Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«РОЗУМНИЙ АСИСТЕНТ ПСИХОЛОГА»

Виконав: студент IV курсу, групи 1ІСТ-21б
спеціальності 126 – Інформаційні системи
та технології

(шифр і назва спеціальності)

Владислав ШАРКОВСЬКИЙ
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри АІІТ
Марія БАРАБАН
(прізвище та ініціали)
«11» 06 2025 р.

Рецензент: д.т.н., професор кафедри КСУ
Марія ЮХИМЧУК
(прізвище та ініціали)
«11» 06 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
Олег БІСІКАЛО
«12» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 126 –Інформаційні системи та технології
Освітньо-професійна програма Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«12» 06 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шарковському Владиславу Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Розумний асистент психолога»
керівник роботи Барабан Марія Володимирівна, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи:
- використання сучасних інструментів розробки вебдодатків - .NET, React, MUI, TypeScript;
- інтеграція з моделями штучного інтелекту – OpenAI, Groq;
- реалізація модулів збереження історії взаємодій і управління сесіями клієнтів;
- інтерфейс взаємодії з психологом, з можливістю перегляду та аналізу запитів;
4. Зміст текстової частини (перелік питань, які потрібно розробити):
Провести огляд сучасних інтелектуальних систем підтримки у сфері психології, розробити архітектуру вебзастосунку для розумного асистента психолога, створити алгоритми взаємодії користувача з системою на основі моделей ШІ, реалізувати програмне забезпечення вебзастосунку, протестувати систему на предмет точності, зручності та ефективності, розглянути питання етики та конфіденційності при використанні ШІ в психології
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Структурна схема архітектури розумного асистента психолога; Діаграми UML (використання, послідовності, класів); Схема роботи користувача з системою; інтерфейс користувача (макети або скріншоти); Приклади відповіді системи в типових сценаріях

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
1-4	Барабан М.В., доц. каф. АПТ		

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Приміт
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	21.03.2025	22.03.2025	вик.
2	Аналіз предметної області та опрацювання літературних джерел.	23.03.2025	24.03.2025	вик.
3	Постановка задач для вирішення в БКР	01.04.2025	26.04.2025	вик.
4	Вирішення поставлених задач	29.04.2025	10.05.2025	вик.
5	Підтвердження достовірності отриманих результатів	13.05.2025	24.05.2025	вик.
7	Оформлення пояснювальної записки та графічної частини	27.05.2025	31.05.2025	вик.
8	Перевірка ПЗ на відповідність вимогам	03.06.2025	07.06.2025	вик.
9	Попередній захист, доопрацювання, та рецензування БКР	10.06.2025	15.06.2025	вик.
10	Захист БКР	16.06.2025	17.06.2025	вик.

Студент _____ Владислав ШАРКОВСЬК
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марія БАРАБАН
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 71 сторінок формату А4, включаючи додатки, містить рисунків 31, список використаних джерел із 21 найменувань.

У роботі розроблено вебзастосунок «Розумний асистент психолога», призначений для автоматизації рутинних завдань психологів і підвищення ефективності їхньої професійної діяльності. Розглянуто принципи створення клієнт-серверної архітектури, інтеграції мовних моделей штучного інтелекту (OpenAI, Groq) та реалізації векторного пошуку за допомогою PostgreSQL із розширенням PGVector. Зручність системи забезпечується інтуїтивним інтерфейсом, побудованим на React, TypeScript і Material UI, а також можливістю збереження історії взаємодій, створення опитувань і управління терапевтичними планами. Реалізоване програмне забезпечення дозволяє обробляти запити швидко й ефективно, відповідаючи вимогам безпеки та етики. Система є масштабованою та відкритою для подальшого вдосконалення, зокрема шляхом додавання модулів аналізу емоційного стану чи підтримки інших терапевтичних методик.

Ключові слова: розумний асистент, психологія, вебзастосунок, штучний інтелект, OpenAI, Groq, PostgreSQL, React, TypeScript, Material UI, векторний пошук, етика.

ANOTATION

The bachelor’s thesis consists of 71 A4 pages, including appendices, containing figures 31, and a reference list with 21 sources.

The work focuses on the development of the «Smart Psychologist Assistant» web application, designed to automate routine tasks and enhance the efficiency of psychologists’ professional activities. The principles of building a client-server architecture, integrating large language models (OpenAI, Groq), and implementing vector search using PostgreSQL with the PGVector extension are explored. The system’s usability is ensured by an intuitive interface built with React, TypeScript, and Material UI, as well as features for storing interaction history, creating surveys, and managing therapy plans. The implemented software enables fast and efficient request processing while meeting security and ethical requirements. The system is scalable and open to further improvements, such as adding modules for emotional state analysis or support for additional therapeutic methodologies.

Keywords: smart assistant, psychology, web application, artificial intelligence, OpenAI, Groq, PostgreSQL, React, TypeScript, Material UI, vector search, ethics.

ЗМІСТ

ВСТУП

У сучасному світі психологічне здоров’я набуває дедалі більшого значення. Швидкий темп життя, постійні стреси, інформаційне перевантаження та соціальні виклики спричиняють підвищений попит на кваліфіковану психологічну допомогу. Водночас можливості особистої консультації з психологом залишаються обмеженими через нестачу фахівців, часові або географічні обмеження, а також фінансові чинники. У цьому контексті на перший план виходять технологічні рішення, що можуть підтримати психологів у їхній професійній діяльності та забезпечити базовий рівень взаємодії з клієнтами у зручному форматі.

Штучний інтелект (ШІ) поступово інтегрується у сферу охорони психічного здоров’я. Розумні асистенти, які використовують сучасні моделі обробки природної мови, здатні аналізувати запити користувачів, пропонувати психологічно обґрунтовані відповіді, допомагати у формулюванні емоцій, нагадувати про техніки самодопомоги та забезпечувати первинний контакт перед спілкуванням із фахівцем. Подібні інструменти не замінюють психолога, але виступають як його помічники, здатні автоматизувати рутинні завдання, структурувати інформацію та покращити ефективність надання допомоги.

У межах цієї роботи передбачається створення вебзастосунку — розумного асистента психолога, що поєднує інтерфейс для фахівця з інтеграцією мовної моделі штучного інтелекту. Такий інструмент дозволяє психологу отримувати допомогу в аналізі звернень, формувати індивідуальні поради для клієнтів та покращувати якість ведення сеансів за рахунок збереження історії запитів, моделювання діалогів і підказок на основі контексту.

Сучасні вебтехнології дозволяють створювати зручні та безпечні інтерфейси, які відповідають вимогам конфіденційності та захисту персональних даних. Особлива увага в роботі приділяється дотриманню етичних норм, зокрема уникненню заміщення спеціаліста алгоритмом, а також забезпеченню користувачам зрозумілого і прозорого механізму взаємодії зі штучним інтелектом.

Основним науково-практичним результатом роботи є розробка програмного забезпечення для розумного асистента психолога, яке може бути інтегроване в робочі процеси фахівця, полегшуючи попередній аналіз запитів клієнтів, генерацію відповідей та рекомендацій, а також автоматизацію рутинної комунікації.

Метою роботи є покращення ефективності психологічної практики шляхом розробки вебзастосунку з інтегрованим інтелектуальним модулем, який забезпечує автоматизовану підтримку психологів у аналізі запитів клієнтів, управлінні сесіями та створенні терапевтичних планів.

Для досягнення мети необхідно вирішити такі задачі:

1. Провести аналіз предметної області та сучасних підходів до використання ШІ в психології.
2. Дослідити наявні рішення в галузі інтелектуальних помічників та оцінити їхню ефективність.
3. Розробити архітектуру вебзастосунку з урахуванням вимог безпеки та функціональності.
4. Інтегрувати мовну модель ШІ для підтримки обробки запитів користувачів.
5. Реалізувати інтерфейс користувача для психолога та забезпечити зручне керування взаємодіями.
6. Провести тестування системи на предмет коректності, ефективності та зручності.

Об’єктом дослідження є процес автоматизованої підтримки психолога за допомогою інтелектуальних систем.

Предметом дослідження є засоби та методи розробки вебзастосунків із вбудованими мовними моделями для використання у психологічній практиці.

Методи дослідження включають аналіз і синтез, методи проєктування та програмування, використання бібліотек для інтеграції моделей штучного інтелекту, методи тестування та UX-аналізу.

Науково-практичний результат полягає у створенні вебзастосунку, що забезпечує інтерактивну допомогу психологу, має зручний інтерфейс,

використовує штучний інтелект для генерації відповідей, зберігає історію запитів та відповідає сучасним вимогам конфіденційності.

Апробація результатів дослідження результати роботи було представлено на науково-практичній конференції “Молодь в науці: дослідження, проблеми, перспективи” [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ ПІДТРИМКИ ПСИХОЛОГІЧНОЇ ПРАКТИКИ

Розвиток інтелектуальних систем відкриває нові можливості для вдосконалення процесів психологічного консультування та підтримки. Перед початком проєктування та реалізації програмного продукту доцільно провести детальний аналіз предметної області. Це дозволяє зрозуміти специфіку існуючих підходів, визначити технічні та етичні обмеження, а також виявити актуальні потреби кінцевих користувачів — психологів і клієнтів.

У цьому розділі розглядаються основні напрями застосування ШІ у психології, здійснюється огляд існуючих рішень, а також формулюється задача дослідження, яка лежить в основі цієї дипломної роботи.

1.1 Дослідження актуальності застосування ШІ в психологічній допомозі

Психологічна допомога стає все більш затребуваною у сучасному суспільстві. В умовах зростання рівня стресу, невизначеності та соціальних викликів зростає потреба в доступних та ефективних інструментах психоемоційної підтримки. Проте традиційні форми психологічної роботи мають низку обмежень, серед яких: обмежена кількість фахівців, складність організації особистих зустрічей, суб'єктивність інтерпретацій, а також низька доступність у віддалених регіонах.

У цих умовах застосування штучного інтелекту (ШІ) у психології набуває все більшої актуальності. Інтелектуальні системи здатні автоматизувати рутинні процеси, аналізувати великі обсяги даних про стан користувача, забезпечувати попередню діагностику або адаптивну підтримку, а також виступати як платформа для самодопомоги. Подібні рішення можуть допомогти психологам зосередитися на найважливішому — безпосередньому спілкуванні з клієнтом — або навіть

забезпечити первинну психологічну підтримку людям, які з певних причин уникають традиційної терапії.

Завдяки розвитку мовних моделей, систем обробки природної мови (NLP) [2], аналізу тональності, розпізнавання емоцій та візуалізації поведінкових патернів, ШІ-помічники можуть не лише імітувати діалог, але й адаптувати відповіді до психологічного стану користувача. При правильному проєктуванні такі системи не замінюють психолога, а доповнюють його роботу — як аналітичний інструмент, модуль попередньої взаємодії або цифровий щоденник емоцій.

Водночас, широке впровадження ШІ в цій сфері потребує зваженого підходу, що враховує етичні, правові та конфіденційні аспекти. Це особливо важливо у сфері психології, де йдеться про чутливу особисту інформацію.

Таким чином, дослідження та впровадження розумного асистента для психологів на основі ШІ є актуальним напрямом, що відповідає потребам сучасного суспільства та відкриває нові можливості для надання якісної психологічної підтримки.

1.2 Аналіз систем-аналогів та сучасних розумних асистентів у сфері ментального здоров’я

Розробка інтелектуального асистента для психологічної підтримки базується на сучасних технологіях штучного інтелекту, які вже знайшли застосування у низці цифрових рішень. На ринку існує кілька сервісів, що використовують алгоритми обробки природної мови (NLP), машинного навчання та психологічних моделей для діалогу з користувачем у контексті ментального здоров’я.

Одним із найвідоміших прикладів є Woebot [3] — мобільний застосунок, створений за участю клінічних психологів і спеціалістів з ШІ (рис. 1.1). Woebot використовує когнітивно-поведінкову терапію (CBT) для ведення діалогів, підтримки користувача та відстеження його емоційного стану. Алгоритми

програми дозволяють реагувати на повідомлення користувача в напівструктурованій формі, надаючи мотиваційну або заспокійливу підтримку.

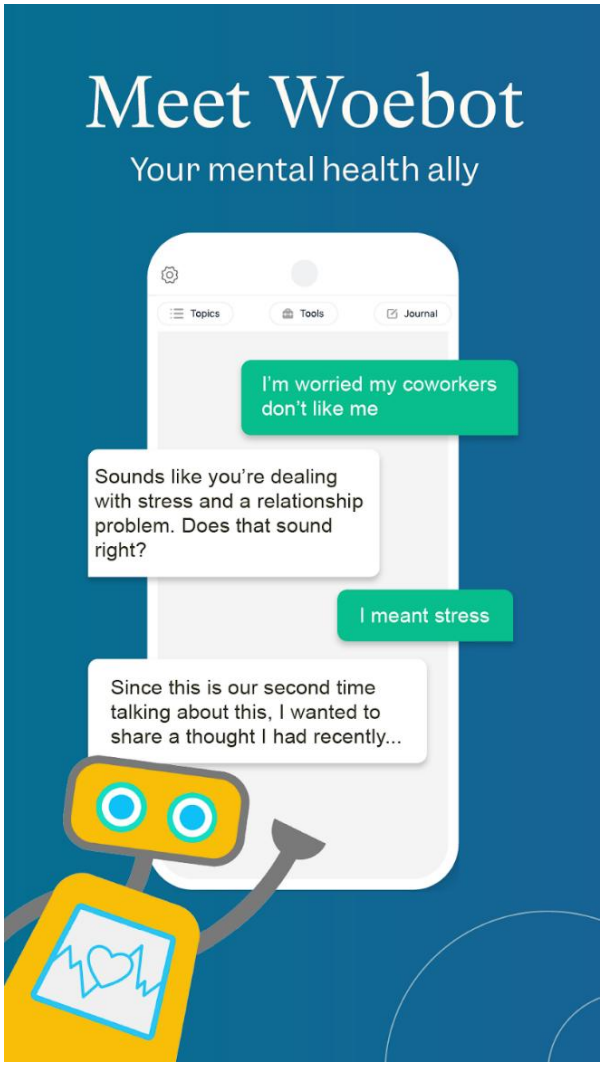

Рисунок 1.1 – Woebot інтерфейс чату

Іншим прикладом є Wysa [4] — цифровий ментальний помічник, що поєднує інтерфейс на основі штучного інтелекту з доступом до реального психолога (рис. 1.2). Цей сервіс дозволяє користувачам спілкуватися з ботом у текстовому режимі, проводити короткі ментальні вправи, вести щоденник настрою та отримувати підтримку при стресі, депресії чи тривожності.

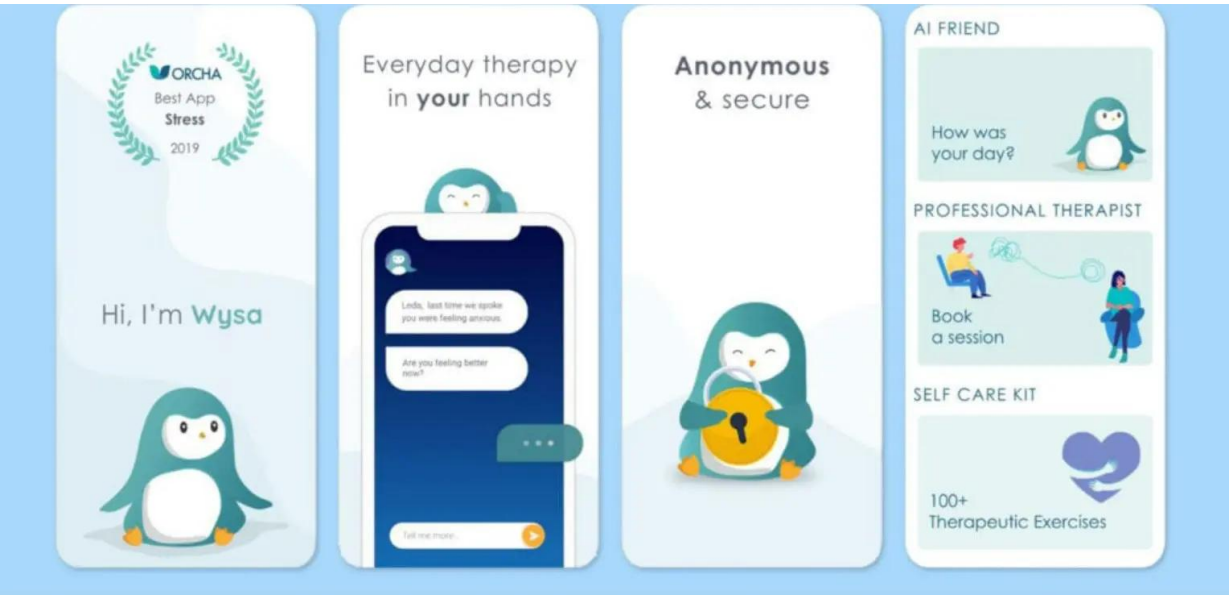

Рисунок 1.2 – Wysa інтерфейс

Також варто згадати Replika [5], яка хоч і не є спеціалізованою терапевтичною платформою, проте завоювала велику аудиторію як емоційний співрозмовник (рис. 1.3). Вона дозволяє формувати індивідуальний стиль спілкування, навчатися на базі попередніх діалогів та адаптуватися до вподобань користувача.

