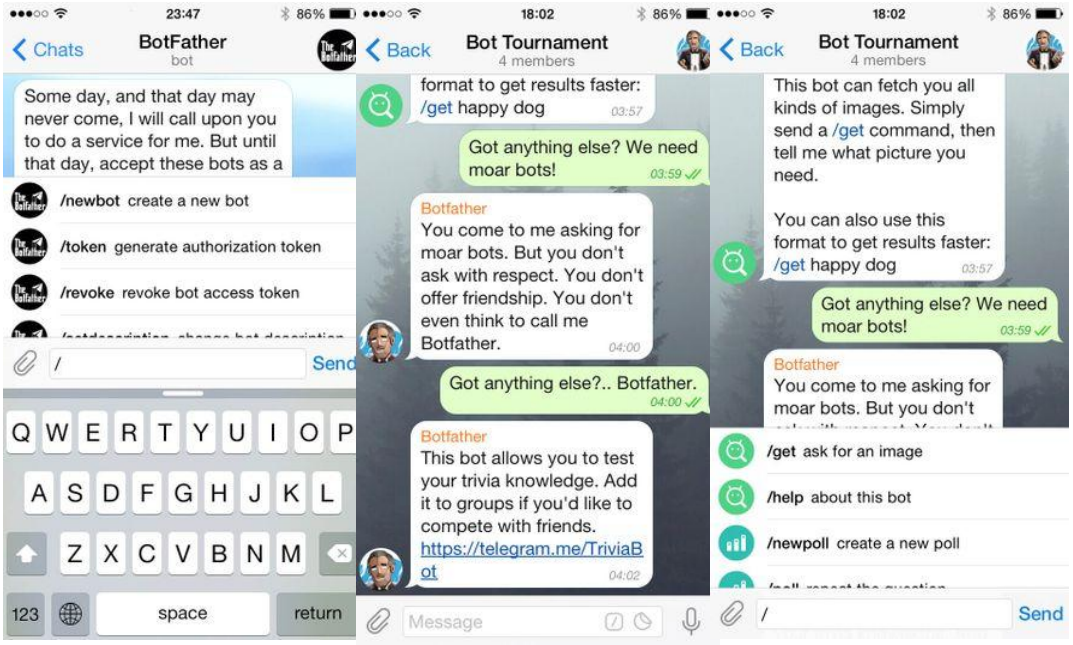

Рисунок 2.7 – Приклад відображення команд

Джерело: авторська розробка

Показувати кнопку (/) у полі введення тексту у всіх чатах із ботами. Натискання цієї кнопки відображає список доступних команд.

Підсвічувати / команди в повідомленнях. При натисканні на таку команду, що підсвічується, вона буде відразу ж відправлена боту.

Якщо у групі є кілька ботів, можна дописати після команди ім’я бота, щоб уникнути колізій у загальних командах:

/start@TriviaBot

/start@ApocalypseBot

Це відбувається автоматично, якщо ви вибираєте команду зі списку доступних.

Глобальні команди.

Щоб користувачам було простіше працювати з ботами, всі розробники реалізовують підтримку кількох простих команд. В інтерфейсі програм Telegram будуть ярлики (швидкі посилання) для цих команд.

/start – починає спілкування з користувачем (наприклад, надсилає вітальне повідомлення). До цієї команди також можна передавати додаткові аргументи.

/help — відображає повідомлення за допомогою команди. Воно може бути коротке повідомленням про бот і список доступних команд.

/settings — (по можливості) повертає список можливих налаштувань та команди для їхньої зміни.

При спробі розпочати спілкування з роботом користувач побачить кнопку СТАРТ. На сторінці профілю бота також будуть доступні посилання Допомога та Налаштування.

2.3.2 Відображення кнопок та меню

Для введення інформації в інтерактивні чат-боти можна використовувати як текст, так і посилання або кнопки. Вони мають різні призначення.

Завдяки посиланням та кнопкам користувачам не потрібно вводити текст. Вони, як правило, будуть представляться у вигляді каруселі, за допомогою зображень можна зробити ще більш зручний та інтуїтивний інтерфейс. Багато користувачів у процесі використання оцінили наявність цієї опції, як зручної.

За допомогою введення тексту користувачі можуть вибрати тип запитань, які вони хотіли поставити, та вийти за рамки (часто надто суворого) сценарію чат-бота.

Окрім звичайних кнопок та посилань, деякі чат-боти також можуть мати елемент меню, який при натисканні відображає набір можливих варіантів. Кнопка для виклику меню знаходилася під полем введення тексту або у вигляді маленької іконки «гамбургера» поряд з ним.

Взаємодії, підтримувані ботами, найкраще можна описати як лінійні потоки з обмеженою кількістю відгалужень, які залежать від відповідей користувача. Робот ставить питання, відповіді служать орієнтиром для просування робота по необхідної гілки потоку.

Приклад лінійного потоку процесу вибору, через який бот проводить користувача: «Що ви хочете замовити?» (піцу) → «Тісто?» (тонке) → «Розмір» (маленький) → «Начинка?».

Якщо користувач не виходить за рамки заданого потоку і його відповіді відповідають очікуванням системи, вони послідовні і не містять невідомих слів, то жодних проблем у взаємодії з чат-ботами не виникає.

Обидва ці механізми можна використовувати по-своєму, вони повинні бути присутніми в добре спроектованому чат-боті.

2.3.3 Робота з даними та можливості їх зберігання

При створенні чат-бота важливо не просто запустити чат-бота, а й спланувати всі очікування клієнтів/користувачів від взаємодії з чат-ботом. Важливо на початковому етапі продумати як зберігатимуться дані і чи є можливість пост-взаємодії з ними.

Перед запуском чат-бота, після аналізу існуючих, з’явилося уявлення про широту типів навичок та функцій, які потрібні чат-боту. Це означає, що в архітектурі даних при взаємодії є специфічні функції для початкових навичок, якими володітиме чат-бот, які також будуть об’єднані із загальною інформацією, яка матиме певні залежності при діалозі з ботом (наприклад, введення та виведення). Це означає, що будь-яка нова навичка, для якої не потрібно певне поле в JSON, буде порожньою, таким чином збільшуючи повідомлення JSON або створення нового повідомлення.

Це створює певні проблеми при розширенні функціоналу чат-бота.

Компроміс у тому, щоб структурувати дані (SQL) першому етапі.

Можна виймати та надсилати дані чат-бота за допомогою API. Три основні дії, які потребуються архітектурі чат-бота – це можливість вилучати дані або введення користувача, ідентифікувати та налаштовувати сховище бази даних, відправляти дані в базу даних (POST) та візуалізувати дані.

Користувальницькі вхідні дані та розмови з чат-ботом повинні бути вилучені та збережені у базі даних. Вхідні дані користувача зазвичай є висловлюваннями, надані користувачем у розмові з чат-ботом.

На основі API-інтерфейсів буде доступна інформація з бази даних для потокової передачі туди і назад на платформу чат-бота.

Потім дані можна отримати за допомогою візуалізатора даних, такого як Tableau, PowerBI або QuickSight, щоб краще проаналізувати взаємодію з клієнтом для покращення моделювання даних, відповідей ботів та загальної інформації, яка може допомогти боту.

Розмови чат-бота зберігатимемо у базі даних SQL, розміщеної на хмарній платформі.

На підставі такої схеми зберігання даних згодом є можливість аналізувати ефективність чат-бота, а згодом поліпшення та швидка модернізація, розширення можливостей робота.

2.4 Висновки за розділом

Під час написання другого розділу було створено модель взаємодії користувача з чат-ботом. Все це було спроєктовано на основі діаграм варіантів використання та діаграми станів. Проаналізовано та обрано модель роботи з серверною частиною. Приведено схеми розміщення та зберігання даних на сервері. Розглянуто варіанти створення користувальницького інтерфейсу, можливості які представляються функціоналом Telegram. Визначено схему зберігання даних, що накопичуються у діалогах з чат-ботом.

3 РОЗРОБКА МЕНЕДЖЕРУ ПАРОЛІВ З ПІДВИЩЕНИМ ЗАХИСТОМ НА ОСНОВІ ТЕЛЕГРАМ БОТУ

1.1 Структура проекту

3.1.1 Середовище розробки PyCharm – встановлення та налаштування

PyCharm від JetBrains — це повне інтегроване середовище розробки, яке включає високоавтоматизований ланцюг інструментів для підвищення продуктивності розробників. Рішення побудовано на основі концепції «розумного коду» та інтегрує функції, які автоматично перевіряють код, відзначають помилки та допомагають розробникам вносити зміни, якщо це необхідно.

Як випливає з назви, PyCharm IDE націлена на програмістів Python. Він містить профіль Python, інтегрований налагоджувач і тестовий запуск, який допомагає виконувати тести на основі графічного інтерфейсу користувача. Вбудований термінал і термінал SSH дозволяють підключатися до будь-якої віддаленої машини. Нарешті, IDE також містить віддалений інтерпретатор для налагодження та профілювання в тестових середовищах, навіть якщо це середовище розташоване на віртуалізованому сервері, створеному за допомогою контейнерів Docker або портативного програмного забезпечення Vagrant.

Усі ці функції, очевидно, роблять PyCharm корисним у задачах веб-розробки, але читачі Embedded Computing Design будуть раді знати, що IDE також включає низку підтримки для наукових інструментів, таких як:

- Panda;
- NumPy;
- Бібліотеки Matplotlib.

У цих вбудованих випадках використання підтримка інтеграції Conda допомагає зберегти залежності ізольованими, а перевірка синтаксису на

льоту з перевірками, збігом дужок і лапок, а також завершенням коду забезпечує швидке програмування в інтерактивній консолі Python.

Найкраще те, що PyCharm побудований на IntelliJ і має повністю відкритий вихідний код. Почнемо встановлювати Python та середовище розробки PyCharm.

Установка Python 3

Послідовність установки:

1. Переходимо на офіційний сайт Python і вибираємо потрібну нам версію 3.6.4: для 32-розрядних або версію для 64-розрядних систем (залежить від твого комп'ютера; дізнатися про розрядність можна, відкривши Провідник — Цей комп'ютер — і за натисканням на праву кнопку миші) вибрати пункт Властивості).

2. Запускаємо установник та проходимо по всій послідовності установки.

3. На першому етапі установки необхідно відзначити Add Python 3.6 to PATH, це спростить роботу. Потім натискаємо на Install now і чекаємо на закінчення установки.

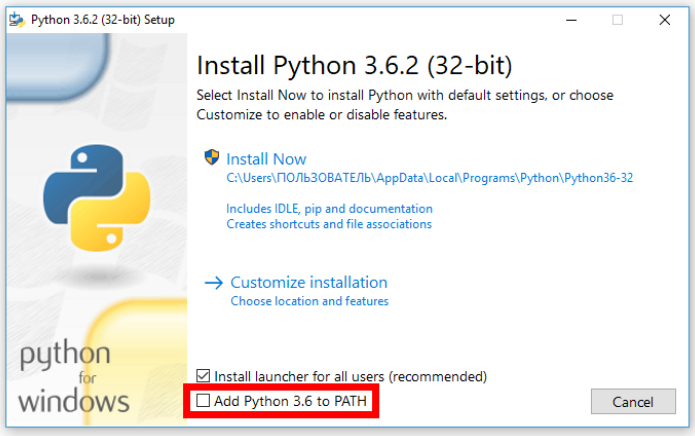

Рисунок 3.1 – Установник Python 3

Джерело: авторська розробка

Щоб перевірити, чи правильно встановився Python, відкриваємо меню Пуск і в рядку пошуку набираємо командний рядок (у Windows 10 відкрити пошук можна просто натиснувши на лупу поруч із кнопкою Пуск) або Win+R.

