

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

**Пояснювальна записка**

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ  
АВТОМАТИЗОВАНИХ СИСТЕМ ПІДТРИМКИ КОРИСТУВАЧІВ  
RESEARCH AND SOFTWARE IMPLEMENTATION OF  
AUTOMATED USER SUPPORT SYSTEMS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-114м  
Спеціальності 121 Інженерія програмного  
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

НІКУЛІНА А.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЕОЩЕНКО С.Д.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет КНТ  
Кафедра програмних засобів  
Ступінь вищої освіти магістр  
Спеціальність 121 Інженерія програмного забезпечення  
(код і найменування)  
Освітня програма (спеціалізація) Інженерія програмного забезпечення  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.  
Сергій СУББОТІН  
“      ”                                  2025 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

НІКУЛІНОЇ Анастасії Сергіївни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація автоматизованих систем підтримки користувачів.

Research and Software Implementation of Automated User Support Systems.

керівник проєкту (роботи) д-р філос., доцент, ЛЕОЩЕНКО Сергій Дмитрович,  
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “30” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Методи та матеріали. 3. Розробка архітектури програми. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | ПРИЗВИЩЕ, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-5 Основна частина | ЛЕОЩЕНКО С.Д., доцент                     |                |                           |
| Нормоконтролер      | КАЛІНІНА М.В., асистент                   |                |                           |

7. Дата видачі завдання “ 30 ” вересня 2025 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)                           | Строк виконання етапів проєкту ( роботи ) | Примітка     |
|-------|--------------------------------------------------------------------|-------------------------------------------|--------------|
| 1     | Постановка завдання роботи.                                        | 1 тиждень                                 | Завдання, ТЗ |
| 2     | Аналіз проблеми, вибір матеріалів та методів.                      | 2 тиждень                                 | Розділ 1     |
| 3     | Розробка архітектури програми.                                     | 3-4 тижні                                 | Розділ 2     |
| 4     | Основні рішення щодо реалізації компонентів системи.               | 4-7 тижні                                 | Розділ 3     |
| 5     | Експлуатація, тестування та експериментальне дослідження програми. | 8 тиждень                                 | Розділ 4     |
| 6     | Організаційно-економічна частина.                                  | 9 тиждень                                 | Розділ 5     |
| 7     | Оформлення пояснювальної записки та документів до неї.             | 10-11 тижні                               | Додатки      |
| 8     | Нормоконтроль та рецензування.                                     | 12 тиждень                                |              |
| 9     | Захист роботи.                                                     | 12 тиждень                                |              |

Студент(ка)

\_\_\_\_\_  
( підпис ) Анастасія НІКУЛІНА  
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

\_\_\_\_\_  
( підпис ) Сергій ЛЕОЩЕНКО  
(Імя ПРИЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:  
105 с., 2 табл., 32 рис., 4 дод., 16 джерел.

СИСТЕМА ПІДТРИМКИ, МАШИННЕ НАВЧАННЯ,  
АДМІНІСТРУВАННЯ, TELEGRAM, BOT, PYTHON, WEB.

Об'єктом дослідження є процес інформаційної підтримки користувачів у цифрових сервісах, зокрема взаємодія між користувачами та службами підтримки в освітньому середовищі.

Предметом дослідження є методи, алгоритми та програмні засоби побудови автоматизованих систем підтримки користувачів, що поєднують інтелектуальні модулі обробки запитів із механізмами адміністрування та зворотного зв'язку.

Метою роботи є підвищення рівня цифровізації інформаційного обслуговування користувачів, завдяки зменшенню часу реагування на запити та підвищення точності відповідей шляхом створення автоматизованої системи підтримки користувачів з використанням методів обробки природної мови, яка забезпечує автоматичну обробку типових звернень, маршрутизацію складних запитів та накопичення знань.

У процесі виконання роботи використано методи обробки природної мови (NLP), алгоритми семантичного пошуку на основі векторних подань тексту, retrieval-based підхід до формування відповідей, асинхронні методи обробки подій; методи клієнт-серверної взаємодії.

Технічна реалізація системи виконана з використанням: мови програмування Python, фреймворків python-telegram-bot та FastAPI, вебтехнологій HTML, CSS, та JavaScript для створення адміністративного інтерфейсу.

У результаті виконання дипломної роботи було розроблено програмну систему автоматизованої підтримки користувачів у вигляді Telegram-бота з

окремою вебадмінпанеллю. У боті було забезпечено підтримку анонімних та звичайних звернень, автоматичне навчання бази знань на основі відповідей адміністратора, зокрема було впроваджено систему збору та візуалізації статистичних даних. Розроблена система продемонструвала стабільну роботу та зменшення навантаження на адміністратора при обробці типових запитів.

Наукова новизна роботи полягає у поєднанні retrieval-based підходу з механізмом поступового саморозширення бази знань на основі реальних діалогів, а також у застосуванні єдиного централізованого механізму доставки відповідей незалежно від каналу адміністрування.

Практичне значення роботи полягає у можливості використання розробленої системи в освітніх установах, службах підтримки, онлайн-платформах та інших сервісах, де необхідна швидка та ефективна взаємодія з користувачами.

Використання автоматизованої системи підтримки дозволяє зменшити витрати часу адміністративного персоналу, скоротити кількість повторюваних звернень та знизити витрати на утримання служби підтримки. Застосування відкритого програмного забезпечення та недорогих серверних рішень забезпечує низьку собівартість впровадження та експлуатації системи.

Розроблений програмний продукт належить до галузі інформаційних технологій, зокрема до напрямів автоматизованих інформаційних систем, систем підтримки користувачів та інтелектуальних діалогових систем. Основною галуззю використання програмного забезпечення є освітня сфера, але продукт може бути адаптованим у інших сфер.

## ABSTRACT

Explanatory note to the master's thesis: 105 pages, 2 tables, 32 figures, 4 appendices, 16 sources.

SUPPORT SYSTEM, MACHINE LEARNING, ADMINISTRATION, TELEGRAM, BOT, PYTHON, WEB.

The object of the study is the process of information support for users in digital services, in particular the interaction between users and support services in the educational environment.

The subject of the study is methods, algorithms and software tools for building automated user support systems that combine intelligent query processing modules with administration and feedback mechanisms.

The aim of the work is to increase the level of digitisation of information services for users by reducing response times to queries and improving the accuracy of responses through the creation of an automated user support system using natural language processing methods, which provides automatic processing of typical queries, routing of complex queries and knowledge accumulation.

In the course of the work, natural language processing (NLP) methods, semantic search algorithms based on vector text representations, a retrieval-based approach to response generation, asynchronous event processing methods, and client-server interaction methods were used.

The technical implementation of the system was carried out using the Python programming language, the python-telegram-bot and FastAPI frameworks, and HTML, CSS, and JavaScript web technologies to create the administrative interface.

As a result of the thesis work, a software system for automated user support was developed in the form of a Telegram bot with a separate web admin panel. The bot provided support for anonymous and regular requests, automatic training of the knowledge base based on the administrator's responses, and a system for collecting and visualising statistical data was implemented. The developed system

demonstrated stable operation and reduced the load on the administrator v processing typical requests.

The scientific novelty of the work lies in the combination of a retrieval-based approach with a mechanism for the gradual self-expansion of the knowledge base based on real dialogues, as well as in the use of a single centralised mechanism for delivering responses regardless of the administration channel.

The practical significance of the work lies in the possibility of using the developed system in educational institutions, support services, online platforms and other services where fast and effective interaction with users is required.

The use of an automated support system reduces the time spent by administrative staff, reduces the number of repetitive requests, and lowers the cost of maintaining the support service. The use of open source software and inexpensive server solutions ensures low cost of implementation and operation of the system.

The developed software product belongs to the field of information technology, in particular to the areas of automated information systems, user support systems, and intelligent dialogue systems. The main area of application for the software is education, but the product can be adapted for use in other areas.

## ЗМІСТ

|                                                                                  |    |
|----------------------------------------------------------------------------------|----|
|                                                                                  | С. |
| ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК .....                                        | 10 |
| ВСТУП .....                                                                      | 11 |
| 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ .....                                                | 13 |
| 1.1 Обґрунтування проблеми .....                                                 | 13 |
| 1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання<br>..... | 14 |
| 1.2.1 Автоматизація через платформу Manychat [1] .....                           | 14 |
| 1.2.2 Служба турботи Monobank .....                                              | 15 |
| 1.2.3 Порівняння існуючих програмних засобів .....                               | 16 |
| 1.3 Висновки за розділом 1 .....                                                 | 17 |
| 2 МЕТОДИ ТА МАТЕРІАЛИ .....                                                      | 18 |
| 2.1 Вибір інструментарію для розробки програмного забезпечення .....             | 18 |
| 2.1.1 Вибір мов програмування .....                                              | 18 |
| 2.1.2 Вибір бібліотек та фреймворків для розробки .....                          | 21 |
| 2.1.3 Вибір моделі машинного навчання .....                                      | 24 |
| 2.2 Висновок до розділу 2 .....                                                  | 25 |
| 3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ .....                                            | 26 |
| 3.1 Структура програмного продукту .....                                         | 26 |
| 3.2 Опис модулів системи .....                                                   | 27 |
| 3.3 Функціональна схема продукту .....                                           | 28 |
| 3.4 Розробка прототипів інтерфейсів системи .....                                | 28 |
| 3.4.1 Вибір інструментарію для прототипування .....                              | 28 |
| 3.4.2 Розробка прототипів для бота .....                                         | 29 |
| 3.4.3 Розробка прототипів для web-частини .....                                  | 33 |
| 3.5 Висновки за розділом 3 .....                                                 | 35 |
| 4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ                            | 36 |
| 4.1 Характеристика та умови застосування програмного продукту .....              | 36 |
| 4.2 Початкові та вихідні данні .....                                             | 37 |

|                                                                              |     |
|------------------------------------------------------------------------------|-----|
| 4.3 Алгоритм роботи програмного продукту .....                               | 37  |
| 4.3.1 Алгоритм навчання моделі.....                                          | 38  |
| 4.3.2 Алгоритм семантичного пошуку з бази знань .....                        | 38  |
| 4.3.3 Алгоритм донавчання нейромережі .....                                  | 39  |
| 4.3.4 Алгоритм анонімних звернень.....                                       | 40  |
| 4.4 Реалізація безпеки та збереження даних .....                             | 41  |
| 4.5 Висновки до розділу 4 .....                                              | 42  |
| 5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ<br>ДОСЛІДЖЕННЯ ПРОГРАМИ ..... | 43  |
| 5.1 Методика проведення тестування .....                                     | 43  |
| 5.2 Результати використання розробленого програмного забезпечення ....       | 43  |
| 5.2.1 Тестування системи зі сторони звичайного користувача .....             | 43  |
| 5.2.2 Тестування системи з боку адміністратора.....                          | 48  |
| 5.3 Висновки до розділу 5 .....                                              | 54  |
| ВИСНОВКИ.....                                                                | 55  |
| ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....                                                 | 57  |
| ДОДАТОК А Технічне завдання .....                                            | 59  |
| ДОДАТОК Б Текст програми .....                                               | 62  |
| ДОДАТОК В Керівництво оператора .....                                        | 96  |
| ДОДАТОК Г Слайди презентації .....                                           | 100 |

**ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

|        |                                             |
|--------|---------------------------------------------|
| API    | – Application Programming Interface         |
| NLP    | – Natural language processing               |
| TF-IDF | – Term Frequency-Inverse Document Frequency |
| JSON   | – JavaScript Object Notation                |
| JSONL  | – JavaScript Object Notation Lines          |

## ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації освітнього і бізнес-середовища зростає потреба в ефективних засобах автоматизованої підтримки користувачів. Значне навантаження на служби підтримки, велика кількість однотипних запитів і необхідність оперативного надання достовірної інформації визначають актуальність впровадження інтелектуальних автоматизованих систем підтримки користувачів.

Одним із перспективних напрямів у цій сфері є застосування чат-ботів, які використовують методи обробки природної мови та семантичного аналізу тексту. Такі системи забезпечують взаємодію з користувачами у діалоговому форматі, надають відповіді на типові запитання, а у складніших випадках передають звернення оператору або адміністратору. Застосування чат-ботів підвищує ефективність обслуговування, зменшує витрати ресурсів і забезпечує безперервну підтримку користувачів.

Особливу актуальність автоматизовані системи підтримки мають у сфері освіти, де студенти регулярно звертаються з питаннями щодо організаційних, навчальних та адміністративних процесів. Використання інтелектуального чат-бота як помічника студентів дозволяє забезпечити швидкий доступ до інформації, знизити навантаження на адміністративний персонал та покращити загальну якість інформаційної підтримки.

У даній роботі розглядається процес дослідження та програмної реалізації автоматизованої системи підтримки користувачів на основі retrieval-based підходу. Реалізована система поєднує можливості семантичного пошуку, інтеграцію з месенджером Telegram, вебінтерфейс для адміністратора та механізми накопичення й оновлення бази знань. Такий підхід дозволяє створити гнучку, масштабовану та практично придатну систему, яка може бути використана в реальних умовах.

Результати роботи можуть бути використані як основа для подальшого розвитку інтелектуальних систем підтримки користувачів, а також адаптовані для інших предметних областей, де існує потреба в автоматизації комунікації між користувачами та організаціями.

## **1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

### **1.1 Обґрунтування проблеми**

Дипломна робота присвячена дослідженню та програмній реалізації автоматизованих систем підтримки користувачів на основі сучасних інформаційних технологій, методів обробки природної мови та клієнт-серверних архітектур. Зокрема для потреб студентського самоврядування та профкому студентів, для покращення комунікації зі студентами.

Студентське самоврядування є важливим елементом системи управління закладом вищої освіти, який забезпечує представництво інтересів студентів, організацію позанавчальної діяльності, участь у прийнятті управлінських рішень, а також захист прав та інтересів здобувачів освіти. Ефективність діяльності студентського самоврядування значною мірою залежить від якості комунікації між його представниками та студентською спільнотою.

У сучасних умовах основними каналами взаємодії між студентським самоврядуванням та студентами є соціальні мережі, месенджери, офіційні вебсайти закладів освіти, а також безпосереднє спілкування під час заходів або особистих звернень. Однак така система комунікації має низку суттєвих недоліків. Інформація часто розпорошена між різними платформами, оновлюється несистемно та не завжди є доступною для всіх студентів. У результаті значна частина студентів не отримує актуальних повідомлень про події, дедлайни, можливості участі в заходах або зміни в організації навчального процесу.

Окремою проблемою є організація зворотного зв'язку. Звернення студентів до студентського самоврядування зазвичай здійснюється через приватні повідомлення в соціальних мережах, коментарі або особисті контакти з представниками самоврядування. Такий підхід не забезпечує централізованого обліку звернень, ускладнює контроль за їх обробкою та не дозволяє систематизувати проблемні питання, з якими найчастіше

звертаються студенти. Це, у свою чергу, знижує оперативність реагування та ефективність прийняття управлінських рішень.

Також варто зазначити відсутність єдиного інформаційного простору, в якому студенти могли б швидко отримати відповіді на типові запитання щодо діяльності студентського самоврядування, гуртожитків, стипендій, навчального процесу та студентських ініціатив. Повторювані запитання створюють додаткове навантаження на представників самоврядування та призводять до нераціонального використання їхнього часу.

У зв'язку з активним поширенням цифрових технологій та месенджерів, зокрема платформи Telegram, актуальним є використання автоматизованих інформаційних систем для підтримки комунікації зі студентами. Telegram-боти широко застосовуються як інструмент швидкого інформування, збору зворотного зв'язку, проведення опитувань та автоматизації відповідей на типові запити користувачів. Їх використання дозволяє створити централізований канал взаємодії, доступний у будь-який час та з будь-якого пристрою.

## **1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання**

### **1.2.1 Автоматизація через платформу Manychat**

В рамках переддипломної практики було реалізовано автоматизованого бота на платформі Manychat [1], для аналізу необхідності такого бота.

Першочергово, Manychat – це провідна платформа для чат-маркетингу та автоматизації, яка дозволяє брендам і авторам контенту взаємодіяти з аудиторією через Instagram, Telegram, WhatsApp, Facebook Messenger, TikTok, SMS та електронну пошту.

Завдяки простому функціоналу було створено просту автоматизацію для Telegram бота. На сайті це робиться у вигляді блок-схеми із побудовою

функціоналу та наповнення інформації. Приклад побудованої системи наведено на рисунку 1.1.

