МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Мобільний додаток підтримки надання інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту

Здобувачки групи ІТ.м-22 Чімирис Юлії Сергіївни
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Юлія ЧІМИРИС
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доцент Володимир НАГОРНИЙ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО

«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Чімирис Юлії Сергіївні
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Мобільний додаток підтримки надання інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «16» грудня 2023 р.

3 Вхідні дані до кваліфікаційної роботи технічна документація OpenAI, технічна документація LangChain, технічна документація React Native

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі, методи дослідження, проектування мобільного додатку для надання інформації студентам факультету ЕлІТ, розробка мобільного додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність проблеми, мета дипломної роботи, задачі проекту, функціональні та технічні вимоги до системи, проектування інформаційної системи, проектування, етапи розробки програмного продукту, практична реалізація, висновки, практичне значення, оприлюднення результатів роботи

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання технічних вимог.	до 29.10.23	
2	Пошук інформаційних ресурсів.	до 29.10.23	
3	Систематизація зібраної інформації.	до 31.10.23	
4	Розроблення теоретичного базису роботи.	до 02.11.23	
5	Створення діаграм IDF.	до 09.11.23	
6	Визначення необхідних інструментів для реалізації проекту.	до 13.11.23	
7	Планування робіт	до 16.11.23	
8	Аналіз та визначення ризиків проекту.	до 18.11.23	
9	Реалізація структури	до 20.11.23	
10	Реалізація функціоналу додатку.	до 25.11.23	
11	Проведення тестування функціональності додатку.	до 10.12.23	

Магістрант _____ Юлія ЧІМИРИС

Керівник роботи _____ доцент кафедри ІТ, к.т.н., доцент Володимир НАГОРНИЙ

АНОТАЦІЯ

The topic of the Master's thesis is «Mobile application for supporting the provision of information services to ElIT faculty students using artificial intelligence tools».

The explanatory note consists of an introduction, 4 chapters, conclusions, a list of used sources with 36 titles, and 2 appendices. The total volume of the work is 97 pages, including 56 pages of the main text, 4 pages of the list of used sources, and 36 pages of appendices.

The relevance of the work lies in the need for quick and effective information retrieval by students in the conditions of the modern educational environment. With the introduction of artificial intelligence tools, the mobile application not only provides access to diverse information but also personalizes this process, taking into account the individual needs and questions of each student.

Objective: Development of a mobile application providing informational services to students of the ElIT faculty, utilizing artificial intelligence tools, aimed at ensuring quick access to diverse and reliable information. The goal is to simplify and enhance the efficiency of user interaction with the application.

The Master's thesis is dedicated to the development of a mobile application to support the provision of informational services to students of the ElIT faculty using artificial intelligence tools. Modern methods of providing information services to students, such as email newsletters and university websites, often fail to meet the requirements and needs. Over time, it becomes evident that these classical approaches often limit interaction and do not guarantee sufficient individualization, complicating effective interaction and communication with students. Websites may not be as intuitive and interactive as students would like. Information is often located on different pages or in different sections, and the interface may be challenging to perceive and non-intuitive, complicating navigation and the search for necessary information. In turn, email, while an effective means, sometimes

becomes overloaded and impractical for quick communication with students. In a world of rapid technological development with high demands for individualized approaches, traditional methods of information provision lose their relevance. Therefore, it is important to consider and implement new, advanced technologies that provide a high level of interaction, individualization, and effective communication to maximize student engagement. To avoid these problems, artificial intelligence (AI) is used.

In modern conditions, the use of artificial intelligence in university educational programs has become a necessary part of technological development, finding broad application in various fields. This innovative tool aims to enhance the quality of education and create personalized approaches for each student. For convenient use of AI, universities increasingly employ innovative approaches to improve interaction and provide informational support by incorporating it into chatbots, which prove to be effective tools for quick and accessible information delivery to students. Many universities are actively implementing chatbots to simplify communication and provide convenient access to various services. By 2022, other initiatives have utilized automated chatbots that respond to specific questions, as well as simulation systems to create the impression of AI usage. Many universities actively utilize automated chatbots that provide basic information regarding admissions, schedules, academic matters, and other aspects of student life. These chatbots can answer standard questions, allowing students to receive real-time information. However, challenges arise in the use of artificial intelligence and chatbots in education. It is important to consider the confidentiality aspects of students' personal data and ensure the security of processing this information. Additionally, it is necessary to improve chatbot algorithms, considering the diversity of user needs, to achieve greater accuracy and efficiency in providing responses. Therefore, the goal of this work is to develop a mobile application that utilizes the capabilities of artificial intelligence to provide informational support to students of the ElIT faculty. The main focus is on addressing the needs of each user and creating a personalized user experience. One of the key features of the application

is the ability to provide quick and reliable information about the ElIT faculty. Our AI-powered chatbot allows students to receive instant answers to their questions. Artificial intelligence algorithms help maintain an extended dialogue and provide accurate context-related answers. The application is designed to address the issue of information fragmentation that often arises when using conventional channels. We aim to provide easily accessible and organized information to facilitate user interaction with our application. This update in information services can be a crucial step in easing communication and supporting students on their academic journey. Various technologies are used to ensure the effective functioning of this mobile application. The frontend of the application, which interacts with the user, is based on tools such as React Native, React Navigation, NativeWind, and Lottie Animations. React Native allows for the development of cross-platform mobile applications, with shared code working on both platforms. React Navigation enables the creation of diverse navigation structures, while NativeWind and Lottie Animations provide fast and stylish graphic processing.

The backend part utilizes the LangChain framework, OpenAI Chat Model, ChromaDB vector storage, and the Telethon API. The Python programming language serves as the foundation for development, providing a clear syntax and support for various programming approaches. The Flask framework is employed for implementing the server-side, and LangChain offers the capability to create applications based on large language models (LLM) considering context. To optimize the system's interaction with users, a document vectorization process is employed, where each received document is transformed into a numerical vector. This operation can be performed using vectorization methods such as utilizing embeddings from OpenAI or other approaches.

Using the IDEF0 notation for developing business processes and describing the stages of request processing, displaying results, and information search, the project's task structure was determined, requirements were formulated, and a work plan was developed, taking into account time and possible delays. For the project " Mobile application for

supporting the provision of information services to ElIT faculty students using artificial intelligence tools," UML sequence and activity diagrams represent model working processes in information systems. The sequence diagram visually represents interactions between objects over time, while the activity diagram replicates business processes and actions in the form of sequences. Additionally, a use case diagram has been created for the mobile application, graphically depicting system and software requirements, with a Use Case diagram schematically illustrating the interaction of users with the system and its functionality.

Overall, this software implementation of the mobile application combines the power and convenience of tools to provide the user with a high-quality and innovative experience. We aim to develop an innovative mobile application that not only addresses current challenges in providing informational support to students but also reflects a new era in education. In our understanding of the future educational process, artificial intelligence plays a key role.

Our mobile application is designed with the needs of modern students from the Faculty of ElIT in mind. It not only solves problems that arise with conventional methods of informational support but goes beyond, utilizing artificial intelligence tools.

Keywords: MOBILE APPLICATION, LLM, ARTIFICIAL INTELLIGENCE, LANGCHAIN, RAG, REACT NATIVE, VECTOR DB.

ЗМІСТ

ВСТУП

Актуальність. З виникненням інтенсивного розвитку інформаційного простору та технологій у сучасному світі виникає розширена потреба в доступі до інформації серед студентів факультету. Основною проблемою, з якою стикаються студенти факультету, полягає в ускладненні процесу отримання необхідної та достовірної інформації через велику кількість інформаційних ресурсів, включаючи веб-сайти та файли університету. Студенти мають труднощі з ефективним переглядом усіх наявних ресурсів та отриманням доступу до необхідних даних. Це призводить до втрати часу та невпорядкованості у процесі отримання інформації, ускладнюючи студентське життя.

У світлі такого прискореного росту інформаційних технологій починають функціонувати різноманітні інформаційні системи та веб-ресурси, які спрощують пошук, перегляд та полегшують доступ до інформації. Особлива увага приділяється створенню нових систем і ресурсів, спрямованих на представлення інформації, відмінної від традиційних ресурсів в яких пошук не завжди є легким. У зв'язку з цим прийнято рішення розробити мобільний додаток спрямований на полегшення життя студентів. Цей додаток призначений надавати інформацію відповідно до запитань, спрощуючи процес взаємодії, зменшуючи витрати часу та роблячи його більш ефективним.

Мета. Розробка мобільного додатку для надання інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту. Додаток має надавати швидкий доступ до різноманітної та достовірної інформації, спрощуючи процес взаємодії та роблячи його більш ефективним для користувачів.

Об’єкт дослідження. Підтримка надання інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту.

Предмет дослідження. Мобільний додаток підтримки надання інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту.

Завданням. Використовуючи нотацію IDEF0, розробляємо бізнес-процеси та детально описуємо кроки, які обробляють запити, відображають результати та виконують пошук інформації. Створити діаграми послідовності та активності UML що візуалізують робочі процеси в інформаційних системах, та Use Case для графічного представлення системних, та програмних вимог. Для досягнення мети проекту необхідно виконати ряд завдань:

- докладно розглянути проблеми в даній області та ознайомитися з останніми статтями, щоб визначити, наскільки актуальною є ця тема, і для кого саме призначений цей мобільний додаток;
- провести порівняльний аналіз існуючих аналогів мобільних додатків у галузі освіти та визначити функціональні та нефункціональні вимоги;
- обрати оптимальні технології, які відповідають вимогам проекту та забезпечать ефективну реалізацію мобільного додатку;
- розробити детальну модель та структуру системи;
- здійснити програмну реалізацію розробленої моделі, структури та впровадити необхідний функціонал мобільного додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Сучасні методи надання інформаційних послуг студентам, такі як веб-сайти університету та електронна пошта, вже не завжди відповідають сучасним вимогам та потребам. З плином часу виявляється, що ці традиційні канали часто обмежені взаємодією та не забезпечують достатньої персоналізації, що ускладнює ефективну комунікацію та взаємодію із студентами [1].

Наприклад, веб-сайти університетів можуть бути не такими інтерактивними, як би хотілося студентам. Часто інформація розташована в різних розділах, інтерфейс може бути складним для сприйняття, що ускладнює пошук та отримання необхідної інформації [2]. Електронна пошта, хоч і є ефективним засобом, але іноді може стати перевантаженою та непрактичною для швидкої та ефективної комунікації.

У світі стрімких технологічних змін і високих вимог до індивідуального підходу традиційні методи інформаційного забезпечення втрачають свою актуальність [3]. З цієї причини необхідно розглядати та впроваджувати нові, більш сучасні технології, що забезпечують високий рівень інтерактивності, персоналізації та ефективної комунікації для максимального залучення студентів.

У цьому контексті використання штучного інтелекту (ШІ) в університетських освітніх програмах стало незамінною складовою технологічного прогресу знаходячи широке застосування в різних сферах. Це ключовий інноваційний інструмент, спрямований на підвищення якості навчання та створення індивідуалізованих підходів для кожного студента [4].

Однією з актуальних областей використання ШІ є розробка мобільних додатків для надання інформаційних послуг студентам [5]. Ці додатки використовують інструменти ШІ для індивідуалізації навчального процесу, адаптації до індивідуальних потреб та створення інтерактивного середовища.

Плюсом використання ШІ в освіті є можливість створення персоналізованих навчальних програм. ШІ аналізує індивідуальні особливості студентів та їх рівень засвоєння матеріалу, надаючи спеціалізовані завдання та рекомендації, що сприяє ефективному навчанню та розвитку потенціалу.

Іншою перевагою є можливість взаємодії із студентами в режимі реального часу. Системи ШІ можуть виступати складовою частиною обробки запитів чат-ботів, які відповідають на питання студентів, надають інформацію та допомагають вирішити навчальні завдання [6].

Використання ШІ в освіті сприяє автоматизації та оптимізації навчальних процесів, робить їх більш ефективними та цікавими для студентів, а інновації вносять значний внесок у сучасну освітню систему, створюючи нові можливості для навчання та розвитку.

Зокрема, ШІ вже успішно використовується в університетах для поліпшення комунікації та надання інформаційних послуг. Застосування інструментів ШІ включає використання алгоритмів машинного навчання для аналізу поведінки студентів та персоналізації інформаційних послуг, що сприяє кращому розумінню їх потреб і наданню індивідуалізованих порад та рекомендацій [7].

Отже, якщо розглядати результати впровадження ШІ, то можна визначити переваги:

- штучний інтелект дозволяє створювати персоналізовані підходи до навчання, враховуючи індивідуальні потреби та здібності студентів;
- автоматизація процесів і використання алгоритмів машинного навчання сприяють оптимізації навчальних процесів;

- інтелектуальні системи можуть адаптуватися до різних стилів навчання, використовуючи різноманітні методи та ресурси;
- створюється можливість створювати інтерактивні чат-боти, що підтримують студентів у режимі реального часу;
- інструменти ШІ забезпечують аналіз даних та відстеження прогресу студентів, сприяючи усуненню слабких місць та вдосконаленню навчального процесу;
- впровадження ШІ в університетах відкриває нові можливості для наукових досліджень та інновацій у сфері освіти.

Ці переваги свідчать про значущий вклад штучного інтелекту у вдосконалення освітнього процесу та покращення навчання для студентів. При створенні мобільного додатку для факультету ЕлІТ можна використовувати різні інструменти ШІ, такі як обробка природної мови для розуміння запитань студентів та автоматичне оновлення інформації з веб-сайту університету або бази даних. Однак слід враховувати не лише переваги, а й можливі недоліки використання інструментів ШІ, такі як:

- збір та обробка великої кількості особистих даних може породжувати занепокоєння стосовно конфіденційності та приватності студентів;
- алгоритми машинного навчання можуть не завжди точно визначати індивідуальні потреби кожного студента, що може призводити до неправильних рекомендацій чи завдань;
- залежність від технічних рішень може призвести до ситуацій, коли системи ШІ не працюють належним чином через технічні збої або несподівані обставини.

Ці недоліки слід враховувати при впровадженні та використанні інструментів штучного інтелекту в освіті. Використання великих обсягів даних та аналітичних інструментів ШІ дозволяє університетам здійснювати стратегічне планування, прогнозування потреб учнів та вдосконалення управлінських рішень, сприяючи створенню адаптивних освітніх середовищ для розвитку ключових навичок [8].

Досвід ведучих університетів, таких як Стенфорд, та активне використання відкритих даних у сховищах, вказують на позитивні тенденції у сприйнятті та імплементації штучного інтелекту в університетське навчання [9].

Дослідження можливостей використання штучного інтелекту в освіті також розкриває нові перспективи, включаючи використання зображень, створених ШІ, та впровадження інтелектуальних систем у віртуальні класи. Ці дослідження свідчать про постійний розвиток сфери та відкривають можливості для подальших інновацій у сфері університетської освіти.

1.2 Аналіз існуючих аналогів

З інтенсивним розвитком технологій у сфері освіти університети все частіше вдаються до інноваційних підходів у покращенні взаємодії зі студентами та наданні інформаційної підтримки. Один із таких напрямків є використання чат-ботів, які стають ефективними інструментами для забезпечення швидкої та доступної інформації студентам. У світлі цього, багато університетів вже впроваджують чат-боти, щоб полегшити комунікацію та забезпечити зручний доступ до різноманітних сервісів. Проте, крім розробок, які ґрунтуються на штучному інтелекті, є також проекти, які використовують автоматизовані чат-боти або системи імітації для досягнення тих самих цілей.

Багато університетів активно впроваджують автоматизовані чат-боти для забезпечення основної інформації щодо вступу, графіків, академічних питань та інших повсякденних аспектів студентського життя. Ці чат-боти можуть відповідати на типові запитання, надаючи студентам доступ до необхідної інформації в режимі реального часу [10].

Системи імітації, яку деякі університети можуть використовувати хоч і не ґрунтуються на штучному інтелекті, але програмовані для імітації розмови зі студентами. Ці системи можуть використовувати запрограмовані сценарії та ключові слова для створення враження взаємодії з реальною особою [11].

Онлайн-платформи чату також використовуються університетами та можуть використовувати стандартні онлайн-платформи чату для спілкування зі студентами. Це може включати в себе використання звичайних текстових чатів або віртуальних асистентів для надання інформації та відповідей на питання. Також університети можуть вирішувати використовувати спеціалізовані платформи чат-ботів, які розроблені для взаємодії зі студентами та надання різноманітних послуг [12].

Цей новаторський проект з розробки мобільного додатка для студентів факультету ЕлІТ, який використовує штучний інтелект, має потенціал стати значущим у навчальному середовищі. Наш мобільний додаток для студентів факультету ЕлІТ - унікальне рішення, позбавлене аналогів. Його особливість полягає у відсутності подібних інструментів, спрямованих на надання інформаційної підтримки саме для цього факультету. Мобільний формат надає студентам неперевершену зручність та доступність до необхідних ресурсів, сприяючи їхній ефективності в навчанні та повсякденному житті. Не існує інших аналогів, які так ефективно поєднують зручність інформаційної навігації з унікальністю, специфічною для студентів факультету ЕлІТ. Однак перед розгортанням обширної системи важливо провести ретельний аналіз різноманітних аналогів визначивши їхні переваги та недоліки.

Для порівняльного аналізу були обрані різного формату додатки від веб-додатків до мобільних додатків та чат-ботів в месенджері, пов'язаних з навчальними закладами. Серед обраних аналогів чат-бот корейського університету " Knubot" [13], чат-бот в месенджері Телеграм від Київського національного університету імені Тараса Шевченка "Абітурієнт економічного", чат-бот на сайті UCD "Rua Bot", та чат-бот на освітній платформі "Coursera".

Knubot - це інноваційний чат-бот, розроблений для Корейського національного університету (KNU), який використовується для покращення взаємодії і обслуговування студентів та відвідувачів. Основною метою цього чат-бота є забезпечення швидкого та ефективного доступу до інформації про університет та його сервіси, зображений на рисунку 1.1.

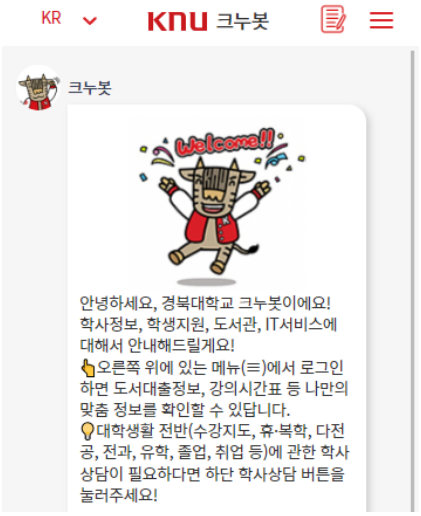

Рисунок 1.1 – Чат-бот «Knubot»

Джерело: [13]

Корейський національний університет створив застосунок для поліпшення обслуговування студентів та відвідувачів. Основною функціональною складовою є інформаційна підтримка, яка призначена для відповіді на різноманітні запитання студентів та відвідувачів стосовно університету, академічних аспектів та розкладів. Крім того, доступна можливість спілкування через чат-інтерфейс, який дозволяє

користувачам отримувати інформацію в режимі реального часу та легко взаємодіяти безпосередньо через месенджер. Це зображено на рисунку 1.2.