У вікні командного рядка виконуємо команду python. У відповідь має відкритися консоль Python (починається з >>>) як на малюнку нижче.

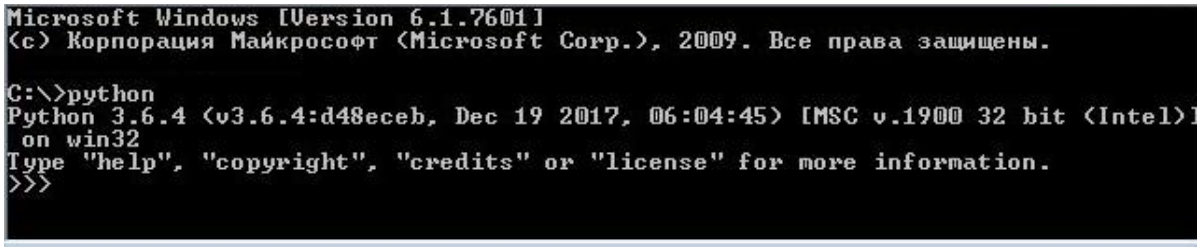

Рисунок 3.2 – Відгук на команду python

Джерело: авторська розробка

Виконуємо команду exit(), щоб вийти з інтерпретатора Python. Менеджер пакетів pip також повинен бути встановлений, щоб перевірити, чи виконає команду pip -V. У відповідь у консолі має відобразитись версія pip:

```
Pip 9.0.1 від c:\users\hpstr\appdata\local\programs\python\python36\li
b\site-packages (python 3.6)
```

Повертаємось у термінал (або в командний рядок) і виконуємо наступну команду:

```
pip install pyTelegramBotAPI
```

Звертаючись до pip та встановлюємо (install) модуль pyTelegramBotAPI, який відповідає за створення та роботу роботів.

Якщо модуль успішно встановився, ми повинні побачити напис Successfully installed pyTelegramBotAPI-x.x.x. Власне кажучи, існує багато

інших модулів для Python (і не тільки), за допомогою яких можна швидко запустити Telegram-бота; простими словами pyTelegramBotAPI - це такий «конструктор», який містить всі необхідні компоненти.

Тепер перейдемо до установки PyCharm.

Переходимо на сторінку завантаження PyCharm. Для завантаження доступно дві версії: професійна та версія для спільноти. Версія для спільноти безкоштовна. Її і скачаємо.

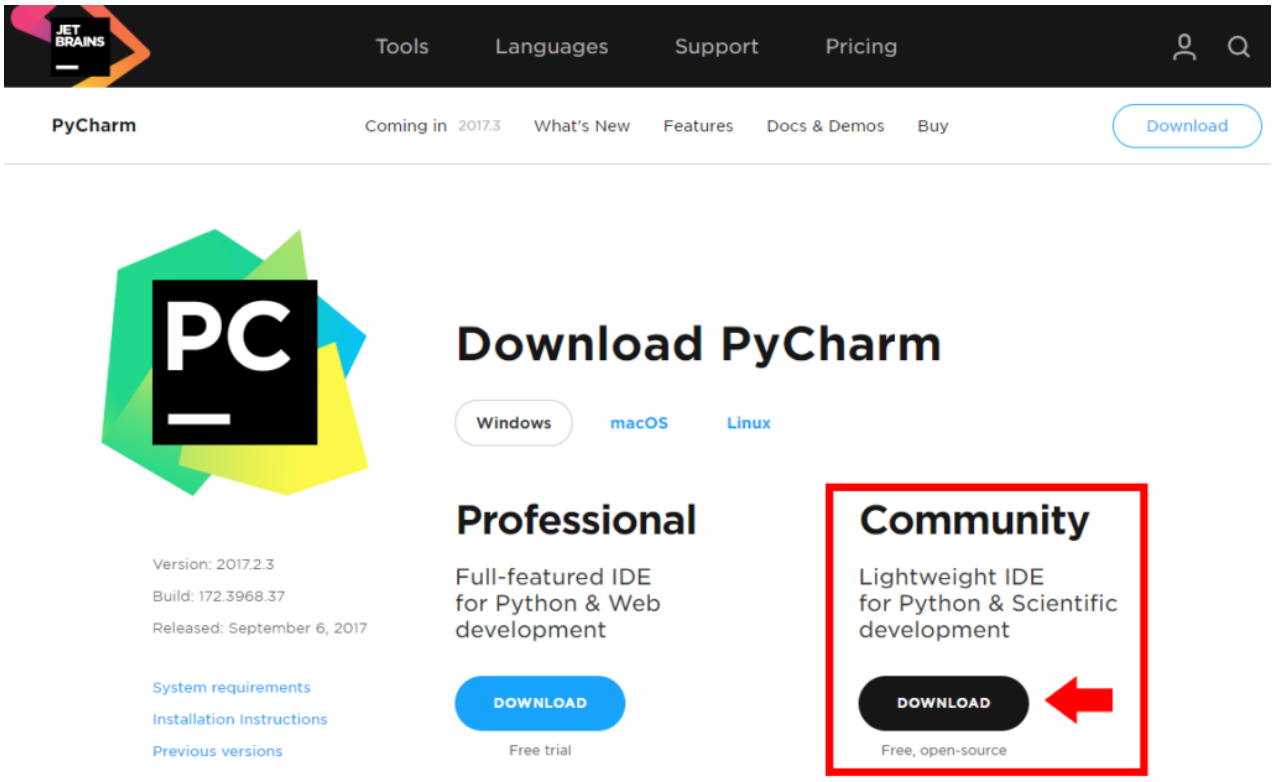

Рисунок 3.3 – Установник PyCharm

Джерело: авторська розробка

Запускаємо завантажений .exe файл. У першому вікні нас вітає сам установник, клацаємо «Next». Далі треба вказати місце встановлення середовища.

На наступному вікні обов’язково виділити галочку «Download and install JRE x86 by JetBrains».

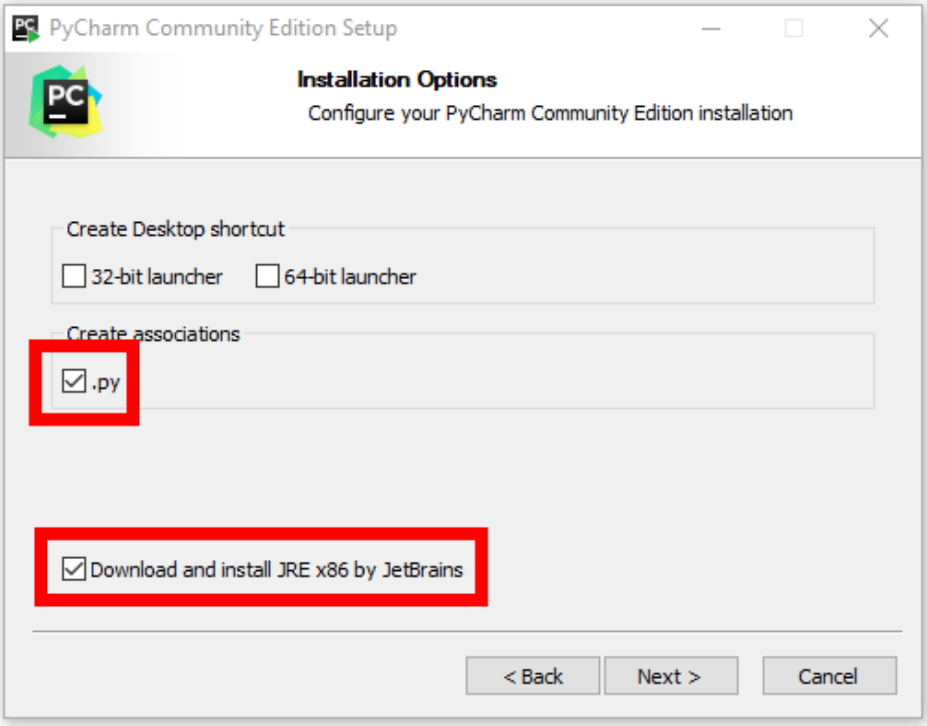

Рисунок 3.4 – Налаштування установки

Джерело: авторська розробка

Після закінчення установки запускаємо PyCharm. Створюємо новий проект, клікнувши на Create New Project.

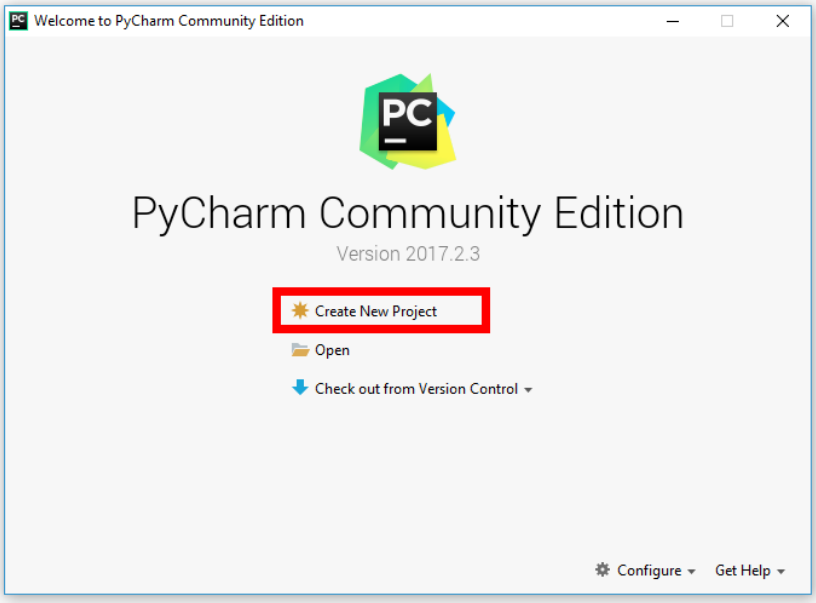

Рисунок 3.5 – Вікно старту PyCharm

Джерело: авторська розробка

Наступним етапом зареєструємо робота.

Створення будь-якого бота починається повідомлення батькові ботів у телеграмі - @BotFather.

Він може керувати всіма існуючими ботами, за допомогою багатьох команд. Їх список будь-якої миті можна викликати командою /help.

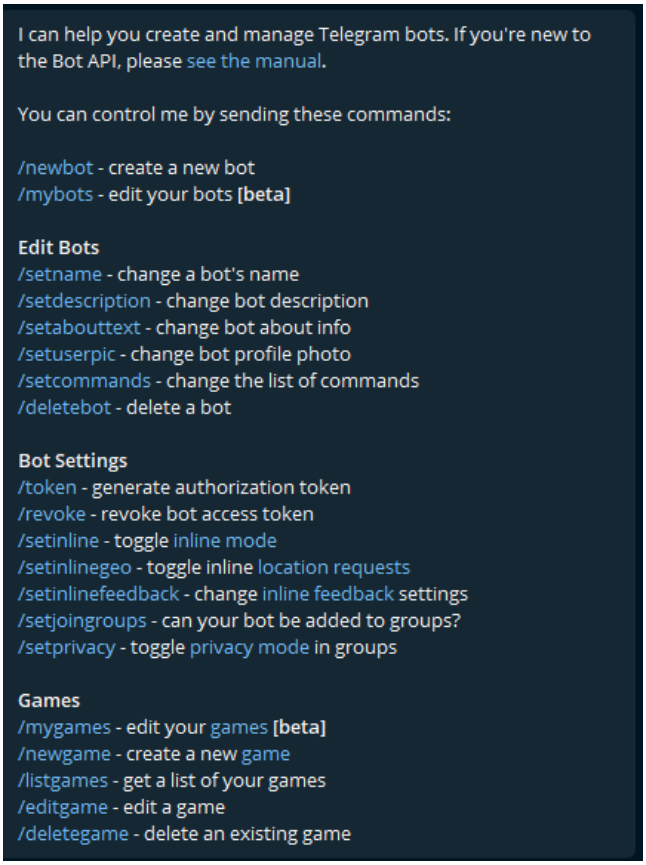

Рисунок 3.6 – Команди бота-батька

Джерело: авторська розробка

Для створення нового робота відправляємо команду /newbot. Після відповідей на пару питань бот буде створено, а батько роботів надішле токен. Його потрібно буде вказувати у коді для взаємодії з BotAPI.