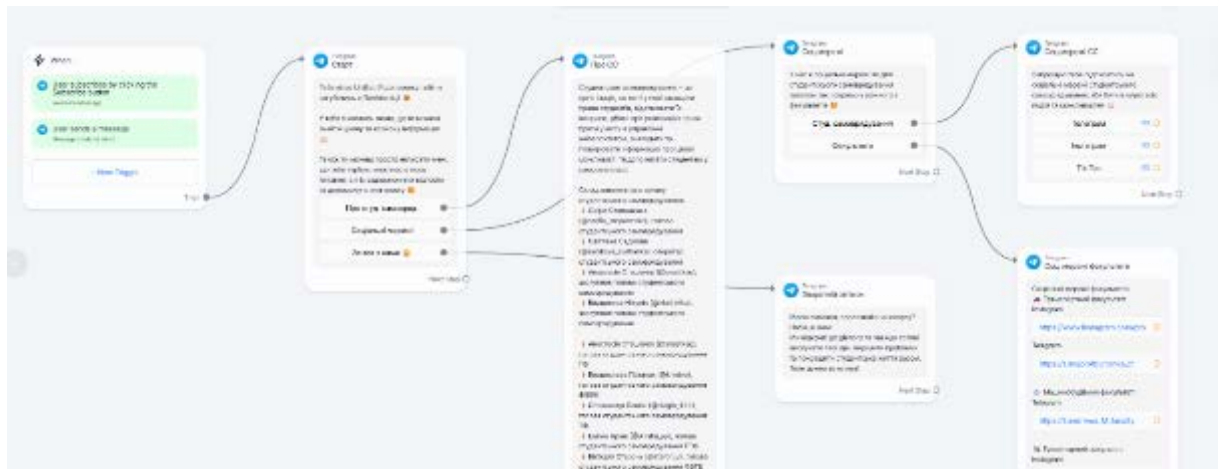


Рисунок 1.1 – Приклад реалізованої автоматизації на платформі Manychat [1]

Головною перевагою даної платформи був простий інтерфейс та інтуїтивно зрозуміла специфіка розробки функціоналу бота.

Недоліком платформи є те, що він не є анонімним, адже анонімність була однією з необхідних функцій даного бота, тож була можливість побачити хто саме звертається до бота.

### 1.2.2 Служба турботи Monobank

Для повноцінної роботи мобільного банкінгу, обов'язково має бути служба підтримки банку. В даному банкінгу окрім гарячої лінії, реалізовано чат-боти у різних месенджерах, зокрема в Telegram [2]

Даний бот є абсолютно безкоштовним, всі звернення фіксуються людьми, без використання автоматизацій відповідей. Приклад чат боту наведено на рисунку 1.2.



Рисунок 1.2 – Чат-бот служби турботи Monobank [2]

### 1.2.3 Порівняння існуючих програмних засобів

У процесі аналізу існуючих систем підтримки користувачів було проведено порівняння чат-бота на платформі ManyChat, служби турботи Monobank та розробленого чат-бота UniBot за основними функціональними характеристиками.

Встановлено, що чат-бот на платформі ManyChat не є безкоштовним рішенням та не підтримує функцію анонімних звернень користувачів, однак забезпечує можливість адміністрування через вебінтерфейс. При цьому платформа використовує елементи машинного навчання для автоматизації обробки запитів.

Служба турботи Monobank є безкоштовною для кінцевих користувачів та підтримує адміністрування через браузер, проте не використовує методи машинного навчання для автоматичної обробки запитів і не надає можливості анонімного звернення до служби підтримки.

Розроблений чат-бот UniBot, на відміну від розглянутих аналогів, поєднує в собі безкоштовність використання, застосування методів машинного навчання для семантичного аналізу запитів, підтримку анонімних звернень користувачів, а також можливість адміністрування як через веббраузер, так і безпосередньо через платформу Telegram. Це забезпечує підвищення зручності керування системою та гнучкість її використання.

### **1.3 Висновки за розділом 1**

В результаті було проведено аналіз предметної області та обґрунтовано необхідність впровадження автоматизованої системи для покращення комунікації між студентським самоврядуванням та студентами. Встановлено, що поточні методи взаємодії через соціальні мережі та особисті повідомлення є неефективними через відсутність централізації, системності оновлень та значне навантаження на адміністраторів через повторювані запити.

На основі проведеного порівняння зроблено висновок про доцільність розробки власного програмного засобу UniBot. Запропонована система вирізняється поєднанням безкоштовного доступу, підтримкою анонімних звернень, гнучким адмініструванням та використанням методів машинного навчання для семантичного аналізу запитів, що дозволяє ефективно вирішити виявлені проблеми комунікації

## **2 МЕТОДИ ТА МАТЕРІАЛИ**

### **2.1 Вибір інструментарію для розробки програмного забезпечення**

#### **2.1.1 Вибір мов програмування**

Одним із ключових етапів розробки автоматизованої системи підтримки користувачів є вибір мови програмування, яка забезпечить ефективну реалізацію поставлених функціональних вимог, масштабованість системи та зручність подальшого супроводу.

Python – це потужна мова програмування, яка легка у вивченні. Він має дієві структури даних високого рівня та простий, але дієвий підхід до об'єктно-орієнтованого програмування. Витончений синтаксис і динамічна типізація разом із його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки застосунків у багатьох сферах на більшості платформ.

Python належить до групи мов високого рівня абстракції з чітким синтаксисом та широкими можливостями для швидкої розробки прикладних рішень. Вона дає можливість зосередитися на логіці, а не на складних деталях низькорівневої реалізації. Серед важливих особливостей, які роблять Python кращим для багатьох розробників, є його величезні бібліотеки та фреймворки, орієнтовані на обробку природної мови, машинне навчання або розробку вебдодатків. Зокрема, в даній роботі використовуються бібліотеки для семантичного аналізу текстів, реалізації алгоритмів пошуку контенту, інтеграції зі сторонніми Application programming interface (API) сервісами та серверними вебінтерфейсами, що дозволить значно скоротити час розробки, одночасно підвищуючи надійність програмного рішення. Python пропонує кросплатформну сумісність. Тож є можливість запускати розроблену систему на будь-якій операційній системі без внесення суттєвих змін до коду [3].

Мова C# є найпопулярнішою мовою для платформи .NET , безкоштовного, кросплатформного середовища розробки з відкритим кодом. Програми на C# можуть працювати на багатьох різних пристроях, від

пристроїв Інтернету речей (IoT) до хмари та будь-де між ними. Ви можете писати програми для телефонів, настільних комп'ютерів, ноутбуків та серверів.

C# — це кросплатформна мова загального призначення, яка робить розробників продуктивними під час написання високопродуктивного коду. З мільйонами розробників C# є найпопулярнішою мовою .NET. C# має широкую підтримку в екосистемі та для всіх робочих навантажень .NET. Базуючись на об'єктно-орієнтованих принципах, вона включає багато функцій з інших парадигм, не в останню чергу функціональне програмування. Низькорівневі функції підтримують високоефективні сценарії без написання небезпечного коду [4].

Java — це мова програмування та обчислювальна платформа, вперше випущена компанією Sun Microsystems у 1995 році. Вона еволюціонувала від скромного початку до того, щоб стати рушійною силою значної частини сучасного цифрового світу, забезпечуючи надійну платформу, на якій побудовано багато сервісів та програм. Нові, інноваційні продукти та цифрові сервіси, розроблені для майбутнього, також продовжують спиратися на Java [5].

Порівняння вищевказаних мов програмування наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльна таблиця мов програмування

| Критерій  | Python                                                                                                           | C#                                                                                                    | Java                                                                                       |
|-----------|------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| 1         | 2                                                                                                                | 3                                                                                                     | 4                                                                                          |
| Синтаксис | Лаконічний та інтуїтивно зрозумілий синтаксис, що зменшує кількість службового коду та полегшує читання програм. | Синтаксис структурований і строгий, вимагає більшої кількості коду для реалізації аналогічної логіки. | Синтаксис строгий та багатослівний, що підвищує надійність, але ускладнює швидку розробку. |

## Продовження таблиці 2.1...

| 1                            | 2                                                                                           | 3                                                                   | 4                                                                                  |
|------------------------------|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Підтримка машинного навчання | Має найбільш розвинену екосистему ML-бібліотек (PyTorch, TensorFlow, SentenceTransformers). | Підтримка машинного навчання обмежена та потребує сторонніх рішень. | Підтримка існує, але значно поступається Python за кількістю та якістю бібліотек.  |
| Обробка природної мови       | Пропонує готові інструменти для NLP, що дозволяє легко реалізувати інтелектуальні чат-боти. | Відсутні потужні нативні NLP-рішення, інтеграція ускладнена.        | Існують NLP-бібліотеки, але вони менш зручні та потребують додаткових налаштувань. |
| Серверна розробка            | Легко використовується з FastAPI та Flask для створення REST-сервісів.                      | Добре підходить для корпоративних серверних застосунків (.NET).     | Широко застосовується для серверних систем великого масштабу.                      |
| Інтеграція з чат-ботами      | Має прямі бібліотеки для Telegram, Discord та інших платформ.                               | Інтеграція можлива через SDK, але потребує додаткової конфігурації. | Інтеграція можлива, але потребує використання сторонніх API.                       |

## Продовження таблиці 2.1...

| 1                         | 2                                                                             | 3                                                            | 4                                                                        |
|---------------------------|-------------------------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------------------|
| Масштабованість           | Добре масштабується при правильній архітектурі та використанні асинхронності. | Висока масштабованість, орієнтована на корпоративні системи. | Дуже висока масштабованість, широко використовується у великих системах. |
| Поріг входу               | Низький, що дозволяє швидко залучати нових розробників.                       | Середній, потребує знання .NET-екосистеми.                   | Високий, потребує глибокого розуміння ООП та JVM.                        |
| Документація та спільнота | Одна з найбільших спільнот та обсягів навчальних матеріалів.                  | Добре розвинена документація від Microsoft.                  | Потужна, але більш орієнтована на корпоративних розробників.             |

У даному проєкті для реалізації автоматизованої системи підтримки користувачів було обрано мову програмування Python, оскільки вона є однією з найефективніших мов для розробки інтелектуальних систем, що використовують методи машинного навчання та обробки природної мови. Python має розвинену екосистему бібліотек для роботи з текстовими даними, зокрема для семантичного аналізу, векторизації та порівняння текстів, що є ключовими задачами у функціонуванні чат-бота. Це дозволяє значно скоротити час розробки та підвищити точність обробки запитів користувачів.

### 2.1.2 Вибір бібліотек та фреймворків для розробки

Для реалізації взаємодії з користувачами через месенджер Telegram було використано бібліотеку `python-telegram-bot`. Вона надає зручний

програмний інтерфейс для створення Telegram-ботів, обробки команд, текстових повідомлень та подій користувачів.

Ця бібліотека надає чистий асинхронний інтерфейс на Python для API бота Telegram. Окрім чистої реалізації API, ця бібліотека містить кілька зручних методів та скорочень, а також низку високорівневих класів, що роблять розробку ботів легкою та зрозумілою [6].

Для реалізації інтелектуального пошуку відповідей було використано бібліотеку SentenceTransformers, яка базується на сучасних трансформерних моделях. Sentence Transformers – це основний модуль Python для доступу, використання та навчання сучасних моделей вбудовування та переранжування. Його можна використовувати для обчислення вбудовування за допомогою моделей Sentence Transformer, для обчислення оцінок подібності за допомогою моделей Cross-Encoder (також відомих як переранжування), або для генерації розріджених вбудовування за допомогою моделей Sparse Encoder. Це відкриває широкий спектр застосувань, включаючи семантичний пошук, семантичну текстову подібність та аналіз перефраз [7].

Для реалізації серверної частини вебінтерфейсу адміністратора було обрано фреймворк FastAPI. Даний фреймворк призначений для створення високопродуктивних вебAPI та базується на асинхронній моделі обробки запитів. Основною перевагою FastAPI є висока швидкодія, простота використання та підтримка сучасних стандартів взаємодії клієнт-серверних застосунків [8].

FastAPI – це сучасний, швидкий (високопродуктивний) вебфреймворк для створення API за допомогою Python на основі стандартних підказок типів Python. Ключовими характеристиками фреймворку є:

- має дуже високу продуктивність, тож вважається одним з найшвидших фреймворків python;
- інтуїтивно зрозумілий, завдяки чудовій підтримці редактора та потребує менше часу на налагодження;
- зручний для використання та навчання;

- мінімізує дублювання коду, має кілька функцій з кожного оголошення параметра, через що менше помилок;
- надійний та готовий до використання код, з автоматичною інтерактивною документацією [9].

Для зберігання та обробки інформації про запитання, відповіді та статистику роботи системи використовувався формат JSON (JavaScript Object Notation).

Формат JSON використовується для впорядкованого зберігання даних у процесі їх обміну між веббраузером (або клієнтською частиною програми) та сервером (або між різними серверами). Більше того, завдяки текстовому вигляду рядка дані JSON можна легко передавати через будь-які інші канали обміну інформацією в інтернеті [10].

Стандартна бібліотека Python для роботи з JSON забезпечує простий та надійний механізм серіалізації і десеріалізації даних, що є достатнім для прототипу та навчального проєкту.

Переваги використання FastAPI:

- висока продуктивність і асинхронність, адже FastAPI побудований на базі ASGI та використовує асинхронну модель обробки запитів, що дозволяє ефективно працювати з великою кількістю одночасних звернень користувачів. Це особливо важливо для систем підтримки, де запити надходять у реальному часі;
- FastAPI природно інтегрується з екосистемою Python, зокрема з бібліотеками машинного навчання, обробки природної мови та асинхронними сервісами, що робить його оптимальним вибором для інтелектуальних чат-ботів;
- завдяки використанню Pydantic, FastAPI забезпечує автоматичну перевірку вхідних і вихідних даних, що підвищує надійність системи та зменшує кількість помилок при обробці запитів;

- FastAPI має мінімалістичну архітектуру та зрозумілу структуру коду, що дозволяє швидко розгортати застосунок і легко масштабувати його в майбутньому без суттєвих змін у логіці системи.

### **2.1.3 Вибір моделі машинного навчання**

Під час вибору моделі були розглянуті кілька основних підходів до реалізації інтелектуального модуля підтримки користувачів:

- класичні методи обробки тексту. До цієї групи належать підходи на основі Term Frequency-Inverse Document Frequency (TF-IDF), N-грам та методів машинного навчання, таких як наївний баєсівський класифікатор або метод опорних векторів;

- генеративні моделі, які на основі великих мовних моделей здатні формувати природні відповіді, проте вони потребують значних обчислювальних ресурсів, складні в контролі результатів та можуть генерувати некоректну або небажану інформацію. Для системи підтримки користувачів, де важлива точність та контроль відповідей, такий підхід є менш доцільним;

- семантичний пошук на основі векторних подань. Даний підхід передбачає перетворення текстових запитів у векторні представлення з подальшим порівнянням їхньої подібності з базою знань. Він поєднує високу точність, швидкодію та можливість контролю джерел відповідей.

У даному проєкті було обрано модель семантичного пошуку на основі бібліотеки SentenceTransformers. Зокрема, використано багатомовну модель paraphrase-multilingual-MiniLM-L12-v2, яка побудована на архітектурі трансформерів. Це модель трансформації речень, вона відображає речення та абзаци у 384-вимірний щільний векторний простір і може використовуватися для таких завдань, як кластеризація або семантичний пошук [11].

## 2.2 Висновок до розділу 2

В даному розділі було обрано методи та матеріали для розробки програмного продукту.

У якості основної мови програмування обрано Python, що зумовлено його високим рівнем абстракції, кросплатформністю та наявністю потужних бібліотек для машинного навчання і семантичного аналізу текстів.

Технологічний стек проєкту включає наступні ключові компоненти:

- бібліотеку `python-telegram-bot` для забезпечення асинхронної взаємодії з користувачами через месенджер;
- бібліотеку `SentenceTransformers` для реалізації інтелектуального пошуку відповідей;
- фреймворк `FastAPI` для створення високопродуктивної серверної частини та вебінтерфейсу адміністратора;
- стандартну бібліотеку для роботи з форматом `JSON`, що забезпечує надійне збереження та обробку даних системи.