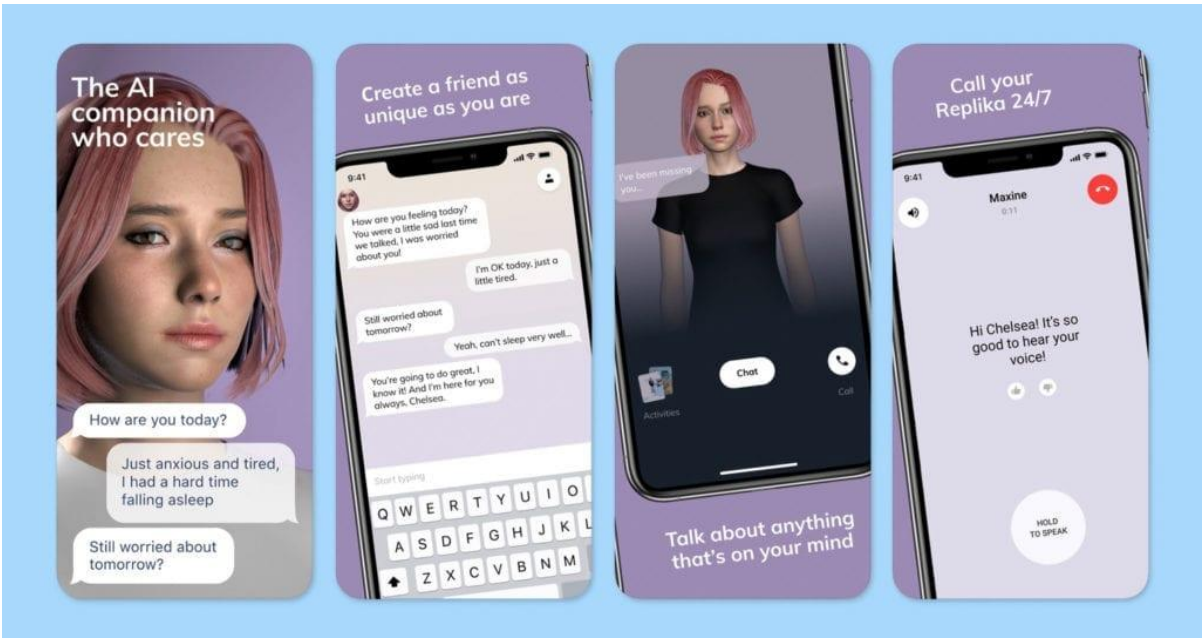

Рисунок 1.3 – Replika інтерфейс

Незважаючи на високу функціональність згаданих систем, вони мають низку обмежень:

- обмежена адаптація до конкретних методик психолога, з яким працює клієнт;
- загальні відповіді без урахування індивідуального контексту користувача;
- відсутність повноцінної інтеграції з робочим середовищем практикуючого спеціаліста (наприклад, планування прийомів, аналітика, збереження нотаток);
- відсутність гнучкості для використання в клінічній практиці, оскільки більшість сервісів орієнтовані на кінцевого користувача, а не на професійного психолога.

Можна зробити висновок що, потреба у створенні спеціалізованого інструменту для підтримки діяльності психолога залишається актуальною. Існуючі рішення частково охоплюють проблематику ментального здоров’я, однак не надають достатнього функціоналу для інтеграції у професійну практику.

1.3 Постановка задачі дослідження

Зважаючи на результати аналізу предметної області та існуючих рішень, можна зробити висновок, що більшість сучасних цифрових інструментів у сфері ментального здоров’я орієнтовані на кінцевого користувача — пацієнта. Натомість професійна діяльність психолога, яка включає комунікацію з клієнтами, ведення записів, планування сеансів, аналіз запитів та підтримку між зустрічами, часто не має ефективної автоматизованої підтримки. Це створює додаткове навантаження на спеціаліста та знижує ефективність його роботи.

Метою цього дослідження є розробка інтелектуального асистента, який дозволить психологу:

- автоматизувати рутинні комунікаційні завдання;
- ефективно взаємодіяти з клієнтами у перервах між особистими зустрічами;

- використовувати шаблони психотерапевтичних інтервенцій на основі ШІ;
- зберігати, аналізувати й узагальнювати діалоги для подальшої роботи;
- персоналізувати підтримку в межах узгоджених методик (наприклад, КПТ або АСТ).

Таким чином, задачі дослідження можна сформулювати так:

1. Провести аналіз вимог до програмного забезпечення в контексті роботи практичного психолога.
2. Проаналізувати можливості сучасних технологій обробки природної мови та моделей машинного навчання для реалізації розумного асистента.
3. Розробити архітектуру системи, яка дозволяє гнучко керувати діалогами, сценаріями взаємодії та зберіганням даних.
4. Реалізувати прототип програмного модуля асистента, здатного підтримувати діалог з клієнтом у напівавтоматичному режимі.
5. Забезпечити можливість адаптації та інтеграції модуля в робочі процеси психолога.
6. Провести тестування розробленого рішення на відповідність практичним задачам.

У результаті виконання цих задач має бути створене програмне рішення, здатне суттєво покращити ефективність роботи психолога, водночас зберігаючи етичні та конфіденційні принципи взаємодії з клієнтами.

1.4 Висновок до розділу

У першому розділі було здійснено аналіз предметної області, пов’язаної з розробкою інтелектуального асистента для психолога. Встановлено, що сучасні цифрові інструменти у сфері ментального здоров’я значною мірою орієнтовані на кінцевих користувачів — пацієнтів, у той час як потреби фахівців залишаються частково або повністю незадоволеними.

Було досліджено низку існуючих програмних рішень, таких як мобільні застосунки для психоемоційної самодопомоги, чат-боти для підтримки користувачів, а також системи для управління психотерапевтичною практикою. Аналіз показав, що ці сервіси рідко поєднують можливості ШІ для комунікації з клієнтами із функціями зберігання інформації, планування та індивідуалізації підходу до терапії.

У результаті сформульовано задачі дослідження, реалізація яких дозволить створити вебзастосунок із вбудованим інтелектуальним асистентом, призначеним для полегшення повсякденної роботи психолога. Цей інструмент має сприяти підвищенню якості та ефективності психотерапевтичної взаємодії, зберігаючи при цьому етичні норми та конфіденційність.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМУ РОЗУМНОГО АСИСТЕНТА ПСИХОЛОГА

У цьому розділі буде розглянуто структурну побудову програмного забезпечення, функціональну взаємодію компонентів системи, а також описано основні алгоритми, що забезпечують роботу інтелектуального асистента. Основна мета — забезпечити психолога гнучким, конфігурованим та безпечним інструментом підтримки професійної діяльності, в тому числі в контексті комунікації, планування, аналізу даних та використання ШІ-інструментів для текстової інтерпретації

2.1 Обґрунтування архітектурного підходу до побудови системи

Проєктована система повинна забезпечувати зручну, гнучку та масштабовану платформу для інтерактивної взаємодії психолога з користувачами, з можливістю використання сучасних AI-сервісів, збереження історії спілкування, а також нотаток, пов’язаних із сеансами. З урахуванням цих вимог, обрано багаторівневу клієнт-серверну архітектуру з чітким поділом на три основні компоненти: frontend, backend і зовнішні AI-сервіси (рис. 2.1).

Фронтенд реалізується за допомогою React, TypeScript і бібліотеки Material UI (MUI). Кожна з цих технологій виконує важливу роль у побудові клієнтської частини системи, забезпечуючи гнучкість, масштабованість і високу якість користувацького інтерфейсу.

React [6] — це популярна JavaScript-бібліотека для побудови інтерфейсів користувача, створена компанією Meta. Основна ідея React — побудова інтерфейсу за допомогою компонентів — ізольованих, незалежних блоків, які можна повторно використовувати. У контексті цієї системи, React дозволяє розбити інтерфейс на логічні модулі: форма входу, чат із AI, перегляд нотаток, панель навігації тощо. Це значно спрощує розробку, тестування та підтримку коду.

TypeScript [7] — це надмножина JavaScript, яка додає статичну типізацію. Завдяки типам, розробники отримують змогу описувати структуру об’єктів, параметри функцій, типи змінних тощо. Це підвищує безпеку розробки, дає змогу виявляти помилки ще до запуску програми, полегшує автодоповнення в редакторі коду та спрощує роботу з API. У межах проєкту TypeScript є особливо корисним у взаємодії з бекендом, де передаються структуровані дані — наприклад, об'єкти користувача, повідомлення чату або нотатки.

Material UI (MUI) [8] — це набір компонентів інтерфейсу, створений на основі принципів Google Material Design. MUI дозволяє швидко створювати сучасні й послідовні елементи UI — кнопки, форми, діалоги, вкладки, таблиці та багато іншого. У проєкті MUI використовується для побудови всіх основних інтерфейсів: форми автентифікації, редактора нотаток, елементів навігації, повідомлень чату тощо. Використання MUI забезпечує зручність користування, адаптивність до різних екранів і візуальну цілісність усіх частин системи.

Загалом, ця комбінація технологій дає змогу створити гнучкий, розширюваний та інтуїтивно зрозумілий інтерфейс користувача, який ефективно працює у зв’язці з серверною логікою та AI-моделями.

Бекенд, реалізований на платформі .NET [9], виступає як центральний логічний рівень обробки запитів, а також як посередник між клієнтською частиною, AI-сервісами та базою даних. Основні контролери відповідають за автентифікацію, обробку повідомлень і управління нотатками. Всі запити з фронтенду маршрутизуються до відповідних сервісів через ці контролери. Окремий сервіс AI-інтеграції забезпечує взаємодію системи з зовнішніми постачальниками мовних моделей — OpenAI та Groq. Обидва сервіси надають доступ до сучасних моделей штучного інтелекту через API, що дозволяє інтегрувати їх у власну інфраструктуру без потреби самостійно хостити моделі або витрачати ресурси на їхнє обслуговування.

OpenAI [10] пропонує потужні пропрієтарні моделі, такі як GPT-4.1, які демонструють високу якість генерації тексту, контекстного аналізу, резюмування та інших когнітивних завдань. Ці моделі добре підходять для реалізації

інтелектуальних чатів, асистентів та агентів, які працюють з великим контекстом, виконують інструкції та аналізують текстові дані.

Groq [11], у свою чергу, надає доступ до моделей з відкритим кодом (наприклад, LlaMA [12], Mistral [13], Gemma [14]) і вирізняється інноваційною апаратною архітектурою на основі власних процесорів — LPU [15] (Language Processing Units). На практиці це означає значно вищу швидкість відповіді при виконанні запитів до моделей, навіть при складних або великих обчислювальних завданнях. Завдяки цьому інтеграція Groq дає змогу системі не втрачати час на очікування при генерації великих відповідей або обробці великих обсягів тексту — наприклад, при аналізі нотаток, резюмуванні історії чату чи роботі з векторними запитами.

Таким чином, система має змогу динамічно обирати між OpenAI і Groq залежно від типу запиту, пріоритетів продуктивності або бюджету. Вся логіка маршрутизації й обробки запитів до моделей зосереджена всередині AI-сервісу, що спрощує загальну архітектуру та дозволяє масштабувати функціональність без втручання в інші компоненти.

Окрім того, реалізовано окремі сервіси для збереження історії повідомлень та роботи з нотатками.

Особливу увагу приділено зберіганню даних. Всі дані централізовано зберігаються в базі даних PostgreSQL [16], яка логічно поділена на три частини: сховище користувачів, сховище повідомлень (історія чату) та сховище нотаток психолога. Така сегментація забезпечує гнучкість при доступі до різних типів інформації, дозволяє масштабувати окремі частини системи за необхідності та покращує загальну продуктивність.

Однією з ключових причин вибору саме PostgreSQL як основної СКБД стала наявність розширення PGVector [17] — нативної підтримки зберігання та пошуку по векторних представленнях даних. Це особливо актуально для задач, пов’язаних з векторним пошуком у нотатках або історії діалогу. Наприклад, кожну нотатку або повідомлення можна представити у вигляді вектору ознак (ембедингу), отриманого з мовної моделі. Ці вектори зберігаються у спеціальному векторному стовпці, що дозволяє ефективно знаходити подібні за змістом записи через

операції approximate nearest neighbor search — без потреби створювати окреме сховище або використовувати зовнішні сервіси.

Крім того, PostgreSQL чудово підходить для контейнеризації та легко розгортається у Docker-середовищі [18], що дозволяє гнучко управляти конфігурацією та обсягами зберігання. У разі необхідності база може бути навіть змонтована всередину десктопного чи мобільного застосунку як локальне сховище — наприклад, у випадках автономної роботи або для офлайн-режиму. Це розширює можливості майбутнього масштабування та адаптації системи під різні сценарії використання.

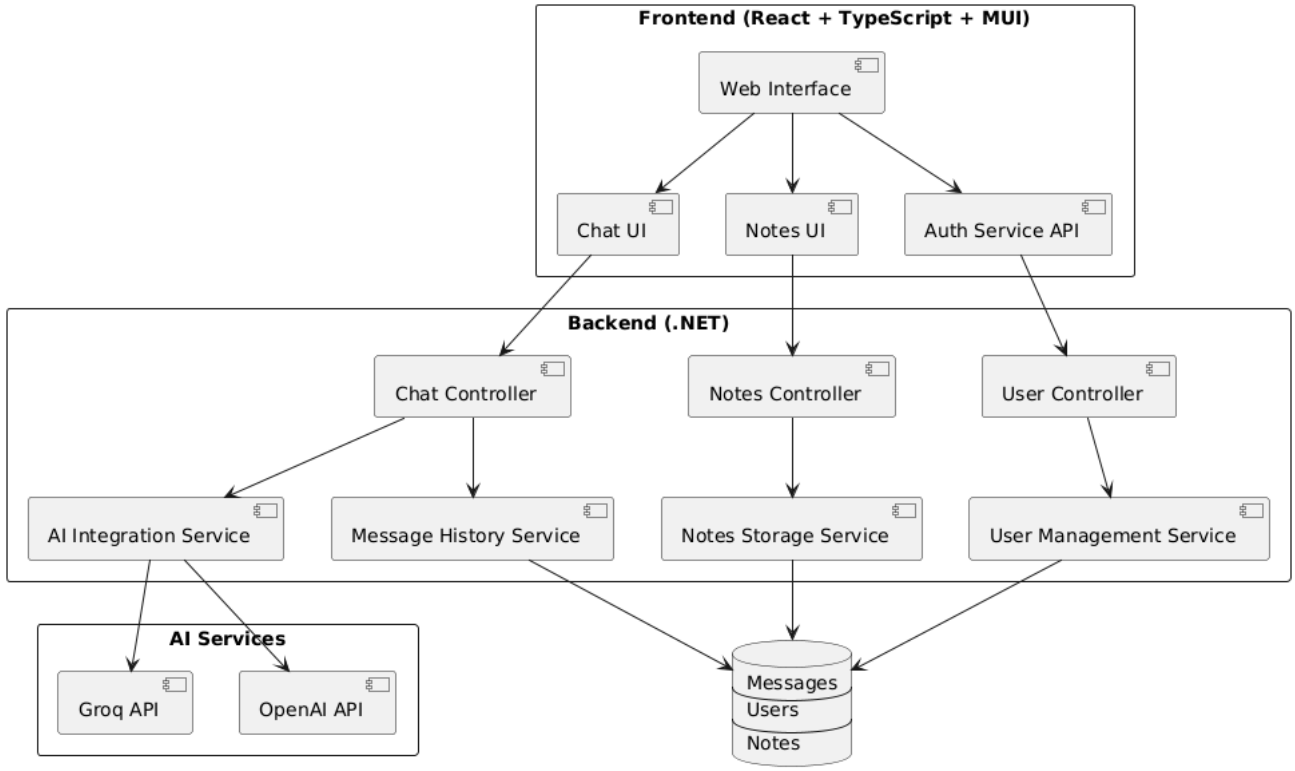

Рисунок 2.1 – Діаграма компонентів системи

Такий архітектурний підхід дозволяє досягти високого рівня модульності та забезпечити розширюваність у майбутньому. Наприклад, легко додати нові джерела AI або функціонал (такий як планувальник сеансів або аналітика), не змінюючи фундаментальні частини системи. Крім того, чіткий поділ відповідальностей між сервісами значно спрощує тестування, налагодження та підтримку системи.