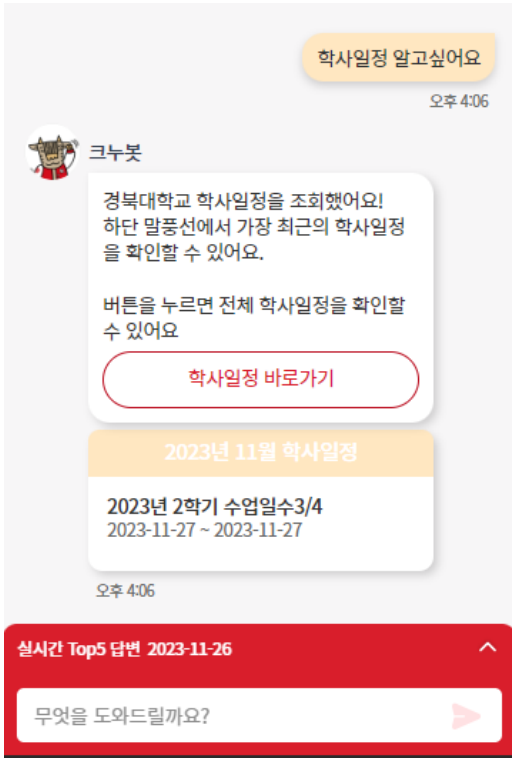

Рисунок 1.2 – Чат-бот «Knubot» при користуванні

Джерело: [13]

Чат-бот підтримує дві мовні варіації, а саме корейську та англійську, що розширює його використання не лише для мешканців Кореї, але і для користувачів, які не володіють корейською мовою. Розглядаючи всі ці аспекти, можна скласти таблицю 1.1 переваг та недоліків, яка відобразить його характеристики.

Таблиця 1.1 – Переваги та недоліки чат-боту «Knubot»

Переваги	Недоліки
Забезпечує швидкий та простий доступ до необхідної інформації.	Єдина можливість отримання доступу через веб-сайт.

Продовження таблиці 1.1

Переваги	Недоліки
Допомагає в автоматизації відповідей на типові запитання, зменшуючи навантаження на персонал університету.	Відсутність впровадження нових технологій в тому числі штучного інтелекту.
Створює зручний канал спілкування для студентів, які звикли до взаємодії через месенджери.	Деяка інформація подається у вигляді посилання, а не безпосередньо в чаті, що вимагає переходу на інший веб-сайт.

Джерело: побудовано автором

У висновку важливо відзначити, що використання чат-бота Knubot студентами Корейського національного університету не лише забезпечує швидке, але й повноцінне реагування на їхні запитання, що покращує їхній загальний навчальний досвід та взаємодію з університетом. В цілому, використання чат-ботів у навчальних закладах сприяє створенню ефективних та інтерактивних інструментів комунікації, спрямованих на підвищення якості обслуговування студентів і спрощення процесів навчання та адміністрування університету.

Наступний чат-бот, який функціонує в популярному месенджері Телеграм[14], є ініціативою Київського національного університету імені Тараса Шевченка і відомий як «Абітурієнт економічного». Цей бот спрямований на поліпшення взаємодії та обслуговування студентів абітурієнтів економічного факультету. Основна мета цього чат-бота полягає в забезпеченні оперативного та ефективного доступу до інформації про університет, його сервісів, а також надання додаткової інформації. Важливо відзначити, що цей чат-бот призначений для зручності та швидкості отримання важливої інформації для абітурієнтів економічного факультету. Бот зображений на рисунку 1.3.

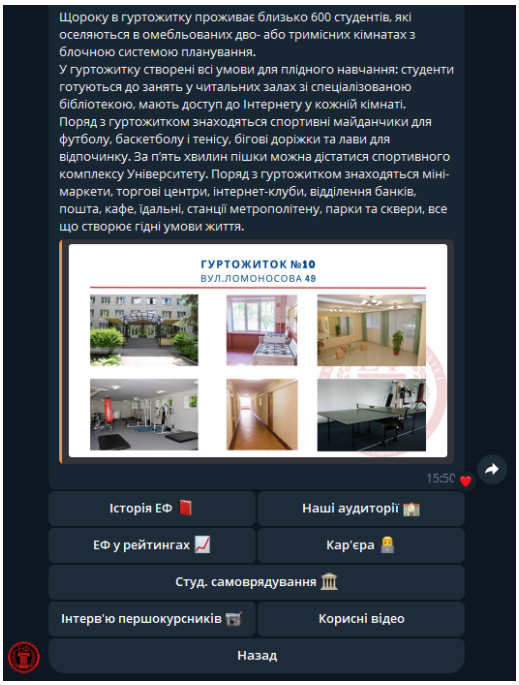

Рисунок 1.3 – Чат-бот «Абітурієнт економічного»

Джерело: [14]

Чат-бот пропонує важливі посилання на канали та чати, де користувачі можуть отримати детальнішу інформацію про університет, конкурси та життя студентів. Він також надає можливість перепитати щодо деталей вступної кампанії та взаємодіяти з іншими абітурієнтами. Все це відбувається за допомогою кнопок, що зображені на рисунку 1.4.

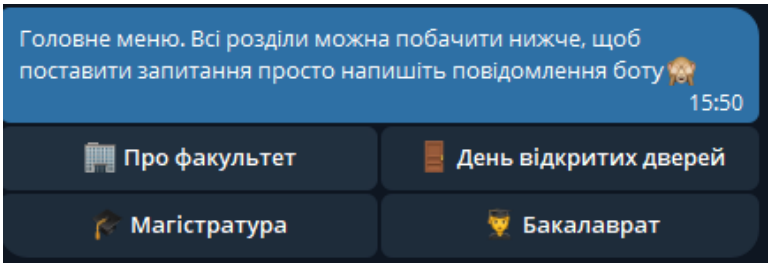

Рисунок 1.4 – Формат взаємодії з ботом

Джерело: [14]

Чат-бот також надає інформацію про гуртожиток економічного факультету, його розташування та умови проживання для студентів. У таблиці 1.2 наведено переваги на недоліки цього бота.

Таблиця 1.2 – Переваги та недоліки чат-боту «Абітурієнт економічного»

Переваги	Недоліки
Забезпечує швидкий та легкий доступ до необхідної інформації за допомогою кнопок переходів.	Відсутність можливості задати питань.
Зручність користування чат-ботом у месенджері.	Надання базової інформації без виходу за рамки заданої інформації.
Можливість зворотного зв’язку через бота.	Складна карта шляху для отримання інформації.
	Пряма залежність від зовнішнього ресурсу (Телеграм).

Джерело: побудовано автором

Наступним у списку чат-ботів є Rua від Університетського коледжу Дубліна [15]. Це онлайн-асистент, розроблений для надання інформації та відповідей на різноманітні питання щодо вступу, навчання та інших аспектів студентського життя. Важливо зауважити, що це не людина, а бот, який оптимально функціонує з короткими фразами. Користувачам рекомендується утримуватися від надання особистої інформації з метою збереження конфіденційності. Бот пропонує важливі посилання на ресурси, пов'язані з вступом та іншими темами, де можна знайти більше інформації про університет, конкурси та студентське життя.

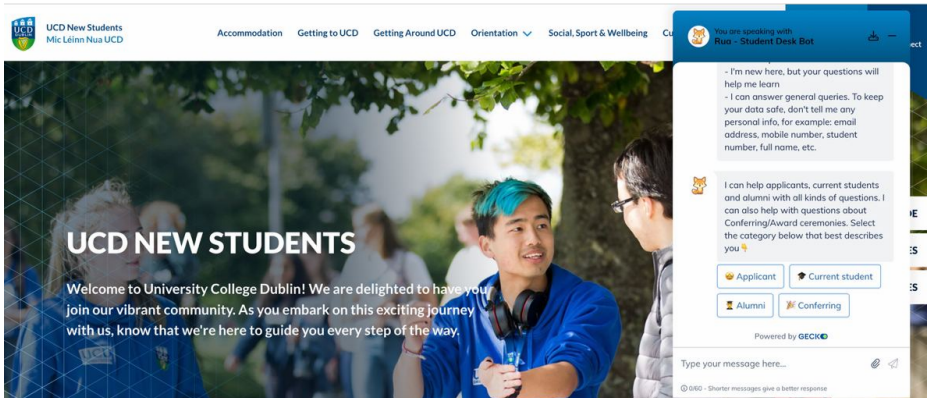

Рисунок 1.5 – Сайт університету UCD

Джерело: [15]

Проте основним недоліком є значні труднощі у розумінні запитань, надання інформації на певні питання та відсутність можливості ставити запитання поза заздалегідь визначеними рамками. Ця проблема зображена на рисунку 1.6.

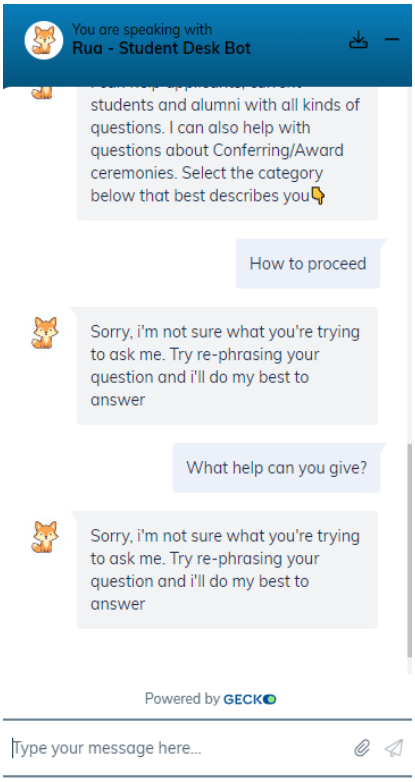

Рисунок 1.6 – Чат бот Rua

Джерело: [15]

Враховуючи, що він може залишатися неактуальним тривалий час, ми можемо скласти таблицю 1.3 з переваг і недоліків.

Таблиця 1.3 – Переваги та недоліки чат-боту «Rua»

Переваги	Недоліки
Відсутні	Надає швидкий доступ лише до обмеженої кількості інформації без ускладнень.

Джерело: побудовано автором

Через недостатнє доопрацювання та неможливість відповідати на запитання, це представляє собою найменш задовільну реалізацію боту.

Останнім до огляду буде чат-бот "Coursera" що використовується для покращення взаємодії з користувачами на своїй освітній платформі. Цей чат-бот допомагає студентам отримувати доступ до інформації про курси, процес навчання та інші пов'язані питання [16]. Основні функції боту на "Coursera" включають:

- чат-бот надає інформацію про різноманітні курси, їх особливості, викладачів та можливості навчання на платформі;
- він допомагає студентам з організацією навчання та взаємодією з різними елементами курсів;
- чат-бот відповідає на питання користувачів, пов'язані з платформою, процесом реєстрації, оплатою та іншими аспектами користування "Coursera";
- чат-бот надуває персоналізовані поради стосовно вибору курсів або плану навчання.

Чат-бот з сайту зображений на рисунку 1.7.

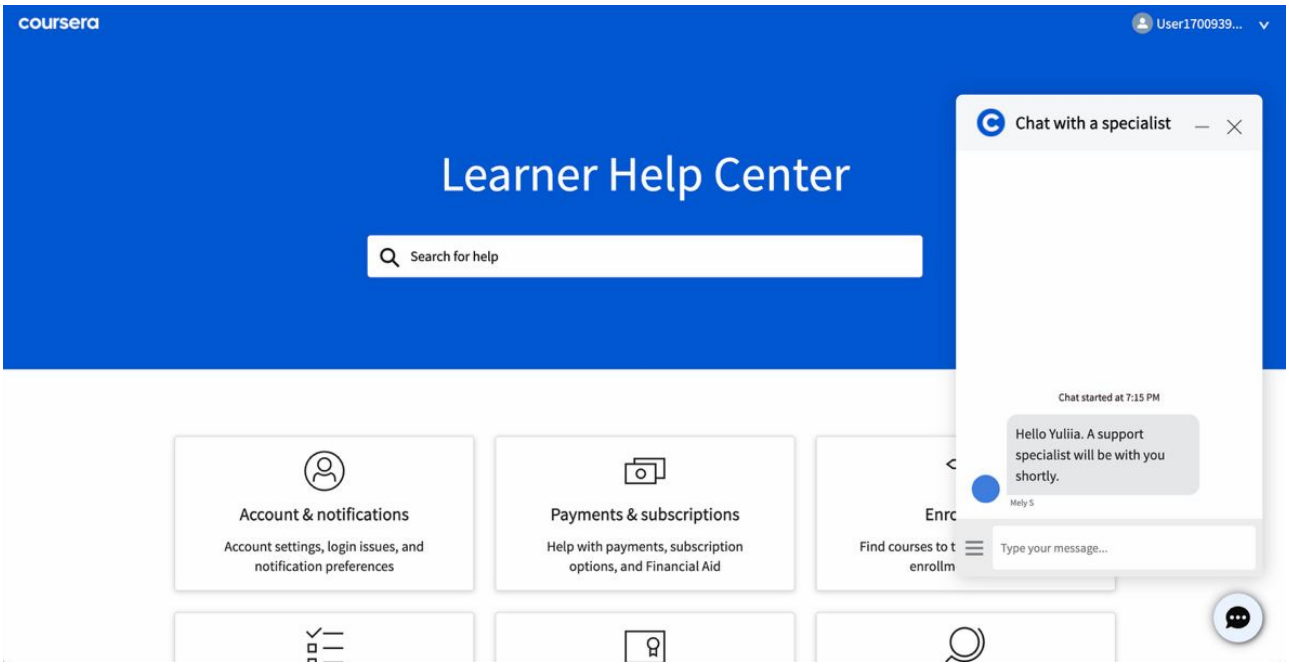

Рисунок 1.7 – "Coursera"

Джерело: [16]

"Coursera", як велика освітня платформа, використовує чат-боти для полегшення навчального процесу своїх користувачів. Однак, як і в усьому, є як позитивні аспекти, так і ряд нюансів, які варто врахувати, для цього створена таблиця 1.4.

Таблиця 1.4 – Переваги та недоліки чат-боту "Coursera"

Переваги	Недоліки
Підтримка в навчанні та організація процесу	Обмежена інтерактивність та глибина розуміння
Персоналізовані поради вибору курсів	Недостатня гнучкість у відповідях на зміни
	Необхідність залучення особи при розгляді складних питань.

Джерело: побудовано автором

В умовах великої конкуренції в галузі чат-ботів для студентів, наша головна задача розробити продукт, який повністю задовольнить унікальні потреби та очікування користувачів. Перед тим як розпочати розробку нашого мобільний додаток із вбудованим чат-ботом для студентів факультету ЕлІТ, ми докладно оглянули найкращі аналоги на ринку. Розглянемо ключові критерії, які визначають ефективність та зручність використання чат-ботів. Викладена таблиця 1.5 є результатом докладного аналізу провідних аналогів чат-ботів, в якій визначені основні аспекти, впливаючи на користувачів та формуючи якість обслуговування.

Таблиця 1.5 – Порівняльна таблиця аналогів

Критерії порівняння	Knubot	Абітурієнт економічного	Rua	Coursera
UI/UX дизайн	Зручна навігація, інтуїтивно зрозумілий інтерфейс	Дизайн обмежений зовнішньою платформою Телеграму	Інтуїтивно зрозумілий інтерфейс, проте відсутність навігації	Зручна навігація, інтуїтивно зрозумілий інтерфейс
Кросплатформність чат-боту	Доступний як через комп’ютер так і на мобільних додатках	Доступний як через комп’ютер так і на мобільних додатках, але з умовою доступу до Телеграму	Доступний як через комп’ютер так і на мобільних додатках	Доступний як через комп’ютер так і на мобільних додатках

Продовження таблиці 1.5

Критерії порівняння	Knubot	Абітурієнт економічного	Rua	Coursera
Засоби для надання персоналізованої відповіді	Відсутні	Відсутні	Відсутні	Присутні, потребують підключення оператору
Контекстуальна свідомість	Відсутня	Відсутня	Відсутня	Присутня лише в разі підключення оператору
Залежність від сторонньої платформи	Відсутня	Залежить від зовнішньої платформи Телеграму	Відсутня	Відсутня

Джерело: побудовано автором

Навіть найкращі аналоги обмежені у розширених можливостях персоналізації та функціонування чат-ботів. Наш проект вирізнятиметься високою швидкістю, доступністю, контекстуальною свідомістю та розширеними можливостями персоналізації, що забезпечить найвищий рівень задоволення користувачів. Аналізуючи порівняльну таблицю, ми визначили, що жоден із існуючих аналогів не відповідає повністю всім вимогам користувачів. Наш проект спрямований на збереження кращих аспектів кожного аналога та подолання їх обмежень. Інтегруючи високу швидкість, розширену персоналізацію та зручну навігацію, ми прагнемо зробити наш чат-бот для студентів ЕлІТ неперевершеним інструментом для отримання необхідної інформації та підтримки.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою даної роботи є створення мобільного додатка, спрямованого на забезпечення студентам факультету ЕлІТ зручного доступу до різноманітної інформації, включаючи дані з сайтів та файлів університету. Головною метою розробки є полегшення процесу отримання студентами актуальної інформації, що сприятиме їхньому більш ефективному навчанню та роботі з ресурсами.

Створене програмне рішення передбачає високий попит серед студентів факультету ЕлІТ, надаючи зручний інтерфейс для доступу до різноманітної інформації. Окрім офіційних джерел університету, додаток охоплюватиме інші корисні ресурси, що полегшить виконання навчальних та дослідницьких завдань студентів.

Такий мобільний додаток сприятиме підвищенню ефективності студентського життя, забезпечуючи оперативний доступ до важливої інформації для успішного навчання. Основні вимоги до функціоналу додатку включають обробку запитів у реальному часі на основі внутрішньої бази знань, надання студентам інформаційної довідки та високу швидкість обробки запитів. Ці етапи детально розглядаються у плані робіт проєкту, який докладно описано в Додатку А.

Така база знань попередньо буде проаналізована штучним інтелектом для надання найбільш релевантних результатів. Це дозволить студентам отримувати інформацію у вигляді змістовних та коротких викладів залежно від їх потреб та запитань, котрі в свою чергу будуть також проаналізовані штучним інтелектом.