Токен для кожного робота унікальний. Не можна, щоб він потрапив у відкритий доступ. Однак якщо це сталося, його можна змінити через ботопапу командою /revoke.

Так буде виглядати діалог створення бота:

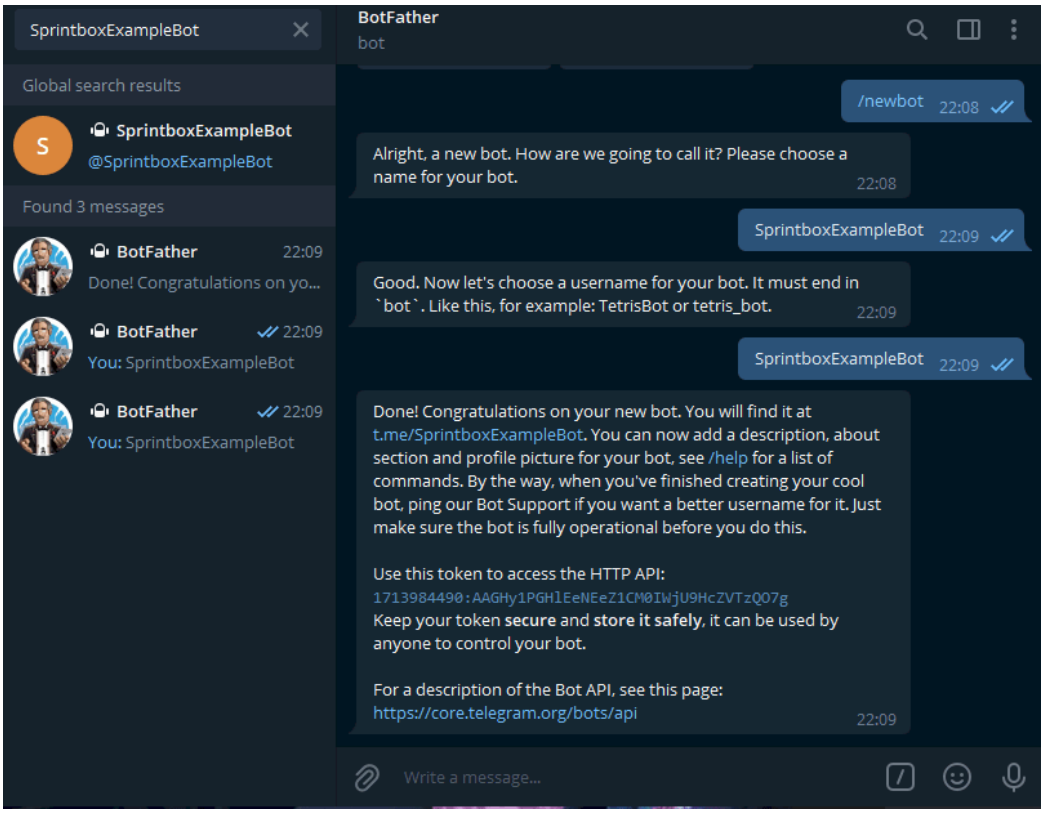

Рисунок 3.7 - Діалог з ботом-батьком для створення власного бота

Джерело: авторська розробка

Настроювання Docker оточення.

Насамперед знадобиться обліковий запис на Docker Hub. Це аналог GitHub, тільки не з вихідними кодами, а з вже створеними контейнерами. Робота з Docker Hub виглядає досить просто:

Локально або за допомогою пайплайнів збудуємо контейнер.

Завантажуємо його на докер хаб.

У будь-якому зручному місці завантажуємо його. Це може бути локальна машина, сервер VPS або хмарний провайдер за типом AWS.

Запускаємо.

Пройдемося цими кроками. Скрізь, де зазначено <docker_hub_username>, вставляємо своє ім’я, використане під час реєстрації на докерхабі.

Створюємо контейнер:

```
docker build -t <docker_hub_username>/my_app
```

Завантажуємо його на докерхаб:

```
docker push <docker_hub_username>/my_app
```

Для перевірки успішності завантаження запускаємо контейнер із Docker Hub за допомогою команди:

```
docker run -d <docker_hub_username>/my_app
```

Далі завантажуємо наш контейнер у AWS Elastic Beanstalk.

3.1.2 Модулі та бібліотеки для інтеграції проекту з Telegram

Для початку створимо каталог для робота, організуємо там virtual environment (далі venv) і встановимо бібліотеку aiogram.

Тепер створимо файл requirements.txt, в якому вкажемо версію aiogram, що використовується.

```
[groosha@main lesson_01]$ python3.7 -m venv venv
[groosha@main lesson_01]$ echo "aiogram==2.9.2" > requirements.txt
[groosha@main lesson_01]$ source venv/bin/activate
(venv) [groosha@main lesson_01]$ pip install -r requirements.txt
Successfully installed Babel-2.8.0 aiogram-2.9.2 aiohttp-3.6.2 async-timeout-3.0.1
attrs-19.3.0 certifi-2020.6.20 chardet-3.0.4 idna-2.10 multidict-4.7.6 pytz-2020.1
typing-extensions-3.7.4.2 yarl-1.5.1
WARNING: You are using pip version 19.2.3, however version 20.2.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(venv) [groosha@main lesson_01]$
```

Почнемо створювати першого бота на тестових даних.

Створимо файл bot.py з базовим шаблоном робота на aiogram:

```
#!venv/bin/python
import logging
from aiogram import Bot, Dispatcher, executor, types
```

```
# Об’єкт бота
bot = Bot(token="12345678:AaBbCcDdEeFfGgHh")
# Диспетчер для бота
dp = Dispatcher(bot)
# Вмикаємо логування, щоб не пропустити важливі повідомлення
logging.basicConfig(level=logging.INFO)

# Хэндлер на команду /test1
@dp.message_handler(commands="test1")
async def cmd_test1(message: types.Message):
    await message.reply("Test 1")

if __name__ == "__main__":
    # Запуск бота
    executor.start_polling(dp, skip_updates=True)
```

Важливий момент: aiogram – асинхронна бібліотека, тому функції також мають бути асинхронними, а перед викликами методів API потрібно ставити ключове слово await, т.к. ці виклики повертають корутини.

Диспетчер реєструє функції-обробники, додатково обмежуючи перелік подій, що викликають їх через фільтри. Після отримання чергового апдейта (події від Telegram), диспетчер вибере потрібну функцію обробки, відповідну по всіх фільтрах, наприклад, «обробка повідомлень, що є зображеннями, у чаті з ID ікс та з довжиною паролю». Якщо дві функції мають однакові за логікою фільтри, буде викликана та, що зареєстрована раніше.

Щоб зареєструвати функцію як обробник повідомлень, потрібно зробити одну з двох дій:

– підключити декоратор;

– безпосередньо викликати метод реєстрації у диспетчера.

Напишемо тестовий код:

```
# Хэндлер на команду /start
@dp.message_handler(commands="/start")
async def cmd_test1(message: types.Message):
    await message.reply("Привет")

# Хэндлер на команду /test2
async def cmd_test2(message: types.Message):
    await message.reply("Привет")
```

Запустимо з ним бота:

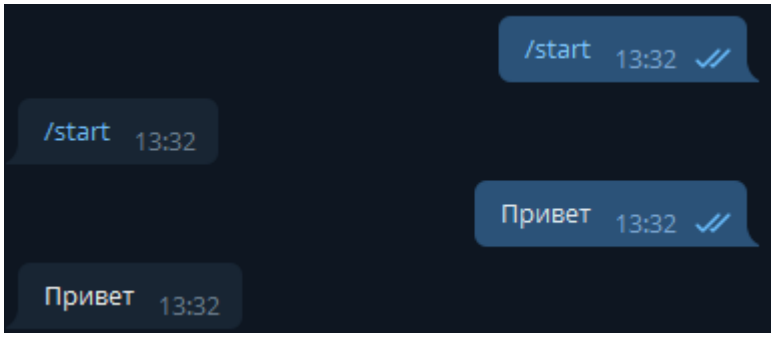

Рисунок 3.8 – Тестовий запуск бота

Джерело: авторська розробка

При роботі бота неминуче виникнення різних помилок, пов’язаних не з кодом, а з зовнішніми подіями. Найпростіший приклад: спроба надіслати повідомлення користувачу, який заблокував роботу. Щоб не обертати кожен раз у try..except, в aiogram існує спеціальний хендлер для винятків, пов’язаних з Bot API.

Наприклад існує затримка-перевірка доступності у 10 секунд.

За ці 10 секунд користувач може встигнути заблокувати робота зі свого боку і спроба викликати метод reply призведе до появи виключення BotBlocked.

Таким чином треба написати обробники на вийнятки. Таким чином, якщо та сама непередбачена ситуація може виникнути в різних хендлерах, то можна винести її обробку в окремий хендлер помилок.

Для того, щоб зробити код чистішим і читальнішим, aiogram розширює можливості стандартних об’єктів Telegram. Наприклад, замість bot.send_message(...) можна написати message.answer(...) або message.reply(...). В останніх двох випадках не потрібно підставляти chat_id, мається на увазі, що він такий самий, як і у вихідному повідомленні.

Різниця між answer і reply проста: перший метод просто відправляє повідомлення у той же чат, другий робить «відповідь» на повідомлення з message:

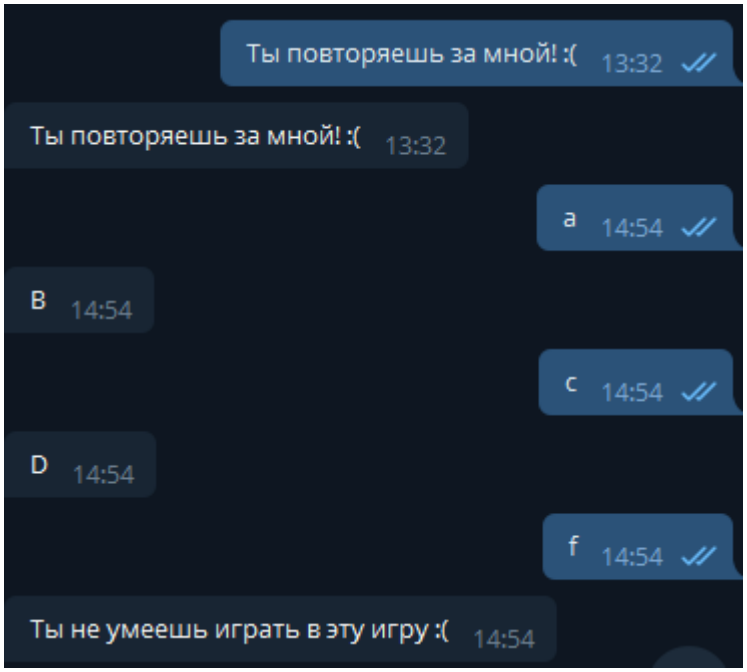

Рисунок 3.9 – Приклад роботи боту

Джерело: авторська розробка

При виклику команди /dice бот відправить у той самий чат гральний кубик. Якщо його треба відправити в якийсь інший чат, то доведеться по-старому викликати await bot.send_dice(...). Але об’єкт bot (примірник класу Bot) може бути недоступний у сфері видимості конкретної функції. На щастя, об’єкт бота доступний у всіх типах апдейтів: Message, CallbackQuery, InlineQuery і т.д.