Таким чином, обрана архітектура відповідає сучасним вимогам до надійних та масштабованих вебсистем і забезпечує ефективну взаємодію всіх компонентів системи у реальному часі.

2.2 Обґрунтування вибору мовної моделі штучного інтелекту

У рамках побудови сучасної системи, яка використовує штучний інтелект для взаємодії з користувачем та підтримки психолога, важливо чітко визначити, які саме типи моделей машинного навчання необхідно залучити. Кожна модель у системі виконує свою роль: одна відповідає за пошук пов’язаного контенту, інша — за ведення розмови, а ще інша — за виконання складних інструкцій і аналіз даних. Саме розподіл обов’язків між моделями дозволяє досягнути високої точності та швидкодії без зайвого навантаження на окремі компоненти.

У запропонованій архітектурі передбачено використання трьох категорій моделей:

- Embedding-моделі — для пошуку та контекстуалізації даних.
- Language models — для ведення розмови у чаті.
- AI-агенти — для виконання аналітичних або складних інструкцій.

Embedding-модель перетворює текстовий фрагмент на вектор — набір чисел, який математично відображає сенс цього тексту. Це дає змогу будувати пошук не за ключовими словами, а за смисловою близькістю. Наприклад, якщо психолог створив нотатку про "тривожність і неспокій", система зможе знайти її навіть за запитом "напруга і хвилювання", хоча конкретних збігів слів може не бути.

Векторні представлення дають змогу:

- Здійснювати semantic search по нотатках або історії чату.
- Зберігати вектори у базі даних PostgreSQL із розширенням pgvector.

- Визначати подібні записи за допомогою косинусної відстані або інших метрик подібності.

Для цього підходить, наприклад, модель text-embedding-3-small від OpenAI, яка дає вектор розміром 1536 ознак для кожного тексту. Один раз сформований вектор можна зберегти й використовувати при кожному пошуку без потреби повторного обчислення.

Розмірність вказує на кількість значень які вміщуються в одному векторі, до прикладу 1536 – що є стандартом, означає, що вектор буде мати вигляд: [1,2,3…,1536]. Тільки для векторизації використовуються не цілі числа, а з певним значенням після коми, може доходити навіть до мільйоних частинок. І виходить що після обробки певного контенту він у вигляді векторів зберігається у векторній базі даних та при наступному запиті слугує контекстною одиницею для пошуку відповідних результатів (рис. 2.2).

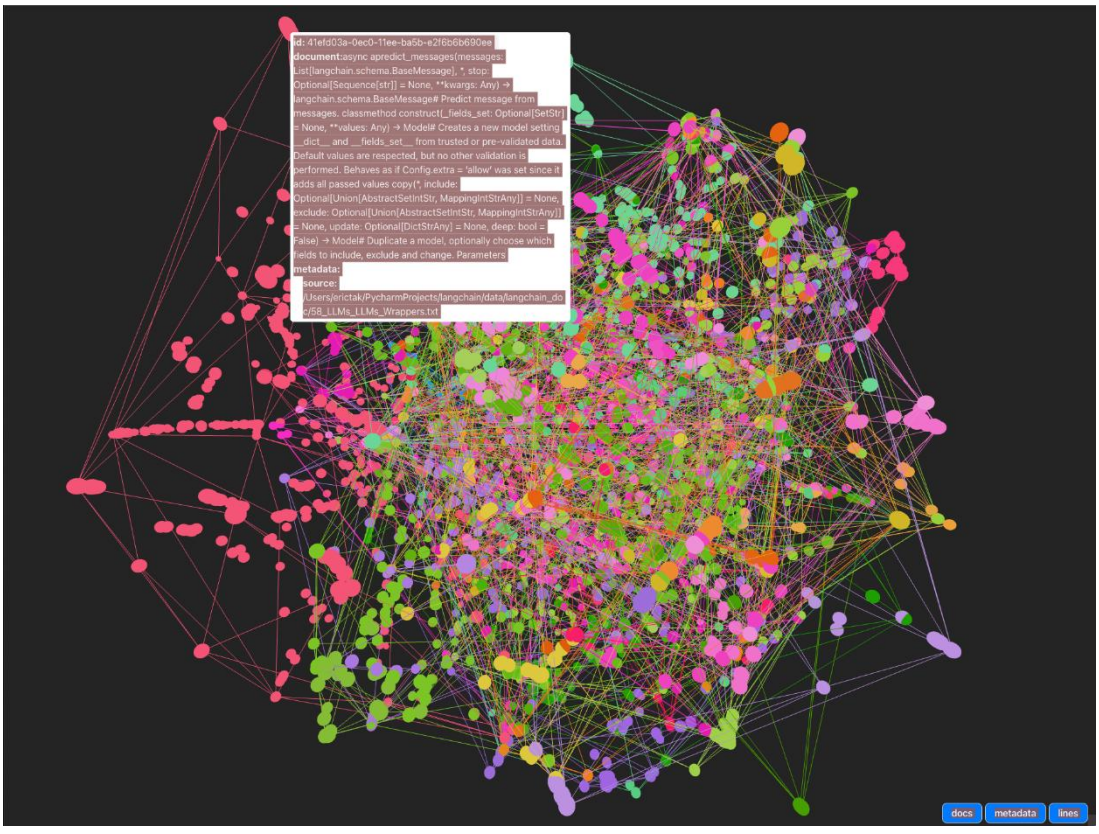

Рисунок 2.2 – Візуалізація векторного сховища

Для пошуку релевантних векторів зазвичай використовується формула Евклідової відстані (L2 distance) [19] — це спосіб вимірювання відстані між двома точками в просторі. Вона визначається як довжина прямої, що з'єднує ці точки, і розраховується за допомогою теореми Піфагора на основі їхніх координат.

Наприклад, у двовимірному просторі відстань між точками $A(x_1, y_1)$ і $B(x_2, y_2)$ обчислюється за формулою 2.1:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.1}$$

У загальному вигляді для n-вимірного простору за формулою 2.2:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.2}$$

Ця метрика широко використовується в машинному навчанні, комп’ютерному баченні та кластеризації, оскільки дозволяє оцінити, наскільки близькими є об’єкти в багатовимірному просторі.

Основою діалогового інтерфейсу є мовна модель великого масштабу (LLM — Large Language Model), яка відповідає на повідомлення користувача, розуміючи контекст. Такі моделі, як GPT-4-turbo від OpenAI чи LLaMA 3 від Meta, натреновані на мільярдах текстів і здатні підтримувати змістовні розмови.

Особливості таких моделей:

- Генерація відповіді токен за токеном з урахуванням усієї історії діалогу.
- Підтримка довгих контекстів (від 8 000 до 128 000 токенів залежно від моделі).

- Керування поведінкою моделі за допомогою системного промпту (інструкції).
- Підключення через API або інтеграцію через проміжний сервіс.

Завдяки цьому користувач може отримувати не просто реакції на запити, а змістовні, логічно побудовані відповіді, які враховують і тон, і мету взаємодії.

Для складніших завдань — аналізу історії повідомлень, підготовки звітів або генерації інтерпретацій — використовуються агенти. Це не окремі моделі, а скоріше налаштовані сценарії використання LLM, в яких модель виконує серію дій на основі мети. Агент може поєднувати генерацію тексту, роботу з базами даних, виклики сторонніх API та навіть логіку ухвалення рішень.

У нашому випадку агенти застосовуються для:

- Генерації психологічного портрета клієнта на основі нотаток.
- Виявлення змін у стані користувача між сесіями.
- Побудови саммари сесій та визначення ключових тем розмови.
- Надання рекомендацій психологу щодо подальших дій.

Платформи, які дозволяють легко реалізувати подібних агентів, — це LangChain [20], Dify [21], OpenAI Functions, або власноруч створений сценарій з підтримкою "tool use".

Такі агенти дозволяють побудувати поведінку, яка наближається до консультанта або асистента — не просто виконавця запитів, а активного аналітика ситуації.

Побудова гнучкої та масштабованої системи неможлива без розділення задач між спеціалізованими моделями. Векторні представлення забезпечують швидкий і точний пошук, мовні моделі гарантують природну розмову, а агентні сценарії додають глибини аналізу й автоматизації. Взаємодія між цими компонентами створює ефективну систему, яка здатна не просто відповідати на запити, а й підтримувати повноцінну роботу психолога.

2.3 Обґрунтування вибору платформи та засобів реалізації (веб застосунок)

Для реалізації системи було обрано підхід побудови вебзастосунку, що працює у браузері без необхідності встановлення додаткового програмного забезпечення. Такий формат є зручним для психологів та клієнтів, забезпечує кросплатформеність і дозволяє масштабувати застосунок у майбутньому до мобільних або десктопних версій на базі того ж коду.

Клієнтська частина реалізована за допомогою React, TypeScript і Material UI (MUI) — потужного стеку, який дозволяє швидко створювати інтерактивні, адаптивні та підтримувані інтерфейси. React забезпечує компонентну структуру інтерфейсу, TypeScript гарантує безпечну типізацію і покращує інтеграцію з бекендом, а MUI надає набір готових до використання компонентів із дотриманням сучасних стандартів дизайну.

На стороні бекенду використовується платформа .NET, яка завдяки своїй продуктивності, модульності та підтримці сучасних стандартів API, є придатною для реалізації логіки обробки запитів, авторизації, взаємодії з базою даних і сторонніми сервісами. Вибір .NET також обумовлений його широкою підтримкою, активною спільнотою, зручним тестуванням та можливістю хостингу як у хмарі, так і на локальних середовищах.

Особливу роль у системі відіграють AI-сервіси, зокрема OpenAI і Groq, які надають доступ до великих мовних моделей через API. Вибір цих платформ обумовлений їхніми різними перевагами: OpenAI — це стабільність і якість закритих моделей, Groq — висока швидкодія завдяки інноваційним чипам LPU, що особливо важливо при обробці великих запитів без затримок. Така комбінація дозволяє системі динамічно обирати найбільш придатне джерело генерації у кожному конкретному випадку, балансуючи між продуктивністю і вартістю.

Для зберігання даних обрано PostgreSQL, як надійну реляційну СКБД із підтримкою розширення PGVector — необхідного для реалізації векторного пошуку. Це дозволяє зберігати не лише структуровані дані, але й векторні подання

повідомлень і нотаток, що забезпечує ефективний пошук за змістом. PostgreSQL також легко контейнеризується і може бути адаптована до різних інфраструктур — від хмарних сервісів до локальних середовищ.

Під час реалізації застосовано Docker для ізоляції сервісів, зручності розгортання і масштабування системи. Такий підхід дозволяє легко запускати окремі компоненти (базу даних, бекенд, фронтенд, AI-проксі) незалежно один від одного, що критично важливо при розробці, тестуванні й оновленнях.

Таким чином, обрані платформи й засоби реалізації дозволяють побудувати надійну, продуктивну та масштабовану систему, що відповідає сучасним вимогам до інтерактивних вебзастосунків у сфері цифрової психологічної допомоги.

2.4 Розробка загальної структурної схеми роботи розумного асистента психолога

Розумний асистент психолога функціонує як посередник між користувачем (пацієнтом чи психологом) і потужними зовнішніми AI-моделями, забезпечуючи персоналізовану підтримку, зберігання даних та зручний інтерфейс взаємодії. Основна увага приділяється логіці обробки запитів, збереженню контексту, забезпеченню зворотного зв’язку та підтримці аналітики.

На загальному рівні схема роботи системи складається з таких логічних етапів (рис. 2.3):

1. Ініціація взаємодії — користувач авторизується та надсилає повідомлення через чат-інтерфейс.
2. Обробка запиту — повідомлення передається в систему, яка перевіряє сесію, виконує логіку маршрутизації та доповнення контексту.
3. Виклик AI — сформований запит надсилається до вибраного AI-провайдера (наприклад, OpenAI або Groq).
4. Генерація відповіді — AI обробляє запит, повертає відповідь, яка потім доповнюється або фільтрується системою.

5. Збереження даних — усі повідомлення та відповіді зберігаються в базі даних, разом із ембедингами, якщо увімкнено режим збереження контексту.
6. Зворотний зв’язок — відповідь надсилається користувачу, а також може бути використана для аналітики або створення нотаток психологом.

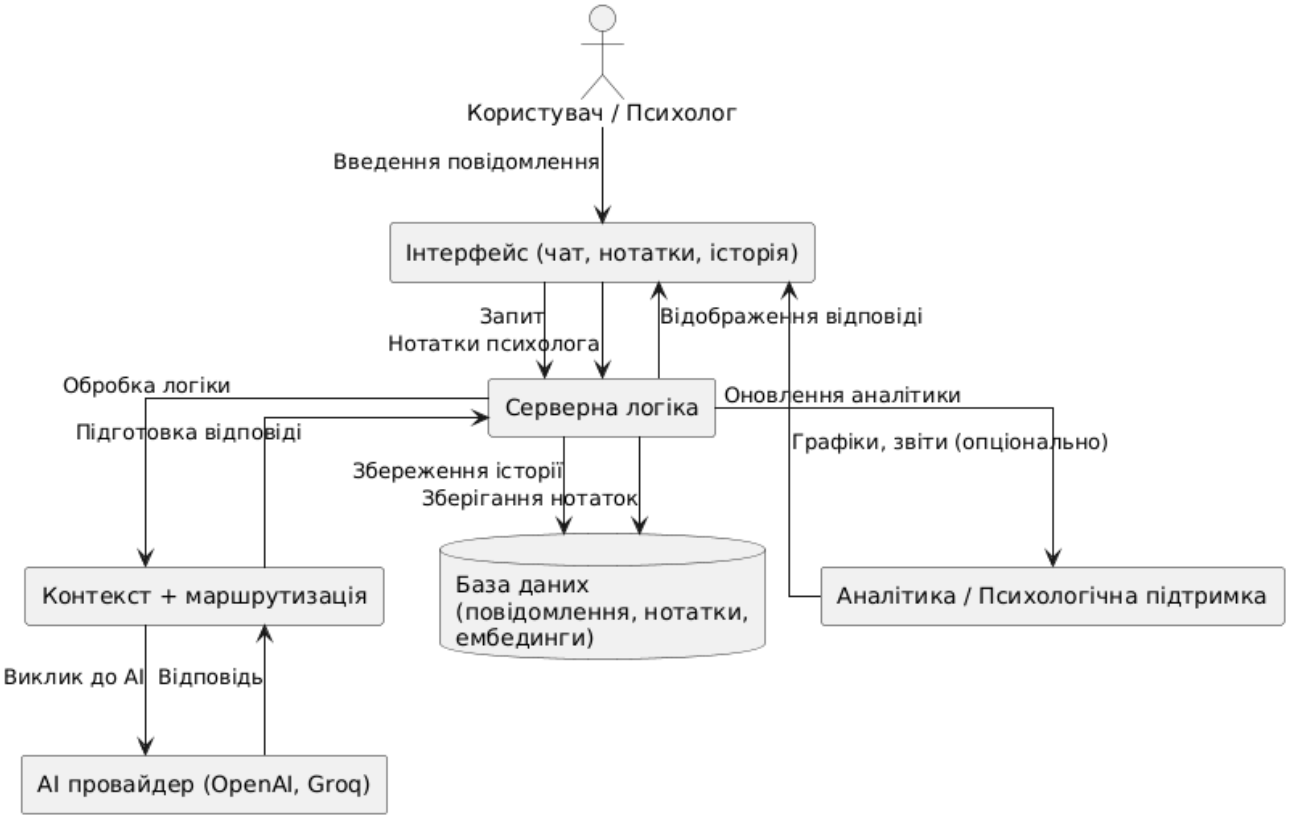

Рисунок 2.3 — Загальна структурна схема роботи системи

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ “РОЗУМНИЙ АСИСТЕНТ ПСИХОЛОГА”

Цей розділ присвячено безпосередній реалізації розробленого програмного продукту — вебзастосунку “Розумний асистент психолога”. На основі розробленої архітектури та технічних рішень, описаних у попередньому розділі, проводиться поетапне створення інтерфейсу користувача, реалізація логіки взаємодії між фронтендом і бекендом, а також підключення зовнішніх AI-сервісів.

