

Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра Математичного моделювання та аналізу даних

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

Володимир СТРУКОВ

(ім'я, прізвище)

(підпис)

«__» _____ 2026 року

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

здобувача Марковчиної Дар'ї Леонідівни рівня вищої освіти
(першого (бакалаврського))

(прізвище, ім'я, по батькові здобувача)

Спеціальність (спеціалізація) 122 Комп'ютерні науки
(код та найменування спеціальності; спеціалізації спеціальності)

Освітня програма Комп'ютерні науки
(назва освітньої програми)

1. Тема роботи: «Розробка інформаційної системи генерування тексту
доповненого пошуком»

керівник роботи Турута Олексій Петрович, доцент, заступник директора
навчально-наукового інституту Комп'ютерних наук та штучного інтелекту
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по Університету від «__» _____ 2026 року,
№

2. Строк подання здобувачем роботи: «_1_» _____ червня _____ 2026 року

3. Вихідні дані до роботи:

4. Перелік питань, які потрібно розробити:

- 1) Провести аналіз обраної теми кваліфікаційної роботи;
- 2) Конкретизувати задачі роботи;
- 3) Реалізувати програму;
- 4) Зробити висновки.

5. План роботи

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Проаналізувати сучасні інформаційні системи та чат-боти освітнього призначення	16.03 – 29.03	Виконано
2	Дослідити методи обробки природної мови та пошуку інформації	30.03 – 12.04	Виконано
3	Проаналізувати технології реалізації RAG-систем	13.04 – 19.04	Виконано
4	Розробити архітектуру інформаційної системи	20.04 – 03.05	Виконано
5	Реалізувати Telegram-бота для взаємодії з користувачами	04.05 – 10.05	Виконано
6	Реалізувати механізм пошуку інформації у документах	11.05 – 17.05	Виконано
7	Реалізувати модуль генерації текстових відповідей	18.05 – 24.05	Виконано
8	Провести тестування системи та оцінити результати її роботи	25.05 – 29.05	Виконано
9	Оформлення документації	30.05 – 31.05	Виконано

6. Дата видачі завдання: «__» _____ 2026 року

Здобувач вищої освіти



(підпис)

Дар'я МАРКОВЧИНА

(ім'я, прізвище)

Керівник роботи



(підпис)

Олексій ТУРУТА

(ім'я, прізвище)

Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра Математичного моделювання та аналізу даних

До захисту допущено

кафедрою ММТАД протокол № _____ від _____

завідувач кафедри Володимир СТРУКОВ
(підпис) (ім'я, прізвище)

КВАЛІФІКАЦІЙНА РОБОТА

здобувача Марковчиної Дар'ї Леонідівни рівня вищої освіти
(першого (бакалаврського))

(тема роботи)

Спеціальність (спеціалізація) 122 Комп'ютерні науки
(код та найменування спеціальності; спеціалізації спеціальності)

Освітня програма Комп'ютерні науки
(назва освітньої програми)

Виконавець  Дар'я МАРКОВЧИНА
(підпис) (ім'я, прізвище)

Науковий керівник  Олексій ТУРУТА
(підпис) (ім'я, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота бакалавра присвячена розробці інформаційної системи генерування тексту, доповненого пошуком, для інформаційної підтримки студентів Харківського національного університету імені В. Н. Каразіна. У роботі розглянуто сучасні підходи до побудови інтелектуальних інформаційних систем, методи обробки природної мови та механізми пошуку інформації у текстових документах.

Метою роботи є створення інформаційної системи у вигляді Telegram-бота, який забезпечує пошук необхідної інформації в університетських документах та формує текстові відповіді на основі знайдених даних. Розроблена система орієнтована на автоматизацію інформаційної підтримки студентів та абітурієнтів університету. У процесі виконання роботи було проаналізовано чат-боти та технології Retrieval-Augmented Generation (RAG), досліджено методи класичного та семантичного пошуку інформації, а також розроблено архітектуру програмної системи. Для реалізації програмного забезпечення використано мову програмування Python, бібліотеку aiogram для створення Telegram-бота, а також метод TF-IDF для пошуку релевантних фрагментів тексту.

Система забезпечує приймання запитів користувачів через Telegram, пошук інформації у базі знань, обробку текстових документів та формування відповідей природною мовою. У якості джерел даних використовуються офіційні документи університету, зокрема правила прийому, положення про освітній процес, інформація щодо академічної доброчесності та стипендіального забезпечення. У результаті виконання роботи було створено працездатний програмний продукт, який може бути використаний для автоматизації довідкової підтримки студентів. Практична цінність роботи полягає у спрощенні доступу користувачів до університетської інформації, зменшенні кількості типових звернень та покращенні взаємодії студентів з освітнім середовищем університету.

Ключові слова: інформаційна система, Telegram-бот, штучний інтелект, обробка природної мови, пошук інформації, RAG, Python, FastAPI, PostgreSQL, семантичний пошук, чат-бот.

ABSTRACT

The bachelor's qualification thesis is devoted to the development of an information system for retrieval-augmented text generation aimed at providing informational support to students of V. N. Karazin Kharkiv National University. The thesis examines modern approaches to the development of intelligent information systems, natural language processing methods, and mechanisms for retrieving information from textual documents.

The aim of the thesis is to create an information system in the form of a Telegram bot that enables users to search for the necessary information in university documents and generates textual responses based on the retrieved data. The developed system is focused on automating informational support for students and applicants of the university. In the course of the research, chatbots and Retrieval-Augmented Generation (RAG) technologies were analyzed, methods of classical and semantic information retrieval were studied, and the architecture of the software system was developed. The Python programming language, the aiogram library for creating the Telegram bot, and the TF-IDF method for retrieving relevant text fragments were used to implement the software.

The system enables users to submit queries via Telegram, search for information in the knowledge base, process textual documents, and generate responses in natural language. Official university documents are used as data sources, including admission rules, regulations on the educational process, information on academic integrity, and scholarship provision. As a result of the thesis, a functional software product was developed that can be used to automate reference and informational support for students. The practical value of the thesis lies in simplifying users' access to university information, reducing the number of typical inquiries, and improving students' interaction with the university educational environment.

Keywords: information system, Telegram bot, artificial intelligence, natural language processing, information retrieval, RAG, Python, FastAPI, PostgreSQL, semantic search, chatbot.

ЗМІСТ

РЕФЕРАТ (АНОТАЦІЯ)	5
ЗМІСТ	7
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ	8
ВСТУП	9
1. ЛІТЕРАТУРНИЙ ОГЛЯД (ТЕОРЕТИЧНА ЧАСТИНА)	11
1.1 Інформаційні системи в освіті	11
1.2 Аналіз сучасних чат-ботів та AI-систем	13
1.3 Методи обробки природної мови	16
1.4 Аналіз технологій розробки	19
1.5 Постановка задачі	22
2. ЕКСПЕРИМЕНТАЛЬНА (ПРАКТИЧНА) ЧАСТИНА	25
2.1 Проєктування системи	25
2.2 Розробка бази знань	27
2.3 Реалізація механізму пошуку	29
2.4 Реалізація генерації відповідей	30
2.5 Розробка Telegram-бота	32
2.6 Тестування системи	35
2.7 Аналіз результатів	37
3. ОХОРОНА ПРАЦІ	40
3.1 Аналіз умов праці розробника	40
3.2 Вимоги до робочого місця	41
3.3 Висновки до розділу	43
ВИСНОВКИ	45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	47
ДОДАТКИ	50

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Скорочення	Пояснення
AI	Artificial Intelligence (штучний інтелект)
API	Application Programming Interface (інтерфейс програмування застосунків)
NLP	Natural Language Processing (обробка природної мови)
RAG	Retrieval-Augmented Generation (генерація тексту, доповнена пошуком)
SQL	Structured Query Language (мова структурованих запитів)
БД	База даних
СКБД	Система керування базами даних
ПЗ	Програмне забезпечення
UML	Unified Modeling Language (уніфікована мова моделювання)
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
REST	Representational State Transfer
PDF	Portable Document Format
DOCX	Формат текстових документів Microsoft Word
TF-IDF	Term Frequency – Inverse Document Frequency
Telegram Bot API	Інтерфейс взаємодії з Telegram-ботами
FastAPI	Web-фреймворк для створення API на Python
PostgreSQL	Реляційна система керування базами даних
pgvector	Розширення PostgreSQL для векторного пошуку
Embeddings	Векторне представлення текстових даних
Full Text Search	Повнотекстовий пошук
Chunk	Фрагмент тексту документа
Cosine Similarity	Косинусна схожість
Docker	Платформа контейнеризації програм
Git	Система контролю версій
OpenAI API	API для використання мовних моделей OpenAI
UX	User Experience (досвід користувача)
UI	User Interface (інтерфейс користувача)
ХНУ	Харківський національний університет
ЄКТС	Європейська кредитно-трансферна система

ВСТУП

Сучасний розвиток інформаційних технологій та систем штучного інтелекту суттєво впливає на всі сфери діяльності людини, зокрема й на освітнє середовище. Заклади вищої освіти активно впроваджують цифрові сервіси, електронні ресурси та автоматизовані системи підтримки користувачів. Одним із перспективних напрямів розвитку освітніх інформаційних систем є використання чат-ботів та технологій обробки природної мови для автоматизації пошуку інформації та підтримки студентів.

У процесі навчання студенти постійно працюють із великим обсягом інформації: правилами прийому, положеннями про освітній процес, силабусами дисциплін, графіками навчального процесу, інформацією про академічну доброчесність, стипендіальне забезпечення та іншими нормативними документами. Значна частина цієї інформації є доступною на офіційних ресурсах університету, проте її пошук часто потребує значного часу та ускладнюється великою кількістю документів і складною структурою сайтів.

У зв'язку з цим актуальним є створення інформаційної системи, здатної автоматизувати пошук необхідної інформації та формування зрозумілих відповідей природною мовою. Особливу увагу сьогодні привертають системи Retrieval-Augmented Generation (RAG), які поєднують механізми пошуку інформації та генерації тексту за допомогою моделей штучного інтелекту. Такі системи дозволяють працювати з локальними документами, забезпечувати актуальність відповідей та зменшувати ризик генерації недостовірної інформації.

Об'єктом дослідження є процес інформаційної підтримки студентів закладу вищої освіти із застосуванням технологій штучного інтелекту та пошуку інформації.

Предметом дослідження є методи та програмні засоби реалізації інформаційної системи генерування тексту, доповненого пошуком, для роботи з університетськими документами.

Метою роботи є розробка інформаційної системи у вигляді Telegram-бота, яка забезпечує пошук релевантної інформації у текстових документах університету та формування відповідей природною мовою на основі знайдених даних.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати сучасні інформаційні системи та чат-боти освітнього призначення;
- дослідити методи обробки природної мови та пошуку інформації;
- проаналізувати технології реалізації RAG-систем;
- розробити архітектуру інформаційної системи;
- реалізувати Telegram-бота для взаємодії з користувачами;
- реалізувати механізм пошуку інформації у документах;
- реалізувати модуль генерації текстових відповідей;
- провести тестування системи та оцінити результати її роботи.

