МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Чат-бот на основі технології штучного інтелекту для підтримки здорового способу життя

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ Владислав ЛЕВЧЕНКО

«__» _____ 20__ р.

Керівник роботи
д-рка техн. наук,
професорка _____ Альона ШВЕД

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Левченко Владислав Васильович

1. Тема кваліфікаційної роботи " Чат-бот на основі технології штучного інтелекту для підтримки здорового способу життя"
2. затверджена наказом ректора ЧНУ ім. Петра Могили № __349___ від «_26_» _грудня_____ 2025 р.
3. Строк представлення кваліфікаційної роботи «___» ____ 2026 р.
4. Очікуваний результат роботи та початкові дані якщо такі потрібні. Запити користувача і відповіді чатбота.
5. Перелік питань, що підлягають розробці:

Відповідно до мети визначено такі завдання:

1. аналіз сучасних технологій створення чатботів на основі технологій штучного інтелекту;
2. проєктування та програмна реалізація чатбота для підтримки здорового способу життя;
3. проєктування та розробка інформаційного забезпечення чатботу;
4. тестування та апробація чатботу.

5. Перелік графічних матеріалів:

Презентація

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «__03__» __лютого_____ 20_26__ р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: **Чат-бот на основі технології штучного інтелекту для підтримки здорового способу життя**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	27.01.2025	03.02.2026	*Виконано*
2.	Огляд літератури за темою роботи	05.09.2025	27.03.2026	*Виконано*
3.	Складання календарного плану КБР	20.01.2026	25.01.2026	*Виконано*
4.	Аналіз предметної області	04.09.2025	05.09.2025	*Виконано*
5.	Розробка проєктних рішень	10.10.2025	10.11.2025	*Виконано*
6.	Моделювання та конструювання ПЗ	04.09.2025	19.12.2025	*Виконано*
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	10.09.2025	10.04.2026	*Виконано*
8.	Відгук керівника КБР	13.04.2026	14.04.2026	*Виконано*
9.	Оформлення КБР та презентації	16.04.2026	05.05.2026	*Виконано*
10.	Попередній захист	27.05.2026	27.05.2026	*Виконано*
11.	Завершення оформлення КБР та презентації	27.05.2026	11.06.2026	*Виконано*
12.	Рецензування	12.06.2026	13.06.2026	*Виконано*
13.	Захист кваліфікаційної роботи	22.06.2026	22.06.2026	*Виконано*

Здобувач _____ Владислав ЛЕВЧЕНКО

«__» _____ 20__ р.

Керівник роботи

д-рка техн. наук,

професорка _____ Альона ШВЕД

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Чат-бот на основі технології штучного інтелекту для підтримки здорового способу життя

Здобувач 409 гр.: Левченко Владислав

Керівник: д-рка техн. наук, професорка Швед Альона

Актуальність. В теперішні складні часи, особливо, є важливим ведення здорового способу життя, адже здоров'я – це повне фізичне, психічне й соціальне благополуччя. На разі, для взаємодії з користувачем існують чатботи, що можуть спілкуватись природньою для людини мовою. В тому числі, існують чатботи, що використовують методи штучного інтелекту.

Мета роботи полягає у розробці чатбота на основі технології штучного інтелекту для підтримки здорового способу життя.

Об’єкт роботи: процес надання рекомендацій користувачу стосовно підтримки здорового способу життя.

Предмет роботи: методи реалізації чатбота на основі технології штучного інтелекту для підтримки здорового способу життя.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі знаходиться базова інформація про кваліфікаційну роботу бакалавра, що було виконано відповідно до завдань і календарного плану.

У першому розділі наведено інформацію про технологію штучного інтелекту і відповідні бази даних.

У другому розділі обгрунтовано вибір стеку технологій розробки.

У третьому розділі наведено рішення з моделювання та проєктування чат-боту.

У четвертому розділі розглянуто питання програмної реалізації чатботу та тестування ПЗ.

.

У висновках наведено загальні висновки стосовно інформації з усіх розділів кваліфікаційної роботи бакалавра.

Кваліфікаційна робота викладена на 70 сторінках машинописного тексту, складається із вступу, _4_ розділів, загальних висновків, переліку джерел посилання з _17_ найменувань та _1_ додатків. Праця містить _8_ таблиць та _33_ рисунків.

Ключові слова: чат-бот, штучний інтелект, здоровий спосіб життя

ABSTRACT

to the qualifying bachelor's thesis

Chatbot Based on Artificial Intelligence Technology to Support a Healthy Lifestyle

Student of 409 group: Levchenko Vladyslav

Supervisor: Dr. science, Professor Shved Aliona

Relevance. In today’s challenging times, maintaining a healthy lifestyle is especially important, since health is defined as complete physical, mental, and social well-being. Currently, chatbots exist that can interact with users in natural human language. Among them are chatbots that employ artificial intelligence methods.

Purpose of the work: the development of a chatbot based on artificial intelligence technology for supporting a healthy lifestyle.

Object of the work: the process of providing recommendations to the user regarding maintaining a healthy lifestyle.

Subject of the work: methods of implementing a chatbot based on artificial intelligence technology for supporting a healthy lifestyle.

The qualification work consists of an introduction, 4 sections, conclusions, and a list of references.

In the introduction, basic information about the bachelor’s qualification work is presented, completed according to the tasks and calendar plan.

The first section provides information about artificial intelligence technology and relevant databases.

The second section justifies the choice of development technology stack.

The third section provides solutions for modeling and designing a chatbot.

The fourth section considers the issue of chatbot software implementation and software testing.

In the conclusions, general findings from all sections of the bachelor’s qualification work are summarized.

The qualification work is presented on _70_ pages of typewritten text, consists of an introduction, _4_ sections, general conclusions, a list of references with _17_ titles, and _1_ appendices. The work contains _8_ tables and _33_ figures.

Keywords: Chat bot, Artificial Intelligence, Healthy Lifestyle.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЗСЖ – здоровий спосіб життя;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект;

CSRF – cross site request forgery;

LLM – learning language machine;

XSS – cross site scripting.

ВСТУП

Актуальність. На даний час, чатботи для підтримки здорового способу життя стали популярними тому, що більша кількість людей потребують введення здорового способу життя. На сьогодні, більшість людей хворіють на ті хвороби, що вони самі ж й викликають. Це хвороби пов’язані з фізичним здоров’ям.

В теперішні складні часи, особливо, є важливим ведення здорового способу життя, адже здоров'я – це повне фізичне, психічне й соціальне благополуччя. Наразі, для взаємодії з користувачем існують чатботи, що можуть спілкуватись природньою для людини мовою. В тому числі, існують чатботи, що використовують методи штучного інтелекту. Чатботи на основі технології штучного інтелекту можуть знаходити інформацію за різними спекторами суспільного життя. Наприклад, вони знаходять інформацію стосовно ведення здорового способу життя. Але, ця інформація не завжди є актуальною через різні джерела інформації. Чатботи для підтримки здорового способу життя завжди знаходять актуальну інформацію. Тому, на сьогодні, чатбот для підтримки здорового способу життя є актуальним.

Метою роботи є розробка чатбота на основі технології штучного інтелекту для підтримки здорового способу життя.

Відповідно до мети визначено такі завдання:

1. аналіз сучасних технологій створення чатботів на основі технологій штучного інтелекту;
2. проєктування та програмна реалізація чатбота для підтримки здорового способу життя;
3. проєктування та розробка інформаційного забезпечення чатботу;
4. тестування та апробація чатботу.

Об’єктом роботи є процес надання рекомендацій користувачу стосовно підтримки здорового способу життя .

Предметом роботи є методи реалізації чатбота на основі технології штучного інтелекту для підтримки здорового способу життя.

Наразі, у сфері охорони здоров’я потребують сучасні версії систем моніторингу стану здоров’я людини, адже від цього напряму залежать життя й здоров’я людей. Такий підхід може суттєво полегшити життя людства. Але, далеко не кожна технологія здатна покращити сферу охорони здоров’я через те, що у них немає більшої специфікації по питаннях медичної здоров’я.

Сьогодні, сфера інформаційних технологій для підтримки здорового способу життя розвивається в геометричній прогресії. Говорячи про це, ми маємо на увазі наступні технології ШІ для підтримки здорового способу життя: чат-боти на машинному навчанні, застосунки на основі технології штучного інтелекту, боти різних месенджерів для підтримки здорового способу життя, системи рекомендацій для підтримки здорового способу життя. Чат-боти на машинному навчанні можуть виконувати наступні функції:

1. аналізувати корисність й поживність страв, за допомогою зображень;
2. нагадувати про час виконання наступних дій:
    1.1 споживання їжі й води;
    1.2 виконання фізичних вправ.

Недоліками даних чатботів є такі речі: не завжди актуальні дані та вимога зображень ідеальної якості. Застосунки на основі технології штучного інтелекту бувають як мобільними, так й вебзастосунками. Застосунки можуть виконувати наступні ролі: трекер спожитої їжі й води, тренер з виконання фізичних вправ. Дані застосунки мають наступні переваги:

1. мобільність;
2. можливість соціальною активності.

З недоліків даних застосунків є наступне:

1. необхідність ручного введення інформації;

2. складність виконання операцій з домашніми стравами в плані трекування.

Ботів різних месенджерів для підтримки здорового способу життя також можна навчити розпізнавати страви за зображеннями й надавати відповідні рекомендації.

Дані боти мають наступні переваги:

1. зручне керування за допомогою команд;
2. швидкість отримання актуальної інформації;
3. знаходження індивідуального підходу до кожного користувача.

Недоліком даних ботів є така річ, як недоступність офлайн.

Системи рекомендацій для підтримки здорового способу життя використовують, в якості навчальних алгоритмів, алгоритми штучного інтелекту й машинного навчання. Це робиться для знаходження індивідуального підходу до кожного з користувачів. Дані системи виконують наступні функції:

1. створюють плани харчування індивідуально для кожного користувача;
2. рекомендують фізичні вправи;
3. аналізують дані про стан здоров’я з відповідних пристроїв;
4. надає поради щодо режиму сну й якості харчування.

Дані системи мають наступні переваги:

1. знаходження індивідуального підходу до кожного з користувачів;
2. інтеграція з іншими пристроями;
3. рекомендації, що постійно оновлюються.

Виходячи з вищесказаного, можна зробити висновок про те, що саме через зростання рівня зручності й популярності відповідних застосунків розвиваються чат-боти на основі технології ШІ. Саме тому, їхнє використання є більш логічним для введення ЗСЖ.

1 АНАЛІЗ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ ЗА ДОПОМОГОЮ ЧАТ-БОТІВ

Наразі, технологія штучного інтелекту стрімко розвивається. Чатботи на основі даної технології, можуть спеціалізовуватись на багатьох сферах. Наприклад, ChatGPT, як й інші, базується на декількох сферах суспільного життя. На рис. 1.1 продемонстровано інтерфейс ChatGPT. Він є одним з простих.

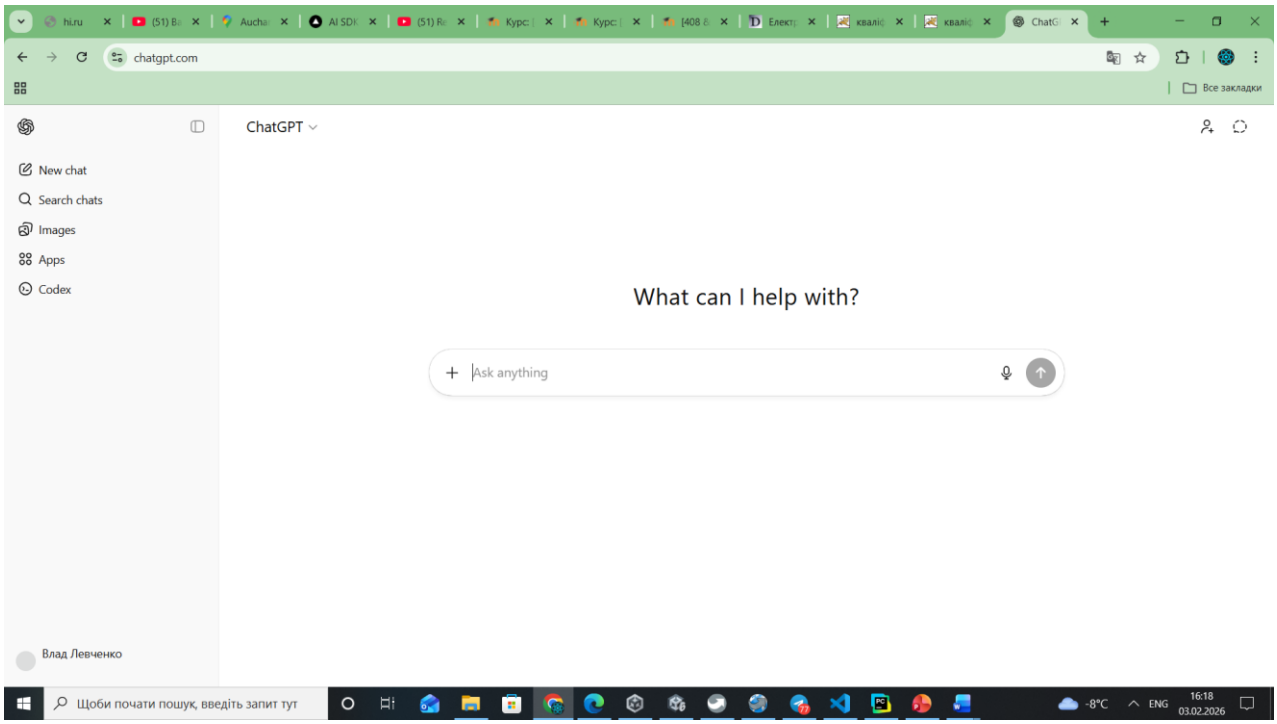

Рисунок 1.1 – Інтерфейс ChatGPT

Функціональні можливості ChatGPT є такими: реєстрація та авторизація користувачів, платні підписки (ChatGPT Plus, корпоративні тарифи), редагування повідомлень у чаті.

Перевагами даного чатботу є такі речі, як: висока якість генерації тексту, доступність через веб та мобільні застосунки. Недоліками даного чатботу є те, що: він потребує інтернет-з’єднання, він має обмеження безкоштовної версії (ліміти на кількість запитів) і не прозорі навчальні дані. На рис. 1.2

продемонстровано інтерфейс Gemini. Він, також, є одним з простих інтерфейсів.

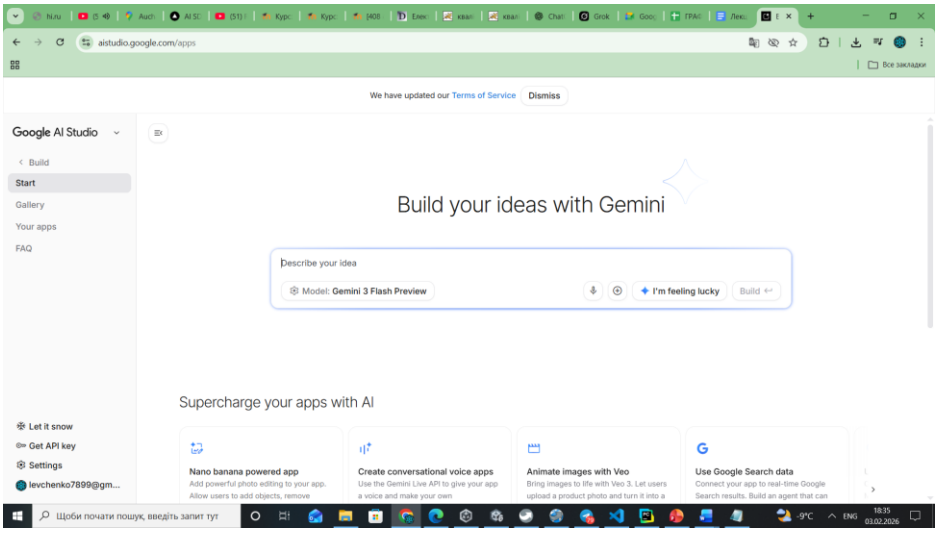

Рисунок 1.2 – Інтерфейс Gemini

Функціональні можливості Gemini є такими: реєстрація та авторизація користувачів, безкоштовно використання, підтримка голосових команд. Перевагами даного чатботу є такі речі, як: простота використання і безкоштовний доступ. Недоліками даного чатботу є наступне: не генерує тексти, менший функціонал, обмежена екосистема. На рис. 1.3 продемонстровано інтерфейс Grok. Він, також, є одним з простих інтерфейсів.

