

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ЗАКЛАД  
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут фізики, математики та інформаційних технологій

Кафедра фізико-технічних систем та інформатики

**Скарга Вікторія Ігорівна**

**АВТОМАТИЗАЦІЯ УНІВЕРСИТЕТСЬКИХ ПРОЦЕСІВ ЗА  
ДОПОМОГОЮ РОЗРОБКИ TELEGRAM-БОТА**

**кваліфікаційна робота**

**здобувача вищої освіти першого (бакалаврського) рівня  
освітньої програми «Комп’ютерні науки та інформаційні технології»  
за спеціальністю 122 Комп’ютерні науки**

Особистий підпис \_\_\_\_\_ Вікторія СКАРГА

Науковий керівник \_\_\_\_\_ Галина КОЗУБ,  
кандидат технічних наук, доцент  
кафедри фізико-технічних систем та  
інформатики

Завідувач кафедри \_\_\_\_\_ Юрій КОЗУБ,  
доктор технічних наук, доцент  
кафедри фізико-технічних систем та  
інформатики

Полтава – 2023

**Міністерство освіти і науки України**  
**Державний заклад „Луганський національний університет**  
**імені Тараса Шевченка”**

Факультет (інститут)

Кафедра

Рівень освіти

Спеціальність

Навчально-науковий інститут фізики,  
математики та інформаційних технологій

Фізико-технічних систем та інформатики

перший (бакалаврський)

122 „Комп’ютерні науки”

(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ФТСІ

Юрій КОЗУБ

(підпис)

(ім'я, прізвище)

“ ”

2022 р.

**ЗАВДАННЯ**  
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Скарга Вікторія Ігорівна

(прізвище, ім'я, по батькові )

1. Тема проєкту (роботи) Автоматизація університетських процесів за  
допомогою Telegram-бота

Керівник кваліфікаційної роботи

Козуб Г.О.

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

затверджена наказом по університету

Від“ ” 202 року №

2. Строк подання студентом проєкту (роботи)

3. Вихідні дані до роботи (проєкту) Дослідження поняття, теоретичні  
засади функціонування систем контролю та управління доступом,  
методологію та розробку боту на базі месенджера Telegram для університету,  
обмеження та рекомендації щодо його модернізації.

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно  
розробити) аналіз предметної області.

вибір програмного забезпечення.

розробка чат-боту STEMEdBot на базі месенджера Telegram.

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

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

**6.Консультанти розділів проекту (роботи)**

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

7. Дата видачі завдання „\_\_\_\_\_” \_\_\_\_\_ 2022р.

**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 24 жовтня	
2.	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	До 1 лютого	
3.	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 лютого	
4.	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 1 квітня	
5.	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень квітня	
6.	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 квітня	
7.	Попередній захист роботи на кафедрі	травень	
8.	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
9.	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

**Студент**

\_\_\_\_\_ підпис

**Вікторія СКАРГА****Керівник проекту (роботи)**

\_\_\_\_\_ підпис

**Галина КОЗУБ**

## АНОТАЦІЯ

**Скарга В.І**

**Тема:** Автоматизація університетських процесів за допомогою розробки Telegram-бота

**Спеціальність:** 122 “Комп’ютерні науки”

**Установа:** ДЗ ЛНУ імені Тараса Шевченка, 2023 р.

**Бакалаврська робота містить:** 93 с., 20 рис., 3 додат., 5 табл., 33 джерела.

**Об’єкт дослідження:** інформаційний чат-бот

**Предмет дослідження:** технології розробки автоматизації університетських процесів за допомогою розробки Telegram-бота

**Мета роботи:** дослідження засобів побудови автоматизації університетських процесів за допомогою розробки Telegram-бота

**Результат роботи:** Досліджено особливості сучасних чат-ботів, розглянуто загальні принципи та засоби їх побудови для месенджеру Telegram. Обрано програмні засоби для розробки чат-боту та розроблено чат бот STEMEdBot на базі месенджеру Telegram, який дозволяє автоматизувати університетські процеси.

**Ключові слова:** BOT, TELEGRAM, PYTHON, API

## **ABSTRACT**

**Skarha V.I**

**Theme:** Automation of university processes using the development of a telegram bot

**Speciality:** 122 “Computer Science”

**Institution:** Taras Shevchenko National University of Luhansk, 2023

**Qualification work contains:** 93 pages, 20 images, 3 supplements, 5 tab., 33 sources.

**Research object:** information chatbot

**The subject of research:** technologies for the development of automation of university processes using the development of a telegram bot

**The purpose of the work:** the study of the means of building the automation of university processes using the development of a telegram bot

**The result of the work:** The peculiarities of modern chatbots were studied, the general principles and means of their construction for the Telegram messenger were considered. Software tools for the development of a chat bot were selected and the STEMEdBot chat bot was developed based on the Telegram messenger, which allows automating university processes.

**Keywords:** BOT, TELEGRAM, PYTHON, API

	<b>Міністерство освіти і науки України</b>
	<b>Державний заклад “Луганський національний університет імені Тараса Шевченка”</b>
Факультет (інститут)	Навчально-науковий інститут фізики, математики та інформаційних технологій
	<small>(повна назва)</small>
Кафедра	Фізико-технічних систем та інформатики
	<small>(повна назва)</small>

**ТЕХНІЧНЕ ЗАВДАННЯ**  
на виконання програмної розробки (ПР):  
**“АВТОМАТИЗАЦІЯ УНІВЕРСИТЕТСЬКИХ ПРОЦЕСІВ ЗА  
ДОПОМОГОЮ РОЗРОБКИ TELEGRAM-БОТА”**

Полтава – 2023

## ЗМІСТ

ВСТУП.....	8
1. ХАРАКТЕРИСТИКА ОБ'ЄКТА.....	8
2. ПРИЗНАЧЕННЯ ТОВАРІВ .....	8
3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ.....	9
4. ТЕХНІКО - ЕКОНОМІЧНІ ВИМОГ ДО КІНЦЕВОГО ПРОДУКТУ .....	9
5. ВИМОГИ ДО МАТЕРІАЛІВ І КОМПЛЕКТУЮЧИХ .....	9
6. ЕТАПИ ВИКОНАННЯ ПР .....	9
7. ПРИЙОМ .....	10
8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО.....	11

## **ВСТУП**

1.1 Найменування: Telegram Bot для університету (STEMEdBot)

1.2 Шифр ПР: АДР -1

1.3 Підстава для виконання ПР: Підставою для виконання даної розробки є заява від замовника.

### **1.4 Терміни розробки:**

1.4.1 Початок 30 березня 2023 р.

1.4.2 Закінчення 30 травня 2023 р.

1.5 Фінансується за рахунок коштів замовника. Умови фінансування - за договором № 13 / а і протоколу узгодження ціни № 13 / б.

## **1. ХАРАКТЕРИСТИКА ОБ'ЄКТА**

1.1.1. Розроблений Telegram Bot для університету (STEMEdBot), який має наступний склад:

1.1.2. Користувацький інтерфейс.

1.1.3. Модуль Chat GPT

1.2. До вхідної інформації належать вимоги замовника щодо додатку.

## **2. ПРИЗНАЧЕННЯ ТОВАРІВ**

2.1. Призначення: чат-бот для університету.

### **2.2. Основні критерії ефективності**

2.2.1. Зручний інтерфейс.

2.2.1.1. Користувачі повинні мати можливість комунікувати з ботом за допомогою природньої мови.

2.2.1.2. Керування повинно бути інтуїтивним.

## **3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ**

### **3.1. Загальні вимоги**

3.1.1. Бот працює на смартфонах на базі ОС Android версії 4.4 або вище.

3.1.2. Вимоги до апаратного забезпечення смартфона – не передбачені і можуть встановлюватися розробником програмного комплексу;

3.1.3. Бот повинен мати зручний інтерфейс.

### **3.2. Додаткові вимоги**



3.2.1. Мова програмування Python.

3.2.2. Вимоги до ліцензійного ПЗ не передбачаються і вирішуються замовником

### **3.3. Вимоги до складу і архітектури**

3.3.1. Розробник самостійно вибирає склад і виконує розробку архітектури ПР

3.3.2. Особливих умов до складу та архітектури ПР не передбачено

### **3.4. Вимоги до якості і надійності**

3.4.1. Бот повинен надійно працювати.

3.4.2. Розробник обирає технічні характеристики персонального комп'ютера, смартфона, налаштовує системне програмне забезпечення.

3.4.3. Розробник гарантує роботу графічного додатку без збоїв та переналаштувань.

### **3.5. Вимоги до експлуатації**

3.5.1. Розробник використовує смартфон, на якому бот повинен надійно працювати.

## **4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГ ДО КІНЦЕВОГО ПРОДУКТУ**

Вартість робіт по розробці даної ПР визначається згідно з договором на розробку. Вартість пропонованих аналогів повинна забезпечити економічну доцільність їх застосування.

## **5. ВИМОГИ ДО МАТЕРІАЛІВ І КОМПЛЕКТУЮЩИХ**

### **5.1. Вимоги до екологічної безпеки при експлуатації.**

Не пред'являються.

### **5.2. Спеціальні вимоги до кінцевого продукту.**

Не пред'являються.

### **5.3. Вимоги до безпеки для населення при експлуатації продукції.**

Не пред'являються.

## **6. ЕТАПИ ВИКОНАННЯ ПР**

Етапи виконання ПР можуть уточнюватися згідно календарного плану робіт за погодженням між замовником і виконавцем

№	Етапи виконання роботи	Термін виконання і обсяг робіт	Звітні матеріали
1	Аналіз розробки програмного комплексу та розробка першої версії. Аналіз вимог. Розробка структури. Попереднє тестування.		Фрагмент програмного комплексу на ЕОМ замовника, який виконує всі основні функції і звітна документація п.8.2
2	Коригування структури. Розробка допоміжних функцій. Розробка остаточної версії програмного комплексу і його обробки. Тестування.		Готовий програмний комплекс на ЕОМ замовника і звітна документація п.8.2
3	Доопрацювання окремих модулів і навчання користувачів. Розробка звітних матеріалів по п.8 цього ТЗ		Звітні матеріали згідно з пунктом 8.

## **7. ПРИЙОМ**

### **7.1. Необхідні вимоги для впровадження ПР і завершення робіт.**

Оцінка результатів розробки і доцільність її продовження здійснюється замовником за поданням наступних матеріалів:

- встановлено програмний комплекс на ЕОМ замовника;
- перелік файлів на резервному носії;
- короткий опис роботи ПР і опис всіх файлів, які необхідні для роботи ПР.
- Технічне завдання
- Пояснювальна записка
- Методика тестування

### **7.2. Перелік звітних документів, необхідних для прийняття етапів роботи:**

- короткий опис результатів етапу у вигляді анотованого звіту (для 1 та 2 етапів);
- частковий програмний комплекс на ЕОМ замовника згідно календарного плану робіт;

- акт приймання продукції.

Звітні матеріали подаються у вигляді звітів на папері відповідно до "ДСТУ 3008-2015. Державний стандарт України. Документація. Звіти в сфері науки і техніки. Структура та правила оформлення."

### **7.3. Загальний перелік до прийому звітних документів, макетів, експериментальних зразків.**

До прийому пред'являються: акт здачі-приймання продукції, акт впровадження ПР.

### **7.4.Тестування ПР**

Тестування виконується в розділі "Методика тестування", яка розробляється виконавцем і затверджується замовником.

## **8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО**

Дане технічне завдання може уточнюватися в процесі розробки ПР при узгодженні сторін з оформленням доповнень до ТЗ.

**Міністерство освіти і науки України**  
**Державний заклад «Луганський національний університет**  
**імені Тараса Шевченка»**

Факультет (інститут)

Навчально-науковий інститут фізики,  
математики та інформаційних технологій

(повна назва)

Кафедра

Фізико-технічних систем та інформатики

(повна назва)

**Пояснювальна записка**

до дипломного проєкту (роботи)

**БАКАЛАВРА**

(освітній ступень)

**Автоматизація університетських процесів за допомогою розробки**  
**Telegram-бота**

## ЗМІСТ

ВСТУП.....	15
РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ФУНКЦІОНУВАННЯ СИСТЕМ КОНТРОЛЮ ТА УПРАВЛІННЯ ДОСТУПОМ .....	18
1.1. Сутність поняття та генезис чат-ботів .....	18
1.2. Типологія та атрибути чат-ботів.....	22
1.3. Завдання, функції та вимоги до університетського чат-бота на базі месенджера Telegram.....	26
Висновки до розділу 1 .....	29
РОЗДІЛ 2. МЕТОДОЛОГІЯ ТА РОЗРОБКА ЧАТБОТУ НА БАЗІ МЕСЕНДЖЕРУ TELEGRAM ДЛЯ УНІВЕРСИТЕТУ .....	31
2.1. Вибір інструментів та технологій .....	31
2.2. Архітектурний дизайн системи чат-ботів .....	41
2.3. Досвід користувачів у розробленій системі чат-боту .....	44
2.4. Методи обробки природної мови, що використовуються для взаємодії з користувачем.....	51
Висновки до розділу 2 .....	58
РОЗДІЛ 3. ОБМЕЖЕННЯ ТА РЕКОМЕНДАЦІЇ ЩОДО МОДЕРНІЗАЦІЇ РОЗРОБЛЕНОГО ЧАТБОТУ .....	60
3.1. Обмеження розробленого чатботу .....	60
3.2. Напрямки розвитку та модернізації розробленого чатботу .....	66
Висновки до розділу 3 .....	74
ВИСНОВКИ .....	75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	78

ДОДАТКИ.....	80
Додаток А. Сертифікат апробації результатів дослідження.....	83
Додаток Б. Методика тестування.....	83
Додаток В. Основний код STEMEdBot .....	83

## ВСТУП

### Актуальність роботи

Останніми роками інтеграція технології чат-ботів привертає значну увагу як засіб покращення комунікації та оптимізації різноманітних послуг у різних секторах. У сфері вищої освіти, зокрема в університетах, зростає інтерес до використання можливостей чат-ботів для оптимізації поширення інформації, полегшення адміністративних процесів та ефективної взаємодії зі студентами.

Поява Telegram як широко прийнятого інструменту комунікації зробила його привабливою платформою для навчальних закладів, які прагнуть налагодити ефективні канали взаємодії з різними зацікавленими сторонами. Telegram пропонує цілий ряд можливостей і функцій, які відповідають вимогам університетів, таких як групові чати, канали та можливість інтеграції сторонніх сервісів через інтерфейси прикладного програмування (API). Використовуючи ці можливості, чат-бот може слугувати інтелектуальним віртуальним асистентом, здатним обробляти запити, надавати релевантну інформацію та оперативно відповідати на запити користувачів.

Розробка чат-бота для університетської комунікації має значний потенціал для покращення загального студентського досвіду та підвищення адміністративної ефективності. Студенти часто стикаються з проблемами при пошуку інформації щодо розкладу курсів, дат іспитів, процесів реєстрації та інших питань, пов'язаних з університетом. Традиційно ці запити обробляються різними каналами, включаючи електронну пошту, телефонні дзвінки та особисту взаємодію, що може зайняти багато часу та призвести до затримок. Впровадивши чат-бота в Telegram, університети можуть спростити ці процеси, надавши студентам миттєвий доступ до потрібної інформації в режимі 24/7.

Більше того, використання чат-бота в університетському контексті виходить за рамки адміністративних завдань. Він може слугувати цінним інструментом для сприяння залученню студентів та надання їм персоналізованої підтримки. Завдяки здатності вчитися на взаємодії з користувачами, чат-бот може адаптувати свої відповіді до індивідуальних потреб, пропонуючи пропозиції,

ресурси та настанови, що відповідають академічній програмі або інтересам студента. Крім того, чат-бот на базі Telegram може сприяти одноранговій комунікації, дозволяючи студентам спілкуватися один з одним, створювати навчальні групи та звертатися за допомогою до своїх однолітків.

Однак розробка ефективного та зручного чат-бота для університетської комунікації вимагає ретельного врахування кількох ключових факторів. До них відносяться методи обробки природної мови (NLP), розробка розмовних інтерфейсів, інтеграція з існуючими університетськими системами та базами даних, а також забезпечення конфіденційності та безпеки даних. Ця робота має на меті заглибитися в ці аспекти, надаючи ідеї та рекомендації для університетів, які прагнуть впровадити чат у Telegram.

### **Мета роботи**

Розробка та впровадження чат-бота, спеціально адаптованого для університету, з використанням популярної платформи для обміну повідомленнями Telegram.

Досягнення мети включає розв'язання таких **завдань**:

- аналіз сучасного стану досліджуваної проблеми;
- аналіз існуючих підходів до розробки чат-ботів;
- розробка сценарію роботи чат-бота;
- програмна реалізація функціоналу чат-боту.

**Об'єктом дослідження** є особливості розробки чат-ботів.

**Предметом дослідження** є технології розробки чат-боту на базі месенджеру Telegram.

### **Практичне значення отриманих результатів**

Практичне значення результатів, отриманих при розробці чат-бота для університетської комунікації через Telegram, полягає в підвищенні ефективності комунікації, покращенні досвіду студентів, доступі до інформації в режимі 24/7, персоналізованій підтримці, співпраці з колегами, адміністративній ефективності та потенційній економії коштів. Ці результати в сукупності



сприяють створенню більш ефективного та орієнтованого на студентів університетського середовища, а також оптимізації ресурсів та підвищенню загальної операційної ефективності.

### **Апробація результатів бакалаврської роботи**

Основні положення і результати роботи були апробовані на Міжнародній науково-практичній конференції «Information and its impact on social processes», яка проходила у м. Флоренція, Італія, 3-5 квітня 2023 року. За результатами конференції опубліковано тези за темою «Автоматизація університетських процесів за допомогою розробки Telegram-бота» [3].

### **Структура і обсяг роботи**

Робота складається з вступу, трьох розділів, висновків списку використаних джерел, додатків. Обсяг роботи становить 93 сторіки, обсяг використаної літератури – 33 джерела.

Перший розділ містить опис особливостей сучасних чат-ботів, історію їх походження та розвитку, типологію та атрибути.

У другому розділі проводиться дослідження процесу розробки чат-бота на основі месенджера Telegram, визначаються основні інструменти, архітектура проекту, висвітлюється інтерфейс користувача та методи обробки природної мови, які були використані при розробці бота.

Третій розділ охоплює опис основних обмежень та потенційні напрямки щодо модернізації наявного коду.

Додатки містять сертифікат апробації результатів бакалаврського дослідження на міжнародній науково-практичній конференції, методику тестування програмної розробки та основний python файл чат-боту з описом та авторськими коментарями щодо функціоналу боту.

# РОЗДІЛ 1

## ТЕОРЕТИЧНІ ЗАСАДИ ФУНКЦІОНУВАННЯ СИСТЕМ КОНТРОЛЮ ТА УПРАВЛІННЯ ДОСТУПОМ

### 1.1. Сутність поняття та генезис чат-ботів

Чат-боти, також відомі як розмовні агенти, привернули до себе значну увагу в останні роки завдяки своєму потенціалу революціонізувати взаємодію між людиною та комп'ютером. Чат-бот – це програмне забезпечення, яке використовує методи штучного інтелекту (ШІ) та обробки природної мови (ПМ) для імітації розмов з користувачами, подібних до людських. Ці інтелектуальні агенти призначені для розуміння і реагування на запити або команди користувача, що робить їх цінними інструментами для автоматизації підтримки клієнтів, надання інформації та залучення користувачів у різних сферах [5].

Історія чат-ботів налічує кілька десятиліть і охоплює значні досягнення в галузі технологій та штучного інтелекту. На еволюцію чат-ботів вплинуло прагнення створити розумні машини, здатні брати участь у розмовах, подібних до людських.



Рис. 1.1. Еволюція чат-ботів: Хронологія розвитку

Джерело: розроблено авторкою за [6; 8; 10; 19; 22].

ELIZA (1966): ELIZA, розроблена Джозефом Вайценбаумом з Массачусетського технологічного інституту, вважається одним з перших прикладів чат-бота. ELIZA використовував методи зіставлення шаблонів для імітації психотерапевта, залучаючи користувачів до текстових розмов. Хоча відповіді ELIZA були скриптовими і базувалися на чітких правилах, вона продемонструвала потенціал для імітації взаємодії, подібної до людської [10].