У процесі виконання роботи використано мову програмування Python, бібліотеку aiogram для створення Telegram-бота, метод TF-IDF для реалізації пошуку інформації, а також сучасні підходи до обробки природної мови.

Практичне значення роботи полягає у створенні програмного засобу, який дозволяє автоматизувати інформаційну підтримку студентів та спростити доступ до навчальної й організаційної інформації університету. Розроблена система може бути використана як основа для подальшого розвитку інтелектуальних сервісів підтримки студентів у закладах вищої освіти.

1. ЛІТЕРАТУРНИЙ ОГЛЯД (ТЕОРЕТИЧНА ЧАСТИНА)

1.1 Інформаційні системи в освіті

Інформаційні системи відіграють важливу роль у цифровій трансформації освітнього середовища. Заклади вищої освіти активно впроваджують електронні сервіси для автоматизації навчального процесу, комунікації між студентами та викладачами, а також зберігання й обробки навчальних матеріалів. Інформаційні системи дозволяють значно підвищити ефективність управління освітнім процесом та забезпечити швидкий доступ до необхідної інформації. В умовах розвитку дистанційного навчання роль таких систем стала особливо важливою. Сучасні університети використовують різноманітні платформи для організації навчання, проведення тестування, обміну файлами та комунікації. Використання інформаційних технологій також сприяє автоматизації адміністративних процесів та зменшенню навантаження на працівників університету.

Одним із головних напрямів розвитку освітніх інформаційних систем є створення інтелектуальних сервісів підтримки користувачів. Такі системи дозволяють студентам отримувати необхідну інформацію у зручній формі без потреби пошуку даних у великій кількості документів. Використання чат-ботів, систем пошуку та технологій штучного інтелекту забезпечує швидке реагування на типові запити користувачів. Інтелектуальні інформаційні системи можуть працювати цілодобово та забезпечувати доступність сервісів незалежно від часу доби. Це особливо актуально для студентів, які навчаються дистанційно або потребують швидкого доступу до навчальної інформації. Такі системи також можуть адаптуватися до потреб користувачів та забезпечувати персоналізовану підтримку [1].

Освітні інформаційні системи мають складну архітектуру та складаються з багатьох функціональних модулів. До основних компонентів належать бази даних, web-інтерфейси, системи авторизації, модулі пошуку інформації та засоби аналітики. Одним із важливих елементів є інтеграція із зовнішніми сервісами та платформами дистанційного навчання. Університети активно використовують

хмарні технології для забезпечення стабільної роботи інформаційних систем. Також важливим аспектом є захист персональних даних користувачів та забезпечення інформаційної безпеки. Розвиток web-технологій та API дозволяє інтегрувати різні освітні сервіси в єдину інформаційну екосистему.

Особливе значення в освітньому середовищі мають системи управління навчанням (LMS). Такі платформи забезпечують зберігання навчальних матеріалів, проведення тестування та організацію взаємодії між викладачами й студентами. Найбільш популярними LMS є Moodle, Google Classroom та Microsoft Teams. Ці системи підтримують дистанційне навчання та дозволяють автоматизувати значну частину освітнього процесу [2]. Крім того, вони забезпечують централізований доступ до навчальних ресурсів. Системи управління навчанням стали невід’ємною частиною сучасної цифрової освіти. Їх використання значно спрощує організацію навчального процесу та контроль успішності студентів.

Важливим напрямом розвитку інформаційних систем є застосування технологій штучного інтелекту. Системи на основі AI здатні аналізувати текстову інформацію, автоматично генерувати відповіді та виконувати семантичний пошук. Це дозволяє створювати інтелектуальні помічники для студентів та викладачів. Такі рішення можуть використовуватися для автоматизації довідкової підтримки та швидкого доступу до нормативних документів. Використання NLP-технологій дозволяє системам розуміти природну мову користувача та формувати більш релевантні відповіді. AI-системи також можуть аналізувати великі обсяги даних та надавати рекомендації. Завдяки цьому підвищується ефективність взаємодії користувачів з освітнім середовищем. Для кращого розуміння сучасних інформаційних систем в освіті доцільно провести їх порівняння за основними характеристиками, таблиця 1.1.

Таблиця 1.1- Порівняльна характеристика

Система	Призначення	Основні можливості	Переваги
Moodle	Дистанційне навчання	Курси, тести, файли	Відкритий код
Microsoft Teams	Комунікація та навчання	Відеозв'язок, чати	Інтеграція з Office
Google Classroom	Організація навчання	Завдання, оцінювання	Простота використання
Telegram Bot	Інформаційна підтримка	Швидкі відповіді	Доступність
AI Chatbot	Інтелектуальна підтримка	NLP, пошук, генерація	Автоматизація

У таблиці наведено порівняння інформаційних систем, що використовуються в освітньому середовищі, аналіз показує, що кожна система має власні особливості та функціональні можливості. LMS-платформи орієнтовані переважно на організацію навчального процесу, тоді як чат-боти забезпечують швидкий доступ до інформації. Системи на основі штучного інтелекту дозволяють автоматизувати підтримку користувачів та працювати з природною мовою [3]. Telegram-боти є зручними завдяки простому інтерфейсу та широкому поширенню месенджера. Поєднання чат-ботів із технологіями AI відкриває нові можливості для створення освітніх сервісів. Саме тому використання таких технологій є перспективним напрямом розвитку інформаційних систем університетів.

1.2 Аналіз сучасних чат-ботів та AI-систем

Чат-боти активно використовуються у різних сферах діяльності людини, включаючи освіту, медицину, електронну комерцію та банківську сферу. Основним призначенням чат-ботів є автоматизація взаємодії з користувачами та швидке надання інформації. Завдяки розвитку технологій штучного інтелекту чат-боти можуть працювати з природною мовою та аналізувати зміст запитів. Це дозволяє значно підвищити якість обслуговування користувачів. У сфері освіти чат-боти використовуються для довідкової підтримки студентів, організації навчального процесу та автоматизації консультацій [4]. Більшість сучасних ботів

працюють через месенджери або web-інтерфейси. Це забезпечує доступність сервісів з будь-якого пристрою.

Одним із головних напрямів розвитку чат-ботів є використання NLP-технологій. Методи обробки природної мови дозволяють системам розуміти зміст повідомлень та формувати відповіді, максимально наближені до природного мовлення. Раніше більшість ботів працювали за принципом заздалегідь визначених сценаріїв. Сучасні AI-системи здатні аналізувати контекст запиту та використовувати великі мовні моделі для генерації тексту. Це значно підвищує ефективність взаємодії з користувачами. Використання AI також дозволяє реалізувати семантичний пошук та обробку великих обсягів текстової інформації. Такі можливості особливо важливі для освітніх інформаційних систем.

Популярними прикладами AI-систем є ChatGPT, Gemini, Claude та Microsoft Copilot. Вони використовують великі мовні моделі та здатні генерувати текстові відповіді природною мовою. Такі системи можуть використовуватися для навчання, аналізу тексту, програмування та автоматизації різних процесів. Основною перевагою AI-систем є здатність працювати із великими обсягами інформації. Водночас існує проблема генерації недостовірних або вигаданих відповідей [5]. Саме тому актуальним є використання технології Retrieval-Augmented Generation, яка дозволяє поєднати пошук інформації та генерацію тексту. Такий підхід значно підвищує достовірність відповідей.

У сфері освіти чат-боти використовуються для автоматизації підтримки студентів та абітурієнтів. Вони можуть відповідати на питання щодо розкладу занять, умов вступу, академічної доброчесності, стипендій та інших аспектів навчального процесу. Це дозволяє зменшити навантаження на працівників університету та покращити доступ студентів до інформації. Боти також можуть інтегруватися з базами даних та інформаційними системами університету. Однією з головних переваг чат-ботів є цілодобова доступність. Крім того, вони забезпечують швидке отримання відповідей на типові питання. Саме тому використання чат-ботів у закладах вищої освіти є перспективним напрямом розвитку цифрових сервісів.

AI-системи мають складну архітектуру та використовують різноманітні технології машинного навчання. Важливим елементом є використання embeddings та vector search для семантичного пошуку інформації. Це дозволяє системі знаходити релевантні відповіді навіть за відсутності точного збігу ключових слів. Також активно використовуються нейронні мережі та трансформерні моделі [6]. Вони забезпечують високу якість генерації тексту та підтримують роботу з природною мовою. Однак такі системи потребують значних обчислювальних ресурсів. Тому при розробці практичних рішень важливо враховувати баланс між функціональністю та складністю реалізації. Для порівняння AI-систем та чат-ботів доцільно розглянути їх основні характеристики, таблиця 1.2.

Таблиця 1.2-Основні характеристики чат ботів

Система	Тип	Основні можливості	Недоліки
ChatGPT	AI chatbot	Генерація тексту	Можливі галюцинації
Gemini	AI assistant	Робота з текстом та файлами	Високі вимоги
Claude	AI assistant	Аналіз великих текстів	Обмежений доступ
Telegram Bot	Інформаційний бот	Простота інтеграції	Обмежений функціонал
RAG-система	AI + Search	Пошук та генерація	Складність реалізації

Наведена таблиця демонструє особливості чат-ботів та AI-систем. Аналіз показує, що класичні Telegram-боти є простими у реалізації, проте мають обмежені можливості обробки природної мови. AI-системи забезпечують значно вищу якість генерації тексту та підтримують роботу з великими обсягами даних. Водночас вони можуть генерувати недостовірну інформацію. Саме тому перспективним напрямом є використання RAG-систем, які поєднують пошук інформації та генерацію відповідей. Такий підхід дозволяє працювати з локальними документами та підвищувати релевантність відповідей. Використання RAG-технологій є доцільним для створення освітніх інформаційних систем підтримки студентів.

1.3 Методи обробки природної мови

Обробка природної мови (Natural Language Processing, NLP) є одним із ключових напрямів сучасного штучного інтелекту. Основною метою NLP є забезпечення взаємодії між людиною та комп'ютерною системою за допомогою природної мови [7]. У сучасних інформаційних системах методи NLP використовуються для аналізу тексту, класифікації документів, пошуку інформації та генерації відповідей. Особливо важливим є застосування таких технологій у чат-ботах та інтелектуальних системах підтримки користувачів. Методи обробки природної мови дозволяють системі розуміти зміст запитів користувача та формувати релевантні відповіді. Це значно покращує якість взаємодії людини з інформаційною системою. У сфері освіти NLP використовується для автоматизації роботи з навчальними матеріалами та нормативними документами.

Одним із базових методів NLP є токенізація тексту. Токенізація полягає у розбитті тексту на окремі слова, символи або речення. Цей етап є необхідним для подальшої обробки текстових даних та аналізу структури повідомлень. Після токенізації часто виконується нормалізація тексту, яка передбачає приведення слів до нижнього регістру та видалення службових символів. Також можуть використовуватися методи лематизації та стемінгу для приведення слів до базової форми. Це дозволяє зменшити кількість різних словоформ та підвищити якість пошуку інформації. Нормалізація тексту є важливим етапом у більшості систем пошуку та аналізу тексту. Для реалізації пошуку інформації у текстових документах широко використовується метод TF-IDF. Даний підхід дозволяє визначати важливість слів у документі відносно всієї колекції текстів. Метод TF-IDF базується на аналізі частоти входження термінів та використовується для побудови векторного представлення тексту. Після перетворення тексту у вектори система може виконувати пошук релевантних фрагментів за допомогою математичних методів. Одним із найбільш поширених способів оцінки схожості є cosine similarity [8]. Такий підхід дозволяє знаходити документи, найбільш схожі до запиту користувача. Використання TF-IDF є ефективним рішенням для побудови простих інформаційно-пошукових систем.