Все добре, але якщо раптом виникне потреба поділитися з кимось кодом, то доведеться щоразу пам’ятати про видалення з вихідних джерел токена бота, інакше доведеться його перевипускати у @BotFather. Щоб убезпечити себе, не будемо вказувати токен прямо в коді, а винесемо його як змінну оточення.

Збережемо данні, як шаблон, та будемо удосконалювати шаблон.

3.1.3 TelegramAPI та створення власного чат-боту

Це перша бібліотека Node.js для Telegram Bot API.

Це просунуті алгоритми, що спрощують розробку бота. Пакет надасть прості функції для отримання повідомлень від користувачів-ботів та надсилання їм відповідей.

Бібліотека не перевіряє жодних параметрів, все, що надсилається до бібліотеки, буде надіслано на сервери Telegram. Це означає, що є можливість скористатися новими параметрами, доданими розробниками Telegram у перший день, оскільки оновлення бібліотеки не потрібно.

Конфігурація API.

Ви повинні передати об’єкт конфігурації в конструктор API, який має такі поля.

Таблиця 3.1 – Конфігурація API

Найменування	Обов’язковість	Опис
token	Обов’язкове	Токен доступу Telegram (отриманий від BotFather)
базовый URL	Необов’язковий	Базова URL-адреса серверів Telegram (може бути корисною, якщо ви підключаєтеся до проксі-сервера Telegram у країнах, де Telegram заблоковано. За замовчуванням:

		(https://api.telegram.org)
http_proxy	Необов’язковий	Цей об’єкт є необов’язковим. Використовуйте його, якщо ви хочете, щоб API підключався через проксі
http_proxy.host	Обов’язкове	Ім’я хоста проксі
http_proxy.port	Обов’язкове	Порт проксі
http_proxy.user	Необов’язковий	Ім’я користувача (автентифікація)
http_proxy.пароль	Необов’язковий	Пароль (автентифікація)
http_proxy.https	Необов’язковий	Https з’єднання

3.2 Алгоритми генерування паролів

Вимоги до пароля користувача.

До складу пароля користувача на вхід до системи або сайту можуть входити наступні групи символів:

– великі літери латинського (A-Z) алфавіту;

– рядкові літери латинського (a-z) алфавіту;

– арабські цифри (0-9);

– спеціальні символи – розділові знаки та математичних дій, різні дужки, знаки #, @.

Дані вимоги можуть відрізнятися, в залежності від призначення паролю, саме тому в боті існує три основні види паролю: літеро-числовий, літеро-символьно-числовий, числово-символьний. Крім того, користувач може обрати потрібну йому довжину паролю: від 8 до 20 символів.

Опираючись на дані вимоги, було прийнято рішення створити універсальний генератор, який буде створювати пароль за трьома основними та найбільш поширеними варіаціями. Це відрізняє даний генератор від більшості інших, вже існуючих. Також, користувач може самостійно задати

розмір (кількість символів) паролю. Далі наведено блок-схему роботи генератора.

Завдяки алгоритмам шифрування, що є у Telegram, чат-бот є надійним та захищеним. Для користувачів є можливість обрання двофакторної автентифікації, яка підвищує захищеність даних, що зберігаються у відповідних чатах.

Блок-схема роботи алгоритму представлена на рисунку нижче:

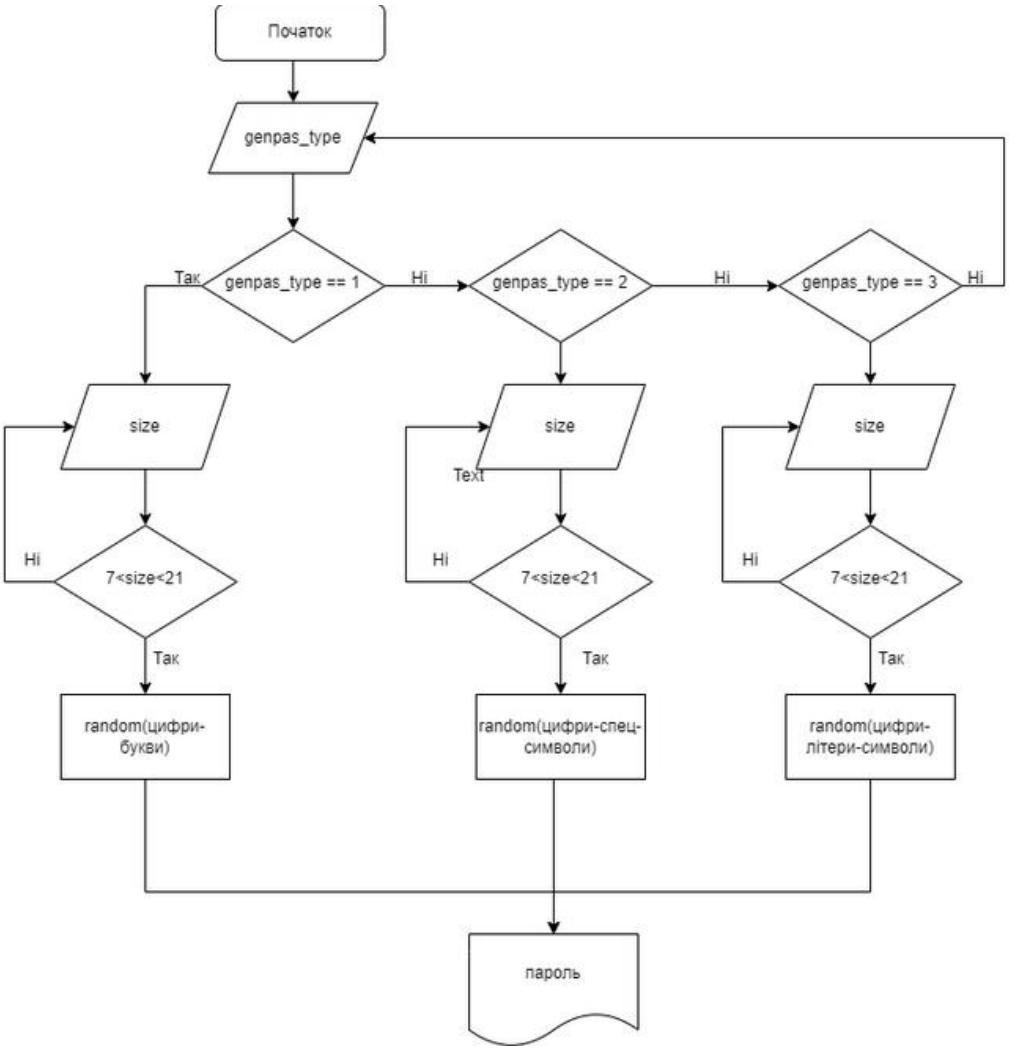

Рисунок 3.10 – Блок-схема алгоритму генерації паролю

Джерело: авторська розробка

3.3 Реалізація інтерфейсу користувача

Спочатку бота потрібно знайти в самому месенджері. Це можна зробити 2 способами: або перейти за готовим посиланням, якщо посилання було на сайті або хтось закинув, або написати назву бота в пошук Telegram.

Для взаємодії користувача та бота на старті роботи зроблено кнопку «розпочати». При нажаті на неї на екрані буде перше повідомлення, яке розпочне спілкування (див. рис. 3.11).

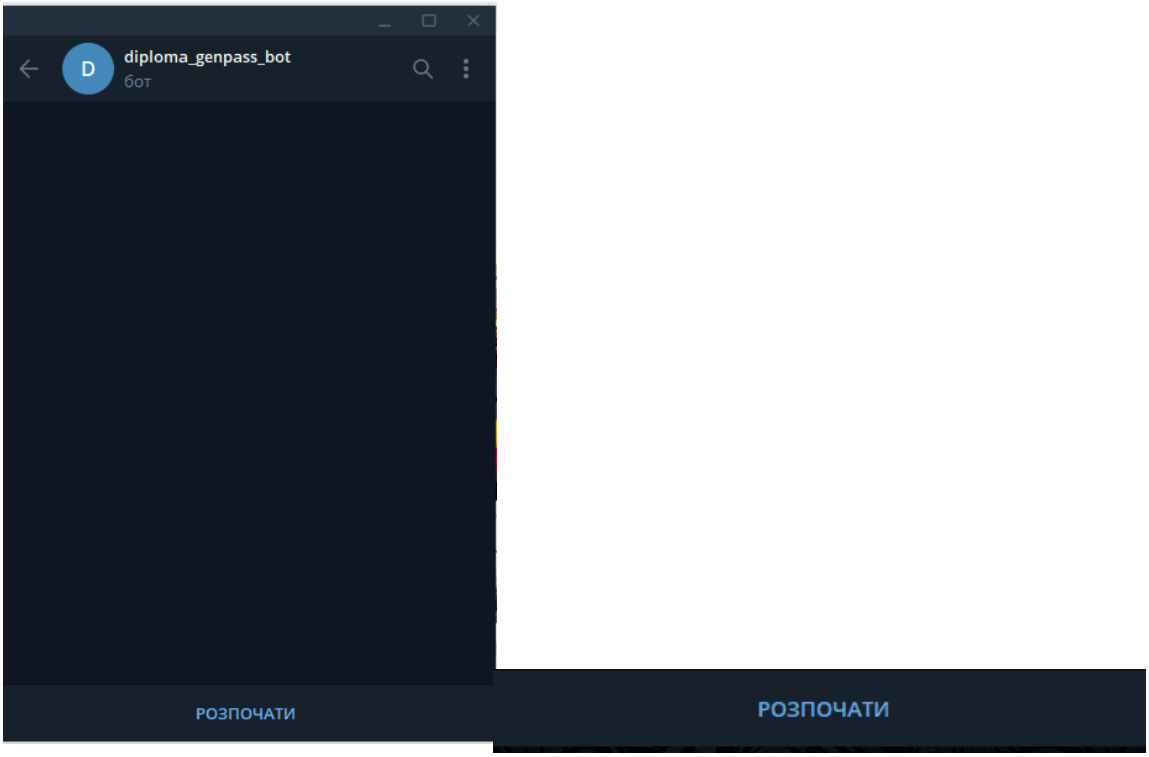

Рисунок 3.11 – Початок спілкування з ботом

Джерело: авторська розробка

Відразу після цього бот почне свою роботу і запропонує користувачам виконати цільову дію: поставити питання, вибрати потрібний пункт меню, написати певне слово і т.д.

Після натискання на кнопку бот надсилає відповідну інформацію прямо в чат (див. рис. 3.12).

Меню є у багатьох ботів, тому що це найзручніший варіант для користувачів. Однак зустрічаються такі «боти», взаємодія з якими трохи складніша, тому що немає візуального меню. Щоб користуватися роботом, потрібно вводити певні команди. Щоб дізнатися про їх повний список, наберіть у чаті значок слєша / (табл. 3.2).