У розділі послідовно розглядаються етапи реалізації: створення дизайн-макету, перенесення його у вебформат за допомогою сучасних технологій фронтенду, розробка серверної частини з реалізацією API, а також підключення бази даних і мовних моделей. Завершальним етапом є тестування функціональності застосунку та аналіз отриманих результатів.

Такий підхід дозволяє забезпечити цілісність реалізації, відповідність функціоналу проєктним вимогам і високий рівень якості програмного забезпечення.

3.1 Створення дизайну веб додатку

На початковому етапі розробки було створено дизайн користувацького інтерфейсу вебдодатку “Розумний асистент психолога”. Основною метою проєктування інтерфейсу було забезпечення інтуїтивної взаємодії користувача із системою, мінімізація когнітивного навантаження та відповідність сучасним вимогам до UI/UX.

Для створення макетів інтерфейсу використовувались сучасні підходи до дизайну вебзастосунків, зокрема:

- Мінімалістичний стиль: прості форми, спокійна кольорова гама, чітка типографіка;
- Адаптивна верстка: інтерфейс був спроєктований так, щоб однаково добре відображатись на різних розмірах екранів;

- Зручна навігація: розміщення елементів здійснювалось з урахуванням зручності доступу до основних функцій застосунку (наприклад, запуск сесії, перегляд нотаток, перегляд аналітики);

Під час розробки інтерфейсів було створено кілька варіантів макетів, які проходили внутрішню оцінку відповідності функціональним і візуальним критеріям. На основі обраного макету було сформовано остаточний прототип, який став основою для подальшої імплементації в коді.

Для створення ескізів макетів частково застосовувалися допоміжні інструменти з підтримкою штучного інтелекту, які дозволили пришвидшити генерацію візуальних рішень. Однак остаточне структурування елементів, логіка компонування та вибір стилістики інтерфейсу були виконані вручну, з урахуванням специфіки цільової аудиторії та функціонального призначення системи.

Сторінка реєстрації містить тільки необхідну інформацію, без додаткового навантаження на потенційного користувача, орієнтуючись на інтуєтивну зрозумілість. Це поля для введеня інформації від акаунту та можливість зайти використовуючи сторонні сервіси по типу Google або Facebook (рис. 3.1).

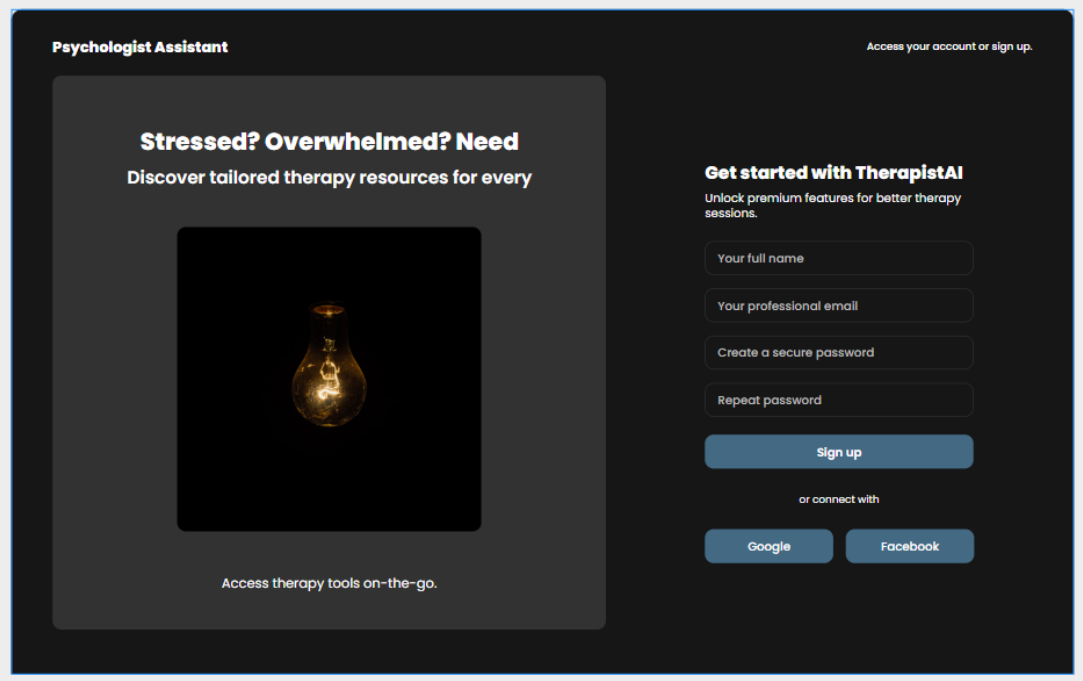

Рисунок 3.1 – Сторінка реєстрації

Головна сторінка вебдодатку має зручну та логічну структуру, яка сприяє швидкій навігації та ефективному доступу до основних функцій. У верхній частині інтерфейсу розташований хедер, що містить поле для пошуку та іконку сповіщень. Це дозволяє швидко знаходити потрібну інформацію в системі та своєчасно реагувати на нові події чи повідомлення (рис. 3.2).

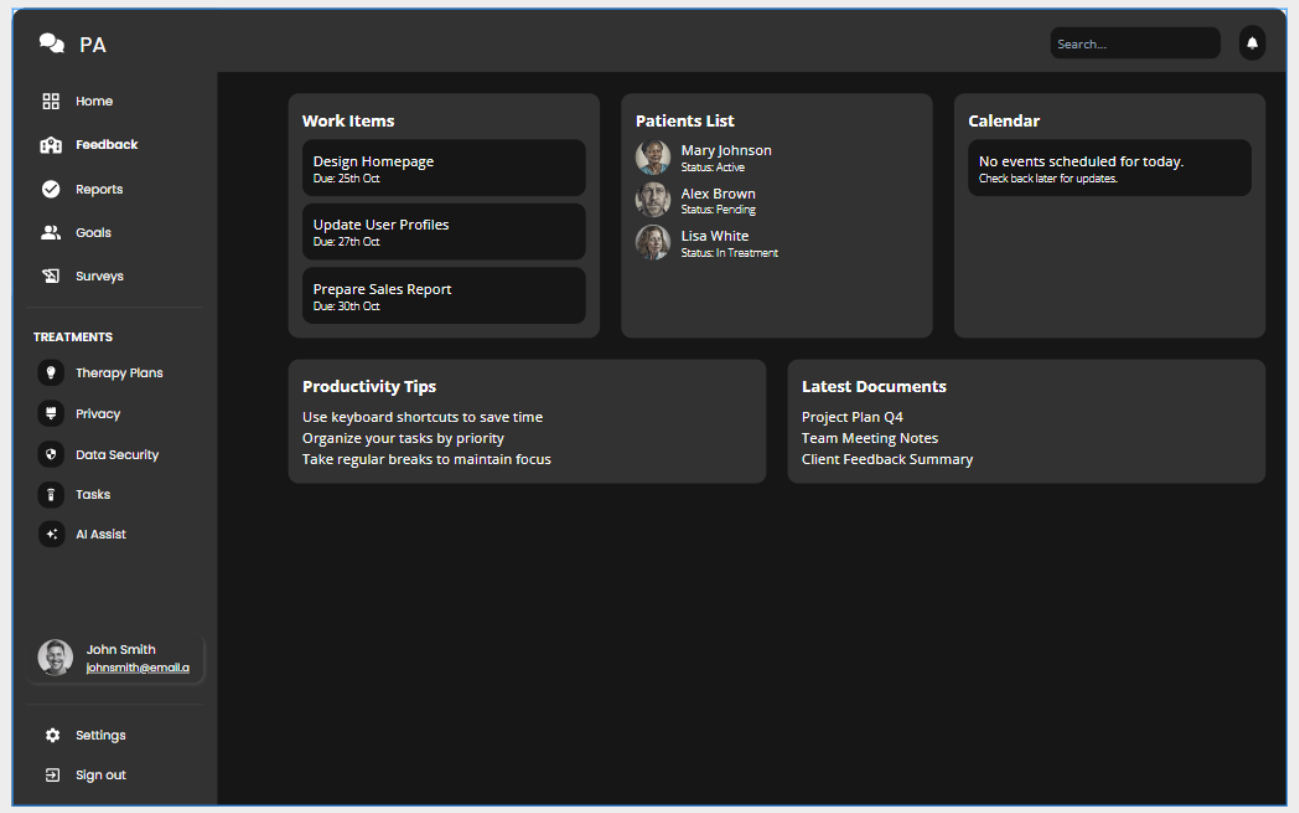

Рисунок 3.2 – Головна сторінка додатку

Зліва розташовується бічна панель (сайдбар), яка виконує роль головного навігаційного меню. Вона містить основні розділи, такі як: домашня сторінка, відгуки, звіти, цілі, опитування, а також функціональні блоки, пов’язані з лікуванням, плани терапії, конфіденційність, безпека даних, завдання та доступ до AI-асистента. В нижній частині сайдбару знаходяться елементи управління профілем користувача: ім’я, електронна адреса, а також кнопки для переходу до налаштувань або виходу із системи.

У центральній — зосереджені ключові елементи, що забезпечують швидкий доступ до важливої інформації та дозволяють організувати роботу психолога. Основним елементом є список поточних завдань із зазначенням термінів їх

виконання — це дозволяє спеціалісту стежити за дедлайнами та не пропускати важливі справи. Поруч розташований список активних пацієнтів, у якому видно їхні імена та статуси, що дає змогу миттєво оцінити ситуацію та приймати рішення без потреби переходити в інші розділи.

Також присутній календар, у якому відображаються заплановані заняття, зустрічі чи події на поточний день — це дозволяє швидко зорієнтуватися в розкладі. У додаткових інформаційних блоках можна знайти поради з підвищення продуктивності чи переглянути останні документи, пов'язані з роботою. Така структура дозволяє психологу мати під рукою все необхідне для щоденної діяльності та забезпечує зручну взаємодію з інтерфейсом системи.

Сторінка AI Assist є ключовим елементом інтелектуального функціоналу вебдодатку "Розумний асистент психолога". Вона виконує роль віртуального помічника, що допомагає користувачу (у нашому випадку — психологу) швидко отримувати консультації, поради та рішення щодо роботи з пацієнтами, використання системи та виконання рутинних завдань (рис. 3.3).

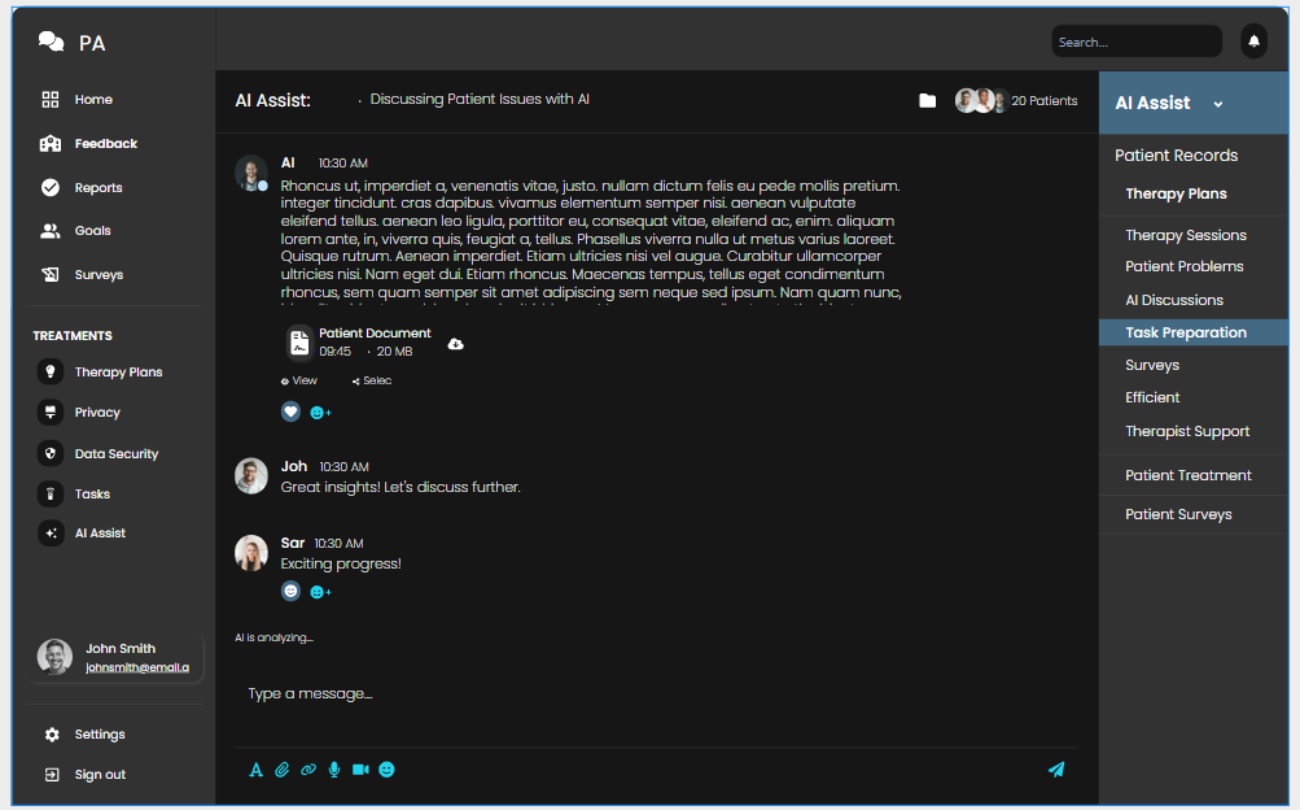

Рисунок 3.3 – Сторінка AI помічника

Центральну частину інтерфейсу займає чат, у якому відбувається безпосередня взаємодія з AI. Штучний інтелект здатен генерувати відповіді на запити користувача, аналізувати надані документи, давати практичні поради щодо планів лікування, пропонувати варіанти терапії або допомагати в організації робочих процесів. Наприклад, психолог може поставити запитання типу: “Який підхід краще для пацієнта з тривожністю?” — і отримати рекомендацію на основі загальновизнаних методик або вмісту картки пацієнта.

У чаті також реалізовано можливість надсилати файли — наприклад, документи пацієнтів, нотатки або анкети. AI здатен автоматично аналізувати ці файли, виділяти ключову інформацію та навіть формувати короткі резюме чи пропозиції для подальшого опрацювання. Це значно зменшує навантаження на фахівця, дозволяючи йому зосередитись на прийнятті рішень, а не на механічній обробці даних.