NLP-системи також активно використовують embeddings та нейронні мережі. Embeddings являють собою векторне представлення тексту, яке дозволяє враховувати зміст слів та контекст їх використання. На відміну від класичних методів, embeddings забезпечують семантичний пошук та дозволяють знаходити схожі за змістом фрагменти тексту. Для побудови embeddings використовуються трансформерні моделі та нейронні мережі. Найбільш популярними є моделі BERT, MiniLM та Sentence Transformers. Такі технології дозволяють реалізувати Retrieval-Augmented Generation системи. Семантичний пошук значно підвищує якість інформаційної підтримки користувачів. У AI-системах важливе місце займає генерація тексту природною мовою. Для цього використовуються великі мовні моделі, здатні аналізувати контекст та формувати логічно пов'язані відповіді. Генерація тексту дозволяє створювати інтелектуальних помічників та автоматизувати консультаційну підтримку. Однак використання генеративних моделей може призводити до появи недостовірної інформації [9]. Для вирішення цієї проблеми застосовується RAG-підхід, який поєднує пошук інформації та генерацію тексту. У такому випадку модель формує відповідь лише на основі знайдених фрагментів документів. Це дозволяє значно підвищити достовірність та релевантність відповідей системи. Для наочного представлення процесу обробки природної мови доцільно використати UML-схему алгоритму роботи NLP-модуля, рис.1.1.

Наведена UML-схема демонструє основні етапи обробки природної мови в інформаційній системі. Спочатку система отримує текстовий запит користувача та виконує його попередню обробку. Після цього здійснюється токенизація та нормалізація тексту для підготовки даних до аналізу. Далі формується векторне представлення тексту та виконується пошук релевантної інформації у базі знань [10]. Для оцінки схожості використовуються математичні методи та алгоритми NLP. На основі знайдених фрагментів система формує відповідь природною мовою. Завершальним етапом є надсилання відповіді користувачу через Telegram-бота.



Рисунок 1.1- Алгоритм роботи NLP-модуля

1.4 Аналіз технологій розробки

Під час розробки сучасних інформаційних систем важливим етапом є вибір технологій та програмних засобів реалізації. Від правильного вибору технологічного стеку залежить продуктивність, масштабованість та ефективність роботи програмного забезпечення. Для створення інформаційної системи генерування тексту, доповненого пошуком, необхідно використовувати інструменти, які підтримують роботу з web-технологіями, базами даних та методами обробки природної мови. Особливу увагу слід приділити технологіям створення Telegram-ботів та реалізації серверної частини системи. Також важливими є засоби для реалізації пошуку інформації та генерації текстових відповідей. Python-бібліотеки забезпечують широкий набір інструментів для

створення AI-систем. Саме тому Python є однією з найбільш популярних мов програмування у сфері штучного інтелекту [11].

Основною мовою програмування у даному проєкті обрано Python. Дана мова програмування має простий синтаксис, велику кількість бібліотек та підтримує роботу з AI-технологіями. Python широко використовується для створення чат-ботів, web-сервісів та систем машинного навчання. Однією з головних переваг є наявність великої екосистеми бібліотек для NLP та роботи з текстовими даними. Також Python підтримує асинхронне програмування, що є важливим для серверних застосунків та Telegram-ботів. Крім того, мова має велику спільноту розробників та значну кількість готових рішень. Це дозволяє прискорити процес створення програмного забезпечення. Для реалізації Telegram-бота використовується бібліотека `aioogram`. Даний фреймворк підтримує асинхронну обробку повідомлень та забезпечує зручну інтеграцію з Telegram Bot API. Використання `aioogram` дозволяє створювати боти з підтримкою команд, меню та обробки текстових повідомлень. Бібліотека має просту структуру та добре підходить для реалізації інформаційних систем підтримки користувачів [12]. Асинхронний підхід дозволяє ефективно працювати з великою кількістю запитів. Також `aioogram` підтримує інтеграцію із зовнішніми сервісами та API. Це робить бібліотеку зручним інструментом для реалізації Telegram-ботів.

Для серверної частини системи доцільно використовувати FastAPI. Даний web-фреймворк дозволяє швидко створювати REST API та підтримує асинхронне програмування. FastAPI забезпечує високу продуктивність та автоматичну генерацію документації API. Це значно спрощує розробку серверної логіки та інтеграцію компонентів системи. Також FastAPI підтримує роботу з JSON та web-технологіями. Використання такого фреймворку дозволяє створити масштабовану серверну архітектуру. Завдяки цьому система може бути легко розширена у майбутньому. Для реалізації пошуку інформації та роботи з текстовими даними використовуються бібліотеки `scikit-learn` та `sentence-transformers` [13]. Бібліотека `scikit-learn` дозволяє реалізувати TF-IDF та cosine similarity для класичного пошуку інформації. `Sentence-transformers`

використовується для побудови embeddings та реалізації семантичного пошуку. Для зберігання інформації використовується PostgreSQL — сучасна реляційна система керування базами даних. У майбутньому система може бути розширена за допомогою pgvector для підтримки vector search. Таке поєднання технологій дозволяє створити сучасну RAG-систему. Використання відкритих технологій також спрощує розгортання та підтримку програмного забезпечення. Для наочного представлення взаємодії технологій доцільно використати UML-схему архітектури системи, рис.1.2.

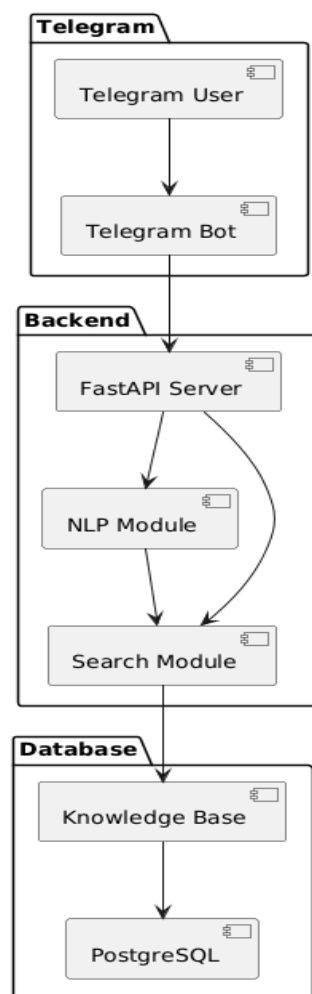


Рисунок 1.2-Архітектура системи

Наведена UML-схема демонструє загальну архітектуру інформаційної системи та взаємодію її компонентів. Користувач взаємодіє із системою через Telegram-бота, який приймає повідомлення та передає їх на сервер. Серверна частина, реалізована за допомогою FastAPI, виконує обробку запиту та взаємодіє

з модулями NLP і пошуку інформації. Модуль пошуку працює з базою знань та PostgreSQL для отримання релевантних фрагментів тексту. Після аналізу знайдених даних NLP-модуль формує відповідь природною мовою. Завдяки такій архітектурі система забезпечує ефективну взаємодію між усіма компонентами. Використання сучасних технологій дозволяє створити масштабовану та продуктивну інформаційну систему.

1.5 Постановка задачі

У закладах вищої освіти значна частина інформації для студентів та абітурієнтів зберігається у вигляді текстових документів. До таких документів належать правила прийому, освітні програми, силабуси дисциплін, положення про освітній процес, графіки навчання та інші нормативні матеріали. Попри відкритий доступ до інформації, користувачі часто стикаються з труднощами під час пошуку необхідних даних [14]. Основною проблемою є великий обсяг документів та складність навігації між ними. Студентам доводиться витратити багато часу на пошук потрібної інформації вручну. Це знижує ефективність взаємодії з освітнім середовищем та створює додаткове навантаження на працівників університету. Саме тому виникає потреба у створенні автоматизованої системи інформаційної підтримки.

Одним із перспективних напрямів вирішення даної проблеми є використання чат-ботів та технологій штучного інтелекту. Сучасні інформаційні системи здатні аналізувати текстові документи, виконувати пошук релевантної інформації та генерувати відповіді природною мовою. Використання Telegram-ботів забезпечує простий та швидкий доступ користувачів до інформаційних сервісів університету. Такі системи дозволяють автоматизувати обробку типових запитів та забезпечити цілодобову підтримку студентів. Крім того, застосування технологій NLP та RAG дозволяє підвищити якість пошуку інформації та зробити відповіді більш релевантними. Використання сучасних AI-технологій також дозволяє працювати з великими обсягами текстових даних. Це робить розробку подібної системи актуальною та перспективною задачею.

Основною метою даної роботи є розробка інформаційної системи генерування тексту, доповненого пошуком, для інформаційної підтримки студентів Харківського національного університету імені В. Н. Каразіна. Система повинна забезпечувати приймання текстових запитів через Telegram, пошук інформації у базі знань та формування зрозумілих текстових відповідей. Як джерело інформації планується використовувати офіційні документи університету [15]. До них належать правила прийому, документи щодо академічної доброчесності, положення про стипендії, силабуси дисциплін та інші навчальні матеріали. Система повинна працювати з текстовими файлами та забезпечувати швидкий доступ до інформації. Важливою вимогою є можливість подальшого розширення бази знань новими документами. Для досягнення поставленої мети необхідно вирішити ряд технічних та програмних задач. Першочерговим завданням є створення Telegram-бота для взаємодії з користувачами. Далі необхідно реалізувати механізм обробки текстових документів та формування бази знань. Також потрібно реалізувати модуль пошуку релевантної інформації на основі методів NLP та TF-IDF. Окремою задачею є реалізація генерації текстових відповідей на основі знайдених фрагментів документів. Крім того, необхідно забезпечити можливість оновлення бази знань та додавання нових документів. Важливим етапом є тестування системи та оцінка ефективності її роботи.

У процесі розробки системи планується використати сучасні програмні технології та інструменти. Основною мовою програмування буде Python, оскільки вона широко використовується у сфері штучного інтелекту та web-розробки. Для реалізації Telegram-бота буде використано бібліотеку aiogram, а для серверної частини — FastAPI. Для роботи з текстовими даними та реалізації пошуку використовуватимуться бібліотеки scikit-learn та sentence-transformers. Зберігання інформації планується реалізувати за допомогою PostgreSQL. У перспективі система може бути розширена за рахунок використання vector search та embeddings. Це дозволить реалізувати семантичний пошук та підвищити релевантність відповідей [16]. Результатом виконання роботи повинна стати працездатна інформаційна система підтримки студентів у вигляді Telegram-бота.