Таблиця 3.2 – Команди меню доступні боту

Команда	Опис
/start	Почати взаємодію
/help	Переглянути можливості
/settings	Налаштувати бота

Надалі бот виконує заложені у нього функції. Генерує паролі та надає можливість видаляти або залишати пароль (див. рис. 3.12).

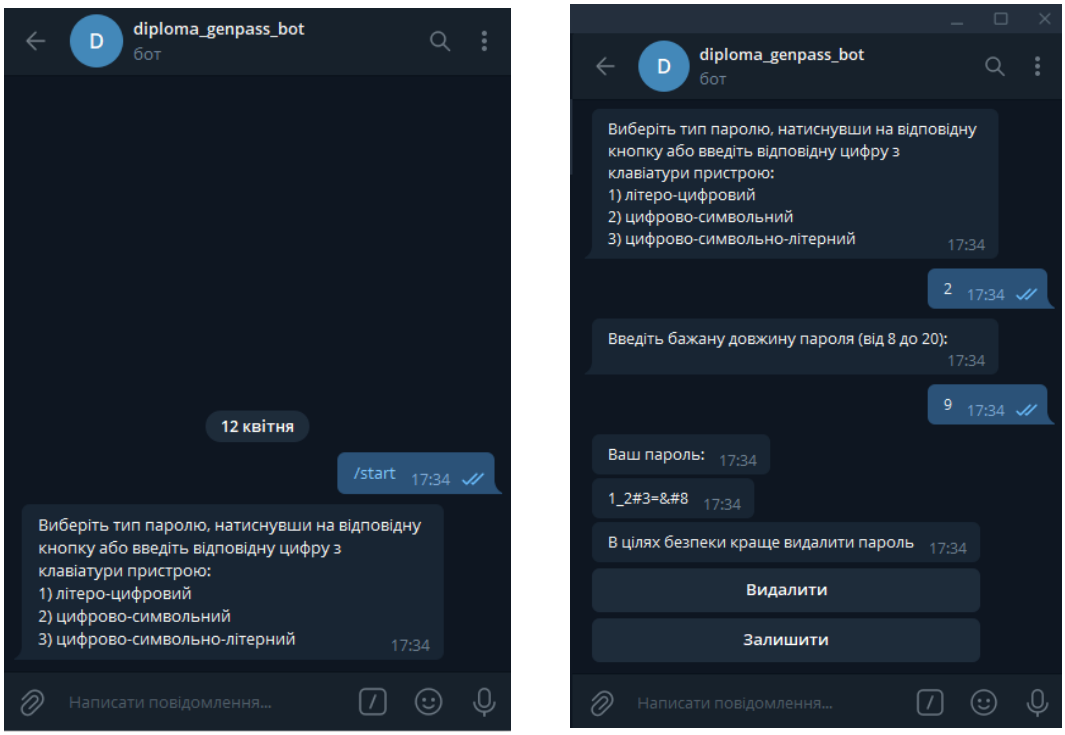

Рисунок 3.12 – Інтерфейс бота

Джерело: авторська розробка

На рисунку 3.13 можна подивитися закінчення однієї гілки спілкування. Та бот надає можливість почати нову гілку спілкування.

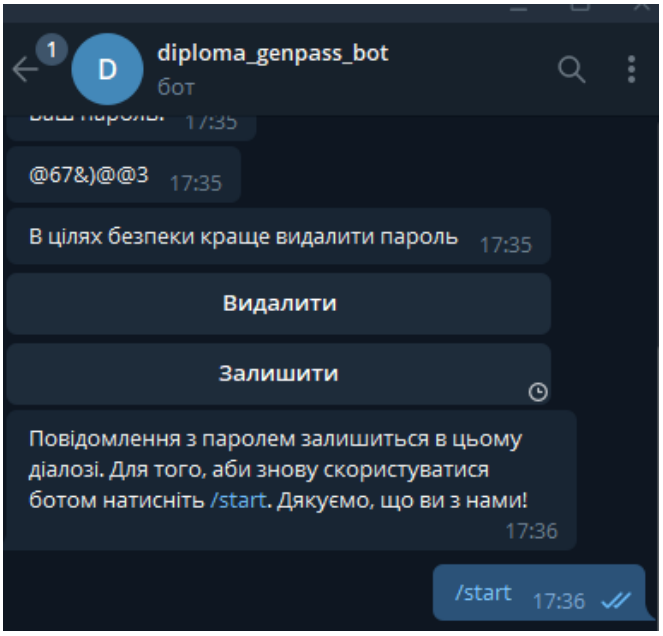

Рисунок 3.13 – Завершення спілкування з ботом

Джерело: авторська розробка

3.4 Висновки за розділом

В ході написання третього розділу було налаштовано середу розробки для чат-боту. Розібрані та обрані варіанти представлення даних. Був послідовно розроблено чат-бот. Описані усі використані бібліотеки та алгоритми для роботи з ботом. Також було розроблено користувальницький інтерфейс. Обрано найзручніший спосіб взаємодії бота та користувача. Бот є надійним та захищеним, тому що розроблено на базі Telegram, у якому є спеціальні протоколи шифрування у чатах та двофакторна автентифікація користувачів.

4 ТЕСТУВАННЯ МЕНЕДЖЕРУ ПАРОЛІВ З ПІДВИЩЕНИМ ЗАХИСТОМ НА ОСНОВІ ТЕЛЕГРАМ БОТУ

4.1 Загальне тестування чат-боту

Загальне тестування – це тести на запитання та відповіді, орієнтовані на загальні питання, на які, як очікується, відповідь найпростіша для боту. Наприклад, перевіряється вітання користувача. Бот повинен пройти загальний тест, щоб розглянути можливість запуску інших тестів.

Очікування цього тесту полягає в тому, що чат-бот підтримує плавну бесіду, і якщо бот вийде з ладу на першому етапі, ймовірно, що користувач вийде з розмови. Це погіршує такі показники чат-бота як показник відмов [13-14].

Чат-бот менеджер паролів пройшов загальний тест. Ніяких відмов не було виявлено.

Повідомлення у діалогах тестувалися за схемою спрямованого ациклічного графа.

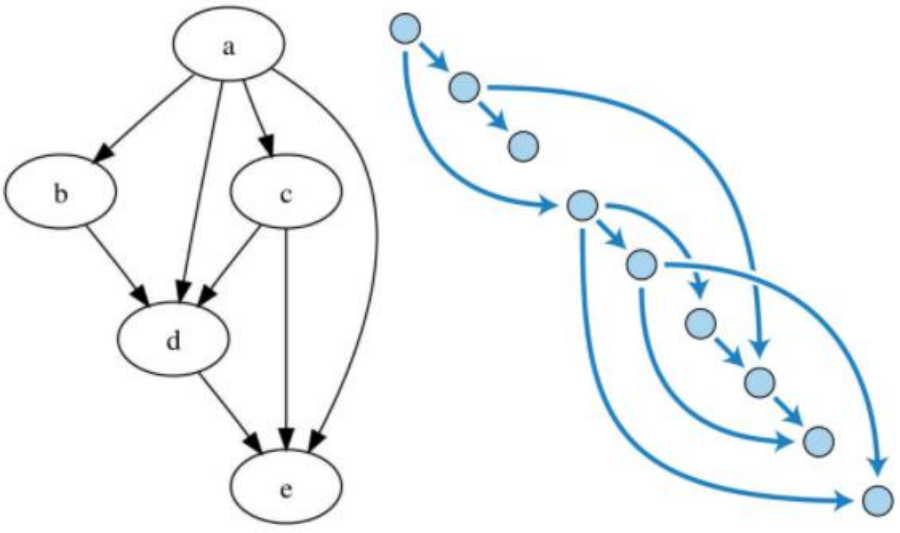

Рисунок 4.1 – Спрямований ациклічний граф

Джерело: авторська розробка

4.2 BDD-тестування чат-тобу

BDD – це відгалуження методології розробки через тестування. Суть BDD полягає в тому, що є бізнес-вимоги до продукту, які описуються предметно-орієнтованою мовою та отримуються сценарії використання системи. Далі реалізується технічна частина системи й у кінцевому підсумку створюється конкурентний продукт [15].

Класичний цикл Behaviour-Driven Development складається з 5 кроків:

– описується поведінку системи;

– визначається технічні вимоги до системи;

– запускаються тести чи руками перевіряється сценарії поведінки, а вони, звісно, падають, бо система ще не готова чи не реалізована;

– дороблюється система, тобто створюється, оновлюється, доповнюється, змінюється;

– добиваємось того, щоб система проходила наші сценарії використання.

У основі BDD лежать специфікації поведінки. Це документи, які мають таку структуру:

– заголовок – опис бізнес-мети;

– опис – суб’єкт, склад історії, цінність для бізнесу;

– сценарії – ситуація поведінки суб’єкта.

Для тестування повного циклу роботи чат-боту було розроблено 5 кейсів перевірки.

Перший кейс перевірки: бот привітає користувача.

Тут є конкретна функція: Бот вміє вітати користувача. Крім того, ми бачимо передісторію, що це за клієнт, в якій ОС він сидить і яку версію додатку використовує. Адже в залежності від різних версій програми можуть бути показані різні віджети. І у нас сценарій, згідно з якому користувач обирає старт для початку діалогу (див. рис. 4.2).

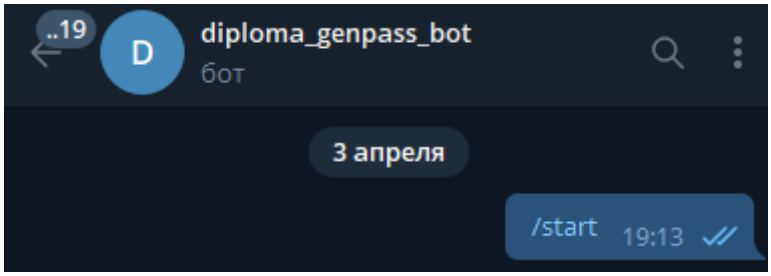

Рисунок 4.2 – Перевірка першого кейсу тестування

Джерело: авторська розробка

Другий кейс перевірки: бот дає вибирати дію – три варіанти генерації паролю.

На цьому етапі користувачу потрібно обрати за яким сценарієм генерувати пароль (див. рис. 4.3).

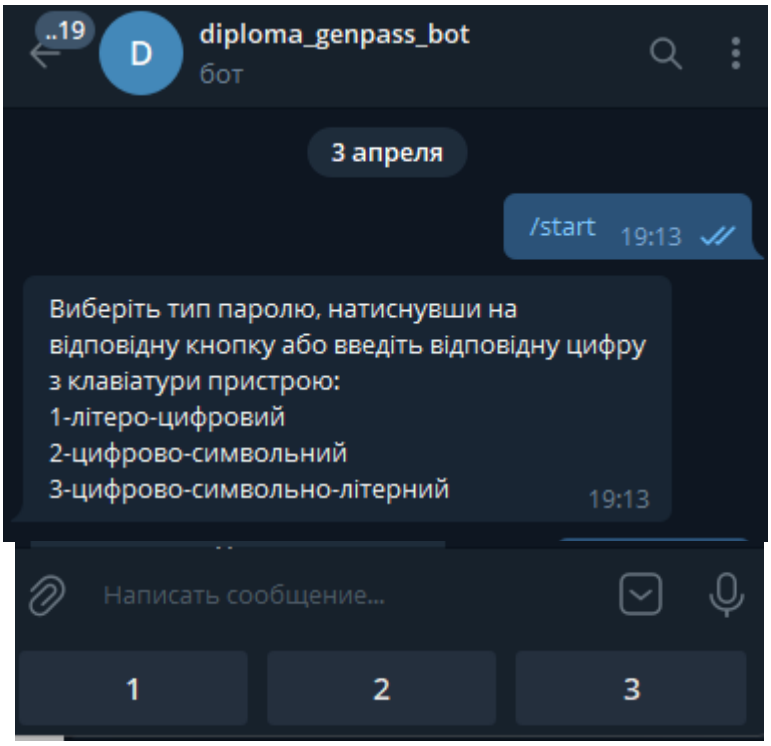

Рисунок 4.3 – Перевірка другого кейсу тестування

Джерело: авторська розробка

Третій кейс перевірки: бот пропонує ввести параметр для генерації паролю – кількість символів від 8 до 20.