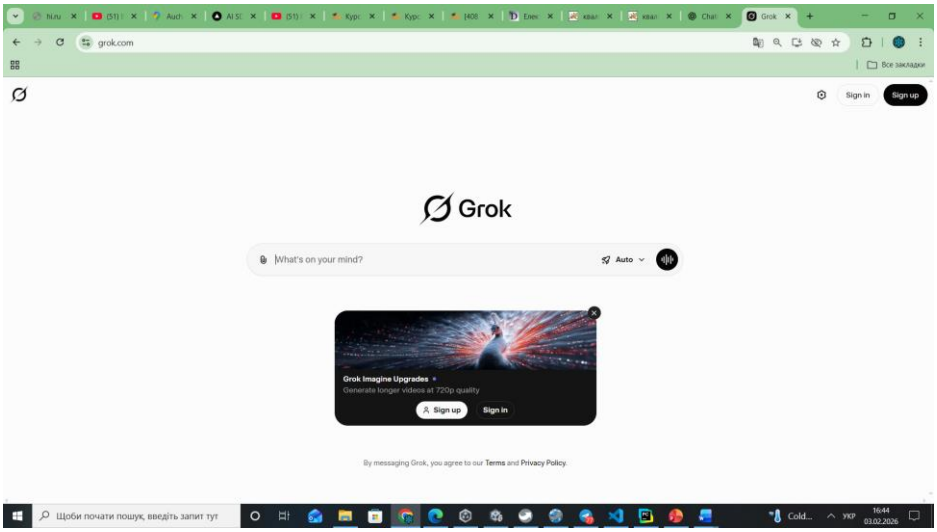

Рисунок 1.3 – Інтерфейс Grok

Функціональні можливості Grok є наступними: реєстрація та авторизація користувачів, безкоштовно використання, підтримка голосових команд, генерація текстів, багатомовність.

Перевагами даного чатботу є наступне: широка інтеграція з екосистемою Microsoft, кросплатформність, багатофункціональінсть. Недоліками даного чатботу є наступне: обмежений функціонал, відсутність персоналізації.

1.1 Аналіз предметної області використання технологій штучного інтелекту для підтримки здорового способу життя

Через стрімке зростання попиту на введення здорового способу життя, інформаційні технології для підтримки здорового способу життя стрімко розвиваються. Існують наступні апаратні й програмні засоби для підтримки здорового способу життя:

1. мобільні застосунки;
2. фітнес браслети;
3. телемедицина;
4. застосунки на основі технології ШІ;
5. віртуальні тренери;
6. фітнес платформи.

Мобільні застосунки є важливим через можливість отримання користувачами персоналізованих рекомендацій. Це проявляється в таких речах, як: формування раціону харчування, контроль калорій, налаштування режиму сну та медитації. Саме через зручність відповідних мобільних застосунків є великий попит на їхню реалізацію. На сьогодні, попит на застосунки даного типу є особливим через індивідуальний досвід їхнього використання кожним з користувачів.

Фітнес-браслети допомагають людям в наступному:

1. підрахувати кількість пройдених кроків;

2. підрахувати кількість витрачених калорій;
3. визначити ритм роботи серця;
4. визначити спожитий кисень.

Телемедицина допомагає людям в плані проходження обстежень лікарями в режимі «онлайн». Особливо, це важливо для тих, хто має обмежений доступ до закладів охорони здоров'я. Тобто, людині не потрібно відвідувати кожен раз лікарні для проходження обстежень свого здоров'я.

Застосунки на основі технології штучного інтелекту надають можливість користувачам дізнатись про свої генетичні особливості, фізичні чинники й історії власних хвороб користувача й створює рекомендації кожному з користувачів індивідуально. Віртуальні тренери й фітнес-платформи допомагають користувачу займатись фізичною активністю в будь-якому місці. Вони надають можливість тренувань від кардіо до силових. Саме таким чином, існуючі програмні рішення допомагають людям вести здоровий спосіб життя.

1.2 Аналіз сучасних програмних систем для підтримки здорового способу життя

Існують такі сучасні програмні системи для підтримки здорового способу життя, як: MyFitnessPal, YAZIO, FatSecret. MyFitnessPal є одним з найвідоміших застосунків для дослідження фізичного стану організму й харчування. Даний застосунок дозволяє користувачам вести здоровий спосіб життя, виконуючи наступні речі: підрахунок кількості калорій, підрахунок кількості виконаних фізичних вправ.

MyFitnessPal має наступні функції: сканер штрих-кодів для додавання продуктів, велика база даних продуктів харчування, інтеграція з іншими фітнес-застосунками й пристроями, соціальна активність [14]. Перевагами даного застосунка можна назвати такі речі, як: велика база даних продуктів, зручний інтерфейс, налаштування цілей.Недоліками даного застосунка є

наступне: деякий функціонал лише для платної версії, багато реклами в безкоштовній версії. Додаткові можливості даного застосунку є наступними:

– дослідження прогресу;
– налаштування цілей;
– спільнота для користувачів;
– інтеграція з іншими пристроями.

Розробляли даний застосунок Майк та Альберт Лі у 2005 році. Даний застосунок реалізовано на наступних мовах програмування: Java, Swift. Даний застосунок є доступним для таких ОС, як: Android, IOS.

Виходячи з того, можна сказати те, що даний застосунок є одним з зручних для тих людей, котрі цінують свій час й інші ресурси. На рис. 1.4 продемонстровано графічний інтерфейс MyFitnessPal.

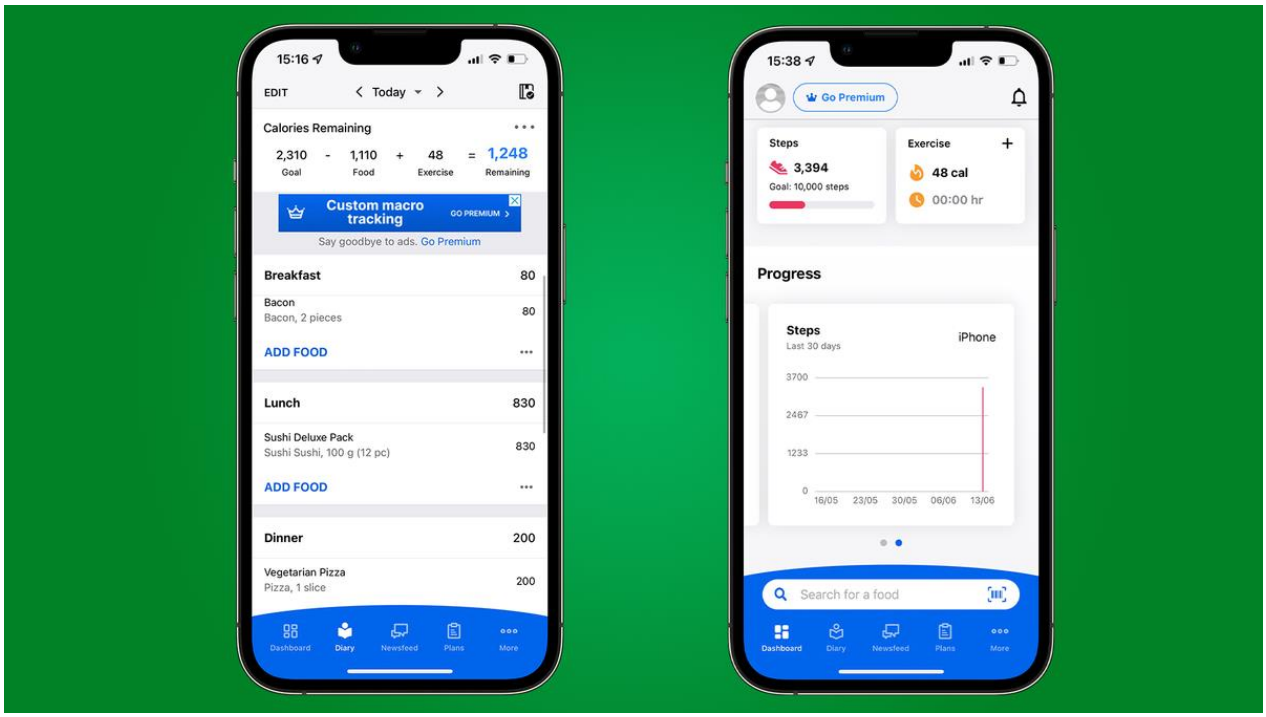

Рисунок 1.4 – Графічний інтерфейс MyFitnessPal

Yazio – одна з програм, що допомагає у виборі дієти. Ця програма підбирає дієти, враховуючи наступні фактори: вікову групу користувача й особливості його організму [15]. Даний застосунок має такі функції, як:

голодування з інтервалом, більше 3 000 рецептів, інтеграція з іншими пристроями, трекування води й активності, AI-помічник в аналізі їжі.

Перевагами даного застосунку можна назвати наступне:

1. персоналізованість;
2. локалізація;
3. зручність дизайну;
4. комплекс;
5. інструменти для мотивації користувача.

Недоліками даного застосунку є наступні речі:

1. найбільша вартість;
2. багато рекламних оголошень у безкоштовній версії;
3. обмежений функціонал для простих акаунтів;
4. сильне обмеження безкоштовної версії;
5. залежність від самодисципліни користувача.

Також, можна виділити такі додаткові можливості, як: індивідуальні плани харчування, записи у щоденник настрою й самопочуття, інтеграція з іншими програмними засобами й пристроями, відстеження елементів різних типів у енгрідієнтах страви, списки рецептів з автоматичним підрахунком калорій. Особливо, даний застосунок підходить тим людям, які готові занурюватись у справжній контроль харчування й дієт та, водночас, готовий сплачувати за Pro-підписку, щоб мати повний пакет функціоналу. Наприклад, щоб інтеграція з іншими програмними засобами й пристроями стала можливою для конкретного користувача. Тобто, даний застосунок також користується популярністю серед тих людей, хто вирішив ввести здоровий спосіб життя. На рис. 1.5 продемонстровано графічний інтерфейс Yazio.

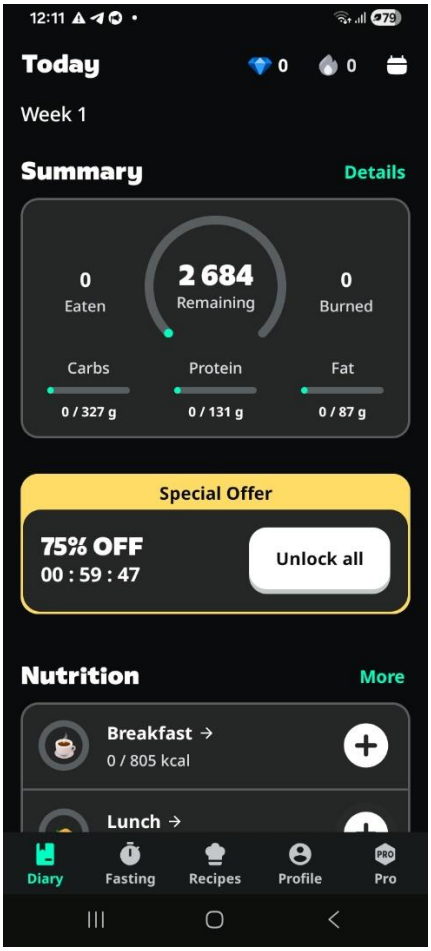

Рисунок 1.5 – Графічний інтерфейс Yazio

FatSecret також є одним з найвідоміших застосунків, що доступний на IOS, Android і Google [16]. Даний застосунок розроблено компанією «FatSecret».

Переваги FatSecret є наступними: широка база даних для їжі й напоїв, цікаво реалізовані блоги, велика кількість елементів для кожної страви, зручність графіків й діаграм. Недоліками даного застосунку можна вважати наступне: несучасний дизайн, відсутність можливості моніторингу споживання води. Функції даного застосунку являються наступними: сканування штрих-коду, спільні групи й блоги, додавання щоденних прийомів їжі, додавання щоденного виконання вправ, інтеграція з Google Fit і Apple health. На рис. 1.6 продемонстровано графічний інтерфейс FatSecret.

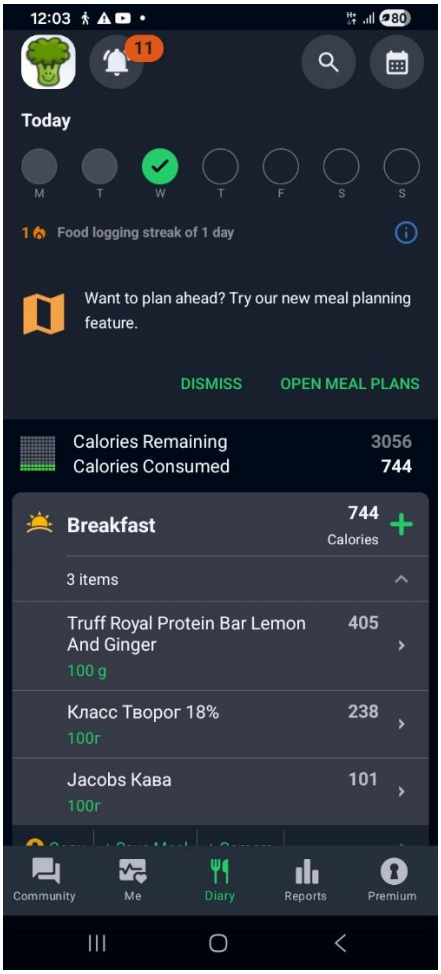

Рисунок 1.6 – Графічний інтерфейс FatSecret

На разі, великий попит серед тих людей, хто введе ЗСЖ, базується саме на FatSecret. Адже, як зазначено вище, даний застосунок має індивідуальний функціонал.

1.3 Аналіз сучасних чат-ботів на основі технології штучного інтелекту

ChatGPT – це чатбот на основі технології штучного інтелекту, який значно перевищує можливості простих роботів. Його особливості полягають у здатності:

– обробляти природну мову та формувати відповіді, максимально наближені до людського стилю спілкування;

– навчатися на великих масивах даних, що дозволяє йому охоплювати широкий спектр тем – від побутових порад до складних технічних питань;

– підтримувати контекст діалогу, завдяки чому користувач отримує більш логічні та послідовні відповіді;

– інтегруватися в різні системи та сервіси – від освітніх платформ до бізнес-рішень, забезпечуючи автоматизацію процесів та підвищення ефективності взаємодії.

Таким чином, ChatGPT є прикладом сучасного чатбота, що демонструє високий рівень розвитку технологій штучного інтелекту. Він не лише відповідає на запитання, а й здатний вести діалог, аналізувати інформацію та адаптуватися до потреб користувача, що робить його універсальним інструментом у різних сферах діяльності.

Copilot – чатбот на основі технології штучного інтелекту, засновником якого є IT компанія «Microsoft». Метою створення даного чатботу є наступне: допомагати людям у навчанні, допомагати людям у роботі; допомагати людям у виконанні завдань у повсякденному житті.

Copilot поєднує сучасні алгоритми обробки природної мови та інтеграцію з продуктами Microsoft, що робить його універсальним інструментом для користувачів різних сфер діяльності. Він здатний аналізувати контекст запитань, надавати структуровані відповіді, допомагати у створенні текстів, коду, документів, а також у плануванні завдань.

Основні характеристики Copilot є такими, як: технологія штучного інтелекту, інтеграція з іншими продуктами, допомога в контекстному плані, інструменти універсальності, безпека й конфіденційність даних користувача.