Для реалізації інтелектуального модуля було обрано метод семантичного пошуку на основі векторних подань, оскільки він поєднує високу точність та можливість контролю відповідей, на відміну від генеративних моделей. Практична реалізація цього підходу здійснюється за допомогою бібліотеки `SentenceTransformers` та моделі `paraphrase-multilingual-MiniLM-L12-v2`, яка ефективно виконує трансформацію речень у векторний простір для пошуку подібності.

Обраний набір інструментів дозволяє зосередитися на логіці продукту та забезпечує масштабованість, швидку розробку і зручність супроводу системи.

## **3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ**

### **3.1 Структура програмного продукту**

Перед початком розробки необхідно визначити структуру програмного продукту, тому далі визначимо основні компоненти майбутнього застосунку.

Програмний продукт має складатись із чат-бота на платформі Telegram та вебсервісу.

Чат-бот виступає головною частиною проєкту, адже саме через чат користувачі мають отримувати відповіді від нейромережі чи адміністратора. Зокрема необхідно виділити функціонал чат-бота:

- автоматична відповідь на повідомлення «\start» із описом функціоналу бота;
- автоматичні відповіді після натискання на кнопки;
- налаштування анонімних звернень до бота;
- налаштування звернення до адміністратора;
- відповідати на звернення користувача, які були знайдені у базі знань;
- передавати адміністратору звернення користувачів, які не були знайдені у базі знань.

Для адміністративної підтримки чат-боту повинен бути вебсервіс, з яким буде вестись зв'язок через API. Зокрема вебсервіс має наступний функціонал:

- авторизація для адміністраторів;
- відповіді на звернення користувачів, на які не змогла відповісти нейромережа самостійно;
- відображення статистики користуванням ботом;
- видалення повідомлень;
- редагування базової інформації, яку користувач отримує після натискання на кнопки.

### 3.2 Опис модулів системи

В таблиці 3.1 наведено опис модулів розроблюваної системи, де також визначено функціонал даних модулів.

Таблиця 3.1 – Опис модулів системи

| Модуль                                               | Опис                                                                                                                                                                |
|------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Реагування ботом на кнопки.                          | Бот отримує повідомлення про натискання кнопки користувачем та має надати відповідну базову інформацію.                                                             |
| Реагування ботом на звернення користувачів           | 1. Якщо за контекстом звернення було знайдено в базі знань – бот надсилає відповідь користувачу.                                                                    |
|                                                      | 2. Якщо за контекстом звернення не було знайдено в базі знань – бот надсилає питання адміністратору та сповіщає користувача, що питання в обробці у адміністратора. |
| Реагування адміністратором на звернення користувачів | Через вебсервіс або в чат-боті відповідає на звернення – після натискання кнопки користувачу приходить відповідь адміністратора від імені бота.                     |
| Видалення повідомлень                                | Через вебсервіс або в чат-боті адміністратор може видалити повідомлення, якщо воно не стосується теми бота, аби не засмічувати базу знань.                          |
| Виведення статистичних даних                         | У відповідному вікні виводиться загальна інформація про користування ботом.                                                                                         |
| Редагування базової інформації                       | У відповідному вікні вебпанелі адміністратор може змінювати базову інформацію, якщо та вже не є актуальною                                                          |

### 3.3 Функціональна схема продукту

Після визначення структури необхідно розробити функціональну схему проєкту. Схему наведено на рисунку 3.1.

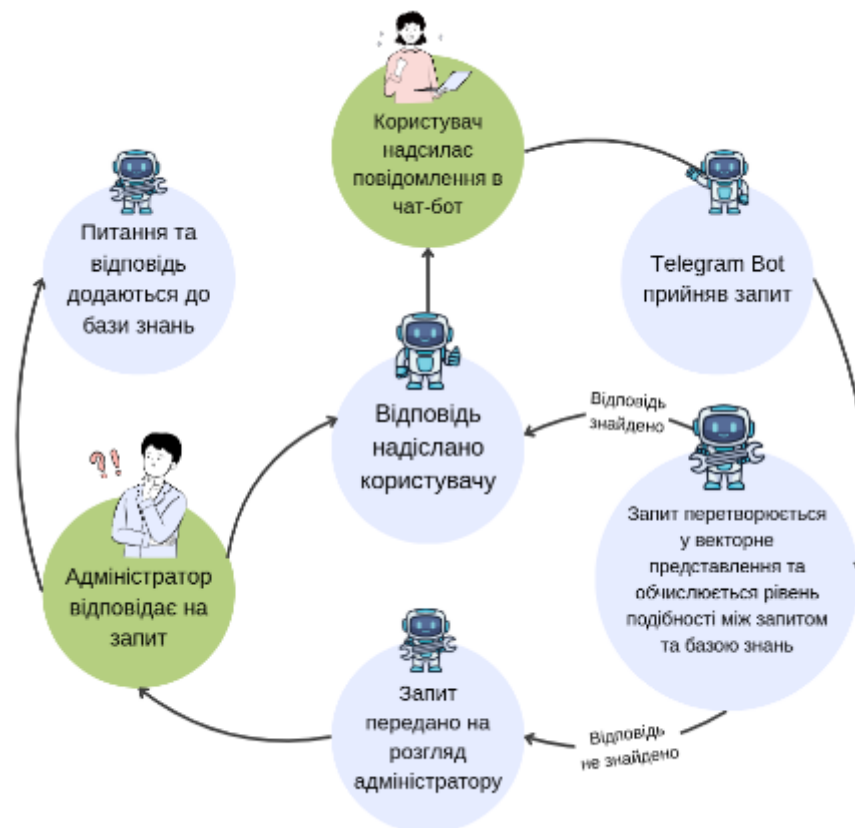


Рисунок 3.1 – Функціональна схема проєкту

### 3.4 Розробка прототипів інтерфейсів системи

#### 3.4.1 Вибір інструментарію для прототипування

Прототипування — це процес створення попередньої моделі або зразка продукту для перевірки його функціональності, дизайну та взаємодії з користувачем перед повномасштабним виробництвом. Це дозволяє дизайнерам оцінювати та вдосконалювати продукт на основі відгуків користувачів, потенційно економлячи час і ресурси.

У розробці програмного забезпечення прототип може бути базовою версією системи, створеної для тестування та переробки, доки не буде досягнуто прийнятний прототип [12].

Figma — це застосунок для роботи із векторною графікою, що дозволяє створювати різні макети, елементи інтерфейсу, зображення, ілюстрації, проєктування і багато інших креативів. Цей інструмент популярний серед веб- та UI/UX-дизайнерів, SMM-менеджерів, stories-мейкерів та інших digital-експертів. Бо він зручний для використання, доступний і тут можливо працювати над проєктом віддалено разом з іншими учасниками [13].

### 3.4.2 Розробка прототипів для бота

Перед початком роботи бота має бути кнопка «Розпочати» та короткий опис про бота (рис. 3.2).



Рисунок 3.2 – Вигляд бота перед початком роботи

Для початку роботи з ботом, користувач має натиснути кнопку «Розпочати» (рис. 3.3).

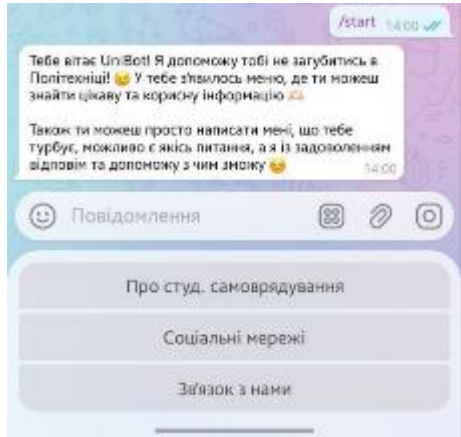


Рисунок 3.3 – Початок роботи з ботом

При натисканні на кнопку меню «Про студ. самоврядування» бот має надіслати інформацію про склад студентського самоврядування (рис. 3.4).

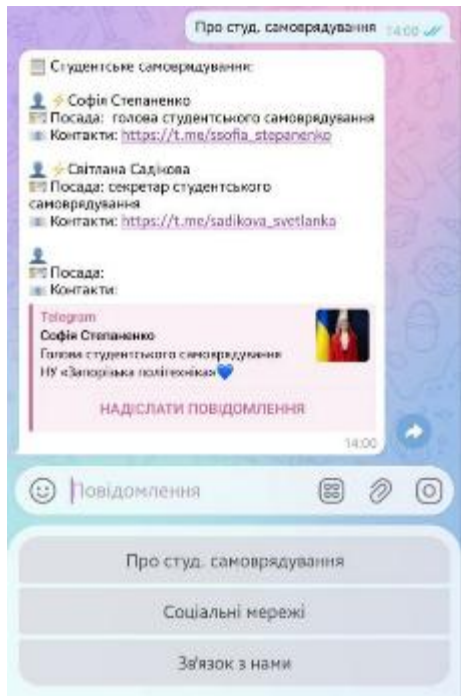


Рисунок 3.4 – Повідомлення про склад студентського самоврядування

При натисканні на кнопку меню «Соціальні мережі» бот має надіслати інформацію про соціальні мережі (рис. 3.5).

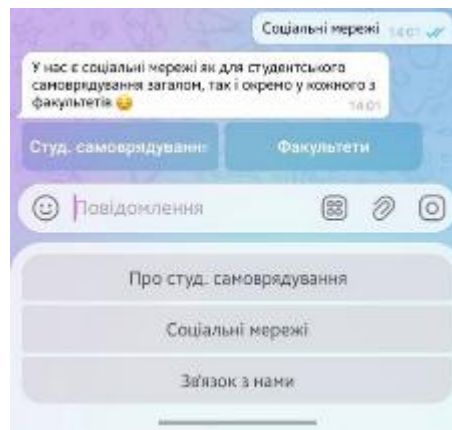


Рисунок 3.5 – Вибір конкретних соціальних мереж

При натисканні на кнопку в повідомленні «Студ. самоврядування» бот надсилає інформацію про соціальні мережі студ. самоврядування (рис. 3.6).

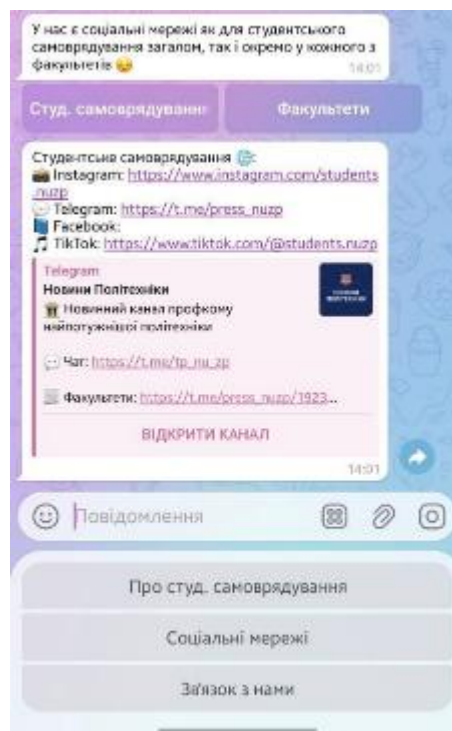


Рисунок 3.6 – Виведення соціальних мереж

При натисканні на кнопку в повідомленні «Факультети» (рис. 3.4) бот має надіслати інформацію про соціальні мережі студентського самоврядування (рис. 3.7).



Рисунок 3.7 – Виведення соціальних мереж

При натисканні на кнопку меню «Зв'язок з нами» бот має надіслати інформаційне повідомлення, що може приймати будь які звернення від користувачів (рис. 3.8).

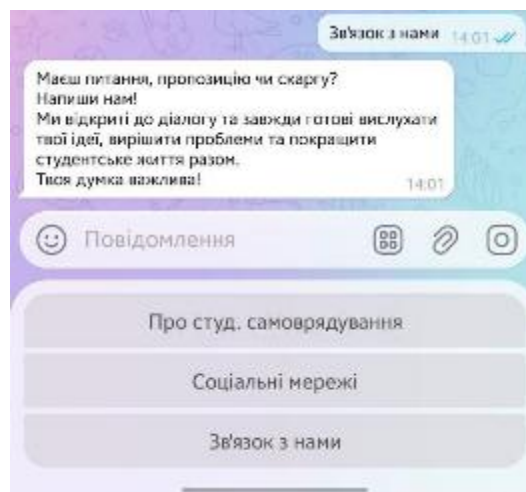


Рисунок 3.8 – Повідомлення про прийом звернень

### 3.4.3 Розробка прототипів для web-частини

Для входу на сайт адміністратор має авторизуватись, для цього він має ввести пароль (рис. 3.9).



Рисунок 3.9 – Сторінка авторизації

Після успішної авторизації перед адміністратором буде сторінка із повідомленнями, якщо в системі будуть повідомлення на які необхідна відповідь адміністратора, вони будуть у списку на цій сторінці (рис. 3.10).

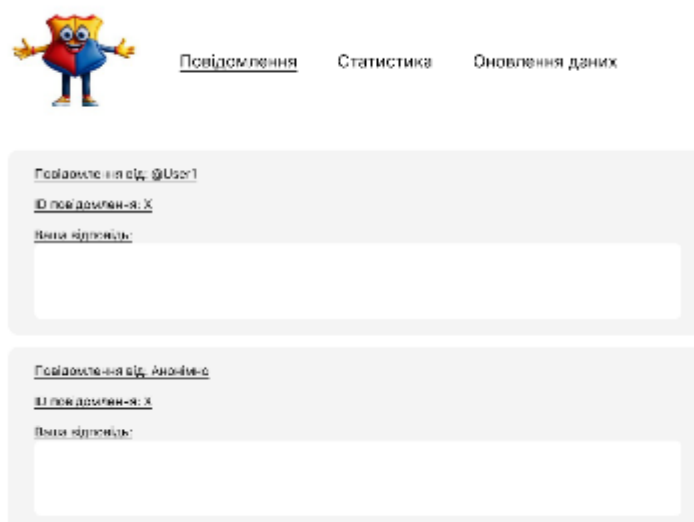


Рисунок 3.10 – Сторінка з повідомленнями, є необроблені повідомлення

Якщо ж в системі не буде знайдено звернень без відповідей, на сторінці буде виведено текст про відсутність запитів (рис. 3.11).

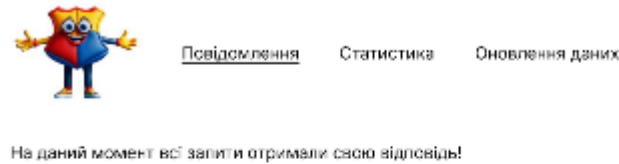


Рисунок 3.11 – Сторінка з повідомленнями, повідомлень немає

Натиснувши на кнопку «Статистика» у верхній частині сторінки, адміністратор отримає статистичні данні про користування ботом (рис. 3.12).

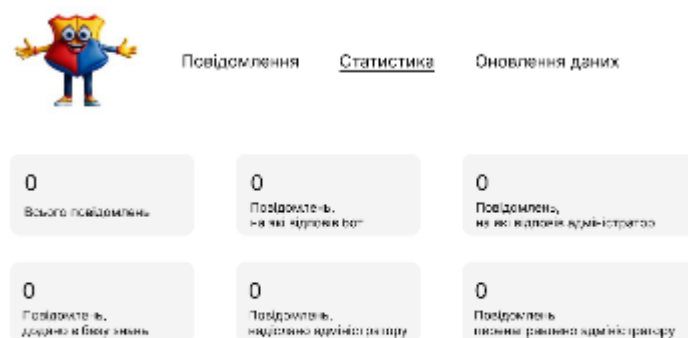


Рисунок 3.12 – Сторінка статистики користуванням ботом

Натиснувши на кнопку «Оновлення даних» у верхній частині сторінки, адміністратор отримає змогу редагувати базову інформацію, а саме відповіді бота, які надсилаються користувачам після натискання ним кнопок (рис. 3.13).