Система має забезпечувати швидкий пошук інформації в університетських документах та формування відповідей природною мовою. Очікується, що використання такого рішення дозволить зменшити навантаження на працівників університету та спростити доступ студентів до навчальної інформації. Розроблена система також може бути використана як основа для створення більш складних AI-сервісів у сфері освіти. Практична цінність роботи полягає у можливості реального використання програмного продукту в освітньому середовищі. Використання сучасних технологій забезпечує перспективність подальшого розвитку системи. Таким чином, поставлена задача є актуальною та має важливе практичне значення для цифровізації освітнього процесу.

2. ЕКСПЕРИМЕНТАЛЬНА (ПРАКТИЧНА) ЧАСТИНА

2.1 Проєктування системи

Проєктування інформаційної системи є одним із найважливіших етапів розробки програмного забезпечення. На цьому етапі визначається загальна архітектура системи, взаємодія між її компонентами та принципи обробки даних. Для розробки системи генерування тексту, доповненого пошуком, було обрано клієнт-серверну архітектуру. Такий підхід дозволяє розділити логіку взаємодії з користувачем та серверну обробку інформації. Основним клієнтським інтерфейсом системи виступає Telegram-бот, через який користувач надсилає запити. Серверна частина забезпечує обробку тексту, пошук інформації та генерацію відповідей. Використання сучасної архітектури дозволяє забезпечити масштабованість та зручність подальшого розвитку системи [17].

Основним завданням проєктування було забезпечення ефективної взаємодії між модулями системи. Telegram-бот приймає повідомлення користувача та передає їх серверній частині. Сервер виконує аналіз тексту запиту та запускає механізм пошуку інформації. Після знаходження релевантних фрагментів система формує відповідь та надсилає її користувачу. Для зберігання документів та фрагментів тексту використовується база знань. Також система підтримує можливість оновлення документів та поповнення бази даних. Це забезпечує актуальність інформації та зручність адміністрування.

Важливим елементом архітектури є модуль NLP, який відповідає за обробку природної мови. Даний модуль виконує токенізацію тексту, нормалізацію та формування векторного представлення запиту. Після цього виконується пошук найбільш релевантних фрагментів тексту. Для реалізації пошуку використовується TF-IDF та cosine similarity. Такий підхід дозволяє швидко знаходити інформацію у текстових документах. Використання NLP забезпечує більш якісну взаємодію користувача із системою. Це також дозволяє реалізувати основу для подальшого впровадження AI-технологій.

У процесі проєктування було враховано можливість подальшого розширення системи [18]. Архітектура дозволяє інтегрувати PostgreSQL, FastAPI та векторний пошук. Також можлива інтеграція з OpenAI API або локальними мовними моделями. Використання модульного підходу спрощує додавання нових функціональних можливостей. Система може бути адаптована для роботи з PDF-документами та web-інтерфейсом. Крім того, архітектура дозволяє реалізувати багатокористувацький режим роботи. Це робить систему перспективною для подальшого розвитку. Для наочного представлення архітектури системи доцільно використати UML-схему, рис.2.1.

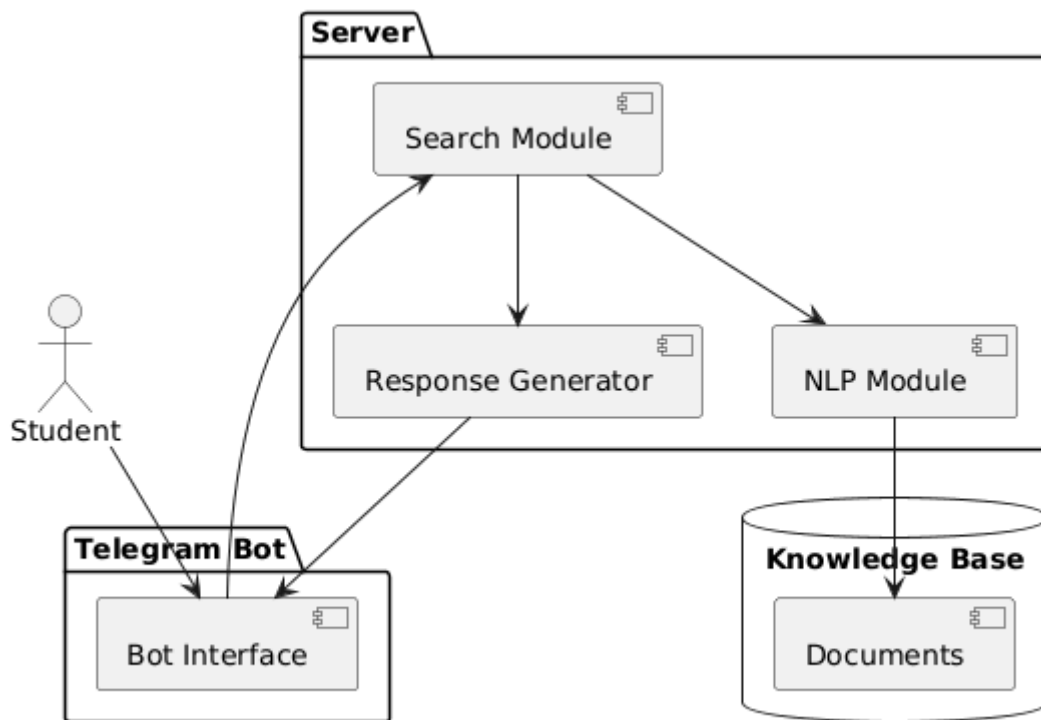


Рисунок 2.1. UML-схем архітектури системи

Наведена UML-схема демонструє взаємодію основних компонентів системи. Користувач взаємодіє із Telegram-ботом, який приймає текстові повідомлення та передає їх серверній частині. Сервер містить модуль пошуку, NLP-модуль та генератор відповідей. Пошуковий модуль працює із базою знань та знаходить релевантні фрагменти тексту. NLP-модуль виконує аналіз запиту та обробку природної мови. Генератор відповідей формує фінальний текст для

користувача. Така архітектура забезпечує ефективну роботу системи та можливість її подальшого масштабування.

2.2 Розробка бази знань

База знань є одним із ключових компонентів інформаційної системи. Саме вона забезпечує зберігання документів, фрагментів тексту та інформації, необхідної для формування відповідей користувачам. У рамках даного проєкту база знань реалізована у вигляді набору текстових документів, які містять нормативну та навчальну інформацію університету [19]. До таких документів належать правила прийому, положення про академічну доброчесність, інформація про стипендії та силабуси дисциплін. Система автоматично обробляє документи та розбиває текст на окремі фрагменти. Це дозволяє виконувати пошук інформації значно швидше та ефективніше. Використання структурованої бази знань є необхідною умовою реалізації RAG-системи. У процесі розробки бази знань було реалізовано механізм обробки текстових файлів. Система автоматично зчитує всі документи з папки data та виконує попередню обробку тексту. Після цього документи розбиваються на окремі fragments або chunks. Кожен фрагмент має обмежений розмір та використовується як окремий елемент для пошуку. Такий підхід дозволяє знаходити найбільш релевантні частини документів. Розбиття тексту також зменшує навантаження на систему під час обробки запитів. Це позитивно впливає на швидкість роботи інформаційної системи.

Одним із важливих аспектів розробки бази знань є підтримка можливості її оновлення. Користувач або адміністратор може додавати нові документи без необхідності зміни програмного коду [20]. Для цього достатньо помістити текстовий файл у папку data та виконати команду оновлення бази знань. Після цього система автоматично виконає індексацію нового документа. Це забезпечує гнучкість та зручність використання системи. Такий підхід також дозволяє масштабувати інформаційну систему без складних змін архітектури. У майбутньому база знань може бути розширена підтримкою PDF та DOCX документів.

Для зберігання та пошуку інформації в системі використовується принцип векторного представлення тексту. Кожен fragment документа перетворюється у математичний вектор за допомогою TF-IDF. Це дозволяє порівнювати запит користувача та текстові фрагменти за допомогою cosine similarity. Векторний підхід забезпечує більш ефективний пошук інформації порівняно зі звичайним пошуком ключових слів. Крім того, у перспективі система може використовувати embeddings та pgvector. Це дозволить реалізувати семантичний пошук інформації. Використання методів зберігання текстових даних значно покращує якість роботи системи. Для наочного представлення структури бази знань використано UML-схему, рис.2.2.

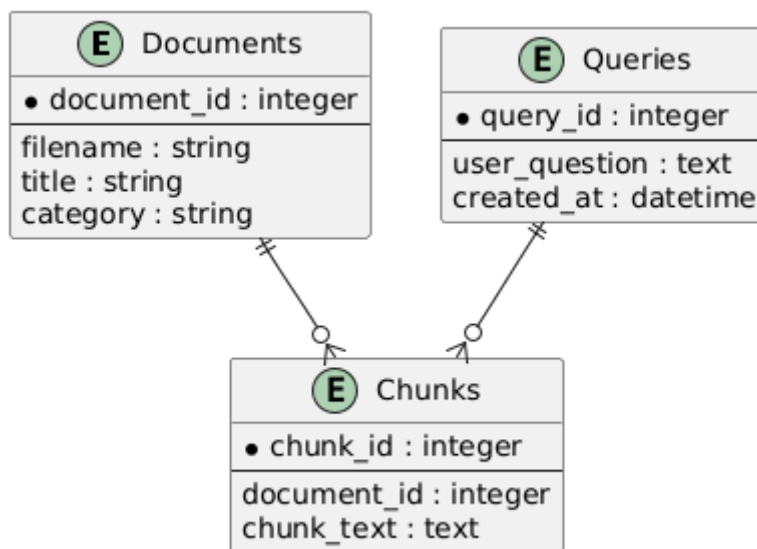


Рисунок 2.2-Структура бази знань

Наведена UML-схема демонструє структуру бази знань інформаційної системи. Основним елементом є таблиця Documents, яка містить інформацію про документи університету. Кожен документ пов'язаний із набором текстових fragments, що зберігаються у таблиці Chunks. Таблиця Queries містить інформацію про запити користувачів. Така структура дозволяє ефективно виконувати пошук релевантної інформації [21]. Взаємозв'язки між таблицями забезпечують логічну організацію даних. Розроблена база знань є основою функціонування всієї інформаційної системи.

2.3 Реалізація механізму пошуку

Реалізація механізму пошуку є одним із ключових етапів створення інформаційної системи. Основним завданням даного модуля є знаходження релевантної інформації у базі знань відповідно до запиту користувача. Для реалізації пошуку було використано метод TF-IDF та cosine similarity. Такий підхід дозволяє виконувати швидкий пошук текстових fragments та визначати ступінь схожості між запитом і документами. Механізм пошуку працює з попередньо підготовленими fragments тексту. Це значно покращує швидкість роботи системи. Реалізований підхід є ефективним для невеликих та середніх баз знань. Під час роботи системи текстові документи автоматично розбиваються на fragments. Кожен fragment зберігається як окремий елемент для подальшого пошуку. Після отримання запиту користувача система формує його векторне представлення за допомогою TF-IDF. Далі виконується обчислення cosine similarity між запитом та всіма fragments бази знань. Найбільш релевантні fragments повертаються як результат пошуку. Це дозволяє системі швидко знаходити необхідну інформацію. Такий підхід є достатньо простим та ефективним для практичного використання.