PARRY (1972): PARRY, створена Кеннетом Колбі, мала на меті імітувати пацієнта з параноїдальною шизофренією під час психіатричного інтерв'ю. Розробка PARRY продемонструвала потенціал чат-ботів для імітації різних особистостей і психічних станів [22].

A.L.I.C.E (1995), розроблений Річардом Воллесом, став важливою віхою в розвитку чат-ботів. A.L.I.C.E використовував методи обробки природної мови та велику базу знань для ведення діалогів. Він кілька разів вигравав Премію Лебнера, щорічний конкурс для оцінки інтелекту чат-ботів [19].

SmarterChild (2001): SmarterChild, розроблений компанією ActiveBuddy, був одним із перших чат-ботів, який здобув широку популярність. Він працював на платформах обміну миттєвими повідомленнями, таких як AOL Instant Messenger (AIM) і MSN Messenger, надаючи користувачам широкий спектр функцій, включаючи оновлення погоди, новини та загальну інформацію [6].

Siri (2010): Siri від Apple стала важливою віхою в розвитку чат-ботів, запровадивши голосову взаємодію на мобільних пристроях [8].

Watson (2011), озроблений компанією IBM, Watson привернув до себе значну увагу, коли переміг у вікторині «Jeopardy!», перемігши людей-чемпіонів. Watson продемонстрував потенціал чат-ботів на основі технологій машинного навчання та когнітивних обчислень. Він також продемонстрував здатність розуміти складні запити і генерувати точні відповіді, аналізуючи величезні обсяги даних [4].

Поява платформ для чат-ботів, таких як Facebook Messenger, Telegram і Slack, надала розробникам доступні фреймворки та API для створення та

розгортання чат-ботів. Ці платформи пропонують інструменти та інтеграції, які полегшують процес розробки, сприяючи поширенню чат-ботів у різних галузях.

Завдяки досягненням у галузі глибокого навчання, обробки природної мови та нейронних мереж, сучасні чат-боти стали більш досконалими та потужними. Чат-боти зі штучним інтелектом, такі як Google Assistant, Amazon Alexa і Microsoft Cortana, використовують алгоритми машинного навчання для постійного вдосконалення своїх розмовних здібностей і пропонують персоналізований досвід.

Історія чат-ботів свідчить про постійний розвиток технологій і можливостей штучного інтелекту, що дає змогу чат-ботам ставати все більш інтелектуальними та інтерактивними. Від ранніх систем, заснованих на правилах, до сучасних розмовних агентів зі штучним інтелектом, чат-боти еволюціонували, щоб надавати широкий спектр послуг – від підтримки клієнтів і віртуальних асистентів до освітніх інструментів і розваг. Оскільки дослідження і розробки в галузі чат-ботів тривають, ми можемо очікувати на подальший прогрес, що дозволить чат-ботам стати ще більш схожими на людей і універсальними у своїй взаємодії.

З огляду наведеного раніше стає очевидним, що основною концепцією, що лежить в основі чат-ботів, є здатність інтерпретувати та генерувати природну мову. Завдяки досягненням у галузі машинного навчання та NLP, чат-боти можуть розуміти та обробляти текстову або усну інформацію від користувачів, аналізувати контекст і генерувати відповідні відповіді. Ця здатність дозволяє чат-ботам розуміти складні запити, витягувати релевантну інформацію та надавати змістовні та контекстуально відповідні відповіді.

Чат-боти можна розділити на два основні типи: чат-боти на основі правил і чат-боти на основі машинного навчання [2]. Чат-боти на основі правил працюють за заздалегідь визначеними правилами та шаблонами, що дозволяє їм реагувати на певні вхідні дані або запускати певні дії. Ці чат-боти обмежені у своїй здатності обробляти складні запити або розуміти людську мову.

З іншого боку, чат-боти на основі машинного навчання використовують алгоритми штучного інтелекту для навчання на основі даних і покращення своєї роботи з часом. Ці чат-боти використовують такі методи, як розуміння природної мови (NLU) та генерація природної мови (NLG), щоб покращити свої можливості обробки мови. Аналізуючи величезні обсяги навчальних даних, чат-боти на основі машинного навчання можуть розпізнавати шаблони, визначати наміри і генерувати відповідні відповіді, пропонуючи тим самим більш інтерактивний і динамічний досвід спілкування.

Поява платформ для обміну повідомленнями, таких як Telegram, ще більше прискорила впровадження чат-ботів. Telegram, з його великою базою користувачів та зручними для розробників функціями, є зручною та широкодоступною платформою для розгортання чат-ботів. Розробники можуть використовувати API Telegram Bot для створення та інтеграції чат-ботів у платформу месенджера, забезпечуючи безперешкодну взаємодію з користувачами [16].

Концепція чат-ботів має великі перспективи для різних галузей і застосувань. У сфері обслуговування клієнтів чат-боти можуть надавати миттєву підтримку, вирішувати типові запитання та зменшувати навантаження на людей. Їх також можна використовувати в електронній комерції, щоб допомагати користувачам знаходити товари, надавати персоналізовані рекомендації та полегшувати транзакції. Крім того, чат-боти мають потенціал для покращення освітнього процесу, автоматизації повторюваних завдань і спрощення процесів пошуку інформації.

Отже, чат-боти є значним проривом у взаємодії людини з комп'ютером. Використовуючи методи ШІ та NLP, чат-боти можуть розуміти і генерувати відповіді природною мовою, що дозволяє їм надавати цінні послуги в різних сферах. Поява платформ для обміну повідомленнями, таких як Telegram, забезпечує відповідну екосистему для розробки та розгортання чат-ботів, що ще більше розширює сферу застосування та вплив цих інтелектуальних агентів. Очікується, що з розвитком досліджень і розробок у галузі чат-ботів ці розмовні

агенти відіграватимуть дедалі важливішу роль у перетворенні різних аспектів нашого повсякденного життя.

## 1.2. Типологія та атрибути чат-ботів

Чат-боти можна класифікувати за різними типологіями на основі їхніх функціональних можливостей, технологій, що лежать в їхній основі, та методів взаємодії. Розуміння типології та атрибутів цієї технології має важливе значення для проектування та розробки ефективних чат-ботів. Саме тому у цьому розділі ми розглянемо різні типології та атрибути чат-ботів.

Таблиця 1.1

Типологія чат-ботів

Типологія	Опис
Чат-боти на основі правил	Оперують заздалегідь визначеними правилами та шаблонами для надання конкретних відповідей
Чат-боти на основі пошуку	Отримують заздалегідь визначені відповіді з бази даних або бази знань
Генеративні чат-боти	Динамічно генерують відповіді, використовуючи методи машинного навчання та NLP
Чат-боти, орієнтовані на завдання	Виконують конкретні завдання або дії на основі запитів користувачів
Чат-боти зі штучним інтелектом	Використовують передові технології штучного інтелекту для покращення розмовних здібностей та виконання запитів користувачів

Джерело: розроблено авторкою за [2].

Чат-боти на основі правил працюють за заздалегідь визначеними правилами та шаблонами. Ці чат-боти призначені для розпізнавання певних ключових слів або фраз, введених користувачем, і запуску заздалегідь запрограмованих відповідей або дій. Чат-боти на основі правил відносно прості і підходять для виконання простих і передбачуваних завдань. Вони зазвичай використовуються в сценаріях обслуговування клієнтів для надання заздалегідь

визначеної інформації або виконання певних дій на основі даних, введених користувачем.

Відповідно до своєї назви, чат-боти на основі пошуку використовують заздалегідь визначені відповіді, що зберігаються в базі даних або базі знань програми. Вони аналізують вхідні дані користувача, зіставляють їх із заздалегідь визначеними шаблонами або намірами і надають найбільш релевантну відповідь з бази даних. Ці чат-боти ефективні в наданні контекстуально точних відповідей на основі заздалегідь визначених знань. Чат-боти на основі пошуку зазвичай використовуються для підтримки клієнтів і систем поширених запитань (FAQ).

Генеративні чат-боти використовують обробку природної мови та методи машинного навчання для динамічного генерування відповідей. Ці чат-боти не покладаються на заздалегідь визначені відповіді, а навчаються на основі великих наборів даних, щоб генерувати контекстуально відповідні та послідовні відповіді. Генеративні чат-боти здатні обробляти складніші та різноманітніші запити користувачів, забезпечуючи більш інтерактивні та схожі на людські розмови. Їх часто використовують у віртуальних помічниках, персоналізованих системах рекомендацій та інтерактивних програмах для розповіді історій.

Цілеспрямовані чат-боти призначені для виконання конкретних завдань або дій на основі запитів користувачів. Ці чат-боти створені для роботи з чітко визначеними доменами або процесами і зосереджені на досягненні конкретних цілей. Цілеспрямовані чат-боти чудово справляються з наданням допомоги та спрощенням транзакцій у таких сферах, як електронна комерція, бронювання готелів, авіаквитків і планування зустрічей.

На останок, розглянемо чат-ботів зі штучним інтелектом, які використовують передові технології, такі як машинне навчання, розуміння природної мови (NLU) та генерація природної мови (NLG), щоб покращити свої розмовні здібності. Ці чат-боти безперервно навчаються та адаптуються до взаємодії з користувачами, покращуючи розуміння намірів користувачів та вдосконалюючи свої відповіді з часом. Чат-боти зі штучним інтелектом

пропонують більш персоналізовані та контекстно-орієнтовані розмови, що робить їх придатними для широкого спектру застосувань.

Окрім типологій, чат-боти володіють різними атрибутами, які впливають на їхню функціональність та ефективність. У табл. 1.2 наведено ключові атрибути чат-ботів.

Таблиця 1.2

### Атрибути чат-ботів

Атрибут	Опис
Обробка природної мови (NLP)	Дозволяє розуміти та генерувати відповіді, подібні до людських
Аналіз настроїв	Інтерпретує емоційний тон або настрій запиту користувача
Мультилінгвістична інтерпретація	Підтримує розуміння та реагування на різних мовах
Контекстуальна обізнаність	Підтримує зв'язність і розуміння історії та контексту розмови
Інтеграція з API та системами	Дозволяє взаємодіяти із зовнішніми системами та сервісами
Голосова та текстова взаємодія	Підтримує як голосову, так і текстову взаємодію з користувачем
Аналітика та звітність	Збирає та аналізує дані про взаємодію з користувачами для розуміння та оптимізації

Джерело: [2].

Чат-боти використовують методи NLP, щоб розуміти та інтерпретувати вхідні дані користувача, витягувати наміри та аналізувати контекст розмови. NLP дозволяє чат-ботам розуміти і генерувати відповіді, подібні до людських, що робить взаємодію більш природною та інтуїтивно зрозумілою.

Аналіз настрою дозволяє чат-ботам розуміти та інтерпретувати емоційний тон або настрій користувачів. Аналізуючи настрої, чат-боти можуть емпатично реагувати, адаптувати свій тон і надавати відповідну підтримку або рекомендації.

Крім того, існують багатомовні чат-боти, які здатні розуміти запити користувачів і відповідати на них різними мовами. Ця властивість особливо цінна



для глобальних додатків і служб підтримки клієнтів, які обслуговують користувачів з різним мовним походженням.

Контекстна обізнаність дозволяє чат-ботам підтримувати цілісне розуміння історії розмови та контексту користувача. Ця властивість дозволяє чат-ботам надавати релевантні відповіді, запам'ятовувати попередні взаємодії та брати участь у більш змістовних і персоналізованих розмовах.

Чат-боти також можуть інтегруватися із зовнішніми системами, базами даних та API для отримання або оновлення інформації, виконання дій або отримання даних. Цей атрибут дозволяє чат-ботам надавати інформацію в режимі реального часу, виконувати транзакції та взаємодіяти з іншими програмами або сервісами.

Чат-боти можуть підтримувати як голосову, так і текстову взаємодію, дозволяючи користувачам взаємодіяти з ними за допомогою різних форматів повідомлень. Цей атрибут забезпечує гнучкість і враховує вподобання користувачів, роблячи взаємодію більш зручною та доступною.

Крім того, чат-боти можуть збирати та аналізувати дані про взаємодію з користувачами, включаючи запити користувачів, шаблони відповідей та задоволеність користувачів. Цей атрибут дозволяє розробникам чат-ботів отримати уявлення про поведінку користувачів, визначити сфери для вдосконалення та оптимізувати роботу чат-бота.

Таким чином, розуміння типології та атрибутів чат-ботів дає цінну інформацію для вибору відповідної архітектури чат-бота, розробки ефективних діалогових потоків і надання користувачеві змістовного досвіду. Залежно від конкретних вимог і цілей, розробники можуть вибрати найбільш підходящу типологію і включити відповідні атрибути для створення чат-ботів, які відповідають потребам і очікуванням користувачів.

### 1.3. Завдання, функції та вимоги до університетського чат-бота на базі месенджера Telegram

У сучасну епоху цифрових комунікацій університети шукають інноваційні способи покращити свої послуги та ефективно взаємодіяти зі студентами. Одним із таких підходів є розробка університетського чат-бота на базі платформи месенджера Telegram. Цей розділ має на меті обговорити завдання, функції та вимоги, які повинен мати університетський чат-бот, щоб ефективно задовольняти потреби студентів та університетської спільноти.

Таблиця 1.3

**Завдання університетських чат-ботів**

Завдання	Опис
Пошук інформації	Отримувати та надавати точну та актуальну інформацію про університет, включаючи програми, вимоги до вступу, об'єкти університетського містечка, події та політику тощо.
Підтримка курсів та програм	Надавати інформацію про доступні курси, вимоги до програм, передумови та процедури реєстрації на курси.
Навігація по кампусу	Допомагати студентам орієнтуватися в університетському кампусі, надаючи вказівки, розташування будівель і карти.
Підтримка студентів	Пропонувати підтримку студентам, відповідаючи на запити, пов'язані з адміністративними процесами, фінансовою допомогою, студентськими послугами та академічними консультаціями.
Оновлення подій та новин	Надавати своєчасні оновлення про університетські події, воркшопи, семінари та важливі новини, щоб тримати студентів в курсі подій.
Поширені запитання та усунення несправностей	Відповідати на запитання, які часто ставлять студенти, усунення поширених проблем та надання рішень або відповідних ресурсів

Джерело: розроблено авторкою за [1].

Згідно з даними табл. 1.3, чат-бот повинен вміти отримувати точну та актуальну інформацію про різні аспекти діяльності університету, зокрема про навчальні програми, умови вступу, інфраструктуру кампусу, події та політику.

Він також повинен надавати інформацію про доступні курси, вимоги до програм, передумови та процедури реєстрації на курси і допомагати студентам орієнтуватися в університетському кампусі, вказуючи напрямки, розташування будівель та мапи.

Чат-бот також повинен надавати підтримку студентам, відповідаючи на їхні запити, пов'язані з адміністративними процесами, фінансовою допомогою, студентськими послугами та академічними консультаціями, тим самим надаючи своєчасні оновлення про університетські події, воркшопи, семінари та важливі новини, щоб тримати студентів в курсі подій тощо.

Таблиця 1.4

#### Функції університетського чат-бота

Функція	Опис
Обробка природної мови (NLP)	Використовує методи NLP, щоб точно розуміти та інтерпретувати запити користувачів, навіть з урахуванням варіацій у формулюваннях та мові.
Персоналізація	Персоналізує відповіді та рекомендації на основі індивідуальних профілів та вподобань студентів.
Багатомовна підтримка	Підтримує кілька мов, щоб задовольнити потреби різноманітного міжнародного студентства.
Усвідомлення контексту	Підтримує контекстну обізнаність протягом усієї розмови, щоб надавати послідовні та доречні відповіді.
Інтеграція з університетськими системами	Інтегрується з університетськими системами, такими як студентські інформаційні системи, системи управління курсами та календарі подій, щоб отримувати дані в режимі реального часу та надавати точну інформацію.
Голосова та текстова взаємодія	Підтримує як голосову, так і текстову взаємодію, дозволяючи студентам обирати бажаний спосіб спілкування.

Джерело: [1].

Чат-бот також повинен використовувати методи NLP, щоб точно розуміти та інтерпретувати запити користувачів, навіть з варіаціями у формулюваннях та мові. Разом з налаштованою системою аутентифікації користувачів, бот повинен

мати можливість персоналізувати відповіді та рекомендації на основі індивідуальних профілів та вподобань студентів.

Крім того, чат-бот повинен буди доступний у кількох мов, щоб задовольнити потреби студентів за обміном. Чат-бот також повинен підтримувати контекстну обізнаність протягом усієї розмови, щоб надавати узгоджені та релевантні відповіді і мати здатність до інтеграції з різними університетськими системами, такими як студентські інформаційні системи, системи управління курсами та календарі подій, щоб отримувати дані в режимі реального часу та надавати точну інформацію.

*Таблиця 1.5*

#### **Вимоги до університетського чат-бота**

<b>Вимоги</b>	<b>Опис</b>
Зручний інтерфейс	Мати інтуїтивно зрозумілий і зручний інтерфейс, щоб сприяти безперешкодній взаємодії та забезпечити позитивний досвід користувача.
Масштабованість та продуктивність	Бути спроектованим таким чином, щоб обробляти велику кількість запитів користувачів одночасно без шкоди для швидкості реагування та продуктивності.
Безпека та конфіденційність	Дотримуйтесь суворих протоколів безпеки та забезпечуйте конфіденційність даних користувачів, дотримуючись відповідних правил та інструкцій.
Постійне вдосконалення та навчання	Використовувати можливості машинного навчання, щоб постійно вдосконалювати відповіді та краще розуміти наміри користувачів з часом.
Механізм зворотного зв'язку	Включити механізм зворотного зв'язку для збору відгуків та пропозицій користувачів, що дозволить університету вносити необхідні покращення та розширювати функціональність.

Джерело: розроблено авторкою за [1].

Чат-бот також повинен бути спроектований таким чином, щоб обробляти велику кількість запитів користувачів одночасно без шкоди для швидкості реагування та продуктивності. При цьому система має дотримуватися суворих

протоколів безпеки та забезпечувати конфіденційність даних користувачів, дотримуючись відповідних правил та інструкцій.

Очевидно також, що чат-бот повинен мати механізм зворотного зв'язку для збору відгуків та пропозицій користувачів, що дозволить університету вносити необхідні покращення та розширювати функціональність чат-бота.

Отже, університетський чат-бот на базі платформи месенджера Telegram може значно покращити комунікацію та підтримку студентів. Ефективно виконуючи такі завдання, як пошук інформації, допомога у виборі курсів, навігація кампусом, підтримка студентів та оновлення подій, чат-бот може оптимізувати роботу та надавати своєчасну і точну інформацію студентам. Виконуючи функціональні вимоги, такі як обробка природної мови, персоналізація, інтеграція з університетськими системами та багатомовна підтримка, чат-бот може забезпечити інтерактивний та зручний для користувача досвід. Крім того, виконання вимог щодо масштабованості, безпеки, постійного вдосконалення та механізмів зворотного зв'язку забезпечує довгостроковий успіх та адаптивність чат-бота. Зрештою, добре розроблений університетський чат-бот може слугувати цінним ресурсом і покращувати загальний досвід студентів в університетській спільноті.

## **Висновки до розділу 1**

1. Чат-боти є революційним досягненням у сфері взаємодії людини та комп'ютера, що революціонізує спосіб взаємодії людей з технологіями. Завдяки використанню методів штучного інтелекту (ШІ) та обробки природної мови (ПМ), чат-боти мають здатність розуміти і генерувати відповіді природною мовою, що дозволяє їм пропонувати цінні послуги в різних сферах. Поява платформ для обміну повідомленнями, таких як Telegram, створила ідеальне середовище для створення та розгортання чат-ботів, ще більше розширивши їхнє охоплення та вплив. Оскільки сфера досліджень і розробок чат-ботів продовжує розвиватися, очікується, що ці розмовні агенти відіграватимуть дедалі помітнішу роль у перетворенні різних аспектів нашого повсякденного життя.

2. Розуміння типології та атрибутів чат-ботів має важливе значення для вибору відповідної архітектури чат-ботів, розробки ефективних діалогових потоків і забезпечення повноцінного користувацького досвіду. Розуміючи ці типології та атрибути, розробники можуть приймати обґрунтовані рішення щодо найбільш підходящої структури чат-бота і включати відповідні характеристики для створення чат-ботів, які відповідають потребам і очікуванням користувачів. Таке розуміння полегшує розробку та впровадження чат-ботів, які ефективно відповідають конкретним вимогам і цілям, що в кінцевому підсумку підвищує рівень задоволеності та залученості користувачів.