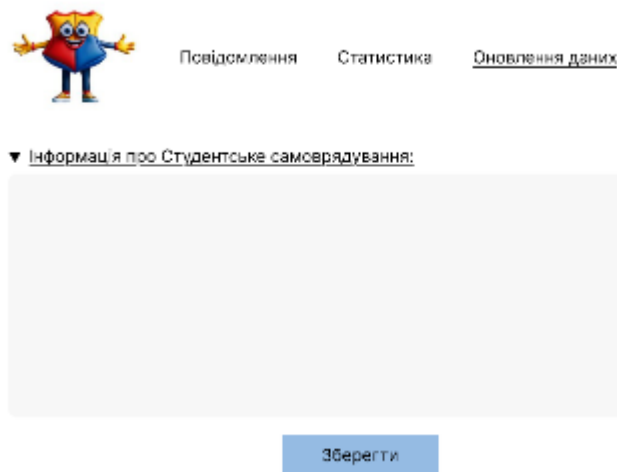


Рисунок 3.13 – Сторінка «Оновлення даних»

### 3.5 Висновки за розділом 3

У розділі визначено структуру програмного продукту, що складається з Telegram-бота для взаємодії з користувачами та вебсервісу для адміністративного керування. Описано функціональні модулі системи, які забезпечують автоматичну обробку запитів за допомогою нейромережі, передачу нерозпізнаних звернень адміністраторам, а також інструменти для перегляду статистики та редагування контенту.

Зокрема було розроблено прототипи розроблюваної системи, з описом того, як має працювати система для різних типів користувачів (звичайний користувач та адміністратор).

## **4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ**

### **4.1 Характеристика та умови застосування програмного продукту**

Програма реалізована у вигляді Telegram бота, що працює за принципом семантичного пошуку відповідей.

Система підтримує два режими функціонування:

- автоматичний режим, де система знаходить релевантну відповідь у базі та одразу надсилається користувачу.
- режим донавчання, де відповідь не знайдено, запит переадресовується адміністратору, і після того як адміністратор надає рішення, це питання та відповідь додаються до бази знань, що дозволяє системі розпізнавати такі звернення в майбутньому.

Для використання чат-бота кінцевому користувачу не потрібне встановлення спеціалізованого програмного забезпечення.

Мінімальні вимоги до пристроїв користувача:

- наявність пристрою з доступом до мережі Інтернет;
- встановлений Telegram клієнт (мобільний додаток або вебверсія);
- будь-яка сучасна операційна система (Android, iOS, Windows, macOS, Linux);
- можливість обміну текстовими повідомленнями.

Уся обчислювальна логіка виконується на серверній частині, що робить вимоги до клієнтських пристроїв мінімальними.

Адміністратор взаємодіє із системою через Telegram (для отримання повідомлень та відповідей) або вебадмінпанель, доступну через браузер.

Мінімальні вимоги до пристроїв адміністратора:

- персональний комп'ютер або мобільний пристрій;
- сучасний веббраузер (Google Chrome, Firefox, Edge тощо);
- стабільне підключення до мережі Інтернет;
- доступ до Telegram-акаунта адміністратора.

Серверна частина програмного продукту забезпечує роботу Telegram бота, бази знань, модуля машинного навчання та вебадмінпанелі.

Мінімальні апаратні вимоги до сервера:

- процесор 1–2 vCPU;
- оперативна пам'ять 4 ГБ для стабільної роботи;
- дисковий простір від 10 ГБ;
- постійне підключення до мережі Інтернет.

Програмні вимоги до сервера:

- операційна система: Linux (Ubuntu 20.04 / 22.04 або аналогічна);
- Python версії 3.10 або вище;
- підтримка асинхронного виконання (asyncio);
- вебсервер для запуску FastAPI (наприклад, Uvicorn);
- можливість запуску фонових задач;
- доступ до файлової системи для збереження JSON/JSONL файлів.

## **4.2 Початкові та вихідні данні**

Початковими даними бота є база знань, яка була зроблена на основі офіційного сайту університету.

Після того як користувач надсилає свій запит до бота, починається обробка бази знань. В залежності наявності необхідних даних у базі знань відповідь надає бот чи адміністратор.

Вихідними даними системи є відповідь яка надішла користувачу від бота чи адміністратора.

## **4.3 Алгоритм роботи програмного продукту**

Головною логікою системи є обробка запитів користувачів, аналізуючи наявну базу знань. Далі в залежності від результату пошуку, надсилати відповідь користувачі, або ж передавати запит адміністратору, який в свою

чергу відповідає на запит. Відповідь надсилається користувачу та зберігається у базу знань, аби в подальшому бот мав змогу відповідати на такі запити.

#### **4.3.1 Алгоритм навчання моделі**

Початковий файл з базою знань необхідно було створювати самостійно, через нестачу ресурсів для створення її автоматично. Тут було розписано близько 70 прикладів питань з відповідями, які стосуються різних сфер діяльності університету, та якими потенційно могли б цікавитись студенти. Потім, за допомогою нейромережі Chat GPT [14] було урізноманітнено базу, адже нейромережа сформувала ще близько 200 записів.

#### **4.3.2 Алгоритм семантичного пошуку з бази знань**

Семантичний пошук – це метод пошуку даних, який виходить за рамки традиційних методів пошуку за ключовими словами. Він використовує обробку природної мови та алгоритми машинного навчання для підвищення точності результатів пошуку, враховуючи наміри користувача і контекстуальне значення термінів, використаних у запиті.

На відміну від класичних пошукових підходів, що базуються на лексичному аналізі або ключових словах, семантичний пошук дозволяє враховувати контекст, синонімію, формулювання та мовні варіації запитів, що є особливо важливим у діалогових системах [15].

Вхідні дані алгоритму:

- текстове повідомлення користувача;
- база знань, що містить пари «запит – відповідь», збережені у форматі JSONL.

Вихідні дані алгоритму:

- найбільш релевантна відповідь з бази знань або;

– сигнал про недостатню впевненість у відповіді (передавання питання адміністратору).

Для реалізації семантичного пошуку використовується метод векторного подання тексту. Кожне текстове повідомлення перетворюється у числовий вектор фіксованої довжини, який відображає семантичний зміст тексту.

Векторизація здійснюється за допомогою попередньо навченої багатомовної трансформерної моделі SentenceTransformer, яка дозволяє працювати з текстами різними мовами та забезпечує високу якість семантичного представлення.

У процесі ініціалізації системи всі питання з бази знань перетворюються у векторні представлення та зберігаються в оперативній пам'яті для пришвидшення подальших обчислень.

#### **4.3.3 Алгоритм донавчання нейромережі**

Донавчання розробленої автоматизованої системи підтримки користувачів реалізовано у вигляді поступового розширення бази знань без повторного навчання нейронної моделі. Такий підхід дозволяє системі поступово підвищувати якість відповідей та адаптуватися до нових типів запитів у процесі експлуатації. Основним джерелом даних для донавчання є питання користувачів, на які система не змогла надати автоматичну відповідь та відповіді, надані адміністратором.

Алгоритм донавчання системи складається з таких основних етапів:

- 1) Фіксація невпевненого запиту, якщо коефіцієнт семантичної подібності між запитом користувача та існуючими записами бази знань нижчий за заданий поріг, система визнає відповідь невпевненою та передає питання адміністратору;

- 2) Отримання відповіді від адміністратора;

3) Валідація та обробка даних, де текст питання та відповіді проходить базову нормалізацію (очищення, усунення зайвих символів, переносів рядків і т.д.);

4) Додавання до бази знань, пара «питання – відповідь» зберігається у файлі бази знань у форматі JSONL (JavaScript Object Notation Lines);

5) Оновлення семантичних векторів після додавання нових даних, тоді здійснюється повторна векторизація всіх питань бази знань, що забезпечує актуальність семантичного простору.

#### **4.3.4 Алгоритм анонімних звернень**

Алгоритм анонімних звернень призначений для забезпечення можливості надсилання користувачами запитів до адміністрації без розкриття їхньої особистості. Такий функціонал є важливим у системах підтримки користувачів, зокрема в освітньому середовищі, де користувач може не бажати публічно асоціювати себе з певним питанням.

Основною вимогою до алгоритму є повне приховування ідентифікаційних даних користувача від адміністратора, при одночасному збереженні технічної можливості надати відповідь саме тому користувачеві, який ініціював звернення.

Алгоритм обробки анонімних звернень складається з таких послідовних етапів:

1) Спочатку користувач активує анонімний режим за допомогою відповідної команди в Telegram боті;

2) Система фіксує стан користувача, ідентифікатор користувача тимчасово зберігається у внутрішній структурі даних як такий, що очікує введення анонімного повідомлення;

3) Наступне повідомлення користувача автоматично інтерпретується як анонімне звернення;

- 4) Звернення зберігається у сховищі питань із відповідною позначкою анонімності, без запису імені користувача або його Telegram-нікнейму;
- 5) Адміністратор отримує повідомлення з позначкою «анонімне», яке містить лише текст питання та службовий ідентифікатор звернення;
- 6) Адміністратор формує відповідь, не маючи доступу до даних про автора звернення;
- 7) Відповідь автоматично надсилається користувачу за збереженим ідентифікатором, при цьому анонімність повністю зберігається.

#### **4.4 Реалізація безпеки та збереження даних**

Під час проєктування та реалізації автоматизованої системи підтримки користувачів основну увагу було приділено забезпеченню цілісності, конфіденційності та доступності даних. Система побудована за принципом мінімізації доступу, відповідно до якого кожен компонент має доступ лише до тих даних і функцій, які необхідні для виконання його завдань.

Архітектура системи передбачає розділення відповідальностей між компонентами, що зменшує ризики витоку даних та помилок у роботі.

Система не зберігає надлишкові персональні дані користувачів. Для ідентифікації користувачів застосовуються лише службові ідентифікатори, які надаються платформою Telegram. Імена користувачів або їхні облікові записи не зберігаються у базі знань і не використовуються для навчання моделі.

Для збереження інформації у системі використовуються файлові сховища у форматах JSON та JSONL. Операції читання та запису файлів виконуються з використанням механізмів синхронізації, що запобігає одночасному доступу з різних потоків і виникненню пошкоджених даних.

#### 4.5 Висновки до розділу 4

У даному розділі описано реалізацію програмного продукту у вигляді Telegram-бота, в основу якого покладено технологію семантичного пошуку з використанням векторних подань текстів.

Детально розглянуто алгоритми функціонування системи, зокрема:

- механізм донавчання, який дозволяє розширювати базу знань шляхом обробки адміністратором запитів, на які бот не зміг відповісти автоматично.
- алгоритм анонімних звернень, що забезпечує конфіденційну комунікацію користувача з адміністрацією без розкриття особистих даних.

Також у розділі визначено технічні вимоги до серверної частини (Python, FastAPI, Linux) та клієнтських пристроїв, описано заходи безпеки, що базуються на принципах мінімізації доступу та безпечного збереження даних у форматі JSON.

## **5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **5.1 Методика проведення тестування**

Для тестування системи було обрано дослідницьке тестування.

Дослідницьке тестування – це особливий тип тестування програмного забезпечення, який має багато переваг для програми, дозволяючи їй повністю розкрити свій потенціал.

Те, як команда інтегрує дослідницьке тестування в свої рутинні перевірки, може навіть визначити, наскільки добре функціонує програмне забезпечення, особливо якщо це підходить до процедур тестування новими і несподіваними способами. Це допомагає тестувальникам виявити проблеми в додатку, які в іншому випадку можуть залишитися непоміченими до запуску і призвести до того, що ключові функції не працюватимуть [16].

Для багато-користувацької системи такий вид тестування буде найбільш прийнятним, адже користуватись системою будуть найрізноманітніші люди, у кожного своя манера спілкування з чат-ботами, що звичайними тестами не проаналізуєш.

### **5.2 Результати використання розробленого програмного забезпечення**

#### **5.2.1 Тестування системи зі сторони звичайного користувача**

На початку роботи з ботом бачимо інформаційну панель, де вказано основний функціонал бота (рис. 5.1).



Рисунок 5.1 – Бот перед початком роботи

Після початку роботи з ботом, було надіслано початкове повідомлення з додатковим інформуванням, зокрема з'явилося інтераактивне меню з кнопками (рис. 5.2).



Рисунок 5.2 – Початкове повідомлення бота

Перше що базово перевіряється це кнопки, тож після натискання на кнопку «Про студентське самоврядування», бот надіслав базову інформацію актуального складу студентського самоврядування (рис. 5.3).

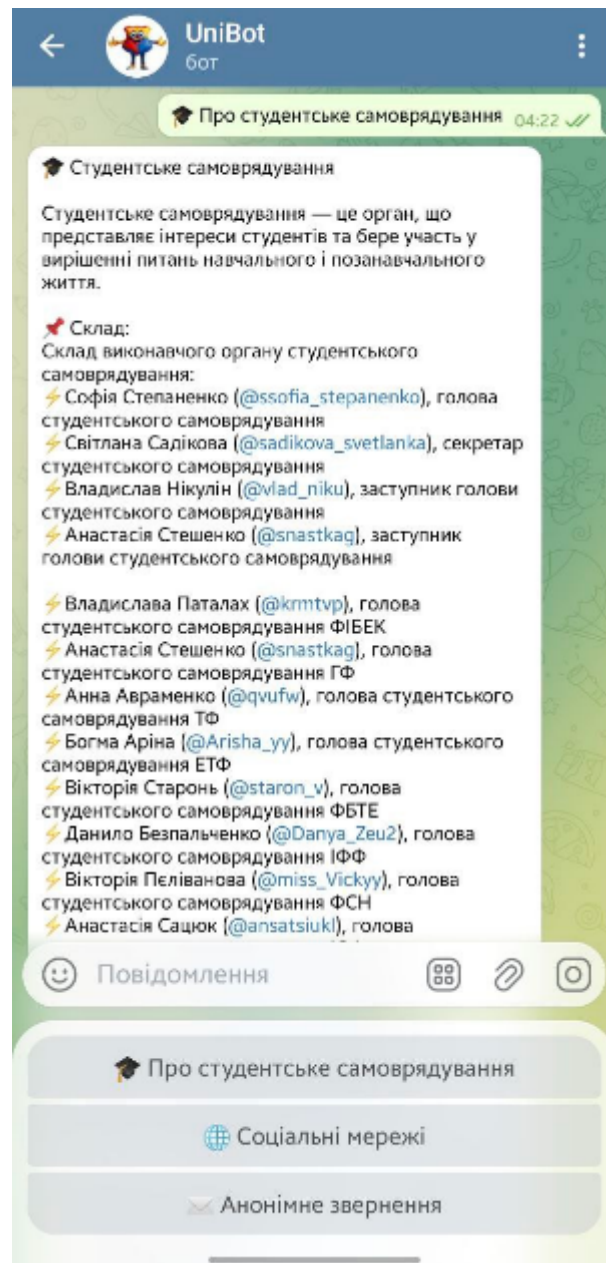
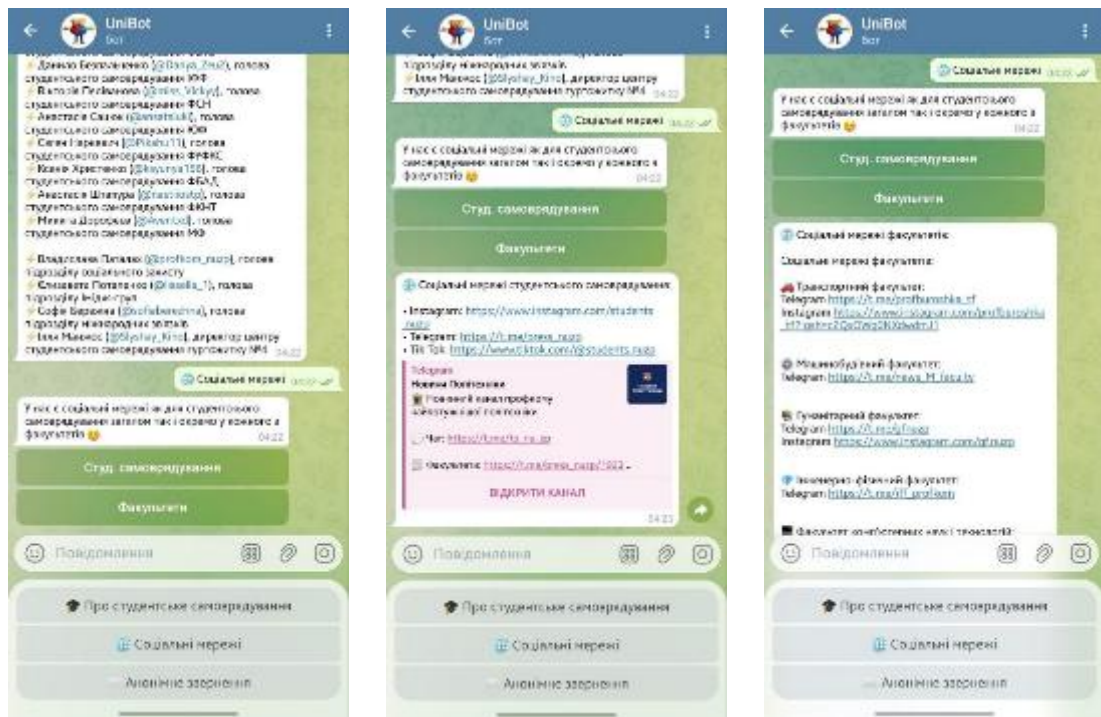


Рисунок 5.3 – Повідомлення про склад студентського самоврядування

Далі було протестовано кнопку «Соціальні мережі», при натисканні данної кнопки меню було виведено повідомлення з 2 кнопками (рис. 5.4а), де можна обрати соціальні мережі «Студентського самоврядування» та «Факультетів», після натискання кожної було видано відповідні дані про соціальні мережі (рис. 5.4б та рис.5.4в).