Перевагами використання Copilot є такі речі, як : відповіді на запитання отримуються без затримок, допомагає з рутинними процесами, допомагає навчатися і саморозвиватися. Недоліки Copilot є такими: немає можливості

генерувати відео, не підтримує деякі формати файлів, це не автоновна система, потрібні інтернет-з'єднання і актуальні дані.

Для більш правильного розуміння важливості теми даної роботи, слід звернутись до історії та контексту розробки Copilot. Розробляючи даний чатбот, Microsoft прагнула надати відповідь інтелектуальним ассистентам, що допомагають людям у всіх сферах повсякденого життя. Ідея виникла під час розвитку «LLM» та їхнього інтегрування у повсякденне життя людей. Microsoft намагалася створити такий чатбот, щоб ним могли спокійно користуватись усі користувачі. Наприклад, бізнесмени.

Сфери застосування Copilot є наступними:

1. медицина;
2. освіта;
3. повсякденне життя користувача;
4. бізнес-процеси;
5. розробка ПЗ;
6. творчі процеси.

Мовні моделі даного чатбота навчені на величезних текстах. Ця перевага передбачає такі можливості, як: аналізувати контекст запитання користувача, враховувати попередні повідомлення користувача у діалозі, надавати людські відповіді на запитання користувача, працювати у різних сферах дійяльності.

Тобто, під час використання нього, можна побачити людський стиль спілкування. Людина не буде бачити те, що даний чатбот генерує відповідь на його запит з локальних масивів відповідей.

У майбутньому, даний чатбот може отримати наступне: підтримку аудіо і відео, більш глибоко інтегровані інструменти розробки, офлайн-версію у базових сценаріях, розширенні функції для персоналізації, для розуміння індивідуальних потреб кожного користувача. Тобто, даний чатбот буде на постійній основі розвивати власний функціонал.

Grok – чатбот, що створений IT-компанією «xAI». Це був проєкт такою відомої особистості, як Ілон Маск. Даний чатбот названо в честь книги Дугласа Адамса «Автостопом по галактиці». Там «grok» має наступні значення: глибоко, інтуїтивне розуміння. Архітектура даного чатбота – xAI модель, інтеграція з соціальною мережею «X».

Ця архітектура має наступний вигляд:

1. grok-1;
2. grok-2;
3. grok-3;
4. grok-4;
5. grok-4.1.

Отже, Grok – це не простий чатбот. Це центральна нейронна мережа для соціальної мережі «X», адже він тренований на соціальних даних й є інтегрованою в рекомендації. Це є одним з популярних рішень. Адже це робить використання соціальних мереж більш зручним. Використання ШІ можливе не тільки в соціальній мережі «X». Наприклад, дану можливість може мати соціальна мережа «FaceBook».

Основні функції даного чатбота є наступними: відповіді на будь-які запити користувача, інтеграція з соцмережею «X», обробка в режимі реального часу. Переваги є наступними: актуальність даних, інтеграція з соцмережею X, швидкість відповіді, орієнтація на користувача. Недоліки є наступними: обмежена доступність, менше документації, залежність від екосистеми X.

Щодо обрання відповідного чатботу для користування, можна відзначити те, що даний процес вимагає індивідуального й концептуального вирішення самого користувача, що базується на висновках з теоретичної інформації про чатботи. Тобто, обираючи чатбот використання, користувачу варто враховувати не тільки функціонал, а й самі його можливості.

Youchat – чатбот на основі технології штучного інтелекту, що розроблений IT-компанією «You.com». Він працює, як універсальний

помічник для підвищення продуктивності та творчості. Наприклад, він допоможе дотримуватись плану на цілий день.

Основними характеристиками даного чатботу є наступне: способи відповіді й пояснення, генерування будь-якого контенту, мистецький контент, багатостильове спілкування, продуктивна робота. Що стосовно мобільного застосунку даного чатботу, то Google Play надає можливість завантажити окремий його мобільний застосунок «YouChat AI».

YouChat, як й Copilot, має великі мовні моделі, які навчені на великих текстових масивах. Це надає йому наступні можливості: розуміння суті запитання користувача, формулювання відповіді у людській формі, продовження розмови за контекстом, генерація різних текстів, робота з реальними даними, високоточні відповіді, підходить для навчання й досліджень, версія для корпорацій. Недоліки YouChat є наступними: після пивоту в enterprise, стало менше оновлень версій, claude чи o3 розумніше за YouChat в довгих тривалих дискусіях**,** менш зручний інтерфейс, голосовий режим, як й інші можливості, з'являються пізніше, ніж у конкурентів.

Agentgpt – це агент ChatGPT, що може створювати й запускати штучні інтелекти в браузері. Розроблено його стартапом «*Reworkd AI*». Він з’явився у квітні 2023 року. Принцип роботи «Agentgpt» полягає в тому, що він виконує запит користувача доти, доки він не виконає його повністю, шукаючи інформацію в інтернеті.

На відміну від ChatGPT, Agentgpt працює багатозадачно. Тобто, користувач може задавати багато запитів одночасно. Але, даний агент треба використовувати з певною обережністю. Тому, що даний агент, як й інші, не використовує необмежені ресурси.

Висновки до розділу 1

В даному розділі проведено аналіз існуючих програмних рішень для підтримки здорового способу життя. Даний аналіз продемонстрував переваги й недоліки програмних рішень, що варто враховувати як при обрані одного з них, так й при реалізації нових. Під час аналізу предметної області використання технологій ШІ, виявлено той факт, що, навіть, фізичні електронні пристрої можуть допомагати в плані ведення здорового способу життя. Наприклад, фітнес-браслети можуть міряти кількість пройдених кроків, пульс й швидкість серцебиття.

Також, проведено аналіз програмних систем, що допомагають в процесі ведення здорового способу життя. Виявлено той факт, що дані застосунки також можуть виконувати ті самі задачі, що й фізичні відповідні пристрої. Вони також можуть міряти кількість пройдених кроків, кількість витрачених або спожитих калорій й швидкість серцебиття. Виявлено те, що дані програмні системи можна також використовувати в якості віртуальних тренерів. Під час аналізу поняття віртуальних тренерів, виявлено той факт, що віртуальні тренери полегшують також ведення здорового способу життя шляхом надання можливості виконання фізичних вправ в довільному зручному для цього місці. Наприклад, за допомогою віртуального тренера можна виконувати фізичні вправи вдома.

Також, проведено аналіз сучасних чат-ботів на основі технології штучного інтелекту. Виділено переваги, недоліки, ключові й додаткові можливості деяких з них. Також, було проаналізовано історію даних чат-ботів й інформацію про їхніх розробників. В ході відповідного аналізу, виявлено той факт, що вони також допомагають вести здоровий спосіб життя.

2 ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЧАТ-БОТУ

2.1 Вибір технологій розробки клієнтської частини чат-боту

Перед початком реалізації проєкту кваліфікаційної бакалаврської роботи, проведено аналіз теоретичних відомостей відповідних технологій. Для реалізації проєкту кваліфікаційної роботи, встановлено наступні технології:

1. openAI;
2. fuse;
3. generateText;
4. nlp;
5. react-router-dom.

OpenAI – це набір офіційних інструментів, що дозволяють інтегрувати самі моделі через REST API. Основними можливостями OpenAI є наступне: типізація даних, автентифікація користувача, можливість синхронізування клієнта, сумісність з різними мовами програмування, можливість доступу до REST API. Технічними плюсами є наступне: автоматичне з’єднання, типізовані параметри й відповіді, синхронізація й асинхронізація клієнтів.

Fuse.js – це JavaScript-бібліотека для пошуку нечіткої відповіді. Основними характеристиками Fuse.js можна вважати таке, як: Logical search, Weighted keys, Nested search, Zero dependencies, Token search, Fuzzy search, малий розмір.

Дана бібліотека застосовується в наступних напрямках: пошук локальної відповіді, фільтрування списків, системи рекомендацій. Додаткові можливості даної бібліотеки є наступними: Include Matches, Include Score, Custom Sort, Extended Search Operators.

Цю бібліотеку розробив Крістофер Джонс. Це відбулось у 2012 році. З того часу, бібліотека є одна з найбільш поширеною серед розробників, в плані використання.

Вона використовується в продуктах у наступних компаній:

1. google;
2. atlassian;
3. spotify;
4. mongoDB;
5. nvidia.

На рис. 2.1 продемонстровано приклад використання Fuse.js. Вона використана в логіці відповідної функції.

```
/**
 * ========================================================
 * LOCAL SEARCH
 * ========================================================
 */

const findLocalAnswer =
  useCallback(
    (text) => {
      const results =
        fuse.search(text);

      if (
        results.length > 0 &&
        results[0].score < 0.45
      ) {
        return results[0].item;
      }

      return null;
    },
    [fuse]
  );
```

Рисунок 2.1 – Приклад використання Fuse.js

GenerateText – це інструмент виклику мовної моделі, щоб створити текст за промптом. Він використовується саме в якості зручності генерації даних. Основними можливостями даного інструмента є наступне: можливість підтримки різних моделей, можливість генерації тексту з промпту, застосування гнучких параметрів, можливість виклику інструментів.

Додатковими можливостями є таке, як: створення промпту, структурне виведення, стримінг, мультимодальність. На рис. 2.2 продемонстровано приклад використання GenerateText.

```
const runInitial = async() =>
{
  try{
    const {text} = await generateText({
      model: "openai/gpt-4o-mini",
      prompt: "Що таке ЗСЖ? Дай визначення українською."
    });
    setGeneratedText(text);
  }catch(err){
    console.error("Початкова генерація не вдалася: ", err);
  }

}
runInitial();
```

Рисунок 2.2 – Приклад використання GenerateText

NLP – напрямок ШІ, що може дозволити комп’ютерам розуміти людську мову. Основні завдання NLP являються такими: аналізування тексту, розуміння людської мови, мовний переклад, розпізнавання мови, аналіз емоційного забарвлення тексту.

Тобто, для повноцінної реалізації чат-боту, необхідно використовувати дану бібліотеку. Адже, вона робить аналіз запиту користувача й шукає відповідну відповідь. На рис. 2.3 продемонстровано приклад використання NLP.

```
const handleNLP = (text) =>
{
    const doc = nlp(
     text
    );
return{
        nounc: doc.nouns().out("array"),
        verbs:doc.verbs().out("array"),
    };
  }
  }
```

Рисунок 2.3 – Приклад використання NLP

React-router-dom – інструмент для створення маршрутів. Це один з зручних інструментів. Основними можливостями даного інструмента являється наступне: перенаправлення користувача через відповідний хук, маршрути всередині інших макетів, маршрути з відповідними параметрами, рендеринг компонентів в залежності від поточної URL-адреси, перемикання між сторінками без повного перезавантаження.

Потенційні ризики являється наступними: конфлікти версій, потреба в пререндерингу й складність вкладених маршрутів. На рис. 2.4 продемонстровано приклад використання react-router-dom.

```
const router = createBrowserRouter([
  {
    path: "/admin",
    element: <Admin />,
  }
]);

return (

 <Routes>

  <ThemeProvider theme={currentTheme}>

    <CssBaseline />


    <div className="App">
      <aside className="sidemenus">
        <div
          className="side-menus-button"
          onClick={() => {
            setChatLog([{ user: "gpt", message: "Новий чат розпочато!" }]);
          }}
        >
          + Новий чат
        </div>
```

Рисунок 2.4 – Приклад використання react-router-dom

Даний інструмент має такі ключові елементи, як: Route, Routes, BrowserRouter, NavLink, Link, useNavigate, useParams. Дані елементи можна ефективно використовувати в реалізації адміністративної панелі.

2.2 Вибір технологій розробки серверної частини чат-боту

Express.js – це один з потужних фрейворків для Node.js, який дозволяє розробляти серверну частину вебзастосунку [7]. На разі, цей інструмент є популярним через власну зручність у впровадженні й використанні.

Основними характеристиками Express.js є наступні речі: middleware, маршрутизація, гнучкість, масштабованість. Додатковими можливостями даної бібліотеки є такі речі, як: middleware, підтримка роботи з шаблонізаторами, безпечна маршрутизація й утиліти для інтерфейсів програмних забезпечень. На рис. 2.5 продемонстровано приклад застосування даної технології.

```
import express from "express";

const app = express();
```

Рисунок 2.5 – Приклад використання Express.js

CORS – це спеціальний механізм безпеки, що має кожен сучасний браузер, що, в свою чергу, дозволяє вебзастосункам отримувати ресурси з іншого домену, якщо сервер явно дозволяє це через спеціальні HTTP-заголовки. Без CORS браузери блокують такі запити через політику "same-origin". Основні поняття для CORS: CORS, HTTP-заголовки, same-Origin Policy. CORS працює наступним чином: простий запит, preflight-запит, відповідь сервера.

Тому, CORS є одним з використовуваних інструментів для захисту даних. Адже, він перевіряє доступ інших доменів до сервера вебзастосунку.

Основними заголовками є наступне:

– access-Control-Allow-Origin;
– access-Control-Allow-Methods;
– access-Control-Allow-Headers;
– access-Control-Allow-Credentials.

Ризики та помилки є наступними: неправильна конфігурація, CORS-атаки, рішення. Корисні ресурси є наступними: MDN Web Docs: детальна довідка по CORS, GeeksforGeeks. На рис. 2.6 продемонстровано приклад використання CORS.

```
import express from "express";
import cors from "cors";

const app = express();
app.use(cors());
```

Рисунок 2.6 – Приклад використання CORS

I18next – це бібліотека «JavaScript» для налаштування локалізації вебзастосоунку, що набула великої популярності. Ця бібліотека впроваджується наступним шляхом: встановлення бібліотеки через «npm», імпорт бібліотеки в відповідних файлах з розширенням «.js».

Основні можливості I18next є наступними: визначення мови користувача, гнучке завантаження перекладів, плюралізація та контекст, кешування, інтерполяція змінних. Додаткові можливості даної бібліотеки являються наступними: граматика та множини, використання перекладів, що знаходяться у середині інших, робота з різними типами даних, кешування й бекенд-частини.

Meyda – JavaScript-бібліотека, що витягує аудіо-інструменти. Це, на кшталт WebSpeechAPI, що можна використовувати як онлайн в браузері, так й офлайн, встановивши через npm.

Основними характеристиками Meyda є такі речі, як: репозиторії Meyda розміщені на GitHub, пакет доступний виключно через npm. Головними можливостями Meyda є такі речі, як: витягування інструментів в режимі реального часу, стандартні інструменти, офлайн аналіз.Обмеження та ризики є наступними: не з усіма браузерами сумісний, можна порушити приватність, менш продуктивний інструмент.

Meyda корисна з наступних причин: Node.js та браузер, Wellness-чатботи, React-проєкти, ML-підходи**.** На рис. 2.7 продемонстровано приклад використання Meyda.

```
import Meyda from "meyda";


// Meyda feature extraction (requires audio context + source)
const audioContext = new AudioContext();
const source = audioContext.createBufferSource();
// You need to load audio data into `source.buffer` before starting
const features = Meyda.extract(["mfcc"], source);
console.log(features);
```

Рисунок 2.7 – Приклад використання Meyda

CryptoJS – це «JavaScript»-бібліотека для шифрування й забезпечення безпеки даних, що набрала велику популярність серед розробників програмного забезпечення. Ця бібліотека підтримує наступні типи шифрування даних:

1. aes;
2. sha-256;
3. md5;
4. hmac.

Основні факти про CryptoJS є наступними: призначена для набору криптографічних алгоритмів на чистому «JavaScript», алгоритми: SHA-256, AES, SHA-1, її ліцензія безкоштовна, останньою версією є4.2.0, її розробка припинена. Обмеження є наступними: без перевірки безпеки, MD5 та SHA-1, краще використовувати Web Crypto API або crypto у Node.js. Альтернативи є наступними: Node.js crypto module, Web Crypto API, libsodium.js.