Використання TF-IDF забезпечує визначення важливості слів у документі. Cosine similarity використовується для оцінки схожості текстових векторів. Чим більшим є значення cosine similarity, тим більш релевантним є fragment документа. Реалізація даного механізму не потребує великих обчислювальних ресурсів. Це дозволяє використовувати систему навіть на звичайному персональному комп'ютері. Крім того, такий підхід легко масштабувати та розширювати. У майбутньому механізм пошуку може бути доповнений embeddings та vector search.

Для реалізації механізму пошуку використано бібліотеку scikit-learn. Вона забезпечує готові інструменти для роботи з TF-IDF та cosine similarity. Основний алгоритм пошуку реалізований у вигляді окремої функції Python. Це забезпечує зручність інтеграції пошукового модуля із Telegram-ботом. Також система підтримує можливість пошуку за декількома documents одночасно. Реалізація механізму пошуку є важливим етапом створення RAG-системи [22]. Саме цей

модуль забезпечує отримання релевантної інформації для генерації відповіді.

Нижче наведено фрагмент коду реалізації механізму пошуку.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def search_relevant_chunks(query, chunks, top_k=3):

    vectorizer = TfidfVectorizer()

    chunk_vectors = vectorizer.fit_transform(chunks)

    query_vector = vectorizer.transform([query])

    similarities = cosine_similarity(
        query_vector,
        chunk_vectors
    ).flatten()

    top_indexes = similarities.argsort()[::-1][:top_k]

    results = []

    for index in top_indexes:
        results.append(chunks[index])

    return results
```

Наведений фрагмент коду демонструє реалізацію механізму пошуку релевантних fragments тексту. Спочатку виконується створення TF-IDF векторів для documents бази знань. Після цього формується векторне представлення запиту користувача. Далі за допомогою cosine similarity виконується обчислення схожості між запитом та fragments тексту. Система визначає найбільш релевантні fragments та повертає їх як результат пошуку. Такий підхід забезпечує швидку та ефективну роботу інформаційної системи. Реалізований механізм пошуку є основою функціонування Telegram-бота.

2.4 Реалізація генерації відповідей

Реалізація генерації відповідей є важливим етапом розробки інформаційної системи, оскільки саме цей модуль формує кінцевий результат для користувача. Після того як система знаходить релевантні фрагменти тексту у базі знань, необхідно перетворити їх у зрозумілу відповідь природною мовою. У межах розробленої системи відповідь формується на основі знайденого фрагмента документа. Якщо підключено зовнішню AI-модель, система може додатково переформулювати знайдену інформацію у більш зручний для користувача вигляд. Такий підхід дозволяє поєднати пошук інформації та генерацію тексту.

Це відповідає принципу Retrieval-Augmented Generation, коли відповідь формується не довільно, а на основі знайдених даних [23].

У розробленій системі генерація відповіді виконується після завершення пошуку. Спочатку користувач надсилає запит через Telegram-бота, після чого система виконує пошук релевантних фрагментів у базі знань. Якщо відповідні фрагменти знайдено, система формує текст відповіді та додає інформацію про джерело. Якщо ж релевантної інформації не знайдено, користувач отримує повідомлення про відсутність даних у базі знань. Такий підхід дозволяє уникнути вигаданих відповідей та підвищити достовірність роботи системи. Відповідь формується українською мовою, що забезпечує зручність використання для студентів. Крім того, користувач може бачити, з якого документа була отримана інформація. Модуль генерації відповідей може працювати у двох режимах. Перший режим передбачає просте повернення знайденого фрагмента тексту з бази знань. Цей варіант є надійним, оскільки система не змінює зміст знайденої інформації. Другий режим передбачає використання мовної моделі для формування більш природної та структурованої відповіді. У такому випадку знайдені фрагменти передаються до моделі як контекст. Модель не повинна вигадувати інформацію, а має відповідати лише на основі знайдених даних. Це дозволяє підвищити якість відповідей та зробити взаємодію з ботом більш зручною.

Важливою вимогою до генерації відповідей є контроль достовірності. Система повинна працювати тільки з тією інформацією, яка є у базі знань. Це особливо важливо для освітніх інформаційних систем, оскільки неправильна відповідь може ввести студента або абітурієнта в оману. Тому у програмній реалізації передбачено перевірку наявності знайдених фрагментів. Якщо база знань не містить відповіді на питання, система повідомляє користувача про відсутність точної інформації. Такий підхід робить роботу системи більш прозорою та безпечною. Крім того, він дозволяє адміністратору зрозуміти, які документи потрібно додати до бази знань.

Нижче наведено фрагмент програмного коду, який реалізує формування відповіді на основі знайдених фрагментів тексту.

```
def generate_answer(question, found_chunks):
    context = "\n\n".join(
        [f"Джерело: {item['source']}\n{item['text']}" for item in found_chunks]
    )

    if not context:
        return "На жаль, я не знайшов відповідної інформації в базі знань."