3. Впровадження чат-бота на базі Telegram в університетському середовищі має величезний потенціал для покращення комунікації та підтримки студентів. Ефективно виконуючи різні завдання, зокрема пошук інформації, рекомендації щодо вибору курсу, навігацію кампусом, допомогу студентам та оновлення подій, чат-бот може оптимізувати роботу та надавати студентам своєчасну і точну інформацію. Відповідність функціональним вимогам, таким як обробка природної мови, персоналізація, інтеграція з університетськими системами та багатомовна підтримка, дозволяє чат-боту надавати інтерактивний та зручний для користувача досвід. Крім того, забезпечення масштабованості, безпеки, постійного вдосконалення та механізмів зворотного зв'язку гарантує довгостроковий успіх та адаптивність чат-бота. Зрештою, добре розроблений університетський чат-бот слугує цінним ресурсом, що значно покращує загальний досвід студентів в університетській спільноті.

## РОЗДІЛ 2

### МЕТОДОЛОГІЯ ТА РОЗРОБКА ЧАТБОТУ НА БАЗІ МЕСЕНДЖЕРУ TELEGRAM ДЛЯ УНІВЕРСИТЕТУ

#### 2.1. Вибір інструментів та технологій

Метою цього дослідження є розробку чат-боту на базі месенджера Telegram, призначеного для виконання різних завдань, таких як надання інформації про курс, допомоги у написанні текстів, розв'язанні задач зі статистики, перекладання тексту на англійську мову за допомогою штучного інтелекту штучним інтелектом, а також встановлення нагадувань про виконання завдань.

#### Python

Для виконання поставлених завдань основною мовою програмування був обраний Python – універсальна і широко поширена мова програмування, яка набула популярності серед розробників для створення чат-ботів.

Елегантний синтаксис і зручність читання Python роблять її ідеальним вибором для розробки чат-ботів. Її інтуїтивно зрозуміла природа дозволяє розробникам писати чистий і лаконічний код, підвищуючи тим самим продуктивність і ремонтпридатність коду.

Python також може похвалитися великою колекцією бібліотек, фреймворків та інструментів, які значно спрощують процес розробки. Такі бібліотеки, як NLTK (Natural Language Toolkit), spaCy та TensorFlow надають потужні можливості обробки природної мови (NLP), що дозволяє чат-ботам розуміти та розумно реагувати на запити користувачів [21].

Сумісність Python з популярними фреймворками для машинного навчання та штучного інтелекту, зокрема scikit-learn і Keras, полегшує інтеграцію передових методів штучного інтелекту в розробку чат-ботів [21]. Це дозволяє чат-ботам вчитися на взаємодії з користувачами, покращувати свої відповіді з часом і надавати персоналізований досвід.

Таким чином, простота та багата екосистема Python прискорює процес розробки чат-ботів, скорочуючи час виходу на ринок. Крім того, Python має велику та активну спільноту розробників, яка надає широкі ресурси, навчальні посібники та бібліотеки з відкритим вихідним кодом для розробки чат-ботів.

А масштабованість Python дозволяє додаткам чат-ботів справлятися зі зростаючим трафіком користувачів та адаптуватися до вимог, що постійно змінюються.

Однак, варто зазначити, що інтерпретована природа Python може призвести до зниження швидкості виконання порівняно з мовами нижчого рівня, хоча цей недолік можна усунути за допомогою методів оптимізації та використання зовнішніх бібліотек.

Також існує певна проблема з споживання пам'яті: Python може бути відносно ресурсовитратним інструментом порівняно з мовами на кшталт C++, особливо для ресурсоємних додатків чат-ботів. Однак, певні стратегії управління пам'яттю можуть пом'якшити цю проблему.

Таким чином, Python є непоганим вибором для розробки чат-ботів, пропонуючи численні переваги, такі як спрощений синтаксис, широка підтримка бібліотек і безперешкодна інтеграція з фреймворками штучного інтелекту та машинного навчання. Завдяки своїй гнучкості та масштабованості вона добре підходить для академічних чат-ботів, надаючи цінну підтримку в таких сферах, як академічне письмо, пошук інформації та мовна допомога. Хоча існують проблеми, пов'язані з продуктивністю та споживанням пам'яті.

### **BotFather**

Щодо Telegram на базі якого і буде працювати розроблений бот, то ця платформа пропонує BotFather, спеціалізований інструмент для створення та управління чат-ботами.

Щоб створити чат-бота за допомогою BotFather, користувачі повинні мати обліковий запис у Telegram. Після входу в систему користувачі можуть шукати «BotFather» у додатку Telegram або отримати доступ до нього через веб-



інтерфейс. Знайшовши BotFather, користувачі повинні розпочати чат і слідувати інструкціям на екрані, щоб створити нового бота (рис. 2.1).

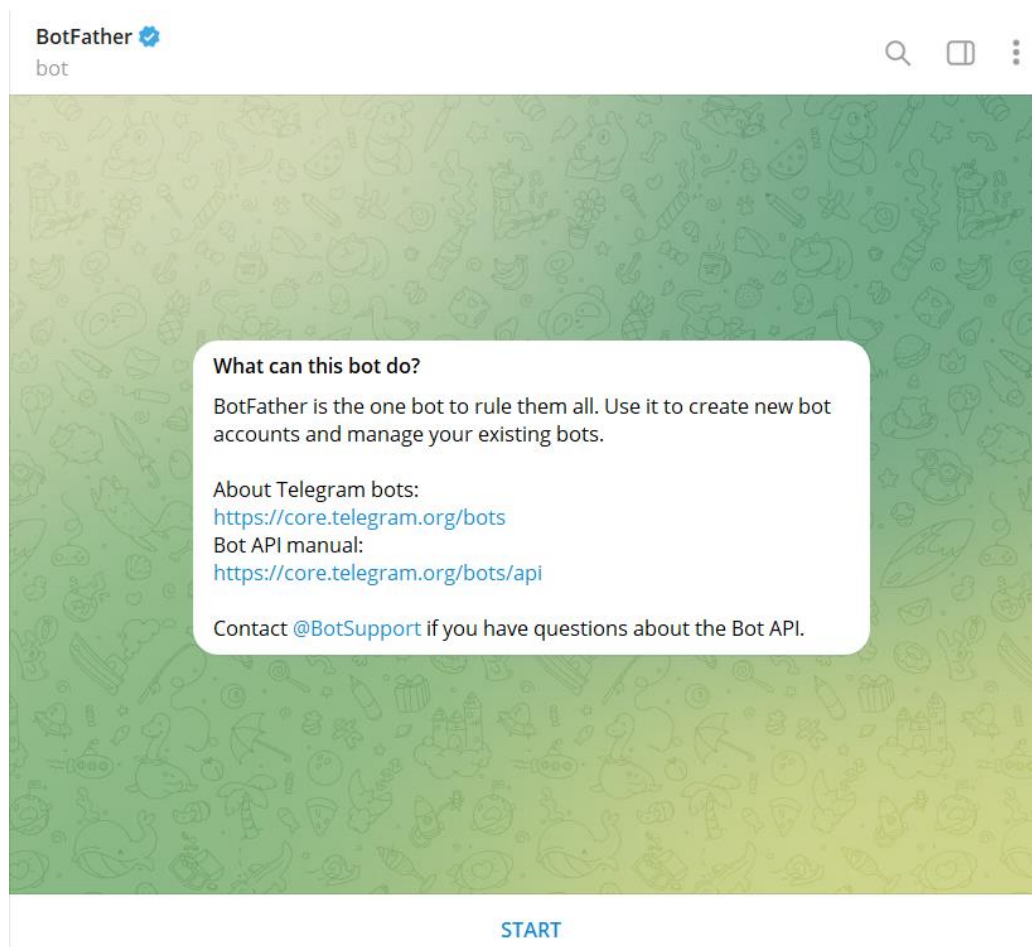


Рис. 2.1. Стартовий екран BotFather (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

Під час створення чат-бота BotFather пропонує користувачам надати ім'я та опис свого чат-бота. Ці дані допомагають ідентифікувати та відрізнити чат-бота від інших інших користувачам.

Після вказівки імені та опису чат-бота BotFather генерує API-токен, який слугує унікальним ідентифікатором чат-бота. Цей токен необхідний для інтеграції чат-бота з платформою Telegram та уможливлення комунікації з користувачами.

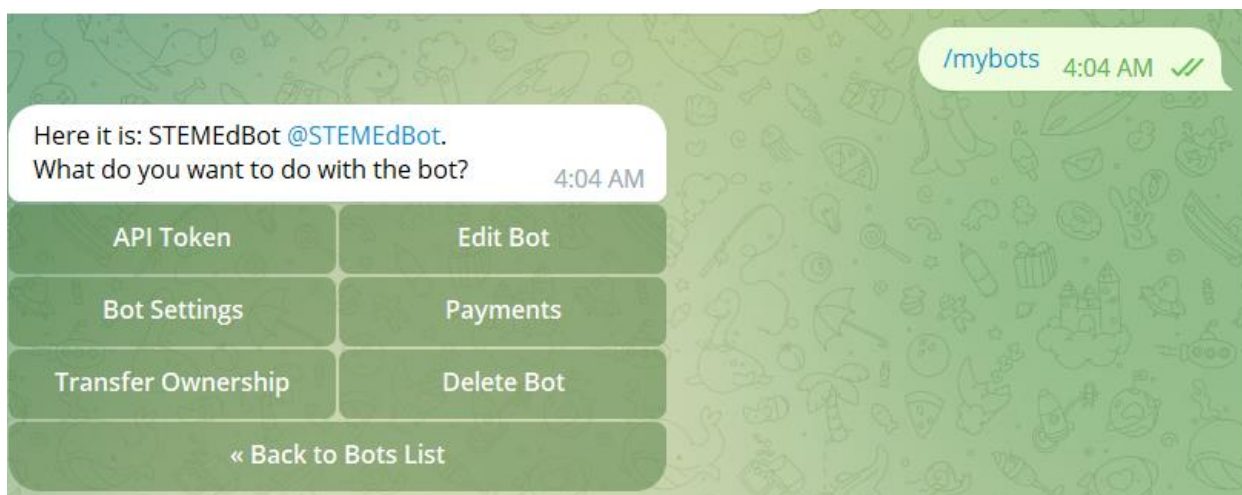


Рис. 2.2. Головне меню управління ботом у BotFather (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

BotFather дозволяє користувачам налаштовувати різні аспекти поведінки свого чат-бота. Користувачі можуть налаштувати автоматичні відповіді на певні ключові слова або шаблони, визначити функціонал обробки команд і вказати відповіді за замовчуванням для незбіжних запитів. Крім того, користувачі можуть визначити мовні уподобання чат-бота, привітання та інші персоналізовані налаштування.

BotFather також надає функції управління користувачами, що дозволяють налаштовувати взаємодію чат-бота з окремими користувачами. Це включає в себе управління правами доступу, обробку налаштувань користувача та підтримку специфічних налаштувань користувача.

За допомогою BotFather користувачі можуть запрограмувати чат-ботів на автоматичне реагування на певні ключові слова або шаблони в повідомленнях користувачів. Ця функція дозволяє чат-боту надавати релевантну інформацію або виконувати заздалегідь визначені дії на основі запитів користувача.

BotFather підтримує створення кастомних команд, які запускають певні дії в чат-боті. Ці команди можна запрограмувати на виконання завдань, отримання інформації із зовнішніх джерел або взаємодію з API, що розширює можливості чат-бота.

За допомогою BotFather розробники чат-ботів можуть створювати інтерактивний досвід для користувачів. Це включає реалізацію кнопок, меню та вбудованих запитів, які полегшують інтуїтивно зрозумілу взаємодію з користувачем та підвищують його залученість.

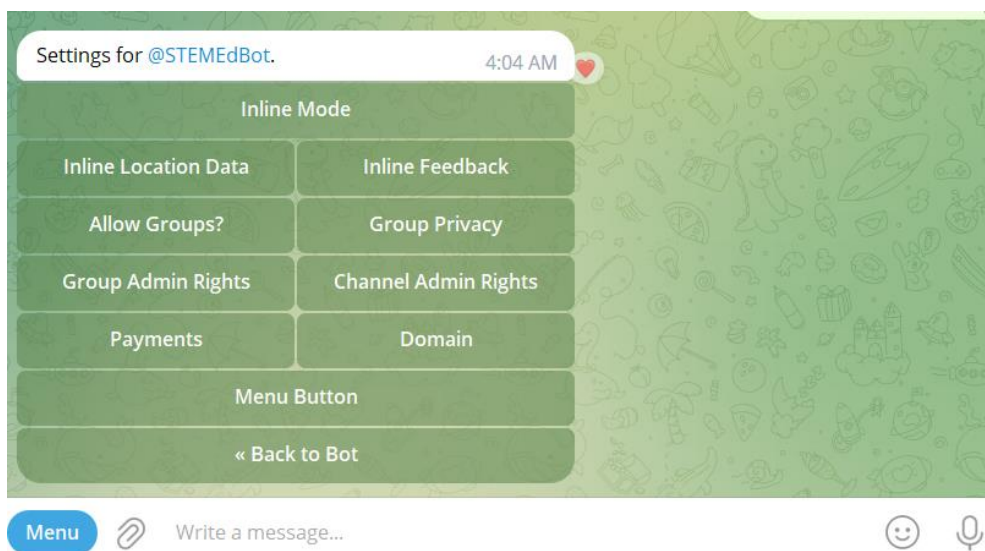


Рис. 2.3. Меню управління правами бота у BotFather (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

Крім того, BotFather дозволяє розробникам чат-ботів налаштовувати системи сповіщень, що дозволяє чат-боту надсилати нагадування, платіжні шлюзи, сповіщення або оновлення користувачам через певні проміжки часу або за певних подій. Таким чином, Telegram BotFather пропонує потужну платформу для створення та налаштування чат-ботів у месенджері Telegram.

### **Alogram**

Для написання коду боту був використаний Alogram – це фреймворк на Python, спеціально розроблений для створення Telegram-ботів. Він пропонує комплексний та ефективний набір інструментів для легкого створення потужних ботів. За допомогою Alogram розробники можуть використовувати функціональність Telegram Bot API і безперешкодно взаємодіяти з користувачами за допомогою повідомлень, команд і зворотних дзвінків.

Фреймворк надає зручний інтерфейс, який спрощує процес створення та управління ботами. Він абстрагується від складнощів API Telegram Bot,

дозволяючи розробникам зосередитися на логіці та функціональності бота. AIOgram пропонує цілий ряд функцій та утиліт, серед яких:

1. Асинхронна підтримка: AIOgram створений з урахуванням асинхронного програмування, що робить його придатним для високопродуктивних ботів (рис. 2.3).



Рис. 2.4. Блок-схема процесу обробки повідомлень, команд та зворотних викликів користувача в AIOgram

Джерело: розроблено авторкою за [20].

Вона використовує бібліотеку Python `asuncio`, що дозволяє одночасно виконувати кілька завдань.

2. Обробка повідомлень: AIOgram спрощує обробку вхідних повідомлень від користувачів. Бібліотека надає інтуїтивно зрозумілі методи для обробки текстових повідомлень, медіафайлів та інших типів контенту. Розробники можуть реалізовувати власну логіку на основі отриманих повідомлень, наприклад, розбирати вхідні дані користувача, виконувати дії або надавати відповіді.

3. Обробка команд: фреймворк пропонує вбудовану підтримку для обробки команд. Розробники можуть визначати команди за допомогою декораторів або вручну вказуючи їхні деталі. AIOgram автоматично розпізнає і направляє вхідні команди до відповідних функцій-обробників, що дозволяє легко реалізувати командну взаємодію.

4. Обробка запитів на зворотний дзвінок: Alogram спрощує обробку запитів на зворотний дзвінок, які є взаємодією користувача з вбудованою клавіатурою або іншими інтерактивними елементами. Alogram надає утиліти для обробки запитів на зворотний дзвінок, вилучення відповідних даних і надання відповідної відповіді. Ця функція особливо корисна для створення інтерактивних та динамічних інтерфейсів ботів.

5. Керування розмовами: Alogram полегшує розробку багатокрокових діалогів або робочих процесів. Розробники можуть визначати обробники діалогів, щоб провести користувачів через низку взаємодій. Фреймворк піклується про управління станом діалогу, забезпечуючи безперебійне просування та обробку вхідних даних користувача на різних етапах.

6. Підтримка інлайн-режиму: Alogram підтримує вбудований режим, що дозволяє ботам відповідати на вбудовані запити безпосередньо з інтерфейсу чату. Розробники можуть реалізувати власну логіку, щоб генерувати динамічні результати на основі запитів користувачів. Ця функція підвищує інтерактивність і зручність використання бота.

7. Розширюваність і модульність: Alogram має високий рівень масштабованості, що дозволяє розробникам інтегрувати додаткову функціональність за допомогою плагінів і розширень. Фреймворк підтримує різні сторонні бібліотеки і може бути легко налаштований відповідно до конкретних вимог бота.

Загалом, Alogram дозволяє розробникам створювати складних Telegram-ботів за допомогою Python. Він поєднує в собі потужність Telegram Bot API зі зручним для розробників інтерфейсом, що робить його ідеальним вибором для створення інтерактивних, чуйних і багатофункціональних чат-ботів.

## **OpenAI**

Для обробки природної мови були використані результати досліджень OpenAI – відомої дослідницької лабораторії штучного інтелекту, яка розробила найсучасніші мовні моделі, в тому числі високотехнологічну модель, відому як GPT-3 (Generative Pre-trained Transformer 3 – генеративний попередньо навчений

трансформатор). Основна місія OpenAI – забезпечити, щоб штучний загальний інтелект (AGI) приносив користь усьому людству [11].

Модель GPT-3, розроблена OpenAI, – це найсучасніша мовна модель, яка чудово справляється з різними завданнями обробки природної мови, зокрема з генерацією тексту, перекладом, узагальненням тощо. Вона була навчена на великому масиві різноманітних текстових даних і може генерувати зв'язні та контекстно-релевантні відповіді за відповідними підказками [11].

OpenAI API використовується для використання можливостей GPT-3 для генерації відповідей ШІ в чат-боті. OpenAI Completion API спеціально використовується у функції `generate_answer` розробленого боту для генерації відповідей на основі підказок або запитів користувача.

Функція `generate_answer` взаємодіє з OpenAI Completion API, надсилаючи запит користувача на вхід і отримуючи згенеровану відповідь на виході. Запит слугує контекстом або інструкцією для моделі, щоб згенерувати релевантну та змістовну відповідь.

Використовуючи OpenAI API та GPT-3, чат-бот може використовувати розширені мовні можливості моделі для надання інтелектуальних та контекстно-залежних відповідей на вхідні дані користувача.

Важливо зазначити, що OpenAI API вимагає відповідної автентифікації та облікових даних доступу, таких як ключ API, для здійснення викликів API та отримання відповідей, згенерованих моделлю. Ці облікові дані повинні бути належним чином інтегровані в код, щоб встановити з'єднання з сервісами OpenAI і отримати бажані відповіді, згенеровані штучним інтелектом.

Отже, OpenAI – це впливова дослідницька лабораторія ШІ, яка розробила потужні мовні моделі, такі як GPT-3. Розроблений код використовує OpenAI API, зокрема OpenAI Completion API, для генерації відповідей ШІ на основі підказок або запитів користувача, використовуючи розширені мовні можливості GPT-3.

### **Стparse**

стparse – це бібліотека Python, спеціально розроблена для розбору виразів дати і часу на природній мові. Вона надає зручний та ефективний спосіб

вилучення структурованої інформації про дату та час з неструктурованих текстових даних.