Існують такі обмеження стосовно використання CryptoJS: припинення розробки, безпечніше використання вбудованого модуля «crypto», низька сумісність. На рис. 2.8 продемонстровано приклад використання CryptoJS.

```
Ask GetbotAI
import CryptoJS, { AES, SHA256 } from "crypto-js";


export const saveToLocalStorage = (key, value) => {
  const encrypted = AES.encrypt(JSON.stringify(value), SECRET_KEY).toString();
  localStorage.setItem(key, encrypted);
};

export const getFromLocalStorage = (key) => {
  const encrypted = localStorage.getItem(key);
  if (!encrypted) return null;

  try {
    const bytes = AES.decrypt(encrypted, SECRET_KEY);
    const decrypted = bytes.toString(CryptoJS.enc.Utf8);
    return JSON.parse(decrypted);
  } catch (error) {
    console.error("Decryption failed:", error);
    return null;
  }
};

// Example SHA256 usage (for hashing, not reversible encryption)
```

Рисунок 2.8 – Приклад використання CryptoJS

Виходячи з вищесказаного, можна зробити висновок про те, що CryptoJS є одним з доцільних для використання у криптографії. Тому, його застосування варто уваги.

DOMPurify – це JavaScript-бібліотека для санітизації HTML-коду від шкідливого коду. Ця бібліотека також набула великою популярності. Основними характеристиками DOMPurify є наступне: призначена для прибирання шкідливого коду з HTML-контенту, підтримує наступні формати: HTML, SVG, MathML, захищає від XSS і DOM, простота інтеграції, підтримує усі нові версії веббраузерів.Інтеграція у проєкти відбувається таким чином: React/Angular/Vue, Node.js, Wellness-чатботи та UI.

Особливо, це корисно з наступних цілей:

1. захистити користувачів від ін’єкцій шкідливого коду;
2. підвищити довіру до застосунка, адже контент буде безпечним;
3. спрощувати роботу з HTML-вставками у React чи Node.js без ризику XSS.

Дані властивості можуть мати не актуальний спосіб роботи з даним інструментом. Тому, варто бути обережними з ними й виконувати вимоги для їхнього використання. Розширені можливості DOMPurify є наступними: конфігурація результатів, контроль контенту, елементи та атрибути, можна явно дозволяти або забороняти певні теги чи атрибути. DOMPurify має hook API, що дозволяє розширювати поведінку: beforeSanitizeElements, afterSanitizeAttributes, uponSanitizeElement, uponSanitizeAttribute.

Jwt-decode – це невелика JavaScript-бібліотека, яка дозволяє швидко розібрати (decode) JSON Web Token (JWT) і отримати його payload та header без перевірки підпису. Важливо: вона не перевіряє валідність токена, лише розбирає його структуру. Тобто, для повної перевірки підписів, важливо застосовувати не тільки дану бібліотеку.

Основні факти є наступними: призначення для розбірки JWT у браузері чи Node.js, щоб отримати claims, без перевірки: бібліотека не перевіряє підпис токена. Для безпеки завжди треба робити валідацію на сервері. Підписані JSON-токени мають бути описані JWS специфікацією «RFC 7515».

Також є такі підтримувані варіанти рекомендованих алгоритмів, як:

– hs384, hs512;
– rs384, rs512;
– es384, es512.

Абревіатури курсивом – це назви інструментів, що використовуються JSON-токенами, що описує специфікація «JWA». Якщо JWT-токен є підписаним, то можна застосувати наступні атрибути: jku, jwk, kid, crit. Зазвичай, в заголовках HTTP, JWT передається такими способами, як: в полі Authorization,в полі Cookie. Дані процеси є вигідними в плані безпеки, адже вся необхідна інформація зберігається в надійному місці й вона унікальна.

У поточному році DOMPurify оновлено до версії. 3.4.2, що включає в себе такі виправлені речі, як: виправлена URI-валідація, розширені тести, покращену підтримку сучасних Node.js-версій і браузерів. Дані речі роблять його ще більше надійнішим від XSS-атак й DOM-атак.

Наразі, автоматизовані тести DOMPurify охоплюють до 12 браузерів. Також, він має оптимізований fuzz-harness із новими інваріантами XSS. Тому, даний інструмент можна використовувати для покращення рівня безпеки вебзастосунку. Цей інструмент є важливим для застосування під час розробки компонентів безпеки вебзастосунку. Адже, саме він гарантує якісну й безпечну роботу. Роботу даних методів можна порівняти наступним чином (табл. 2.1).

Таблиця 2.1 – Порівняння методів передачі «JWT»

Метод	Bearer-Token	Cookie
Заголовок	Bearer <JWT-Token>	token=<JWT-Token>
CORS	Працює з CORS на JavaScript.	Cookie зберігаються в браузері лише для конкретного домену.
Зберігання	Можливі використовувати всі доступні способи зберігання для JavaScript.	Cookie розміщуються в Cookie-Store.
Захист від MITM	Наявність TLS повинна перевірятись виключно в JavaScript.	Коли на coockie встановлюється прапорець secure, примусово застосовується TLS.
Захист від XSS	Повинен реалізовуватись виключно в JavaScript.	Якщо для coockie встановлено прапорець HttpOnly, то захист від XSS є неявним через не надання доступу JavaScript
Захист від CSRF	Є неможливим через необхідність застосування інших заходів.	Повинен реалізовуватись виключно в JavaScript

2.3 Аналіз технологій захисту даних

XSS – це вид вразливості у вебзастосунках, за допомогою чого зловмисник може вставити шкідливий JavaScript-код. Даний код може виконатись в браузері іншого користувача, як частина сайту. Основними видами XSS є наступне: stored XSS, dOM-based XSS, reflected XSS. Щодо захисту даних, то застосовують такі практики, як: екранування, валідація коду й content security Policy.

Під час роботи з XSS, існують типові помилки. Серед них це: немає заборон на завантаження SVG-файлів, CSP має неправильну конфігурацію, «innerHTML» використовується без очищення. На рис. 2.9 продемонстровано приклад обробки XSS-атак.

```
Ask GetbotAI
import React, {useState} from "react";
import { DOMPurify } from "dompurify";


// create a XSS class
class XSSProtect {
  constructor(){
    this.logs = [];
    this.xssAttack = false;

  }

  // create a method of handling XSS
sanitizeHtml(html){
const clean= DOMPurify.sanitize(html);
this.logs.push({original: html, clean, time: new Date()});
}

getLogs () {
  return this.logs;
}
}
```

Рисунок 2.9 – Приклад обробки XSS-атак

CSRF – атака, яка змушує браузер користувача поводитись небажаним чином на сайті, де користувач вже авторизований.

Основними принципами захисту від CSRF-атак є наступне:

1. метадані й заголовки сайтів;
2. атрибут Samesite Cookies;
3. використання CSRF-токенів;
4. додаткові практики запобігання CSRF-атак;
5. синхронізатор токенів;
6. double submit cookies.

Методи CSRF можна порівняти між собою (табл. 2.2).

Таблиця 2.2 – Порівняння методів CSRF

Метод	Переваги	Недоліки
CSRF-токени	Надійний захист від CSRF	Потреба в інтеграції в усі форми
SameSite Cookies	Просте налаштування безпеки	Немає підтримки всіх сценаріїв
Fetch Metadata Headers	Захист API від CSRF	Не підтримує старі браузери

2.4 Специфікація вимог

Призначення та межі проєкту

1. підтримка здорового способу життя ;
2. функціонал контролю здоров’я;
3. орієнтація на користувачів країн Європи.

Загальний опис

1. сфера: охорона здоров’я;
2. цільова аудиторія: молоді люди віком до 25 років;
3. інтерфейс: графічний, інтерактивний, адаптивний;
4. обмеження: тематичні запити лише у сфері здорового способу життя.

Функції системи

Трекер води

1. визначає достатність кількості спожитої води;
2. джерела: дані користувача;
3. формує рекомендації щодо добової норми;

Трекер пульсу

1. визначає рівень пульсу;
2. джерела: дані користувача, інтеграція з пристроями;

3. формує оцінку стану серцево-судинної системи.

Трекер тиску

1. визначає рівень артеріального тиску;
2. джерела: дані користувача;
3. формує оцінку стану здоров’я.

Вимоги до інформаційного забезпечення

1. використання перевірених вебсайтів про здоровий спосіб життя;
2. централізовані та безпечні джерела;
3. фільтрація інформації для уникнення дезінформації.

Вимоги до технічного забезпечення

1. використання сучасної операційної системи та апаратного забезпечення.

Вимоги до програмного забезпечення

1. архітектура: 3-tier web application;
2. чатбот на основі технологій штучного інтелекту;
3. база даних для збереження даних користувачів;
4. технології: JavaScript + React.js.

Вимоги до зовнішніх інтерфейсів

1. графічний дизайн;
2. мобільна адаптація;
3. зрозумілий вміст;
4. логотип;
5. використання HTTPS.

Властивості програмного забезпечення

1. висока доступність;
2. підтримка користувачів;
3. вебсайт, як основна платформа;
4. багатозадачність;
5. надійна логіка;

6. сучасні стандарти кібербезпеки;

7. коректний час відповіді системи.

Визначення меж проєкту

– даний проєкт передбачає процес розробки клієнтської та серверної частин;

– розробка інфраструктури датасетів;

– інтеграція з іншими пристроями;

– включає розробку мобільного застосунка;

– охоплює апаратні частини сервера;

– Загальна структура і склад системи;

– клієнтська частина: адаптивний інтерфейс, реалізований на Vite;

– серверна частина: Node.js-сервер, передаючий запити базі даних на PostreSQL;

– інтеграція: інтеграція до інших пристроїв (фітнес-браслетів, годинників);

– панель адміністратора: окрема сторінка для виконання спеціальних функцій: забезпечення безпеки й дотримання відповідних рекомендацій .

– загальні обмеження;

– потреба в наявності інтернета;

– залежність кількості користувачів, що знаходяться онлайн одночасно, від можливості витримки сервера;

– обмеження на кількість операцій обробки транзакційних запитів, залежно від можливостей фізичного сервера.

Додаткові вимоги

1. користувач має дотримуватись принципів цензури та етичного використання системи.

Висновки до розділу 2

В другому розділі проведено аналіз обраних технологій реалізації клієнтської частини вебзастосунку. Дані технології можуть надають можливість генерування відповіді. Також, дані технології надають можливість користувачу обирати модель нейронної мережі для подальшої взаємодії з нею. Кожна модель буде відповідати користувачу по різному, адже дані моделі працюють на основі генерації випадкової відповіді користувачу за певною темою спілкування. Також, за допомогою їхнього застосування можна реалізувати функціонал для взаємодії з користувачем. Наприклад, можна створити власний функціонал генерації відповіді.

Проведено аналіз стосовно обрання технологій реалізації серверної частини чатботу. Виявлено той факт, що технології даного напрямку можуть бути застосовані як для роботи з базою даних, так й для роботи з внутрішньою частиною сервера. За допомогою даних технологій, сервер отримує запит користувача й звертається до бази даних стосовно даного запиту. Потім, сервер отримує відповідь від бази даних й надає її клієнтській частині.

Проаналізовано технології захисту даних користувача. Це ті технології, що можуть захистити користувача від наслідків шкідливого коду чи його браузер від небажаної поведінки під час відвідування вебсайту.

Наприклад, щоб браузер користувача не завантажував небажані файли з вебсайту. В свою чергу, проведено аналіз відповідних методів застосування даних технологій.

Розроблено специфікацію вимог для вебзастосунку. В тому числі, визначено призначення та межі проєкту й загальний опис програмної системи, що реалізується.

3. АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура мовної моделі штучного інтелекту для реалізації чатботу

На сьогодні, сучасні чат-боти орієнтовані на такі архітектури, як: LlaMA, Gemini, GPT. Вони також використовують наступні підходи NLP: CNN, RNN, seg2seg. Це застосовується для розуміння людських мов спілкування. GPT є однією з найпоширеніших архітектур через те, що вона виконує генерацію зв’язного тексту та підтримку контексту запиту користувача.

Для проєкту обрано GPT-архітектуру. Переваги GPT-архітектури є такими, як: легка інтеграція через API у React.js, глибокі знання моделі з багатьох сфер, підтримка усіх мов й стилів, створення текстів у будь-якому стилі й здатність моделі утримувати довгі діалоги й враховувати попередні запити користувача. Недоліками даної архітектури є такі речі, як: потреба хмарних сервісів або потужних вебсерверів, не знання моделі реального світу, обмеження за контекстом, недостовірна інформація по фактах, генерація упередженого або небезпечного контенту. Перевагами LlaMA є такі речі, як: код, що знаходиться у відкритому доступі, ефективна робота, великий масштаб, різноманітне застосування, низька ймовірність відмови.

Недоліками LlaMA є такі речі, як: велика кількість обчислювальних витрат, вигадування фактів, складне налаштування, обмежений контекст, ризики небезпечного контенту. На рис. 3.1 продемонстровано приклад впровадження GPT-архітектури.

```
const handleUseModel = async () =>
{
  const response = await createAgent({
    model: "gpt-4o-mini",

  })
  setModel(response);

}
```

Рисунок 3.1 – Впровадження GPT-архітектури

Gemini – набір великих мовних моделей, що реалізував Google. Даний набір є конкурентом для GPT-архітектури. Поєднуючи відповідну потужність й багатозадачність, він може розуміти аудіо, відео, текст й зображення користувача. Архітектура даного набора базується на наступних речах: трансформер, інтеграція з продуктами від Google і Youtube, багатозадачність й покращення навчання.

Досліджуючи Gemini, варто дослідити його історію. Першу версію Gemini створено в 2023 році. В ньому створено першу модель LLM, що інтегрована у Bard. У 2024 році створено версію 1.5 Gemini, де збільшено кількість токенів до 1 мільйона. У 2024-2025 роках створено актуальні версії Gemini. Це Nano/Pro/Ultra. Вони відрізняються тим, що вони застосовують різні потужності

Перевагами Gemini є такі речі, як: багатозадачність, можливість написання довгого повідомлення, взаємодія з іншими продуктами Google, оптимізація для мобільних пристроїв, висока якість кодингу на різних мовах програмування. Недоліками Gemini є наступні речі: ризик порушення етичності контенту, вигадування фактів, потреба в потужних GPU, неповний доступ до функціоналу, закритий API.

Виходячи з даного контексту, можна сказати те, що, на даний час, Gemini є однією з актуальних архітектур штучного інтелекту. Gemini має актуальні свої версії й її інтегрування у сучасні програмні засоби є одним з

логічно обгрунтованих рішень. Адже, Gemini має інтеграцію з продуктами декількох IT-компаній. Тобто, даний інструмент для розробки штучного інтелекту, є одним з сучасних інструментів для забезпечення зручності користувача, незалежно від соціального статусу. Його, як й інші, можуть використовувати люди всіх форм зайнятості. Немає ніяких обмежень стосовно фінансового благополуччя користувачів.

3.2 Архітектура чатботу

Для дослідження архітектурних підходів реалізації проєкту обрано такі архітектури, як: гібридна, мультимодальна, локальна і клієнт-серверна.

Клієнт-серверна архітектура – архітектура, що розподіляє розробку програмної системи на клієнтську й серверну частини. Її переваги полягають в наступному: у гнучкості, масштабованості, безпеці й модульності. А недоліки полягають у: можливості затримки, навантаженні на сервер й залежності від мережі.

Гібридна архітектура – архітектура, що поєднує декілька інших архітектур або технологій для отримання переваг кожної з них. Наприклад, вона може поєднувати клієнт-серверну й локальну архітектури. Переваги даної архітектури полягають у таких речах, як: комбінування хмарних сервісів й локальної мережі, запам’ятовування користувача, можливість використання локальної моделі й легке додавання нових модулів. Недоліки полягають у таких речах, як: складність налаштування й підтримки, постійна недостатність ресурсів й потреба в ретельному контролі доступу до API-ключів.

Локальна архітектура – архітектура, що дає можливість запуску програмного продукту й сервера для нього на локальному комп’ютері. Тобто, дана архітектура застосовує локальний сервер для запуску застосунка. Переваги даної архітектури є такі речі, як: неможливість виходу даних користувачів за межі архітектури, незалежність роботи від зовнішнього API, vможливість навчання моделей на власних датасетах, відсутність потреби в

постійних витратах на API-запити. Недоліками даної архітектури є такі речі, як: потреба в потужних GPU, потреба в налаштуванні інструментарію для розгортання, потреба в постійних оновленнях, складність обслуговування великої кількості користувачі без застосування хмарних сервісів.