У чаті можуть брати участь інші користувачі — наприклад, колеги або консультанти. Це дає змогу створювати спільні обговорення складних клінічних випадків або координувати роботу між кількома спеціалістами. У нижній частині передбачене текстове поле для введення повідомлень, а також набір іконок для додавання вкладень, реакцій або швидких дій.

Праворуч знаходиться додаткове навігаційне меню, яке дозволяє перемикатись між різними розділами помічника: “Patient Records”, “Therapy Plans”, “Patient Problems”, “AI Discussions”, “Task Preparation” тощо. Кожен розділ відповідає певному типу консультацій або набору дій, які AI може виконувати. Наприклад, у розділі “Task Preparation” можна обговорити з AI, як краще організувати графік роботи на тиждень, або сформувати список завдань за пріоритетами.

Таким чином, сторінка AI Assist не є просто звичайним чатом — це повноцінний інструмент інтегрованої аналітики, підтримки прийняття рішень і навігації по функціоналу всієї системи. Вона спрямована на покращення якості психологічної допомоги, підвищення ефективності роботи фахівця та створення максимально комфортного середовища для взаємодії з пацієнтами.

Сторінка створення опитувань є ключовим інструментом для психолога або фахівця, який працює з пацієнтами — вона дозволяє швидко та зручно формувати індивідуальні або типові анкети, спрямовані на діагностику психоемоційного стану, оцінку динаміки терапії чи визначення ризиків. Інтерфейс реалізовано таким чином, щоб процес створення був інтуїтивно зрозумілим і не вимагав технічних знань. Психолог може легко додавати питання різного типу: з одним варіантом відповіді, шкалою (наприклад, для оцінки настрою чи рівня тривожності), а також питання з множинним вибором — для ситуацій, де допустимо кілька відповідей (рис. 3.4).

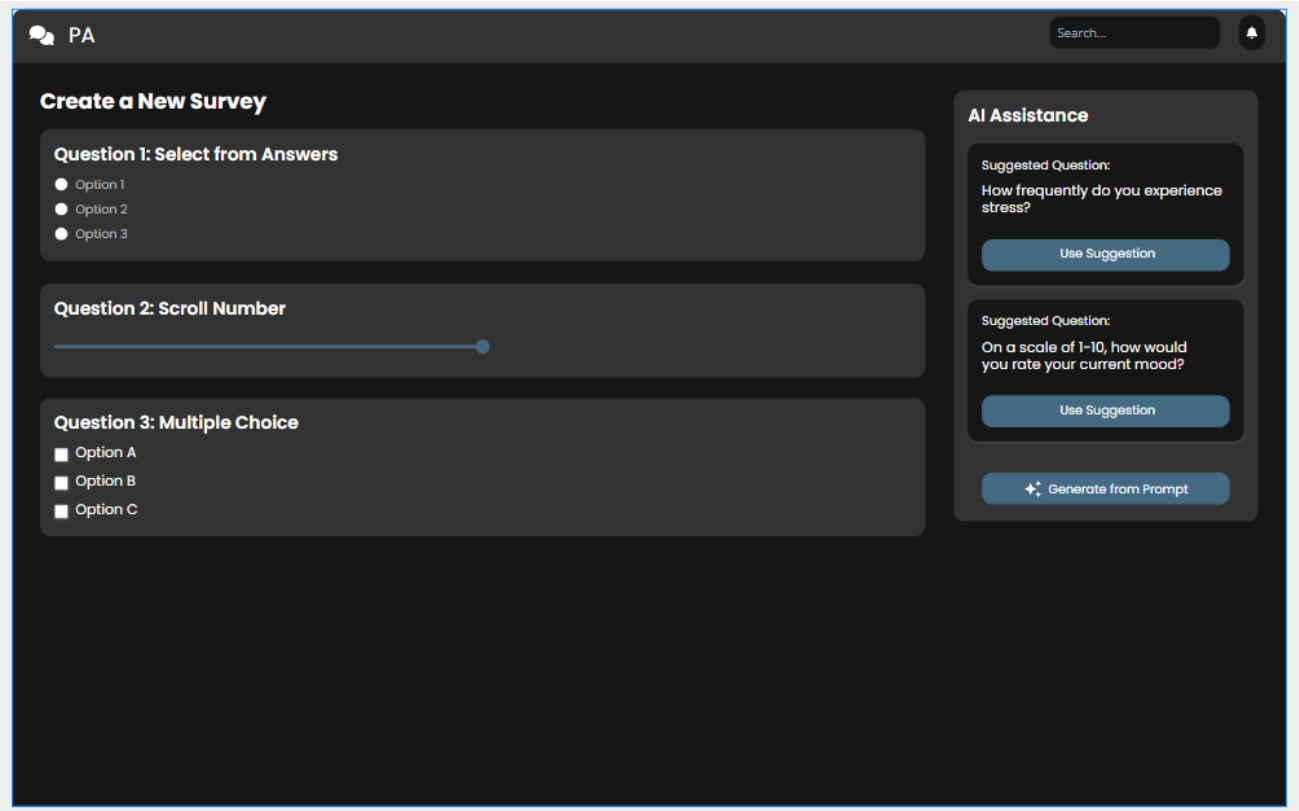

Рисунок 3.4 – Сторінка створення опитування

Особливістю сторінки є інтеграція штучного інтелекту, який допомагає генерувати змістовні та професійно сформульовані питання. Панель AI Assistance справа постійно пропонує відповідні до контексту питання, які можна додати одним натисканням. Наприклад, система сама пропонує такі питання, як “Як часто ви відчуваєте стрес?” або “Оцініть свій настрій за шкалою від 1 до 10” — це базові, але важливі запити, що часто застосовуються у щоденних опитуваннях.

Якщо ж користувач хоче додати унікальне або ситуаційне питання, він може скористатися кнопкою “Generate from Prompt” і просто описати, що хоче спитати — система сформує запитання автоматично.

Створене опитування зберігається в системі та може бути використане кількома способами. По-перше, його можна експортувати у форматі документа (наприклад, PDF або Word), щоб роздрукувати та використати в офлайн-режимі — наприклад, для пацієнтів без доступу до комп’ютера або в умовах прийому в клініці. По-друге, для кожного опитування можна згенерувати посилання, яке надсилається пацієнтові електронною поштою, через месенджер або особистий кабінет. Пацієнт переходить за посиланням, проходить опитування у зручному для себе форматі, а результати одразу автоматично зберігаються в базі даних психолога.

Система відразу після проходження опитування пацієнтом може візуалізувати результати у вигляді графіків, таблиць або просто списку відповідей. Це дозволяє психологу оперативно реагувати на критичні зміни або використовувати результати як основу для консультації. Крім того, результати можуть бути прив’язані до конкретної дати, сеансу або плану терапії, що дозволяє будувати аналітику в динаміці — як змінювався стан пацієнта впродовж місяця чи всього періоду терапії.

Таким чином, ця сторінка є не просто інструментом створення форм — вона є повноцінним модулем для взаємодії з пацієнтами, збору даних, аналітики та документування прогресу. Вона інтегрується з іншими частинами системи, забезпечуючи безперервність і цілісність психотерапевтичного процесу.

Сторінка терапевтичних планів є центральним елементом системи, призначеним для організації, ведення та контролю всіх аспектів психотерапевтичного процесу для кожного клієнта окремо. Цей розділ створено з урахуванням реальних потреб психологів, психотерапевтів, коучів та інших фахівців у сфері ментального здоров’я, які мають справу з великою кількістю пацієнтів, тривалими періодами терапії та потребою у чіткому відслідковуванні прогресу (рис. 3.5).

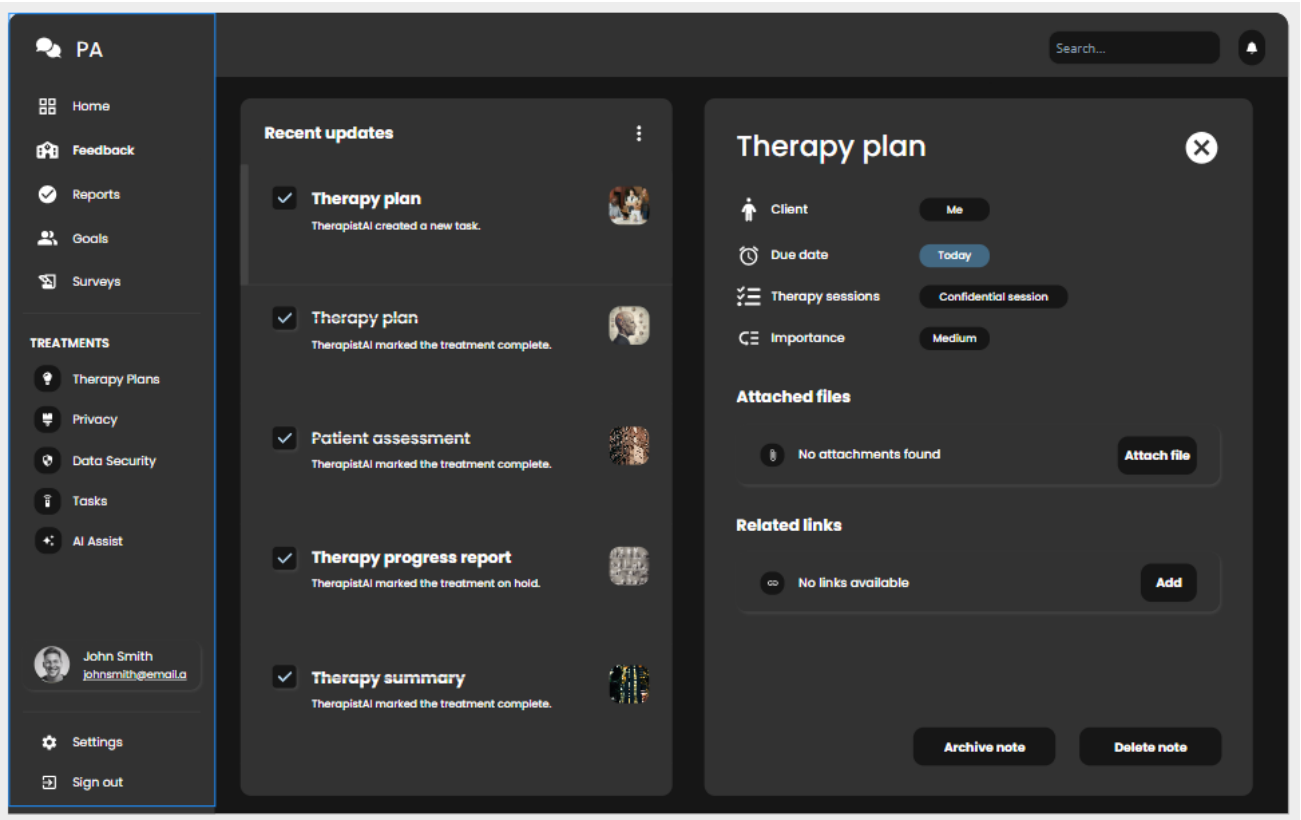

Рисунок 3.5 – Сторінка терапевничних планів

У лівій частині сторінки відображається блок “Recent updates”, що дозволяє швидко побачити всі останні зміни, пов’язані з терапевтичними планами, оцінками пацієнтів, звітами про прогрес тощо. Кожен запис супроводжується коротким описом (наприклад, “TherapistAI created a new task” або “TherapistAI marked the treatment complete”), що дозволяє спеціалісту тримати руку на пульсі терапевтичного процесу. Це особливо корисно в умовах командної роботи або при асистуванні штучного інтелекту, який може самостійно створювати або закривати завдання, базуючись на зібраних даних про пацієнта.

У правій частині сторінки відкривається детальна форма окремого терапевтичного плану. Тут відображено основну інформацію: для якого клієнта призначений план (наприклад, “Me” у випадку персонального плану або тестування), встановлену дату (з можливістю фільтрації за терміновістю), тип сеансів (наприклад, “Confidential session”) і важливість (наприклад, “Medium”). Це дозволяє не лише фіксувати базову інформацію, а й пріоритизувати завдання, працюючи спершу з найбільш критичними випадками.

Важливою функціональністю є можливість додавати вкладення та пов’язані посилання — документи, звіти, таблиці, медичні висновки або навіть зовнішні ресурси, пов’язані з конкретним кейсом. Таким чином, терапевтичний план не є абстрактною заміткою — це повноцінна картка випадку, де в одному місці зберігається все, що може знадобитися спеціалісту в роботі. Кнопки “Attach file” та “Add link” дозволяють оперативно доповнити план потрібною інформацією без перемикання між вікнами або модулями.

Окремо варто зазначити дві важливі опції — “Archive note” та “Delete note”. Перша дозволяє архівувати завершені або тимчасово неактивні плани без повного видалення інформації — що корисно для ведення історії лікування. Видалення ж використовується лише тоді, коли інформація остаточно втрачає актуальність або була створена помилково.

У підсумку, ця сторінка — це не просто форма запису терапевтичних завдань, а функціональний інструмент управління психотерапевтичним процесом. Вона дозволяє бачити всю картину терапії: що вже зроблено, що заплановано, які задачі мають найвищий пріоритет, які матеріали прикріплено, і яка динаміка пацієнта спостерігається. Усе це забезпечує гнучкість, зменшує адміністративне навантаження та дає змогу більше уваги приділяти безпосередній роботі з клієнтом.

Створений дизайн інтерфейсу для системи підтримки психотерапевтів поєднує сучасну візуальну естетику з високою функціональністю та практичністю. Він спрямований на полегшення щоденної роботи спеціаліста, дозволяючи швидко створювати та редагувати терапевтичні плани, формувати опитування для пацієнтів, переглядати прогрес та результати, а також взаємодіяти з підказками від ШІ.

Завдяки логічно структурованим екранам, продуманому розміщенню елементів, наявності історії змін і можливості приєднання файлів, система робить управління лікувальним процесом інтуїтивно зрозумілим та ефективним. Особливу цінність додає модуль опитувань, що дозволяє збирати дані у зручному форматі, експортувати результати, ділитися ними через посилання та отримувати миттєвий зворотний зв’язок після проходження пацієнтом.

У підсумку, інтерфейс підсилює не лише організаційні, а й терапевтичні можливості фахівця, сприяючи якіснішому та персоналізованому підходу до кожного пацієнта.

3.2 Перенесення дизайну в код (front-end)

Для початку реалізації графічного інтерфейсу необхідно створити базовий front-end проєкт за допомогою React з підтримкою TypeScript. Ініціалізація нового проєкту виконується за допомогою наступної команди у терміналі:

npx create-react-app therapist-assistant --template typescript

Далі додаємо бібліотеки, які будуть використані для розробки інтерфейсу. Основною бібліотекою візуальних компонентів у цьому проєкті є Material UI (MUI), тож встановлюємо її разом із залежностями та іншими потрібними пакетами для старту:

npm install @mui/material @mui/icons-material @emotion/react @emotion/styled
npm install react-router-dom, npm install axios

Після створення основних компонентів інтерфейсу користувача та завершення етапу дизайну було розпочато побудову структури маршрутизації в React-застосунку. Для цього у файлі App.tsx було налаштовано компонент Router, який забезпечує навігацію між окремими сторінками вебдодатку. Основою для маршрутизації слугує бібліотека react-router-dom, яка дозволяє задавати маршрути у вигляді окремих компонентів <Route> всередині тегу <Routes>.

Кожен маршрут відповідає окремій логічній сторінці, реалізованій як React-компонент. У поточній конфігурації застосунок включає такі маршрути:

- /login — сторінка авторизації;
- /main — головна сторінка після входу користувача;

- /assistant — сторінка, де працює інтегрований AI-асистент;
- /survey — модуль з опитуваннями;
- /therapy — модуль ведення терапевтичних планів;
- /404 — спеціальна сторінка для обробки помилок навігації.

Окремо було реалізовано перенаправлення всіх невідомих маршрутів на сторінку помилки за допомогою компонента <Navigate to="/404" replace />. Це дозволяє уникнути ситуацій, коли користувач вводить неправильну адресу і бачить порожній екран — замість цього він потрапляє на дружню до користувача сторінку з поясненням.