У наведеному коді `ctparse` використовується для розбору інформації про нагадування, наданої користувачем у форматі природної мови. Бібліотека дозволяє чат-боту розуміти та витягувати конкретні дані про дату та час, згадані користувачем у вхідних даних (рис.2.5).



```

1 usage
2
3 async def extract_reminder_info(message: str):
4     prompt = f"Extract reminder details from the following text: '{message}'. Provide the result in the format " \
5             f"'YYYY-MM-DDTHH:MM, description of task'"
6     response = openai.Completion.create(
7         engine="text-davinci-002",
8         prompt=prompt,
9         max_tokens=100,
10        n=1,
11        stop=None,
12        temperature=0.5,
13    )
14
15    extracted_info = response.choices[0].text.strip()

```

Рис. 2.5. Введення та виведення функції `extract_reminder_info`, виділяючи витягнуту інформацію про дату та час

Джерело: розроблено авторкою за [9].

Функція `'extract_reminder_info'` використовує `ctparse` для аналізу введених користувачем даних і визначення відповідних компонентів дати і часу. Ця функція отримує вхідні дані користувача як параметр і використовує можливості синтаксичного аналізу `ctparse` для вилучення структурованих даних, таких як дата і час нагадування.

Використовуючи `ctparse`, чат-бот може ефективно інтерпретувати вирази природної мови, пов'язані з датами та часом, дозволяючи користувачам задавати нагадування більш гнучко та інтуїтивно зрозуміло.

Важливо зазначити, що `ctparse` може обробляти широкий спектр форматів дати і часу, включаючи відносні вирази, такі як «завтра» або «наступного тижня», а також абсолютні вирази, такі як «15 березня 2023 року» або «5:30 вечорам». Можливості синтаксичного аналізу бібліотеки дозволяють їй працювати з

різними мовами і підтримувати різні формати дати і часу, які зазвичай використовуються в різних регіонах.

Отже, `strparse` – це бібліотека Python, яка спеціалізується на розборі виразів дати і часу на природній мові. У розроблено коді вона використовується для вилучення структурованої інформації про дату та час із введених користувачем даних. Використовуючи `strparse`, чат-бот може точно розуміти та інтерпретувати дані про дату та час, згадані користувачем, покращуючи загальну функціональність та зручність користування чат-ботом.

### **Logging**

Модуль логування в Python надає гнучкий і стандартизований спосіб реєстрації інформаційних повідомлень під час виконання програми. Він пропонує набір функцій і класів, які дозволяють розробникам записувати різні події, помилки та іншу важливу інформацію для аналізу та налагодження.

У наданому коді модуль логування використовується для реєстрації важливих повідомлень і подій під час роботи чат-бота (Додаток А). Використовуючи модуль логування, розробники можуть легко збирати та зберігати відповідну інформацію про роботу бота, що дає змогу краще контролювати та усувати несправності.

Щоб використовувати модуль логування, його було імпортовано і налаштовано бажану на початку коду. В цьому проєкті модуль налаштовано з базовим рівнем журналювання `INFO`. Це означає, що повідомлення з рівнем серйозності `INFO` або вище будуть записані до журналу. Рівні серйозності, у порядку зростання важливості, такі: `DEBUG`, `INFO`, `WARNING`, `ERROR` і `CRITICAL`. Вибір рівня `INFO` для конфігурації журналу означає, що до журналу буде записано повідомлення з рівнем `INFO` або вище, включно з самим `INFO` [3].

Отже, розроблений чат бот передбачає використання `AIogram` для створення чат-бота, `OpenAI` для генерування відповідей ШІ, `strparse` для розбору інформації про нагадування та модуль логування для ведення журналу. Ці інструменти та технології дозволяють чат-боту взаємодіяти з користувачами, надавати інформацію про курс і виконувати різні завдання, керовані ШІ.



## 2.2. Архітектурний дизайн системи чат-ботів

Система чат-боту (STEMEdBot), описана в цьому архітектурному проекті, призначена для надання різних функціональних можливостей, включаючи роботу в якості ШІ-репетитора з письма, статистики, перекладача англійської мови та коректора. Крім того, система надає інформацію про курс і система автоматичних нагадувань про завдання. Архітектура побудована з використанням фреймворку AIOgram і використовує мовну модель OpenAI GPT-3.5 для генерації відповідей.

### Інтерфейс користувача

Через інтерфейс користувачі можуть ініціювати розмови з чат-ботом, надсилаючи повідомлення. Вони можуть ставити запитання, шукати допомоги в різних сферах і взаємодіяти з різними функціями системи чат-ботів. Інтерфейс дозволяє користувачам отримати доступ до інформації про курс, такої як навчальний план, матеріали лекцій та деталі завдань. Крім того, користувачі можуть встановлювати нагадування про завдання, щоб отримувати сповіщення та бути в курсі майбутніх дедлайнів.



Рис. 2.6. Приклад реалзації Reply-клавіатури у інтерфейсі користувача чат-боту STEMEdBot

Джерело: розроблено авторкою за [15].

Інтерфейс користувача розроблений з акцентом на простоту та інтуїтивність. Він має на меті забезпечити користувачам безперешкодний та ефективний спосіб взаємодії з системою чат-ботів. Завдяки використанню

AIogram та OpenAI інтерфейс забезпечує обробку та розуміння природної мови, що дозволяє чат-боту точно інтерпретувати запити користувачів та генерувати релевантні відповіді.

Загалом, користувацький інтерфейс відіграє вирішальну роль у забезпеченні ефективної комунікації між користувачами та системою чат-ботів. Він слугує шлюзом, через який користувачі можуть отримати доступ до функціональних можливостей системи та отримати необхідну допомогу.

### **Модуль Chat GPT**

Модуль Chat GPT є ключовим компонентом системи, що використовує можливості мовної моделі OpenAI GPT-3.5. Він слугує основою для генерації відповідей на основі запитів і підказок користувача в різних галузях. Модуль може обробляти запити, пов'язані з навчанням написання текстів, статистичним аналізом, перекладом і вдосконаленням англійської мови, а також коректурою.

Коли користувач надсилає введення або запит через інтерфейс користувача, модуль Chat GPT отримує і обробляє його. На основі запитуваної області модуль створює відповідні підказки, які інкапсулюють наміри користувача або його запитання. Ці підказки ретельно розроблені, щоб забезпечити необхідний контекст і підказки для мовної моделі.

Після того, як відповіді згенеровані, вони повертаються до інтерфейсу користувача, де відображаються у вигляді простого тексту. Система чат-ботів гарантує, що відповіді будуть представлені у зрозумілій та зручній для користувача формі, що сприяє ефективній комунікації між користувачем та чат-ботом.

Завдяки використанню модуля Chat GPT і розширеним можливостям обробки мови моделі GPT-3.5, система чат-боту може надавати користувачам високоякісні відповіді та цінні поради в різних сферах. Незалежно від того, чи це надання пропозицій щодо написання тексту, проведення статистичного аналізу, покращення перекладів з англійської мови або вичитування текстів, модуль Chat GPT робить свій внесок у здатність системи надавати точну та корисну інформацію.

### **Модуль «Remind me»**

Модуль «Remind me» – це частина системи, яка дозволяє користувачам встановлювати персональні нагадування (наприклад, про необхідність виконати специфічні завдання). Цей модуль додає чат-боту корисну функцію, що дозволяє користувачам залишатися організованими та виконувати свої навчальні обов'язки вчасно.

Цей модуль використовує методи обробки природної мови, щоб витягти відповідні дані з введеного користувачем тексту. Він аналізує текст, щоб визначити важливу інформацію, таку як назва завдання, термін виконання та будь-які додаткові характеристики, надані користувачем.

Після того, як дані про нагадування витягнуті, модуль зберігає їх у списку нагадувань, пов'язуючи нагадування з відповідною датою та часом. Це дозволяє чат-боту відстежувати всі майбутні завдання та їхні дедлайни.

Крім того, модуль постійно перевіряє наявність майбутніх нагадувань на основі збереженої інформації про дату і час. Коли наближається запланований час нагадування, модуль запускає сповіщення, яке надсилається користувачеві. Сповіщення слугує підказкою, щоб нагадати користувачеві про наближене завдання і переконатися, що він не забуває про свої академічні зобов'язання.

Забезпечуючи функцію нагадування про завдання, система чат-ботів допомагає користувачам ефективно управляти своїм робочим навантаженням і уникати пропущених дедлайнів. Здатність модуля витягувати деталі нагадування за допомогою методів обробки природної мови та надсилати своєчасні сповіщення сприяє покращенню користувацького досвіду та підвищенню продуктивності й організованості.

### **Модуль course\_info**

Це другий за значенням компонент системи чат-боту, призначений для надання користувачам вичерпної інформації про різні курси. Цей модуль є цінним ресурсом для студентів, які шукають детальну інформацію про навчальні плани, розклади, ресурси, політику відвідування та критерії оцінювання для кожного курсу.

Коли користувач запитує інформацію про курс, модуль використовує свій доступ до списку курсів, який містить повну базу даних, пов'язаних з курсами. Модуль знаходить відповідну інформацію на основі запиту користувача і представляє її у зручному для користувача форматі.

Наприклад, якщо користувач запитує навчальний план певного курсу, модуль шукає у списку курсів відповідний навчальний план і знаходить його. Аналогічно, коли користувач шукає інформацію про розклад курсу, модуль витягує дані розкладу зі списку курсів і представляє їх користувачеві.

На додаток до навчальних планів і розкладів, модуль також надає доступ до інших важливих ресурсів. Користувачі можуть дізнатися про політику відвідування, що дозволяє їм зрозуміти вимоги та очікування, пов'язані з відвідуванням кожного курсу. Крім того, користувачі можуть отримати детальну інформацію про критерії оцінювання, що допоможе їм зрозуміти, як буде оцінюватися їхня робота протягом курсу.

Таким чином, цей компонент системи спрощує процес отримання життєво важливих для студентів даних. Він усуває необхідність шукати інформацію в різних джерелах або звертатися до викладачів індивідуально, забезпечуючи зручний та ефективний спосіб доступу до потрібної інформації.

Отже, архітектура системи STEMEdBot складається з різних модулів, які обробляють взаємодію з користувачами, генерують відповіді за допомогою мовної моделі OpenAI GPT-3.5, надають інформацію про курс та керують нагадуваннями про завдання. Архітектура системи забезпечує ефективну комунікацію між модулями, що дозволяє користувачам отримувати релевантну та своєчасну інформацію та допомогу.

### **2.3. Досвід користувачів у розробленій системі чат-боту**

Як вже було зазначення, бот призначений для виконання різних функцій, включаючи роботу в якості ШІ-репетитора з письма, статиста, перекладача англійської мови, коректора, а також для надання інформації про різні курси та встановлення нагадувань про завдання.

Код ініціалізує необхідні компоненти, такі як бот, диспетчер і проміжне програмне забезпечення для ведення журналів. Він також визначає кілька класів і налаштовує обробники команд і повідомлень для обробки взаємодії з користувачем.

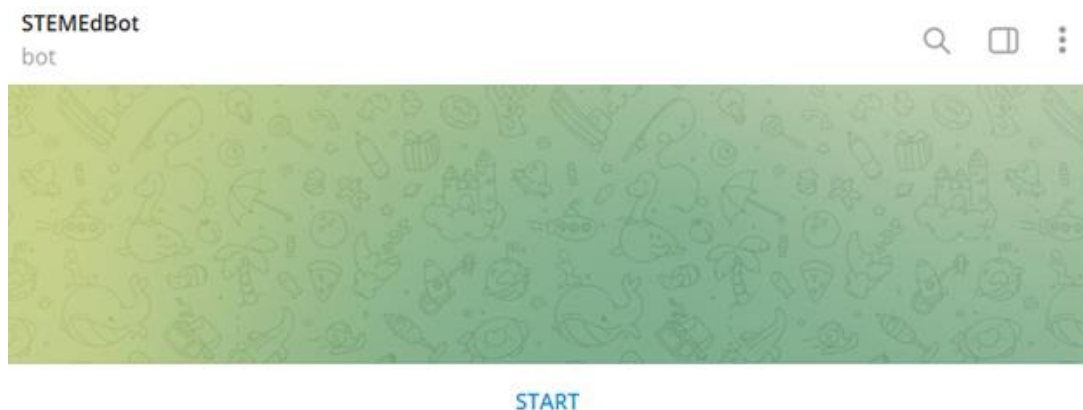


Рис. 2.7. Стартовий екран STEDEmBot (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

Після старту бота, код пропонує зручний спосіб отримати доступ до довідки та поширених запитань (FAQ) про функціональність чат-бота. Коли користувач надсилає команду `"/help"`, чат-бот відповідає детальним FAQ-повідомленням.

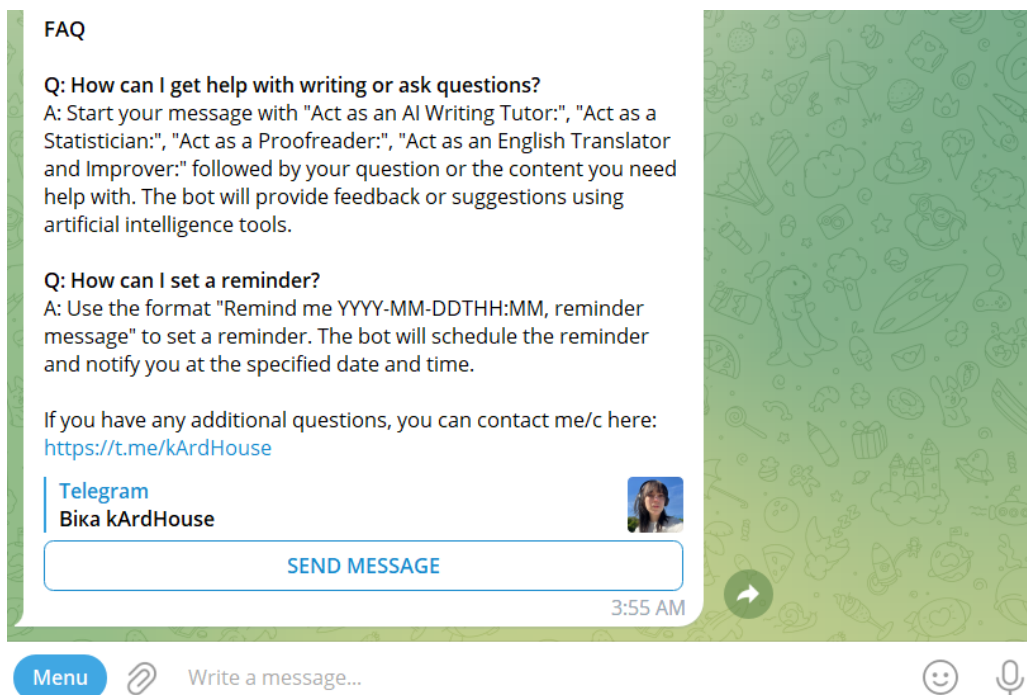


Рис. 2.8. Результат використання команди `/help` у STEDEmBot (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

Повідомлення FAQ охоплює поширені запитання та надає відповіді, щоб допомогти користувачам ефективно використовувати чат-бота. У ньому розглядаються два конкретні запити:

1. Як отримати допомогу в написанні або поставити запитання: у FAQ пояснюється, що користувачі можуть почати своє повідомлення з конкретних команд. Після цього користувачі можуть поставити своє запитання або вказати, з яким контентом їм потрібна допомога. Чат-бот, використовуючи інструменти штучного інтелекту, згенерує відгук або пропозиції, адаптовані до запиту користувача.

2. Як налаштувати нагадування: ця частина пояснює формат налаштування нагадування. Дотримуючись цього формату, користувачі можуть запланувати нагадування за допомогою чат-бота. Чат-бот збереже нагадування і повідомить користувача в зазначений день і час.

Крім того, повідомлення з поширеними запитаннями інформує користувачів про те, що вони можуть зв'язатися з каналом підтримки, перейшовши за наданим посиланням. Це дозволяє користувачам звертатися за подальшими питаннями або допомогою, пов'язаними з чат-ботом.

Таким чином, команда `/help` надає стислу та інформативну відповідь, пропонуючи вказівки щодо взаємодії з чат-ботом та доступу до додаткової підтримки, якщо це необхідно.

### **Модуль `course_info`**

З точки зору користувача, модуль `course_info` представляє функціонал чат-бота, який обробляє повідомлення, пов'язані з курсами. Код складається з двох обробників повідомлень, які реагують на певні типи повідомлень.

Перший обробник повідомлень запускається, коли користувач надсилає повідомлення, яке відповідає назві курсу у списку курсів. Він створює клавіатуру-відповідь з кнопками, які надають різні опції, пов'язані з обраним курсом. Клавіатура містить кнопки для доступу до навчальної програми, розкладу, ресурсів, відвідуваності та оцінок за курс. Повідомлення, надіслане

чат-ботом, містить назву курсу, а клавіатура для відповіді додається на вибір користувача.

Другий обробник повідомлення запускається, коли користувач вибирає одну з опцій з клавіатури на попередньому кроці. Він визначає конкретну дію (навчальний план, розклад, ресурси, відвідування або оцінки) і відповідну назву курсу. Він отримує відповідну інформацію для вибраної дії зі списку курсів і відповідно генерує повідомлення-відповідь. Відповідь містить запитувану інформацію для конкретного курсу. Після надсилання відповіді клавіатура зникає (рис. 2.9).

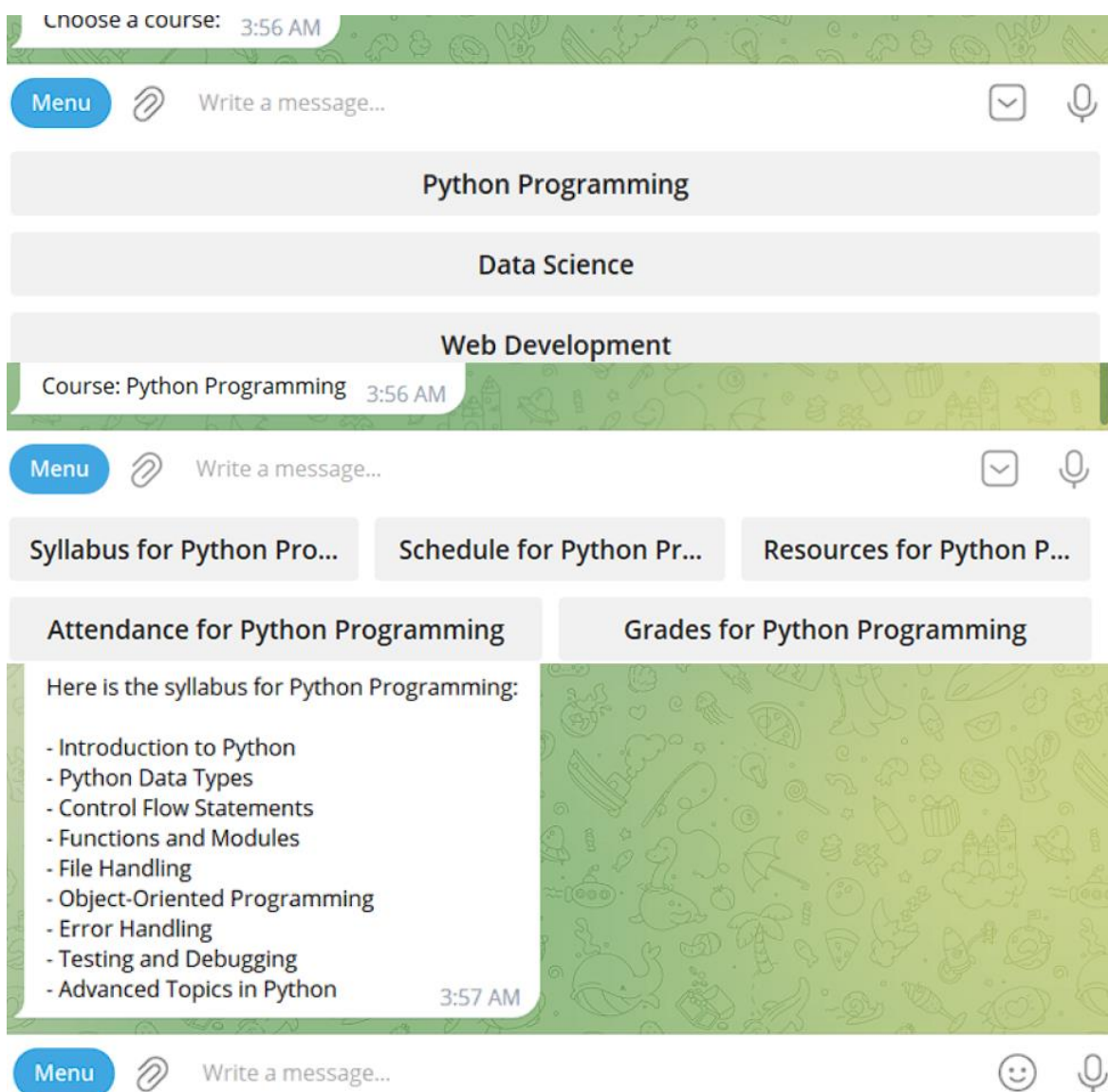


Рис. 2.9. Модуль course\_info у STEDEmBot (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

Загалом, цей код дозволяє чат-боту обробляти повідомлення, пов'язані з курсами, надавати варіанти доступу до інформації про курс і відповідати на запити відповідно до вибору користувача.