Для досягнення мети проекту потрібно вирішити кілька завдань. Перше з них - провести дослідження області, спрямоване на поліпшення доступу студентів до інформаційних послуг. Основні виклики включають обмежений доступ до ресурсів

та неособистий характер існуючих підходів. З аналізу публікацій випливає, що є потреба в інноваційному рішенні для студентів з метою спрощення отримання інформації. Аналіз існуючих чат-ботів у галузі освіти, таких як платформи навчання та довідкові системи, визначає функціональні вимоги, такі як персоналізація інформації та легкий доступ. Розуміючи усі функціональні та нефункціональні вимоги на базі вже отриманих знань обрати найкращу технологію для реалізації проекту.Програмна реалізація здійснюється з використанням обраної моделі та структури. Впровадження функціоналу, який полегшить отримання інформації для студентів, враховуючи їхні індивідуальні потреби. Очікується покращення зручності та ефективності взаємодії з додатком для кінцевих користувачів.

2.2 Вибір методів та засобів реалізації

Після визначення мети проекту, функціональних та нефункціональних вимог до розроблюваного мобільного додатку, який має надавати інформаційні послуги студентам факультету ЕлІТ, з використанням інструментів штучного інтелекту, а також формалізації переліку обов’язкових завдань проєкту, було обрано методи дослідження.

Існує чотири основних методи проведення дослідження:

- теоретичний метод;
- емпіричний метод;
- системно-функціональний метод;
- метод моделювання.

Ці методи є важливими для комплексного дослідження та реалізації додатку, що сприятиме покращенню процесів надання інформаційних послуг студентам факультету ЕлІТ.

Теоретичний метод передбачає детальний аналіз особливостей та можливостей предметної області. У ході розробки проєкту "Мобільний додаток підтримки інформаційних послуг студентам факультету ЕлІТ" теоретичний метод використовувався для аналізу бізнес-процесів, вивчення потреб та тенденцій у галузі ІТ. Також цей метод застосовувався при виборі інструментів розробки, щоб виділити переваги та недоліки технологій, їхні особливості в залежності від розміру, тривалості, методів збереження даних програмного продукту та інше [17].

Використання емпіричного методу у цьому контексті включає в себе застосування спостережень, вимірювань та експериментів для отримання пізнання предметної області[18]. Його використання відбувалося на етапі аналізу процедур зміни позиції та перегляду заробітної плати, базуючись на особистому досвіді та інтерв'ю з колегами в ІТ-компанії[19].

Метод моделювання включав розробку схем та діаграм, які відображали процес розробки програмного продукту та його подальше використання [20]. У рамках проєкту "Мобільний додаток підтримки інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту" цей метод застосовувався на етапі побудови діаграм IDEF0 та Use Case [21].

Системно-функціональний метод, спрямований на детальний опис компонентів системи, їх взаємодію та залежності в контексті цілісного уявлення про систему[22]. Цей метод використовувався під час розробки програмного модулю проекту, зокрема для аналізу варіантів використання, які були відображені на діаграмі Use Case.

У програмній реалізації мобільного додатку використано різні технології. Для frontend частини додатку що бачить користувач використовується React Native, React Navigation, NativeWind, Lottie animations. Для частини backend використовуються фреймворк LangChain, OpenAI chat model, векторне сховище ChromaDB та Telethon API.

Почнімо з frontend. React Native [23] популярний фреймворк для розробки мобільних додатків ми можемо позбутися деяких проблем таких як:

- React Native дозволяє розробляти мобільні додатки, які працюють як на iOS, так і на Android, використовуючи велику частину спільного коду. Це економить час і ресурси розробки, оскільки не потрібно писати окремий код для кожної платформи;
- React Native користується великою та активною спільнотою розробників. Це означає, що є велика кількість додаткових бібліотек, модулів та плагінів, які можна використовувати для розширення функціоналу додатка;
- React Native дає можливість використовувати сторонні модулі і бібліотеки для реалізації різноманітних функцій.

Використання React Native стає привабливим вибором для розробки мобільних додатків, особливо для тих, хто вже знайомий з екосистемою React або шукає швидкий та ефективний спосіб створення мобільних застосунків для обох платформ.

React Navigation — це бібліотека для навігації в мобільних застосунках React Native, яка дозволяє створювати різноманітні навігаційні структури, такі як стеки, таби та інші [24].

NativeWind або Tailwind CSS — це CSS-фреймворк, який дозволяє швидко створювати стилі, використовуючи класи безпосередньо в розмітці [25].

Lottie Animations — це бібліотека для відтворення векторних анімацій у мобільних додатках та веб-сайтах за допомогою формату JSON [26]. Анімації можуть бути створені в програмах дизайну, таких як Adobe After Effects.

За допомогою цих інструментів створюється користувацький інтерфейс (frontend) мобільного додатка. Однак основна логіка та штучний інтелект знаходяться в backend.

Основою звичайно є мова програмування така як Python, яка підходить як для початківців, так і для професіоналів завдяки своєму зрозумілому синтаксису. Вона підтримує різні підходи до програмування і користується популярністю у великій спільноті розробників [27]. Серверна частина реалізована за допомогою фреймворк flask.

LangChain - це фреймворк для розробки додатків на основі великих мовних моделей. Він дозволяє створювати додатки, які підтримують за наявності контекст та мають здатність опрацювання виводу великою мовною моделлю [28]. LangChain надає можливість використовувати стандартні розширювані інтерфейси та інтеграції для великих мовних моделей та внутрішньої бази даних. Він підтримує імплементацію різноманітній Use кейсів один із найголовніших Retrieval-Augmented Generation:

- збір інформації у вигляді документів;
- розділення документів на частини;
- перетворення у embeddings;
- векторизація документів;
- пошук за схожістю та відповіді;
- генерація відповіді.

Для поліпшеного розуміння схема роботи зображено на рисунку 2.1.

Рисунок 2.1 – Етапи RAG техніки

Джерело: [28]

Описуючи усі етапи детально, тоді збір інформації включатиме в себе наступні етапи:

- отримуємо документи за допомогою DataLoaders [29]: Система використовує файл sitemap.xml для отримання переліку всіх сторінок та URL-адрес веб-сайту. Цей файл, як правило, містить структуру сайту та URL-адреси його сторінок;
- аналіз структури сторінок: Після отримання списку URL-адрес сторінок система може переходити до кожної сторінки для отримання текстового вмісту. Цей підхід застосовується, наприклад, для відображення відео на прикладі сайту OpenAI, і може бути використаний для будь-якого веб-сайту.

Наступним є розділення сторінок на документи. Після отримання текстового вмісту кожної сторінки, система поділяє кожну сторінку на менші документи або частини. Це робиться за допомогою текстового розбивача (text splitter), і кожен документ отримується з відомим контекстом сторінки.

Векторизація документів відбувається так, кожен отриманий документ (частина сторінки) конвертується в числовий вектор. Це може виконуватися за допомогою векторизації, наприклад, використовуючи embeddings з OpenAI або інших методів [30]. Векторизовані документи (embeddings) зберігаються в "векторному сховищі" (vector store), що є базою знань системи. Це дозволяє ефективно зберігати та отримувати вектори для подальшого використання. Цей етап дозволяє системі отримати і структурувати інформацію з веб-сайту, готуючи її до подальших етапів взаємодії з користувачем. Далі ми можемо перейти до етапу взаємодії з користувачем та отримання відповідей на його запитання. Пошук за схожістю та відповіді:

- коли користувач вводить питання, це питання також векторизується за допомогою того ж методу embeddings, що і документи;

- здійснюється пошук векторних представлень, які найбільш схожі на вектор питання. Зазвичай використовується метод семантичного пошуку або схожості;
- визначаються найбільш схожі документи, і вони використовуються як контекст для наступного етапу.

Етап "Пошук за схожістю та відповіді" поліпшує точність відповідей, використовуючи векторизацію питань та пошук схожих документів для визначення найбільш відповідного контексту. Наступним є генерація відповіді.

Взявши найбільш схожі документи, система використовує їх як контекст для подальшого питання до великої мовної моделі, якою може бути, наприклад, davinchi від OpenAI [31]. Мовний модель генерує відповідь на питання, використовуючи отриманий контекст. Може використовуватися генерація тексту з врахуванням лінгвістичних та семантичних аспектів. І в результаті отримуємо представлення відповіді користувачеві. Система повертає користувачеві згенеровану відповідь та вказує джерело, з якого взято інформацію. Це може бути URL сторінки чи інша інформація, яка дозволяє користувачеві перевірити джерело.

Цей процес дозволяє системі відповідати на запитання користувача, використовуючи інформацію, яку вона зібрала з веб-сайту та зберігає в своєму векторному сховищі. Головна ідея полягає в тому, що система використовує великий мовний модель для генерації контекстно адаптованих відповідей на основі запитань користувачів та інформації з власної бази знань.

RAG (Retrieval-Augmented Generation) - це техніка, яка дозволяє розширювати знання мовних моделей за рахунок додаткових даних, наприклад приватних чи більш актуальних на даний період часу [32]. Вона використовується для поліпшення здатностей моделей працювати з інформацією, яка з'явилася після їх навчання. Процес RAG включає індексацію, зберігання даних, витягування та генерацію відповідей.

3. МОДЕЛЮВАННЯ

Після проведення аналізу предметної області, визначення актуальності та формулювання мети та завдань для реалізації проекту, наступним етапом є моделювання додатку. Ми використовуємо нотацію IDEF0 для розробки бізнес-процесів та детально описуємо послідовність етапів, що обробляють запити, відображають результати та виконують пошук інформації.

3.1 Структурно-функціональне моделювання в нотації IDEF0

Після ретельного аналізу предметної області, було визначено структуру основних завдань проекту, сформульовано функціональні та нефункціональні вимоги, розглянуто методи дослідження та розроблено план робіт, враховуючи критичний час та можливі фактори затримок.

На початковому етапі використовувалася нотація IDEF0 для створення діаграм [33]. IDEF0, як функціональна модель, оперує інформаційно-матеріальним потоком, організаційною структурою управління та діяльністю команди проекту. Ця нотація відзначається деталізованим описом бізнес-процесів [34] і можливістю узагальнення та деталізації даних та потоків інформації. Проте серед недоліків можна відмітити ускладнену читабельність діаграм, численні розгалуження на різних рівнях та труднощі у зв'язуванні декількох процесів, представлених у різних моделях організації. Для проекту " Мобільний додаток підтримки надання інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту " використовуються вхідні дані, управління, механізми та вихідні дані. Контекстна діаграма проекту подана на рисунок 3.1.

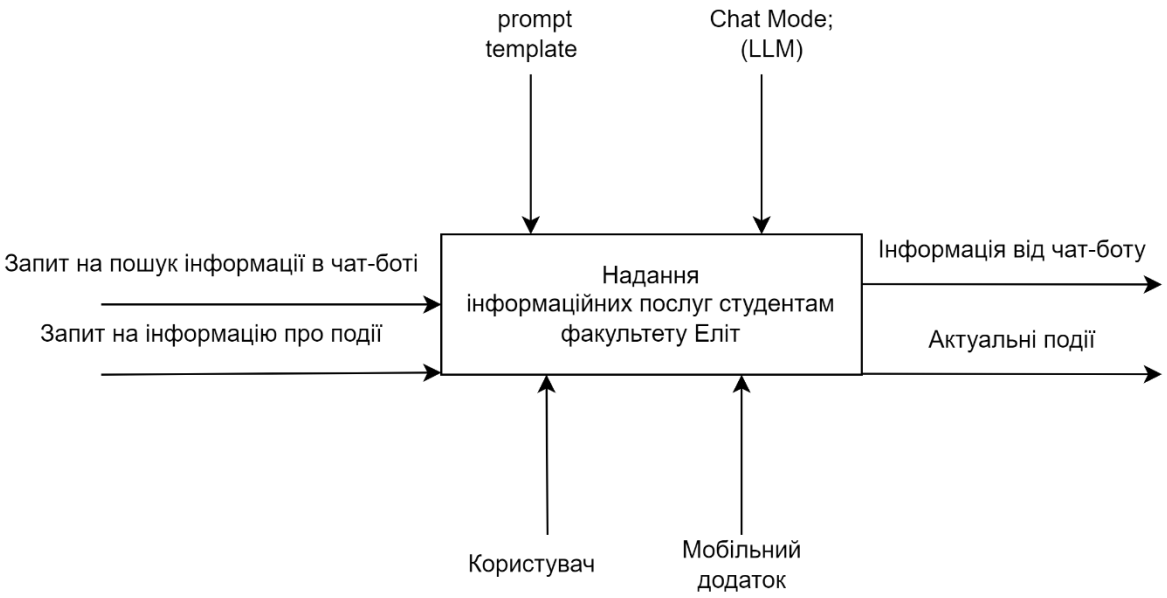

Рисунок 3.1. – Контекстна діаграма IDEF0

Джерело: побудовано автором

Розглядаючи діаграму IDEF0, можна визначити, що під час процесу надання інформаційних послуг студентам факультету ЕлІТ існує користувач, який формулює запит, та мобільний додаток, який забезпечує надання інформації. Після входу в додаток користувач має дві можливості отримання інформації: через інформаційні події та за допомогою чат-бота для пошуку необхідної інформації. Процеси у чат-боті керуються LLM-частиною модуля та створенням промтів, які відповідають запитанням для отримання потрібної інформації. Основними об'єктами процесу є користувач та мобільний додаток. У результаті роботи додатку користувач отримує або інформацію від чат-бота згідно з його індивідуальним запитом, або актуальні події. Після проведення розкладання діаграми А-0 на основні процеси проекту, було необхідно деталізувати кожен із цих процесів. Для досягнення цієї мети було виконано розкладання конкретного процесу на більш дрібні етапи. Кожне під завдання генерує результат, який слугує вхідними даними для наступного етапу. На рисунку 3.2 представлена діаграма декомпозиції.

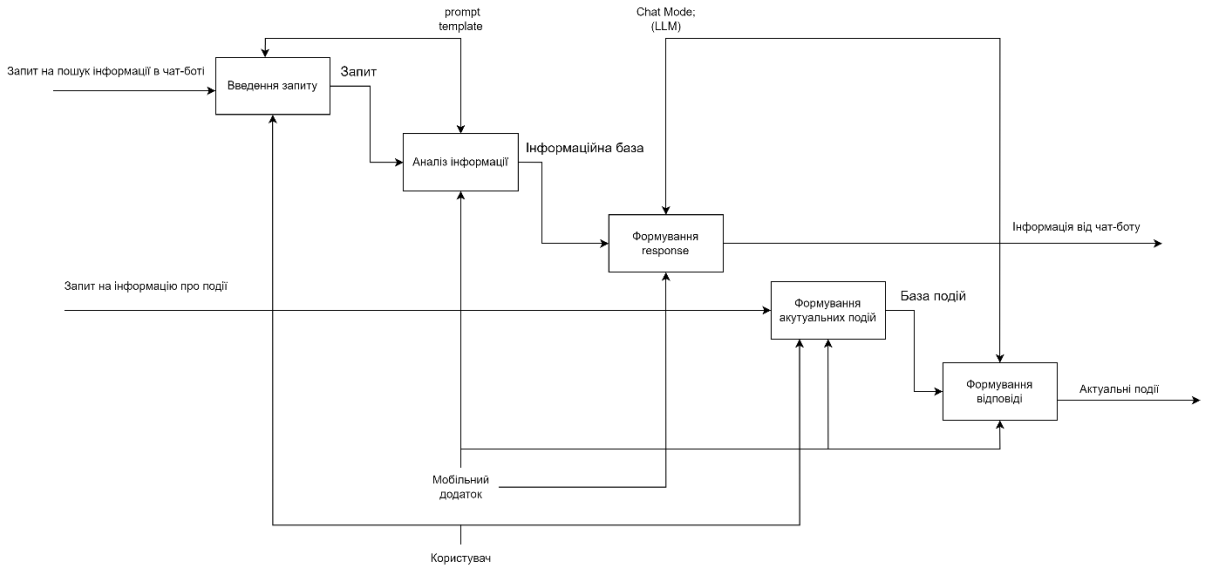

Рисунок 3.2. – Діаграма декомпозиції 1 рівня

Джерело: побудовано автором

3.2 Діаграми UML

UML (Unified Modeling Language)[35] діаграми послідовності та активності є потужними інструментами для моделювання та візуалізації робочих процесів в інформаційних системах. Вони допомагають розробникам краще розібратися в логіці системи, ідентифікувати ключові етапи взаємодії та уточнити дії, необхідні для досягнення конкретних цілей. Розглянемо кожен тип діаграм окремо та визначимо, як вони вносять цінність у наш проєкт.

UML діаграма послідовності є діаграмою що дозволяє візуально зобразити взаємодії між різними об'єктами або компонентами системи в часі. Кожна взаємодія представляє собою послідовність повідомлень, що передаються між об'єктами, і дозволяє розглядати систему з точки зору виконання конкретного сценарію. Зображена на рисунку 3.3.

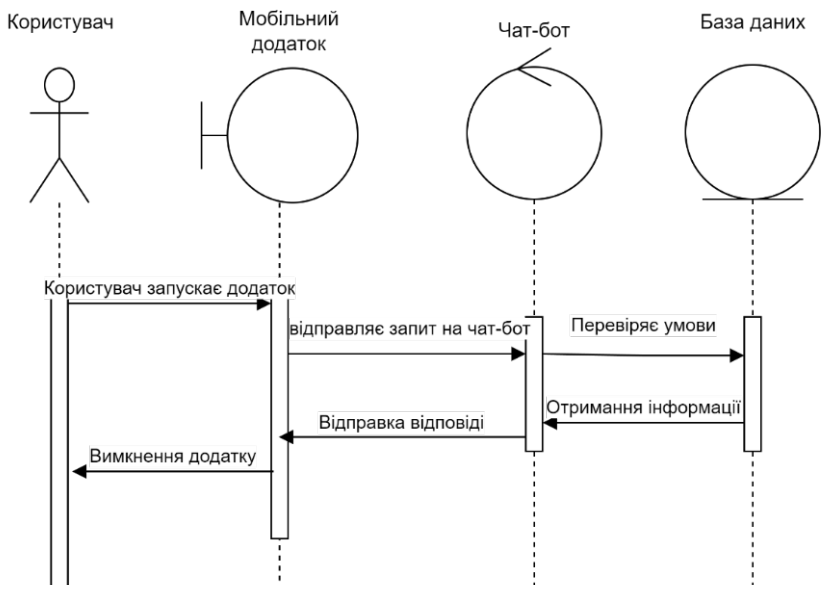

Рисунок 3.3 – Діаграма активності

Джерело: побудовано автором

UML Діаграма активності відтворює бізнес-процеси та дії, представляючи їх у вигляді структурованих послідовностей. Ця діаграма дозволяє деталізувати необхідні дії для досягнення конкретних цілей та відображає стан системи на різних етапах взаємодії. Подана на рисунку 3.4.