Мультимодальна архітектура – архітектура, за допомогою чого програмна система може працювати з декількома типами даних. Перевагами даної архітектури являються: можливість користувача говорити голосом, поєднання текстових пояснень з візуалізацією або аудіо, можливість застосування в різних сферах, можливість створення нових форматів застосунків. Недоліками даної архітектури є: потреба в високих ресурсах, неможливість легкої інтеграції, можливість неправильних інтерпретацій даних, обмежена можливість доступу.

Для реалізації проєкту обрано клієнт-серверну архітектуру. На рис. 3.2 продемонстровано приклад даної архітектури.

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'



// https://vite.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      "http://loclahost:5173/": "http://localhost:5000"
    }
  }
})
```

Рисунок 3.2 – Приклад клієнт-серверної архітектури

Варто зазначити те, що сервер може працювати на будь-якому вільному порті. Головне, цей порт має бути активованим й працювати виключно з серверами.

3.3 UML-моделювання програмного забезпечення чатботу

Для проєктування ПЗ створено такі діаграми, як: діаграма прецендентів, діаграма пакетів, діаграма компонентів, діаграми взаємодії, діаграми класів, діаграми станів та переходів й діаграми діяльності. Як зазначалось раніше, створення діаграм для програмної системи дає розуміння способу її реалізації.

Перевагами діаграми прецендентів є такі речі, як: легкість для всіх користувачів, візуалізація функціональних вимог, допомога в визначені ролей користувачів й їхньою взаємодією з системою. Недоліки діаграми прецендентів є такі речі, як: прихованість внутрішньої логіки, занадто велика кількість загальних фактів для створення продукту. На рис. 3.3 продемонстровано приклад діаграми прецендентів.

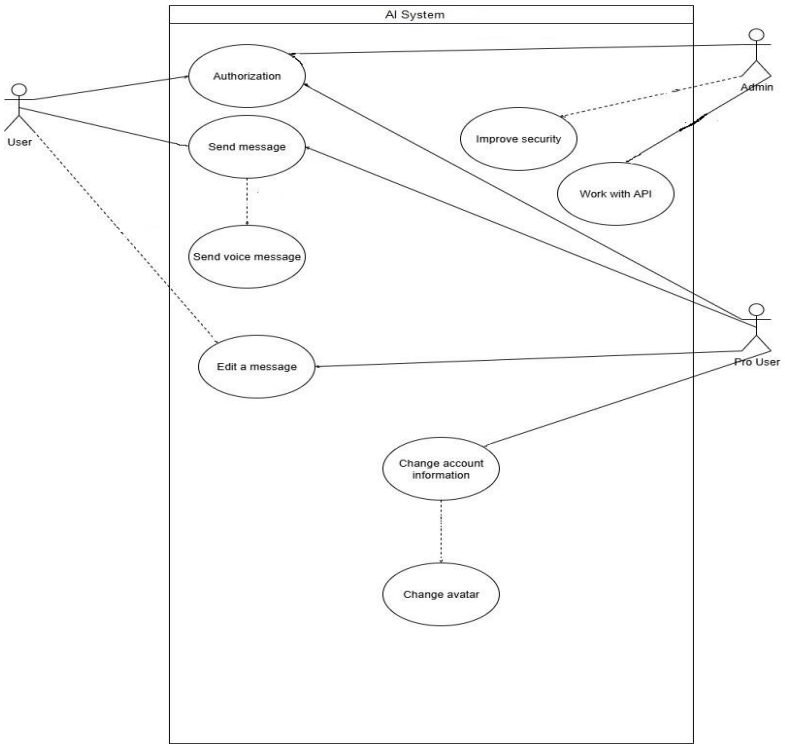

Рисунок 3.3 – Діаграма прецендентів

Далі, створено діаграму послідовності з запитом стосовно їжі. В ній продемонстровано наступні повідомлення:

1. користувач вводить відповідний запит;
2. запит відправляється до інтерфейсу;
3. запит відправляється до контролеру;
4. запит відправляється до моделі;
5. модель аналізує запит;
6. модель звертається до бази даних;
7. база даних звертається до модуля;
8. модуль генерує відповідь на основі машиного навчання;
9. контролер повертає відповідь;
10. інтерфейс повертає відповідь користувачу.

На рис. 3.4 продемонстрована діаграма послідовності стосовно їжі.

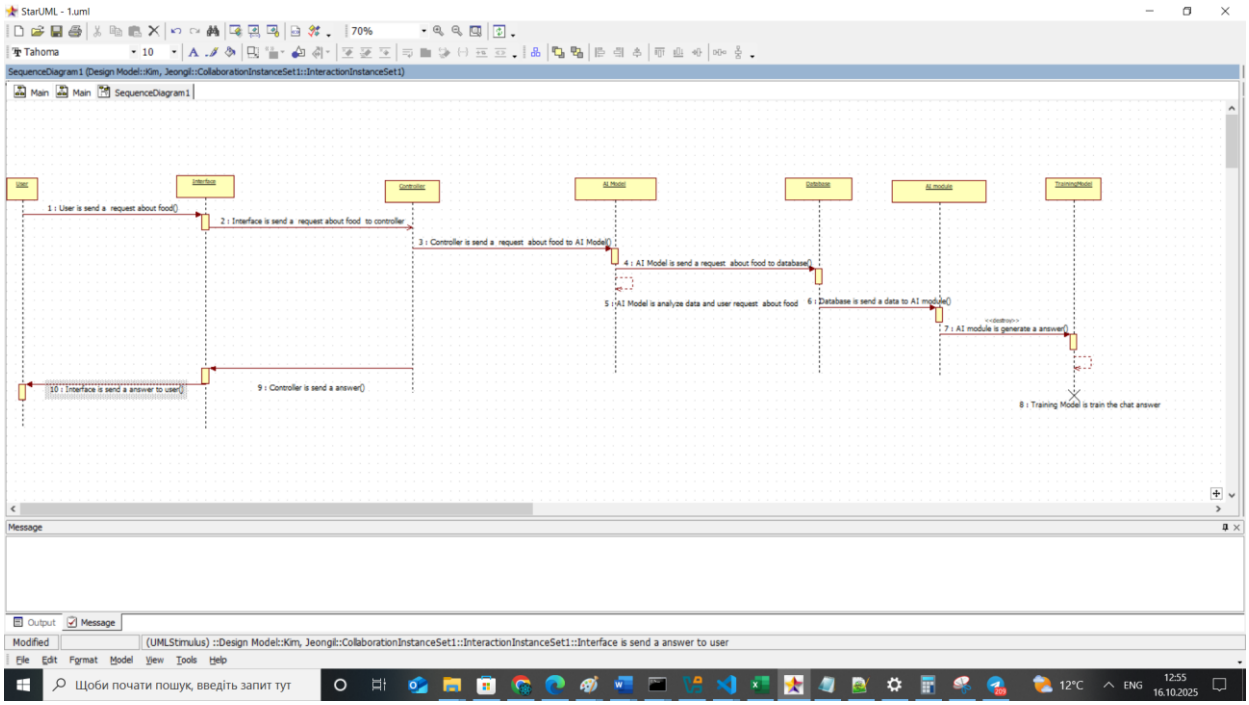

Рисунок 3.4 – Діаграма послідовності стосовно їжі

Також для проєктування програмної системи, створено діаграму класів. На діаграмі продемонстровано наступні елементи: клас «User», клас «Interface», клас «Controller», клас «Model», клас «Database», клас «AIModule», клас «TrainingModel», клас «Template Request», клас «Create Request», клас «Activate Request», клас «WebApplication», клас «User Request», клас «Handle Request». На рис. 3.5 продемонстровано діаграму класів.

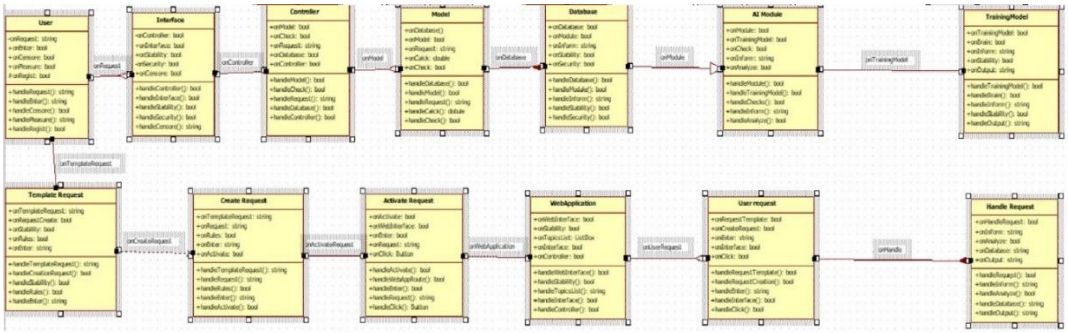

Рисунок 3.5 – Діаграма класів

Під час проєктування системи також створено діаграму компонентів.. На діаграмі продемонстровано наступні елементи: swimlane “User”, swimlane “Programmers”, swimlane “Design”, swimlane “Cloud”, swimlane “AIChat”, swimlane “Support”, стан “EnterRequest”, стан “ProgrammingAnalyze”, стан “CreateGoodDesign”, стан “SearchingInform”, стан “FormatAnswer”, стан “Support”, зв’язок “There_are_tests_failure”, зв’язок “All_ok”. На рис. 3.6 продемонстровано діаграму компонентів.

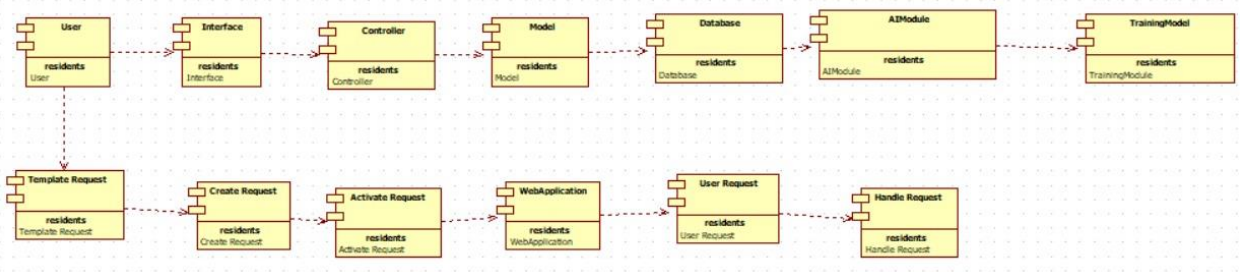

Рисунок 3.6 – Діаграма компонентів

Також створено діаграму станів. На діаграмі продемонстровано наступні елементи:

1. стан “Entering”;
2. стан “Waiting”;
3. стан “Outputing”;
4. стан “Confirm”;
5. стан “Analyzing”;
6. стан “Handle Request”;
7. зв’язок “OnInit [all_ok] / Tone”;
8. зв’язок “OnEntered”;
9. зв’язок “OnWaiting”;
10. зв’зок “OnOutputed”;
11. зв’зок “OnConfirmed”;
12. зв’язок “OnHandleing”;
13. зв’язок “OnAnalyzing”;
14. зв’язок “[data_is_valid] / EnterToMainPage”;
15. InitialState.

На рис. 3.7 продемонстровано діаграму станів. Вона також відображає процес обробки запиту користувача.

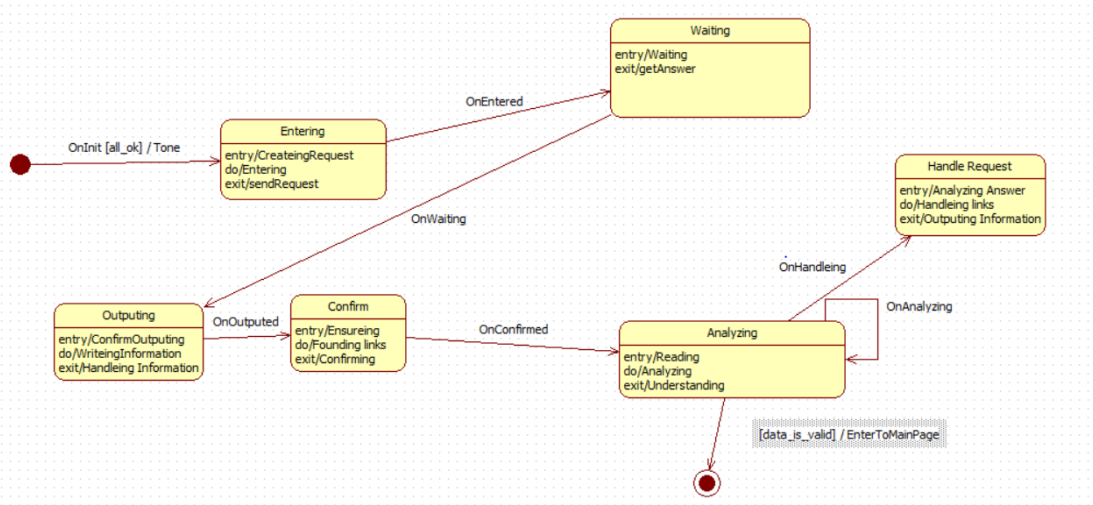

Рисунок 3.7 – Діаграма станів

Також створено діаграму діяльності для системи. На діаграмі продемонстровано наступні елементи:

1. swimlane “User”;
2. swimlane “Programmers”;
3. swimlane “Design”;
4. swimlane “Cloud”;
5. swimlane “AIChat”;
6. swimlane “Support”;
7. стан “EnterRequest”;
8. стан “ProgrammingAnalyze”;
9. стан “CreateGoodDesign”;
10. стан “SearchingInform”;
11. стан “FormatAnswer”;
12. стан “Support”;
13. зв’язок “There_are_tests_failure”;
14. зв’язок “All_ok”.

На рис. 3.8 продемонстровано діаграму діяльності. Вона демонструє саме ланки взаємодії між користувачем й відповідною IT-компанією, що створила програмний продукт.