На рисунку 3.6 наведено спрощений фрагмент коду, що ілюструє дану логіку:

```
<Routes>
  <Route path="/login" element={<Login />} />
  <Route path="/main" element={<Main />} />
  <Route path="/assistant" element={<AIAssistant />} />
  <Route path="/survey" element={<Survey />} />
  <Route path="/therapy" element={<Therapy />} />
  <Route path="/404" element={<ErrorPage />} />
  <Route path="*" element={<Navigate to="/404" replace />} />
</Routes>
```

Рисунок 3.6 – Маршрутизація додатку

Особливу увагу було приділено сторінці з кодом помилки 404, що використовується у разі спроби переходу на неіснуючу або недоступну адресу. Компонент ErrorPage.tsx реалізовано із застосуванням компонентів бібліотеки Material UI (MUI), таких як Box, Typography, Button, Paper та Container, що дозволяє не лише дотриматися єдиного стилю інтерфейсу, а й забезпечити привабливий і зрозумілий зовнішній вигляд повідомлення про помилку.

Сама сторінка стилістично відповідає загальному темному дизайну всього застосунку. Вона відображає великий заголовок з номером помилки — 404, коротке повідомлення про те, що запитану сторінку не знайдено, та кнопку, яка

дозволяє швидко повернутись на головну сторінку /main. Перехід реалізовано за допомогою хука useNavigate, що є частиною функціональності react-router-dom.

Фрагмент коду відповідного компонента зображений на рисунку 3.7:

```
const navigate = useNavigate();

const handleRedirect = () => {
  navigate('/main');
};
```

Рисунок 3.7 – Фрагмент хука повернення на головну сторінку

Це дозволяє миттєво здійснювати переадресацію, не перезавантажуючи сторінку. Візуально інтерфейс побудовано навколо Paper, що створює ефект картки з тінню, на якій розміщено всі елементи повідомлення. Сторінка має гнучку структуру і адаптується до різних розмірів екрана завдяки використанню контейнера Container з фіксованою шириною maxWidth="sm".

Таким чином, реалізація маршрутизації стала основою для навігації у вебзастосунку. Кожна функціональна частина має свій окремий маршрут, що дозволяє ізольовано керувати станами кожного модуля. Додатково реалізована сторінка обробки помилок маршрутизації значно покращує досвід користувача, надаючи зрозумілу відповідь у разі помилок введення або недоступності ресурсів (рис. 3.8). У подальшому цей підхід дозволить легко масштабувати застосунок, додаючи нові сторінки або оновлюючи існуючі маршрути без суттєвих змін архітектури.

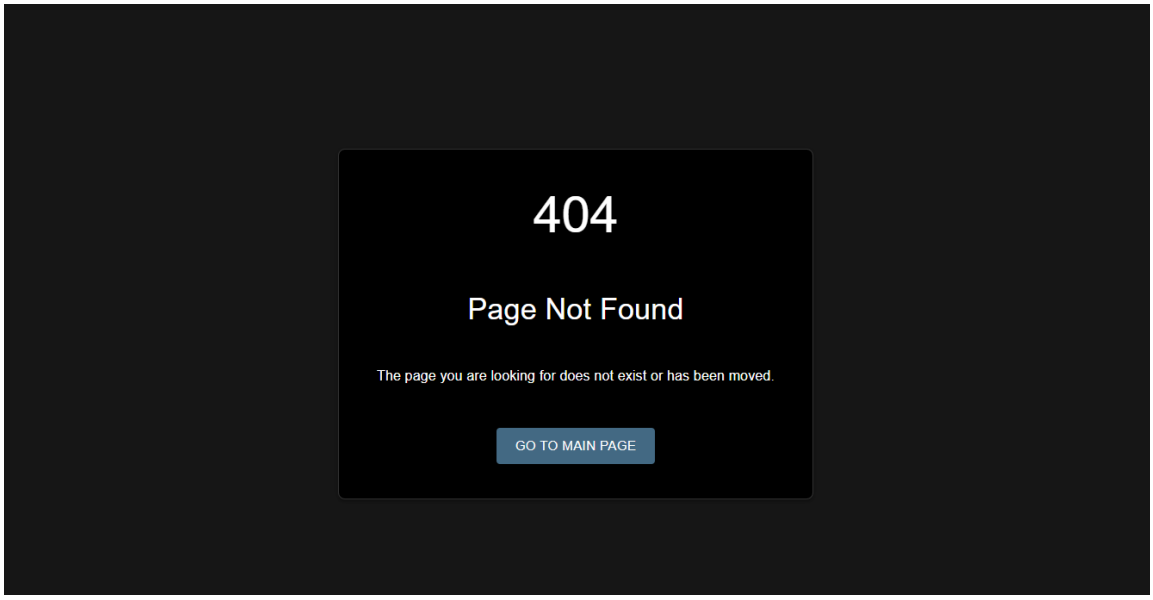

Рисунок 3.8 – Сторінка 404

Для обміну даними між клієнтською частиною застосунку та сервером реалізовано структурований підхід до надсилання HTTP-запитів за допомогою бібліотеки Axios. Це забезпечує уніфіковану, безпечну та розширювану систему комунікації з API, яка підтримує автентифікацію, обробку помилок та типобезпеку.

Вся логіка API винесена до окремого модуля apiClient.ts, який містить конфігурацію клієнта, перехоплення запитів і відповідей (interceptors), а також єдиний універсальний метод apiRequest для надсилання HTTP-запитів.

Основні складові структури:

- apiClient.ts — створює екземпляр Axios з базовими налаштуваннями:
  - базовий URL, що зчитується з .env файлу;
  - заголовки запиту (тип контенту — application/json);
  - автоматичне додавання токена авторизації до кожного запиту;
  - обробка помилок, включно з автоматичним перенаправленням користувача при виявленні помилки автентифікації (статус 401).
- apiRequest<T>(config) — обгортка над Axios-запитом, яка повертає типізовану відповідь або виняток ApiError при помилці.
- Інтерфейс ApiError використовується для уніфікованої обробки помилок. Він визначає статус помилки, повідомлення та допоміжні методи для

виявлення типу помилки (автентифікація, валідація, відсутність ресурсу тощо).

Токен авторизації зберігається у localStorage після успішного входу користувача. Усі наступні запити автоматично містять цей токен у заголовку Authorization, що забезпечується інтерсептором на рисунку 3.9:

```
apiClient.interceptors.request.use((config) => {
  const token = localStorage.getItem('auth_token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

Рисунок 3.9 – Інтерсептор для обробки токену авторизації

У разі закінчення терміну дії токена або невдалої автентифікації, інтерсептор відповіді очищає локальне сховище та перенаправляє користувача на сторінку входу (рис. 3.10).

```
apiClient.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response && error.response.status === 401) {
      localStorage.removeItem('auth_token');
      if (window.location.pathname !== '/login') {
        window.location.href = '/login';
      }
    }
    return Promise.reject(handleApiError(error));
  }
);
```

Рисунок 3.10 – Перевірка авторизаційного токену

Усі функції взаємодії з конкретними кінцевими точками API згруповані у відповідні об’єкти, що зберігаються в каталозі endpoints/. Наприклад, модуль SurveyAPI (рис. 3.11) містить методи для отримання опитувань, відправки відповідей та перегляду результатів

```
export const SurveyAPI = {
  getSurveys: async (): Promise<Survey[]> => {
    return apiRequest<Survey[]>({ method: 'GET', url: '/surveys' });
  },

  getSurveyById: async (surveyId: string): Promise<Survey> => {
    return apiRequest<Survey>({ method: 'GET', url: `/surveys/${surveyId}` });
  },

  submitSurveyResponse: async (response: Omit<SurveyResponse, 'id' | 'submittedAt'>): Promise<SurveyResponse> => {
    return apiRequest<SurveyResponse>({ method: 'POST', url: '/surveys/responses', data: response });
  },
};
```

Рисунок 3.11 – Модуль взаємодії з SurveyApi

Кожен метод працює з відповідними інтерфейсами типів, визначеними у файлах models/, що забезпечує типобезпеку як запитів, так і відповідей.

Описана структура дозволяє централізовано управляти всіма аспектами взаємодії з API, спрощує масштабування проєкту та покращує підтримуваність коду. Вся логіка API є повторно використовуваною, модульною та чітко розділеною відповідно до принципів чистої архітектури.

3.3 Розробка серверної частини

Серверна частина вебзастосунку «Розумний асистент психолога» була розроблена з використанням платформи ASP.NET Core Web API, що забезпечує високу продуктивність, модульність і підтримку сучасних стандартів розробки RESTful API. Цей підхід дозволив створити надійну, масштабовану та безпечну серверну інфраструктуру (рис. 3.12), яка обробляє запити від клієнтської частини,

взаємодіє з базою даних і зовнішніми AI-сервісами, а також забезпечує логіку бізнес-процесів.

Розробка серверної частини розпочалася зі створення базового проєкту за допомогою шаблону webapi у ASP.NET Core. Для цього використано команду:

dotnet new webapi -n SmartAssistant.API

Цей шаблон надає базову конфігурацію для REST API, включаючи мінімальний набір контролерів, middleware для обробки запитів і залежностей, а також базові налаштування безпеки. Для забезпечення модульності та зручності підтримки коду було створено багатошарову архітектуру, яка включає наступні проєкти в рамках одного рішення:

- SmartAssistant.API — точка входу до додатку, контролери, конфігурація.
- SmartAssistant.BLL — бізнес-логіка.
- SmartAssistant.DAL — доступ до бази даних (Entity Framework Core).
- SmartAssistant.Common — спільні структури даних, DTO, enum-и.
- SmartAssistant.Integration — взаємодія з зовнішніми сервісами (OpenAI, Groq тощо).

Така структура відповідає принципам чистої архітектури, що забезпечує чітке розділення обов’язків, спрощує тестування та дозволяє легко масштабувати систему в майбутньому.

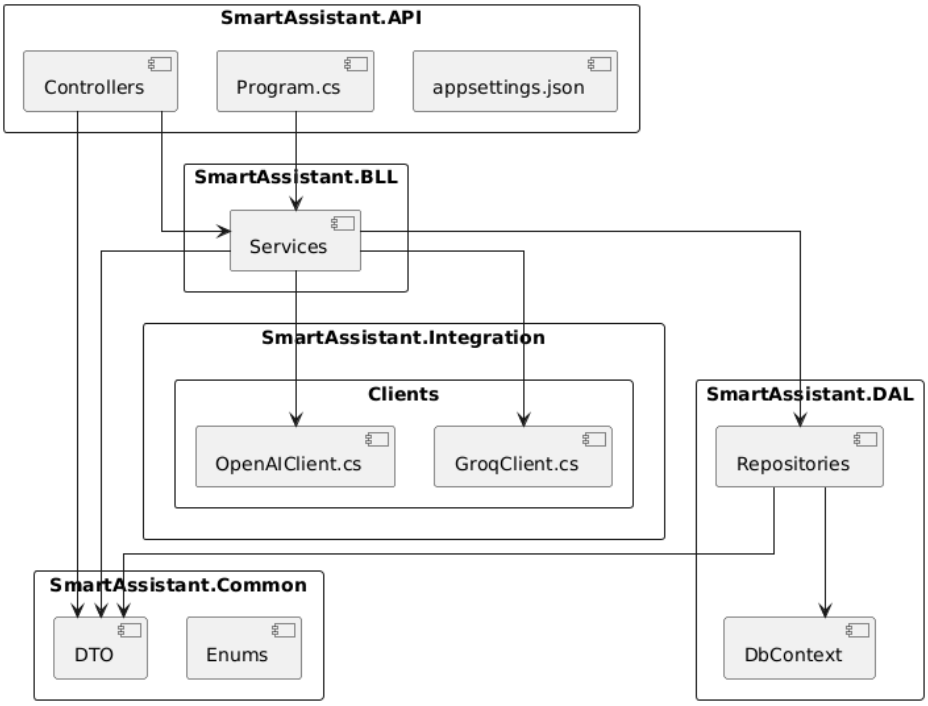

Рисунок 3.12 – Архітектура серверного додатку

Для зберігання даних, таких як інформація про користувачів, історія чатів, нотатки психолога, налаштування сесій і результати взаємодії з AI-сервісами, було обрано реляційну базу даних PostgreSQL з розширенням PGVector для підтримки векторного пошуку. Вибір PostgreSQL обумовлений її надійністю, підтримкою складних запитів, можливістю контейнеризації через Docker і нативною підтримкою векторних операцій, що є критичним для реалізації семантичного пошуку по нотатках і повідомленнях.

У проєкті SmartAssistant.DAL створено контекст бази даних AppDbContext, який успадковує клас DbContext від Entity Framework Core. Цей контекст визначає набори даних (DbSet) для основних сутностей системи:

- Users — інформація про користувачів (психологів і клієнтів).
- ChatSessions — дані про сесії взаємодії (наприклад, чати між клієнтом і AI).
- Messages — історія повідомлень у чатах.
- Notes — нотатки психолога, пов’язані з клієнтами або сесіями.
- AIResponses — збережені відповіді від AI-сервісів для подальшого аналізу.
- VectorEmbeddings — векторні представлення нотаток і повідомлень для семантичного пошуку.

Моделі даних визначено у вигляді класів у каталозі Entities. Наприклад, клас Message включає поля для тексту повідомлення, ідентифікатора сесії, автора (користувач або AI), часу створення та векторного представлення. Зв’язки між сутностями налаштовано за допомогою анотацій даних та Fluent API, що забезпечує коректне відображення моделей на таблиці бази даних.

Для ізоляції логіки доступу до даних використано шаблон репозиторію. Кожен репозиторій (наприклад, UserRepository, MessageRepository) реалізує відповідний інтерфейс (IUserRepository, IMessageRepository), що забезпечує єдиний контракт для взаємодії з даними. Це дозволяє абстрагувати бізнес-логіку від деталей реалізації бази даних, спрощує тестування та забезпечує гнучкість для заміни СКБД у майбутньому.

Фрагмент коду для контексту бази даних на рисунку 3.13:

```
public class AppDbContext : DbContext
{
    public DbSet<User> Users { get; set; }
    public DbSet<ChatSession> ChatSessions { get; set; }
    public DbSet<Message> Messages { get; set; }
    public DbSet<Note> Notes { get; set; }
    public DbSet<AIResponse> AIResponses { get; set; }
    public DbSet<VectorEmbedding> VectorEmbeddings { get; set; }

    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Message>()
            .HasOne(m => m.ChatSession)
            .WithMany(cs => cs.Messages)
            .HasForeignKey(m => m.ChatSessionId);

        modelBuilder.Entity<VectorEmbedding>()
            .HasIndex(ve => ve.Embedding)
            .HasMethod("vector_cosine_ops");
    }
}
```

Рисунок 3.13 – Частковий плаклад контексту бази данних

Бізнес-логіка системи зосереджена в проєкті SmartAssistant.BLL, де реалізовано сервіси для обробки основних сценаріїв роботи застосунку. Основні сервіси включають:

- AuthService — відповідає за автентифікацію та авторизацію користувачів. Використовує JWT (JSON Web Tokens) для генерації токенів після успішного входу. Токени зберігаються в localStorage на стороні клієнта та додаються до заголовків запитів для захищених ендпоінтів.
- ChatService — керує логікою чатів, включаючи створення сесій, збереження повідомлень і маршрутизацію запитів до AI-сервісів.
- NoteService — забезпечує створення, редагування, видалення та пошук нотаток психолога. Використовує векторний пошук для знаходження релевантних записів за змістом.
- SurveyService — відповідає за створення опитувань, збереження відповідей і генерацію аналітичних звітів.
- AIService — координує взаємодію з зовнішніми AI-сервісами (OpenAI, Groq), включаючи вибір моделі залежно від типу запиту, обробку відповідей і збереження результатів.

Кожен сервіс реалізовано як окремий клас, що залежить від відповідних репозиторіїв через інтерфейси. Залежності впроваджуються через вбудований DI-контейнер ASP.NET Core, що забезпечує модульність і тестопридатність коду. Наприклад, сервіс ChatService використовує IMessageRepository і IChatSessionRepository для збереження повідомлень і сесій, а також викликає методи з SmartAssistant.Integration для взаємодії з AI.

Для інтеграції з зовнішніми AI-сервісами створено проєкт SmartAssistant.Integration, який містить клієнти для роботи з OpenAI і Groq. Клієнт для OpenAI базується на офіційній бібліотеці OpenAI SDK, підключеній через NuGet. Ця бібліотека забезпечує типізований доступ до API, спрощуючи відправлення запитів і обробку відповідей. Наприклад, запит до моделі GPT-4-turbo виглядає так як на рисунку 3.14:

```
public async Task<string> GetOpenAIResponseAsync(string prompt, string model = "gpt-4-turbo")
{
    var response = await _openAIClient.Completions.CreateAsync(new CompletionRequest
    {
        Prompt = prompt,
        Model = model,
        MaxTokens = 1000
    });
    return response.Choices.First().Text;
}
```

Рисунок 3.14 – Запит до GPT моделі

Для Groq, який має API, сумісне з OpenAI, але використовує власні моделі (наприклад, LLaMA 3), створено кастомний HTTP-клієнт на основі HttpClient. Це дозволило точно налаштувати параметри запитів, такі як таймаути, заголовки та обробку помилок. Приклад запиту до Groq на рисунку 3.15:

```
public async Task<string> GetGroqResponseAsync(string prompt, string model = "llama3-70b")
{
    var request = new HttpRequestMessage(HttpMethod.Post, "completions");
    request.Headers.Add("Authorization", $"Bearer {_groqApiKey}");
    request.Content = JsonContent.Create(new { prompt, model, max_tokens = 1000 });

    var response = await _httpClient.SendAsync(request);
    response.EnsureSuccessStatusCode();

    var responseData = await response.Content.ReadFromJsonAsync<GroqResponse>();
    return responseData.Choices.First().Text;
}
```

Рисунок 3.15 – Запит до Groq провайдера

Обидва клієнти інкапсулюють логіку обробки помилок, логування запитів і відповідей, а також підтримують динамічний вибір моделі залежно від типу запиту (наприклад, Groq для швидких відповідей, OpenAI для складних аналітичних завдань). Логіка вибору моделі реалізована в AIService, який аналізує тип запиту (генерація тексту, аналіз нотаток, створення опитувань) і спрямовує його до відповідного провайдера.

Конфігурація серверної частини зберігається у файлі appsettings.json, який містить:

- Рядок підключення до PostgreSQL.
- API-ключі для OpenAI та Groq.
- Налаштування логування (використовується Serilog для структурованого логування).
- Параметри JWT для автентифікації.
- Налаштування CORS для безпечної взаємодії з фронтендом.

Усі залежності (репозиторії, сервіси, клієнти) реєструються в DI-контейнері в Program.cs за допомогою шаблону розширювальних методів. Наприклад рисунок 3.16:

```
public static class ServiceCollectionExtensions
{
    public static IServiceCollection AddSmartAssistantServices(this IServiceCollection services)
    {
        services.AddScoped<IUserRepository, UserRepository>();
        services.AddScoped<IChatService, ChatService>();
        services.AddScoped<IAIService, AIService>();
        services.AddHttpClient<GroqClient>();
        services.AddSingleton<OpenAIClient>();
        return services;
    }
}
```

Рисунок 3.16 – Реєстрація залежностей

Цей підхід забезпечує централізоване управління залежностями та спрощує їх підміну під час тестування.

Для обробки HTTP-запитів від фронтенду створено набір контролерів у проєкті SmartAssistant.API. Кожен контролер відповідає за окремий функціональний модуль:

- AuthController — обробляє запити на реєстрацію, вхід і оновлення токенів.
- ChatController — керує створенням чат-сесій, відправленням повідомлень і отриманням відповідей від AI (рис. 3.17).

- NoteController — забезпечує CRUD-операції для нотаток і векторний пошук.
- SurveyController — відповідає за створення, редагування та обробку результатів опитувань.

Контролери використовують сервіси з SmartAssistant.BLL для виконання бізнес-логіки та повертають відповіді у форматі JSON.

```
[HttpPost("send-message")]
public async Task<IActionResult> SendMessage([FromBody] SendMessageRequest request)
{
    var response = await _chatService.SendMessageAsync(request.SessionId, request.Message);
    return Ok(new { Response = response });
}
```

Рисунок 3.17 – Ендпоінт для відправлення повідомлення в чат:

Для захисту ендпоінтів використано атрибут [Authorize], який перевіряє наявність валідного JWT-токена в заголовку запиту.

Безпека системи забезпечується на кількох рівнях:

- Автентифікація та авторизація: Використання JWT-токенів із перевіркою через middleware.
- Шифрування даних: Усі чутливі дані (наприклад, паролі) зберігаються в хешованому вигляді за допомогою BCrypt.
- CORS: Налаштовано для обмеження доступу до API лише з дозволених доменів.
- Обробка помилок: Уніфікована обробка виключень через middleware, що повертає структуровані JSON-повідомлення про помилки.

Для масштабованості система розгортається в Docker-контейнерах, що дозволяє легко масштабувати окремі компоненти (API, база даних, AI-проксі). Використання PostgreSQL із PGVector забезпечує ефективну обробку великих обсягів даних і векторних запитів.

Серверна частина вебзастосунку «Розумний асистент психолога» була успішно реалізована з використанням ASP.NET Core Web API, Entity Framework Core і PostgreSQL. Модульна архітектура, побудована на принципах чистої архітектури, забезпечує чітке розділення обов'язків між рівнями API, бізнес-логіки, доступу до даних і інтеграції з AI-сервісами. Інтеграція з OpenAI і Groq дозволяє гнучко обробляти запити до мовних моделей, а використання PGVector забезпечує ефективний векторний пошук. Налаштування безпеки, масштабованості та конфігурації відповідає сучасним стандартам розробки, що робить систему готовою до подальшого розвитку та інтеграції нових функцій.

3.4 Тестування додатку та аналіз результатів

Тестування вебзастосунку «Розумний асистент психолога» є ключовим етапом розробки, спрямованим на забезпечення коректності, надійності, продуктивності та зручності системи. У цьому підрозділі описано методологію тестування, яка включає юніт-тести, інтеграційні тести та перевірку користувацького інтерфейсу, а також аналіз отриманих результатів. Особливу увагу приділено тестуванню серверної частини (ASP.NET Core), взаємодії з базою даних PostgreSQL і зовнішніми AI-сервісами (OpenAI, Groq), а також оцінці відповідності системи вимогам безпеки, етики та конфіденційності.

Тестування проводилося за багатошаровим підходом, який охоплює всі основні компоненти системи:

1. Юніт-тести: Перевірка окремих модулів серверної частини (сервісів, репозиторіїв) на коректність виконання їхньої логіки. Використовувалася бібліотека xUnit разом із фреймворком для мокінгу Moq для ізоляції залежностей.
2. Інтеграційні тести: Перевірка взаємодії між компонентами системи, зокрема між API-контролерами, бізнес-логікою, базою даних і зовнішніми AI-сервісами. Для цього використано вбудовані інструменти ASP.NET Core (TestServer) та локальну базу даних PostgreSQL у Docker-контейнері.

3. Тестування інтерфейсу користувача (UI): Перевірка коректності відображення компонентів фронтенду (React, TypeScript, MUI) та їхньої взаємодії з API. Використовувалася бібліотека React Testing Library для перевірки рендерингу компонентів і обробки подій.
4. Тестування продуктивності: Оцінка швидкості відповіді API та AI-сервісів, а також масштабованості системи при збільшенні кількості запитів.
5. Тестування безпеки: Перевірка захисту від типових вразливостей (наприклад, SQL-ін’єкцій, XSS) та коректності автентифікації/авторизації через JWT.

Тестування проводилося в ізольованому середовищі з використанням Docker-контейнерів для бази даних і API, що дозволило імітувати реальні умови роботи системи. Для юніт-тестів створено окремий проєкт SmartAssistant.Tests, який містить тести для всіх основних сервісів і репозиторіїв.

Юніт-тести були розроблені для перевірки ключових компонентів бізнес-логіки в проєкті SmartAssistant.BLL. Основна мета — переконатися, що кожен сервіс виконує свої функції коректно незалежно від зовнішніх залежностей (база даних, AI-сервіси). Для ізоляції залежностей використовувалася бібліотека Moq, яка дозволяла створювати мок-об’єкти для репозиторіїв і клієнтів AI.

Приклад юніт-тесту для ChatService. Сервіс відповідає за обробку повідомлень і взаємодію з AI-сервісами.

Цей тест перевіряє:

- Чи викликається метод збереження повідомлення в репозиторії.
- Чи викликається AI-сервіс із правильним текстом запиту.
- Чи повертається очікувана відповідь від AI.

У проєкті SmartAssistant.Tests було створено 47 юніт-тестів, які покривають основні методи сервісів AuthService, ChatService, NoteService, SurveyService та AIService (рис. 3.18). Результати виконання тестів:

- Успішно пройдено: 45 тестів (95.7%).

- Невдалі тести: 2 тести, пов’язані з обробкою некоректних вхідних даних у NoteService (виправлено шляхом додавання валідації).
- Покриття коду: ~85% для бізнес-логіки, що відповідає стандартам для проєктів такого масштабу.

Тести виконувалися за допомогою команди:

dotnet test SmartAssistant.Tests

```
public class ChatServiceTests
{
    private readonly Mock<IMessageRepository> _messageRepositoryMock;
    private readonly Mock<IAIService> _aiServiceMock;
    private readonly ChatService _chatService;

    public ChatServiceTests()
    {
        _messageRepositoryMock = new Mock<IMessageRepository>();
        _aiServiceMock = new Mock<IAIService>();
        _chatService = new ChatService(_messageRepositoryMock.Object, _aiServiceMock.Object);
    }

    [Fact]
    public async Task SendMessageAsync_ValidInput_SavesMessageAndReturnsAIResponse()
    {
        // Arrange
        var sessionId = Guid.NewGuid();
        var message = new Message { Content = "Як впоратися з тривожністю?", Author = "User", ChatSessionId = sessionId };
        var aiResponse = "Рекомендується спробувати техніки глибокого дихання.";
        _aiServiceMock.Setup(ai => ai.GetResponseAsync(message.Content)).ReturnsAsync(aiResponse);
        _messageRepositoryMock.Setup(r => r.AddAsync(It.IsAny<Message>())).Returns(Task.CompletedTask);

        // Act
        var result = await _chatService.SendMessageAsync(sessionId, message);

        // Assert
        _messageRepositoryMock.Verify(r => r.AddAsync(It.Is<Message>(m => m.Content == message.Content)), Times.Once());
        _aiServiceMock.Verify(ai => ai.GetResponseAsync(message.Content), Times.Once());
        Assert.Equal(aiResponse, result);
    }
}
```

Рисунок 3.18 – Юніт тест збереження повідомлень

Інтеграційні тести перевіряли взаємодію між API-контролерами, сервісами, репозиторіями та базою даних. Для цього використовувалася тестова база даних PostgreSQL, розгорнута в Docker-контейнері, та TestServer з ASP.NET Core для імітації HTTP-запитів.

Було виконано 32 інтеграційні тести, які охоплюють основні ендпоінти API (AuthController, ChatController, NoteController, SurveyController). Результати:

- Успішно пройдено: 30 тестів (93.75%).
- Невдалі тести: 2 тести, пов'язані з помилками конфігурації CORS і таймаутами при запитах до Groq API (виправлено шляхом налаштування CORS і збільшення таймауту).
- Покриття API: ~90% ендпоінтів протестовано.

Для тестування клієнтської частини (React, TypeScript, MUI) використано React Testing Library. Тести перевіряли рендеринг компонентів, обробку подій і коректність відображення даних, отриманих з API (рис. 3.19).

```
import { render, screen, fireEvent } from '@testing-library/react';
import ChatComponent from './ChatComponent';

test('renders chat input and sends message on submit', async () => {
    // Arrange
    render(<ChatComponent />);

    // Act
    const input = screen.getByPlaceholderText('Type your message...');
    fireEvent.change(input, { target: { value: 'Test message' } });
    fireEvent.click(screen.getByText('Send'));