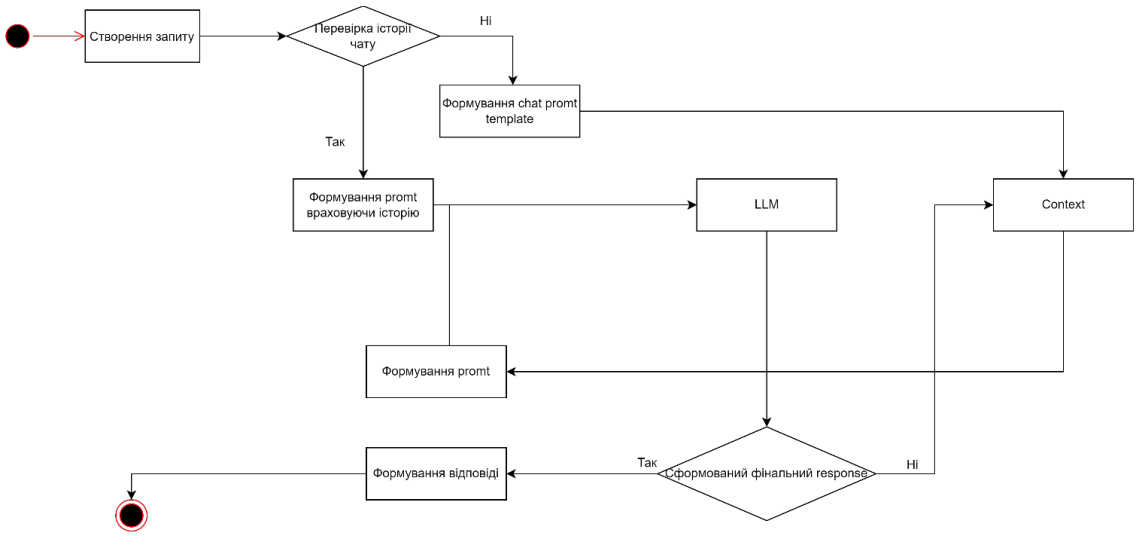

Рисунок 3.4 – Діаграма послідовності

Джерело: побудовано автором

3.3 Діаграма Use Case

Після проведення моделювання та опису бізнес-процесів для майбутнього мобільного додатку, було вирішено розробити діаграму варіантів використання. За допомогою діаграми Use Case системні та програмні вимоги до розроблюваної інформаційної системи графічно представлені. Дана діаграма дозволяє визначити, як користувачі будуть взаємодіяти з системою та які функціональності вони можуть очікувати.

Use Case дозволяє схематично відобразити системні та програмні вимоги до нашої мобільного додатку. Важливими елементами є актори, зв'язки, примітки та механізми розширення [36].

Для інтерактивності ми виділили наступні варіанти використання:

- введення запиту до чат-боту;
- перегляд подій;
- відповідь на запит;
- обробка запиту зі сторони ШІ;
- отримання інформації про події;
- зберігання інформації про події;
- отримання інформації з БД.

На діаграмі Use Case ключовими акторами є користувач, OpenAI, векторна база даних і база даних подій. На наступному етапі, розглядаючи діаграму варіантів використання, ми враховуємо актора-користувача, який взаємодіє з мобільним додатком. Рисунок 3.5 містить повний перелік можливих дій, які може виконувати користувач в додатку, а також взаємодію з OpenAI, векторною базою даних та базою даних подій.

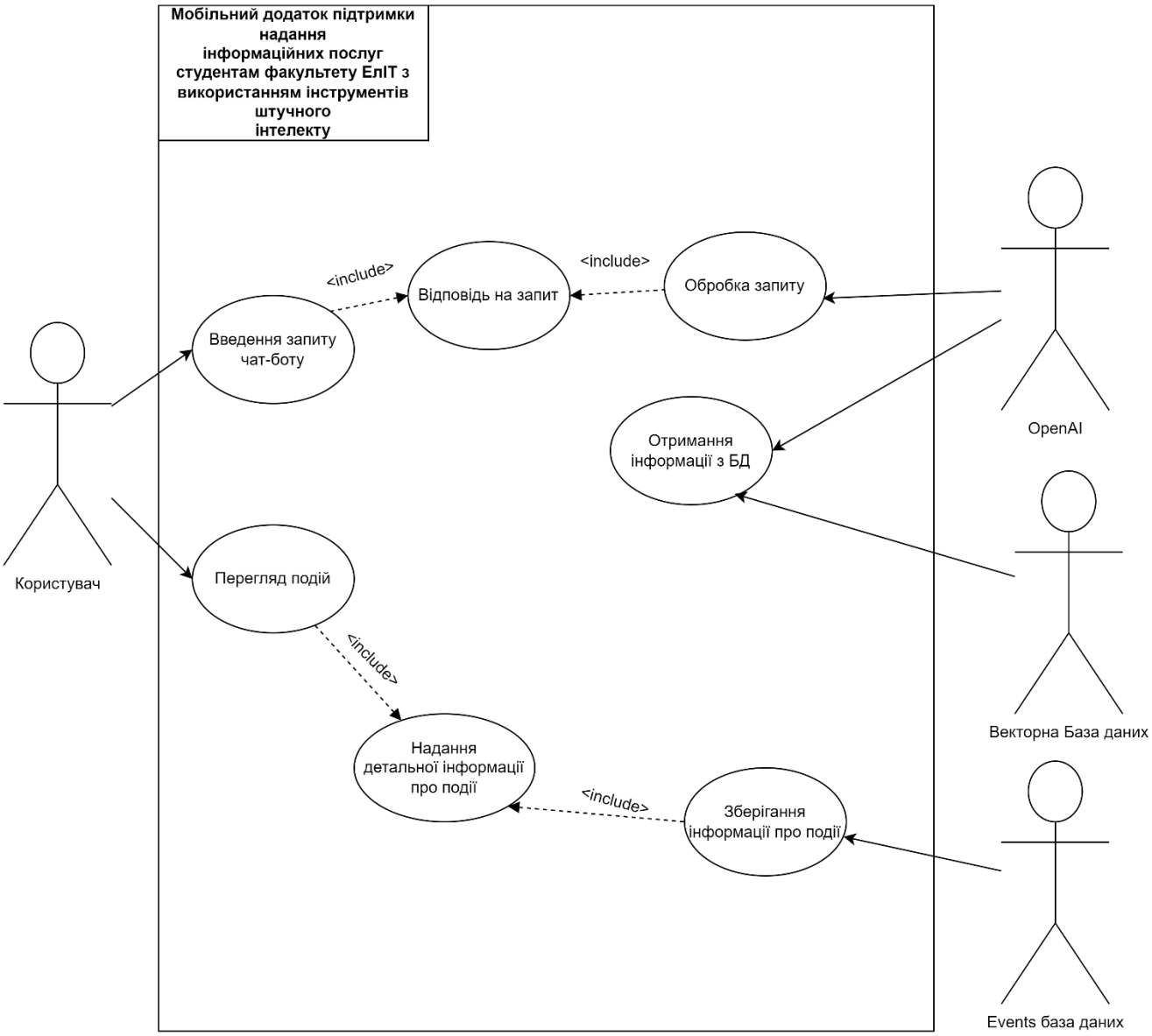

Рисунок 3.5 – Діаграма Use Case

Джерело: побудовано автором

4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

4.1 Архітектура мобільного додатку

Процес створення мобільного додатку починається з визначення архітектури програмного забезпечення за допомогою діаграми HLD (High Level Design). HLD визначає загальну структуру системи на високому рівні, надаючи абстрактне представлення компонентів продукту, які можуть бути модифіковані відповідно до змін вимог та ризиків (рис. 4.1). Цей етап дозволяє отримати стратегічний огляд та зрозуміти взаємодію ключових елементів системи перед тим, як переходити до деталей реалізації.

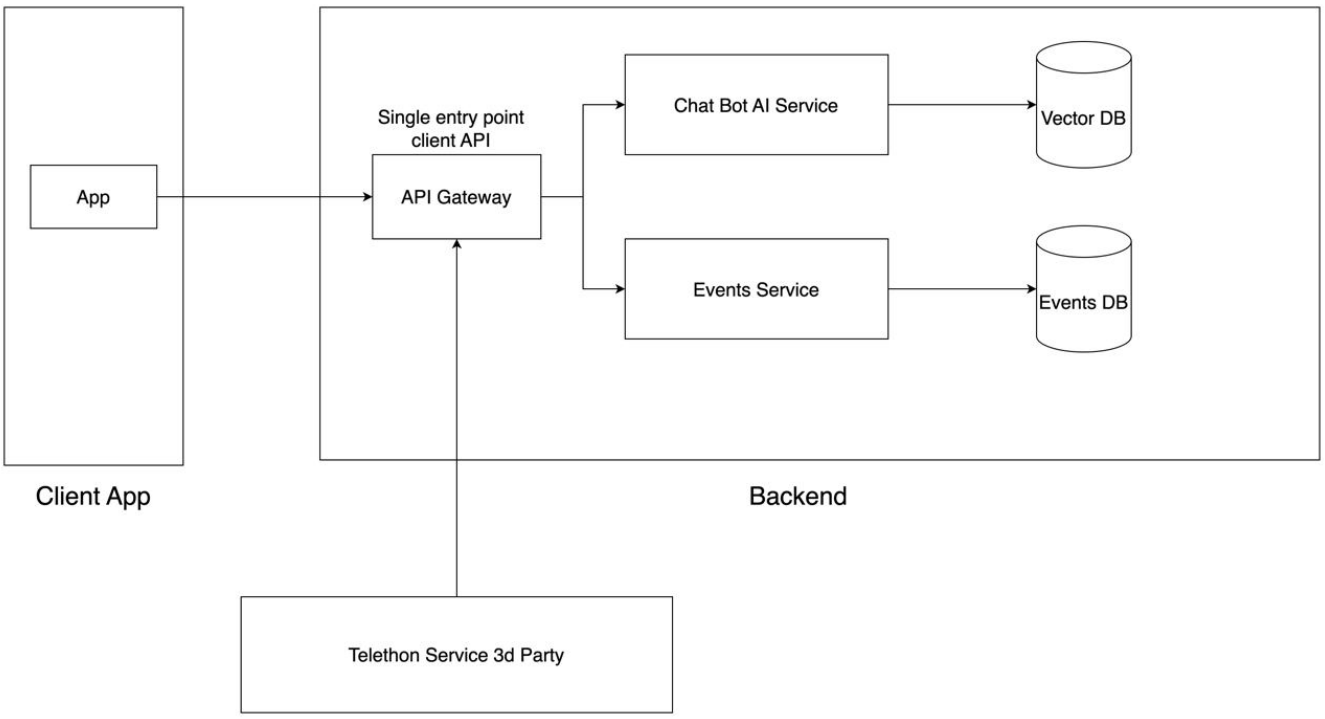

Рисунок 4.1 – Діаграма високого рівня

Джерело: побудовано автором

Системний дизайн мобільного додатку для надання інформації студентам факультет ЕлІТ представлений клієнтською та серверною частинами. Вхідною точкою до API (інтерфейс програмування застосунків) є єдиний API gateway який перенаправляє до клієнта до таких мікро-сервісів як чат-бот та події. Сервіс чат-боту використовує векторну базу даних у якості retrieval-системи. У свою чергу, сервіс подій використовує реляційну базу даних, де зберігаються інформація про події, синхронізована із зовнішнім сервісом.

4.2 Програмна реалізація мобільного додатку

Представлений фрагмент програми на мові програмування Python, яка взаємодіє з базою даних для отримання та виведення інформації про події. Давайте розглянемо, що він робить крок за кроком:

- імпортує клас Event з модуля database пакету app.events.model;
- визначає функцію print_all_events();
- здійснює запит до бази даних, використовуючи метод query.all(), для отримання всіх записів (подій) з таблиці, пов'язаної з класом Event.

```
def print_all_events():
    events = Event.query.all()
    for event in events:
        print(f"Event ID: {event.id}")
        print(f"Description: {event.description}")
        print(f"Date: {event.date}")
        print(f"Location: {event.location}")
        print(f"Time: {event.time}")
        print(f"Link: {event.link}")
        print(f"Additional Details: {event.additional_details}")
        # print(f"Picture: {event.picture}")
        print(f"Created At: {event.created_at}")
        print(f"Title: {event.title}")
```

Отже, функція get_all_events_json() повертає список всіх подій з бази даних у форматі JSON, готовий для використання в веб-застосунку. Це використано для забезпечення даними API для фронтенду, які споживають ці дані у форматі JSON.

```
from flask import jsonify
from app.events.model.database import Event
def get_all_events_json():
    events = Event.query.all()
    return jsonify([event.to_dict() for event in events])
```

Використання Flask в додатка, що відповідає за конфігурацію бази даних та визначення моделі для подій (Event) де основним елементами коду є:

- імпортується необхідні бібліотеки, такі як os, SQLAlchemy для роботи з базою даних, і dotenv для завантаження змінних оточення з файлу .env.;
- завантажуються змінні оточення для конфігурації бази даних, такі як ім'я користувача, пароль, порт, хост та ім'я схеми бази даних;
- ініціалізується об'єкт db з SQLAlchemy, який буде використовуватися для роботи з базою даних;
- визначається функція init_events_db, яка конфігурує з'єднання з базою даних і ініціалізує додаток Flask.
- визначається клас Event, який є моделлю для таблиці бази даних, що містить різні поля, такі як опис події, дата, місце, час, посилання та інші.

У класі Event визначається метод to_dict(), який конвертує об'єкт ORM (об'єкт моделі бази даних) в словник, щоб його можна було легко спеціалізувати в JSON.

```
user = os.getenv("SQL_USER")
password = os.getenv("SQL_PASS")
port = os.getenv("SQL_PORT")
host = os.getenv("SQL_HOST")
schema_name = os.getenv("SCHEMA_NAME")
```

```
    db = SQLAlchemy()
    def init_events_db(app):
        app.config[
            "SQLALCHEMY_DATABASE_URI"
        ] =
f"mysql+pymysql://{user}:{password}@{host}:{port}/{schema_name}"
        db.init_app(app)
    class Event(db.Model):
        __tablename__ = "event"
        id = db.Column(db.Integer, primary_key=True)
        description = db.Column(db.String(255))
        date = db.Column(db.String(25))
        location = db.Column(db.String(255))
        time = db.Column(db.String(25))
        link = db.Column(db.String(100))
        additional_details = db.Column(db.String(255))
        picture = db.Column(db.JSON)
        created_at = db.Column(db.DateTime)
        title = db.Column(db.String(50))
```

Наступному відноситься до завантаження та обробки текстових даних для подальшого використання, клас DataLoader. Конструктор класу (__init__) приймає список URL-адрес, які мають бути завантажені. Метод load_data використовує клас UnstructuredURLLoader з бібліотеки langchain для завантаження даних з вказаних URL-адрес. Відправник HTTP-заголовка (headers) вказує, що це запит від користувача з агентом користувача "value".

Метод split_data використовує клас RecursiveCharacterTextSplitter для розділення завантажених текстових даних на фрагменти або документи. Параметри chunk_size та chunk_overlap визначають розмір фрагментів та їхнє перекриття.

```
    class DataLoader:
        def __init__(self, urls):
            self.urls = urls
        def load_data(self):
            loader      =      UnstructuredURLLoader(urls=self.urls,
headers={"User-Agent": "value"})
            data = loader.load()
            return data
        def split_data(self, data):
            text_splitter = RecursiveCharacterTextSplitter(
                chunk_size=1000, chunk_overlap=200
            )
            splits = text_splitter.split_documents(data)
            return splits
```

Код, що використовує шаблон проектування "Одинак" (Singleton) для створення єдиного екземпляру класу ChromaDBInstance, який ініціалізується та завантажує дані при його створенні. __new__ та __init__ методи:

- Метод __new__ визначається для створення єдиного екземпляру класу. Якщо екземпляр ще не існує, він створюється, ініціалізується та повертається;
- Метод __init__ викликається при ініціалізації екземпляру, але в даному випадку він не викликається, так як вже є __new__.

Get_instance метод повертає існуючий екземпляр класу, якщо він вже існує, або створює новий, якщо екземпляр відсутній. Тоді як initialize_vectorstore метод викликається при створенні екземпляру та викликає два інші методи: load_data та create_vectorstore. Load_data метод створює екземпляр класу DataLoader, завантажує дані та розбиває їх на фрагменти. Create_vectorstore метод створює об'єкт Chroma з використанням отриманих фрагментів та embeddings з OpenAI.

```
class ChromaDBInstance:
    _instance = None
    def __new__(cls):
        if cls._instance is None:
            cls._instance = super().__new__(cls)
            cls._instance.initialize_vectorstore()
        return cls._instance @classmethod
    def get_instance(cls):
        if cls._instance is None:
            cls._instance = cls()
        return cls._instance
    def load_data(self):
        data_loader = DataLoader(list_urls)
        data = data_loader.load_data()
        self.splits = data_loader.split_data(data)
```

URL-адрес, які представляють собою веб-ресурси або сторінки в Інтернеті на яких знаходиться інформація. Кожен елемент цього списку є рядком, що містить URL-адресу конкретного веб-ресурсу або сторінки. У нашому випадку ці URL-адреси, здається, пов'язані з факультетом ЕлІТ Сумського Державного Університету та містять інформацію про людей, новини, деканат та інші аспекти цього факультету.

```
list_urls = [
    "https://elit.sumdu.edu.ua",
    "https://itp.elit.sumdu.edu.ua/people",
    "https://itp.elit.sumdu.edu.ua/news",
    "https://itp.elit.sumdu.edu.ua/people/baranova-iryna-
volodymyrivna.html",
    "https://itp.elit.sumdu.edu.ua/people/nahornyy-volodymyr-v-
iacheslavovych.html",
    "https://elit.sumdu.edu.ua/uk/pro-fakultet/dekanat.html",
```

```
]
```

Flask додаток, який використовує базу даних SQLAlchemy для збереження інформації про події. Додаток також взаємодіє з Telegram, слідкуючи за новими повідомленнями в певному каналі і зберігає інформацію про події з цих повідомлень в базі даних.

Основні етапи роботи коду:

1. Ініціалізація Flask додатку та підключення до бази даних.

2. Оголошення моделі Event для представлення даних про події у базі даних.

3. Визначення класу EventView для виведення інформації про події.