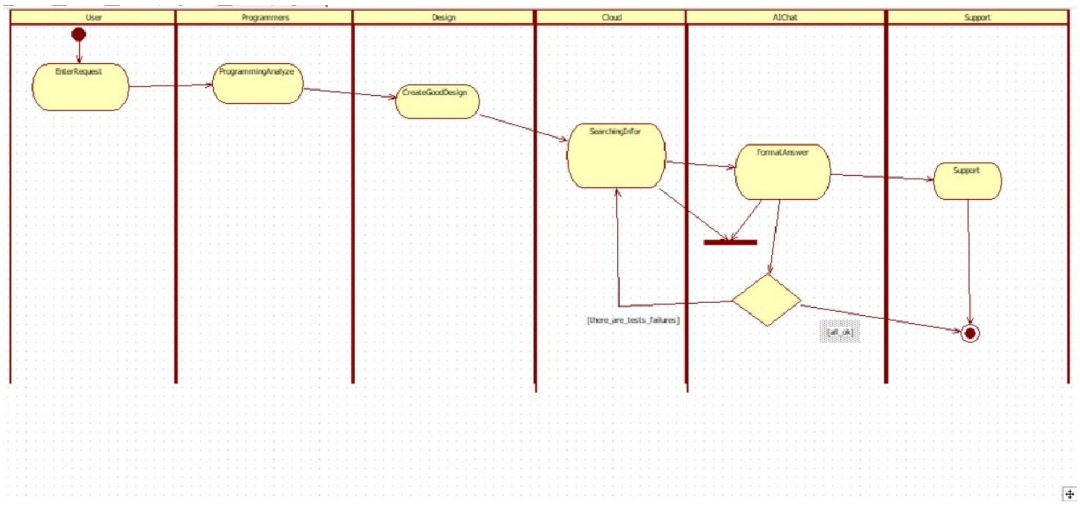

Рисунок 3.8 – Діаграма діяльності

Також створено діаграму пакетів. Під час створення діаграми пакетів, створено такі елементи, як:

1. пакет “User”;
2. пакет “Interface”;
3. пакет “Controller”;
4. пакет “Model”;
5. пакет “Database”;
6. пакет “AIModule”;
7. пакет “TrainingModel;
8. пакет “Create RequestTemplate”;
9. пакет “Create Request”;
10. пакет “Activate Request”;
11. пакет “WebApplication”;
12. пакет “User Request”;
13. пакет “Handle Request”.

На рис. 3.19 продемонстровано діаграму пакетів. Дана діаграма демонструє те, що саме в плані пакетів буде реалізовано.

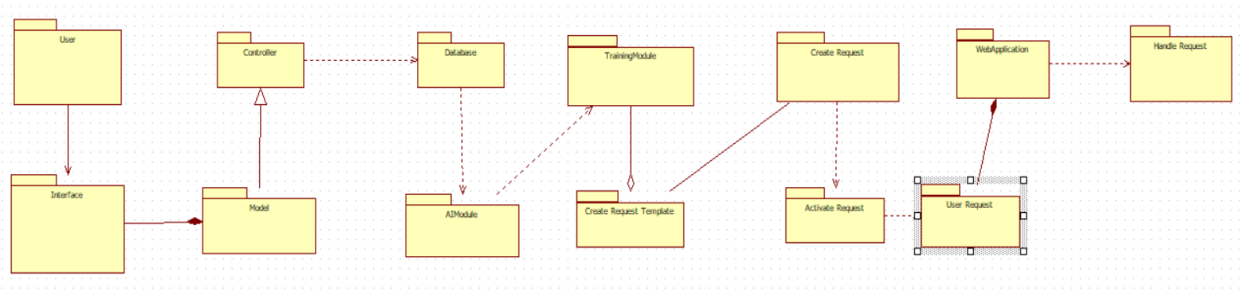

Рисунок 3.19 – Діаграма пакетів

Діаграма пакетів реалізовано для розуміння способу структурного збереження даних. Наприклад, конфіденційних даних.

3.4 Мокапи для програмного забезпечення

Для реалізації проєкту, реалізовано два відповідні мокапи. На рис. 3.10 продемонстровано перший мокап проєкту.

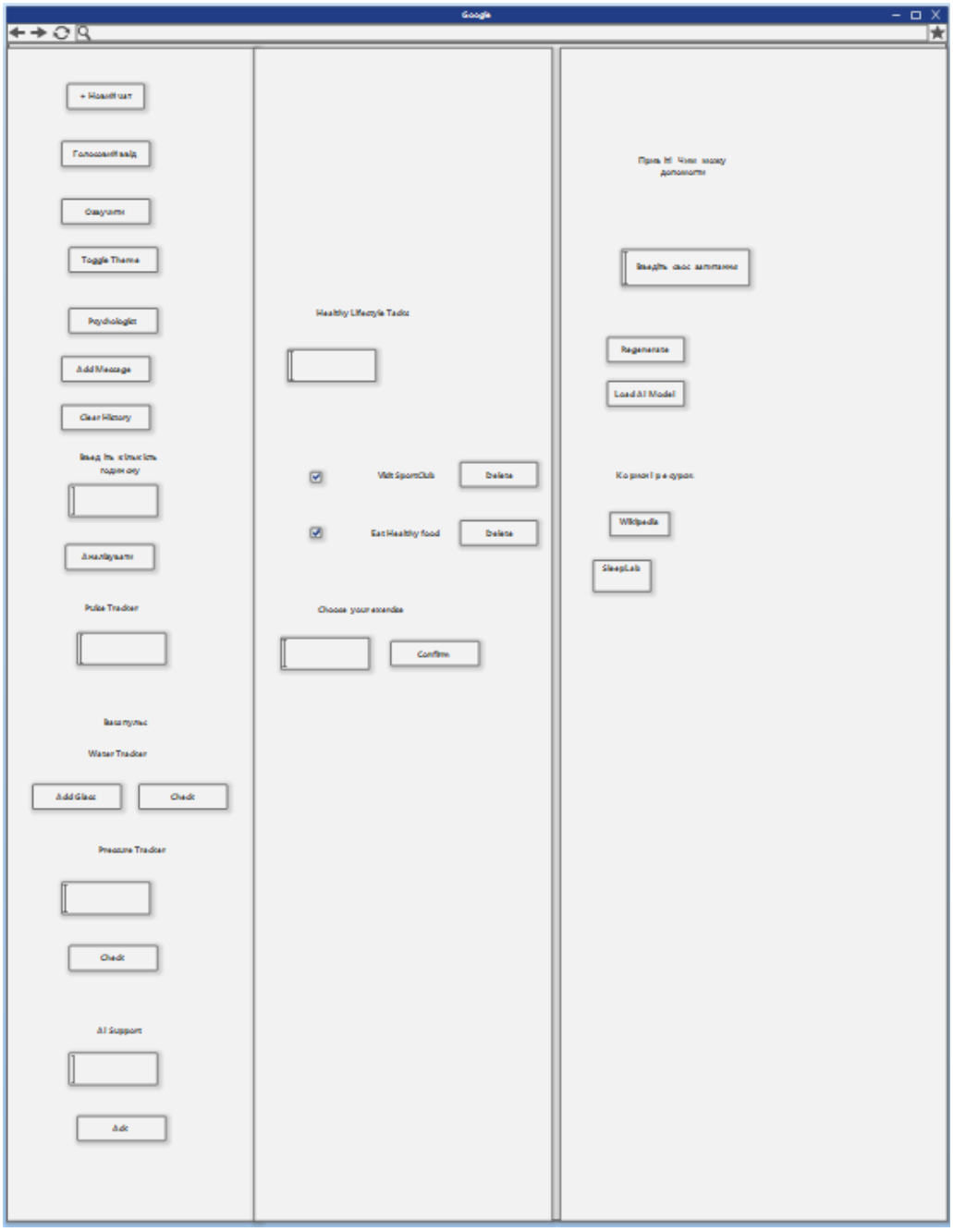

Рисунок 3.10 – Перший мокап проєкту

Даний мокап є унікальним рішенням. Це забезпечить особливу популярність серед користувачів.

Другий мокап відрізняється від першого розташуванням деякого функціоналу. На рис. 3.11 продемонстровано другий мокап.

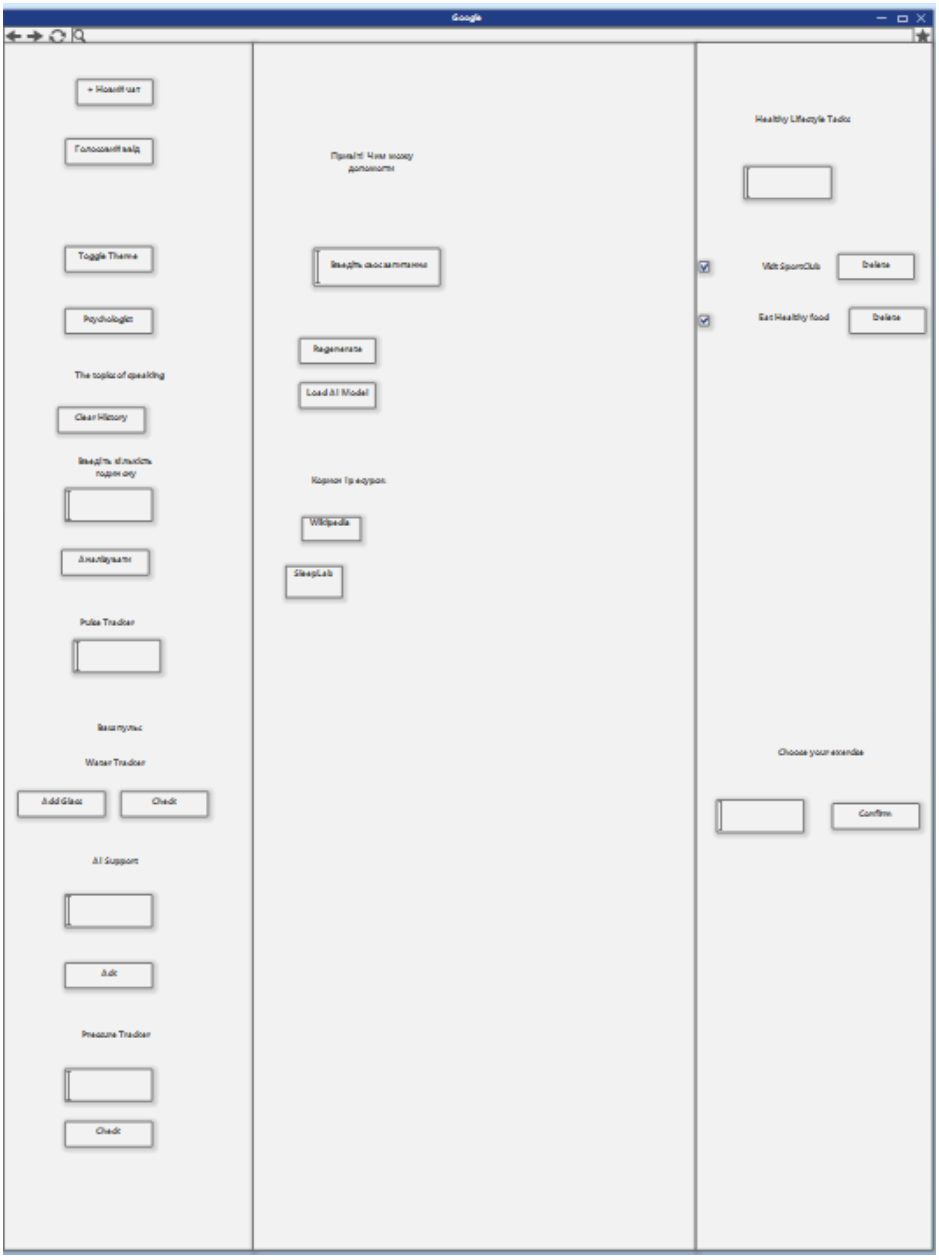

Рисунок 3.11 – Другий мокап

Другий мокап також забезпечить зручність для користувача. На рис. 3.12 продемонстровано третій мокап.

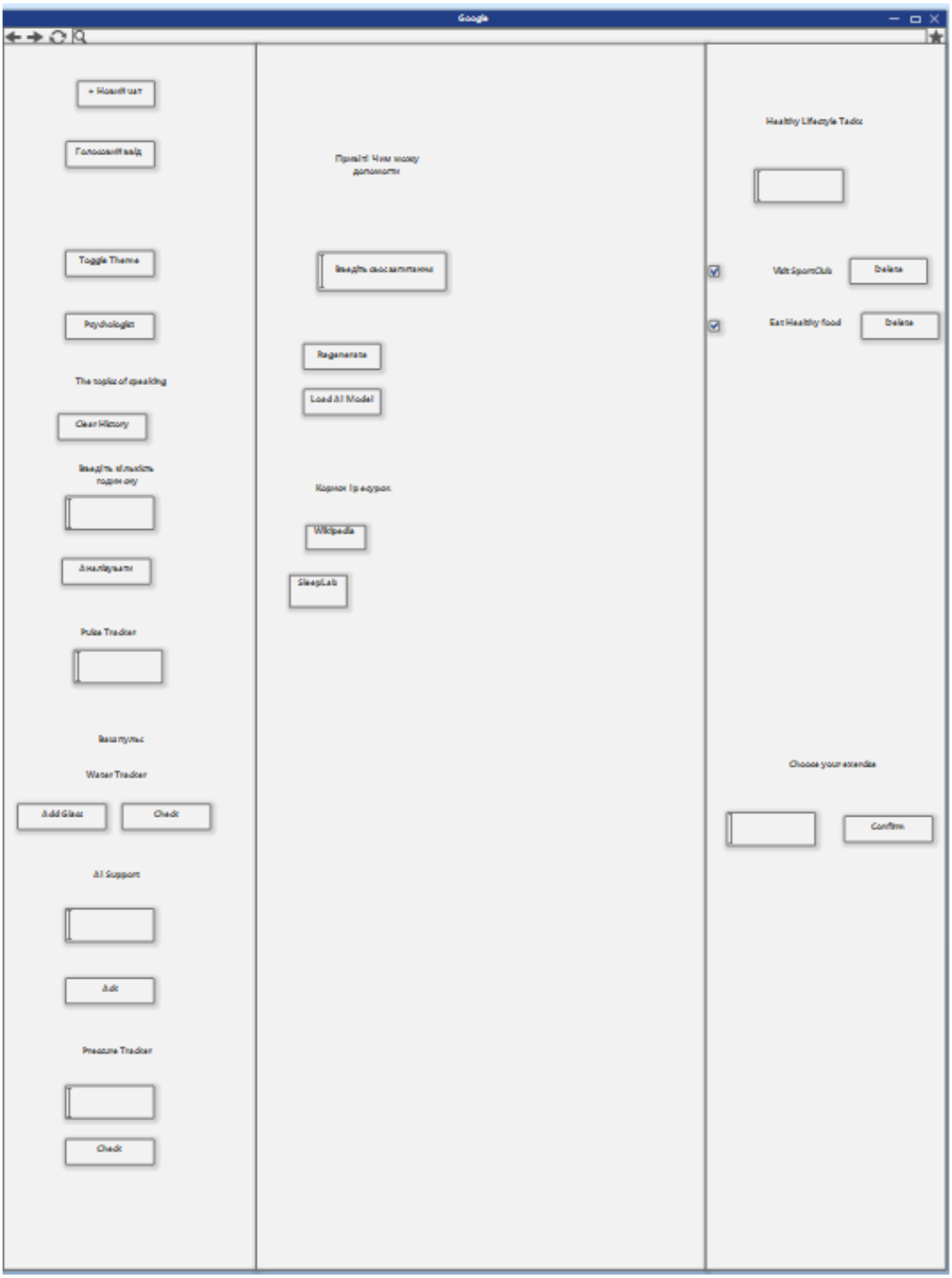

Рисунок 3.12 – Третій мокап

Даний мокап є унікальним в плані розміщення чату для користувача і списку завдань. Але вигідним є обрання першого мокапу.

Висновки до розділу 3

В даному розділі розібрано питання стосовно архітектури, моделювання й проєктування програмної системи проєкту. Розглянуто такі типи архітектур, як: клієнт-серверна, локальна, мультимодальна й гібридна. Розглянуто переваги й недоліки по кожній з них. На основі теоретичної інформації, зроблено відповідні висновки й обрано клієнт-серверну архітектуру.

Створено відповідні діаграми для моделювання ПЗ. Розглянуто такі діаграми, як: діаграма прецендентів, діаграма пакетів, діаграма компонентів, , діаграми взаємодії, діаграми класів, діаграми станів та переходів й діаграми діяльності. Це зроблено для розуміння способу реалізації програмного продукту й логіки її функціоналу. Виявлено той факт, що даних діаграм слід дотримуватись для правильної реалізації програмного продукту. Це зроблено для усунення подальших проблем стосовно реалізації проєкту. Також, виявлено той факт, що дані діаграми дають розробникам розуміти те, як саме реалізувати програмну систему та її функціонал.

Розглянуто теоретичні відомості стосовно програмних забезпечень для створення відповідних діаграм. Виділено такі теоретичні аспекти, як: історія створення й розвитку ПЗ, переваги й недоліки. На основі аналізу відповідних теоретичних відомостей, виконано вибір відповідних програмних засобів.

Розглянуто теоретичну інформацію стосовно архітектур для роботи самого чатбота. Виділено наступні аспекти кожної з роглянутих архітектур: історію створення, переваги й недоліки.

Реалізовано мокап для графічного інтерфейсу проєкту. Графічний інтерфейс виконано відповідно першому створенному мокапу.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЧАТБОТУ

В сьогоденні, сучасні програмні системи вимагають реалізацію сучасного функціоналу для зручності усіх користувачів. Кожна функція зручності є плюсом як для користувача, так й для системи. Користувач отримує користь, а система популярність серед інших користувачів. Сьогодні, важливо враховувати користувачів усіх груп населення.

4.1 Модульне тестування

Модульне тестування – тестування, що дозволяє виявити проблеми й помилки на ранній стадії розробки ПЗ. Перевагами даного тестування є такі речі, як: можливість раннього виявлення помилок, економія ресурсів, стимуляція написання чистого й структурованого коду, опис очікуваної поведінки функцій й можливість перевірки протестованих частин. Недоліками даного тестування є наступні речі: велика витрата зусиль, тісний зв’язок з реалізацією, можливість уповільнення роботи, можливість залишення критичних помилок на рівні користувача. Даний вид тесту має наступні інструменти тестування: describe, test, render, screen.

Describe – інструмент для проведення тестування. Його ідея полягає у групуванні тестів у логічні набори. Test – інструмент для опису конкретного тестування. Його ідея полягає у перевірці роботи певного модуля. Render – інструмент для відображення процесу тестування. Його ідея полягає в інформуванні користувача про проходження тестування. Screen – інструмент для доступу до вже відрендерених об’єктів. Його ідея полягає у наданні доступу до відповідних об’єктів. На рис. 4.1 продемонстровано код компонента «csrfDefend». Повна логіка представлена нижче (додаток А).