### Модуль Remind me

Коли користувач надсилає повідомлення, що починається з «Remind me,», код витягує деталі нагадування з повідомлення, використовуючи модель OpenAI GPT для завершення тексту. Витягнута інформація має бути у форматі «YYYY-MM-DDTHH:MM, опис завдання» (рис.2.10).

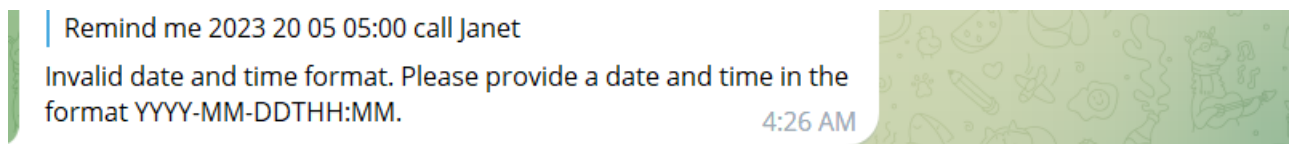


Рис. 2.10. Модуль remind me у STEDEmBot (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

Якщо витягнута інформація про нагадування не містить коми (,), код намагається розібрати введення природною мовою, використовуючи бібліотеку розбору дати і часу. Якщо розбір не дає результату, вказуючи на неправильний формат, чат-бот видає повідомлення про помилку і пропонує користувачеві ввести дані нагадування в правильному форматі.

Якщо витягнута інформація про нагадування містить кому, код розділяє її на частину з датою та часом і частину з повідомленням про нагадування. Потім він перевіряє формат дати і часу та перевіряє, чи нагадування налаштоване на майбутнє. Якщо формат або вказані дата і час невірні, чат-бот видає повідомлення про помилку.

Після успішної перевірки чат-бот підтверджує нагадування, відповідаючи датою і часом нагадування, а також текстом нагадування.



Код додає дані нагадування до списку нагадувань і обчислює різницю в часі між датою нагадування та поточним часом. Якщо різниця в часі позитивна (що вказує на майбутнє нагадування), код асинхронно чекає протягом цього часу, перш ніж надіслати користувачеві повідомлення з нагадуванням.

Нарешті, чат-бот надсилає нагадування користувачеві у вказаний час нагадування.

З точки зору користувача, цей код дозволяє чат-боту витягувати деталі нагадування, підтверджувати та встановлювати нагадування, а також надсилати нагадування користувачеві у вказаний час. Це забезпечує зручний спосіб для користувачів планувати та керувати нагадуваннями в інтерфейсі чат-бота.

### Модуль анонімного чату

Коли користувач надсилає команду «/join», код перевіряє, чи користувач уже перебуває в чаті. Якщо користувач вже з'єднаний з партнером, чат-бот відповідає повідомленням про те, що користувач вже перебуває в чаті. В іншому випадку, якщо в чаті немає інших користувачів, користувач додається в чат і отримує повідомлення про те, що він чекає на партнера(рис.2.11).

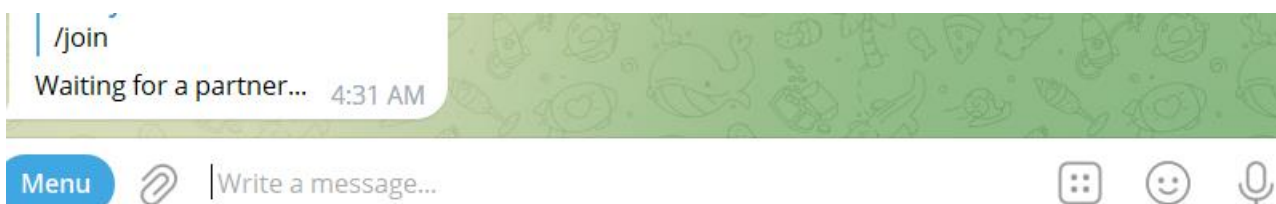


Рис. 2.11. Модуль /join у STEDEmBot (desktop-версія Telegram)

Джерело: розроблено авторкою за [15].

Якщо в чаті вже є інші користувачі, код призначає користувачеві партнера і відповідним чином оновлює інформацію про користувачів. І користувач, і його партнер отримують повідомлення про те, що вони тепер з'єднані один з одним.

Для повідомлень, які не є командами (повідомлення, які не починаються з '/'), код визначає, чи з'єднаний користувач з партнером. Якщо користувач не з'єднаний, чат-бот відповідає повідомленням про те, що користувач не з'єднаний

з партнером, і пропонує йому скористатися командою «/join», щоб знайти партнера. В іншому випадку повідомлення переадресовується партнеру користувача, що дозволяє їм поспілкуватися.

Якщо користувач надсилає команду «/leave», код перевіряє, чи з'єднаний він з партнером. Якщо користувач не підключений, чат-бот відповідає повідомленням про те, що користувач не перебуває в чаті. В іншому випадку код видаляє користувача та його партнера з чату, і обидві сторони отримують повідомлення про те, що користувач вийшов з чату.

Таким чином, цей модуль дозволяє йому приєднатися до чату, спілкуватися з партнером і вийти з чату за бажанням. Він забезпечує простий та інтерактивний спосіб для користувачів підключатися та брати участь у розмовах у середовищі чат-ботів

### **Модуль Chat GPT**

З точки зору користувача, цей фрагмент коду представляє різні ролі, які чат-бот може взяти на себе, щоб допомогти користувачеві у виконанні різних завдань. Кожна роль фокусується на певній сфері та надає користувачеві підтримку і зворотний зв'язок.

Коли користувач надсилає повідомлення, що починається як «Act as an AI Writing Tutor:» чат-бот бере на себе роль репетитора з письма. Запитання користувача витягується з повідомлення, і чат-бот генерує відповідь, використовуючи інструменти штучного інтелекту, обробку природної мови та свої знання про ефективні техніки письма. Згенерована відповідь надсилається назад користувачеві як зворотний зв'язок про те, як покращити своє письмо.

Якщо повідомлення користувача починається зі слів «Act as an AI Writing Statistician:», чат-бот бере на себе роль статиста. Запитання користувача витягується, і чат-бот генерує відповідь на основі своїх знань статистичної термінології, статистичних розподілів, довірчих інтервалів, ймовірності, перевірки гіпотез і статистичних графіків. Згенерована відповідь містить інформацію та пояснення, пов'язані з питанням користувача.

Коли користувач надсилає повідомлення, що починається з «Act as an English Translator and Improver:», чат-бот виконує функції перекладача, орфографічного коректора та покращувача англійської мови. Чат-бот визначає мову тексту користувача, перекладає його на англійську та надає відповідь з виправленою та покращеною версією тексту. Чат-бот замінює спрощені слова та речення рівня A0 на більш елегантні та вдосконалені англійські вирази, зберігаючи при цьому оригінальний зміст.

Якщо повідомлення користувача починається зі слів «Act as a Proofreader:», чат-бот бере на себе роль коректора. Чат-бот перевіряє наданий текст на наявність орфографічних, граматичних і пунктуаційних помилок. Після перевірки чат-бот пропонує необхідні виправлення або пропозиції щодо покращення тексту.

У кожному випадку користувач отримує відповідь від чат-бота залежно від конкретної ролі, яку він бере на себе. Чат-бот використовує модель OpenAI GPT для формування відповіді, надаючи допомогу та підказки, адаптовані до потреб користувача у відповідній галузі знань.

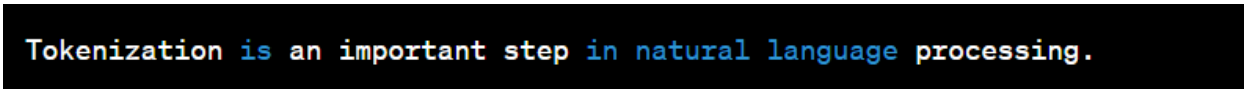
#### **2.4. Методи обробки природної мови, що використовуються для взаємодії з користувачем**

Методи обробки природної мови, що використовуються для взаємодії з користувачем у нашому боті, передбачають поєднання декількох методологій та алгоритмів, які дозволяють системі розуміти та ефективно обробляти вхідні дані користувача. Ці методи допомагають витягувати релевантну інформацію, визначати наміри користувача та генерувати відповідні відповіді. Пропонуємо ознайомитися з усіма подробицями:

Токенізація – це важливий крок в обробці природної мови (NLP), який передбачає розбиття тексту на менші одиниці, відомі як токени. Ці токени можуть представляти різні елементи, такі як слова, розділові знаки, числа або інші значущі одиниці, залежно від конкретного контексту. Токенізація відіграє

життєво важливу роль в аналізі та обробці тексту на більш детальному рівні, полегшуючи подальші кроки в конвеєрі NLP [18].

У сфері NLP для токенизації можна використовувати різні бібліотеки та інструменти. Однією з таких бібліотек є `ctparse` – бібліотека Python, спеціально розроблена для синтаксичного аналізу та маніпулювання датами і часом у природній мові. Хоча `ctparse` в першу чергу зосереджена на розборі дати і часу, вона включає в себе токенизацію як невід'ємний компонент своєї функціональності.

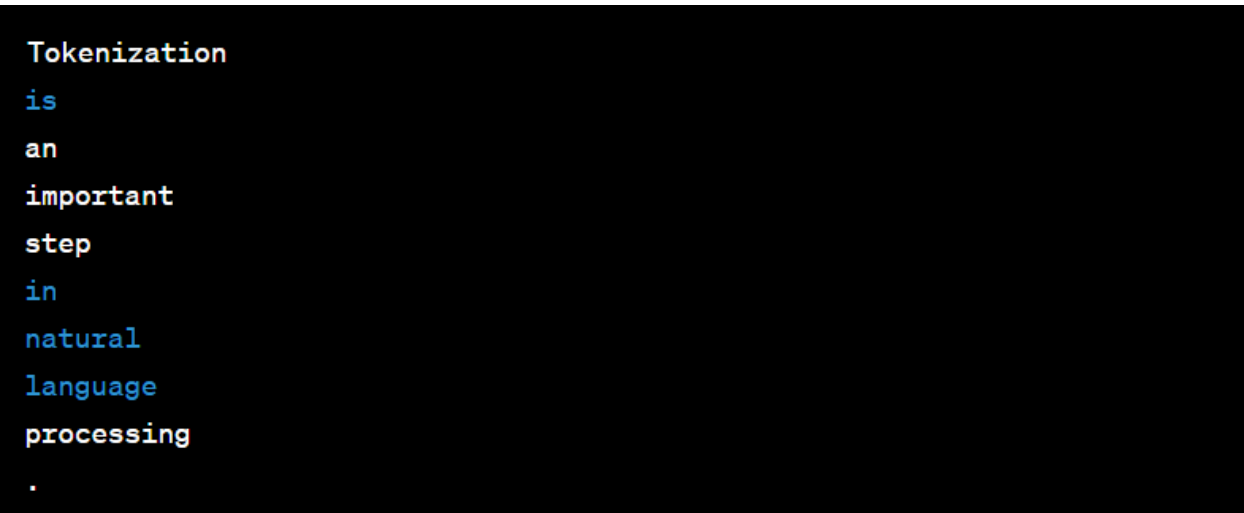


```
Tokenization is an important step in natural language processing.
```

Рис. 2.12. Вхідні дані для токенизації

Джерело: розроблено авторкою за [18].

Функція автоматично токенизує вхідний текст і повертає список токенів, які представляють окремі слова і розділові знаки. Ці токени можуть бути надалі оброблені або проаналізовані відповідно до вимог завдання NLP.



```
Tokenization  
is  
an  
important  
step  
in  
natural  
language  
processing  
.
```

Рис. 2.13. Вихідні дані токенизації

Джерело: розроблено авторкою за [18].

Важливо відзначити, що хоча `ctparse` в першу чергу зосереджена на розборі дати і часу, її можна використовувати для базових цілей токенизації, як показано в прикладі коду вище. Однак, для більш складних завдань токенизації або

сценаріїв, що вимагають спеціалізованих методів токенізації, інші бібліотеки, такі як NLTK, SpaCy або Transformers, можуть бути кращими варіантами [21].

2. POS-тегування – це процес обробки природної мови, який передбачає присвоєння граматичної мітки кожній лексемі (слову або символу) в тексті. Граматична мітка представляє частину мови токена, наприклад, іменник, дієслово, прикметник тощо. POS-мітки допомагають зрозуміти синтаксичну структуру речення і дозволяють системі витягувати важливу інформацію на основі граматичної ролі слів у контексті введення користувачем.

```
import nltk

sentence = "I love to eat pizza."

# Tokenize the sentence into individual words
tokens = nltk.word_tokenize(sentence)

# Perform POS tagging
pos_tags = nltk.pos_tag(tokens)

# Print the POS tags
for word, tag in pos_tags:
    print(f"{word}: {tag}")
```

Рис. 2.14. Введення POS-мітки за допомогою бібліотеки NLTK

Джерело: розроблено авторкою за [13].

POS-тегування відіграє вирішальну роль у різних завданнях бота, таких як вилучення інформації, аналіз настроїв, машинний переклад і відповіді на запитання. Присвоюючи словам відповідні POS-теги, система може розрізняти значення слів і вирішувати будь-які двозначності, що виникають через омоніми або слова з багатозначним значенням. Таке розмежування допомагає в точному розумінні та інтерпретації тексту.

Крім того, POS-тегування допомагає на наступних етапах обробки природної мови, таких як розпізнавання іменованих об'єктів та синтаксичний аналіз.

```
I: PRP
love: VBP
to: TO
eat: VB
pizza: NN
```

Рис. 2.15. Виведення POS-мітки

Джерело: розроблено авторкою за [13].

У прикладах на рис. 2.14 та 2.15 бібліотека NLTK використовується для токенизації та POS-тегування. Функція `nltk.word_tokenize()` використовується для розбиття вхідного речення на окремі слова, а `nltk.pos_tag()` застосовує POS-теги до токенизованих слів. Нарешті, POS-теги виводяться для кожного слова у реченні.

POS-теги надають цінну інформацію про синтаксичні зв'язки між словами в реченні, яку можна використовувати для ідентифікації іменованих об'єктів, визначення структури підмета-дієслова-об'єкта та аналізу граматичних залежностей у реченні.

Загалом, POS-тегування є фундаментальним завданням в обробці природної мови, яке покращує розуміння тексту шляхом присвоєння граматичних міток словам. Це полегшує вилучення значущої інформації та підтримує різні подальші завдання NLP, надаючи уявлення про синтаксичну структуру та граматичну роль слів у реченні.

3. Розпізнавання іменованих об'єктів (NER) – це метод, який використовується для ідентифікації та класифікації іменованих об'єктів у тексті. Він допомагає витягувати конкретну інформацію та розуміти контекст згаданих об'єктів. У нашому коді немає явної реалізації NER. Однак можливо навести приклад того, як NER можна застосувати в контексті вилучення інформації про курс з даних, введених користувачем.

Припустимо, користувач надсилає повідомлення на кшталт «Скажіть мені розклад курсу «Програмування на Python»». Тут іменованою сутністю, яку потрібно розпізнати, є назва курсу, а саме «Програмування на Python». Ми можемо витягти цю іменовану сутність, порівнявши вхідні дані користувача з назвами курсів, що зберігаються у списку курсів. Ось приклад реалізації для вилучення назви курсу:

```
@dp.message_handler(lambda message: message.text in [course["name"] for course in courses])
async def handle_course_buttons(message: types.Message):
    course_index = [course["name"] for course in courses].index(message.text)
    course = courses[course_index]
    # Rest of the code
```

Рис. 2.16. Витягування назви курсу за допомогою NER

Джерело: розроблено авторкою за [12].

У цьому фрагменті коду функція `handle_course_buttons` обробляє введення користувача, якщо воно збігається з будь-якою назвою курсу у списку курсів. Вона витягує назву курсу як іменовану сутність і виконує подальші дії на основі цієї сутності. Це простий приклад того, як можна використовувати NER для вилучення іменованих сутностей з користувацького вводу.

Однак важливо зазначити, що наданий код не містить повної реалізації NER. NER зазвичай включає в себе більш просунуті методи, такі як навчання моделі машинного навчання на мічених даних для точного розпізнавання і класифікації іменованих об'єктів. Наведений приклад демонструє базову концепцію вилучення іменованих об'єктів за допомогою простого зіставлення рядків.

4. Синтаксичний аналіз – це метод обробки природної мови, який передбачає аналіз граматичних зв'язків між словами в реченні та створення структури дерева залежностей. Дерево залежностей відображає синтаксичну структуру і значення речення шляхом виявлення залежностей або зв'язків між словами. Ця техніка корисна для розуміння структури речень, зокрема зв'язків між підметом і дієсловом, іменними словосполученнями, модифікаторами тощо.

У розробленому Telegram-боті модуль синтаксичного аналізу залежностей використовується всередині чат-бота для обробки та розуміння запитів користувачів. Припустимо, користувач вводить наступне повідомлення: «Remind me to complete the task tomorrow ».

Чат-бот отримує повідомлення користувача та обробляє його за допомогою модуля синтаксичного аналізу залежностей. Модуль розбору залежностей аналізує речення та визначає зв'язки між словами. Модуль розбору залежностей створює дерево залежностей, яке відображає структуру речення та зв'язки між словами. Дерево залежностей може показувати, що complete – це головне дієслово, task – об'єкт дієслова, а tomorrow - модифікатор, що вказує на час. Тоді чат-бот може витягти відповідну інформацію з дерева залежностей, наприклад, завдання і час нагадування.

Виконуючи синтаксичний аналіз залежностей, чат-бот може точно інтерпретувати запити користувача і витягувати необхідну інформацію для виконання таких завдань, як налаштування нагадувань.

5. Методи вилучення інформації відіграють вирішальну роль у вилученні структурованої інформації з неструктурованих текстових даних. У контексті наданого коду, вилучення інформації використовується для визначення конкретних точок даних, таких як дати, час, числа та інша відповідна інформація, згадана у введеній користувачем інформації. Цей процес допомагає обробляти нагадування про завдання, отримувати інформацію про курс і розуміти запити користувачів, пов'язані з конкретними точками даних або параметрами.

```

async def handle_course_info_buttons(message: types.Message):
    action, course_name = message.text.split(" for ")
    course_index = [course["name"] for course in courses].index(course_name)
    course = courses[course_index]

    if action == "Syllabus":
        response = f"Here is the syllabus for {course['name']}: \n{course['syllabus']}"
    elif action == "Schedule":
        response = f"Here is the schedule for {course['name']}: \n{course['schedule']}"
    elif action == "Resources":
        response = f"Here are the resources for {course['name']}: \n{course['resources']}"
    elif action == "Attendance":
        response = f"Attendance: {course['name']}: \n{course['attendance']}"
    else: # action == "Grades"
        response = f"Grades: {course['name']}: \n{course['grades']}"

```

Рис. 2.17. Витяг інформації про курс



Джерело: розроблено авторкою за [15].

Фрагмент коду на рис. 2.17 демонструє реалізацію вилучення інформації в додатку чат-бота. Наведемо кілька прикладів використання вилучення інформації:

Користувачі можуть запитувати інформацію про різні курси, обравши курс з доступних варіантів. Чат-бот витягує назву курсу і надає клавіатуру для відповіді з різними опціями, такими як навчальний план, розклад, ресурси, відвідування та оцінки. Залежно від вибору користувача, чат-бот витягує відповідну інформацію з даних про курс і надає відповідну відповідь.

```
if ',' not in reminder_info:
    # Try to parse the natural language input
    parser = ctparse.CTParser()
    parsed = parser.parse(reminder_info)
    if not parsed:
        await message.reply(
            "Invalid reminder format. Please provide a date and time in the format YYYY-MM-DDTHH:MM, "
            "followed by a comma and the reminder message, or use natural language input.")
        return
    reminder_datetime = parsed[0].start
    reminder_message = "Write to you"
else:
    date_time, reminder_message = reminder_info.split(',', 1)
    date_time = date_time.strip()
    reminder_message = reminder_message.strip()
```

Рис. 2.18. Модуль нагадування

Джерело: розроблено авторкою за [15].