а

б

в

а - кнопка «Соціальні мережі»; б - кнопка «Студентське самоврядування»; в - кнопка «Факультети».

Рисунок 5.4 – Результат натискання на кнопки

Далі я тестувала бота різними питаннями, які могли б стосуватись університету (рис. 5.5).



Рисунок 5.5 – Відповіді бота на питання

Потім було задано питання, яке не виявилось у базі знань та було перенаправлене адміністратору (рис. 5.6).



Рисунок 5.6 – Повідомлення передано адміністратору

Для тестування функції анонімності необхідно натиснути на відповідну кнопку меню «Анонімне звернення» та написати своє анонімне повідомлення (рис. 5.7).

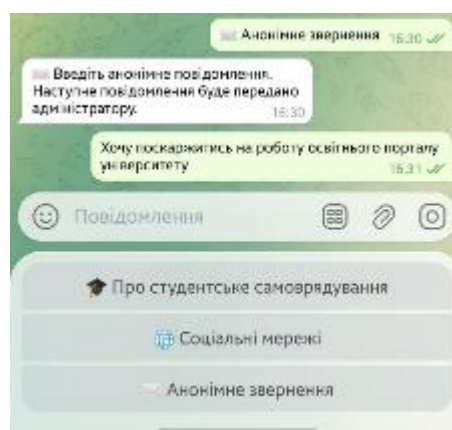


Рисунок 5.7 – Анонімне звернення

## 5.2.2 Тестування системи з боку адміністратора

При вході до адмін-панелі одразу з'являється сторінка авторизації, де необхідно ввести пароль для входу (рис. 5.8).



Рисунок 5.8 – Сторінка авторизації

При введенні невірного паролю видається відповідна помилка (рис. 5.9).



Рисунок 5.9 – Помилка входу

Після введення вірного паролю, користувач бачить сторінку де будуть повідомлення, але якщо їх немає виведено повідомлення про відсутність повідомлень (рис. 5.10).

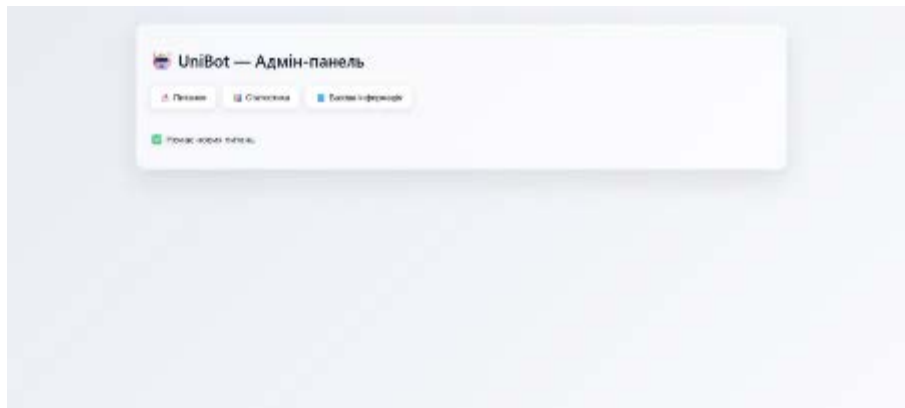


Рисунок 5.10 – Сторінка з питаннями без питань

Коли користувач надсилає повідомлення, яке не було виявлено в базі знань, як це сталося при тестуванні (рис. 5.6), на сторінці адмін-панелі з'являється відповідне повідомлення (рис. 5.11).

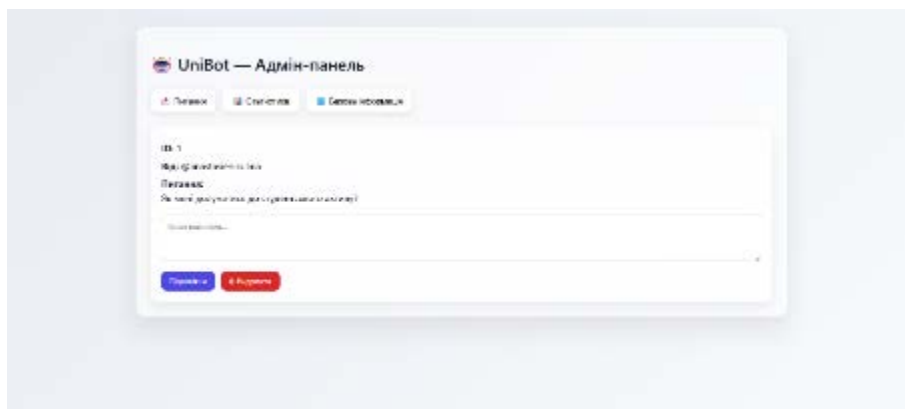


Рисунок 5.11 – Повідомлення в адмін-панелі

Після відправки повідомлення, поле для вводу повідомлення зникає та показується сторінка без повідомлень (рис. 5.10).

В свою чергу користувач отримує зі свого боку відповідь від адміністратора (рис. 5.12).



Рисунок 5.12 – Користувач отримав відповідь

В продовження тестування сторінки, після натискання на кнопку «Статистика», на сторінці з'являється інформаційне повідомлення про статистику використання бота (рис. 5.13).



Рисунок 5.13 – Статистика використання бота

Після натискання кнопки «Базова інформація» відкривається відповідна сторінка (рис. 5.14), з полем де вже виведено інформацію, яку користувач отримує під час натискання кнопок в боті.

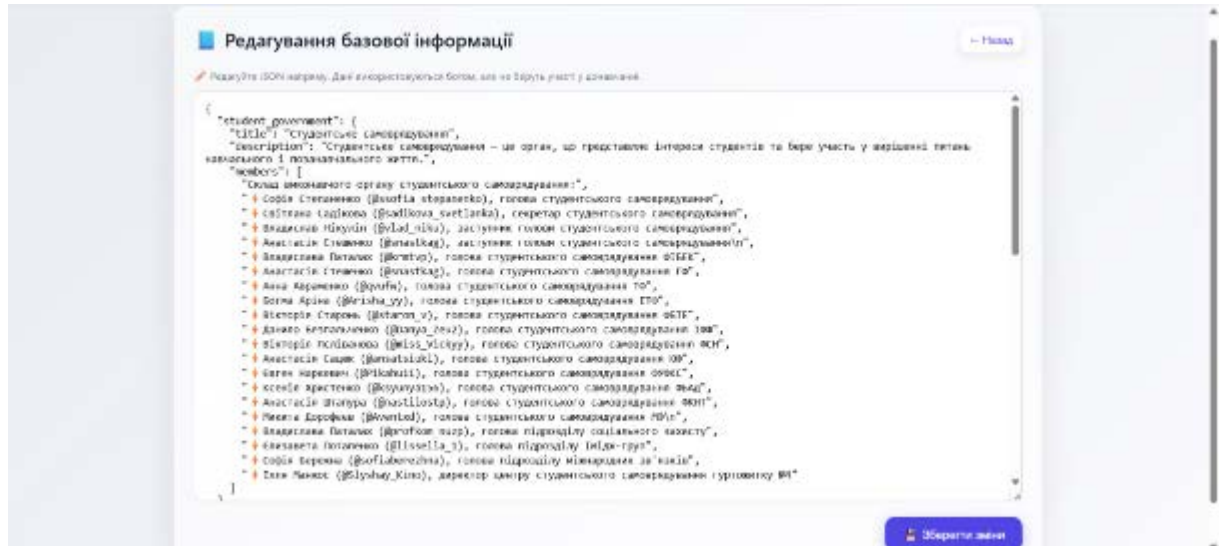


Рисунок 5.14 – Сторінка редагування базової інформації

Після невеликого редагування тексту в боті оновлюється інформація, наприклад у файлі приберемо рядок «Склад виконуючого органу студентського самоврядування:», після чого з боку користувача побачимо, як змінилась відповідь бота. Повідомлення до змін наведено на рисунку 5.3, зокрема відповідь бота після оновлення базової інформації наведено на рисунку 5.15.

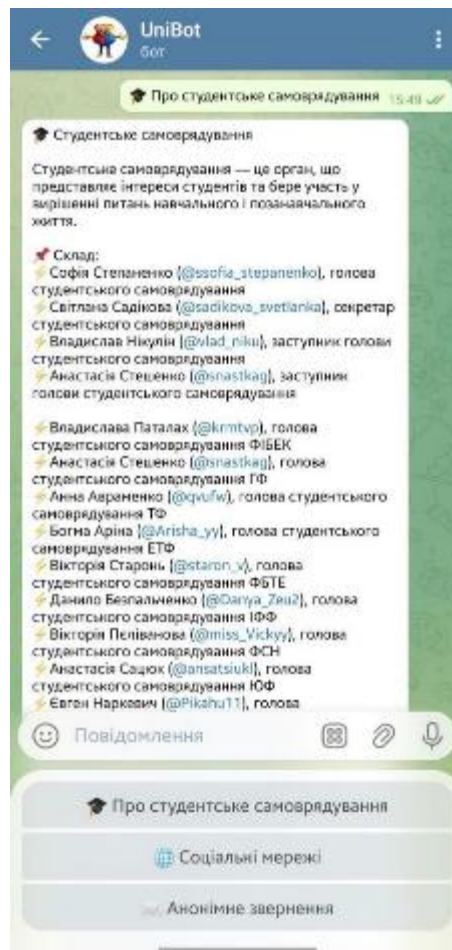


Рисунок 5.15 – Оновлене повідомлення з базовою інформацією

У разі коли користувач надсилає анонімне звернення в адмін-панелі не видається ім'я користувача з вказанням що повідомлення є анонімним (рис. 5.16).

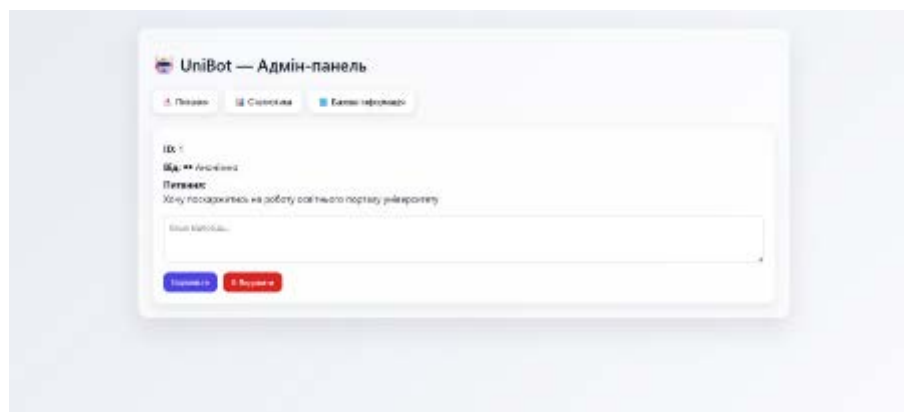


Рисунок 5.16 – Анонімне звернення

Після відправки повідомлення, поле для вводу повідомлення зникає та показується сторінка без повідомлень (рис. 5.10).

В той час користувач отримує зі свого боку відповідь від адміністратора (рис. 5.17).

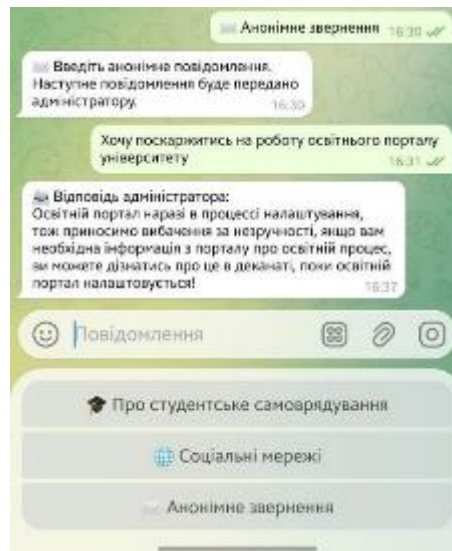


Рисунок 5.17 – Відповідь на анонімне звернення

### 5.3 Висновки до розділу 5

В результаті тестування програмного продукту було виявлено що система працює коректно, без фантазій. Якщо бот не знайшов відповідь на питання в своїй базі знань, він про це повідомляє та передає запит адміністратору, який в свою чергу відповідає на запитання. А бот запише цю відповідь, аби потім мати можливість відповідати на запитання.

Зокрема побачили специфіку роботи анонімних звернень, і побачили що інформація про користувача не була розкрита і адміністратор побачив лише питання.

## ВИСНОВКИ

У ході виконання магістерської дипломної роботи було вирішено актуальне науково-практичне завдання, яке полягає у підвищенні ефективності інформаційної підтримки користувачів в освітньому середовищі шляхом розробки та впровадження автоматизованої системи на базі технологій машинного навчання та семантичного аналізу.

У роботі проведено дослідження процесів комунікації між органами студентського самоврядування та здобувачами освіти. Виявлено, що існуючі підходи (соціальні мережі, прямі звернення) мають суттєві недоліки: розпорошеність інформації, відсутність централізованого обліку звернень, високе навантаження на адміністраторів через однотипні запитання та неможливість забезпечення повної анонімності для чутливих питань. Порівняльний аналіз існуючих аналогів, таких як автоматизація через Manychat або служби підтримки банків, показав, що розробка власного програмного продукту є доцільною для забезпечення специфічних вимог, зокрема анонімності та контрольованого машинного навчання.

Для реалізації системи було обрано мову програмування Python завдяки її потужним можливостям у сфері Data Science та наявності спеціалізованих бібліотек. В якості ядра для взаємодії з користувачами використано бібліотеку python-telegram-bot, що забезпечило доступність сервісу на найпопулярнішій серед студентів платформі. Серверну частину та API реалізовано на базі фреймворку FastAPI, який гарантує високу продуктивність завдяки асинхронній моделі роботи. Збереження даних організовано у форматі JSON/JSONL, що дозволило створити легку та переносиму систему без необхідності розгортання складних СУБД на етапі прототипування.

Ключовим елементом системи став модуль семантичного пошуку. На відміну від генеративних моделей, які можуть надавати неконтрольовані відповіді, було обрано retrieval-based підхід на основі бібліотеки

SentenceTransformers. Використання моделі paraphrase-multilingual-MiniLM-L12-v2 дозволило перетворювати запити користувачів у векторні подання та знаходити найбільш релевантні відповіді у базі знань з високою точністю, враховуючи контекст і синоніми, а не лише ключові слова. У роботі спроектовано та програмно реалізовано унікальні алгоритми, що забезпечують автономність та здатність системи до самонавчання, а саме алгоритм донавчання та анонімності.

Результатом роботи стала повноцінна система «UniBot», що складається з двох взаємопов'язаних компонентів:

- Telegram-бот для студентів, який надає довідкову інформацію через інтерактивне меню, відповідає на питання природною мовою та приймає анонімні звернення;
- вебпанель адміністратора, яка дозволяє зручно обробляти складні запити, переглядати статистику використання (кількість звернень, ефективність бота) та редагувати базову інформацію в режимі реального часу.

Експлуатація та тестування системи підтвердили її стабільність та відповідність поставленим вимогам. Система коректно обробляє як стандартні запити, так і ті, що потребують втручання адміністратора, забезпечуючи при цьому надійний захист даних.

Практична цінність роботи полягає у створенні універсального, масштабованого та недорогого інструменту, який може бути впроваджений не лише в органах студентського самоврядування, а й у будь-яких освітніх установах чи службах підтримки. ПЗ дозволяє суттєво зменшити часові витрати адміністративного персоналу на обробку рутинних запитів та забезпечити цілодобову підтримку користувачів.