```
const csrfDefend = () =>
{

const [csrfToken, setCsrfToken] = useState("");
useEffect(() =>
{
  fetch("http://localhost:5173/csrf-token",
}, []);
```

Рисунок 4.1 – Компонент CSRF-атак

Даний компонент запобігає CSRF-атакам. Тому, перед початком розробки системи, варто враховувати можливість CSRF-атак. На рис. 4.3 представлено клас AiSafetyGuard.

```
class AiSafetyGuard {

  // Основна перевірка
  isSafe(text) {
    if (!text) return { safe: true, reason: null };

    const lower = text.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "");

    // 1. Перевірка заборонених слів
    for (const word of [...this.blockedWordsUA, ...this.blockedWordsRU]) {
      if (lower.includes(word)) {
        return { safe: false, reason: `Заборонене слово: "${word}"` };
      }
    }

    // 2. Перевірка патернів
    for (const pattern of this.blockedPatterns) {
      if (pattern.test(lower)) {
        return { safe: false, reason: `Заборонена тема: ${pattern.source}` };
      }
    }
```

Рисунок 4.2 – Клас AiSafetyGuard

Для проведення даного тестування необов’язково реалізовувати схожі алгоритми. На рис. 4.4 продемонстровано код компонента «BruteForce».

```
import React, { useState } from "react";

const BruteForce = () => {
  const [targetPassword, setTargetPassword] = useState("1234");
  const [attempt, setAttempt] = useState("");
  const [isFound, setIsFound] = useState(false);
  const [tries, setTries] = useState(0);
  const [isRunning, setIsRunning] = useState(false);

  const startBruteForce = async () => {
    if (isRunning) return;
    setIsRunning(true);
    setIsFound(false);
    setTries(0);
```

Рисунок 4.3 – Компонент BruteForce

Варто враховувати те, що компоненти безпеки також треба тестувати. Адже, від цього залежить рівень безпеки користувача. Далі, проведено модульне тестування відповідних компонентів й класів. На рис. 4.5 продемонстровано код модульного тесту компонента для запобігання CSRF-атак.

```
import {screen, render} from "@testing-library/react";
import csrfDefend from "./CSRF/CSRF";

describe("CSRF component", () => {
    test("renders CSRF component", () => {
      render(<csrfDefend />);

      // приклад: шукаємо елемент за роллю
      const element = screen.getByRole("heading", { name: /csrf/i });

      expect(element).toBeInTheDocument();
    });
  });
```

Рисунок 4.4 – Модульне тестування компонента для запобігання CSRF-атак

Дане тестування проведено успішно. На рис. 4.6 продемонстровано модульне тестування класа «Censore».

```
import { render, screen } from "@testing-library/react";
import AiSafetyGuard from "./Censore";

describe("Censore component", () => {
  test("renders Censore component", () => {
    render(<AiSafetyGuard />);

    // приклад: шукаємо елемент за роллю
    const element = screen.getByRole("heading", { name: /Censore/i });

    expect(element).toBeInTheDocument();
  });
});
```

Рисунок 4.5 – Модульне тестування класа «Censore»

Дане тестування також відбулось успішно. На рис. 4.7 продемонстровано модульне тестування компонента «BruteForce».

```
import {render, screen} from "@testing-laibrary/react";
import BruteForce from "./BruteForce";
describe("Brute force", () =>
{
    test(
        "render brute force", () =>{
            render(<BruteForce />);

            const bruteForce = screen.getByRole();
            expect(bruteForce).toBeInTheDocument();
        }
    )
});
```

Рисунок 4.6 – Модульне тестування компонента «BruteForce»

4.2 Валідація облікових даних користувачів

Валідація облікових даних користувачів є одним з важливих кроків для реалізації безпечної роботи системи. Таким чином, користувачі не зможуть вводити можливу недостовірну інформацію в плані повноти. За допомогою валідації також можна не допустити облікових записів не справжніх користувачів на сервер. Перевагами даної методології забезпечення безпеки є такі речі, як: не надання несанкціонованого доступу, забезпечення чистоти сервера від підозрілої активності й стабільність роботи системи. Недоліками вважаються наступні речі: можливість витоку даних, фальсифікація даних й затриманні спроби верифікування. На рис. 4.7 продемонстровано проведення валідації для адміністратора системи.

```
// create a validEmail function

const validEmail = () =>
{
  const expressionEmail = `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`
  if(!useremail.includes(expressionEmail))
  {
    setUserEmail("The email isn`t valid. Try again");
  }else{
    setUserEmail("The email is valid.");
  }

}

const validPassword = () =>
{
  const exPassword = `^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$`
  if(!userpassword.includes(exPassword))
  {
    setUserPassword("The password isn`t valid. Try again");
  }else{
    etUserPassword("The password is valid");
  }

}
```

Рисунок 4.7 – Проведення валідації для адміністратора системи

Даний функціонал є надійним в плані інформування кожного користувача про невалідність тих даних, що введено ним. На рис. 4.8 продемонстровано валідацію даних для простих користувачів.

```
const validEmail = () =>
{
  const expressionEmail = `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`
  if(!useremail.includes(expressionEmail))
  {
    setUserEmail("The email isn`t valid. Try again");
  }else{
    setUserEmail("The email is valid.");
  }

}

const validPassword = () =>
{
  const exPassword = `^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$`
  if(!userpassword.includes(exPassword))
  {
    setUserPassword("The password isn`t valid. Try again");
  }else{
    setUserPassword("The password is valid");
  }