На третьому етапі бот генерує пароль по заданому алгоритму. Користувач обирає довжину паролю та очікує на генерацію (див. рис. 4.4).

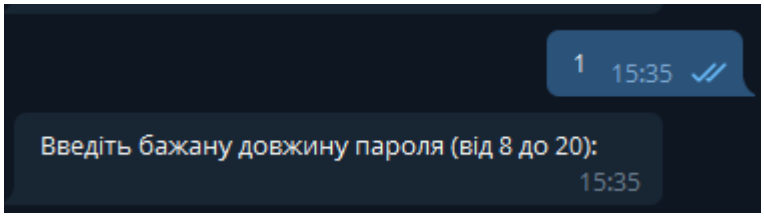

Рисунок 4.4 – Перевірка третього кейсу тестування

Джерело: авторська розробка

Четвертий кейс перевірки: бот генерує пароль.

На цьому етапі генерується пароль заданої довжини. Та пропонується або видалення задля безпеки, або залишення у чаті.

П’ятий кейс перевірки: бот пропонує видалити сгенерований пароль та почати сесію з початку.

На цьому етапі користувач може або видалити, або залишити пароль, та почати сесію з початку.

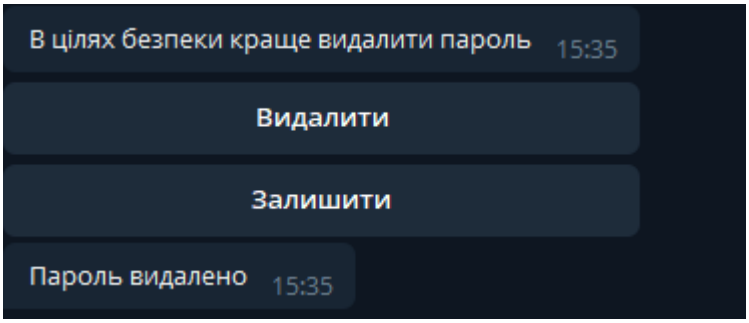

Рисунок 4.5 – Перевірка п’ятого кейсу тестування

Джерело: авторська розробка

Усі кейси робочі та пройшли перевірку.

4.3 Пост-продакшн тестування

Після-виробниче (або пост-продакшн) тестування або тестування після запуску перевіряє різні аспекти функціональності чат-бота, коли він активний для користувачів.

Звичайно, практично неможливо навчити свого чат-бота відповідати на кожен запит, який користувачі вводять в чат-бота, і хтось може написати один й той же запит різними способами, з орфографічними помилками або іншими проблемами.

Як вже було з’ясовано раніше, відповіді мають бути якомога ширшими, або тільки вибірними, щоб передбачити ці ситуації. Цей етап здебільшого складається з приділення особливої уваги тому, як бот реагує, і чи є інформація точною.

Будь-яке тестування перед запуском чат-бота є бета-тестом. Зворотній зв’язок життєво важливий, і необхідно періодично запускати тести з запитаннями, що задаються користувачами, які проводять реальне тестування чат-бота.

Іншими словами: тестування треба проводити на регулярній основі. Треба постійно налаштовувати чат-бота на відгуки, які ви отримаєте після запуску. Ціль полягає в тому, щоб підтримувати актуальність відповідей, поточну продуктивність можна поліпшити в майбутньому, піддавши чат-боту процесу безперервного тестування.

4.4 Висновки за розділом

У процесі тестування не було виявлено явних відхилень. Тестування проводилося на загальних підставах (ручне тестування) та бізнес-тестування з написанням кейсів перевірки. Також було проведено пост-продакшн тестування вже на запущеному варіанті чат-боту. Усі тести бот пройшов.

ВИСНОВКИ

В ході написання дипломної роботи було сформульовано наступну тему: «Захист конфіденційної інформації в месенджерах».

Поставлено наступні задачі на розробку:

– проаналізувати літературу та інтернет-джерела, з метою відбору та використання матеріалу по створенню чат-боту;

– визначити програмне забезпечення для розробки чат-боту;

– реалізувати менеджер паролів з підвищеним захистом на основі телеграм боту.

Під час написання першого розділу було розглянуто методи захисту даних в месенджерах, розглянуто вимоги до парольного захисту та складності. Визначені вимоги до менеджеру паролів з підвищеним захистом та поставлено завдання розробки. Обрано засоби для реалізації, а також розглянуто існуючи аналоги чат-ботів розроблених тими ж самими засобами.

В ході написання другого розділу було створено модель взаємодії користувача з чат-ботом. Все це було спроєктовано на основі діаграм варіантів використання та діаграми станів. Проаналізовано та обрано модель роботи з серверною частиною. Приведено схеми розміщення та зберігання даних на сервері. Розглянуто варіанти створення користувальницького інтерфейсу, можливості які представляються функціоналом Telegram. Визначено схему зберігання даних, що накопичуються у діалогах з чат-ботом.

Розглянуто деякі моменти інтеграції, визначено складні моменти та обрано план дій на подальшу розробку чат-боту.

В ході написання третього розділу було налаштовано середу розробки для чат-боту. Розібрані та обрані варіанти представлення даних. Був послідовно розроблено чат-бот. Описані усі використані бібліотеки та алгоритми для роботи з ботом. Також було розроблено користувальницький

інтерфейс. Обрано найзручніший спосіб взаємодії бота та користувача. Бот є надійним та захищеним, тому що розроблено на базі Telegram, у якому є спеціальні протоколи шифрування у чатах та двофакторна автентифікація користувачів.

У процесі тестування не було виявлено явних відхилень. Тестування проводилося на загальних підставах (ручне тестування) та бізнес-тестування з написанням кейсів перевірки. Також було проведено пост-продакшн тестування вже на запущеному варіанті чат-боту. Усі тести бот пройшов.

Усі задачі поставлені на дипломування були виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Scott J. Social network analysis. Sage, 2017.

2. Kacprzyk J. Group decision making with a fuzzy linguistic majority // Fuzzy sets and systems. 1986. Vol. 18. P. 105-118.

3. Herrera-Viedma E., Cabrerizo F.J., Kacprzyk J., Pedrycz W. A Review of Soft Consensus Models in Fuzzy Environment // Information Fusion. 2014. Vol. 17. P. 4-13.

4. Herrera-Viedma E., Cabrerizo F.J., Chiclana F., Wu J., Cobo M.J., Samuylov K. Consensus in Group Decision Making and Social Networks // Studies in Informatics and Control. 2017. Vol. 26, No. 3. P. 259-268.

5. Morente-Molinera J.A., Pérez I.J., Ureña M.R., Herrera-Viedma E. On multi-granular fuzzy linguistic modeling in group decision making problems: a systematic review and future trends // Knowledge Based Systems. 2015. Vol. 74. P. 49-60.

6. Chukhno N., Chukhno O., Gaidamaka A., Samuylov K., Herrera-Viedma E. A New Ranking Method of Alternatives for Group Decision Making in Social Networks // 10th International Congress on Ultra-Modern Telecommunications and Control Systems and Workshops (ICUMT). 2018.

7. Abdul-Kader S.A., Woods J. Survey on Chatbot Design Techniques in Speech Conversation Systems // International Journal of Advanced Computer Science and Applications. Vol. 2015. Vol. 6, No. 7.

8. Dahiya M. A tool of conversation: Chatbot // International Journal of Computer Sciences and Engineering. 2017. Vol. 5, No. 5. P. 158-161.

9. Kluwer T. From chatbots to dialog systems // Conversational agents and natural language interaction: Techniques and effective practices. IGI Global. 2011. P. 1-22.

10. Sayed S., Jain R., Lokhandwala B., Barodawala F., Rajkotwala M. Android based Chat-Bot // International Journal of Computer Applications. 2016. Vol. 137, No. 10. P. 29-32.

11. Mikic Fonte F.A., Nistal M.L., Burguillo Rial J.C., Rodríguez M.C. NLAST: A natural language assistant for students // IEEE Global Engineering Education Conference (EDUCON). 2016. P. 709-713.

12. Kumar M.N., Chandar P.C. L., Prasad A.V., Sumangali K. Android based educational Chatbot for visually impaired people // 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). Chennai, 2016. P. 1-4.

13. Experimentally induced visual projections into auditory thalamus and cortex/ Sur, M., P.E. Garraghty and A.W. Roe // Science 242 1988.P.1437–1441.

14. Tactile sensory substitution studies / Bach-y-Rita P // Annals of the New York Academy of Sciences -2004. P. 83–91, 2004.

15. Brainport technologies– helping people with disabilities live a better life [electronic resource]/ Wicab, Inc. Middleton, WI, USA / URL: https://www.wicab.com/, free access – Lang. Eng. Access date: 25.04.2020

Додаток А

Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
ПЗ		Нещільне заповнення сторінок ПЗ
С.1 6-19		Невірні переліки двох рівней
Таблиця 3.1		Невірне продовження таблиці

Додаток Б

Заява на затвердження теми та керівника випускної кваліфікаційної роботи

Завідувачу кафедри прикладної
математики і інформатики
факультету комп'ютерно-
інформаційних технологій та
автоматизації,
д.т.н., проф. Дмитрієвій О.А.
студента гр. КІБ-18
Парубов В.А.

ЗАЯВА

Прошу затвердити наступну тему випускної кваліфікаційної роботи «Захист конфедиційної інформації в месенджерах» («Confidential information protection in messengers») зі спецчастиною «Розробка менеджеру паролів з підвищенним захистом на основі телеграм боту» («The password manager with enhanced protection development based on telegram bot»).

Науковим керівником призначити: проф. кафедри ПМІ, д.т.н., проф. Башков Є.О.