    return (
        "Я знайшов таку інформацію:\n\n"
        f"{found_chunks[0]['text']}\n\n"
        f"Джерело: {found_chunks[0]['source']}"
    )
```

Наведений фрагмент коду демонструє базовий механізм генерації відповіді. Функція отримує питання користувача та список знайдених фрагментів тексту. Якщо релевантних фрагментів не знайдено, система повертає повідомлення про відсутність інформації [24]. Якщо результат пошуку існує, бот формує відповідь на основі першого найбільш релевантного фрагмента. Також до відповіді додається назва джерела, з якого було отримано інформацію. Такий підхід дозволяє забезпечити просту, зрозумілу та контрольовану генерацію відповідей. У подальшому цей модуль може бути розширений підключенням мовної моделі для покращення стилю відповіді.

2.5 Розробка Telegram-бота

Telegram-бот є основним інтерфейсом взаємодії користувача з інформаційною системою. Саме через нього студент або абітурієнт надсилає питання та отримує відповідь. Вибір Telegram як інтерфейсу є доцільним, оскільки цей месенджер є поширеним, зручним та доступним на різних пристроях. Користувачу не потрібно встановлювати окрему програму або переходити на спеціальний сайт. Достатньо відкрити Telegram, знайти бота та написати запитання. Це значно спрощує використання системи та робить її доступною для широкого кола користувачів.

Створення Telegram-бота виконується за допомогою сервісу BotFather. Спочатку у Telegram потрібно знайти офіційного бота BotFather та виконати команду /newbot. Після цього необхідно вказати назву бота та його унікальний username. Після створення BotFather надає спеціальний токен доступу. Цей токен використовується у програмному коді для підключення розробленої системи до Telegram Bot API. Саме завдяки токenu програма може приймати повідомлення

користувачів та надсилати відповіді. На рисунку можна показати етап створення бота через BotFather та отримання токена, рис. 2.3.

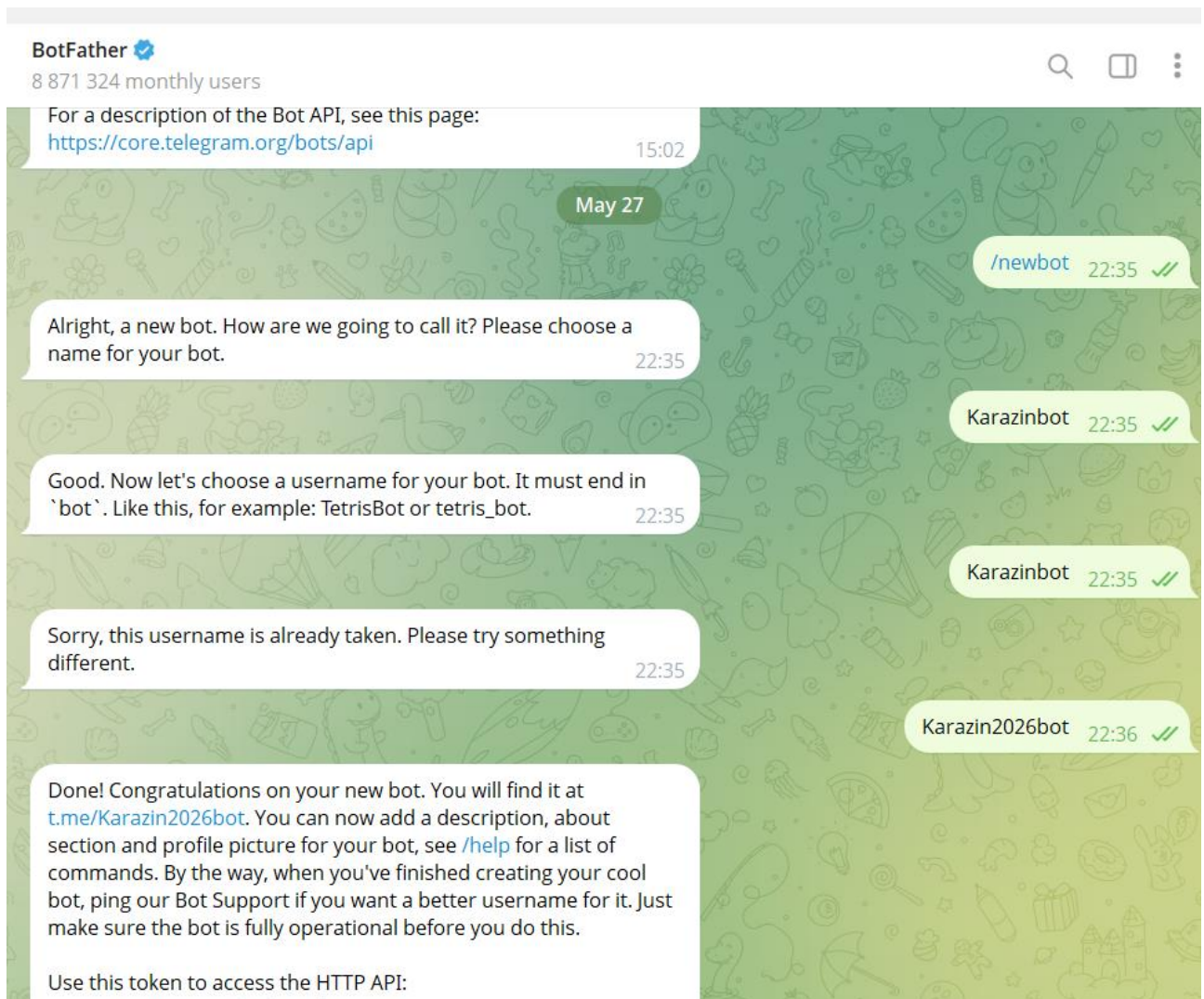


Рисунок 2.3- Вікно створення чат боту

Після отримання токена він додається у файл `.env`, щоб не зберігати його безпосередньо у програмному коді. Це є правильною практикою з точки зору безпеки та організації проєкту. У файлі `.env` створюється змінна `BOT_TOKEN`, у яку записується токен Telegram-бота. Далі програма зчитує цей токен під час запуску та створює об'єкт бота. Для реалізації Telegram-бота використовується бібліотека `aiogram`, яка підтримує асинхронну обробку повідомлень. Це дозволяє боту швидко реагувати на запити користувачів [25]. Також у програмі передбачено обробку команд `/start`, `/help` та `/reload`. Інтерфейс Telegram-бота виглядає як звичайний чат. Після запуску користувач натискає кнопку Start або

вводить команду /start. Бот надсилає привітальне повідомлення та коротко пояснює, які питання можна ставити. Наприклад, користувач може запитати про правила вступу, академічну доброчесність, стипендію або кількість кредитів. Після введення запиту бот повідомляє, що виконує пошук інформації. Потім система повертає знайдений фрагмент тексту та вказує джерело, рис.2.4.

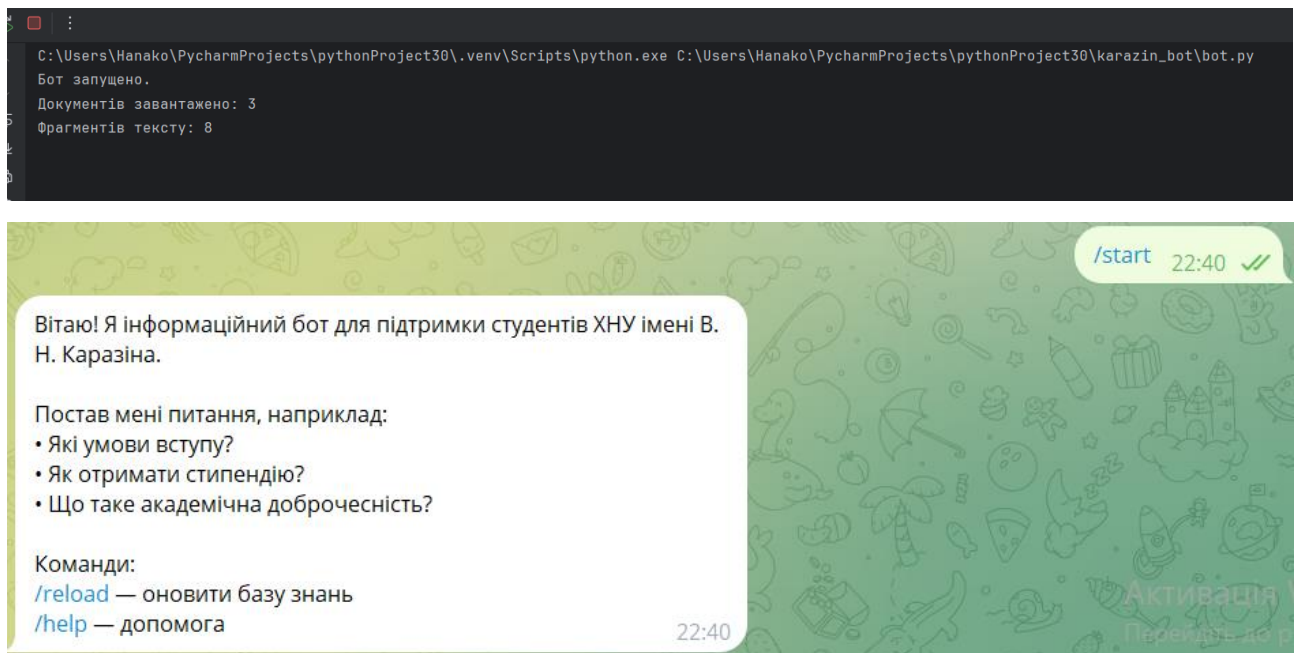


Рисунок 2.4- Чат бот в роботі

Послідовність створення Telegram-бота можна описати такими кроками. Спочатку створюється бот через BotFather та отримується токен доступу. Далі створюється проєкт у середовищі PyCharm і встановлюються необхідні бібліотеки. Після цього у папці проєкту створюється файл `.env`, файл `bot.py`, файл `requirements.txt` та папка `data`. У папку `data` додаються текстові документи бази знань. Після запуску програми бот підключається до Telegram та починає обробляти повідомлення користувачів. Така реалізація дозволяє швидко створити робочий прототип інформаційної системи.

2.6 Тестування системи

Тестування системи є необхідним етапом перевірки правильності її роботи. Метою тестування є визначення того, чи виконує розроблена інформаційна система поставлені функціональні вимоги. У процесі тестування перевіряється

запуск Telegram-бота, обробка команд, пошук інформації у базі знань та формування відповідей. Також перевіряється коректність роботи системи у випадку, коли відповідь на запит відсутня у документах. Це дозволяє оцінити стабільність роботи програми та виявити можливі помилки. Тестування проводилося на локальному комп'ютері у середовищі PyCharm. Для перевірки використовувалися тестові документи, розміщені у папці data.

Першим етапом тестування була перевірка запуску програмного забезпечення. Для цього було створено віртуальне середовище Python та встановлено необхідні бібліотеки. Після запуску файлу bot.py система завантажила документи з папки data та розбила їх на текстові фрагменти. У консолі було відображено повідомлення про успішний запуск бота та кількість завантажених документів. Це підтвердило, що базові компоненти системи працюють правильно. Також було перевірено, що бот підключається до Telegram за допомогою токена. Після цього користувач зміг відкрити чат із ботом та виконати команду /start. Другим етапом тестування була перевірка обробки команд. Команда /start повинна повертати привітальне повідомлення та пояснювати призначення бота. Команда /help використовується для отримання короткої інструкції щодо роботи із системою. Команда /reload призначена для оновлення бази знань після додавання нових документів. Усі команди були протестовані через Telegram-інтерфейс. У результаті перевірки встановлено, що бот коректно реагує на команди та повертає відповідні повідомлення. Це підтверджує правильність реалізації базової логіки взаємодії з користувачем.

Третім етапом тестування була перевірка пошуку інформації. Для цього користувач надсилав боту питання, пов'язані з правилами прийому, стипендіями, академічною доброчесністю та навчальним процесом. Система виконувала пошук релевантних фрагментів у текстовому документі pravila_pryiomu.txt. У більшості випадків бот знаходив відповідний фрагмент та повертав його користувачу. Наприклад, на запит про стипендію система знаходила розділ, у якому описано умови її отримання, рис.2.5.

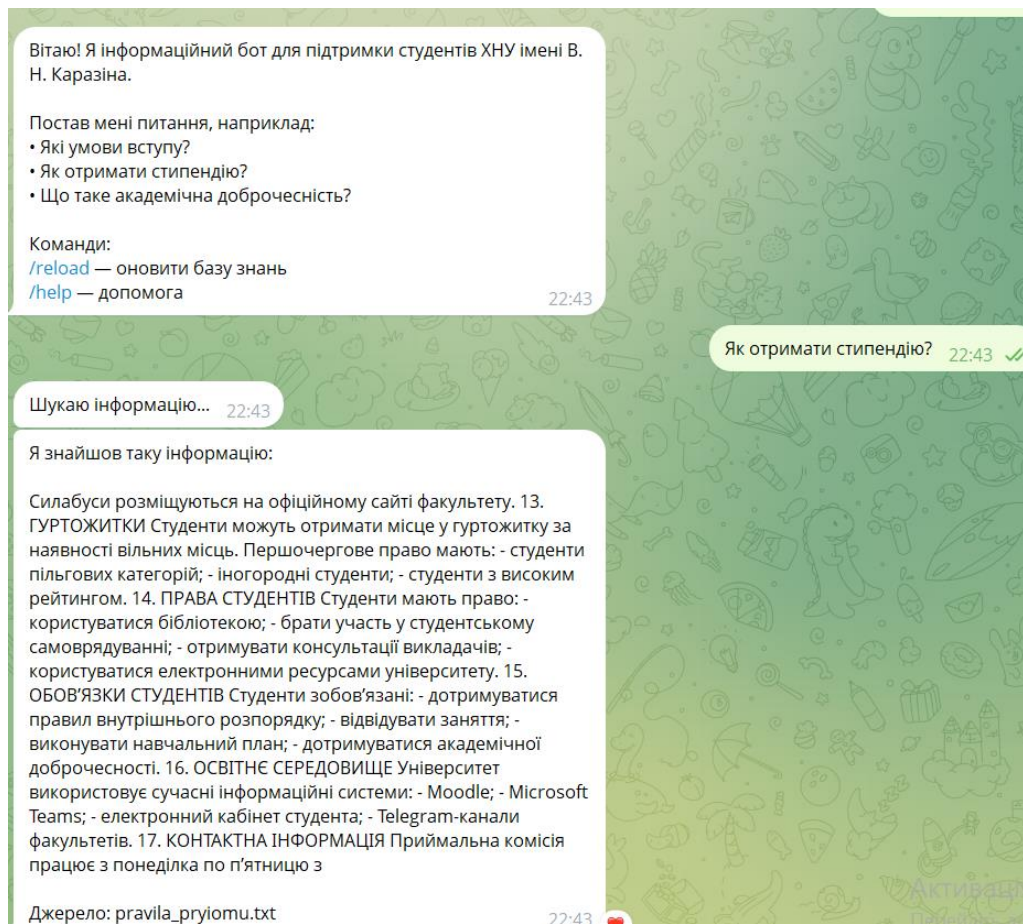


Рисунок 2.5- Приклад запиту до чат боту

На запит про академічну доброчесність бот повертав інформацію про заборонені порушення. Це свідчить про працездатність реалізованого механізму пошуку. Останнім етапом тестування була перевірка роботи системи у нестандартних ситуаціях. До таких ситуацій належать запити, на які в базі знань немає відповіді, порожні повідомлення або неточно сформульовані питання. Якщо система не знаходила релевантних фрагментів, користувач отримував повідомлення про відсутність інформації. Це є важливою особливістю, оскільки бот не повинен вигадувати відповідь. Також було перевірено оновлення бази знань після додавання нового документа. Після виконання команди `/reload` система повторно завантажувала документи та використовувала їх у пошуку. За результатами тестування встановлено, що система працює стабільно та виконує основні функції.

2.7 Аналіз результатів

Після завершення розробки та тестування було проведено аналіз результатів роботи інформаційної системи. Основна увага приділялася перевірці якості пошуку, швидкості відповіді та зручності використання Telegram-бота. Розроблена система показала працездатність під час обробки типових запитів студентів та абітурієнтів. Бот коректно знаходив інформацію у базі знань та повертав користувачу відповідний фрагмент тексту. Особливо добре система працювала з питаннями, які містили ключові слова з документа. Це пояснюється використанням методу TF-IDF, який ефективно працює з текстовими збігами. Отримані результати підтверджують доцільність використання такого підходу для початкової версії системи.

Аналіз роботи системи показав, що головною перевагою є простота використання. Користувач взаємодіє з ботом через звичайний Telegram-чат і не потребує спеціальних технічних знань. Це робить систему зручною для студентів, абітурієнтів та працівників університету. Також перевагою є можливість швидкого оновлення бази знань. Для додавання нової інформації достатньо розмістити текстовий файл у відповідній папці та виконати команду оновлення. Такий підхід дозволяє легко підтримувати актуальність даних. У результаті система може бути використана як основа для автоматизованої довідкової підтримки.

Водночас під час аналізу були виявлені певні обмеження. Метод TF-IDF добре працює з ключовими словами, але гірше обробляє складні або перефразовані запити. Наприклад, якщо користувач ставить питання іншими словами, ніж написано у документі, якість пошуку може знижуватися. Це є типовим обмеженням класичних методів пошуку. Для покращення результатів у майбутньому доцільно додати семантичний пошук на основі embeddings. Також систему можна розширити підтримкою PDF та DOCX документів. Це дозволить використовувати реальні офіційні документи університету без попереднього перетворення у TXT. Для оцінки роботи системи було використано набір тестових запитів. Кожен запит перевіряв окрему функціональну можливість бота. Результати тестування наведено у таблиці 2.1.

Таблиця 2.1-Аналіз тестування

№	Тестовий запит	Очікуваний результат	Фактичний результат	Оцінка
1	Які документи потрібні для вступу?	Пошук розділу про документи	Знайдено перелік документів	Успішно
2	Як отримати стипендію?	Пошук умов стипендії	Знайдено розділ про стипендію	Успішно
3	Що таке академічна доброчесність?	Пошук розділу про доброчесність	Знайдено відповідний фрагмент	Успішно
4	Скільки кредитів має бакалаврська програма?	Пошук інформації про кредити	Знайдено 240 кредитів ЄКТС	Успішно
5	Як отримати гуртожиток?	Пошук інформації про гуртожитки	Знайдено умови поселення	Успішно
6	Який розклад автобуса?	Повідомлення про відсутність інформації	Відповідь не знайдено	Успішно
7	Як перескласти іспит?	Пошук правил перескладання	Знайдено інформацію про перескладання	Успішно

Результати тестування свідчать про те, що система виконує основні функції відповідно до поставлених вимог. Бот успішно приймає запити користувачів, виконує пошук у базі знань та формує відповіді на основі знайденої інформації. У випадках, коли відповідь відсутня у документах, система не генерує неправдиву інформацію, а повідомляє про відсутність даних. Це є важливою перевагою для освітніх систем, де достовірність інформації має велике значення. Загалом результати підтверджують працездатність розробленого програмного продукту. У подальшому система може бути покращена за рахунок впровадження семантичного пошуку, бази даних PostgreSQL та підтримки додаткових форматів документів.

3. ОХОРОНА ПРАЦІ

3.1 Аналіз умов праці розробника

Під час розробки інформаційної системи програміст значну частину робочого часу проводить за персональним комп'ютером. Робота пов'язана з постійним використанням монітора, клавіатури та інших технічних засобів. Такий вид діяльності належить до категорії робіт із високим навантаженням на органи зору та нервову систему людини. Основними факторами, що впливають на умови праці розробника, є освітлення, рівень шуму, температура приміщення та ергономічність робочого місця. Також важливим є режим праці та відпочинку, оскільки тривала робота за комп'ютером може призводити до втоми. Робота програміста потребує високої концентрації уваги та значного інтелектуального навантаження. Саме тому необхідно забезпечити комфортні та безпечні умови праці.

Одним із найбільш важливих факторів є освітлення робочого місця. Недостатнє або надмірне освітлення може спричиняти перевтому очей та погіршення самопочуття працівника. Для комфортної роботи необхідно використовувати природне та штучне освітлення. Робоче місце повинно бути розташоване таким чином, щоб на екран монітора не потрапляли прямі сонячні промені. Також слід уникати появи відблисків на екрані. Для освітлення приміщення доцільно використовувати світлодіодні лампи з нейтральним світлом. Дотримання цих вимог позитивно впливає на працездатність розробника.

Важливим аспектом є мікроклімат приміщення. Температура повітря у робочому кабінеті повинна перебувати в межах 20–24 °С. Відносна вологість повітря має становити приблизно 40–60 %. Також необхідно забезпечити достатню вентиляцію приміщення. Недотримання цих умов може негативно впливати на працездатність та самопочуття працівника. Крім того, важливим фактором є рівень шуму. У приміщенні для роботи програміста не повинно бути

зайвих шумових подразників, які можуть відволікати від роботи. Комфортний мікроклімат дозволяє зменшити втому та підвищити ефективність праці.

Під час роботи за комп'ютером особливе значення має ергономіка робочого місця. Монітор повинен розташовуватися на відстані приблизно 50–70 сантиметрів від очей користувача. Верхня межа екрана повинна знаходитися на рівні очей або трохи нижче. Робоче крісло повинно мати регулювання висоти та підтримку спини. Клавіатура і миша мають розташовуватися таким чином, щоб руки перебували у зручному положенні. Неправильна організація робочого місця може призвести до порушень постави та захворювань опорно-рухового апарату. Тому дотримання ергономічних вимог є важливою умовою безпечної роботи програміста.