}
```

Рисунок 4.8 – Проведення валідації для простих користувачів системи

Виходячи з вищенаведеного, можна зробити висновок про те, що валідація облікових даних кожного з користувачів є важливим етапом реалізації безпечного середовища системи. Адже, це дає можливість блокування несанкційного доступу до системи.

4.3 Метричний аналіз якості програмного забезпечення

Спершу, проєкт оцінено за метриками loc- і fp-метрик. Для цього обчислено наступні 5 відповідних змінних (табл. 4.1) [17].

Таблиця 4.1 – Розрахунок 5 змінних

Тип елемента	Низька	Середня	Висока
EI	x3	x4	x6
EO	x4	x5	x7
EQ	x3	x4	x6
ILF	x7	x10	x15
ELF	x5	x7	x10

Також, надано відповідні значення для даних змінних. Нижче наведено відповідні значення (табл. 4.2) [17].

Таблиця 4.2 – Надання значення 5 змінним

Тип елемента	Низька	Середня	Висока
EI	3(x10)	0(x0)	0(x0)
EO	0(x0)	5(x5)	0(x0)
EQ	3(x4)	0(x0)	0(x0)

Продовження таблиці 4.2

ILF	0(x0)	0(x0)	15(x2) = 30
ELF	0(x0)	7 (x3) = 21	0(x0)

Після того, як відбулось оцінювання всіх елементів даної системи та отримання балів згідно їх складності, UFP має розраховуватись за їхньою сумою. Розрахунок VAF відбувається за таким алгоритмом, як: проведення аналізу системи за шкалою від 0 балів (немає впливу) до 5 балів (сильний вплив): F1 Data Communications: 3 (немає впливу), F2 Distributed Data : 4 (немає впливу), F3 Performance: 5, F4 Heavily Used Configuration: 5 (немає впливу), F5 Transaction Rate: 5 (дуже висока), F6 Online Data Entry: 5 (немає впливу), F7 End-User Efficiency: 5 (немає впливу), F8 Online Update: 1 (немає впливу), F9 Complex Processing: 5(немає впливу), F10 Reusability: 5(немає впливу), F11 Installation Ease: 5 (немає впливу), F12 Operational Ease: 5(немає впливу), F13 Multiple Sites: 5(немає впливу), F14 Facilitate Change: 5(немає впливу).

Розрахунок S відбувається за наступною формулою:

$$S = \frac{numbers_{code}*FP+methods}{classes}, \quad (4.1)$$

де $numbers_{code}$ – кількість рядків коду;

FP – кількість функціональних точок;

Methods – кількість методів;

Classes – кількість класів.

$S=\frac{1473*4+2000}{6} = \quad 6225$

Розрахунок UFP відбувається за наступною формулою:

$$UFP = \sum(ILF + EIF + EI + EO + EQ) \quad (4.2)$$

де ILF, EIF, EI, EO, EQ – типи елементів.

UFP = 30 + 55 + 12 + 30 + 21 = 118

Розрахунок VAF відбувається за наступною формулою:

$$VAF = 0.65 + (0.01 \cdot \sum i) \quad (4.3)$$

де $\sum i$ – сума оцінок відповідних показників.

Отже: VAF = 0.65 + (0.01 * 68) = 1.33

Розрахунок AFP відбувається за наступною формулою:

$$\text{AFP} = \text{UFP} * \text{VAF} \quad (4.4)$$

де UFP – сума функціональних точок,

VAF – коефіцієнт, враховуючий 14 факторів.

Отже: AFP = 118 * 1,33 = 156.94

Далі, розраховано оцінку дипломного проєкту з використанням моделі COCOMO II. Визначено PM та TDEV. Дана оцінка дає розуміння того, наскільки програмна система є чутливою й що треба покращити для можливої зміни рівня чутливості. Виявлено 5 факторів обчислення показника E (табл. 4.3).

Таблиця 4.3 – 5 факторів обчислення показника E

Змінна	Назва	Що означає
PREC	Precedentedness	Наскільки проєкт новий для команди. 0 балів – команда робила схожі проєкти сотні разів. 5 балів – абсолютно нова незвідана зона проєкту.
FLEX	Dev. Flexibility	Наскільки суворі вимоги. 0 балів – будь-який крок від планів суворо карається. 5 балів – команда вирішує сама, що їй робити.

Продовження таблиці 4.3

RESL	Architecture/Risk	Ступінь аналізу ризиків. 0 балів – команда знає всі ризики. 5 балів – аналіз ще не був проведений взагалі.
TEAM	Team Cohesion	Згуртованість команди. 0 балів – всі працюють як один механізм. 5 балів – команда постійно конфліктує.
PMAT	Process Maturity	Рівень зрілості процесів. 0 балів – високий рівень організації. 5 балів – повний хаос.

Також виявлено 17 множників EM (табл. 4.4). Дані множники також застосовуються в розрахунку оцінки проєкту.

Таблиця 4.4 – 17 множників EM

Множник	Що означає	Значення
RELY	Наскільки критично, якщо в продукті є помилки	Low (0.82). Невеликі незручності. Nominal (1.00). Помірні збитки. High (1.10). Великі фінансові втрати. Very High (1.26). Загроза життю
DATA	Співвідношення розміру БД до коду. Якщо у вас гігабайти даних на пару рядків коду – цей коефіцієнт зростає.	Від 0.90 (багато коду, мало БД) до 1.28 (багато БД, мало коду)
CPLX	Наскільки комплексний код і продукт.	Від 0.73 (дуже простий проєкт) до 1.66 (дуже складний)

Продовження таблиці 4.4

RUSE	Чи можна перевикористати код в інших проєктах.	Від 1.07 (лише для цього проєкту) до 1.56 (універсальний код)
DOCU	Наскільки детальна документація	Від 0.81 (невелика) до 1.23 (велика з певними стандартами)
TIME	Наскільки хороша оптимізація	Від 1 (працює лише на сильному залізі) до 1.63 (працює і на слабкому залізі)
STOR	Обмеження оперативної пам’яті	Від 1 (пам’яті достатньо) до 1.46 (пам’яті мало)
PVOL	Як часто змінюється середовище розробки	Від 0.87 (все стабільно) до 1.30 (змінюється кожен місяць)
ACAP	Здатність проєктувати систему	Від 0.71 (чудові навички проєктування) до 1.42 (жахливі навички)
PCAP	Навички програмування	Від 0.76 (швидке програмування без помилок) до 1.34 (повільне програмування з помилками)
PCON	Стабільність команди	Від 0.81 (постійна команда) до 1.29 (члени команди змінюються кожен місяць)
APEX	Досвід у галузі проєкту	Від 0.81 (ідеальні знання галузі) до 1.22 (вперше стикаюсь)
PLEX	Досвід з платформою на яку розробляється проєкт	Від 0.85 (ідеальні знання) до 1.19 (знання погані)
LTEX	Досвід з мовою програмування ПЗ	Від 0.84 (ідеальне знання мови програмування) до 1.20 (жахливі знання)

Продовження таблиці 4.4

TOOL	Інструментарій	Від 0.78 (використання сучасних IDE та інших програм) до 1.17 (використання старого софту)
SITE	Локація команди	Від 0.86 (команда знаходиться в одному приміщенні та має ідеальну комунікацію) до 1.22 (команда розкидана по різним країнам з поганим зв'язком)
SCED	Стислість графіку	Від 1.00 (звичайний графік) до 1.43 (скорочені строки)

Розрахунок метрики E відбувається за наступною формулою (див. додаток А):

$$E = 0.91 + 0.01 \times (PREC + FLEX + RESL + TEAM + PMAT), \quad (4.5)$$

де $PREC, FLEX, RESL, TEAM, PMAT$ – фактори обчислення показника E. Отже:

E = 0.91 + 0.01 * (3 + 0 + 3 + 0 + 3) = 1.0

Розрахунок EM_values відбувається за наступною формулою (див. додаток А):

$$EM\_values = \sum_{n=0}^{i} EM \times 17, \text{ де:} \quad (4.6)$$

де $\sum_{n=0}^{i} EM$ – сума усіх показників EM.

EM_values = (1.00 × 0.90×1.66×1.07×1.23×1×1.46 ×0.87 ×0.91 × 0.90×0.8×1.00×1.00×0.78 ×0.86 ×1.00) = 1.0976870425710203

Розрахунок EM_product відбувається за наступною формулою (див. додаток А):

$$EM\_product = math.prod(EM\_values) \quad (4.7)$$

де math.prod – добуток усіх елементів**;**

EM_values – набір значень EM.

EM_product = math.prod(EM_values) = 1.111408130603158

Розрахунок PM відбувається за наступною формулою:

$$\text{PM} = \text{A} \times (\text{Size})^{\text{E}} \times \prod_{\text{i}=1}^{17} \text{EM}_{\text{i}}, \text{ де:} \quad (4.8)$$

де A – базовий коефіцієнт (константа);

Size – розмір проєкту у тисячах рядків коду (KSLOC) або функціональних точках;

E – експонента масштабу;

$\prod_{\text{i}=1}^{17} \text{EM}_{\text{i}}$ – добуток множників EM.

Отже: $\text{PM} = A \times (Size)^{E} \times \prod_{i=1}^{17} EM_i = 2.94 * pow(1.473, E) * EM\_values = 4.813086278552649$

Розрахунок F відбувається за наступною формулою:

$$\text{F} = 0.28 + 0.2 \times (\text{E} - 0.91), \text{ де:} \quad (4.9)$$

де E – експонента масштабу.

$F = 0.28 + 0.2 \times (E - 0.91)$ = 0.29800000000000004

Розрахунок TDEV відбувається за наступною формулою:

$$\text{TDEV} = (\text{C} \times (\text{PM})^{\text{F}} \times \frac{\text{SCED\%}}{100}) \quad (4.10)$$

де C – константа;

PM – трудовитрати людина-місць.

$TDEV = (C \times (PM)^{F} \times \frac{SCED\%}{100})$ = (3.67 * pow(PM, F) * (100/100)) = 1.0683217974247137

Далі, проєкт оцінено за іншими метриками. Проаналізовано клас «History».

Клас «History» має 3 методи:

1. saveHistory();
2. deleteHistory();
3. clearHistory().

Метрика CS обрахована за наступною формулою :

$$\mathrm{CS} = \mathrm{M_{total}} + \mathrm{P_{total}} \quad (4.11)$$

де M_{total} – загальна кількість усіх методів класу;

P_{total} – загальна кількість усіх властивостей класу.

Клас «History» наслідується від класу «User». В свою чергу, клас «User» наслідується від базового класу «Object». Отже: CS = 3. Метрику NOO обчислено наступним чином:

$$\mathrm{NOO} = \mathrm{n1}+\mathrm{n2} \quad (4.12)$$

де n1 і n2 – перевизначені операції.

Клас «History» має 1 перевизначену операцію.

Звідси: NOO = 1.

Метрику NOA обчислено наступним чином:

$$\mathrm{NOA} = \sum \text{attributes} \quad (4.13)$$

де ∑ attributes – сума атрибутів.

У класа «History» є 1 підклас, що дає 3 методи. Це клас «UserHistory». Звідси: NOA = 1. Клас «History» пов’язаний з класами «UserHistory», «User», «Object» шляхом наслідування. Тому, CBO = 3. Метрику SI обчислено наступним чином:

$$\mathrm{SI} = \frac{\mathrm{NOO} \times \text{рівень}}{\mathrm{M_{заг}}} \quad (4.14)$$

де NOO – кількість перевизначених методів класу;

Рівень – рівень класу в ієрархії;

$M_{заг}$ – загальна кількість методів класу (включно з унаслідуваними батьківськими методами). Клас «History» має 0.1 спеціалізованого індексу. Звідси, SI = 0.1 * 0.3 / 0.3 = 0.1. Метрику OSAVG обчислено наступним чином:

$$\text{OSAVG} = \frac{\sum \text{LOC}\ (\text{method}_\text{i})}{\text{N}} \quad (4.15)$$

де $\sum LOC\ (method_i)$ – сума кількості рядків коду усіх методів класу;
N – загальна кількість методів класу.

Методи класу «History» мають невелику кількість рядків коду. Звідси, OSAVG = 4 / 4 = 1. Метрику ОС обчислено наступним чином:Складність окремих методів класу «History» є середньою (табл. 4.5).

Таблиця 4.5 – Параметри окремих методів й їхні ваги

Параметр	Вага
Виклик API	2.0
Присвоєння значення	1.0
Арифметична операція	1.0
Використання іншого методу з параметрами	5.0
Використання іншого методу без параметрів	5.0
Прості виклики (якщо при використанні іншого методу присутня складна логіка)	6.0
Вкладені вирази (if, for тощо)	1.0
Параметри методу	1.0
Тимчасові змінні	0

Звідси, ОС = 22.

Метрику NPAVG обчислено наступним чином:

$$NPAVG = \frac{\sum \text{параметрів}}{\sum \text{методів}} \quad (4.16)$$

де $\sum$ параметрів – кількість всіх параметрів класу;

$\sum$ методів – кількість всіх методів класу.

Середня кількість параметрів на метод класу «History» є невеликою.

Звідси, NPAVG = 2/2 = 1.

Метрику NSUB обчислено наступним чином:

$$\text{NSUB} = \sum_{k=0}^{n} \text{subclasses} \quad (4.17)$$

де $\sum_{k=0}^{n}$ subclasses – кількість підкласів.

Кількість підкласів системи дорівнює 2 підкласам. Це підкласи «UserHistory» і «Request». Звідси, NSUB = 2. Отримано наступні результати (табл. 4.6).

Таблиця 4.6 – Результати обчислень

Клас	E	CS	NOO	N0A	SI	OSAVG	OC	NPAVG	NSUB
History	1	3	1	1	0.1	1	22	1	2

Згідно з результатами, код проєкту рефакторингу не вимагає, оскільки значення метрик знаходяться в нормальних значеннях. Тому, проєкт процесу рефакторингу коду не вимагає.

4.4 Керівництво користувача

Графічний інтерфейс чатбота виконано з дотриманням структури створеного мокапу. Завдяки відповідному дизайну, користувач може легко орієнтуватись в програмній системі. Тому, при розробці графічного дизайну вебзастосунку, варто враховувати усі особливості користувача. На рис. 4.10 продемонстровано першу частину графічного інтерфейсу.

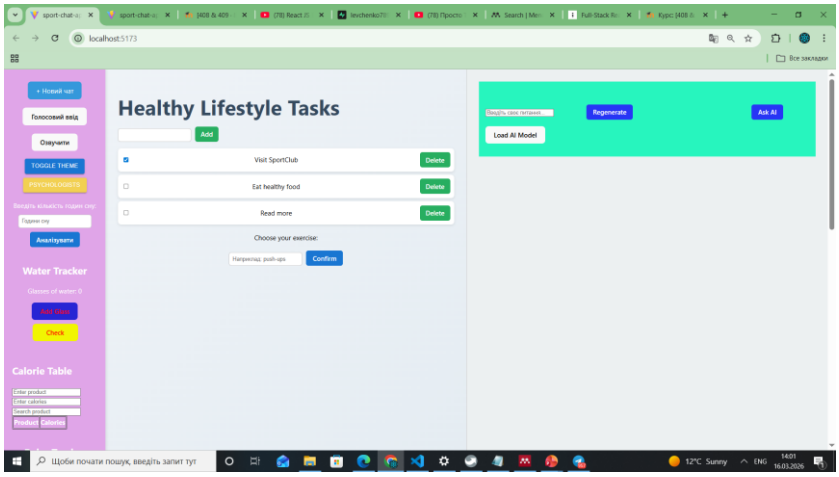

Рисунок 4.10 – Головна сторінка

Дана частина демонструє те, що користувач може виконувати такі речі, як: відправка запиту текстом або голосом, вибір фізичних вправ, трекер води, аналізатор якості сну, можливість підрахунку спожитих калорій, створення нового чату, перемикання тем вебзастосунку ймовірність отримання психологічної підтримки. На рис. 4.11 продемонстровано другу частину графічного інтерфейсу.

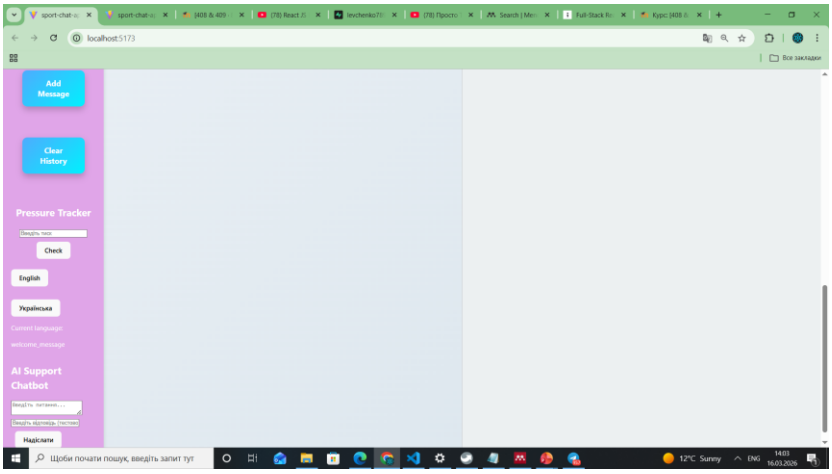

Рисунок 4.11 – Продовження головної сторінки

Дана частина графічного інтерфейсу демонструє наступні функціональні можливості: трекер пульсу, збереження повідомлень, переклад українською й англійською мовами й звернення до технічної підтримки. Даний чатбот адаптовано до вирішення сучасних проблем.

Висновки до розділу 4

В даному розділі розглянуто теоретичні відомості з WebSpeechAPI, модульного тестування, валідації облікових даних користувачів, розрахунків для проєкту й створення керівництва користувача. Розібрано окремі методи й приклади застосування відповідних технологій до даного проєкту. Розібрано відповідні метрики для проведення розрахунків відповідних оцінок проєкту.

Розібрано призначення кожної з метрик. Наприклад, призначення метрики ОС полягає у її використанні для оцінки стану реалізованого проєкту відповідно запланованого.

Реалізовано Frontend і Backend-частини чатботу. Обрано сучасні технології реалізації функціоналу на обох частинах.

Виділені наступні переваги модульного тестування: раннє виявлення помилок, можливість економії ресурсів, написання чистого й структурованого коду, можливості опису очікуваної поведінки функцій й можливість перевірки протестованих частин. Виділені наступні недоліки модульного тестування: можливість великої витрати зусиль, можливий тісний зв’язок з реалізацією, уповільнена робота, ризик залишення критичних помилок на рівні користувача.

Виділено наступні переваги валідації даних: неможливість надання несанкціонованого доступу, можливість забезпечення чистоти сервера від підозрілої активності й стабільність роботи системи. Виділено наступні недоліки валідації даних: витік даних, можливість фальсифікації й підробки даних й затримка верифікації.

ВИСНОВКИ

В межах КБР розроблено чатбот на основі технології штучного інтелекту для підтримки здорового способу життя. Наразі, сфера охорони здоров’я не може залишатись без постійного розвитку як інформаційних, так й роботизованих технологій. Адже, постійний розвиток є важливим для збереження життя й здоров’я усього людства. Для цього потрібен розвиток чат-ботів, що будуть підтримувати введення здорового способу життя певними людьми. Для того, щоб чат-боти спілкувались з людьми природньою мовою, треба застосовувати інструменти генеративного штучного інтелекту. Наприклад, це можуть бути інструменти для генерації випадкових відповідей користувачу.

Досліджено актуальний стан предметної області для програмного рішення, актуальність його створення, переваги й недоліки інших систем, інструментарій для реалізації й специфікацію вимог до системи. Проведено вибір технологій реалізації програмного продукту. Серед них проведено вибір відповідних бібліотек. Це надає можливість для реалізації як для клієнтської, так й для серверної частин вебзастосунку. Важливо враховувати той факт, що користувач може обирати моделі для спілкування. Для реалізації чат-боту була обрана архітектура «gpt-4o-mini». Реалізовано відповідні тести й сучасний функціонал.

Спроєктовано та реалізовано чатбот для підтримки здорового способу життя. Для виконання проєктування, застосовано сучасні програмні забезпечення. Застосовано мову моделювання «UML» в редакторі діаграм «StarUML». Спроєктовано та реалізовано інформаційне забезпечення чатботу. Серед розглянутих вибрано такий інструментарій, як: React.js, CryptoJS, I18next, Meyda, DOMPurify,Node.js, react-router-dom, nlp, openAI, generateText й fuse.. Проведено модульне тестування відповідних компонентів. Наведено керівництво користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Карчевська О. В. ChatGPT, розум та чудовиська. Академічна доброчесність, відкрита наука та штучний інтелект: як створити д оброчесне освітнє середовище: тези доповідей міжнар. наук.-
-практ. конф. Львів: Liha-Pres, 2023. С. 81.
DOI: http://dx.doi.org/10.36059/978-966-397-345-6-81 (дата звернення: 25.05.2026);

2) М. Антонюк: Microsoft Copilot як цифровий ресурс у системі навчання інформаційно-комунікаційним технологіям під час навчання : Вебсайт: URL: https://pse.itup.com.ua/index.php/pse/article/view/148 (дата звернення: 25.04.2025);

3) Н. С. Конопля: UI/UX дизайн мобільно додатку для підтримки та спостереженням за станом здоров'я людини: Вебсайт: URL: https://surl.li/hicqfr (дата звернення: 30.04.2026);

4) Луценко Я. В. – Проєктування дизайну інтерфейсу мобільного застосунку для відстеження харчування вегетаріанців: Вебсайт:URL: https://openarchive.nure.ua/entities/publication/0d667dbc-11e7-47d7-8734-237cf20b7b1e (дата звернення: 30.04.2026);

5) Sarthak Verma: Express.Js and its Usage in Web Development.2024. International Journal of Research Publication and Reviews. Vol. 5 P. 2582-7421. DOI: https://www.ijrpr.com/archive.php?volume=5&issue=4 (accessed: 30.04.2026);

6) Ningxian Zhu Security of CORS on LocalStorage: Proc. of the 2021 IEEE Int. Conf. Of Internet, Suzhou, China, Apr. 16–18, 2021 / School of Cyberspace Security Shandong University of Political Science and Law Jinan, China (accessed: 15/05/2026);

7) Daniel Amoah, Samuel Sambasivam: A Comparison of Internationalization and Localization Solutions for Web and Mobile

Applications.2022. Journal of Information Systems Applied Research. Vol. 15 P. 1946-1836. URL: https://www.jisara.org/2022-15/n2/JISARv15n2.pdf#page=39 (accessed: 16.0.2026)

8) Rakesh Reddy Peddamallu: Web Development in Advanced Threat Prevention. Vol. 9 P. 2395-0072. URL: https://surl.li/xckpdi (accessed: 16.05.2026);

9) Adam Freeman: Forms and Validation.2019. Apress. Vol. 10 P. 1007 (978-1-4842-4451-7).DOI: https://link.springer.com/chapter/10.1007/978-1-4842-4451-7_15 (accessed: 22.05.2026);

10) Prof Nidhi Upadhyay, Prof Nidhi Singh: Micro Frontends in React.js: A Comprehensive Analysis and Implementation Guide.2024. International Journal of Scientific Research in Engineering and Management.Vol.8. P.12.DOI: https://doi.org/10.55041/IJSREM39990 (accessed: 22.05.2026);

11) Zhongcheng Lei, Hong Zhou, Wenshan Hu, Guo-Ping Liu: Toward an international platform: A web-based multi-language system for remote and virtual laboratories using react framework.2022.Heliyon.Vol.8. P.10. DOI: http://dx.doi.org/10.1016/j.heliyon.2022.e10780(accessed: 25.05.2026);

12) Swetha Talakola: Automated end to end testing with Playwright for React applications.2024. International Journal of Emerging Research in Engineering and Technology.Vol.5.P.1. DOI: https://surl.li/jrfzfc (accessed: 25.05.2026);

13) Luigi Lavazza, Angela Locoro, Roberto Meli: Software Development and Maintenance Effort Estimation Using Function Points and Simpler Functional Measures.2024. Software.Vol.3.P.4. DOI: https://surl.li/lpyjvs (accessed: 25.05.2026).

14) MyFitnessPal: Calorie Tracker & BMR Calculator to Reach: Вебсайт.URL: https://www.myfitnesspal.com (accessed: 25.05.2026);

15) Healthy Weight Loss & Eating: Lose Weight Fast with Yazio.Вебсайт.URL: https://www.yazio.com/ (accessed: 29.05.2026);

16) Досягай своїх цілей зі схуднення разом із fatsecret.Вебсайт.URL: https://www.fatsecret.com/ (дата звернення: 29.05.2026).

17) ISO/IEC 20926:2009. Software and systems engineering — Software measurement — IFPUG functional size measurement method. Geneva: ISO, 2009. URL: https://www.iso.org/obp/ui/en (accessed: 09.06.2026).

ДОДАТОК А

Код компонента csrfDefend

```
 import React, {useEffect, useState} from "react";
import csrfToken from "csrf";
// create a object of a csrf defend
const csrfDefend = () =>
{
const [csrfToken, setCsrfToken] = useState("");
useEffect(() =>
{
        fetch("http://localhost:5173/csrf-token", {credentials: "include"}).then(res => res.json()).then(data => setCsrfToken(data.csrfToken));
}, []);
const sendRequest = () =>
{
fetch("http://localhost:5000/support", {
   method: "POST",
   credentials: "include",
   headers: {
      'Content-Type' : 'application/json',
      "X-CSRF-Token" : csrfToken
   },
   body: JSON.stringify({ amount: 1000, to: "recipient" })
})
.then(res => res.json())
     .then(data => console.log(data));
};
```