«___06___» __квітня____ 2022р. _____

(підпис студента)

Додаток В

Лістинг програмного коду

```
import aiogram
import telebot
from telebot import types
import random

bot =
telebot.TeleBot("5218438398:AAG0kNQMjkXk_bKPBS0FZk8HCOmsjlRdWzQ")

@bot.message_handler(content_types=[‘text’])
def start(message):
    global genpas_type

    if message.text == ‘1’ or message.text == ‘2’ or message.text ==
‘3’:
        bot.send_message(message.from_user.id, "Введіть бажану довжину
пароля (від 8 до 20):")
        genpas_type = int(message.text)
        bot.register_next_step_handler(message, get_size)
    else:
        bot.send_message(message.from_user.id, "Виберіть тип паролю,
натиснувши на відповідну кнопку або введіть відповідну цифру з
клавіатури пристрою:"
                         "\n1) літеро-цифровий \n2) цифрово-символьний
\n3) цифрово-символьно-літерний")


def get_size(message):
    global size
    keyboard = types.InlineKeyboardMarkup()
    try:
        size = int(message.text)
        if size > 7 and size < 21:
            bot.send_message(message.from_user.id, ‘Ваш пароль:’)
            if genpas_type == 1:
              #  bot.register_next_step_handler(message,
first_genpass)
                password = ‘‘
                for x in range(size):
```

```
                    password = password +
random.choice(list('1234567890abcdefghigklmnopqrstuvyxwzABCDEFGHIGKLMN
OPQRSTUVYXWZ'))
                bot.send_message(message.from_user.id, password)
            elif genpas_type == 2:
               # bot.register_next_step_handler(message,
second_genpass)
                password = ''
                for x in range(size):
                    password = password +
random.choice(list('1234567890!@#$%^&*()_+=-'))
                bot.send_message(message.from_user.id, password)
            else:
              #  bot.register_next_step_handler(message,
third_genpass)
                password = ''
                for x in range(size):
                    password = password +
random.choice(list('1234567890abcdefghigklmnopqrstuvyxwz!@#$%^&*()_+=-
ABCDEFGHIGKLMNOPQRSTUVYXWZ'))
                bot.send_message(message.from_user.id, password)
            key_del = types.InlineKeyboardButton(text='Видалити',
callback_data='del_message')
            key_stay = types.InlineKeyboardButton(text='Залишити',
callback_data='del_message')
            # додаємо кнопку на екран
            keyboard.add(key_del)
            keyboard.add(key_stay)
            bot.send_message(message.from_user.id, 'В цілях безпеки
краще видалити пароль', reply_markup=keyboard)
        else:
            bot.send_message(message.from_user.id, 'Введіть цифрове
значення від 8 до 20:')
            bot.register_next_step_handler(message, get_size)
    except Exception:
        bot.send_message(message.from_user.id, 'Введіть цифрове
значення від 8 до 20:')
        bot.register_next_step_handler(message, get_size)

bot.polling(none_stop=True, interval=1)
```

Додаток Г

Роздатковий матеріал

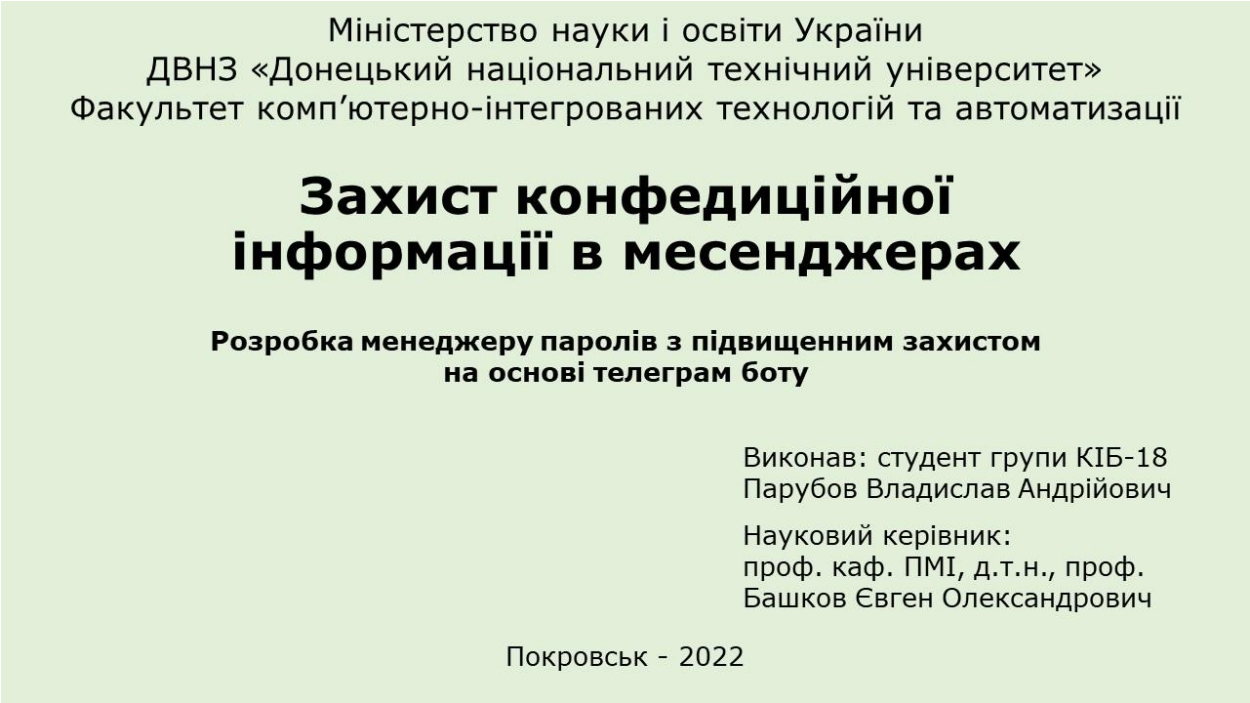

Рисунок Г.1 – Титульна сторінка

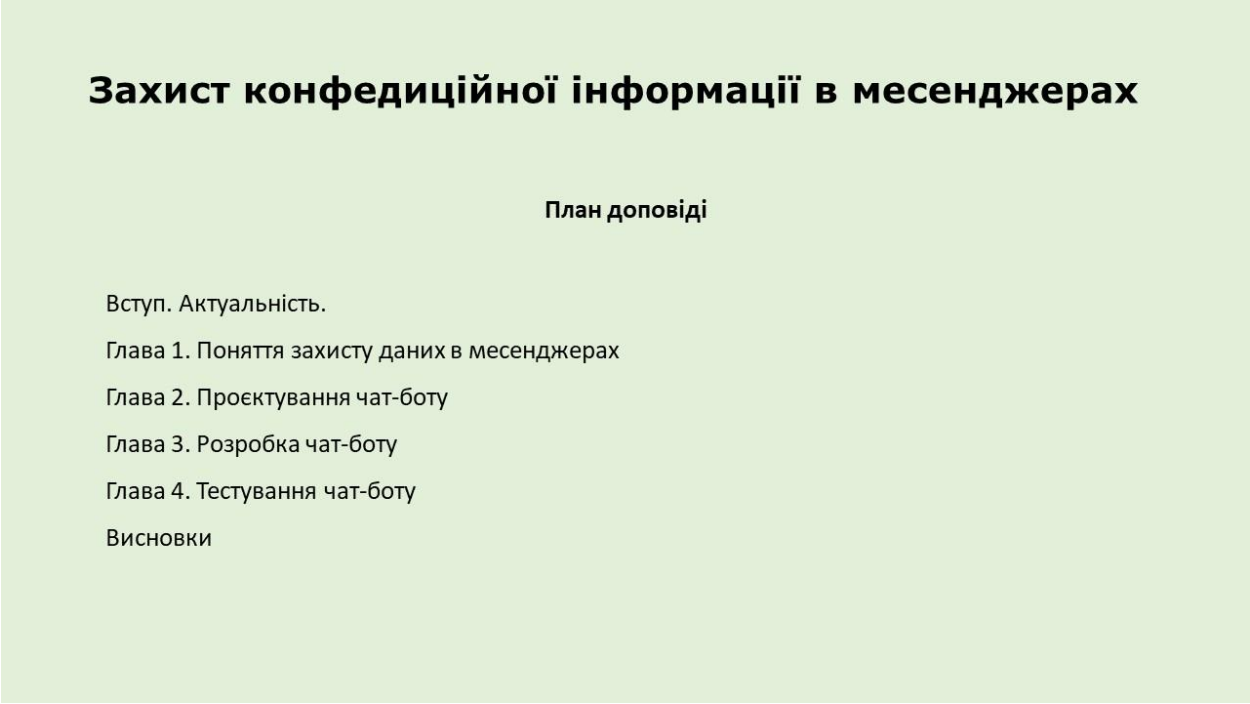

Рисунок Г.2 – План доповіді

ВСТУП

Об’єкт дослідження – процеси та процедури захисту інформації в соціальних мережах.
Предмет дослідження – менеджер паролів для месенджеру з підвищеним захистом на основі телеграм боту.
Мета роботи – розробка дизайну і функціоналу менеджеру паролів з підвищеним захистом на основі телеграм боту.
Методи дослідження – аналіз додатків існуючої проблеми менеджерів паролів, виявлення переваг і недоліків; визначення вимог до структури, поведінки і функціоналу чат-боту; вибір інструментальних засобів розробки.
Задачі розробки:

- проаналізувати літературу та інтернет-джерела, з метою відбору та використання матеріалу по створенню чат-боту;
- визначити програмне забезпечення для розробки чат-боту;
- реалізувати менеджер паролів з підвищеним захистом на основі телеграм боту.

Рисунок Г.3 – Вступ

Поняття захисту даних в месенджерах

Месенджери добре вписалися в сучасний процес цифровізації суспільства, і цілком об’єктивно очікувати від цифрового спілкування приблизно такої ж безпеки, як і при вербальному (особистому) спілкуванні співрозмовників.
Основні параметри безпеки:

- конфіденційність розмови (вербальну розмову між співрозмовниками можна вважати конфіденційною за умови, що ніхто не чує розмови учасників);
- ідентифікація та автентифікація учасника розмови (співрозмовника ми зазвичай впізнаємо (ідентифікуємо) за його персональними особливостями (голос, зовнішність та ін.), запам’ятовуємо їх разом з ім’ям учасника діалогу при першому знайомстві, а згодом переконуємось у тому, що перед нами саме та людина, за яку він себе видає);
- автентифікація джерела даних (перевірка та підтвердження того, що інформація створена саме заявленим учасником розмови).

Рисунок Г.4 – Поняття захисту даних в месенджерах

Методи захисту даних в месенджерах

Процес взаємодії двох учасників розмови A і B починається з вироблення загального секретного ключа та автентифікації абонентів. Вироблення загального секретного ключа здійснюється за допомогою протоколу Діффі – Хеллмана:

$$A \rightarrow B: Sing(g_A^x, k_A), K_A$$
$$B \rightarrow A: Sing(g_B^x, k_B), K_B$$

Рисунок Г.5 – Методи захисту даних в месенджерах

Загальні поняття чат-ботів

Чат-боти – це спеціальні акаунти, за якими не закріплена будь-яка людина, а повідомлення, надіслані з них або на них, обробляються зовнішньою системою. Крім того, для користувача спілкування з ботом виглядає як звичайне листування з реальною людиною.

Чат-бот – це розумна програма, яка живе у месенджерах та виконує різні функції.

Функції чат-бота:

- підтримка клієнтів;
- клієнтський сервіс;
- маркетинг;
- робота всередині компанії;
- рекрутинг.