4. Створення функції `handle_new_event`, яка встановлює обробник для нових повідомлень у Telegram каналі. Цей обробник відповідає за обробку нових подій та зберігання їх в базі даних.

5. Запуск Telegram клієнта та Flask додатку.

Під час обробки нового повідомлення Telegram:

- Витягується текст повідомлення та опрацьовується для отримання інформації про подію.

- Якщо у повідомленні є фотографія, вона конвертується у формат base64 та зберігається в базі даних.

- Інформація про подію зберігається в базі даних за допомогою SQLAlchemy.

Цей код дозволяє динамічно оновлювати інформацію про події у базі даних за допомогою нових повідомлень, які надходять у ваш Telegram канал.

```
class Event(db.Model):
    __tablename__ = "event"
    id = db.Column(db.Integer, primary_key=True)
    description = db.Column(db.String(255))
    date = db.Column(db.String(25))
    location = db.Column(db.String(255))
    time = db.Column(db.String(25))
```

```
    link = db.Column(db.String(100))
    additional_details = db.Column(db.String(255))
    picture = db.Column(db.JSON)
    created_at = db.Column(db.DateTime)
    title = db.Column(db.String(50))
```

Надалі наведено класс евентів.

```
class EventView:
    @staticmethod
    def display_event(event):
        print("Event id:", event.id)
        print("Description:", event.description)
        print("Date:", event.date)
        print("Location:", event.location)
        print("Time:", event.time)
        print("Link:", event.link)
        print("Additional Details:", event.additional_details)
        print("Picture:", event.picture)
        print("Created At:", event.created_at)
        print("Title:", event.title)
        print("=====================================")
```

Однією з головних функція чат-боту є модель вибору мови. FormatLanguage приймає рядок language і повертає англійську мову, або українську за замовчуванням, якщо немає відповідності.

```
function formatLanguage(language: string) {
  switch (language) {
    case "en":
      return "English";
    case "ua":
      return "Ukrainian";
    default:
```

```
        return "Ukrainian";}
    }
```

Зображено клас SSUElitAPI з методом sendAIRequest. Цей метод використовує fetch для відправки POST-запиту на сервер із заданими властивостями, такими як prompt і language. Відповідь з сервера парситься у форматі JSON.

```
export class SSUElitAPI {
  async sendAIRequest(properties: HumanPrompt) {
    return      fetch(process.env.EXPO_PUBLIC_SSU_API_URL      +
"/chatbot", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        prompt: properties.prompt,
        language: formatLanguage(properties.language),
      }),
    })
```

4.3 Демонстрація роботи мобільного додатку

Мобільний додаток для студентів факультету ЕлІТ пропонує дві корисні можливості. По-перше, він забезпечує інформацію за допомогою чат-бота зі штучним інтелектом, де ви знайдете всю необхідну інформацію про факультет та актуальні події. При вході до додатку вас вітає анімоване зображення бота ЕлІТ чату, який пояснює, що ви можете запитати його про все, що пов'язано з факультетом. Зображено на рисунку 4.2.

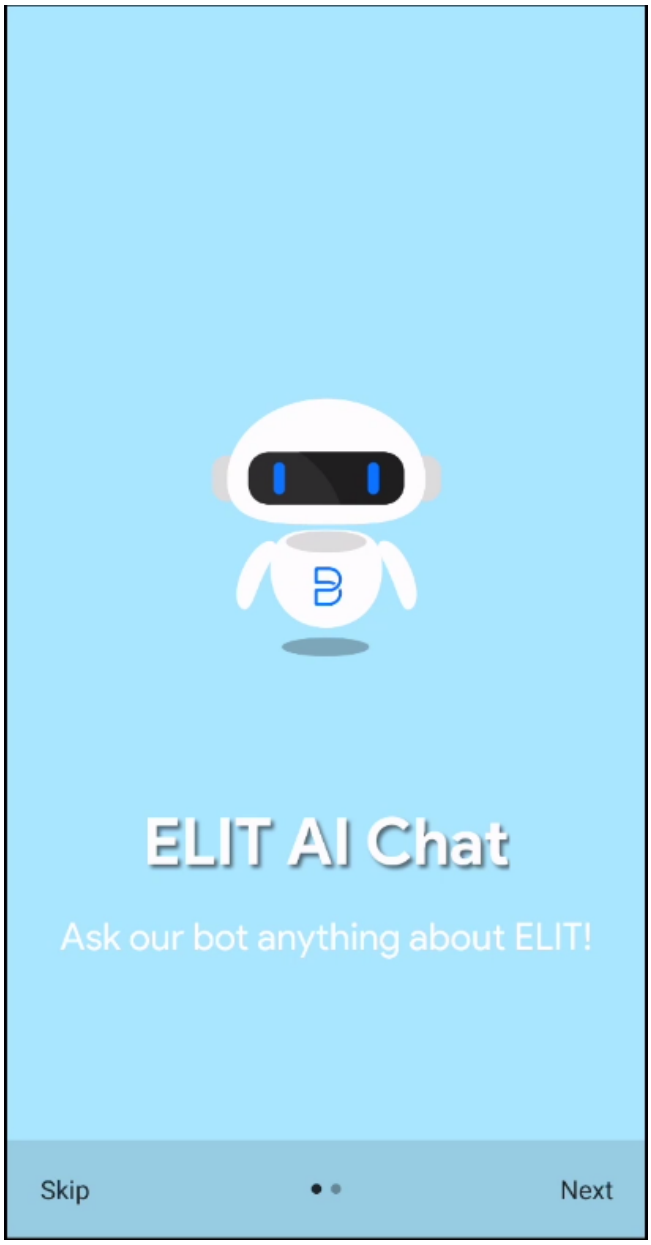

Рисунок 4.2 – Екран що з'являється при старті програми.

Джерело: побудовано автором

Потім користувач переходить до меню вибору, де він може обрати, що конкретно хоче дізнатися - чи це інформація через чат-бот, чи актуальні події. Завдяки мінімалістичному дизайну ми спрощуємо завдання студента. Кольорова гамма відображає два основні кольори факультету. У додатку доступні лише три кнопки: "Чат-бот", "Події" та "Інформація про додаток", це зображено на .рисунку 4.3.

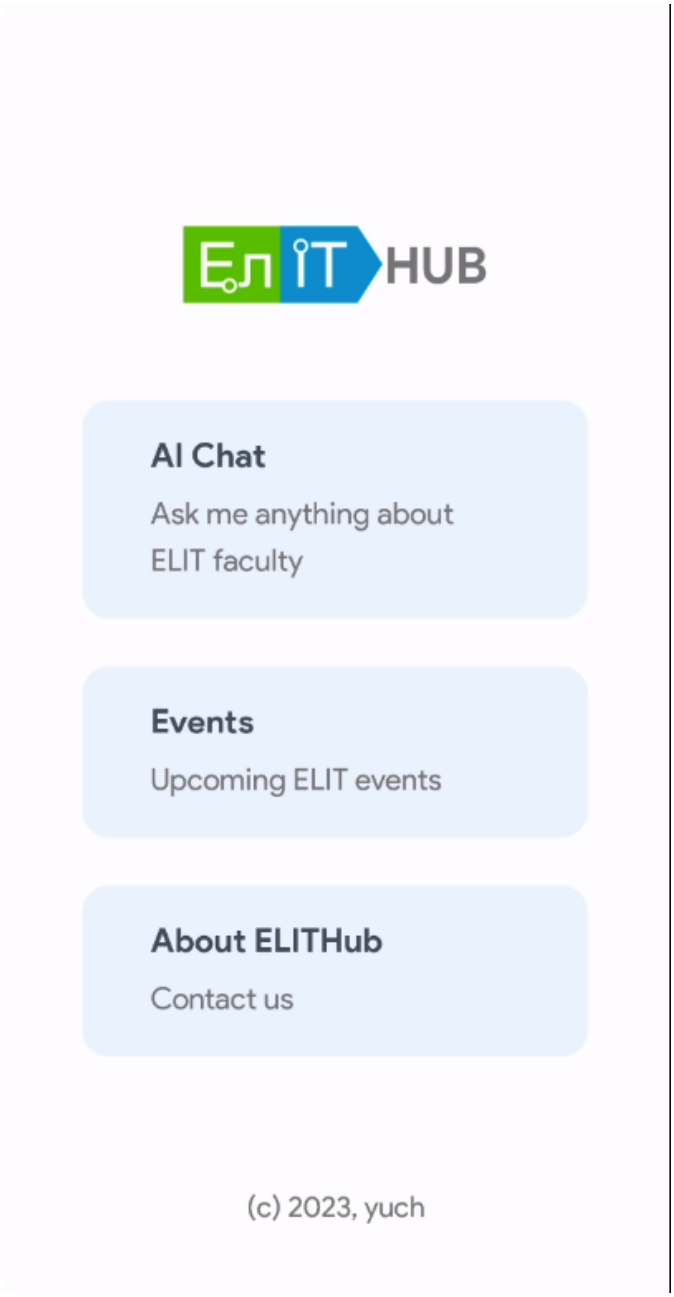

Рисунок 4.3 – Меню додатку.

Джерело: побудовано автором

Якщо обрати "Актуальні події" у меню, вас перенесе до плиток з зображеннями, де надається коротка інформація про кожну подію. Прокручуючи вправо чи вліво, користувач зможете переглядати всі актуальні події та обрати те, що цікавить. Цей процес зображений на рисунку 4.4.

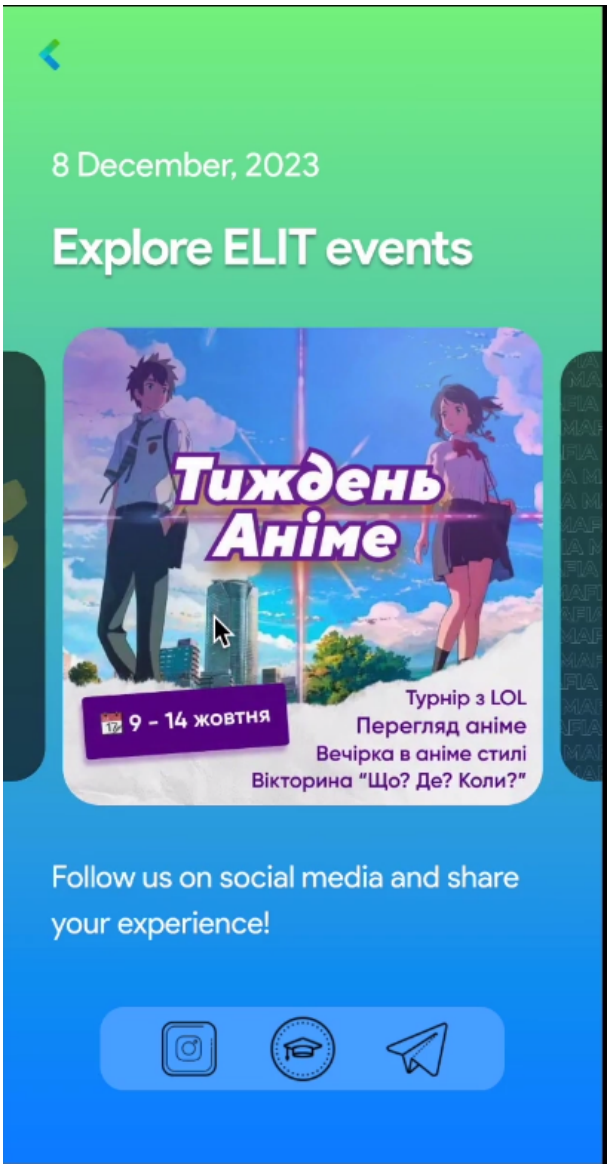

Рисунок 4.4 – Меню вибору актуальних подій.

Джерело: побудовано автором

Коли студент торкаєтеся зображення події, відкривається докладна інформація, включаючи назву, місце, дату, час і опис. Також унизу є кнопка "Приєднатися", яка перенаправляє вас на форму реєстрації на подію, якщо це потрібно. Це дає можливість швидко долучитися до події та забезпечує інформацію для аналізу, яка дозволяє університету визначити, які активності є найбільш популярними серед студентів. Цей процес показаний на рисунку 4.5.

Рисунок 4.5 – Більше інформації про події.

Джерело: побудовано автором

Натисканням на стрілку у лівому верхньому куті, ви повертаєтеся назад, що дає можливість переміщатися між подіями або повертатися до головного меню. Крім того, при вперше переходженні з головного меню до розділу з чат-ботом, користувачеві пропонується вибрати мову для спілкування таку як англійську чи українську., після чого бот автоматично визначає мову для надання відповідей студенту. Це зображено на рисунку 4.6.

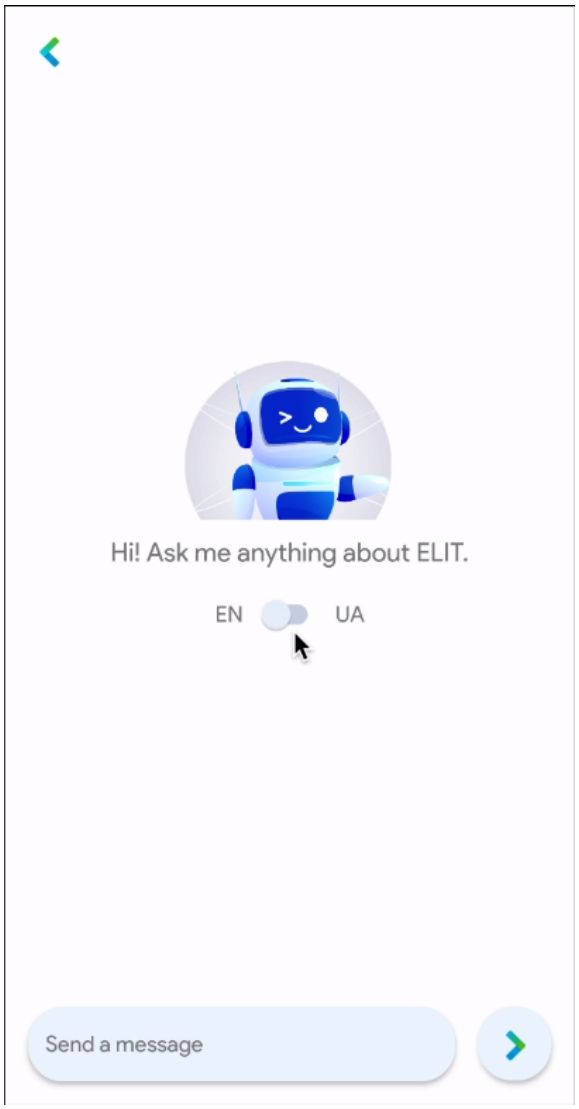

Рисунок 4.6 – Вибір мови спілкування в чат-боті.

Джерело: побудовано автором

Обираючи українську мову, користувач вводить свій запит, і через короткий проміжок часу чат-бот відображає відповідь, подібну до прикладу на рисунку 4.7. Наприклад, при введенні запиту "Мене цікавить інформація про викладача Ірину Баранову" чат-бот аналізує інформацію, отриману від студента, порівнює її з даними на сервері та формує відповідь. Таким же чином обираючи мову спілкування англійську в самому початку, чат-бот автоматично розпізнає потрібну мову для взаємодії та розпочинає діалог, готуючи відповіді.

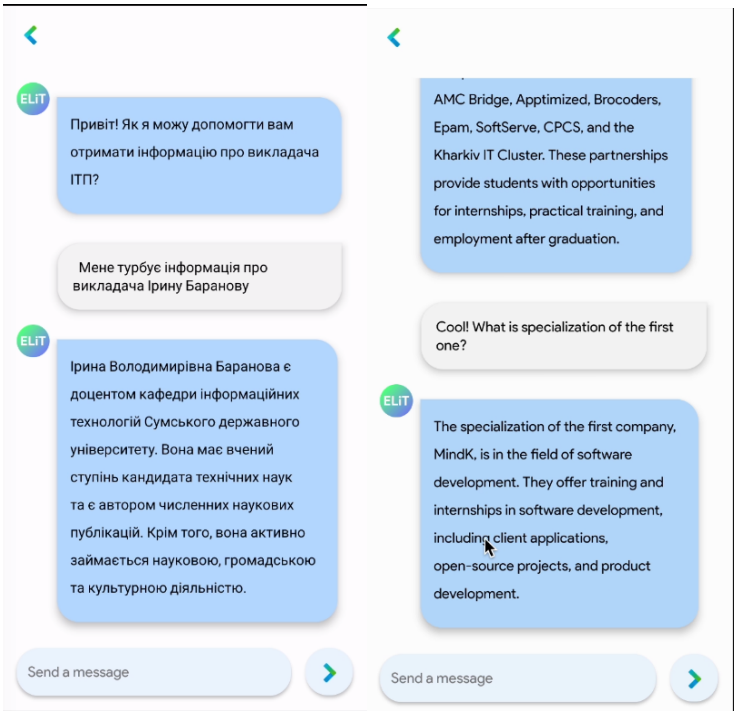

Рисунок 4.7 – Розмова з чат-ботом.

Джерело: побудовано автором

Якщо користувач хочете скопіювати інформацію, достатньо натискати на потрібну відповідь, і вона автоматично додається до буфера обміну. Цей процес зображений на рисунку 4.8.

Рисунок 4.8 – Розмова з чат-ботом.

Джерело: побудовано автором

ВИСНОВОК

У сучасному університетському середовищі, де студенти стикаються з ускладненим доступом до інформації, мобільний додаток із використанням інструментів штучного інтелекту може ефективно вирішити проблеми та покращити якість навчання. На відміну від традиційних методів, таких як веб-сайти та електронна пошта, інноваційні підходи забезпечують персоналізований підхід, інтерактивність та швидку обробку інформації.

Важливим етапом є використання штучного інтелекту, яке може ефективно покращити комунікацію зі студентом та якісно надати інформаційні послуги в зручному форматі чату. Аналіз запитів студентів, персоналізація інформаційних послуг та адаптація до різних стилів навчання є ключовими перевагами використання ШІ.

Згадані переваги свідчать про важливий внесок штучного інтелекту в удосконалення освітнього процесу. Проте, враховуючи можливі недоліки, такі як конфіденційність та точність алгоритмів, важливо ретельно враховувати і балансувати ризики та вигоди при розробці та впровадженні подібних інноваційних рішень.

Під час роботи над проектом виявлено потребу в мобільному додатку для надання інформаційних послуг студентам у зручній та інтерактивній формі чат-боту. Детальний аналіз предметної області та порівняльний аналіз існуючих рішень допомогли сформулювати вимоги до системи.

Далі була проведена ідентифікація та деталізація мети проекту, встановлення функціональних та нефункціональних вимог, а також планування робіт для оцінки часових рамок проекту. Розроблено реальні кейси та сценарії використання

інформаційної системи, використовуючи діаграми IDEF0. Цей етап дозволив деталізувати завдання, визначити функціональні можливості та рівні доступу.