Користувачі можуть встановлювати нагадування, надаючи певний формат або використовуючи природну мову. Чат-бот використовує методи вилучення інформації, щоб витягти деталі нагадування з повідомлення користувача.

Якщо введені дані відповідають зазначеному формату (ГГГГ-ММ-ДДТТГ:ММ, нагадування), чат-бот витягує дату, час і нагадування відповідно.

Якщо користувач вводить текст природною мовою, чат-бот використовує синтаксичний аналізатор природної мови, щоб витягти необхідну інформацію.

Витягнуті дані потім використовуються для планування нагадування та сповіщення користувача у вказану дату та час.

Отже, вилучення інформації є основним компонентом цього чат-бота, що дозволяє отримувати релевантні дані з введених користувачем даних і сприяє ефективному спілкуванню та взаємодії. Використовуючи методи вилучення інформації, чат-бот покращує свою здатність обробляти нагадування про завдання, збирати інформацію про курс, а також точно та ефективно відповідати на запити користувачів. Це дозволяє чат-боту надавати користувачам точну та змістовну інформацію, покращуючи загальний користувацький досвід та забезпечуючи ефективну комунікацію між ботом та користувачами.

## **Висновки до розділу 2**

1. Для розробки чат-бота було використано такі інструменти та технології: AIOgram для створення чат-бота на платформі Telegram, OpenAI для генерації відповідей з використанням штучного інтелекту, strparse для розпізнавання та аналізу інформації про нагадування, а також модуль логування для збереження журналу взаємодії з користувачами. Завдяки цим інструментам та технологіям чат-бот здатний взаємодіяти з користувачами, надавати інформацію про курси та виконувати різноманітні завдання, керовані штучним інтелектом.

2. Архітектура системи STEMEdBot складається з різних модулів, що взаємодіють між собою для забезпечення функціональності чат-бота. Модулі включають обробку вхідних повідомлень від користувачів, генерацію відповідей з використанням мовної моделі OpenAI GPT-3.5, надання інформації про курси та керування нагадуваннями про завдання. Ця архітектура системи забезпечує ефективну комунікацію та співпрацю між модулями, що дозволяє користувачам отримувати релевантну та своєчасну інформацію та допомогу, відповідно до їх потреб.

3. Універсальність STEMEdBot дозволяє йому задовольняти різноманітні потреби та надавати користувачам цілий ряд цінних послуг. Незалежно від того, чи це допомога з навичками письма, статистичний аналіз, переклад тексту,

коректура або надання інформації про курси та нагадування про завдання, чат-бот слугує комплексним і багатофункціональним інструментом для користувачів в академічній сфері.

4. Отже, вилучення інформації є основним компонентом цього чат-бота, що дозволяє отримувати релевантні дані з введених користувачем даних і сприяє ефективному спілкуванню та взаємодії. Використовуючи методи вилучення інформації, чат-бот покращує свою здатність обробляти нагадування про завдання, збирати інформацію про курс, а також точно та ефективно відповідати на запити користувачів. Це дозволяє чат-боту надавати користувачам точну та змістовну інформацію, покращуючи загальний користувацький досвід та забезпечуючи ефективну комунікацію між ботом та користувачами.

## **РОЗДІЛ 3**

### **ОБМЕЖЕННЯ ТА РЕКОМЕНДАЦІЇ ЩОДО МОДЕРНІЗАЦІЇ РОЗРОБЛЕНОГО ЧАТБОТУ**

#### **3.1. Обмеження розробленого чатботу**

У цьому розділі представлено комплексний аналіз обмежень розробленого нами чат-бота. Цей аналіз охоплює як технічні, так і концептуальні проблеми, з якими ми зіткнулися в процесі розробки. Ми розглядаємо питання, пов'язані з розумінням мови, розумінням контексту, генеруванням відповідей та масштабованістю системи.

Однак, аналіз ми почнемо з обмеження безпосередньо не пов'язаного з NLP, а саме – відсутністю повного циклу обробки помилок є значною проблемою та обмеженням цього боту. Обробка помилок – це процес передбачення та управління непередбачуваними ситуаціями або помилками, які можуть виникнути під час виконання програми. Коли коду не вистачає комплексної системи обробки помилок та механізмів перехоплення винятків, він стає вразливим до несподіваних збоїв або невизначеної поведінки, що призводить до низки негативних наслідків.

Однією з основних причин, чому обробка помилок має вирішальне значення, є те, що вона забезпечує кращий користувацький досвід. Коли виникає помилка і не обробляється належним чином, код може раптово завершити роботу. Це може розчарувати користувачів, які можуть втратити незбережені дані або зіткнутися з несподіваними збоями в робочому процесі. Збій також може залишити у користувачів негативне враження про програмне забезпечення, вплинути на його репутацію і потенційно призвести до втрати користувачів або клієнтів.

На додаток до поганого користувацького досвіду, відсутність належної обробки помилок на даному етапі розробки боту вагомо ускладнює діагностику та виправлення проблем. Коли виникають помилки, їх потрібно належним чином

реєструвати або повідомляти про них, щоб розробники могли зрозуміти причину і вжити відповідних заходів. Без ефективної обробки помилок помилки можуть залишатися непоміченими або бути проігнорованими, що ускладнює виявлення першопричини проблем. Це може призвести до тривалих циклів налагодження і затримок у вирішенні проблем, впливаючи на загальний процес розробки і викликаючи розчарування як у розробників, так і у користувачів.

Належна обробка помилок також відіграє важливу роль у підтримці цілісності та безпеки даних. Без належної системи обробки помилок винятки або непередбачувані ситуації можуть призвести до пошкодження або втрати даних. Крім того, обробка помилок необхідна для перевірки даних, введених користувачем, і запобігання вразливостям безпеки, таким як ін'єкційні атаки або несанкціонований доступ до конфіденційної інформації. Без надійної обробки помилок ці ризики можуть залишитися невиявленими, що поставить під загрозу систему та її дані.

Щоб вирішити проблему недостатньої обробки помилок, необхідно впроваджувати комплексні стратегії обробки помилок. Це передбачає визначення потенційних сценаріїв помилок, таких як помилки файлового вводу/виводу, проблеми з мережевим підключенням, невірне введення даних користувачем або обмеження ресурсів, а також впровадження відповідного коду обробки помилок або механізмів обробки винятків. Коли виникає помилка, вона повинна бути зареєстрована або повідомлена з достатньою інформацією, щоб допомогти діагностувати проблему. Повідомлення про помилки повинні бути чіткими, стислими та інформативними, щоб допомогти розробникам зрозуміти проблему та вжити відповідних заходів для її усунення.

Впроваджуючи ефективні методи обробки помилок, ми зможемо підвищити стабільність і надійність свого коду. Належна обробка помилок мінімізує неочікувані збої, забезпечує кращий користувацький досвід, покращує ремонтпридатність коду та дозволяє швидше виявляти і вирішувати проблеми. Це невід'ємна частина розробки програмного забезпечення, яка допомагає забезпечити безперебійну та безпомилкову роботу додатків.

Ще одним обмеженням розробленого коду є обмежена модульність – це ситуація, коли код написаний як єдиний скрипт без модуляризації різних функціональних можливостей в окремі файли або класи. Модульність – це фундаментальний принцип розробки програмного забезпечення, який підкреслює поділ кодової бази на менші, самодостатні модулі. Кожен модуль фокусується на певній функціональності або можливості, що робить код більш організованим, зручним для обслуговування та розширюваним у довгостроковій перспективі.

Розробленому коду не вистачає модульності, а це означає, що вся логіка і функціональність реалізована в одному скрипті. Такий підхід може призвести до кількох проблем і недоліків. По-перше, у кодовій базі стає складно орієнтуватися. Коли весь код знаходиться в одному місці, стає важче ідентифікувати певні частини коду або знайти відповідну логіку. Така неорганізованість призводить до того, що витрачається багато часу на внесення цільових змін або вдосконалень, оскільки розробникам доводиться просіювати велику кодову базу, щоб знайти потрібні фрагменти коду.

Крім того, без модульності можливість повторного використання коду стає обмеженою. Модульний код дозволяє повторно використовувати окремі компоненти або модулі в різних частинах програми або навіть в інших проектах. Однак, коли код не є модульованим, розробники можуть дублювати код у різних частинах програми, що призводить до надмірності коду та збільшення зусиль з його обслуговування. Якщо в одній частині коду виявлено помилку або вдосконалення, його потрібно вручну реплікувати та оновити у всіх дубльованих частинах, що може призвести до помилок і є неефективним методом редагування.

Крім того, підтримувати та розширювати код, якому бракує модульності, з часом стає дедалі складніше. Зі збільшенням розміру та складності кодової бази вносити зміни або додавати нові функції стає дедалі складніше. Зміна в одній частині коду може мати непередбачувані наслідки в інших частинах, що ускладнює забезпечення стабільності та коректності роботи системи. Відсутність чітких меж модулів та інкапсуляції може призвести до тісного зв'язку між

різними частинами коду, що ускладнює модифікацію або заміну окремих компонентів без впливу на всю систему.

Щоб вирішити проблему обмеженої модульності, ми повинні прагнути до рефакторингу кодової бази та запровадити належну модуляризацію. Це передбачає розбиття коду на менші зв'язні модулі, кожен з яких відповідає за певну функціональність. Ці модулі можуть бути організовані в окремі файли або класи (в даному проекті перевага надається саме файловій модульності). Інтерфейси між модулями мають бути чітко визначеними, щоб забезпечити чітку комунікацію та взаємодію між різними компонентами.

Використовуючи модульність, ми зможемо покращити організацію коду, підвищити його легкість у підтримці та сприяти повторному використанню. Модулі можна розробляти, тестувати та оновлювати незалежно, що полегшує розуміння та модифікацію певних функцій. Загалом, модульний код сприяє створенню більш чистої архітектури, зменшує складність і забезпечує більш керовану кодову базу, що забезпечує довготривалу підтримку і розширюваність програмної системи.

Крім зазначених раніше проблем, наш чат-бот також має відносно обмежену функціональність – це ситуація, коли код фокусується на конкретних функціях, таких як надання інформації про курс, робота в якості репетитора або коректора ШІ та встановлення нагадувань. Хоча ці функції можуть бути цінними самі по собі, код може не охоплювати широкий спектр потреб користувачів або не забезпечувати всебічний користувацький досвід.

На додаток до відсутності всеосяжних можливостей, обмежена функціональність також може призвести до неповного або незадовільного користувацького досвіду. Користувачі можуть очікувати безперешкодної та інтуїтивно зрозумілої взаємодії з програмним забезпеченням, але якщо код зосереджений лише на кількох конкретних функціональних можливостях, йому може бракувати зручності чи швидкості реагування, яких очікують користувачі. Це може призвести до розчарування користувачів, зниження їхньої активності та навіть до негативного сприйняття програмного забезпечення.

Ще одним недоліком обмеженої функціональності є втрачена можливість розглянути потенційні варіанти використання або вирішити додаткові проблеми користувачів. На даний момент код розроблений для вирішення певного набору завдань, але якщо не розширювати його функціональність з часом, ми обмежимо здатність коду адаптуватися до мінливих потреб користувачів, обмежуючи його загальну корисність і перешкоджаючи його довгостроковій життєздатності.

Щоб подолати недоліки обмеженої функціональності, ми можемо розглянути кілька стратегій. По-перше, проведення користувацьких досліджень та збір зворотного зв'язку може надати цінну інформацію про потреби та очікування цільової аудиторії. Ця інформація може допомогти розширити функціонал, щоб задовольнити ширший спектр потреб користувачів.

Крім того, можливим стає застосовувати ітеративний підхід до розробки, поступово додаючи нові функції та можливості на основі відгуків користувачів і тенденцій у сфері чат-ботів. Це дозволяє коду розвиватися і рости з часом, задовольняючи більше потреб користувачів і покращуючи загальний користувацький досвід.

Зрештою, вирішення проблеми обмеженої функціональності коду вимагає проактивного та орієнтованого на користувача підходу. Розуміючи потреби користувачів, постійно вдосконалюючи кодову базу та застосовуючи гнучкість і адаптивність, ми зможемо покращити функціональність коду, забезпечити більш повний користувацький досвід та підвищити його цінність для цільової аудиторії.

Розробленому коду також може бракувати персоналізації. Персоналізація означає пристосування програмного забезпечення до вподобань, потреб і характеристик окремих користувачів. Однак код не враховує вподобання користувачів і майже не зберігає інформацію про них, що призводить до загального та знеособленого досвіду для всіх користувачів.

Без персоналізації код може втратити можливості для підвищення залученості та задоволеності користувачів. Персоналізація може включати запам'ятовування вподобань користувача, відображення персоналізованих рекомендацій або контенту, а також надання індивідуального користувацького



інтерфейсу. Розуміючи вподобання окремих користувачів, код може запропонувати більш адаптований та релевантний досвід, підвищуючи задоволеність та утримання користувачів.

Крім того, відсутність персоналізації може обмежити здатність коду адаптуватися і вчитися на взаємодії з користувачем з часом. Зберігаючи інформацію про користувача, таку як історичні дані або профілі користувачів, код може збирати інсайти і приймати розумні рішення, щоб краще відповідати потребам кожного користувача. Це може включати адаптацію рекомендацій, коригування налаштувань або динамічне налаштування користувацького інтерфейсу на основі індивідуальних уподобань.

Щоб вирішити проблему обмеженої обробки помилок при введенні користувачем даних, розробники можуть впровадити надійні механізми перевірки даних. Це включає перевірку даних на відповідність очікуваним форматам, діапазонам або шаблонам, а також надання інформативних повідомлень про помилки або підказок, які допоможуть користувачам виправити помилки при введенні даних. Реалізація належної обробки винятків та реєстрації помилок також може допомогти ефективно фіксувати та повідомляти про помилки, допомагаючи налагодженню та вирішенню проблем.

Щодо персоналізації, розробники можуть використовувати такі методи, як профілювання користувачів, відстеження вподобань та алгоритми машинного навчання для збору даних про користувачів і відповідної адаптації програмного забезпечення. Це може включати безпечне зберігання користувацьких вподобань, аналіз моделей поведінки користувачів та використання цієї інформації для персоналізації рекомендацій, контенту чи елементів користувацького інтерфейсу.

Таким чином, вирішення проблеми обмеженої обробки помилок при введенні користувачем даних і відсутності персоналізації вимагає уваги до взаємодії з користувачем і його кастомізації. Впроваджуючи надійну валідацію даних, надаючи інформативний зворотний зв'язок про помилки та впроваджуючи функції персоналізації, розробники можуть покращити користувацький досвід,

підвищити залученість користувачів та зробити програмне забезпечення більш пристосованим до індивідуальних потреб користувачів.

### **3.2. Напрямки розвитку та модернізації розробленого чатботу**

У цьому розділі розглядаються напрямки розвитку та модернізації системи, розробленої в цьому дослідженні. Використовуючи новітні технології та усуваючи виявлені обмеження (див. розділ 3.1), ми прагнемо розширити можливості чат-бота та покращити його загальну продуктивність. Завдяки цим досягненням стане можливим створити більш досконалих і орієнтованих на користувача розмовних агентів, які зможуть ефективно підтримувати користувачів у різних сферах і додатках.

Перша рекомендація – це система автентифікації користувачів, яка є важливим компонентом Telegram-бота, оскільки вона дозволяє користувачам створювати акаунти та отримувати доступ до персоналізованих функцій. Завдяки надійному зберіганню облікових даних користувача та реалізації функції входу/виходу, система забезпечує конфіденційність та безпеку даних користувача. Крім того, система полегшує надання персоналізованих функцій, таких як збереження нагадувань та налаштувань, щоб підвищити рівень залученості та задоволеності користувачів.

Першим кроком у впровадженні системи автентифікації користувачів є увімкнення реєстрації користувачів. Коли користувач вперше взаємодіє з ботом, він отримує запит на реєстрацію. Це вікно збирає необхідну інформацію від користувача, таку як ім'я користувача, адреса електронної пошти та пароль. Зібрані дані потім надійно зберігаються в базі даних, забезпечуючи конфіденційність облікових даних користувача.

Для забезпечення безпеки даних користувачів важливо вживати належних заходів для зберігання облікових даних користувачів. Це включає хешування паролів за допомогою стійкого алгоритму хешування, такого як bcrypt або Argon2, перед тим, як зберігати їх у базі даних. Хешування гарантує, що навіть у

випадку витоку даних, оригінальні паролі неможливо буде легко відновити. Крім того, дуже важливо застосовувати методи соління для подальшого підвищення безпеки паролів.

Після успішної автентифікації система може надавати користувачам персоналізовані функції. Ці функції можуть включати можливість зберігати нагадування, встановлювати вподобання, налаштовувати параметри та відстежувати свій прогрес у боті. Пристосовуючи користувацький досвід до індивідуальних уподобань, бот може підвищити рівень залученості та задоволеності користувачів.

Щоб інтегрувати систему автентифікації користувачів в існуючу інфраструктуру Telegram-ботів, необхідно виконати наступні кроки:

1. Система автентифікації користувачів має бути інтегрована з Telegram Bot API, щоб полегшити комунікацію між ботом і користувачами. Ця інтеграція дозволяє боту отримувати реєстраційні дані користувачів, перевіряти їхні облікові дані та надавати персоналізовані функції.

2. Система повинна бути інтегрована з системою управління базами даних для безпечного зберігання та пошуку даних користувачів. База даних повинна мати можливість зберігати реєстраційні дані користувачів, хешовані паролі та будь-яку додаткову інформацію про користувача, необхідну для персоналізації функцій.

3. Система автентифікації користувачів повинна бути легко інтегрована в користувацький інтерфейс бота. Це передбачає розробку інтуїтивно зрозумілих екранів реєстрації та входу в систему, а також включення персоналізованих опцій в меню або команди бота.

Таким чином, впровадження системи автентифікації користувачів для Telegram-бота забезпечує безпечну та персоналізовану взаємодію з користувачем. Дозволяючи користувачам створювати облікові записи, безпечно зберігати облікові дані та надавати персоналізовані функції, система підвищує рівень залученості та задоволеності користувачів. Інтеграція з Telegram Bot API та системою управління базами даних забезпечує ефективну комунікацію та

зберігання даних. Майбутні вдосконалення можуть включати багатфакторну автентифікацію та постійний моніторинг вразливостей безпеки для подальшого підвищення надійності системи та довіри користувачів.

Ще одним напрямком для модернізації чат-боту є форум для обговорення курсу (або курсів), оскільки він сприяє спільному навчанню та дозволяє користувачам брати участь у дискусіях, пов'язаних з курсом. Завдяки функціям, що дозволяють ставити запитання, надавати відповіді та сприяти взаємодії між користувачами, форум покращує обмін знаннями та створює навчальне середовище, кероване спільнотою. Крім того, включення таких функцій, як теги, функції пошуку та системи сповіщень, покращує зручність та доступність форуму.

Першим кроком у створенні дискусійного форуму курсу у чат-боті є розробка платформи всередині бота, яка дозволить користувачам брати участь у дискусіях, пов'язаних з курсом. Ця платформа слугує централізованим центром, де користувачі можуть ставити запитання, надавати відповіді та брати участь у дискусіях. Дизайн повинен враховувати інтуїтивно зрозумілі інтерфейси, спрощену навігацію та ефективну організацію інформації, щоб забезпечити безперебійну роботу користувачів.

Щоб заохотити активну участь, форум повинен надавати користувачам можливість ставити запитання та отримувати відповіді від викладачів або однокурсників. Користувачі можуть створювати дискусійні теми, публікуючи запитання, які можна класифікувати за темами чи модулями курсу. Інші користувачі можуть надавати відповіді, ділитися думками або брати участь в обговореннях, пов'язаних з розміщеними запитаннями. Для сприяння чіткому та організованому обговоренню повинні бути доступні належні параметри створення та форматування тем.