Таким чином, мета дипломної роботи досягнута в повному обсязі. Розроблена система демонструє поєднання наукової новизни у підході до поступового саморозширення бази знань та високого рівня практичної придатності для реальних умов експлуатації

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Instagram Automation for Business. ManyChat [Електрон. ресурс]. – Режим доступу: <https://manychat.com/product/instagram>
2. Про Monobank. Monobank [Електрон. ресурс]. – Режим доступу: <https://monobank.ua/about>
3. Навчальний посібник з Python 3.13. Python Documentation [Електрон. ресурс]. – Режим доступу: <https://docs.python.org/uk/3.13/tutorial/index.html>
4. A tour of the C# language. Microsoft Learn [Електрон. ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>
5. What is Java technology and why do I need it? Java [Електрон. ресурс]. – Режим доступу: [https://www.java.com/en/download/help/whatis\\_java.html](https://www.java.com/en/download/help/whatis_java.html)
6. Python Telegram Bot Documentation. Python Telegram Bot [Електрон. ресурс]. – Режим доступу: <https://docs.python-telegram-bot.org/en/stable/>
7. Sbert Platform. Sbert [Електрон. ресурс]. – Режим доступу: <https://sbert.net/>
8. FastAPI. Wikipedia [Електрон. ресурс]. – Режим доступу: <https://en.wikipedia.org/wiki/FastAPI>
9. FastAPI Documentation. FastAPI [Електрон. ресурс]. – Режим доступу: <https://fastapi.tiangolo.com/>
10. Що таке JSON і для чого він потрібен. APIX-Drive Blog [Електрон. ресурс]. – Режим доступу: <https://apix-drive.com/ua/blog/useful/scho-take-json>
11. Paraphrase Multilingual MiniLM Model. Hugging Face [Електрон. ресурс]. – Режим доступу: <https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2>
12. Що таке Figma та для кого вона потрібна. CASES Media [Електрон.

ресурс]. – Режим доступу: <https://cases.media/article/sho-take-figma-ta-dlya-kogo-vona-potribna>

13. Що таке прототипування. Dizz.in.ua [Електрон. ресурс]. – Режим доступу: <https://dizz.in.ua/uk/chto-takoe-prototipirovanie/>

14. ChatGPT. OpenAI [Електрон. ресурс]. – Режим доступу: <https://chatgpt.com/>

15. Fine-tuning Semantic Search Model with Sentence Transformers. NLP Cloud [Електрон. ресурс]. – Режим доступу: <https://nlpcloud.com/uk/fine-tuning-semantic-search-model-with-sentence-transformers-for-rag-application.html>

16. Дослідницьке тестування: глибоке занурення. ZAPTEST [Електрон. ресурс]. – Режим доступу: <https://www.zaptest.com/uk/дослідницьке-тестування-глибоке-зан>

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

В умовах стрімкої цифровізації освітнього середовища та зростання обсягів інформаційних потоків, ефективна комунікація між органами студентського самоврядування та здобувачами освіти набуває критичного значення. Традиційні методи взаємодії, такі як особисті звернення або листування через соціальні мережі, демонструють недостатню ефективність через розпорошеність даних, відсутність системності та значне часове навантаження на адміністраторів, які змушені обробляти велику кількість однотипних запитів. Це зумовлює актуальність розробки автоматизованих систем підтримки, здатних забезпечити оперативне надання інформації та оптимізувати роботу адміністративного персоналу.

Призначенням розробки є створення єдиного централізованого каналу комунікації, який забезпечує цілодобову підтримку користувачів та можливість анонімних звернень для вирішення чутливих питань. Система передбачає гібридний режим роботи: автоматичні відповіді на типові запитання та маршрутизацію складних запитів до оператора з подальшим донавчанням бази знань на основі реальних діалогів. Це дозволить суттєво підвищити якість інформаційного обслуговування в університеті та зменшити навантаження на службу підтримки.

### **A.1 Підстави для розробки**

Підставою для виконання розробки автоматизованої системи підтримки користувачів є завдання на дипломний проєкт (роботу) магістра.

Необхідність створення системи зумовлена потребою в оптимізації комунікаційних процесів в освітньому середовищі, зокрема між органами студентського самоврядування та здобувачами освіти. Існуючі методи взаємодії (соціальні мережі, особисті звернення) характеризуються низькою ефективністю через розпорошеність інформації, відсутність системного

підходу до обробки запитів та значне часове навантаження на адміністраторів, викликане великою кількістю однотипних звернень.

## **A.2 Вимоги до програмного продукту**

### **A.2.1 Вимоги до структури та функціонування**

Система повинна складатися з двох основних модулів:

#### **1) Telegram-бот (клієнтська частина):**

- наявність інтерактивного меню з кнопками: «про студентське самоврядування», «соціальні мережі», «анонімне звернення» тощо.
- підтримка двох режимів роботи: автоматичний (відповідь з бази знань) та режим переадресації адміністратору.
- реалізація семантичного пошуку відповідей на основі векторних подань тексту.

#### **2) Вебсервіс (адміністративна частина):**

- авторизація доступу для адміністраторів.
- інтерфейс для відповіді на переадресовані запити.
- модуль редагування базової інформації та контенту кнопок.
- модуль перегляду статистики (кількість повідомлень, ефективність бота, анонімні звернення).

## **A.3 Вимоги до програмної документації**

До складу програмної документації повинні входити:

- пояснювальна записка до дипломного проєкту;
- керівництво оператора з інструкціями щодо обробки запитів;
- текст програми (вихідний код).

**ДОДАТОК Б**  
**Текст програми**

## Б.1 Код файла «bot.py»:

```

import logging

logging.getLogger("apscheduler").setLevel(logging.WARNING)

import json
import asyncio
from telegram import Update
from telegram.ext import (
    ApplicationBuilder,
    CommandHandler,
    MessageHandler,
    ContextTypes,
    filters,
)

from scripts.retrieval_engine import RetrievalEngine
from scripts.stats import Stats
from scripts.question_storage import (
    add_question,
    get_unanswered,
    answer_question,
    delete_question,
    get_answered_not_notified,
    mark_notified,
)

from telegram import ReplyKeyboardMarkup, InlineKeyboardMarkup,
InlineKeyboardButton
from scripts.info_storage import (
    get_student_government,
    get_socials_student_government,
    get_socials_faculties
)

from telegram import ReplyKeyboardMarkup, KeyboardButton

from telegram.ext import CallbackQueryHandler

# =====
# НАСТРОЙКИ
# =====

#TOKEN = "7936830698:AAFZCPXujyeXqcZEhaoR9YeFoUi8W00t0-M" # wally
TOKEN = "8158072096:AAG6RkEPX4oT32hZpNzfZ0QWsAcM67KLwUk"
ADMIN_ID = 838269662
DATA_PATH = "data/unibot_train.jsonl"

# =====

```

```

# ІНІЦІАЛІЗАЦІЯ
# =====

logging.basicConfig(level=logging.INFO)

engine = RetrievalEngine(DATA_PATH)
stats = Stats()

pending_anon = set()
pending_admin = set()

# =====
# ДОДАВАННЯ В DATASET
# =====

def append_to_dataset(question: str, answer: str):
    record = {
        "instruction": question.replace("\n", " ").strip(),
        "output": answer.replace("\n", " ").strip(),
    }

    with open(DATA_PATH, "a", encoding="utf-8") as f:
        f.write(json.dumps(record, ensure_ascii=False) + "\n")

# =====
# МЕНЮ
# =====

def main_menu():
    keyboard = [
        ["☐ Про студентське самоврядування"],
        ["☐ Соціальні мережі"],
        ["☒ Анонімне звернення"]
    ]

    return ReplyKeyboardMarkup(
        keyboard,
        resize_keyboard=True
    )

# =====
# КОМАНДИ СТУДЕНТА
# =====

    async def start(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        await update.message.reply_text(

```

```

        "Тебе вітає UniBot! Я допоможу тобі не загубитись в
        Політехніці! □\n\n"
        "У тебе з'явилося меню, де ти можеш знайти цікаву та
        корисну інформацію □□\n\n"
        "Я працюю як невелика нейромережа, адже маю певну базу
        знань щодо нашого університету! □"
        "Також ти можеш просто написати мені, що тебе турбує,
        можливо є якісь питання,"
        "а я із задоволенням відповім та допоможу з чим зможу ☺"
        "Таке звернення не буде анонімним, але якщо ти хочеш
        анонімно залишити повідомлення, воно обов'язково потрапить
        адміністратору, без розкриття твоєї особистості□\n\n"
        "Якщо я не буду впевнений у відповіді, передам твій запит
        адміністратору, який допоможе!",
        reply_markup=main_menu()
    )

    async def commands(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        await update.message.reply_text(
            "/start – почати роботу\n"
            "/commands – список команд\n"
            "/anonim – анонімне звернення\n"
            "/admin – звернення напряму до адміністратора"
        )

    async def anonim(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        pending_anon.add(update.effective_user.id)
        await update.message.reply_text(
            "☒ Введіть анонімне повідомлення.\n"
            "Наступне повідомлення буде передано адміністратору."
        )

    async def admin_direct(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        pending_admin.add(update.effective_user.id)
        await update.message.reply_text(
            "☒ Введіть повідомлення для адміністратора.\n"
            "Наступне повідомлення буде передано напряму."
        )

    # =====
    # АДМІН-КОМАНДИ
    # =====

    async def admin_messages(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        if update.effective_user.id != ADMIN_ID:
            return

```

```

unanswered = get_unanswered()

if not unanswered:
    await update.message.reply_text("□ Немає нових повідомлень")
    return

text = "□ Неопрацьовані питання:\n\n"
for q in unanswered:
    text += f"#{q['id']} – {q['question']}\n"

await update.message.reply_text(text)

async def admin_answer(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if update.effective_user.id != ADMIN_ID:
        return

    if len(context.args) < 2:
        await update.message.reply_text("Формат: \n/answer #ID текст відповіді")
        return

    qid = context.args[0].lstrip("#")
    if not qid.isdigit():
        await update.message.reply_text("□ Невірний ID")
        return

    answer_text = " ".join(context.args[1:])
    q = answer_question(int(qid), answer_text)

    if not q:
        await update.message.reply_text("□ Питання не знайдено")
        return

    await context.bot.send_message(
        chat_id=q["user_id"],
        text=f"□ Відповідь адміністратора: \n{answer_text}"
    )

    if not q["anonymous"]:
        append_to_dataset(q["question"], answer_text)
        engine.reload()
        stats.inc("dataset_added")

    stats.inc("admin_answers")
    await update.message.reply_text("□ Відповідь надіслано")

```

```

        async def admin_delete(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        if update.effective_user.id != ADMIN_ID:
            return

        if len(context.args) != 1:
            await update.message.reply_text("Формат:\n/delete #ID")
            return

        qid = context.args[0].lstrip("#")
        if not qid.isdigit():
            await update.message.reply_text("❑ Невірний ID")
            return

        if delete_question(int(qid)):
            await update.message.reply_text(f"❑ Питання #{qid}
видалено")
        else:
            await update.message.reply_text("❑ Питання не знайдено")

        # =====
        # ОБРОБКА ПОВІДОМЛЕНЬ
        # =====

        async def handle_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
            user = update.effective_user
            text = update.message.text.strip()

            stats.inc("total_messages")

            if text == "❑ Про студентське самоврядування":
                info = get_student_government()

                if not info:
                    await update.message.reply_text("Інформація
тимчасово недоступна.")
                    return

                msg = f"❑ {info.get('title', '')}\n\n"
                msg += info.get("description", "") + "\n\n"
                msg += "❑ Склад:\n"

                for m in info.get("members", []):
                    msg += f"{m}\n"

            await update.message.reply_text(msg,
reply_markup=main_menu())
            return

```

```

        if text == "☐ Соціальні мережі":
            keyboard = InlineKeyboardMarkup([
                [InlineKeyboardButton("Студ. самоврядування",
callback_data="socials_gov")],
                [InlineKeyboardButton("Факультети",
callback_data="socials_fac")]]
            ])

            await update.message.reply_text(
                "У нас є соціальні мережі як для студентського
самоврядування загалом так і окремо у кожного з факультетів ☺",
                reply_markup=keyboard
            )
            return

        if text == "✉ Анонімне звернення":
            pending_anon.add(update.effective_user.id)
            await update.message.reply_text(
                "✉ Введіть анонімне повідомлення.\n"
                "Наступне повідомлення буде передано адміністратору."
            )
            return

        if user.id in pending_anon:
            pending_anon.remove(user.id)
            stats.inc("anonymous_messages")
            stats.inc("to_admin")

            q = add_question(text, user.id, user.username, True)

            await context.bot.send_message(
                chat_id=ADMIN_ID,
                text=f"☐ Нове анонімне питання\nID:
#{q['id']}\n\n{text}"
            )

            await update.message.reply_text("☐ Анонімне повідомлення
надіслано")
            return

        if user.id in pending_admin:
            pending_admin.remove(user.id)
            stats.inc("to_admin")

            q = add_question(text, user.id, user.username, False)
            sender = f"@{user.username}" if user.username else
f"ID:{user.id}"

            await context.bot.send_message(

```

```

        chat_id=ADMIN_ID,
        text=f"❏ Повідомлення студента\nID:
#{q['id']}\n{sender}\n\n{text}"
    )

    await update.message.reply_text("❏ Повідомлення передано
адміністратору")
    return

    answer, score = engine.ask(text)
    if answer is not None:
        stats.inc("bot_answers")
        await update.message.reply_text(answer)
        return

    stats.inc("to_admin")

    q = add_question(text, user.id, user.username, False)
    sender = f"@{user.username}" if user.username else
f"ID:{user.id}"

    await update.message.reply_text(
        "Я не впевнений у відповіді.\nВаше питання передано
адміністратору."
    )

    await context.bot.send_message(
        chat_id=ADMIN_ID,
        text=f"❏ Нове питання\nID:
#{q['id']}\n{sender}\n\n{text}"
    )

    async def handle_callback(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        query = update.callback_query
        await query.answer()

        if query.data == "socials_gov":
            socials = get_socials_student_government()
            msg = "❏ Соціальні мережі студентського
самоврядування:\n\n"
            for name, link in socials.items():
                msg += f"• {name}: {link}\n"

            await query.message.reply_text(msg,
reply_markup=main_menu())

        if query.data == "socials_fac":
            socials = get_socials_faculties()

```

```

msg = "☐ Соціальні мережі факультетів:\n\n"
for m in socials.get("All Media", []):
    msg += f"{m}\n"

    await query.message.reply_text(msg,
reply_markup=main_menu())

# =====
# ФОНОВА ПЕРЕВІРКА ВІДПОВІДЕЙ
# =====

async def watch_admin_answers(context:
ContextTypes.DEFAULT_TYPE):
    app = context.application

    answered = get_answered_not_notified()
    if not answered:
        return

    for q in answered:
        try:

            await app.bot.send_message(
                chat_id=q["user_id"],
                text=f"☐                                     Відповідь
адміністратора:\n{q['answer']}"
            )

            if not q["anonymous"]:
                append_to_dataset(q["question"], q["answer"])
                engine.reload()
                stats.inc("dataset_added")

            await app.bot.send_message(
                chat_id=ADMIN_ID,
                text=f"☐     Відповідь     на     питання     #{q['id']}
надіслано студенту"
            )

            mark_notified(q["id"])
            stats.inc("admin_answers")

        except Exception as e:
            logging.error(f"Помилка відповіді #{q['id']}: {e}")

#####
#####

def main():

```

```

app = ApplicationBuilder().token(TOKEN).build()

app.add_handler(CommandHandler("start", start))
app.add_handler(CommandHandler("commands", commands))
app.add_handler(CommandHandler("anonim", anonim))
app.add_handler(CommandHandler("admin", admin_direct))
app.add_handler(CommandHandler("message", admin_messages))
app.add_handler(CommandHandler("answer", admin_answer))
app.add_handler(CommandHandler("delete", admin_delete))
app.add_handler(MessageHandler(filters.TEXT &
~filters.COMMAND, handle_message))
app.add_handler(CallbackQueryHandler(handle_callback))

app.job_queue.run_repeating(
    watch_admin_answers,
    interval=2,
    first=2
)

print("□ UniBot запущено")
app.run_polling()

if __name__ == "__main__":
    main()