Висновок полягає в тому, що на підставі аналітичної інформації визначено, що використання мобільного додатка з штучним інтелектом стане ефективним і корисним інструментом для факультету ЕлІТ.

У підсумку, розроблено мобільний додаток котрий є ефективним інструментом для оптимізації надання інформаційних послуг студентам факультету ЕлІТ, спрощуючи процеси пошуку інформації та її візуальне представлення.

СПИСОК ЛІТЕРАТУРИ

1. Дубчак А., Литвиненко Я. Напрями використання штучного інтелекту за сучасних умов. 2019. № 3. С. 4-6. DOI: http://elartu.tntu.edu.ua/bitstream/lib/32876/2/IPJINU_2020_Dubchak_ADirections_of_use_artificial_64-65.pdf

2. Дем'яненко В. Системи штучного інтелекту в адаптивному навчанні 2021. № 3. С. 2-5. DOI : https://lib.iitta.gov.ua/715956/1/%D0%97%D0%B1%D1%96%D1%80%D0%BD%D0%B8%D0%BA%20%D1%82%D0%B5%D0%B7%20%D0%B7%D0%B2%D1%96%D1%82%D0%BD%D0%BE%D1%96%CC%88%202019_.pdf#page=19.

3. Петручок Ю. Штучний інтелект: чого очікувати? 2020. № 1. С. 1−5. DOI:http://elartu.tntu.du.ua/bitstream/lib/30258/2/FVT_2019_Petruchok_YArtificial_intelligence_102-103.pdf.

4. Мар'єнко, М. В., Шишкіна, М. П., & Коновал, О. А. Методологічні засади формування хмаро орієнтованих систем відкритої науки 2022. № 10. С. 57−85. DOI: https://doi.org/10.33407/itlt.v89i3.4981.

5. Bykov, V., Mikulowski, D., Moravcik, O. The use of the cloud-based open learning and research platform 2020. № 10. С. 15-35. DOI: https://doi.org/10.33407/itlt.v76i2.3706.

6. Перспективи розвитку інформаційних технологій в Україні. URL:http://www.rusnauka.com/17_AND_2010/Informatica/68784.doc.htm(дата звернення: 15.11.2023).

7. The use of mobile applications in higher education classes: URL: https://slejournal.springeropen.com/articles/10.1186/s40561-021-00159-6 (дата звернення: 17.11.2023).

8. Artificial Intelligence And Chatterbots Application In Foreign Language Learning URL: http://www.innovpedagogy.od.ua/archives/2021/32/part_2/35.pdf (дата звернення: 17.11.2023).

9. From Open Science and Datasets to AI and Discovery. Trends URL: http://doi.org/10.13140/RG.2.2.20360.70404(дата звернення: 18.11.2023).

10. Pradeep Menon. Introduction to Large Language Models and the Transformer Architecture. URL: https://rpradeepmenon.medium.com/introductionto-large-language-models-and-the-transformer-architecture-534408ed7e61 (дата звернення: 18.11.2023)..

11. Best Test Automation Frameworks For Your Project. URL: https://www.clariontech.com/blog/best-test-automation-frameworks-for-yourproject(дата звернення: 18.11.2023).

12. Khan, M., & Lulwani, M. Inspiration of Artificial Intelligence in Adult Education URL:https://doi.org/10.31219/osf.io/zjqmn(дата звернення: 19.11.2023).

13. «Корейський національний університет» https://knubot.knu.ac.kr/#none(дата звернення: 19.11.2023).

14. «Абітурієнт економічного» https://t.me/abit_knu_bot(дата звернення: 19.11.2023).

15. «University College Dublin»https://www.ucd.ie/newstudents/(дата звернення: 19.11.2023).

16. «Coursera»https://www.coursera.org(дата звернення: 19.11.2023).

17. Ding, J. Towards Openness Beyond Open Access URL:http://doi.org/10.48550/arXiv.2301.08488(дата звернення: 20.11.2023).

18. Chaka, C. (2023). Fourth industrial revolution URL:http://rptel.apsce.net/index.php/RPTEL/article/view/2023-18002(дата звернення: 20.11.2023).

19. Де пояснювати роботу алгоритму та описувати бізнес-процеси URL:https://robotdreams.cc/uk/blog/198-udobnyh-servisov-dlya-sozdaniya-blok-

shem(дата звернення: 20.11.2023). Large language model URL: https://www.elastic.co/what-is/large-language-models(дата звернення: 19.11.2023).

20. What are Tokens? URL: https://learn.microsoft.com/en-us/semantic-kernel/prompt-engineering/tokens(дата звернення: 19.11.2023).

21. What is a vector database? URL:https://learn.microsoft.com/en-us/semantic-kernel/memories/vector-db(дата звернення: 19.11.2023).

22. Управління IT-проектами URL: https://core.ac.uk/download/pdf/84838992.pdf (дата звернення: 20.11.2023).

23. React Native URL:https://reactnative.dev (дата звернення: 21.11.2023).

24. React Navigation URL: https://reactnavigation.org (дата звернення: 21.11.2023).

25. NativeWind URL: https://www.nativewind.dev (дата звернення: 22.11.2023).

26. Lightweight Lottie URL: https://lottiefiles.com (дата звернення: 22.11.2023).

27. Building applications with LLMs through composability URL:https://pypi.org/project/langchain/ (дата звернення: 22.11.2023).

28. LangChain Quickstart URL: https://python.langchain.com/docs/get_started/quickstart (дата звернення: 22.11.2023).

29. Sitemaps XML format URL:https://platform.openai.com/docs/guides/embeddings (дата звернення: 23.11.2023).

30. Embeddings URL:https://platform.openai.com/docs/guides/embeddings (дата звернення: 24.11.2023).

31. OpenAI URL:https://blog.openai.com/openai-supporters/ (дата звернення: 25.11.2023).

32. Retrieval Augmented Generation (RAG) URL: https://www.promptingguide.ai/techniques/rag (дата звернення: 26.11.2023).

33. Створення схем IDEF0 URL: https://support.microsoft.com/uk-ua/office/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D1%81%D1%85%D0%B5%D0%BC-idef0-ea7a9289-96e0-4df8-bb26-a62ea86417fc(дата звернення: 26.11.2023).

34. Що idef0? URL:https://www.comindware.ru/blog/азы-моделирования-в-idef0/(дата звернення: 27.11.2023).

35. Що таке UML-діаграми?https://evergreens.com.ua/ua/articles/uml-diagrams.html(дата звернення: 27.11.2023).

36. Use Case Diagram Tutorial (Guide with Examples) URL:https://creately.com/blog/diagrams/use-case-diagram-tutorial/(дата звернення: 27.11.2023).

ДОДАТОК А

ПЛАНУВАННЯ РОБІТ

З огляду на стрімкий розвиток технологій та потребу студентів у доступі до інформації, виникає необхідність у створенні мобільного додатка. Цей додаток має на меті полегшити доступ до достовірної інформації для студентів факультету, пропонуючи швидкий і зручний спосіб отримання потрібної інформації через чат-інтерфейс. Такий підхід враховує актуальні тенденції у сфері інформаційних технологій та відповідає потребам сучасного студентського співтовариства.

Додаток також служитиме інструментом мотивації для студентів, спонукаючи їх до активної участі та розвитку в нових напрямках. Цей підхід дозволить факультету розвиватися і сприяти професійному зростанню студентів за допомогою актуальної інформації.

Отже, створений мобільний додаток, базований на інструментах штучного інтелекту, стане невід'ємним засобом для студентів факультету ЕлІТ, надаючи їм можливість отримати персоналізовану інформацію та спонукаючи до активної участі у діяльності факультету.

Деталізація мети методом SMART. Формулювання чіткої та зрозумілої мети на стадії концептуального проектування є важливою умовою успіху всього проекту. Результати розгортання мети проекту за допомогою методу SMART подані в таблиці А.1.

Таблиця А.1 – Деталізація мети проекту методом SMART

Specific (Конкретна)	Створити мобільний додаток підтримки надання інформаційних послуг студентам факультету ЕлІТ з використанням інструментів штучного інтелекту.
Measurable (Вимірювана)	Результатом роботи є розроблений мобільний додаток для надання інформаційних послуг.

Продовження таблиці А.1

Achievable (Досяжна)	Для успішного виконання проекту використано React Native, React Navigation, NativeWind, Lottie Animations, фреймворк Langchain, OpenAI та векторне сховище Chroma.
Relevant (Реалістична)	Розроблений мобільний додаток надає зручний та швидкий доступ до необхідної інформації через чат-інтерфейс.
Time-framed (Обмежена у часі)	Термін виконання проекту обговорений із замовником і встановлений на два місяці.

Розробка мобільного додатку для підтримки інформаційних послуг студентам факультету ЕлІТ за допомогою інструментів штучного інтелекту включає аналіз робіт та розроблення плану. Структура WBS (Work Breakdown Structure) визначає елементи проекту у вигляді графічної ієрархії, яка організована за рівнями та пов'язана з продуктом проекту. На найвищому рівні ієрархії знаходиться продукт проекту, а дії для його досягнення представлені на наступному рівні. Далі проводиться детальна декомпозиція завдань до елементарно простих робіт, які визначаються чітким результатом, відповідальною особою та обчислюваними термінами виконання і витратами робочого часу. Рисунок А.1 демонструє приклад WBS для розробки інформаційної системи оцінки професійних досягнень студентів факультету ЕлІТ за допомогою інструментів штучного інтелекту.

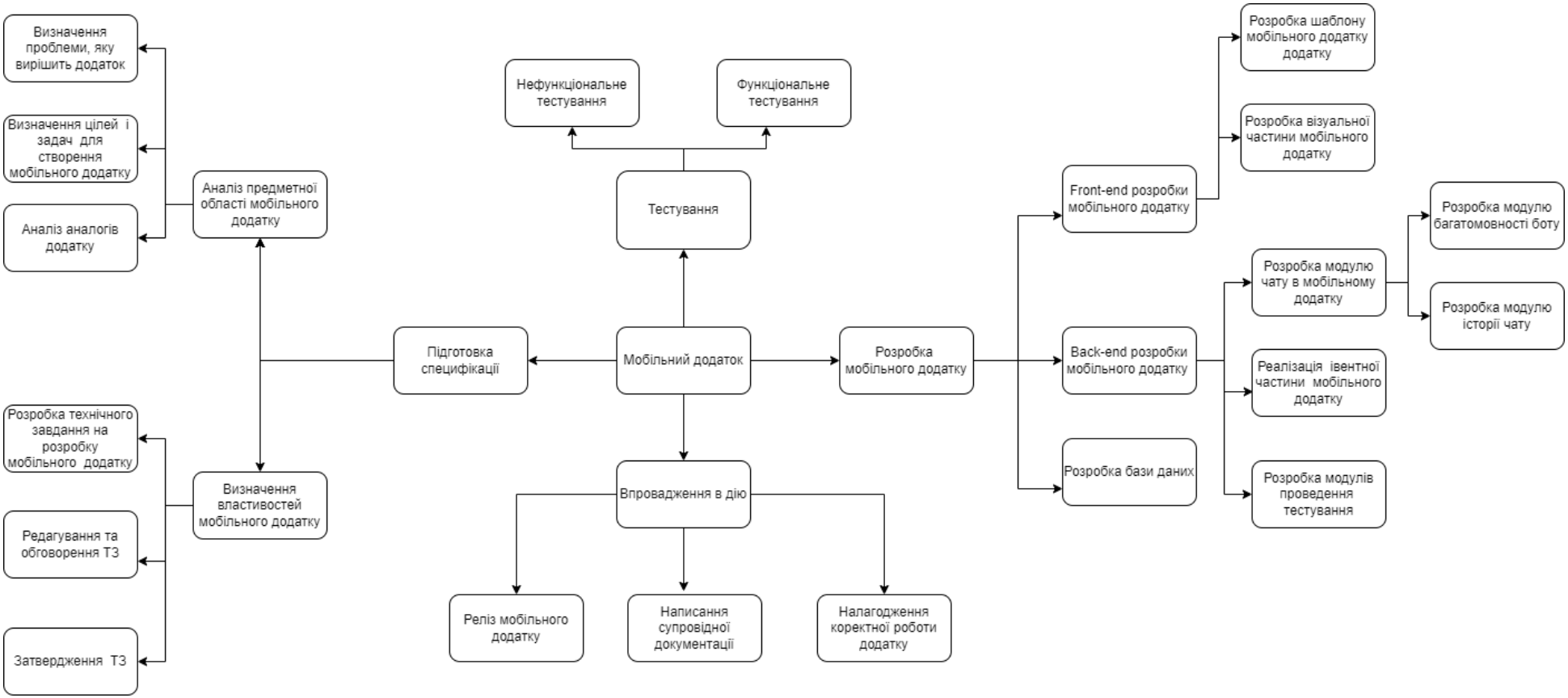

Рисунок А.1 – WBS-структура робіт проекту

Створення структури виконавців – це наступний етап після розкладання процесів у розробці мобільного додатку. Організаційна структура виконавців (OBS) представляє графічний вигляд відповідальних осіб, які беруть участь у проекті та відповідають за виконання конкретних завдань. На рисунку А.2 зображена організаційна структура планування робіт проекту, де учасників описано у таблиці А.2.

Таблиця А.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Чімирис Ю.С.	Виконує frond-end та back-end розробку
Проектувальник	Чімирис Ю.С.	Виконує проектування та розробляє мобільний додаток
Тестувальник	Чімирис Ю.С.	Проводить тестування мобільного додатку
Керівник проекту	Нагорний В.В.	Формує завдання на розробку проекту
Менеджер проекту	Чімирис Ю.С.	Відповідає за дотримання термінів, розподіл ресурсів та завдань між учасниками проекту. Виконує збір та аналіз даних.

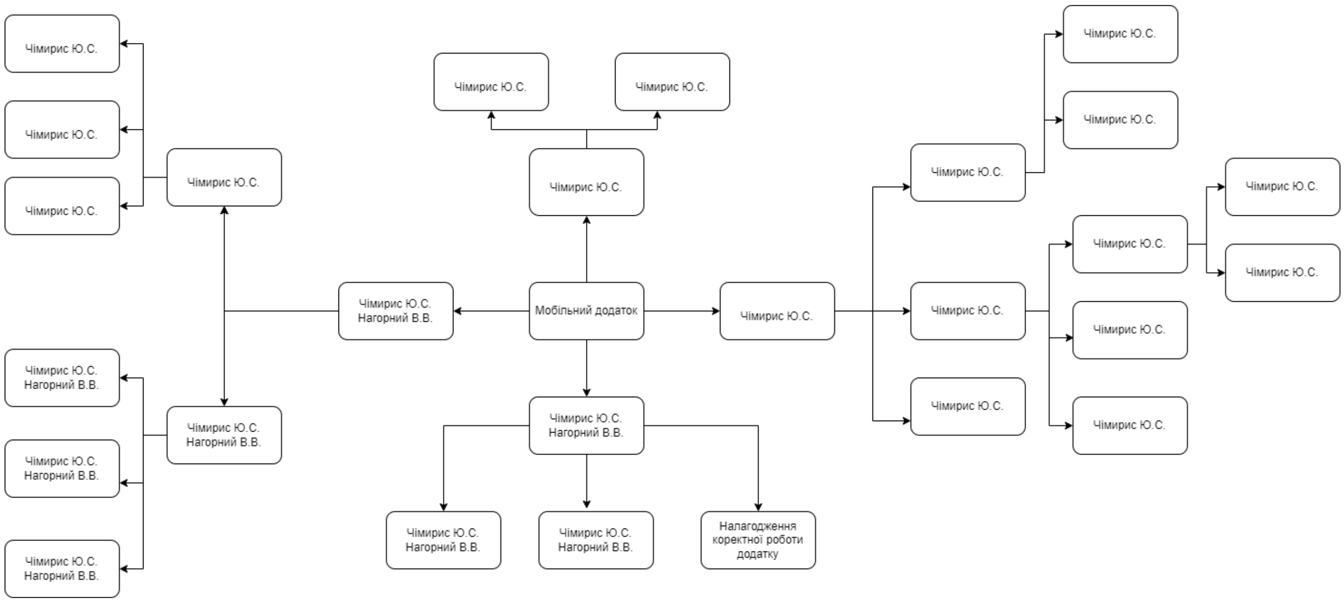

Рисунок А.2 – OBS-структура робіт проекту

Діаграма Ганта. Розробка графіка виконання робіт є ключовим етапом у роботі над проектом, представляючи собою календарний план з реальним розподілом дат. Цей план дозволяє отримати достовірне уявлення про тривалість кожного етапу проекту, з урахуванням обмежень у ресурсах.

Календарний графік проекту представлено на рисунку А.3.

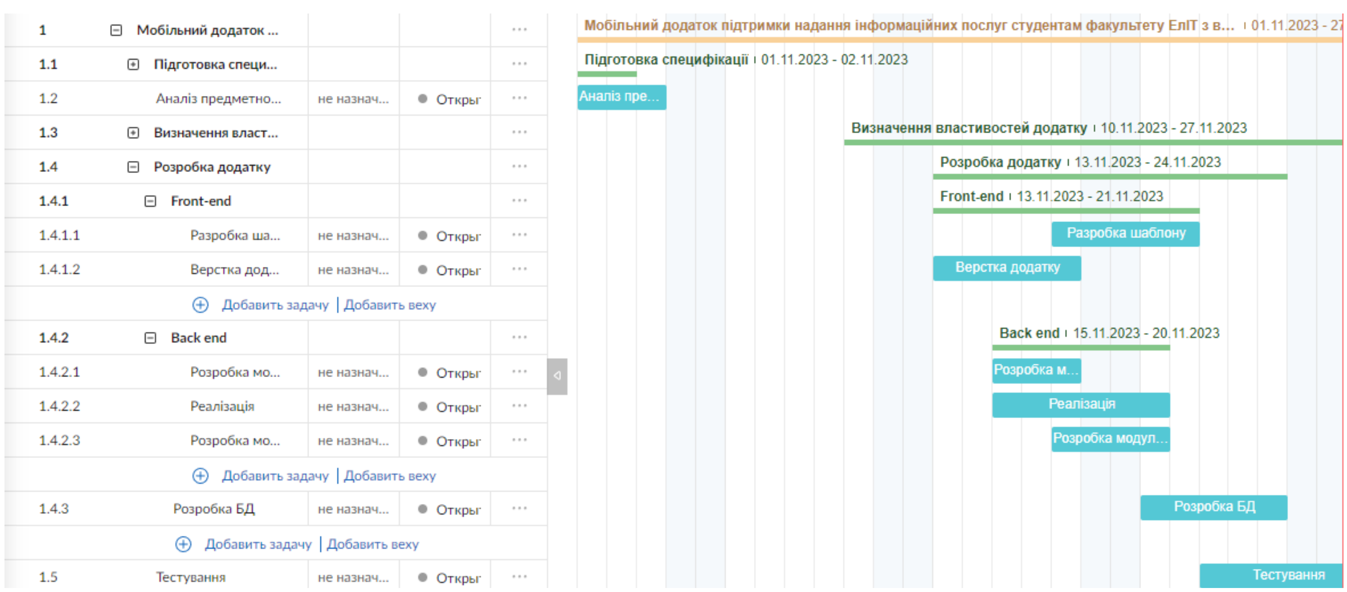

Рисунок А.3 – Календарний графік виконання проекту

Управління ризиками проекту. Наступним етапом планування робіт проекту є оцінка та робота над ризиками: кількісне та якісне оцінювання та заходи для усунення або зменшення впливу ризиків на проект. Перелік можливих ризиків даного проєкту надано в таблиці А.3 Результати їхньої оцінки представлено в таблиці А.4. Таблиця А.5 демонструє шкалу для класифікації ризиків за величиною впливу на проєкт та ймовірністю їх виникнення.