Під час розробки програмного забезпечення необхідно дотримуватися правил техніки безпеки та режиму праці. Тривала безперервна робота за комп'ютером може спричиняти перевтому та зниження концентрації уваги. Для запобігання цьому рекомендується робити короткі перерви через кожні 45–60 хвилин роботи. Під час перерв доцільно виконувати вправи для очей та змінювати положення тіла. Також необхідно контролювати правильність підключення електричного обладнання. Робоче місце повинно бути оснащено справними електричними пристроями та захистом від перевантаження мережі. Дотримання правил охорони праці забезпечує безпечні умови роботи та зменшує ризик виникнення професійних захворювань.

3.2 Вимоги до робочого місця

Робоче місце розробника програмного забезпечення повинно відповідати санітарним, ергономічним та технічним вимогам. Основною метою правильної організації робочого місця є створення комфортних умов праці та зменшення негативного впливу комп'ютерної техніки на здоров'я людини. Робоча зона повинна бути достатньо просторою для розміщення персонального комп'ютера, монітора, клавіатури та іншого обладнання. Важливо забезпечити зручний доступ до всіх необхідних пристроїв. Також необхідно враховувати освітлення та вентиляцію приміщення. Організація робочого місця безпосередньо впливає

на продуктивність праці програміста. Саме тому дотримання встановлених вимог є обов'язковим.

Однією з основних вимог є правильне розташування монітора. Екран повинен бути встановлений таким чином, щоб користувач міг працювати без зайвого напруження очей. Рекомендована відстань від очей до монітора становить приблизно 50–70 сантиметрів. Також важливо правильно налаштувати яскравість та контрастність екрана. Монітор не повинен створювати відблиски або мерехтіння. Робоче місце необхідно розташовувати перпендикулярно до вікна для уникнення прямого потрапляння світла на екран. Виконання цих вимог дозволяє зменшити втому очей та покращити умови роботи.

Робочий стіл та крісло повинні відповідати ергономічним вимогам. Висота столу має забезпечувати правильне положення рук під час роботи з клавіатурою. Робоче крісло повинно бути регульованим та підтримувати правильне положення спини. Бажано використовувати крісла з анатомічною спинкою та підлокітниками. Ноги користувача повинні вільно розташовуватися на підлозі або спеціальній підставці. Неправильна поза під час роботи може призводити до болю у спині та порушень опорно-рухового апарату. Тому ергономіка робочого місця є важливим елементом охорони праці.

Важливе значення мають електробезпека та технічний стан обладнання. Усі електричні пристрої повинні бути справними та мати належне заземлення. Робоче місце повинно бути обладнане стабільним джерелом живлення та захистом від короткого замикання. Кабелі живлення мають бути розташовані таким чином, щоб не створювати небезпеки для користувача. Також необхідно регулярно перевіряти стан електрообладнання. Використання несправної техніки може призвести до аварійних ситуацій або ураження електричним струмом. Дотримання правил електробезпеки є обов'язковою умовою безпечної роботи.

Для підтримання працездатності працівника необхідно правильно організувати режим праці та відпочинку. Рекомендується робити короткі перерви після кожної години роботи за комп'ютером. Під час перерв доцільно виконувати вправи для очей та легку фізичну розминку. Також необхідно

підтримувати чистоту та порядок на робочому місці. Надмірна кількість предметів на столі може створювати незручності та відволікати від роботи. Важливо забезпечити достатній рівень освітлення та вентиляції приміщення. Виконання всіх зазначених вимог сприяє підвищенню продуктивності праці та збереженню здоров'я працівника.

3.3 Висновки до розділу

У даному розділі було проведено аналіз умов праці розробника інформаційної системи та розглянуто основні вимоги до організації робочого місця програміста. Встановлено, що робота за персональним комп'ютером пов'язана з підвищеним навантаженням на органи зору, нервову систему та опорно-руховий апарат. Саме тому важливим є дотримання санітарно-гігієнічних норм, вимог ергономіки та правил техніки безпеки. Особливу увагу необхідно приділяти освітленню, мікроклімату приміщення та правильному розташуванню обладнання. Виконання цих вимог дозволяє створити комфортні умови праці та підвищити продуктивність роботи. Також це сприяє зменшенню ризику професійних захворювань.

У ході аналізу було встановлено, що правильно організоване робоче місце позитивно впливає на ефективність праці програміста. Використання ергономічних меблів, правильне розташування монітора та дотримання режиму праці дозволяють зменшити втому та зберегти працездатність працівника. Важливим аспектом є також забезпечення електробезпеки та контроль технічного стану обладнання. Усі електричні пристрої повинні бути справними та відповідати вимогам безпеки. Не менш важливим є дотримання правил користування комп'ютерною технікою та регулярне проведення перерв під час роботи. Це дозволяє мінімізувати негативний вплив тривалої роботи за комп'ютером.

Таким чином, дотримання вимог охорони праці є необхідною умовою безпечної та ефективної роботи розробника програмного забезпечення. Створення комфортного робочого середовища сприяє підвищенню

продуктивності праці та забезпечує збереження здоров'я працівника. Отримані результати можуть бути використані під час організації робочих місць для фахівців у сфері інформаційних технологій. Крім того, дотримання вимог охорони праці є важливим елементом професійної діяльності програміста. Це забезпечує не лише безпеку роботи, але й підвищує якість виконання професійних завдань. Отже, питання охорони праці є важливою складовою процесу розробки інформаційних систем.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено інформаційну систему генерування тексту, доповненого пошуком, для інформаційної підтримки студентів Харківського національного університету імені В. Н. Каразіна. Основною метою роботи було створення програмного засобу у вигляді Telegram-бота, який забезпечує пошук інформації у текстових документах університету та формування відповідей природною мовою. У ході виконання роботи було проаналізовано сучасні інформаційні системи освітнього призначення, досліджено можливості чат-ботів та технологій штучного інтелекту, а також розглянуто методи обробки природної мови та пошуку інформації. У теоретичній частині роботи було проведено аналіз сучасних інформаційних систем в освіті та досліджено особливості використання AI-технологій у сфері підтримки користувачів. Було розглянуто сучасні чат-боти, системи генерації тексту та Retrieval-Augmented Generation підхід. Також було проаналізовано методи обробки природної мови, зокрема TF-IDF, cosine similarity та принципи семантичного пошуку. Проведений аналіз показав, що використання NLP та RAG-технологій є перспективним напрямом розвитку освітніх інформаційних систем. Крім того, було обґрунтовано вибір технологій Python, aiogram, FastAPI та PostgreSQL для реалізації програмного забезпечення.

У практичній частині роботи було спроектовано архітектуру системи та реалізовано основні програмні модулі. Було створено Telegram-бота для взаємодії з користувачами, реалізовано механізм обробки текстових документів та розроблено базу знань. Для пошуку інформації використано метод TF-IDF та cosine similarity, що дозволило забезпечити ефективне знаходження релевантних fragments тексту. Також було реалізовано механізм генерації відповідей на основі знайдених фрагментів документів. Розроблена система підтримує додавання нових документів та оновлення бази знань без зміни програмного коду.

У ході тестування було перевірено роботу всіх основних функцій системи. Результати тестування показали, що Telegram-бот успішно обробляє запити

користувачів, знаходить необхідну інформацію та формує коректні відповіді. Система стабільно працює з текстовими документами та забезпечує швидкий доступ до університетської інформації. Аналіз результатів підтвердив працездатність реалізованого механізму пошуку та доцільність використання обраних технологій. Водночас було встановлено, що для покращення якості пошуку у майбутньому доцільно реалізувати semantic search та embeddings.

Практична цінність роботи полягає у створенні реального програмного продукту, який може бути використаний для автоматизації інформаційної підтримки студентів та абітурієнтів. Розроблена система дозволяє зменшити кількість типових звернень до працівників університету та спрощує доступ користувачів до нормативної й навчальної інформації. Використання Telegram як основного інтерфейсу забезпечує простоту та доступність роботи із системою. Отримані результати можуть бути використані як основа для подальшого розвитку інтелектуальних освітніх сервісів.

У подальшому система може бути розширена за рахунок використання PostgreSQL, pgvector, embeddings та сучасних AI-моделей для реалізації семантичного пошуку та генерації більш якісних відповідей. Також перспективним є додавання web-інтерфейсу, підтримки PDF-документів, OCR та інтеграції з університетськими інформаційними системами. Таким чином, поставлену мету роботи досягнуто, а всі основні задачі успішно виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*. 2020. Vol. 33. Pp. 9459–9474.
2. Gao Y., Xiong Y., Gao X., Jia K., Pan J., Bi Y., et al. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997*. 2023.
3. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention Is All You Need. *Advances in Neural Information Processing Systems*. 2017. Vol. 30. Pp. 5998–6008.
4. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2019. Vol. 1. Pp. 4171–4186.
5. OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*. 2026.
6. Karpukhin V., Oguz B., Min S., Lewis P., Wu L., Edunov S., Chen D., Yih W.-T. Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020. Pp. 6769–6781.
7. Pan J. J., Wang J., Li G. Survey of Vector Database Management Systems. *The VLDB Journal*. 2024. Vol. 33. Pp. 1591–1615.
8. Seyed Monir S., Lau I., Yang S., Zhao D. VectorSearch: Enhancing Document Retrieval with Semantic Embeddings and Optimized Search. *arXiv preprint arXiv:2409.17383*. 2024.
9. Abdul Razak M. S., et al. Enhancing Student Support with AI: A College Assistance Chatbot Using NLP and ANN. *Proceedings of the Second International Conference on Networks, Multimedia and Information Technology (NMITCON)*. 2024. DOI: 10.1109/nmitcon62075.2024.10699279.

10. Ahriz S., et al. Enhancing Information Technology Governance in Universities: A Smart Chatbot System Based on Information Technology Infrastructure Library. *Engineering, Technology & Applied Science Research*. 2024. Vol. 14(6). Pp. 18199–18206.
11. Zhao L., Wang Y. ChatGPT in Education: Opportunities and Challenges. *Educational Technology & Society*. 2024. Vol. 27(2). Pp. 134–145.
12. Chen X., Li H. Integrating ChatGPT into Healthcare Chatbots: A Review. *Journal of Medical Systems*. 2024. Vol. 48(1). Pp. 12–24.
13. Ding Y., Najaf M. Interactivity, Humanness, and Trust: A Psychological Approach to AI Chatbot Adoption in E-commerce. *BMC Psychology*. 2024. Vol. 12. Pp. 20–56.
14. Neupane S., Hossain E., Keith J., Tripathi H., Ghiasi F., et al. From Questions to Insightful Answers: Building an Informed Chatbot for University Resources. *arXiv preprint arXiv:2405.08120*. 2024.
15. Ma L., Zhang Z., Wang X., Li J., Li G. A Comprehensive Survey on Vector Database: Storage and Retrieval Techniques, Challenges. *arXiv preprint arXiv:2310.11703*. 2023.
16. Shuster K., Poff S., Chen M., Kiela D., Weston J. Retrieval Augmentation Reduces Hallucination in Conversation. *Findings of the Association for Computational Linguistics: EMNLP 2021*. 2021. Pp. 3784–3803.
17. Huang J., Chang K. C.-C. Towards Reasoning in Large Language Models: A Survey. *Findings of the Association for Computational Linguistics: ACL 2023*. 2023. Pp. 1049–1065.
18. Wei J., Wang X., Schuurmans D., Bosma M., Ichter B., Xia F., et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*. 2022. Vol. 35. Pp. 24824–24837.
19. Izacard G., Grave E. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*. 2021. Pp. 874–880.

20. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. 2019. Pp. 3982–3992.
21. Robertson S., Zaragoza H. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval. 2009. Vol. 3(4). Pp. 333–389.
22. Smutny P., Bojko M. Comparative Analysis of Chatbots Using Large Language Models for Web Development Tasks. Applied Sciences. 2024. Vol. 14(21). Pp. 172–205.
23. Li Y. Enhancing Chatbot Responses Based on Natural Language Processing Techniques. Proceedings of the 2023 International Conference on Image, Algorithms and Artificial Intelligence. 2023. Pp. 574–579.
24. Singh R., Gupta N. Enhancing Chatbot Responsiveness Using Transformer Architectures. International Journal of Computer Applications. 2023. Vol. 175(7). Pp. 25–33.
25. Bălan C. Chatbots and Voice Assistants: Digital Transformers of the Company–Customer Interface. Journal of Theoretical and Applied Electronic Commerce Research. 2023. Vol. 18(2). Pp. 995–1019.

ДОДАТОК А