```

## Б.2 Код файла «retrieval\_engine.py»:

```

import json
import numpy as np
from pathlib import Path
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity

class RetrievalEngine:
    def __init__(self, data_path: str, threshold: float = 0.6):
        self.data_path = data_path
        self.threshold = threshold

        self.model = SentenceTransformer(
            "paraphrase-multilingual-MiniLM-L12-v2"
        )

        self.questions = []
        self.answers = []

        self._load_data()

    def _load_data(self):

```

```

self.questions.clear()
self.answers.clear()

with Path(self.data_path).open(encoding="utf-8") as f:
    for line in f:
        obj = json.loads(line)
        self.questions.append(obj["instruction"])
        self.answers.append(obj["output"])

self.embeddings = self.model.encode(
    self.questions,
    normalize_embeddings=True,
    show_progress_bar=True
)

def ask(self, query: str):
    q_vec = self.model.encode(
        [query],
        normalize_embeddings=True
    )

    sims = cosine_similarity(q_vec, self.embeddings)[0]
    idx = int(np.argmax(sims))
    score = float(sims[idx])

    if score < self.threshold:
        return None, score

    return self.answers[idx], score

def reload(self):
    self._load_data()

```

### Б.3 Код файла «question\_storage.py»:

```

import json
import time
from pathlib import Path
from threading import Lock
from typing import List, Optional, Dict

QUESTIONS_PATH = Path("data/questions.json")
_lock = Lock()

def _load() -> List[Dict]:
    if not QUESTIONS_PATH.exists():
        QUESTIONS_PATH.parent.mkdir(parents=True, exist_ok=True)
        QUESTIONS_PATH.write_text("[]", encoding="utf-8")

    with QUESTIONS_PATH.open(encoding="utf-8") as f:

```

```

        return json.load(f)

def _save(data: List[Dict]) -> None:
    with QUESTIONS_PATH.open("w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2)

def _next_id(data: List[Dict]) -> int:
    if not data:
        return 1
    return max(q["id"] for q in data) + 1

def add_question(
    question: str,
    user_id: int,
    username: Optional[str],
    anonymous: bool
) -> Dict:
    with _lock:
        data = _load()

        q = {
            "id": _next_id(data),
            "question": question.strip(),
            "user_id": user_id,
            "username": None if anonymous else username,
            "anonymous": anonymous,
            "answered": False,
            "answer": None,
            "notified": False,
            "created_at": int(time.time()),
        }

        data.append(q)
        _save(data)

    return q

def get_unanswered() -> List[Dict]:
    with _lock:
        return [
            q for q in _load()
            if not q.get("answered", False)
        ]

def answer_question(qid: int, answer: str) -> Optional[Dict]:
    with _lock:

```

```

    data = _load()

    for q in data:
        if q["id"] == qid:
            q["answered"] = True
            q["answer"] = answer.strip()
            _save(data)
            return q

    return None

def delete_question(qid: int) -> bool:
    with _lock:
        data = _load()
        new_data = [q for q in data if q["id"] != qid]

        if len(new_data) == len(data):
            return False

        _save(new_data)
        return True

def get_answered_not_notified() -> List[Dict]:
    with _lock:
        return [
            q for q in _load()
            if q.get("answered") and not q.get("notified")
        ]

def mark_notified(qid: int) -> None:
    with _lock:
        data = _load()

        for q in data:
            if q["id"] == qid:
                q["notified"] = True
                break

        _save(data)

def cleanup_old(hours: int = 8) -> int:
    cutoff = time.time() - hours * 3600

    with _lock:
        data = _load()
        new_data = [

```

```

        q for q in data
        if q.get("created_at", 0) >= cutoff
    ]

    removed = len(data) - len(new_data)
    if removed > 0:
        _save(new_data)

    return removed

```

#### Б.4 Код файла «info\_storage.py»:

```

import json
from pathlib import Path
from threading import Lock

INFO_PATH = Path("data/info.json")
_lock = Lock()

def _load():
    if not INFO_PATH.exists():
        return {}

    with INFO_PATH.open(encoding="utf-8") as f:
        return json.load(f)

def get_student_government():
    data = _load()
    return data.get("student_government", {})

def get_socials_student_government():
    data = _load()
    return data.get("socials", {}).get("student_government", {})

def get_socials_faculties():
    data = _load()
    return data.get("socials", {}).get("faculties", {})

def save_info(raw_json: str) -> bool:
    try:
        data = json.loads(raw_json)

        with _lock:
            INFO_PATH.parent.mkdir(parents=True, exist_ok=True)
            with INFO_PATH.open("w", encoding="utf-8") as f:

```

```

        json.dump(data, f, indent=2, ensure_ascii=False)

    return True
except Exception as e:
    print("SAVE INFO ERROR:", e)
    return False

```

### Б.5 Код файла «admin\_panel.py»:

```

from fastapi import FastAPI, Request
from fastapi.responses import HTMLResponse, JSONResponse
from fastapi.templating import Jinja2Templates
from pathlib import Path
import json
from threading import Lock
from fastapi import Request, Form
from fastapi.responses import RedirectResponse
from starlette.middleware.sessions import SessionMiddleware
from scripts.stats import Stats
import logging
from scripts.info_storage import save_info, _load

logging.getLogger("apscheduler").setLevel(logging.WARNING)

# =====
# НАЛАШТИВАННЯ
# =====
ADMIN_PASSWORD = "admin123"
QUESTIONS_PATH = Path("data/questions.json")

app = FastAPI()
app.add_middleware(
    SessionMiddleware,
    secret_key="UNIBOT_SUPER_SECRET_KEY"
)
templates = Jinja2Templates(directory="templates")

stats = Stats()
lock = Lock()

def load_questions():
    if not QUESTIONS_PATH.exists():
        QUESTIONS_PATH.parent.mkdir(parents=True, exist_ok=True)
        QUESTIONS_PATH.write_text("[]", encoding="utf-8")

    with QUESTIONS_PATH.open(encoding="utf-8") as f:
        return json.load(f)

```

```

def save_questions(data):
    with QUESTIONS_PATH.open("w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2)

#####          #####          #####

@app.get("/admin")
def admin_page(request: Request):
    if not request.session.get("admin"):
        return templates.TemplateResponse(
            "login.html",
            {"request": request}
        )

    return templates.TemplateResponse(
        "admin.html",
        {"request": request}
    )

@app.get("/admin/info")
def admin_info_page(request: Request):
    if not request.session.get("admin"):
        return RedirectResponse("/admin", status_code=302)
    info = _load()
    info_text = json.dumps(
        info,
        indent=2,
        ensure_ascii=False
    )
    return templates.TemplateResponse(
        "info_edit.html",
        {
            "request": request,
            "info_text": info_text
        }
    )

@app.post("/admin/info/save")
async def save_info_api(request: Request):
    if not request.session.get("admin"):
        return RedirectResponse("/admin", status_code=302)

    form = await request.form()
    raw_json = form.get("info")

    if not raw_json:
        return JSONResponse({"error": "empty"}, status_code=400)

```

```

        if save_info(raw_json):
            return RedirectResponse("/admin/info", status_code=302)

        return JSONResponse({"error": "invalid json"},
status_code=400)

```

```

@app.post("/login")
def login(request: Request, password: str = Form(...)):
    if password == ADMIN_PASSWORD:
        request.session["admin"] = True
        return RedirectResponse("/admin", status_code=302)

    return templates.TemplateResponse(
        "login.html",
        {
            "request": request,
            "error": "Невірний пароль"
        }
    )

```

```

@app.get("/api/questions")
def get_questions():
    with lock:
        questions = load_questions()

    # показываем только НЕ отвеченные
    return [
        q for q in questions if not q.get("answered", False)
    ]

```

```

@app.post("/api/answer")
async def answer_question(payload: dict):
    qid = payload.get("id")
    answer = payload.get("answer")

    if not qid or not answer:
        return JSONResponse(
            {"error": "Invalid data"},
            status_code=400
        )

    with lock:
        questions = load_questions()

        for q in questions:
            if q["id"] == qid:

```

```

        q["answered"] = True
        q["answer"] = answer
        q["notified"] = False    # ☐ БАЖХО
        save_questions(questions)

        stats.inc("admin_answers")
        return {"status": "ok"}

    return JsonResponse(
        {"error": "Question not found"},
        status_code=404
    )

@app.post("/api/delete")
async def delete_question(payload: dict):
    qid = payload.get("id")

    if not qid:
        return JsonResponse(
            {"error": "Invalid ID"},
            status_code=400
        )

    with lock:
        questions = load_questions()
        questions = [q for q in questions if q["id"] != qid]
        save_questions(questions)

    return {"status": "deleted"}

@app.get("/api/stats")
def api_stats():
    return stats.get_all()

```

## Б.6 Код файла «stats.py»:

```

import json
from pathlib import Path
from threading import Lock

class Stats:
    def __init__(self, path: str = "data/stats.json"):
        self.path = Path(path)
        self.lock = Lock()

        self.data = {
            "total_messages": 0,

```

```

        "bot_answers": 0,
        "admin_answers": 0,
        "to_admin": 0,
        "anonymous_messages": 0,
        "dataset_added": 0,
    }

    self._load()

def _load(self):
    if self.path.exists():
        try:
            with self.path.open(encoding="utf-8") as f:
                saved = json.load(f)
                self.data.update(saved)
        except Exception:
            # если файл битый — стартуем с нуля
            pass
    else:
        self._save()

def _save(self):
    self.path.parent.mkdir(parents=True, exist_ok=True)
    with self.path.open("w", encoding="utf-8") as f:
        json.dump(self.data, f, ensure_ascii=False, indent=2)

def inc(self, key: str, value: int = 1):
    with self.lock:
        if key not in self.data:
            self.data[key] = 0
        self.data[key] += value
        self._save()

def get_all(self):
    return dict(self.data)

```

## Б.7 Код файла «admin.html»:

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>UniBot Admin</title>

    <style>
        :root {
            --bg: #f2f4f8;
            --glass: rgba(255, 255, 255, 0.65);
            --border: rgba(255, 255, 255, 0.4);
            --accent: #4f46e5;

```

```

        --danger: #dc2626;
        --radius: 16px;
    }

    * {
        box-sizing: border-box;
    }

    body {
        margin: 0;
        min-height: 100vh;
        font-family: "Segoe UI", system-ui, sans-serif;
        background: linear-gradient(135deg, #e9ecf3,
#f8fafc);
        color: #111827;
        padding: 32px;
    }

    h1 {
        margin-bottom: 24px;
        font-size: 28px;
        font-weight: 600;
    }

    /* ===== Glass container ===== */
    .panel {
        max-width: 1100px;
        margin: 0 auto;
        background: var(--glass);
        backdrop-filter: blur(14px);
        border-radius: var(--radius);
        padding: 24px;
        border: 1px solid var(--border);
        box-shadow: 0 20px 40px rgba(0,0,0,0.08);
    }

    /* ===== Navigation ===== */
    .nav {
        display: flex;
        gap: 12px;
        margin-bottom: 24px;
        flex-wrap: wrap;
    }

    .nav button {
        padding: 10px 16px;
        border-radius: 10px;
        border: none;
        background: white;
        cursor: pointer;
    }

```

```

        font-size: 14px;
        font-weight: 500;
        box-shadow: 0 4px 10px rgba(0,0,0,0.08);
        transition: all 0.2s ease;
    }

    .nav button:hover {
        transform: translateY(-1px);
        background: var(--accent);
        color: white;
    }

    /* ===== Content ===== */
    #content {
        display: flex;
        flex-direction: column;
        gap: 16px;
    }

    /* ===== Question card ===== */
    .question {
        background: rgba(255,255,255,0.85);
        border-radius: 14px;
        padding: 18px;
        box-shadow: 0 10px 20px rgba(0,0,0,0.06);
    }

    .question p {
        margin: 6px 0;
        line-height: 1.5;
    }

    textarea {
        width: 100%;
        min-height: 80px;
        border-radius: 10px;
        border: 1px solid #d1d5db;
        padding: 10px;
        font-family: inherit;
        resize: vertical;
        margin-top: 10px;
    }

    textarea:focus {
        outline: none;
        border-color: var(--accent);
    }

    .actions {
        display: flex;

```

```

        gap: 10px;
        margin-top: 12px;
    }

    .actions button {
        padding: 8px 14px;
        border-radius: 10px;
        border: none;
        cursor: pointer;
        font-size: 13px;
        transition: 0.2s;
    }

    .btn-answer {
        background: var(--accent);
        color: white;
    }

    .btn-answer:hover {
        opacity: 0.9;
    }

    .btn-delete {
        background: var(--danger);
        color: white;
    }

    .btn-delete:hover {
        opacity: 0.9;
    }

    /* ===== Stats ===== */
    ul {
        padding-left: 20px;
    }

    li {
        margin-bottom: 6px;
    }

    @media (max-width: 600px) {
        body {
            padding: 16px;
        }
    }
</style>
</head>
<body>

<div class="panel">

```

```

<h1>❑ UniBot – Адмін-панель</h1>

<div class="nav">
  <button onclick="loadQuestions()">❑ Питання</button>
  <button onclick="loadStats()">❑ Статистика</button>
  <button          onclick="openInfo()">❑          Базова
інформація</button>
</div>

<div id="content">
  <p>Завантаження...</p>
</div>
</div>

<script>
async function loadQuestions() {
  const res = await fetch("/api/questions");
  const data = await res.json();

  const container = document.getElementById("content");
  container.innerHTML = "";

  if (data.length === 0) {
    container.innerHTML = "<p>❑ Немає нових питань</p>";
    return;
  }

  data.forEach(q => {
    const div = document.createElement("div");
    div.className = "question";
    div.innerHTML = `
      <p><b>ID:</b> ${q.id}</p>
      <p><b>Питання:</b><br>${q.question}</p>

      <textarea          placeholder="Ваша
відповідь..."></textarea>

      <div class="actions">
        <button          class="btn-answer"
onclick="sendAnswer(${q.id}, this)">Відповісти</button>
        <button          class="btn-delete"
onclick="deleteQuestion(${q.id})">❑ Видалити</button>
      </div>
    `;
    container.appendChild(div);
  });
}

async function sendAnswer(id, btn) {

```

```

const textarea =
btn.closest(".question").querySelector("textarea");
const answer = textarea.value.trim();
if (!answer) return alert("Введіть відповідь");

await fetch("/api/answer", {
  method: "POST",
  headers: {"Content-Type": "application/json"},
  body: JSON.stringify({id: id, answer: answer})
});

loadQuestions();
}

async function deleteQuestion(id) {
  if (!confirm("Видалити питання?")) return;

  await fetch("/api/delete", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({id: id})
  });

  loadQuestions();
}

async function loadStats() {
  const res = await fetch("/api/stats");
  const stats = await res.json();

  const container = document.getElementById("content");
  container.innerHTML = `
    <h2>📊 Статистика</h2>
    <ul>
      <li>Усього повідомлень: ${stats.total_messages}</li>
      <li>Відповідей бота: ${stats.bot_answers}</li>
      <li>Відповідей адміністратора:
${stats.admin_answers}</li>
      <li>Передано адміну: ${stats.to_admin}</li>
      <li>Анонімних: ${stats.anonymous_messages}</li>
      <li>Додано в базу знань: ${stats.dataset_added}</li>
    </ul>
  `;
}

function openInfo() {
  window.location.href = "/admin/info";
}

setInterval(() => {

```

```

        if (document.activeElement.tagName === "TEXTAREA") return;
        loadQuestions();
    }, 5000);

loadQuestions();
</script>

</body>
</html>

```

### Б.8 Код файлу «info\_edit.html»:

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Редагування базової інформації</title>

    <style>
        :root {
            --glass: rgba(255, 255, 255, 0.65);
            --border: rgba(255, 255, 255, 0.4);
            --accent: #4f46e5;
            --danger: #dc2626;
            --radius: 16px;
        }

        * {
            box-sizing: border-box;
        }

        body {
            margin: 0;
            min-height: 100vh;
            font-family: "Segoe UI", system-ui, sans-serif;
            background: linear-gradient(135deg, #e9ecf3,
#f8fafc);

            color: #111827;
            padding: 32px;
        }

        .panel {
            max-width: 1100px;
            margin: 0 auto;
            background: var(--glass);
            backdrop-filter: blur(14px);
            border-radius: var(--radius);
            padding: 24px;
            border: 1px solid var(--border);
            box-shadow: 0 20px 40px rgba(0,0,0,0.08);
        }
    </style>