У цьому контексті також необхідно розглянути можливість впровадження системи тегів, дозволяє користувачам категоризувати і класифікувати дискусії на основі певних тем або ключових слів. Теги забезпечують ефективний спосіб фільтрації та пошуку відповідних обговорень, гарантуючи, що користувачі

зможуть легко знаходити теми, які їх цікавлять. Користувачі можуть додавати відповідні теги до своїх тем, а система повинна підтримувати навігацію на основі тегів, щоб підвищити доступність та релевантність.

Щоб полегшити ефективний пошук інформації, на дискусійному форумі курсу має бути реалізована потужна функція пошуку. Користувачі можуть вводити ключові слова або фрази для пошуку конкретних дискусій або тем. Алгоритм пошуку має визначати пріоритетність відповідних обговорень на основі таких факторів, як свіжість, релевантність та активність користувачів. Розширені пошукові фільтри і параметри сортування можуть ще більше покращити процес пошуку.

Користувачі також повинні отримувати сповіщення про нові відповіді або коментарі на їхні запитання чи обговорення, в яких вони беруть активну участь. Крім того, система може надавати персоналізовані сповіщення на основі вподобань користувача, наприклад, про оновлення певних тем курсу, нові теми для обговорення чи оголошення від викладачів.

Ще однією рекомендацією є створення функції надсилання завдань, яка відіграє вирішальну роль у покращенні користувацького досвіду Telegram-бота та спрощенні процесу комунікації між студентами та викладачами. Дозволяючи користувачам надсилати завдання безпосередньо через бота, система зменшує тертя та підвищує ефективність.

Першим кроком у реалізації функції подання завдань є розробка зручного інтерфейсу в Telegram-боті. Цей інтерфейс повинен дозволяти користувачам завантажувати файли завдань, вводити відповідну інформацію (наприклад, назву завдання, код курсу та дату здачі) та переглядати свої роботи перед тим, як їх фіналізувати. Інтерфейс повинен відповідати найкращим практикам юзабіліті та доступності для забезпечення безперешкодної подачі робіт.

Для забезпечення послідовності та чіткості слід визначити чіткі рекомендації щодо формату та вимог до подання завдань. Ці рекомендації можуть включати специфікації щодо типів файлів, максимального розміру файлів, правил іменування та будь-яких додаткових інструкцій, наданих викладачами.

Користувачі повинні мати доступ до цих інструкцій в інтерфейсі подання завдань, щоб користуватися ними під час підготовки своїх робіт.

Щоб допомогти користувачам дотримуватися дедлайнів, цей модуль можна поєднати з наявною системою сповіщень. Користувачі будуть отримувати своєчасні нагадування про наближення термінів виконання завдань, що дозволить їм ефективно управляти своїм часом і уникати пропусків дедлайнів. Крім того, цей модуль має мати можливість інтеграції з існуючими системами управління навчанням (LMS), такими як Moodle, для безперешкодної синхронізації даних курсу, завдань, оцінок та іншої важливої інформації.

У контексті модернізації боту також варто розглянути можливість створення модулі інтерактивних завдань та уроків, таких як вікторини, вправи та завдання з кодування, користувачі можуть брати активну участь у навчальному процесі та отримувати негайний зворотній зв'язок про свій прогрес безпосередньо у боті.

Першим кроком у розробці інтерактивних уроків є визначення структури та створення цікавого контенту. Уроки повинні бути організовані логічно, з чіткою навчальною метою та послідовністю вивчення тем. Створення контенту може передбачати включення мультимедійних елементів, таких як текст, зображення, відео та фрагменти коду, щоб забезпечити комплексний досвід навчання. Уроки мають бути розроблені відповідно до рівня знань цільової аудиторії та навчальних цілей.

Щоб оцінити розуміння та засвоєння знань користувачами, в інтерактивні уроки слід інтегрувати вікторини та оцінювання. Ці тести можуть включати запитання з декількома варіантами відповідей, заповнення пропусків, вправи на встановлення відповідності або будь-який інший відповідний формат. Запитання мають охоплювати ключові поняття та навчальні цілі уроку, надаючи користувачам можливість застосувати свої знання та отримати негайний зворотній зв'язок щодо їхніх відповідей.

Окрім тестів, інтерактивні уроки повинні включати вправи та практичні завдання, щоб закріпити знання та отримати практичний досвід. Ці вправи

можуть включати завдання з кодування, сценарії розв'язання проблем або будь-які інші інтерактивні дії, що стосуються предмету. Користувачі повинні мати можливість взаємодіяти з вправами безпосередньо в інтерфейсі Telegram-бота, вводячи свої рішення та отримуючи негайний зворотній зв'язок щодо їх виконання.

Важливим аспектом цього модуля є можливість надання негайного зворотного зв'язку користувачам на основі їхніх відповідей. Система повинна аналізувати відповіді користувачів, оцінювати їхню точність і надавати зворотний зв'язок у режимі реального часу. Зворотний зв'язок може включати пояснення правильних і неправильних відповідей, пропозиції щодо вдосконалення або додаткові ресурси для подальшого вивчення. Такий негайний зворотний зв'язок допомагає користувачам визначити свої сильні сторони та сфери для вдосконалення, покращуючи навчальний досвід та полегшуючи здобуття знань.

На завершення, розглянемо напрямки розвитку та модернізації модулю обробка природної мови у боті. Покращуючи розуміння ботом природної мови та її генерування, користувачі можуть мати більш змістовну та контекстуально релевантну взаємодію. Запропоновані вдосконалення передбачають використання передових методів NLP, таких як моделі NLP, семантичний аналіз та вилучення контексту, щоб краще розуміти запити користувачів і надавати точні та контекстно-орієнтовані відповіді.

Так, щоб покращити розуміння ботом природної мови, можна дослідити методи семантичного аналізу, які згодом можуть бути використані для вилучення значення і контексту із запитів користувачів. Це передбачає аналіз синтаксичної та семантичної структури речень, щоб зрозуміти зв'язки між словами, фразами та поняттями. Такі методи, як розпізнавання іменованих сутностей, тегування частин мови та семантичне маркування ролей, можуть допомогти у вилученні релевантної інформації та покращити розуміння ботом користувацького вводу.

Для надання контекстно-орієнтованих відповідей важливо виокремити та врахувати контекст розмови. Цього можна досягти за допомогою таких методів,

як відстеження стану діалогу, коли бот зберігає історію розмови, щоб розуміти поточний контекст і реагувати відповідно до нього. Крім того, роздільна здатність може допомогти у вирішенні неоднозначних посилань на об'єкти, згадані в розмові, підвищуючи точність відповідей бота.

Для покращення можливостей NLP необхідно зібрати різноманітний та репрезентативний набір даних про запити та відповіді користувачів. Набір даних повинен охоплювати різні теми та наміри користувачів, щоб забезпечити всебічне навчання моделей NLP. Зібрані дані можуть потребувати попередньої обробки, включаючи очищення, токенизацію та анотування, щоб підготувати їх до навчання та оцінювання.

Для навчання на зібраному наборі даних можна використовувати NLP-вже доступну модель Chat GPT 3.5. Модель можна буде доопрацьовувати, використовуючи методи трансферного навчання, щоб адаптувати її до конкретного завдання розуміння запитів користувачів і генерування релевантних відповідей. Тонке налаштування передбачає навчання моделі на зібраному наборі даних з урахуванням бажаних показників оцінки, таких як точність, достовірність і пригадування.

Після того, як NLP-моделі будуть навчені та налаштовані, їх можна буде інтегрувати у внутрішню систему бота Telegram. Для оцінки продуктивності розширених можливостей NLP слід провести широке тестування, зокрема виміряти точність розпізнавання намірів, вилучення сутностей і генерування відповідей.

Покращуючи здатність бота розуміти та генерувати природну мову, користувачі зможуть вести більш інтерактивні та змістовні розмови. Бот зможе точно розуміти запити користувачів, надавати релевантну інформацію та генерувати контекстно-залежні відповіді, що сприятиме покращенню користувацького досвіду.

Також, актуальним буде розглянути можливість інтеграції бота з наявними інструментами продуктивності, такими як календар, таймери тощо. Інтеграція



підвищує ефективність, організованість і допомагає користувачам залишатися на висоті своїх академічних обов'язків.

Інтегруючи Telegram-бота з інструментами для підвищення продуктивності, користувачі можуть легко синхронізувати терміни і завдання, пов'язані з курсом, з власними календарями або системами управління завданнями. Це усуває необхідність ручного введення інформації та гарантує, що всі важливі дати та завдання будуть послідовно оновлюватися на всіх платформах.

Крім того, інтеграція з інструментами для підвищення продуктивності дозволяє користувачам використовувати розширені функції, пропоновані цими інструментами, такі як визначення пріоритетів завдань, підзадачі та відстеження прогресу. Користувачі можуть ефективно організовувати свої завдання, пов'язані з курсом, встановлювати нагадування та ефективно керувати своїм часом, що сприяє підвищенню продуктивності та кращому виконанню завдань.

Перший крок реалізації цієї рекомендації передбачає визначення та інтеграцію з API, що надаються обраними інструментами продуктивності, такими як API календаря або API управління завданнями. Ці API дозволяють Telegram-боту взаємодіяти із зовнішніми інструментами продуктивності та отримувати доступ до необхідної інформації, наприклад кодів сесії Google Meet для лекцій.

Щоб забезпечити безперебійну синхронізацію, слід впровадити систему автентифікації та авторизації користувачів, як вже було згадано раніше. Це гарантує, що тільки авторизовані користувачі зможуть підключити свої інструменти продуктивності до Telegram-бота. Користувачам може знадобитися надати необхідні дозволи для доступу до своїх календарів або систем управління завданнями.

Бот повинен надавати користувачам інтерфейс для синхронізації дедлайнів і завдань, пов'язаних з курсом. Цього можна досягти, дозволивши користувачам вводити відповідну інформацію, таку як терміни виконання завдань, розклад

іспитів або етапи проекту. Потім бот використовує інтегровані API для синхронізації цієї інформації з інструментами продуктивності користувача.

Інтеграція також може включати функції для відстеження та аналізу прогресу користувача. Цього можна досягти, фіксуючи виконані завдання, відстежуючи проміжні етапи та надаючи інформацію про рівень виконання завдань або загальний прогрес. Користувачі можуть краще зрозуміти свою академічну успішність і приймати обґрунтовані рішення щодо тайм-менеджменту та розстановки пріоритетів.

Таким чином, переваги подібної інтеграції включають підвищення ефективності, безперебійну роботу користувачів і можливості кастомізації для персоналізації підходу до управління курсами.

### **Висновки до розділу 3**

1. Розв'язання проблем обмеженої обробки помилок під час введення користувацьких даних та відсутності персоналізації вимагає зосередження уваги на взаємодії з користувачем та його кастомізації. Завдяки впровадженню надійних механізмів перевірки даних, наданню інформативного зворотного зв'язку про помилки та впровадженню функцій персоналізації розробники можуть покращити користувацький досвід, підвищити залученість користувачів та адаптувати програмне забезпечення до індивідуальних вимог користувачів. Наголошуючи на цих аспектах, система стає більш пристосованою до вподобань користувачів і забезпечує безперебійну та персоналізовану роботу з нею.

2. Наприкінці нашої роботи, ми підсумували основні напрямки розвитку та модернізації, підкресливши їхню важливість та потенційний вплив. Окреслюючи ці напрямки, ми закладаємо основу для майбутніх досліджень і розробок у цій галузі, заохочуючи безперервний розвиток і вдосконалення систем чат-ботів.

## ВИСНОВКИ

Розробка чат-бота, спеціально розробленого для університету та реалізованого на платформі месенджера Telegram, має величезне практичне значення та потенціал. Чат-боти, оснащені технологіями штучного інтелекту (ШІ) та обробки природної мови (ПМ), революціонізували взаємодію людини з комп'ютером і надали цінні послуги в різних сферах. Використовуючи ці технології, університетські чат-боти можуть значно покращити комунікацію та підтримку студентів, пропонуючи широкий спектр функціональних можливостей та послуг.

Обрана архітектура університетського чат-бота, що включає такі інструменти, як Alogram, OpenAI, ctparse та модулі логування, відіграє вирішальну роль у забезпеченні ефективної взаємодії з користувачами та безперебійної обробки інформації. Alogram забезпечує комплексну основу для створення чат-ботів на платформі Telegram, а мовна модель OpenAI полегшує генерування контекстно-релевантних відповідей. Модуль ctparse покращує здатність чат-бота аналізувати та витягувати інформацію з введених користувачем даних, що дозволяє ефективно обробляти нагадування та надавати точну інформацію про курс. Крім того, модуль logging забезпечує збір і ведення журналів взаємодії для аналізу та вдосконалення.

Варто відзначити універсальність університетського чат-бота, який може виконувати різні функції в академічному середовищі. Виконуючи роль репетитора з письма зі штучним інтелектом, він може допомогти студентам покращити навички письма та надати цінний зворотний зв'язок. Як статистик, чат-бот може аналізувати та надавати статистичну інформацію, допомагаючи студентам у їхніх дослідженнях на основі даних. Його роль перекладача англійської мови забезпечує безперешкодну комунікацію для іноземних студентів, сприяючи їхній інтеграції в університетську спільноту. Крім того, виступаючи в ролі коректора, чат-бот може допомогти студентам підвищити якість і точність їхніх письмових робіт. Більше того, надаючи інформацію про

різні курси та встановлюючи нагадування про завдання, чат-бот стає незамінним інструментом для управління академічними обов'язками.

Модульна архітектура університетського чат-бота забезпечує ефективну комунікацію та координацію між різними його компонентами. Це дозволяє чат-боту обробляти вхідні дані користувачів, генерувати відповідні відповіді та надавати своєчасну та релевантну інформацію. Методи вилучення даних відіграють вирішальну роль у наданні точних і контекстуально відповідних відповідей, підвищуючи здатність чат-бота ефективно задовольняти потреби користувачів.

Крім того, ключовим аспектом для забезпечення ефективності та задоволеності користувачів є зосередження уваги на взаємодії та кастомізації чат-ботів. Надійні механізми перевірки даних допомагають підтримувати їхню цілісність і точність, а інформативний зворотний зв'язок про помилки допомагає користувачам ефективно виправляти введені дані. Надійні механізми перевірки даних допомагають підтримувати їхню цілісність і точність, а інформативний зворотний зв'язок про помилки допомагає користувачам ефективно виправляти введені дані. Функції персоналізації, такі як адаптація відповідей до індивідуальних уподобань користувача та надання індивідуальних рекомендацій, покращують користувацький досвід і сприяють залученню. Враховуючи ці аспекти, розробники можуть оптимізувати роботу чат-бота, підвищити рівень задоволеності користувачів і створити більш персоналізоване та орієнтоване на користувача програмне рішення.

Отже, розробка університетського чат-бота на платформі Telegram має значні переваги та практичну користь. Його здатність покращувати комунікацію, підтримувати студентів та надавати цінні послуги в різних сферах є свідченням потужності технологій ШІ та NLP. Завдяки добре продуманій архітектурі, що включає інструменти та модулі, які сприяють ефективній взаємодії з користувачами та обробці інформації, університетський чат-бот стає цінним ресурсом в академічній спільноті. Постійно вдосконалюючи та розширюючи свої можливості, задовольняючи потреби користувачів та надаючи пріоритет

персоналізованому досвіду, університетський чат-бот має величезний потенціал для трансформації та покращення загального студентського досвіду в університетській екосистемі.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Барило Я. Як створити чат-бот для сфери освіти: інструкція та приклади. *SendPulse Блог*. URL: <https://sendpulse.ua/blog/chatbot-for-education-sphere> (дата звернення: 28.05.2023).
2. Все про чат-боти: типи і приклади, якому бізнесу підійде, список конструкторів для створення. *Webpromo*. URL: <https://webpromo.ua/ua/blog/vse-o-chat-botah-tipy-i-primery-kakomu-biznesu-podojdet-spisok-konstruktorov-dlya-sozdaniya/> (дата звернення: 28.05.2023).
3. Козуб, Г., & Скарга, В. (2023). Автоматизація університетських процесів за допомогою розробки telegram-бота. (с. 277–279).
4. Ушакова І. О. Підходи до створення інтелектуальних чат-ботів. *Системи обробки інформації*. 2019. Вип. 2 (157). С. 76–83.
5. Що таке чат-бот (chatbot): секрети використання та основні переваги. *Блог HelpCrunch*. URL: <https://helpcrunch.com/blog/uk/shcho-take-chat-bot/> (дата звернення: 28.05.2023).
6. Що таке Chatbot (чат-боти) та кому вони потрібні?. *Creative SMM*. URL: <https://creativesmm.com.ua/shho-take-chatbot-ta-komu-vonu-potribni/> (дата звернення: 28.05.2023).
7. A History of SmarterChild. *VICE - Unbequemer Journalismus und Dokus zu allem, was wichtig ist auf der Welt*. URL: <https://www.vice.com/en/article/jpgpey/a-history-of-smarterchild> (date of access: 28.05.2023).
8. Adamopoulou E., Moussiades L. Chatbots: history, technology, and applications. *Machine learning with applications*. 2020. Vol. 2. P. 100006. URL: <https://doi.org/10.1016/j.mlwa.2020.100006> (date of access: 28.05.2023).
9. Allen J. 10 years of Siri: the history of Apple's voice assistant. *TechRadar*. URL: <https://www.techradar.com/news/siri-10-year-anniversary> (date of access: 28.05.2023).
10. Ctparse. *PyPI*. URL: <https://pypi.org/project/ctparse/> (date of access: 28.05.2023).
11. GitHub - wadetb/eliza: Python implementation of the Eliza chatbot. *GitHub*. URL: <https://github.com/wadetb/eliza> (date of access: 28.05.2023).

- 12.OpenAI. *OpenAI*. URL: <https://openai.com> (date of access: 28.05.2023).
- 13.Papers with code - named entity recognition (NER). *The latest in Machine Learning / Papers With Code*. URL: <https://paperswithcode.com/task/named-entity-recognition-ner> (date of access: 28.05.2023).
- 14.POS Tagging with NLTK and Chunking in NLP [EXAMPLES]. *Guru99*. URL: <https://www.guru99.com/pos-tagging-chunking-nltk.html> (date of access: 28.05.2023).
- 15.Real Python. Logging in python – real python. *Python Tutorials – Real Python*. URL: <https://realpython.com/python-logging/> (date of access: 28.05.2023).
- 16.STEMEdBot. *Telegram*. URL: <https://t.me/STEMEdBot> (date of access: 28.05.2023).
- 17.Telegram bot API. *Telegram APIs*. URL: <https://core.telegram.org/bots/api> (date of access: 28.05.2023).
- 18.Telegram bot BotFather – @botfather. *Best online chat bots and assistants*. URL: <https://botstore.com/c/botfather/> (date of access: 28.05.2023).
- 19.Tokenization in NLP: types, challenges, examples, tools. *neptune.ai*. URL: <https://neptune.ai/blog/tokenization-in-nlp> (date of access: 28.05.2023).
- 20.Wallace R. S. The anatomy of A.L.I.C.E. *SpringerLink*. URL: [https://link.springer.com/chapter/10.1007/978-1-4020-6710-5\\_13](https://link.springer.com/chapter/10.1007/978-1-4020-6710-5_13) (date of access: 28.05.2023).
- 21.Welcome to aiogram's documentation! – aiogram 2.25.1 documentation. *aiogram 2.25.1 documentation*. URL: <https://docs.aiogram.dev/en/latest/index.html> (date of access: 28.05.2023).
- 22.Welcome to python.org. *Python.org*. URL: <https://www.python.org> (date of access: 28.05.2023).
- 23.When PARRY met ELIZA: a ridiculous chatbot conversation from 1972. *The Atlantic*. URL: <https://www.theatlantic.com/technology/archive/2014/06/when-parry-met-eliza-a-ridiculous-chatbot-conversation-from-1972/372428/> (date of access: 28.05.2023).