```

import os
import asyncio
from pathlib import Path

from dotenv import load_dotenv
from aiogram import Bot, Dispatcher, types
from aiogram.filters import CommandStart, Command

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

try:
    from openai import OpenAI
except ImportError:
    OpenAI = None

load_dotenv()

BOT_TOKEN = os.getenv("BOT_TOKEN")
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")

DATA_DIR = Path("data")

if not BOT_TOKEN:
    raise ValueError("Не знайдено BOT_TOKEN у файлі .env")

bot = Bot(token=BOT_TOKEN)
dp = Dispatcher()

documents = []
chunks = []
chunk_sources = []
vectorizer = None
chunk_vectors = None

def load_documents():
    global documents, chunks, chunk_sources, vectorizer, chunk_vectors

    documents = []
    chunks = []

```

```

chunk_sources = []

DATA_DIR.mkdir(exist_ok=True)

for file_path in DATA_DIR.glob("*.txt"):
    text = file_path.read_text(encoding="utf-8",
errors="ignore")
    documents.append({
        "filename": file_path.name,
        "text": text
    })

    parts = split_text(text)

    for part in parts:
        chunks.append(part)
        chunk_sources.append(file_path.name)

if chunks:
    vectorizer = TfidfVectorizer(
        lowercase=True,
        max_features=10000
    )
    chunk_vectors = vectorizer.fit_transform(chunks)
else:
    vectorizer = None
    chunk_vectors = None

def split_text(text, chunk_size=900, overlap=150):
    text = " ".join(text.split())

    result = []
    start = 0

    while start < len(text):
        end = start + chunk_size
        result.append(text[start:end])
        start += chunk_size - overlap

    return result

def search_relevant_chunks(query, top_k=3):
    if not chunks or vectorizer is None:
        return []

```

```

    query_vector = vectorizer.transform([query])
    similarities = cosine_similarity(query_vector,
chunk_vectors).flatten()

    top_indexes = similarities.argsort()[::-1][:top_k]

    results = []

    for index in top_indexes:
        if similarities[index] > 0:
            results.append({
                "text": chunks[index],
                "source": chunk_sources[index],
                "score": float(similarities[index])
            })

    return results

def generate_answer(question, found_chunks):
    context = "\n\n".join(
        [f"Джерело: {item['source']}\n{item['text']}" for item in
found_chunks]
    )

    if not context:
        return "На жаль, я не знайшов відповідної інформації в
базі знань."

    if not OPENAI_API_KEY or OpenAI is None:
        return (
            "Я знайшов таку інформацію:\n\n"
            f"{found_chunks[0]['text']}\n\n"
            f"Джерело: {found_chunks[0]['source']}"
        )

    client = OpenAI(api_key=OPENAI_API_KEY)

    prompt = f"""

```

Ти інформаційний помічник для студентів Харківського національного університету імені В. Н. Каразіна.

Відповідай тільки на основі наданого контексту.

Якщо інформації недостатньо, скажи, що в базі знань немає точних даних.

Питання користувача:

```
{question}
```

Контекст:

```
{context}
```

Сформулюй коротку, зрозумілу відповідь українською мовою.

```
"""
```

```

    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": "Ти корисний університетський інформаційний помічник."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.2
    )

    answer = response.choices[0].message.content

    sources = "\n".join(set([item["source"] for item in found_chunks]))

    return f"{answer}\n\nДжерела:\n{sources}"

```

```

@dp.message(CommandStart())
async def start_handler(message: types.Message):
    await message.answer(
        "Вітаю! Я інформаційний бот для підтримки студентів ХНУ імені В. Н. Каразіна.\n\n"
        "Постав мені питання, наприклад:\n"
        "• Які умови вступу?\n"
        "• Як отримати стипендію?\n"
        "• Що таке академічна доброчесність?\n\n"
        "Команди:\n"
        "/reload — оновити базу знань\n"
        "/help — допомога"
    )

```

```

@dp.message(Command("help"))
async def help_handler(message: types.Message):
    await message.answer(
        "Просто напиши питання звичайним текстом.\n\n"
    )

```

```

        "Бот знайде відповідну інформацію в документах з папки
data і сформує відповідь."
    )

```

```

@dp.message(Command("reload"))
async def reload_handler(message: types.Message):
    load_documents()
    await message.answer(
        f"Базу знань оновлено.\n"
        f"Документів: {len(documents)}\n"
        f"Фрагментів тексту: {len(chunks)}"
    )

```

```

@dp.message()
async def question_handler(message: types.Message):
    user_question = message.text

    await message.answer("Шукаю інформацію...")

    found = search_relevant_chunks(user_question, top_k=3)
    answer = generate_answer(user_question, found)

    await message.answer(answer)

```

```

async def main():
    load_documents()

    print("Бот запущено.")
    print(f"Документів завантажено: {len(documents)}")
    print(f"Фрагментів тексту: {len(chunks)}")

    await dp.start_polling(bot)

```

```

if __name__ == "__main__":
    asyncio.run(main())

```