```

```

}

.top {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

h2 {
  margin: 0;
  font-weight: 600;
}

a.back {
  text-decoration: none;
  font-size: 14px;
  color: var(--accent);
  background: white;
  padding: 8px 12px;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0,0,0,0.08);
  transition: 0.2s;
}

a.back:hover {
  background: var(--accent);
  color: white;
}

textarea {
  width: 100%;
  min-height: 520px;
  font-family: ui-monospace, SFMono-Regular, Menlo,
Consolas, monospace;
  font-size: 14px;
  padding: 14px;
  border-radius: 12px;
  border: 1px solid #d1d5db;
  resize: vertical;
  background: rgba(255,255,255,0.9);
}

textarea:focus {
  outline: none;
  border-color: var(--accent);
}

.actions {
  display: flex;

```

```

        justify-content: flex-end;
        margin-top: 16px;
    }

    button {
        padding: 12px 22px;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        background: var(--accent);
        color: white;
        cursor: pointer;
        box-shadow: 0 6px 16px rgba(79,70,229,0.35);
        transition: 0.2s;
    }

    button:hover {
        opacity: 0.9;
        transform: translateY(-1px);
    }

    .error {
        margin-bottom: 14px;
        padding: 12px;
        border-radius: 10px;
        background: rgba(220,38,38,0.1);
        color: var(--danger);
        font-size: 14px;
    }

    .hint {
        margin-bottom: 10px;
        font-size: 13px;
        color: #374151;
        opacity: 0.8;
    }

    @media (max-width: 600px) {
        body {
            padding: 16px;
        }
    }
</style>
</head>
<body>

<div class="panel">

    <div class="top">
        <h2>□ Редагування базової інформації</h2>

```

```

        <a href="/admin" class="back">← Назад</a>
    </div>

    <div class="hint">
        ⚠ Редагуйте JSON напряду. Дані використовуються ботом, але не
        беруть участі у донавчанні.
    </div>

    {% if error %}
    <div class="error">{{ error }}</div>
    {% endif %}

    <form method="post" action="/admin/info/save">
        <textarea name="info">{{ info_text }}</textarea>

        <div class="actions">
            <button type="submit">□ Зберегти зміни</button>
        </div>
    </form>

</div>
</body>
</html>

```

## Б.9 Код файлу «login.html»:

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>UniBot – Вхід</title>

    <style>
        :root {
            --glass: rgba(255, 255, 255, 0.65);
            --border: rgba(255, 255, 255, 0.4);
            --accent: #4f46e5;
            --danger: #dc2626;
            --radius: 18px;
        }

        * {
            box-sizing: border-box;
        }

        body {
            margin: 0;
            min-height: 100vh;
            font-family: "Segoe UI", system-ui, sans-serif;
        }
    </style>

```

```

background: linear-gradient(135deg, #e9ecf3,
#f8fafc);

display: flex;
align-items: center;
justify-content: center;
color: #111827;
}

.card {
width: 100%;
max-width: 380px;
background: var(--glass);
backdrop-filter: blur(16px);
border-radius: var(--radius);
padding: 32px;
border: 1px solid var(--border);
box-shadow: 0 25px 50px rgba(0,0,0,0.12);
}

h1 {
margin: 0 0 8px;
text-align: center;
font-size: 22px;
font-weight: 600;
}

.subtitle {
text-align: center;
font-size: 14px;
color: #374151;
margin-bottom: 24px;
opacity: 0.8;
}

label {
display: block;
font-size: 14px;
margin-bottom: 6px;
}

input[type="password"] {
width: 100%;
padding: 12px 14px;
border-radius: 12px;
border: 1px solid #d1d5db;
font-size: 15px;
background: rgba(255,255,255,0.9);
}

input:focus {

```

```

        outline: none;
        border-color: var(--accent);
    }

    button {
        width: 100%;
        margin-top: 18px;
        padding: 12px;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        background: var(--accent);
        color: white;
        cursor: pointer;
        box-shadow: 0 8px 18px rgba(79,70,229,0.35);
        transition: 0.2s;
    }

    button:hover {
        opacity: 0.9;
        transform: translateY(-1px);
    }

    .error {
        margin-top: 14px;
        padding: 10px;
        font-size: 14px;
        border-radius: 10px;
        background: rgba(220,38,38,0.1);
        color: var(--danger);
        text-align: center;
    }

    .footer {
        margin-top: 22px;
        text-align: center;
        font-size: 12px;
        color: #6b7280;
    }
</style>
</head>
<body>

<div class="card">
    <h1>❏ UniBot</h1>
    <div class="subtitle">
        Вхід до адміністративної панелі
    </div>

    <form method="post" action="/login">

```

```

<label for="password">Пароль адміністратора</label>
<input
    type="password"
    name="password"
    id="password"
    placeholder="....."
    required
>

<button type="submit">□ Вийти</button>
</form>

{% if error %}
<div class="error">{{ error }}</div>
{% endif %}

<div class="footer">
    © UniBot Admin
</div>
</div>

</body>
</html>

```

### Б.10 Код файлу «stats.html»:

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>UniBot – Статистика</title>

    <style>
        :root {
            --glass: rgba(255, 255, 255, 0.65);
            --border: rgba(255, 255, 255, 0.4);
            --accent: #4f46e5;
            --radius: 18px;
        }

        * {
            box-sizing: border-box;
        }

        body {
            margin: 0;
            min-height: 100vh;
            font-family: "Segoe UI", system-ui, sans-serif;
            background: linear-gradient(135deg, #e9ecf3,
#f8fafc);

```

```

        padding: 40px;
        color: #111827;
    }

    .header {
        display: flex;
        justify-content: space-between;
        align-items: center;
        margin-bottom: 32px;
    }

    h1 {
        margin: 0;
        font-size: 26px;
        font-weight: 600;
    }

    a {
        text-decoration: none;
        color: var(--accent);
        font-size: 14px;
    }

    .cards {
        display: grid;
        grid-template-columns:                repeat(auto-fit,
minmax(220px, 1fr));
        gap: 24px;
    }

    .card {
        background: var(--glass);
        backdrop-filter: blur(16px);
        border-radius: var(--radius);
        padding: 24px;
        border: 1px solid var(--border);
        box-shadow: 0 20px 40px rgba(0,0,0,0.08);
    }

    .value {
        font-size: 38px;
        font-weight: 700;
        margin-bottom: 8px;
        color: #111827;
    }

    .label {
        font-size: 14px;
        color: #374151;
        opacity: 0.85;
    }

```

```

    }

    .updated {
        margin-top: 28px;
        font-size: 13px;
        color: #6b7280;
        text-align: right;
    }
</style>
</head>
<body>

<div class="header">
    <h1>☐ Статистика UniBot</h1>
    <a href="/admin">← Назад до адмін-панелі</a>
</div>

<div class="cards">
    <div class="card">
        <div class="value" id="total_messages">-</div>
        <div class="label">Усього повідомлень</div>
    </div>

    <div class="card">
        <div class="value" id="bot_answers">-</div>
        <div class="label">Відповідей бота</div>
    </div>

    <div class="card">
        <div class="value" id="admin_answers">-</div>
        <div class="label">Відповідей адміністратора</div>
    </div>

    <div class="card">
        <div class="value" id="unknown_questions">-</div>
        <div class="label">Передано адміну</div>
    </div>

    <div class="card">
        <div class="value" id="anonymous_messages">-</div>
        <div class="label">Анонімних звернень</div>
    </div>
</div>

<div class="updated" id="updated_at">
    Оновлення...
</div>

<script>
async function loadStats() {

```

```

        const res = await fetch("/api/stats");
        const data = await res.json();

        document.getElementById("total_messages").textContent =
data.total_messages;
        document.getElementById("bot_answers").textContent =
data.bot_answers;
        document.getElementById("admin_answers").textContent =
data.admin_answers;
        document.getElementById("unknown_questions").textContent =
data.to_admin ?? data.unknown_questions;
        document.getElementById("anonymous_messages").textContent =
data.anonymous_messages;

        document.getElementById("updated_at").textContent =
            "Останне оновлення: " + new Date().toLocaleTimeString();
    }

    // старт
    loadStats();
    // автооновлення
    setInterval(loadStats, 5000);
</script>

</body>
</html>

```

**ДОДАТОК В**  
**Керівництво оператора**

## **В.1 Призначення оператора системи**

Оператором системи є відповідальна особа (адміністратор), яка здійснює керування роботою автоматизованої системи підтримки користувачів UniBot.

Основними завданнями оператора є:

- перегляд звернень користувачів, отриманих через Telegram-бот;
- надання відповідей на запитання студентів;
- обробка анонімних звернень;
- адміністрування базової інформації (інформаційні розділи);
- контроль статистики роботи системи.

## **В.2 Умови доступу до системи та вхід**

Для роботи з системою оператор повинен мати:

- доступ до адміністративної вебпанелі;
- облікові дані (пароль адміністратора);
- стабільне підключення до мережі Інтернет;
- сучасний веббраузер (Google Chrome, Mozilla Firefox, Microsoft Edge тощо).

Доступ до адміністративної частини системи здійснюється через механізм авторизації, що запобігає несанкціонованому використанню функціоналу. Для входу до системи оператор повинен:

- відкрити у браузері адресу адміністративної панелі;
- перейти на сторінку авторизації;
- ввести пароль адміністратора;

Після успішної перевірки облікових даних оператор отримує доступ до основного інтерфейсу керування.

У разі введення некоректного пароля доступ до системи не надається.

### **В.3 Робота з вебсайтом**

#### **В.3.1 Надання відповідей на звернення**

Для відповіді на звернення оператор виконує наступні дії:

- 1) обирає потрібне звернення зі списку;
- 2) вводить текст відповіді у відповідне поле;
- 3) натискає кнопку підтвердження відповіді.

Після цього:

- відповідь автоматично надсилається користувачу в Telegram;
- звернення змінює свій статус на оброблене;
- за необхідності інформація додається до бази знань системи.

Якщо питання не стосується теми чат-бота, оператор може видалити повідомлення аби на основі таких запитів не засмічувалась база знань.

#### **В.3.2 Робота з базовою інформацією**

Базовою інформацією вважається така інформація, яку користувач отримує від бота при натисканні на кнопки (в меню та повідомленнях). Це загальна інформація про студентське самоврядування, яка з часом може змінюватись, тож для підтримки актуальності інформації, оператор має змінювати та оновлювати дану інформацію у відповідному розділі.

#### **В.3.3 Контроль статистики використання бота**

Адміністративна панель надає оператору можливість перегляду статистичних даних, зокрема:

- загальна кількість отриманих повідомлень;
- кількість відповідей, наданих автоматично;
- кількість відповідей адміністратора;
- кількість анонімних звернень;

- кількість записів, доданих до бази знань.

Статистична інформація дозволяє оцінити ефективність роботи системи. Також може бути використана у різноманітних звітах про роботу зі студентами.

**ДОДАТОК Г**  
**Слайди презентації**



Рисунок Г.1 – Слайд №1



Рисунок Г.2 – Слайд №2



Рисунок Г.3 – Слайд №3

## ІНСТРУМЕНТАРІЙ ДЛЯ РОЗРОБКИ

У якості основної мови програмування обрано Python, що зумовлено його високим рівнем абстракції, кросплатформністю та наявністю потужних бібліотек для машинного навчання і семантичного аналізу текстів.

Ключові компоненти:

- бібліотеку python-telegram-bot для забезпечення асинхронної взаємодії з користувачами через месенджер;
- бібліотеку SentenceTransformers для реалізації інтелектуального пошуку відповідей;
- фреймворк FastAPI для створення високопродуктивної серверної частини та вебінтерфейсу адміністратора;
- стандартну бібліотеку для роботи з форматом JSON, що забезпечує надійне збереження та обробку даних системи.

Рисунок Г.4 – Слайд №4



Рисунок Г.5 – Слайд №5

## Алгоритми роботи

```

class RetrievalEngine:
    def __init__(self, data_path: str, threshold: float = 0.6):
        self.data_path = data_path
        self.threshold = threshold

        self.model = SentenceTransformer(
            "paraphrase-multilingual-MiniLM-L12-v2"
        )

        self.questions = []
        self.answers = []

        self._load_data()
  
```

Семантичний пошук

```

async def admin_answer(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if update.effective_user.id != ADMIN_ID:
        return

    if len(context.args) < 2:
        await update.message.reply_text("Введіть ID запиту та відповідь")
        return

    qid = context.args[0].lstrip("/")
    if not qid.isdigit():
        await update.message.reply_text("❌ Введіть ID")
        return

    answer_text = " ".join(context.args[1:])
    q = answer_question(int(qid), answer_text)
    if not q:
        await update.message.reply_text("❌ Питання не знайдено")
        return
  
```

Відповіді користувачам

Рисунок Г.6 – Слайд №6



Рисунок Г.7 – Слайд №7

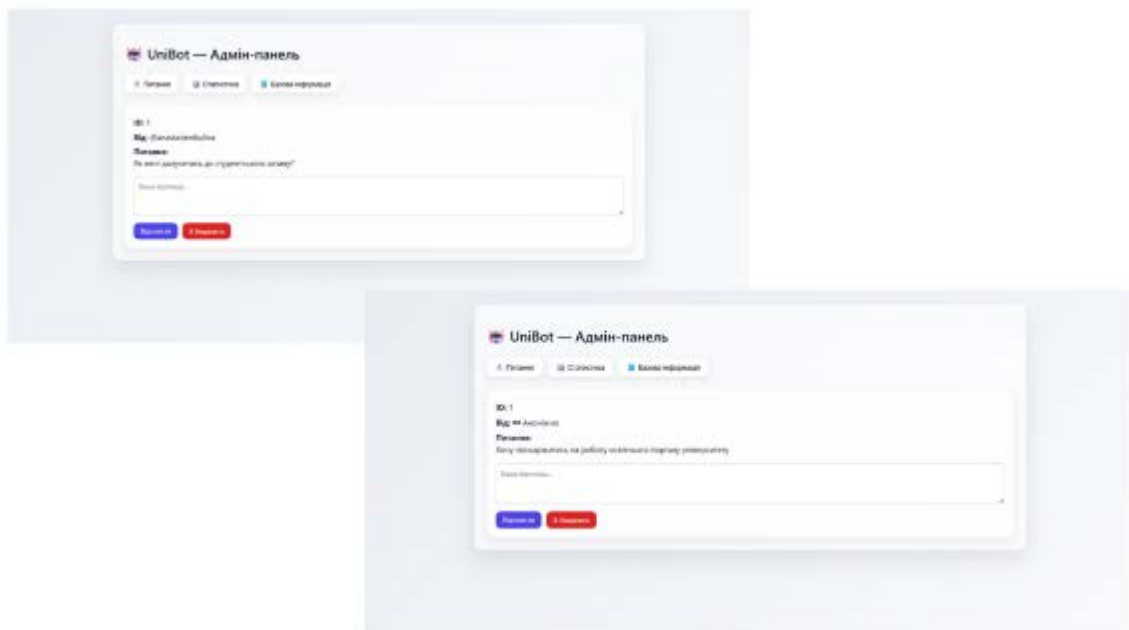


Рисунок Г.8 – Слайд №8

## Висновки

У роботі успішно вирішено завдання підвищення ефективності інформаційної підтримки в освітньому середовищі шляхом створення системи «UniBot», що базується на технологіях машинного навчання. Аналіз недоліків існуючих методів комунікації (відсутність анонімності, перевантаження адміністраторів однотипними запитами) зумовив вибір власної розробки на базі Python та FastAPI, яка забезпечує централізовану взаємодію та надійне збереження даних у форматі JSON/JSONL.

Інтелектуальну основу системи становить модуль семантичного пошуку (retrieval-based), реалізований через бібліотеку SentenceTransformers, що дозволяє точно знаходити відповіді за векторними поданнями запитів. Реалізовано унікальні алгоритми функціонування, зокрема механізм анонімних звернень та алгоритм донавчання, який забезпечує автономне розширення бази знань системи на основі відповідей адміністратора на складні питання.

Практичним результатом є впровадження комплексу, що складається з Telegram-бота для студентів та вебпанелі для адміністрування, стабільність якого підтверджена тестуванням. Система має високу практичну цінність завдяки своїй універсальності та здатності суттєво зменшити часові витрати персоналу, підтверджуючи досягнення мети роботи та наукову новизну у підході до самонавчання.

09

Рисунок Г.9 – Слайд №9