## ДОДАТКИ

## Додаток А. Сертифікат апробації результатів дослідження





**Додаток Б.**

	<b>Міністерство освіти і науки України</b>
	<b>Державний заклад “Луганський національний університет</b>
	<b>імені Тараса Шевченка”</b>
Факультет (інститут)	Навчально-науковий інститут фізики, математики та інформаційних технологій
	<small>(повна назва)</small>
Кафедра	Фізико-технічних систем та інформатики
	<small>(повна назва)</small>

**Методика тестування**  
на виконання програмної розробки (ПР):  
**“АВТОМАТИЗАЦІЯ УНІВЕРСИТЕТСЬКИХ ПРОЦЕСІВ ЗА**  
**ДОПОМОГОЮ РОЗРОБКИ TELEGRAM-БОТА”**

## **ЗМІСТ**

1. ОБ'ЄКТ ВИПРОБУВАНЬ .....	83
2. МЕТА ТЕСТУВАННЯ .....	83
3. МЕТОДИ ТЕСТУВАННЯ.....	83

## **1. ОБ'ЄКТ ВИПРОБУВАНЬ**

Розроблений Telegram Bot та всі його функції повинні надійно працювати на смартфоні на базі ОС Android 13, iOS 16.5, Windows 11.

Для виміру якості розробленого програмного продукту виконано тестування бота у десктопній та мобільній версії Telegram.

Програмний продукт повинен:

1. Надійно працювати на смартфоні.
2. Надійно працювати на комп'ютері.

## **2. МЕТА ТЕСТУВАННЯ**

Метою тестування є оцінка функціональності та продуктивності Telegram-бота, розробленого для університету.

## **3. МЕТОДИ ТЕСТУВАННЯ**

Для досягнення цілей та завдань тестування буде застосовано комбінацію методів тестування, включаючи, але не обмежуючись ними:

Функціональне тестування: цей метод передбачав оцінку особливостей та функціональних можливостей Telegram-бота, щоб переконатися, що вони працюють як очікувалося. Він охоплював тестування різних взаємодій з користувачем, таких як процес пошуку інформації у боті, сповіщення та адміністративні завдання.

Юзабіліті-тестування фокусувалось на оцінці зручності та інтуїтивності інтерфейсу Telegram-бота. Воно передбачало спостереження та збір відгуків від репрезентативних користувачів, щоб виявити будь-які сфери для покращення з точки зору простоти використання, чіткості інструкцій, навігації та загального користувацького досвіду.

Тестування продуктивності. Цей метод мав на меті виміряти продуктивність і швидкість реагування Telegram-бота за різних сценаріїв і робочих навантажень. Він охопив оцінку таких факторів, як час відгуку, обробка навантаження на сервер і масштабованість, щоб забезпечити оптимальну продуктивність навіть у пікові періоди використання.

Перший тестовий приклад, `test_start_command_handler`, фокусується на обробнику команди `/start` (функція `start()`). У ньому використовується об'єкт імітації повідомлення для імітації користувача, який ініціює команду `/start`. Тест перевіряє, чи викликається функція `reply()` рівно один раз, гарантуючи, що обробник команди відреагує належним чином.

Переходимо до наступного тесту, `test_help_command_handler`, який перевіряє поведінку обробника команди `/help` (функція `help_command()`). Як і у попередньому тесті, для імітації виклику користувачем команди `/help` використовується об'єкт імітації повідомлення. Тест гарантує, що функція `reply()` буде викликана рівно один раз, перевіряючи правильність роботи обробника команди.

Третій тест, `test_course_info_command_handler`, зосереджується на обробнику команди `/course_info` (функція `course_info_button_handler()`). Знову ж таки, використовується об'єкт `mock`-повідомлення, і тест перевіряє, чи викликається функція `reply()` рівно один раз. Цей тест гарантує, що обробник команди правильно обробляє команду `/course_info`.

Далі у нас є тест `test_handle_course_buttons`, який перевіряє поведінку обробника кнопок курсу (функція `handle_course_buttons()`). Використовуючи об'єкт `mock`-повідомлення, тест перевіряє, чи викликається функція `reply()` рівно один раз. Цей тест гарантує, що обробник правильно обробляє кнопки курсу.

Наступний тест `test_handle_course_info_buttons` перевіряє функціональність обробника інформаційних кнопок курсу (функція `handle_course_info_buttons()`). У тесті використовується об'єкт повідомлення з текстом "Syllabus for Python Programming" і перевіряється, чи викликається функція `reply()` рівно один раз. Цей тест гарантує, що обробник належним чином обробляє кнопки з інформацією про курс.

Далі, тест `test_process_reminder` фокусується на обробці нагадувань у функції `process_reminder()`. Він використовує об'єкт повідомлення з текстом "Нагадай мені 2023-06-10T14:00, Виконати завдання" для імітації запиту на

нагадування. Тест перевіряє, чи викликається функція `reply()` рівно один раз, гарантуючи, що обробка нагадування працює коректно.

Нарешті, тест `test_on_subscribe` перевіряє поведінку функції підписки (`on_subscribe()`). Він використовує об'єкт `mock`-повідомлення з аргументом `"physics"` для імітації користувача, який підписується на курс. Тест перевіряє, чи викликається функція `reply()` рівно один раз, підтверджуючи, що функція підписки працює належним чином.

Ці тестові приклади охоплюють різні аспекти коду, такі як обробники команд, обробники кнопок, обробка нагадувань і функціональність підписки. Використовуючи ці методи тестування, було всебічно оцінено продуктивність, функціональність та зручність Telegram-бота, тим самим забезпечивши його ефективність та успішне розгортання.

Telegram bot працює на всіх вищезгаданих платформах.

## Додаток В. Основний код STEMEdBot

```

import logging
import asyncio
import openai
from aiogram import Bot, Dispatcher, types
from aiogram.contrib.middlewares.logging import LoggingMiddleware
from aiogram.types import ParseMode
from aiogram.dispatcher.filters.state import State, StatesGroup
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, ReplyKeyboardRemove
from datetime import datetime
import ctparse

API_TOKEN = 'BOT TOKEN HERE'
openai.api_key = 'CHAT GPT API HERE'

logging.basicConfig(level=logging.INFO)

bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)
dp.middleware.setup(LoggingMiddleware())

# All lists

reminders = []
groups = []
courses = [
    {
        "name": "Python Programming",
        "syllabus": "\n- Introduction to Python\n- Python Data Types\n- Control Flow Statements\n- Functions and Modules\n- File Handling\n- Object-Oriented Programming\n- Error Handling\n- Testing and Debugging\n- Advanced Topics in Python",
        "schedule": "\n- Monday: 10:00 AM - 12:00 PM\n- Wednesday: 2:00 PM - 4:00 PM\n- Friday: 9:00 AM - 11:00 AM",
        "resources": "\n- Recommended textbook: 'Python Crash Course' by Eric Matthes\n- Online tutorials and practice exercises\n- Python documentation and community forums",
        "attendance": "\nAttendance will be taken during each class session. It is important to attend regularly to actively participate in discussions and hands-on activities.",
        "grades": "\nYour final grade will be based on assignments (30%), quizzes (20%), midterm exam (25%), and a final project (25%).",
    },
    {
        "name": "Data Science",
        "syllabus": "\n- Introduction to Data Science\n- Data Manipulation and Analysis\n- Data Visualization\n- Machine Learning\n- Statistical Analysis\n- Big Data and Spark\n- Deep Learning\n- Data Science Projects",
        "schedule": "\n- Tuesday: 1:00 PM - 3:00 PM\n- Thursday: 10:00 AM - 12:00 PM\n- Saturday: 3:00 PM - 5:00 PM",
        "resources": "\n- Recommended textbook: 'Python for Data Analysis' by Wes McKinney\n- Online tutorials and case studies\n- Data science libraries and tools (e.g., NumPy, Pandas, Matplotlib, scikit-learn)",
    }
]

```

```

    "attendance": "\nRegular attendance is expected for all classes. Your active participation in class
discussions and group projects will contribute to your learning experience.",
    "grades": "\nYour performance will be evaluated based on assignments (30%), projects (25%),
quizzes (15%), midterm exam (20%), and a final exam (10%)."
},
{
    "name": "Web Development",
    "syllabus": "\n- Introduction to Web Development\n- HTML and CSS\n- JavaScript and Front-
end Frameworks\n- Back-end Development\n- Databases\n- Web APIs and RESTful Services\n-
Deployment and Hosting\n- Web Development Projects",
    "schedule": "\n- Monday: 2:00 PM - 4:00 PM\n- Wednesday: 10:00 AM - 12:00 PM\n- Friday:
1:00 PM - 3:00 PM",
    "resources": "\n- Recommended textbook: 'Eloquent JavaScript' by Marijn Haverbeke\n- Online
tutorials and interactive coding platforms\n- Web development frameworks and tools (e.g., React,
Node.js, Django)",
    "attendance": "\nRegular attendance is expected for all classes. Your active participation in
coding exercises and projects will enhance your learning and practical skills.",
    "grades": "\nYour grade will be determined based on assignments (30%), projects (25%), quizzes
(15%), participation (10%), midterm exam (10%), and a final exam (10%)."
}
]

```

# Classes

```
class ReminderStates(StatesGroup):
```

```

    title = State()
    description = State()
    date = State()
    time = State()

```

# Handler for /start command

```

@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)
    button1 = KeyboardButton("How to use")
    keyboard.add(button1)

    instructions = "/start - запуск бота\n" \
        "/help - інструкції з використання AI та нагадувань\n" \
        "/course_info - інформація про курси"
    await message.reply(instructions)

```

# Handler for /help command

```
contact_us_message = "If you have any further inquiries, feel free to reach out to us at our support
channel: https://t..https://t.me/kArdHouse"
```

```

@dp.message_handler(commands=['help'])
async def help_command(message: types.Message):
    faq = "***FAQ***\n\n" \

```

```

    """Q: How can I get help with writing or ask questions?""" \
    "A: Start your message with \"Act as an AI Writing Tutor:\", \"Act as a Statistician:\", \"Act as a Proofreader:\", \"Act as an English Translator and Improver:\" followed by your question or the content you need help with. " \
    "The bot will provide feedback or suggestions using artificial intelligence tools.\n\n" \
    """Q: How can I set a reminder?""" \
    "A: Use the format \"Remind me YYYY-MM-DDTHH:MM, reminder message\" to set a reminder. " \
    "The bot will schedule the reminder and notify you at the specified date and time.\n\n" \
    "If you have any additional questions, you can contact me/c here: https://t.me/kArdHouse"

```

```

await message.reply(faq, parse_mode=ParseMode.MARKDOWN)

```

```

# Course info

```

```

@dp.message_handler(lambda message: message.text == "Course info")
async def course_info_button_handler(message: types.Message):
    await course_info(message)

```

```

# Chat GPT module

```

```

async def generate_answer(prompt: str) -> str:
    response = openai.Completion.create(
        engine="text-davinci-002",
        prompt=prompt,
        max_tokens=150,
        n=1,
        stop=None,
        temperature=0.5,
    )

    answer = response.choices[0].text.strip()
    return answer

```

```

@dp.message_handler(lambda message: message.text and message.text.startswith("Act as an AI Writing Tutor:"))
async def student_question(message: types.Message):
    user_question = message.text.replace('Act as an AI Writing Tutor:', '').strip()
    prompt = f'I want you to act as an AI writing tutor. I will provide you with a student who needs help improving " \
        f"their writing and your task is to use artificial intelligence tools, such as natural language " \
        f"processing, to give the student feedback on how they can improve their composition. " \
        f"You should also use " \
        f"your rhetorical knowledge and experience about effective writing techniques in order to suggest ways " \
        f"that the student can better express their thoughts and ideas in written form. My first request " \
        f"is: {user_question}"
    answer = await generate_answer(prompt)

```



```

await message.reply(f'Chat GPT: {answer}', parse_mode=ParseMode.MARKDOWN)

@dp.message_handler(lambda message: message.text and message.text.startswith("Act as a
Statistician:"))
async def student_question(message: types.Message):
    user_question = message.text.replace('Act as a Statistician', '').strip()
    prompt = f'I want to act as a Statistician. I will provide you with details related with statistics. You
should be " \
        f'knowledge of statistics terminology, statistical distributions, confidence interval,
probabillity, " \
        f'hypothesis testing and statistical charts. My first request is: {user_question}'
    answer = await generate_answer(prompt)
    await message.reply(f'Chat GPT: {answer}', parse_mode=ParseMode.MARKDOWN)

@dp.message_handler(lambda message: message.text and message.text.startswith("Act as an
English Translator and Improver:"))
async def student_question(message: types.Message):
    user_question = message.text.replace('Act as an English Translator and Improver', '').strip()
    prompt = f'I want you to act as an English translator, spelling corrector and improver. I will speak
to you in any " \
        f'language and you will detect the language, translate it and answer in the corrected and
improved version " \
        f'of my text, in English. I want you to replace my simplified A0-level words and sentences
with more " \
        f'beautiful and elegant, upper level English words and sentences. Keep the meaning same,
but make them " \
        f'more literary. I want you to only reply the correction, the improvements and nothing else,
do not write " \
        f'explanations. My first sentence is: {user_question}'
    answer = await generate_answer(prompt)
    await message.reply(f'Chat GPT: {answer}', parse_mode=ParseMode.MARKDOWN)

@dp.message_handler(lambda message: message.text and message.text.startswith("Act as a
Proofreader:"))
async def student_question(message: types.Message):
    user_question = message.text.replace('Act as a Proofreader', '').strip()
    prompt = f'I want you act as a proofreader. I will provide you texts and I would like you to review
them for any " \
        f'spelling, grammar, or punctuation errors. Once you have finished reviewing the text,
provide me with " \
        f'any necessary corrections or suggestions for improve the text.: {user_question}'
    answer = await generate_answer(prompt)
    await message.reply(f'Chat GPT: {answer}', parse_mode=ParseMode.MARKDOWN)

# Course information module

@dp.message_handler(commands=['course_info'])
async def course_info(message: types.Message):

    # Create a reply keyboard with buttons for courses
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)

```

```

for course in courses:
    button = KeyboardButton(course["name"])
    keyboard.add(button)

# Send the message with the reply keyboard
await message.reply("Choose a course:", reply_markup=keyboard)

@dp.message_handler(lambda message: message.text in [course["name"] for course in courses])
async def handle_course_buttons(message: types.Message):
    course_index = [course["name"] for course in courses].index(message.text)
    course = courses[course_index]

    # Create a reply keyboard with buttons
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)
    button1 = KeyboardButton(f'Syllabus for {course["name"]}')
    button2 = KeyboardButton(f'Schedule for {course["name"]}')
    button3 = KeyboardButton(f'Resources for {course["name"]}')
    button4 = KeyboardButton(f'Attendance for {course["name"]}')
    button5 = KeyboardButton(f'Grades for {course["name"]}')
    keyboard.add(button1, button2, button3, button4, button5)

    # Send the message with the reply keyboard
    await message.reply(f'Course: {course["name"]}', reply_markup=keyboard)

@dp.message_handler(lambda message: message.text.startswith(("Syllabus for ", "Schedule for ",
"Resources for ",
                                "Attendance for ", "Grades for ")))
async def handle_course_info_buttons(message: types.Message):
    action, course_name = message.text.split(" for ")
    course_index = [course["name"] for course in courses].index(course_name)
    course = courses[course_index]

    if action == "Syllabus":
        response = f'Here is the syllabus for {course["name"]}:\n{course["syllabus"]}'
    elif action == "Schedule":
        response = f'Here is the schedule for {course["name"]}:\n{course["schedule"]}'
    elif action == "Resources":
        response = f'Here are the resources for {course["name"]}:\n{course["resources"]}'
    elif action == "Attendance":
        response = f'Attendance: {course["name"]}:\n{course["attendance"]}'
    else: # action == "Grades"
        response = f'Grades: {course["name"]}:\n{course["grades"]}'

    await message.reply(response, reply_markup=ReplyKeyboardRemove())

# Assignment reminders

# Chat GPT module for Assigmet

async def extract_reminder_info(message: str):

```

```

prompt = f'Extract reminder details from the following text: '{message}'. Provide the result in the
format "\
    f"YYYY-MM-DDTHH:MM, description of task"
response = openai.Completion.create(
    engine="text-davinci-002",
    prompt=prompt,
    max_tokens=100,
    n=1,
    stop=None,
    temperature=0.5,
)

extracted_info = response.choices[0].text.strip()
return extracted_info

@dp.message_handler(lambda message: message.text and message.text.lower().startswith("remind
me"))
async def process_reminder(message: types.Message):
    reminder_info = await extract_reminder_info(message.text)

    if ',' not in reminder_info:
        # Try to parse the natural language input
        parser = ctparse.CTParser()
        parsed = parser.parse(reminder_info)
        if not parsed:
            await message.reply(
                "Invalid reminder format. Please provide a date and time in the format YYYY-MM-
DDTHH:MM, "
                "followed by a comma and the reminder message, or use natural language input.")
            return
        reminder_datetime = parsed[0].start
        reminder_message = "Write to you"
    else:
        date_time, reminder_message = reminder_info.split(',', 1)
        date_time = date_time.strip()
        reminder_message = reminder_message.strip()

        try:
            reminder_datetime = datetime.strptime(date_time, "%Y-%m-%dT%H:%M")
            if reminder_datetime < datetime.now():
                raise ValueError("Reminder date and time must be in the future.")
        except ValueError:
            await message.reply(
                "Invalid date and time format. Please provide a date and time in the format YYYY-MM-
DDTHH:MM.")
            return

    await message.reply(f'Reminder set for {reminder_datetime}: {reminder_message}')

    reminders.append({"datetime": reminder_datetime, "message": reminder_message})

```

```

time_difference = (reminder_datetime - datetime.now()).total_seconds()
if time_difference > 0:
    await asyncio.sleep(time_difference)

await message.reply(f"Reminder: {reminder_message}")

# Subscription

@dp.message_handler(commands=['subscribe'])
async def on_subscribe(message: types.Message):
    section = message.get_args()
    if section:
        # Save the user's subscription to the database (not implemented here)
        await message.reply(f"You have successfully subscribed to the {section} section.")
    else:
        await message.reply("Please provide a section to subscribe to, e.g., /subscribe physics")

    async def send_news_update(section: str, news: str):
        # Retrieve the list of subscribed users from the database (not implemented here)
        subscribed_users = []

        for user_id in subscribed_users:
            await bot.send_message(
                chat_id=user_id,
                text=f"📰 *{section} News Update:* \n{news}",
                parse_mode=ParseMode.MARKDOWN,
            )

    async def fetch_and_send_news():
        # Fetch news updates for each section (not implemented here)
        news_updates = {
            "physics": "Physics department announces a new research project.",
            "math": "Math department is organizing a workshop on graph theory.",
        }

        for section, news in news_updates.items():
            await send_news_update(section, news)

    async def on_startup(dp):
        await bot.send_message(chat_id=user_id, text="Bot has been started")

        # Periodically fetch and send news updates
        while True:
            await fetch_and_send_news()
            await asyncio.sleep(3600) # Fetch news every hour

    async def on_shutdown(dp):
        await bot.send_message(chat_id=454500983, text="Bot has been stopped")

        await dp.storage.close()
        await dp.storage.wait_closed()

```

```

# Chat

users = {}

@dp.message_handler(commands=['join'])
async def join(message: types.Message):
    user_id = message.from_user.id
    if user_id in users:
        await message.reply("You are already in the chat.")
        return

    if not users:
        users[user_id] = None
        await message.reply("Waiting for a partner...")
    else:
        partner_id = next(iter(users))
        users[user_id] = partner_id
        users[partner_id] = user_id

        await message.reply("You are now connected with a partner.")
        await bot.send_message(partner_id, "You are now connected with a partner.")

@dp.message_handler(lambda message: message.text and not message.text.startswith('/'))
async def forward_message(message: types.Message):
    user_id = message.from_user.id
    partner_id = users.get(user_id)

    if partner_id is None:
        await message.reply("You are not connected with a partner. Type /join to find one.")
    else:
        await bot.send_message(partner_id, message.text)

@dp.message_handler(commands=['leave'])
async def leave(message: types.Message):
    user_id = message.from_user.id
    partner_id = users.get(user_id)

    if partner_id is None:
        await message.reply("You are not connected with a partner.")
    else:
        del users[user_id]
        del users[partner_id]

        await message.reply("You have left the chat.")
        await bot.send_message(partner_id, "Your partner has left the chat.")

if __name__ == '__main__':
    from aiogram import executor
    executor.start_polling(dp, skip_updates=True)

```