Таблиця А.3. – Ризики проекту

Номер	Ризик	Вірогідність	Шкода
R1	Непостійність програмного забезпечення.	2	3
R2	Неправильно працюючий функціонал додатку.	2	2
R3	Непередбачувані події.	4	4
R4	Додаткові вимоги.	2	5
R5	Відсутність актуальності у додатку.	3	1
R6	Втрата проекту	1	3

Таблиця А.4 – Результати визначення ймовірності, впливу та рангу ризиків проєкту

№ ризику	**Назва (опис) ризику**	**Ймовірність (0,1-0,9)**	**Вплив (0,05-0,8)**	**Ранг**
1	Непорозуміння між замовником та розробниками	0,2	0,03	0,006
2	Погано поставлена задача замовником	0,4	0,06	0,024
3	Поганий менеджмент часу	0,3	0,2	0,06
4	Використання непідходящих технологій	0,3	0,2	0,06
5	Відсутність резервних копій	0,4	0,2	0,08
6	Присутність багів	0,7	0,04	0,028
7	Збої у роботі при використанні користувачем	0,4	0,1	0,04
8	Поганий моніторинг виконання проєкту	0,5	0,03	0,015
9	Відсутній доступ до мережі Інтернет	0,3	0,08	0,024
10	Відсутність світла	0,2	0,05	0,01

Таблиця А.5 – Шкала оцінювання ризиків за ймовірністю виникнення на величиною впливу

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні

Продовження табл. А.5

2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для зменшення негативного впливу ризиків на ефективність роботи мобільного додатку важливо розробити план реагування на них. Аналіз проводиться на основі даних, представлених у таблиці А.4. На рисунку А.4 представлена остаточна матриця ризиків, яка служить основою для формування стратегії реагування на ризики проекту. Ризики, які вважаються прийнятними, позначені зеленим кольором, виправданими - жовтим, а неприпустимими - червоним.

Ймовірність виникнення ризику	**Вплив ризику**				
	0,05	**0,1**	**0,2**	**0,4**	**0,8**
0,9	0,045	0,09	0,18	0,36	0,72
0,7	0,035	0,07	0,14	0,28	0,56
0,5	0,025 **R2, R6**	0,05 **R10**	0,10	0,20	0,40
0,3	0,015 **R8**	0,03	0,06 **R3,R4**	0,12	0,24
0,1	0,005 **R1**	0,01	0,02	0,04 **R7**	0,08 **R5, R9**

Рисунок А.4 – Матриця ймовірності

Класифікація ризиків здійснена відповідно до набраних рангів у таблиці А.6. Опис ризиків та відповідних стратегій реагування представлений у таблиці А.7.

Таблиця А.6 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять (номера)
1	Прийнятні	0,005≤R≤0,05	1,2,6,7,8,10
2	Виправдані	0,05≤R≤0,14	3,4,5,9
3	Недопустимі	0,14≤R≤0,72	

Таблиця А.7 – Ризики та стратегії реагування

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_1	Відкритий	Погана комунікація між замовником та розробникам и	Низька	Низький	0.006	Зменшення	1.Встановити позитивні взаємини між розробником і керівником. 2.Строго дотримуватися правил ділового етикету під час спілкування. 3.Створити сприятливі умови для ефективної співпраці.	Коли виникає непорозуміння, важливо розібратися в його причинах, обговорити їх та сприяти створенню позитивної атмосфери в колективі.
RS_2	Відкритий	Погано поставлена задача замовником	Низька	Низький	0.024	Зменшення	1.Чітко і зрозуміло обговорити всі вимоги з замовником. 2.Розробити глосарій для уникнення непорозумінь у використанні термінів та визначень. 3.Регулярний контроль з боку замовника на різних етапах виконання робіт.	При виявленні розбіжностей у характеристиках продукту порівняно з вимогами, важливо уважно і точно вказати, в чому полягає невідповідність, та внести необхідні корективи.

Продовження табл. А.7

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_3	Відкритий	Поганий менеджмент часу	Низька	Високий	0.06	Зменшення	Здійснити оцінку актуальності ключових процесів та завдань, приділяючи особливу увагу ефективному управлінню часом. Визначити пріоритети в роботі та точно дотримуватися графіку виконання завдань у календарному плані.	Переставити порядок пріоритетів завдань. Розглянути можливі варіанти оптимізації роботи в рамках існуючого графіку. Оговорити з замовником можливі варіанти корекцій у терміни виконання.
RS_4	Відкритий	Використання непідходящих технологій	Низька	Середній	0.06	Зменшення	1.Провести аналіз методів та засобів для реалізації проєкту. 2.Вибрати технологію розробки, яка є зрозумілою та легкою у використанні.	Відвести час і ресурси на пошуки можливостей поліпшення використаної технології та скористатися допоміжними ресурсами.
RS_5	Відкритий	Відсутність резервних копій	Низька	Середній	0.08	Зменшення	1.Налаштувати автоматичне збереження даних. 2.Зберігати дані на різних носіях інформації.	Робити копію даних після кожного виконаного етапу.

Продовження табл. А.7

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_6	Відкритий	Присутність багів	Низька	Низький	0.028	Зменшення	1.Провести тестування перед запуском 2. Звернутися до фахівця для виправлення несправностей	Замінити програмне забезпечення.
RS_7	Відкритий	Збої у роботі при використанні користувачем	Низька	Низький	0.04	Зменшення	1.Розробка проєкту з урахуванням вимог до програмного забезпечення користувачів проєкту.	Модифікація проєкту з урахуванням різних версій програмного забезпечення, яке буде застосовуватися.
RS_8	Відкритий	Поганий моніторинг виконання проєкту	Низька	Низький	0.015	Зменшення	Здійснювати проміжний контроль результатів в ході виконання проєкту. Здійснювати моніторинг проєкту працівниками.	Здійснювати моніторинг проєкту замовником. Надання проміжних результатів виконання проєкту після кожного етапу.

Продовження табл. А.7

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_9	Відкритий	Відсутній доступ до мережі Інтернет	Низька	Середній	0,08	Зменшення	Мати сім-карти для підключення мобільного Інтернету.	Перейти у місце з доступом до мережі Інтернет.
RS_10	Відкритий	Відсутність світла	Низька	Низький	0,05	Ухилення	Мати пристрій для автономного живлення (пауербанк, генератор тощо).	Знайти місце де є світло.

ДОДАТОК Б

Лістинг основних модулів

Файл app.py

```
import base64
import datetime
import json
import os

from io import BytesIO
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from dotenv import find_dotenv, load_dotenv
from telethon import TelegramClient, events

app = Flask(__name__)
dotenv_path = find_dotenv()
load_dotenv(dotenv_path)

user = os.getenv("SQL_USER")
password = os.getenv("SQL_PASS")
port = os.getenv("SQL_PORT")
host = os.getenv("SQL_HOST")
schema_name = os.getenv("SCHEMA_NAME")
app.config[
    "SQLALCHEMY_DATABASE_URI"
] = f"mysql+pymysql://{user}:{password}@{host}:{port}/{schema_name}"
db = SQLAlchemy(app)


class Event(db.Model):
    __tablename__ = "event"
    id = db.Column(db.Integer, primary_key=True)
    description = db.Column(db.String(255))
    date = db.Column(db.String(25))
    location = db.Column(db.String(255))
    time = db.Column(db.String(25))
    link = db.Column(db.String(100))
    additional_details = db.Column(db.String(255))
    picture = db.Column(db.JSON)
    created_at = db.Column(db.DateTime)
    title = db.Column(db.String(50))


def handle_new_event():
    @client.on(events.NewMessage(chats=os.getenv("TG_CHANNEL_LINK")))
    async def my_event_handler(event):
```

```
        event_dict = {}

        try:
            if event.photo:
                img_bytes = await event.download_media(BytesIO())
                img_base64 =
base64.b64encode(img_bytes.getvalue()).decode("utf-8")
                img_json = {"image": img_base64}
                event_dict["picture"] = json.dumps(img_json)

            for line in event.raw_text.split("\n"):
                if ":" in line:
                    key, value = line.split(":", 1)
                    event_dict[key.strip()] = value.strip()

            event_dict["created_at"] =
datetime.datetime.now().isoformat()
            event_json = json.dumps(event_dict)

        except ValueError as e:
            print("Value Error:", str(e))

        event_data = json.loads(event_json)
        new_event = Event(**event_data)

        EventView.display_event(new_event)

        if new_event.title is not None and new_event.description is
not None:
            with app.app_context():
                db.session.add(new_event)
                db.session.commit()

    client.start()
    client.run_until_disconnected()


if __name__ == "__main__":
    api_id = os.getenv("TG_API_ID")
    api_hash = os.getenv("TG_API_HASH")

    client = TelegramClient("anon", api_id, api_hash)

    handle_new_event()
    app.run(port=os.getenv("PORT"))
```

Файл EventCard.tsx

```
import {
  View,
  Text,
  TouchableWithoutFeedback,
  Image,
  Dimensions,
} from "react-native";
import React, { useState } from "react";

import { useNavigation } from "@react-navigation/native";
let { width, height } = Dimensions.get("window");

export const EventCard = ({ item }) => {
  const navigation = useNavigation();
  return (
    <TouchableWithoutFeedback
      className="bg-blue-500"
      onPress={() => navigation.navigate("EventDetails")}
    >
      <Image
        source={source}
        style={{ width: width * 0.8, height: height * 0.4 }}
        className="rounded-3xl"
      />
    </TouchableWithoutFeedback>
    // </View>
  );
};
```

Файл Header.tsx

```
import { View, Text, TouchableOpacity, Image } from "react-native";
import React from "react";
import { useNavigation } from "@react-navigation/native";
import {
  widthPercentageToDP as wp,
  heightPercentageToDP as hp,
} from "react-native-responsive-screen";
interface HeaderProps {
  color: string;
  headerText: string;
  navigationPath: string;
  icon: string;
}
export function Header({
  color,
  headerText,
```

```
  navigationPath = "Home",
  icon = "",
}: HeaderProps) {
  const navigation = useNavigation();
  return (
    <View
      className={color}
      style={{
        paddingTop: hp(2),
        paddingBottom: hp(2),
        paddingLeft: wp(5),
        paddingRight: wp(5),
      }}
    >
      <View
        style={{
          flexDirection: "row",
          justifyContent: "space-between",
        }}
      >
        <TouchableOpacity onPress={() =>
navigation.navigate(navigationPath)}>
          <Image
            className="my-1"
            source={require("../../assets/icons/back-icon.png")}
            style={{
              height: hp(3),
              width: hp(3),
            }}
          ></Image>
        </TouchableOpacity>
        {icon && (
          <Image
            style={{
              height: hp(5),
              width: hp(5),
            }}
            source={icon}
          />
        )}
        <Text
          className="pt-1"
          style={{
            fontSize: 16,
            fontWeight: "bold",
            textAlign: "right",
            color: "grey",
          }}
```

```
        >
          {headerText}
        </Text>
      </View>
    </View>
  );
}
```

Файл ListEvents.tsx

```
import { View, Text, Dimensions } from "react-native";
import React from "react";
import Carousel from "react-native-snap-carousel";
import { EventCard } from "./EventCard";
import ElitSSUAssistantStrings from
"../ElitSSUAssistantStrings.json";
let { width } = Dimensions.get("window");

export default function ListEvents({ data }) {
  const getCurrentDate = () => {
    const d = new Date();
    return (
      d.getDate() + " " + d.getMonth() + ", " + d.getFullYear()
    );
  };
  return (
    <View className="space-y-3">
      <Text
        className="mx-8 text-white mb-2"
        style={{
          fontFamily: "ProductSans-Regular",
          fontSize: 22,
        }}
      >
        {getCurrentDate()}
      </Text>
      <Text
        className="text-white mx-8 mb-5"
        style={{
          fontFamily: "ProductSans-Bold",
          fontSize: 33,
          letterSpacing: -1,
          textShadowColor: "rgba(0, 0, 0, 0.25)",
          textShadowOffset: { width: 0, height: 2 },
          textShadowRadius: 4,
        }}
      >
        {ElitSSUAssistantStrings.events_adv}
```

```
      </Text>
      <View className="mb-5">
        <Carousel
          data={data}
          renderItem={({ item }) => <EventCard item={item} />}
          firstItem={2}
          inactiveSlideOpacity={0.6}
          sliderWidth={width}
          itemWidth={width * 0.79}
          slideStyle={{ display: "flex", alignItems: "center" }}
        ></Carousel>
      </View>
      <Text
        className="px-8 text-white"
        style={{
          fontFamily: "ProductSans-Regular",
          fontSize: 20,
          lineHeight: 30,
        }}
      >
        {ElitSSUAssistantStrings.social_media_adv}
      </Text>
    </View>
  );
}
```

Файл ListOptions.tsx

```
import { View, Text, Image } from "react-native";
import React from "react";
import {
  widthPercentageToDP as wp,
  heightPercentageToDP as hp,
} from "react-native-responsive-screen";
import Option from "./Option";
import ElitSSUAssistantString from "../ElitSSUAssistantStrings.json";
import icons from "../../constants/icons";

export default function ListOptions() {
  return (
    <View style={{ height: hp(115) }}>
      <Text
        className="text-center mb-5 pb-3"
        style={{
          color: "#808080",
          fontFamily: "ProductSans-Bold",
          fontSize: 30,
        }}
```

```
      >
        <Image
          source={icons.elit_logo}
          style={{
            width: wp(30),
            height: hp(30),
            resizeMode: "contain",
            marginRight: 10,
          }}
        ></Image>
        {ElitSSUAssistantString.elit_assistant_options_title}
      </Text>

      <Option
        title={ElitSSUAssistantString.ai_chat_option_title}

description={ElitSSUAssistantString.ai_chat_option_description}
        color={"bg-blue-50"}
        option={"Chat"}
      />

      <Option
        title={ElitSSUAssistantString.elit_events_option_title}

description={ElitSSUAssistantString.elit_events_option_description}
        color={"bg-blue-50"}
        option={"Events"}
      />


      <Text
        className="text-center"
        style={{
          marginTop: wp(20),
          color: "#808080",
          fontFamily: "ProductSans-Regular",
          fontSize: 17,
        }}
      >
        (c) 2023, yuch
      </Text>
    </View>
  );
}
```

Файл Option.tsx

```
import { View, Text, TouchableOpacity } from "react-native";
import React from "react";
import { widthPercentageToDP as wp } from "react-native-responsive-
screen";
import { useNavigation } from "@react-navigation/native";

interface OptionProps {
  title: string;
  description: string;
  color: string;
  option: "Chat" | "Maps" | "Events";
}

export default function Option({
  title,
  description,
  color,
  option,
}: OptionProps) {
  const navigation = useNavigation();
  return (
    <TouchableOpacity
      onPress={() => navigation.navigate({ name: option } as never)}
    >
      <View className={color + " px-10 py-5 rounded-2xl space-y-2 mt-
7"}>
        <View className="flex-row items-center">
          <Text
            style={{
              color: "#4b5563",
              fontSize: wp(5),
              fontFamily: "ProductSans-Bold",
            }}
          >
            {title}
          </Text>
        </View>
        <Text
          style={{
            fontSize: wp(4.5),
            color: "#808080",
            fontFamily: "ProductSans-Regular",
            lineHeight: wp(7),
          }}
          className="pr-5"
        >
```

```
          {description}
        </Text>
      </View>
    </TouchableOpacity>
  );
}
```

Файл SocialMediaIcon.tsx

```
import { TouchableOpacity, Image, Linking } from "react-native";
import React from "react";
import {
  widthPercentageToDP as wp,
  heightPercentageToDP as hp,
} from "react-native-responsive-screen";
import icons from "../../constants/icons";

interface SocialMediaIconProps {
  color?: string;
  icon: string;
  height: number;
  width: number;
  url: string;
}

const onPress = (url: string) =>
  Linking.canOpenURL(url).then(() => {
    Linking.openURL(url);
  });

export const SocialMediaIcon = ({
  icon,
  height,
  width,
  url,
}: SocialMediaIconProps) => {
  const selectedIcon = icons[icon];
  return (
    <TouchableOpacity onPress={() => onPress(url)}>
      <Image
        source={selectedIcon}
        style={{
          height: hp(height),
          width: hp(width),
        }}
      ></Image>
    </TouchableOpacity>
```

```
  );
};
```

Файл SocialMediaTab.tsx

```
import { View } from "react-native";
import React from "react";
import { SocialMediaIcon } from "./SocialMediaIcon";
import urls from "../../constants/urls";

export default function SocialMediaTab() {
  return (
    <View className="justify-center mx-16 rounded-2xl bg-blue-400">
      <View className="flex-row mx-auto justify-center">
        <View className="mx-7">
          <SocialMediaIcon
            icon={"insta_logo"}
            height={7}
            width={6}
            url={urls.elit_insta_url}
          />
        </View>
        <View className="mr-7 mt-0.5">
          <SocialMediaIcon
            icon={"university_logo"}
            height={6.5}
            width={6}
            url={urls.elit_ssu_ru_url}
          />
        </View>
        <View className="mr-7 mt-1">
          <SocialMediaIcon
            icon={"telegram_logo"}
            height={6}
            width={6}
            url={urls.elit_telegram_url}
          />
        </View>
      </View>
    </View>
  );
}
```