Рисунок Г.6 – Загальні поняття чат-ботів

Проєктування моделі чат-боту

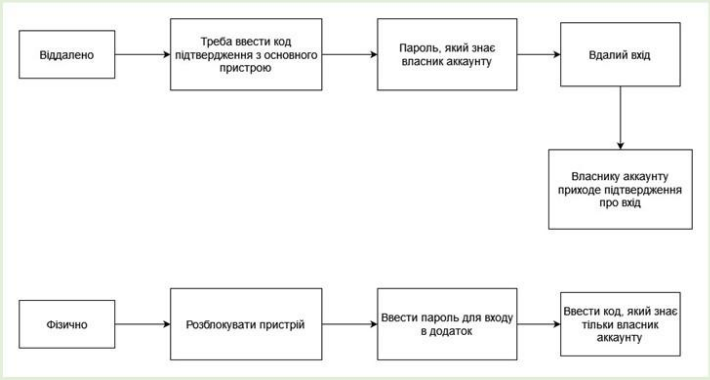

Рисунок 1 – Схема входу у Telegram
Джерело: авторська розробка

Рисунок Г.7 – Проєктування моделі чат-боту

Проєктування моделі чат-боту

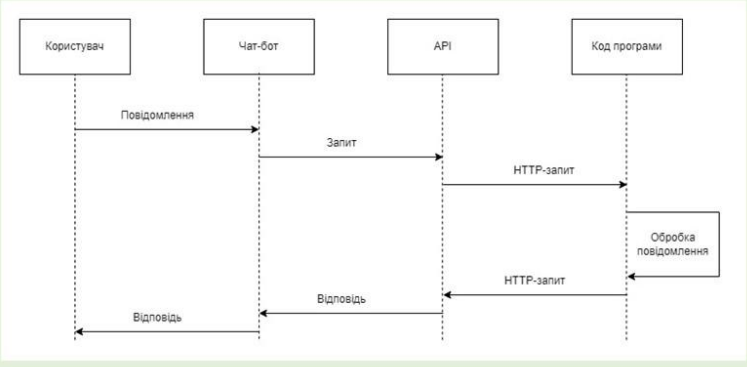

Рисунок 2 – Діаграма станів
Джерело: авторська розробка

Рисунок Г.8 – Діаграма станів

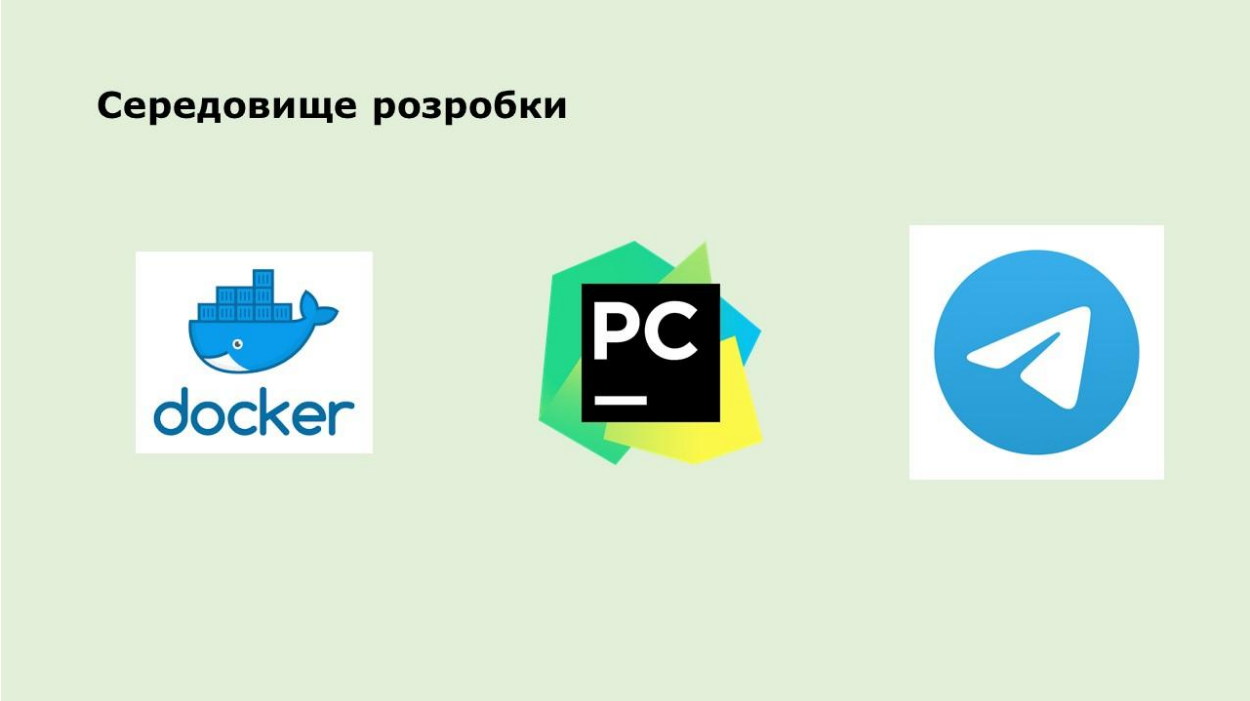

Рисунок Г.9 – Середовище розробки

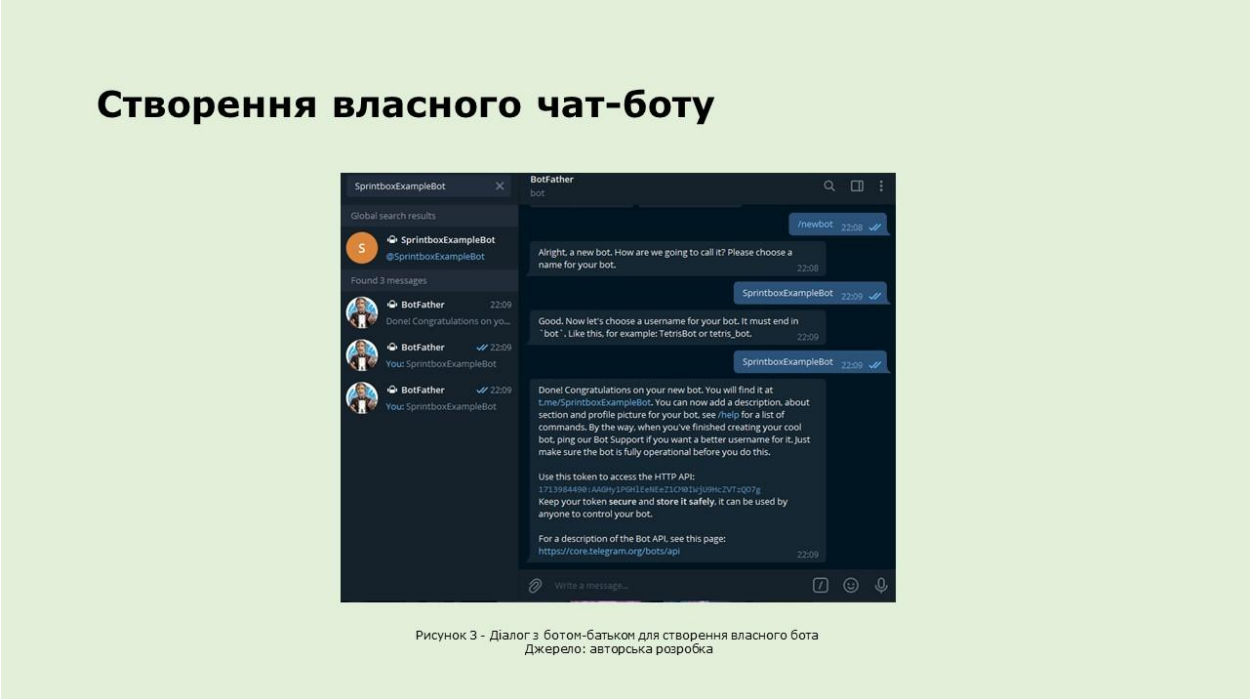

Рисунок 3 - Діалог з ботом-батьком для створення власного бота
Джерело: авторська розробка

Рисунок Г.10 – Створення власного чат-боту

Алгоритм генерування паролів

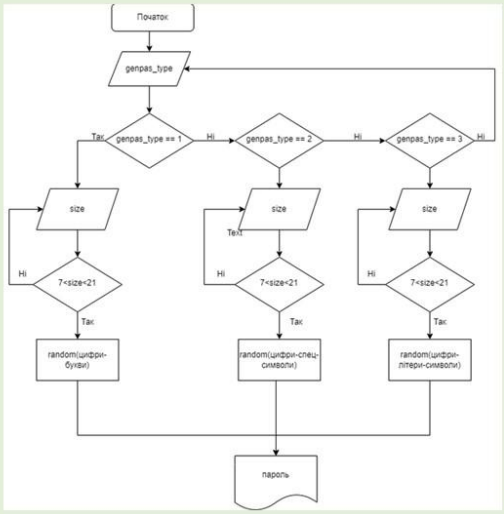

Рисунок 4 – Блок-схема алгоритму генерації паролю
Джерело: авторська розробка

Рисунок Г.11 – Алгоритм генерування паролів

Тестування чат-боту

Загальне тестування – це тести на запитання та відповіді, орієнтовані на загальні питання, на які, як очікується, відповідь найпростіша для боту. Наприклад, перевіряється вітання користувача.

BDD – це відгалуження методології розробки через тестування. Суть BDD полягає в тому, що є бізнес-вимоги до продукту, які описуються предметно-орієнтованою мовою та отримуються сценарії використання системи.

Після-виробниче (або пост-продакшн) тестування або тестування після запуску перевіряє різні аспекти функціональності чат-бота, коли він активний для користувачів.

Рисунок Г.12 – Тестування чат-боту

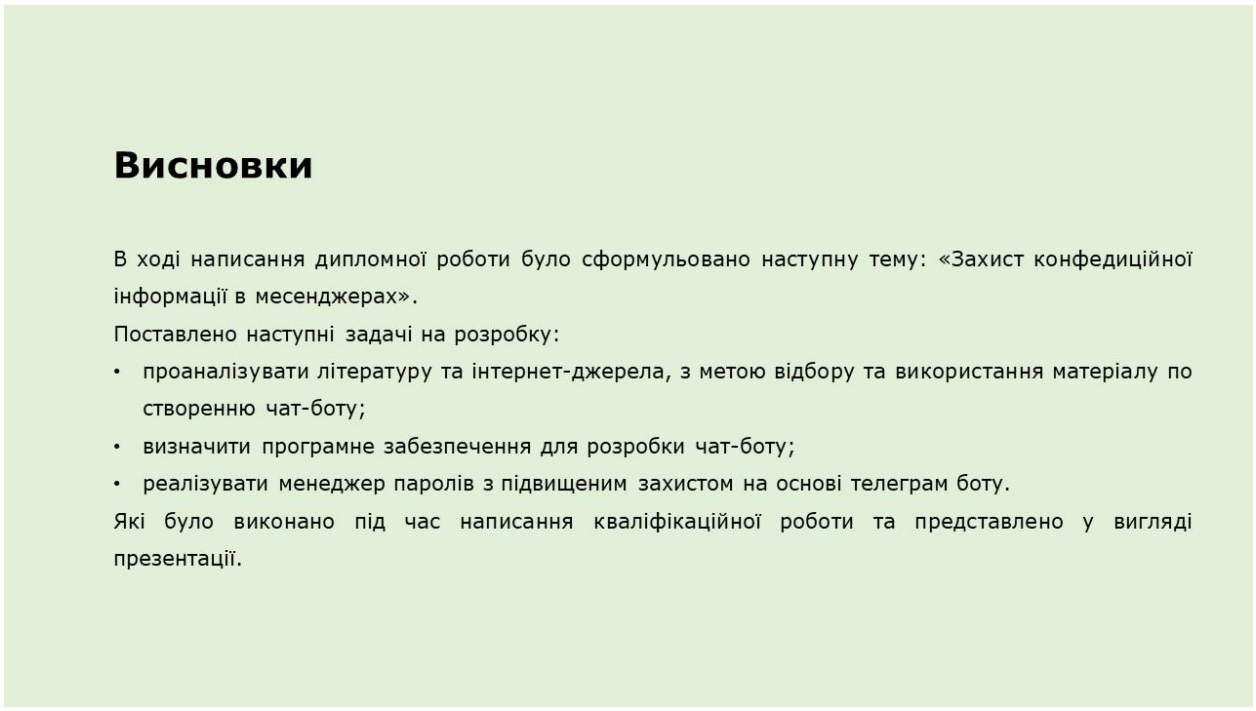

Рисунок Г.13 – Висновки

Рисунок Г.14 – Слайд «Дякую за увагу»