Файл index.tsx

```
import * as React from "react";
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-
stack";
import WelcomeScreen from "../screens/WelcomeScreen";
import HomeScreen from "../screens/HomeScreen";
import ChatScreen from "../screens/ChatScreen";
import EventsScreen from "../screens/EventsScreen";
import EventDetailsScreen from "../screens/EventsDetailsScreen";

const Stack = createNativeStackNavigator();

export default function AppNavigation() {
  return (
    <NavigationContainer>
      <Stack.Navigator
        screenOptions={{
          headerShown: false,
          animation: "fade",
          animationDuration: 1000,
        }}
        initialRouteName="Start"
      >
        <Stack.Screen name="Start" component={WelcomeScreen} />
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Chat" component={ChatScreen} />
        <Stack.Screen name="Events" component={EventsScreen} />
        <Stack.Screen name="EventDetails"
component={EventDetailsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Файл ChatScreen.tsx

```
import React, { useEffect, useState } from "react";
import {
  View,
  Text,
  ScrollView,
  Switch,
  SafeAreaView,
  TouchableOpacity,
  Image,
} from "react-native";
import {
  widthPercentageToDP as wp,
  heightPercentageToDP as hp,
```

```
} from "react-native-responsive-screen";
import { AutoGrowingTextInput } from "react-native-autogrow-
textinput";
import LottieView from "lottie-react-native";
import { LinearGradient } from "expo-linear-gradient";
import * as Clipboard from "expo-clipboard";

import ElitSSUAssistantStrings from
"../ElitSSUAssistantStrings.json";
import icons from "../../constants/icons";
import animations from "../../constants/animations";
import { Header } from "../components/Header";
import { SSUElitAPI } from "../services/api/SSUElitAPI";

export default function ChatScreen() {
  const [messages, setMessages] = useState([]);
  const [userInput, setUserInput] = useState("");
  const [userPrompt, setUserPrompt] = useState("");
  const [isLoading, setIsLoading] = useState(false);
  const [isEnabled, setIsEnabled] = useState(false);
  const [language, setLanguage] = useState("en");

  const toggleSwitch = () => {
    setLanguage((prevLanguage) => (prevLanguage === "en" ? "ua" :
"en"));
    setIsEnabled((prevIsEnabled) => !prevIsEnabled);
  };

  const ssuElitApi = new SSUElitAPI();

  const handleOnChangeText = (text) => {
    setUserInput(text);
  };

  useEffect(() => {
    if (userPrompt !== "" && isLoading) {
      setMessages([
        ...messages,
        { role: "bot", content:
ElitSSUAssistantStrings.loading_message },
      ]);
      ssuElitApi
        .sendAIRequest({ prompt: userPrompt, language: language })
        .then((datatest) => {
          setIsLoading(false);
          setMessages([...messages, { role: "bot", content:
datatest.body }]);
          setUserPrompt("");
```

```
      })
      .catch((error) => {
        setMessages([
          ...messages,
          {
            role: "bot",
            content: ElitSSUAssistantStrings.internal_server_error,
          },
        ]);
        setIsLoading(false);
      });
  }
}, [userPrompt]);

const handleOnPressButton = () => {
  if (!/^[a-zA-Za-яA-ЯёЁiIїЇєЄ\s\p{P}]+$/u.test(userInput)) {
    return;
  }
  setUserPrompt(userInput);
  setIsLoading(true);
  setUserInput("");
  setMessages((prevMessages) => [
    ...prevMessages,
    { role: "user", content: userInput },
  ]);
};

return (
  <SafeAreaView className="flex-1 pb-4 justify-center bg-white">
    <Header
      color={""}
      headerText={"elit_icon"}
      navigationPath={"Home"}
      icon={""}
    />
    <ScrollView
      bounces={false}
      className="space-y-4 mx-4 mb-3 mt-3"
      showsVerticalScrollIndicator={false}
    >
      {messages.length === 0 && (
        <View className="pt-40 items-center">
          <TouchableOpacity>
            <LottieView
              style={{
                height: 120,
                alignContent: "center",
                paddingRight: 10,
```

```
      paddingLeft: 10,
    }}
    source={animations.welcome_bot}
    autoPlay
  />
</TouchableOpacity>
<Text
  className="mt-3"
  style={{
    color: "#808080",
    fontFamily: "ProductSans-Regular",
    fontSize: 18,
  }}
>
  {ElitSSUAssistantStrings.chat_bot_greeting}
</Text>

<View
  className="mt-2"
  style={{
    flexDirection: "row",
    justifyContent: "center",
    alignItems: "center",
  }}
>
  <Text
    className="mr-2"
    style={{
      color: "#808080",
      fontFamily: "ProductSans-Regular",
      fontSize: 15,
    }}
  >
    EN
  </Text>
  <Switch
    trackColor={{ false: "#cbd5e1", true: "#dbeafe" }}
    thumbColor={isEnabled ? "#7dd3fc" : "#eff6ff"}
    ios_backgroundColor="#3e3e3e"
    onValueChange={toggleSwitch}
    value={isEnabled}
  />
  <Text
    className="ml-2"
    style={{
      color: "#808080",
      fontFamily: "ProductSans-Regular",
      fontSize: 15,
```

```
                }}
              >
                UA
              </Text>
            </View>
          </View>
        )}

        {messages.map((message, index) => {
          if (message.role === "user") {
            return (
              <View key={index} className="flex row justify-end pl-
8">
                <View
                  style={{
                    backgroundColor: "#F5F5F5",
                    elevation: 5,
                    borderRadius: 20,
                    shadowColor: "#000",
                    shadowOffset: {
                      width: 0,
                      height: 2,
                    },
                    shadowOpacity: 0.25,
                    shadowRadius: 3.84,
                  }}
                  className="rounded-3xl p-4 rounded-tr-none ml-3 mr-
3"
                >
                  <Text
                    selectable={true}
                    style={{
                      fontFamily: "ProductSans-Regular",
                      fontSize: 15,
                      lineHeight: wp(5),
                    }}
                  >
                    {message.content}
                  </Text>
                </View>
              </View>
            );
          } else {
            return (
              <View
                key={index}
                className="flex row justify-start pr-14"
                style={{ flexDirection: "row" }}
```

```
            >
              <View
                className="mr-2"
                style={{
                  width: wp(9),
                  height: hp(4.5),
                }}
              >
                <LinearGradient
                  colors={["rgba(0, 255, 0, 0.5)", "rgba(0, 0, 255,
0.5)"]}
                  start={[0, 0]}
                  end={[1, 1]}
                  style={{
                    flex: 1,
                    justifyContent: "center",
                    alignItems: "center",
                    borderRadius: 9999,
                  }}
                >
                  <Text
                    style={{
                      color: "#fff",
                      fontFamily: "ProductSans-Bold",
                      fontSize: 14,
                    }}
                  >
                    ELiT
                  </Text>
                </LinearGradient>
              </View>
              <View
                className="rounded-3xl p-3.5 pr-5 rounded-tl-none
bg-blue-200 mt-4 mb-4"
                style={{
                  elevation: 5,
                  shadowColor: "#000",
                  shadowOffset: {
                    width: 0,
                    height: 2,
                  },
                  shadowOpacity: 0.25,
                  shadowRadius: 3.84,
                }}
              >
                <TouchableOpacity
                  onPress={() =>
Clipboard.setStringAsync(message.content)}
```

```
                    >
                      <Text
                        selectable={true}
                        className="pb-2"
                        style={{
                          fontFamily: "ProductSans-Regular",
                          fontSize: 15,
                          lineHeight: wp(5),
                        }}
                      >
                        {message.content ===
                        ElitSSUAssistantStrings.loading_message ? (
                          <LottieView
                            style={{
                              paddingRight: 5,
                              height: 30,
                            }}
                            source={animations.loading}
                            autoPlay
                          />
                        ) : (
                          message.content
                        )}
                      </Text>
                    </TouchableOpacity>
                  </View>
                </View>
              );
            }
          })}
        </ScrollView>

        <View className="row">
          <AutoGrowingTextInput
            value={userInput}

placeholder={ElitSSUAssistantStrings.send_message_placeholder}
            onChangeText={handleOnChangeText}
            placeholderTextColor={"grey"}
            multiline={true}
            className="rounded-3xl p-3 left-4 bg-blue-50"
            style={{
              color: "#808080",
              fontFamily: "ProductSans-Regular",
              fontSize: 15,
              width: wp(75),
              height: hp(7.5),
              shadowColor: "#000",
```

```
            shadowOffset: {
              width: 0,
              height: 1,
            },
            shadowOpacity: 0.2,
            shadowRadius: 1.41,
            elevation: 2,
          }}
        ></AutoGrowingTextInput>

        <TouchableOpacity
          className={"rounded-3xl p-3 absolute right-4 bg-blue-50"}
          style={{
            width: wp(13),
            height: hp(6.5),
            shadowColor: "#000",
            shadowOffset: {
              width: 0,
              height: 1,
            },
            shadowOpacity: 0.2,
            shadowRadius: 1.41,
            elevation: 2,
          }}
          onPress={handleOnPressButton}
        >
          <View className="flex justify-center items-center">
            <Image
              className="mx-1 my-1"
              source={icons.push}
              style={{
                height: hp(3),
                width: hp(3),
              }}
            ></Image>
          </View>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
}
```

Файл HomeScreen.tsx

```
import { View } from "react-native";
import React from "react";
import { SafeAreaView } from "react-native-safe-area-context";
import ListOptions from "../components/ListOptions";
```

```
export default function HomeScreen() {
  return (
    <View className="flex-1 bg-white p-12">
      <SafeAreaView className="flex-1 justify-center">
        <ListOptions />
      </SafeAreaView>
    </View>
  );
}
```

Файл WelcomeScreen.tsx

```
import {
  View,
  Text,
  SafeAreaView,
  Image,
  TouchableOpacity,
} from "react-native";
import React from "react";
import {
  widthPercentageToDP as wp,
  heightPercentageToDP as hp,
} from "react-native-responsive-screen";
import { useNavigation } from "@react-navigation/native";
import ElitSSUAssistantStrings from
"../ElitSSUAssistantStrings.json";

export default function WelcomeScreen() {
  const navigation = useNavigation();
  return (
    <SafeAreaView className="flex-1 flex justify-around bg-white">
      <View className="spacy-y-2">
        <Text
          style={{ fontSize: wp(8) }}
          className="text-center font-bold text-grey-700"
        >
          {ElitSSUAssistantStrings.welcome_title}
        </Text>
        <Text
          style={{ fontSize: wp(4) }}
          className="text-center text-gray-700 font-semibold"
        >
          {ElitSSUAssistantStrings.welcome_message}
        </Text>
      </View>
      <View className="flex-row justify-center">
        <Image
          style={{ width: wp(60), height: hp(10) }}
          source={require("../../assets/images/logo.png")}
```

```
        ></Image>
      </View>
      <TouchableOpacity
        onPress={() => navigation.navigate("Home")}
        className="bg-blue-400 mx-5 p-4 rounded-2xl"
      >
        <Text
          style={{ fontSize: wp(6) }}
          className="text-center font-bold text-white"
        >
          {ElitSSUAssistantStrings.welcome_button_message}
        </Text>
      </TouchableOpacity>
    </SafeAreaView>
  );
}
```

Файл SSUElitAPI.tsx

```
import ElitSSUAssistantStrings from
"../../ElitSSUAssistantStrings.json";

interface HumanPrompt {
  prompt: string;
  language: string;
}

function formatLanguage(language: string) {
  switch (language) {
    case "en":
      return "English";
    case "ua":
      return "Ukrainian";
    default:
      return "Ukrainian";
  }
}

export class SSUElitAPI {
  async sendAIRequest(properties: HumanPrompt) {
    return fetch(process.env.EXPO_PUBLIC_SSU_API_URL + "/chatbot", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        prompt: properties.prompt,
        language: formatLanguage(properties.language),
      }),
    })
      .then(async (response) => {
```

```
        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }
        return await response.json();
      })
      .catch((error) => {
        console.error(ElitSSUAssistantStrings.request_failed, error);
      });
  }
}
```

Файл ImageUtils.tsx

```
import * as FileSystem from 'expo-file-system';

export const saveBase64Image = async (picture: { image: string },
filename: string) => {
  const filePath = FileSystem.documentDirectory + filename;
   if (picture.image) {
     const image_data =
picture.image.replace(/^data:image\/\w+;base64,/, "");

     try {
       await FileSystem.writeAsStringAsync(filePath, image_data, {
encoding: FileSystem.EncodingType.Base64 });
       console.log('Image saved to:', filePath);
     } catch (err) {
       console.log(err.message);
     }
   } else {
     console.log('No image data available');
   }
};
```

Файл events_controller.py

```
from flask import jsonify
from app.events.model.database import Event


def get_all_events_json():
    events = Event.query.all()
    return jsonify([event.to_dict() for event in events])
```

Файл routes.py

```
from app.events import events_bp
from app.events.conroller.events_controller import
get_all_events_json
```

```
@events_bp.route("/get_events", methods=["GET"])
def get_events():
    events = get_all_events_json()
    return events


@events_bp.route("/events_test")
def get_events_test():
    return "This is the events blueprint!"
```

Файл database.py

```
import os
from flask_sqlalchemy import SQLAlchemy
from dotenv import load_dotenv


load_dotenv()

# Database configuration
user = os.getenv("SQL_USER")
password = os.getenv("SQL_PASS")
port = os.getenv("SQL_PORT")
host = os.getenv("SQL_HOST")
schema_name = os.getenv("SCHEMA_NAME")

db = SQLAlchemy()


def init_events_db(app):
    app.config[
        "SQLALCHEMY_DATABASE_URI"
    ] =
f"mysql+pymysql://{user}:{password}@{host}:{port}/{schema_name}"
    db.init_app(app)


class Event(db.Model):
    __tablename__ = "event"
    id = db.Column(db.Integer, primary_key=True)
    description = db.Column(db.String(255))
    date = db.Column(db.String(25))
    location = db.Column(db.String(255))
    time = db.Column(db.String(25))
    link = db.Column(db.String(100))
    additional_details = db.Column(db.String(255))
    picture = db.Column(db.JSON)
    created_at = db.Column(db.DateTime)
    title = db.Column(db.String(50))
```

```
    def to_dict(self):
        return {
            "id": self.id,
            "description": self.description,
            "date": self.date,
            "location": self.location,
            "time": self.time,
            "link": self.link,
            "additional_details": self.additional_details,
            "picture": self.picture,
            "created_at": self.created_at.isoformat() if
self.created_at else None,
            "title": self.title,
        }
```

Файл __init__.py

```
from flask import Blueprint

events_bp = Blueprint("events", __name__)

from .conroller import routes
```

Файл data_loader.py

```
from langchain.document_loaders import UnstructuredURLLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter


class DataLoader:
    def __init__(self, urls):
        self.urls = urls

    def load_data(self):
        loader = UnstructuredURLLoader(urls=self.urls,
headers={"User-Agent": "value"})
        data = loader.load()
        return data

    def split_data(self, data):
        text_splitter = RecursiveCharacterTextSplitter(
            chunk_size=1000, chunk_overlap=200
        )
        splits = text_splitter.split_documents(data)
        return splits
```

Файл databases.py

```
import os
from .data_loader import DataLoader
from app.config import list_urls
```

```
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings


class ChromaDBInstance:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super().__new__(cls)
            cls._instance.initialize_vectorstore()
        return cls._instance

    @classmethod
    def get_instance(cls):
        if cls._instance is None:
            cls._instance = cls()
        return cls._instance

    def initialize_vectorstore(self):
        self.load_data()
        self.create_vectorstore()

    def load_data(self):
        data_loader = DataLoader(list_urls)
        data = data_loader.load_data()
        self.splits = data_loader.split_data(data)

    def create_vectorstore(self):
        self.vectorstore = Chroma.from_documents(
            documents=self.splits,
            embedding=OpenAIEmbeddings(),
      )
```

Файл routes.py

```
from flask import Blueprint, request
from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain.schema import StrOutputParser
from langchain.schema.runnable import RunnablePassthrough
from langchain.schema.messages import HumanMessage
from app.chatbot.model.database import ChromaDBInstance
from app.chatbot import chatbot_bp


chat_history = []


@chatbot_bp.route("/chatbottest")
```

```
def get_chatbot_test():
    return "This is the chatbot blueprint!"


@chatbot_bp.route("/chatbot", methods=["POST"])
def query_ai():
    db = ChromaDBInstance.get_instance()

    content_type = request.headers.get("Content-Type")
    if content_type != "application/json":
        return "Content-type is not supported", 400

    json_payload = request.json
    prompt = json_payload.get("prompt")
    language = json_payload.get("language")

    if not prompt or not language:
        return "Invalid request payload", 400

    retriever = db.vectorstore.as_retriever()

    llm = ChatOpenAI(model_name="gpt-3.5-turbo-1106", temperature=0)

    system_prompt = """Given a chat history and the latest user
question \
    which might reference the chat history, formulate a standalone
question \
    which can be understood without the chat history. Do NOT answer
the question, \
    just reformulate it if needed and otherwise return it as is."""

    condense_q_prompt = ChatPromptTemplate.from_messages(
        [
            ("system", system_prompt),
            MessagesPlaceholder(variable_name="chat_history"),
            ("human", "{question}"),
        ]
    )

    history_q_chain = condense_q_prompt | llm | StrOutputParser()

    qa_system_prompt = """You are AI chat bot assistant for ELIT
faculty at Sumy State University.
        Use following pieces of context to give an answer.
        If the question is not related to Sumy State University, ELIT
or context - just say that you don't know the answer.
        Answers should be informative and helpful but not too long -
3 sentences maximum. Answer should be always in language_placeholder.
```

```
        Try to give answer as fast as possible. You are replying to
students, so be friendly.
        If you don't know the answer, just say that you don't know,
don't try to make up an answer.

        Context: {context}
        Question: {question}
        Helpful answer: """.replace(
        "language_placeholder", language
    )

    qa_prompt = ChatPromptTemplate.from_messages(
        [
            ("system", qa_system_prompt),
            MessagesPlaceholder(variable_name="chat_history"),
            ("human", "{question}"),
        ]
    )

    def doc_format(docs):
        return "\n\n".join(doc.page_content for doc in docs)

    def condense_question(input_data: dict):
        if input_data.get("chat_history"):
            return history_q_chain
        else:
            return input_data["question"]

    rag_chain = (
        RunnablePassthrough.assign(context=condense_question |
retriever | doc_format)
        | qa_prompt
        | llm
    )

    try:
        ai_msg = rag_chain.invoke({"question": prompt,
"chat_history": chat_history})
        chat_history.extend([HumanMessage(content=prompt), ai_msg])
        return {"body": ai_msg.content, "statusCode": 200}
    except Exception as e:
        return {"body": ai_msg.content, "statusCode": 500}
```

Файл __init__.py

```
from flask import Blueprint

chatbot_bp = Blueprint("chatbot", __name__)

from .controller import routes
```