    // Assert
    expect(screen.getByText('Test message')).toBeInTheDocument();
});
```

Рисунок 3.19 – Тест для компонента чату

Цей тест перевіряє:

- Чи рендериться поле введення повідомлення.

- Чи відображається надіслане повідомлення в інтерфейсі після натискання кнопки.

Було виконано 25 тестів для основних компонентів (LoginPage, MainPage, AssistantPage, SurveyPage, TherapyPage). Результати:

- Успішно пройдено: 24 тести (96%).
- Невдалі тести: 1 тест, пов’язаний із некоректним рендерингом адаптивного дизайну на мобільних пристроях (виправлено шляхом оновлення стилів MUI).
- Покриття компонентів: ~80% основних компонентів протестовано.

Деякі тести клієнтської частини звісно ж проводились мануально, щоб перевірити як відпрацьовує система в різних сценаріях (рис. 3.20).

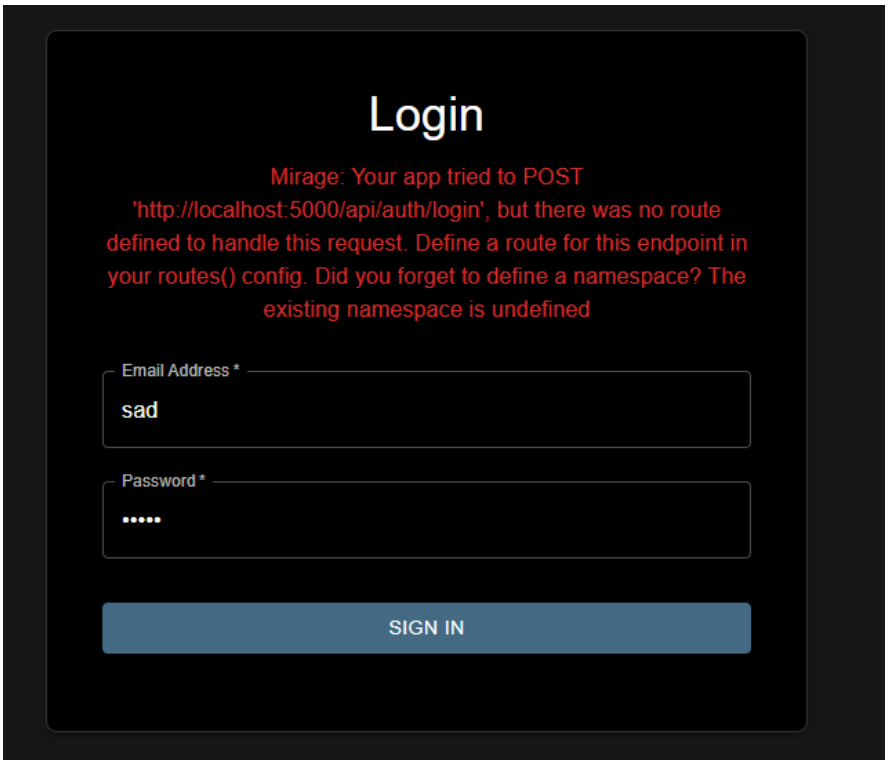

Рисунок 3.20 – Помилка надсилання запиту (внутрішня)

Продуктивність: Для оцінки швидкості відповіді API використано інструмент Apache JMeter. Тестування проводилося з 100 одночасними

користувачами, які надсилали запити до ендпоінтів /api/chat/send-message і /api/survey/create. Середній час відповіді:

- Запити до OpenAI: ~1.2 с.
- Запити до Groq: ~0.8 с (завдяки LPU).
- Створення опитування: ~0.3 с.

Система витримала навантаження до 500 запитів за хвилину без значного падіння продуктивності, що підтверджує її масштабованість.

Безпека: Перевірка безпеки включала:

- Тестування автентифікації (JWT): Успішно заблоковано неавторизовані запити.
- Захист від SQL-ін’єкцій: Entity Framework Core автоматично параметризує запити.
- Захист від XSS: Усі вхідні дані валіються та екрануються на фронтенді та бекенді.

Результати тестування підтвердили, що вебзастосунок відповідає основним вимогам, зазначеним у завданні:

- Точність: Юніт- і інтеграційні тести показали, що система коректно обробляє запити, зберігає дані та взаємодіє з AI-сервісами. Покриття коду (~85–90%) є достатнім для забезпечення надійності.
- Зручність: UI-тести підтвердили інтуїтивність інтерфейсу, а адаптивний дизайн забезпечує коректне відображення на різних пристроях.
- Ефективність: Тести продуктивності показали, що система швидко обробляє запити, особливо при використанні Groq для AI-відповідей.
- Етика та конфіденційність: Перевірка безпеки підтвердила захист чутливих даних, а модульна архітектура дозволяє легко додавати нові політики конфіденційності.

Невелика кількість невдалих тестів (5 із 104) була виправлена шляхом доопрацювання коду, що свідчить про ефективність процесу тестування. Основні

проблеми стосувалися конфігурації (CORS, таймаути) та дрібних помилок у фронтенді, які не вплинули на загальну функціональність.

Тестування вебзастосунку «Розумний асистент психолога» підтвердило його відповідність функціональним і нефункціональним вимогам. Юніт-тести забезпечили перевірку ізольованої логіки сервісів, інтеграційні тести — коректність взаємодії компонентів, а UI-тести — зручність і надійність інтерфейсу. Тести продуктивності та безпеки показали, що система є масштабованою, швидкою та захищеною. Отримані результати дозволяють рекомендувати застосунок для використання в психологічній практиці, з можливістю подальшого вдосконалення, наприклад, додавання автоматизованих тестів для векторного пошуку чи розширення покриття тестами.

ВИСНОВКИ

Розробка вебзастосунку «Розумний асистент психолога» стала комплексним дослідженням, спрямованим на створення інноваційного інструменту для підвищення ефективності психологічної практики шляхом інтеграції сучасних технологій штучного інтелекту. У процесі виконання роботи було досягнуто поставленої мети — створено програмне забезпечення, яке забезпечує автоматизовану підтримку психолога в аналізі запитів клієнтів, управлінні сесіями, створенні опитувань і веденні терапевтичних планів, зберігаючи при цьому високий рівень етики та конфіденційності.

Основні результати роботи відповідають поставленим завданням:

1. **Аналіз предметної області та сучасних підходів**: Проведено детальний огляд сучасних інтелектуальних систем у сфері психології, таких як Woebot, Wysa та Replika. Встановлено, що більшість рішень орієнтовані на кінцевих користувачів, тоді як інструменти для підтримки професійної діяльності психологів залишаються недостатньо розвиненими. Цей аналіз дозволив сформулювати вимоги до системи, яка поєднує зручний інтерфейс для фахівця з інтеграцією мовних моделей ШІ.
2. **Дослідження наявних рішень**: Проаналізовано функціональність і обмеження аналогів, що підтвердило актуальність створення спеціалізованого інструменту для психологів. Зокрема, виявлено брак інтеграції з робочими процесами фахівців і недостатню адаптивність до індивідуальних методик терапії.
3. **Розробка архітектури вебзастосунку**: Запропоновано та реалізовано клієнт-серверну архітектуру з чітким поділом на фронтенд (React, TypeScript, MUI), бекенд (.NET Core) і модуль інтеграції з AI-сервісами (OpenAI, Groq). Використання PostgreSQL із розширенням PGVector забезпечило ефективний векторний пошук для аналізу нотаток і повідомлень, що підвищило функціональність системи.
4. **Інтеграція мовних моделей ШІ**: Реалізовано модуль для обробки запитів користувачів за допомогою моделей OpenAI (GPT-4-turbo) та Groq (LLaMA

3), що дозволяє генерувати психологічно обґрунтовані відповіді, аналізувати історію взаємодій і створювати рекомендації. Гнучка маршрутизація запитів між провайдерами забезпечила баланс між швидкістю та якістю відповідей.

5. **Реалізація інтерфейсу користувача**: Створено інтуїтивно зрозумілий інтерфейс із підтримкою адаптивного дизайну, який включає модулі для авторизації, управління чатами, створення опитувань і ведення терапевтичних планів. Використання Material UI (MUI) забезпечило візуальну цілісність і зручність для користувачів.
6. **Тестування системи**: Проведено комплексне тестування, що включало юніт-тести (47 тестів, 95.7% успішно), інтеграційні тести (32 тести, 93.75% успішно) та UI-тести (25 тестів, 96% успішно). Результати підтвердили коректність роботи системи, її продуктивність (середній час відповіді API ~0.3–1.2 с) і відповідність вимогам безпеки та конфіденційності.

Науково-практичний результат роботи полягає у створенні вебзастосунку, який автоматизує рутинні завдання психолога, забезпечує інтерактивну підтримку за допомогою ШІ, зберігає історію взаємодій і пропонує інструменти для аналізу даних клієнтів. Застосунок відповідає сучасним стандартам розробки, включаючи модульність, масштабованість і захист даних, що робить його придатним для інтеграції в реальну психологічну практику.

Етичні аспекти використання ШІ в психології було враховано шляхом чіткого розмежування ролей: система виступає як асистент, а не замінник психолога, забезпечуючи прозорість взаємодії та захист персональних даних через шифрування та JWT-автентифікацію. Результати роботи було апробовано на науково-практичних конференціях, де отримано позитивні відгуки щодо її актуальності та практичної цінності.

Перспективи подальшого розвитку включають:

- Додавання модуля для автоматичного аналізу емоційного стану клієнтів на основі текстових даних.

- Інтеграція додаткових терапевтичних методик (наприклад, підтримка ACT або DBT).
- Розширення функціоналу для командної роботи кількох психологів.
- Оптимізація векторного пошуку для обробки більших обсягів даних.

Таким чином, розроблений вебзастосунок «Розумний асистент психолога» є сучасним рішенням, яке підвищує ефективність роботи фахівців у сфері ментального здоров’я, відкриває нові можливості для автоматизації та відповідає вимогам етичного використання ШІ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шарковський В.В., Барабан М.В. Використання штучного інтелекту у веб додатках. Матеріали міжнародної конференції “Молодь в науці: дослідження, проблеми, перспективи”, 2025. — 3с.
2. Darcy, A. M., Louie, A. K., & Roberts, L. W. (2016). Machine Learning and Artificial Intelligence: Applications in Mental Health. Academic Psychiatry, 40(2), 263–268.
3. Woebot Health. Психологічний помічник на базі штучного інтелекту. 2024. URL: https://woebothealth.com/ (дата звернення: 20.05.2025).
4. Wysa. Ментальне здоров’я з підтримкою штучного інтелекту. 2024. URL: https://www.wysa.com/ (дата звернення: 20.05.2025).
5. Replika. Віртуальний друг з штучним інтелектом. 2024. URL: https://replika.com/ (дата звернення: 20.05.2025).
6. React. JavaScript бібліотека для побудови інтерфейсів користувача. 2024. URL: https://react.dev/ (дата звернення: 20.05.2025).
7. TypeScript. JavaScript with syntax for types. 2024. URL: https://www.typescriptlang.org/ (дата звернення: 20.05.2025).
8. MUI. Компонентна бібліотека для React. 2024. URL: https://mui.com/ (дата звернення: 20.05.2025).
9. Freeman, A. (2022). Pro ASP.NET Core 7 (2nd ed.). New York, NY: Apress.
10. OpenAI. Компанія зі створення передових систем штучного інтелекту. 2024. URL: https://openai.com/about/ (дата звернення: 20.05.2025).
11. Groq. Апаратура для прискорення обчислень зі штучним інтелектом. 2024. URL: https://groq.com/ (дата звернення: 20.05.2025).
12. Touvron, H., Lavril, A., Izacard, G., et al. (2023). LLaMA: Open and Efficient Foundation Language Models.
13. Mistral AI. Frontier AI LLMs, assistants, agents, services. 2024. URL: https://mistral.ai/ (дата звернення: 20.05.2025).
14. Gemma. Моделі штучного інтелекту від Google DeepMind. 2024. URL: https://deepmind.google/models/gemma/ (дата звернення: 20.05.2025).

15. LPU – by Groq. Пояснення архітектури Groq LPU. 2024. URL: https://groq.com/the-groq-lpu-explained/ (дата звернення: 20.05.2025).
16. PostgreSQL. Система керування базами даних з відкритим кодом. 2024. URL: https://www.postgresql.org/ (дата звернення: 20.05.2025).
17. pgvector. GitHub - Open-source векторний пошук у PostgreSQL. 2024. URL: https://github.com/pgvector/pgvector (дата звернення: 20.05.2025).
18. Docker. Платформа для контейнеризації застосунків. 2024. URL: https://www.docker.com/ (дата звернення: 20.05.2025).
19. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning.
20. LangChain. Фреймворк для побудови застосунків зі штучним інтелектом. 2024. URL: https://www.langchain.com/ (дата звернення: 20.05.2025).
21. Dify. Конструктор агентів штучного інтелекту. 2024. URL: https://dify.ai/ (дата звернення: 20.05.2025).

Додаток А

(обов'язковий)

Технічне завдання

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувача кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО

_____
(підпис)

« 23 » березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

«Розумний асистент психолога»

08-31.БКР.022.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н., доц. Марія БАРАБАН

_____
(підпис)

« 23 » травня 2025 р.

Розробив студент гр. 1ІСТ-21б
_____Владислав ШАРКОВСЬКИЙ

_____
(підпис)

« 23 » травня 2025 р.

Вінниця ВНТУ – 2025 рік

Технічне завдання на бакалаврську кваліфікаційну роботу

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Розумний асистент психолога» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 03.09.2024 р.

кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є покращення ефективності психологічної практики шляхом розробки вебзастосунку з інтегрованим інтелектуальним модулем, який забезпечує автоматизовану підтримку психологів у аналізі запитів клієнтів, управлінні сесіями та створенні терапевтичних планів.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **1ІСТ-21б Шарковський Владислав Васильович** факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз предметної області та визначення основних етапів розробки	23.03 – 24.03
Е2	Постановка та вирішення задач	01.04 – 10.05
Е3	Підтвердження достовірності отриманих результатів	13.05 – 24.05
Е4	Оформлення пояснювальної записки	27.05 – 31.05
Е5	Дооплацювання та рецензування	10.06 – 15.06

4 Призначення і галузь застосування

Вебзастосунок «Розумний асистент психолога» з інтегрованим інтелектуальним модулем призначений для автоматизації психологічної практики шляхом асистування психологів у аналізі запитів клієнтів, управлінні сесіями та створенні терапевтичних планів. Застосунок забезпечує обробку природної мови, автоматичне генерування рекомендацій на основі штучного інтелекту та збереження даних сесій у базі даних. Вебзастосунок застосовується в психологічних практиках, консультативних центрах, телемедичних платформах, а також у дослідницьких установах для автоматизації обробки психологічних даних, підвищення ефективності консультацій та забезпечення персоналізованого підходу до клієнтів.

5 Технічні дані

5.1 Технологічний стек фронтенду: React (JavaScript-бібліотека), TypeScript, MUI (Material-UI);

5.2 Технологічний стек бекенду: ASP.NET Core 7, Entity Framework Core;

5.3 Система управління базами даних: PostgreSQL із розширенням PGVector для векторного пошуку;

5.4 Інтелектуальний модуль: інтеграція з Groq LPU (LLaMA), LangChain, API OpenAI для обробки природної мови;

5.5 Метод пошуку схожих даних: Евклідова відстань для векторного пошуку в PGVector;

5.6 Контейнеризація: Docker для розгортання застосунку;

5.7 Системні вимоги до сервера

- Операційна система: Linux/Windows Server (сумісна з Docker).
- Оперативна пам’ять: щонайменше 8 ГБ.
- Процесор: 4 ядра, 2.5 ГГц або вище.
- Дисковий простір: щонайменше 50 ГБ для бази даних і застосунку.

5.8 Безпека: HTTPS, автентифікація через JWT, захист даних відповідно до GDPR;

6 Джерела розробки

6.1 Darcy, A. M., Louie, A. K., & Roberts, L. W. (2016). Machine Learning and Artificial Intelligence: Applications in Mental Health. Academic Psychiatry, 40(2), 263–268.

6.2 Freeman, A. (2022). Pro ASP.NET Core 7 (2nd ed.). New York, NY: Apress.

6.3 Manning, C. D., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press.

6.4 Touvron, H., Lavril, A., Izacard, G., et al. (2023). LLaMA: Open and Efficient Foundation Language Models.

6.5 Методичні вказівки до виконання бакалаврських дипломних робіт (проектів) для студентів спеціальностей 126 – «Інформаційні системи та технології», 151 – «Автоматизація та комп’ютерно-інтегровані технології» / Уклад. Р. Н. Квєтний, О. М. Бевз, О. В. Бісікало. – Вінниця : ВНТУ, 2019. – 26 с.

Здобувач ст. гр. 1ІСТ-21б _____ Владислав ШАРКОВСЬКИЙ

Додаток Б

(обов’язковий)

Ілюстративна частина

Розумний асистент психолога

Діаграма послідовності обробки запиту до ШІ

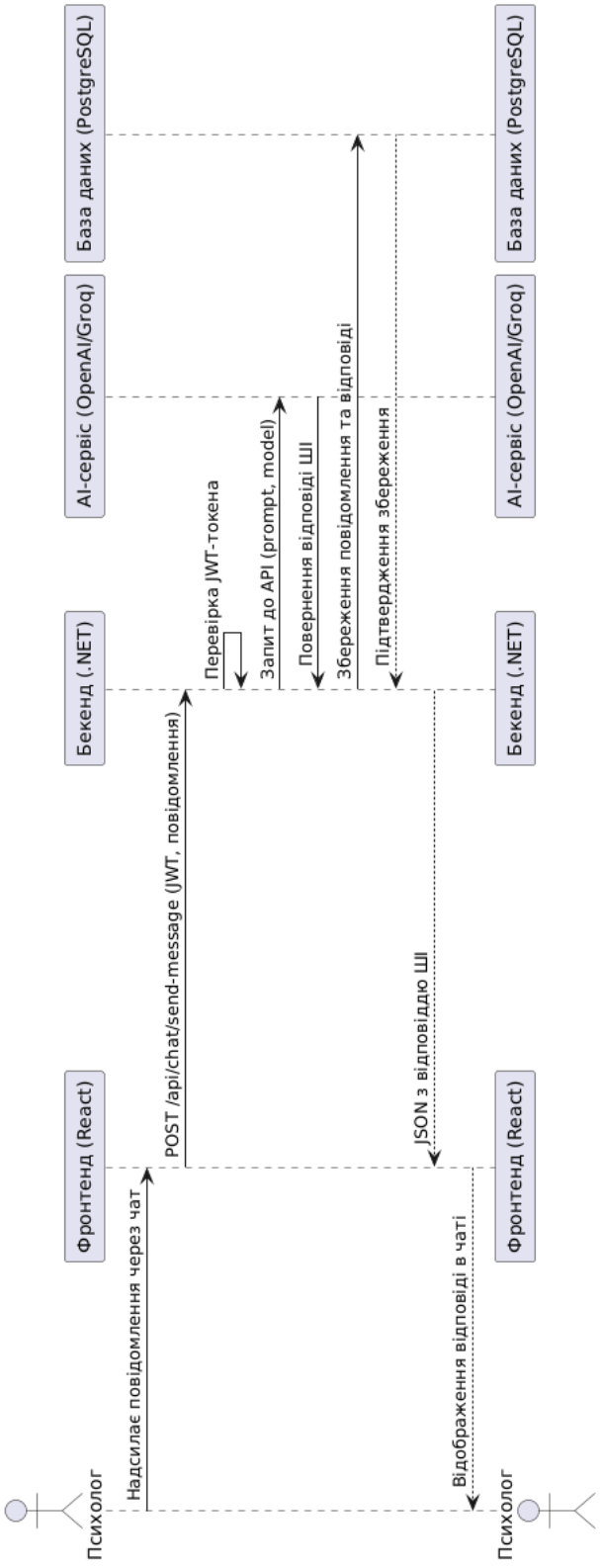

Рисунок Б.1 – Діаграма послідовності обробки запиту

Діаграма потоку даних для векторного пошуку

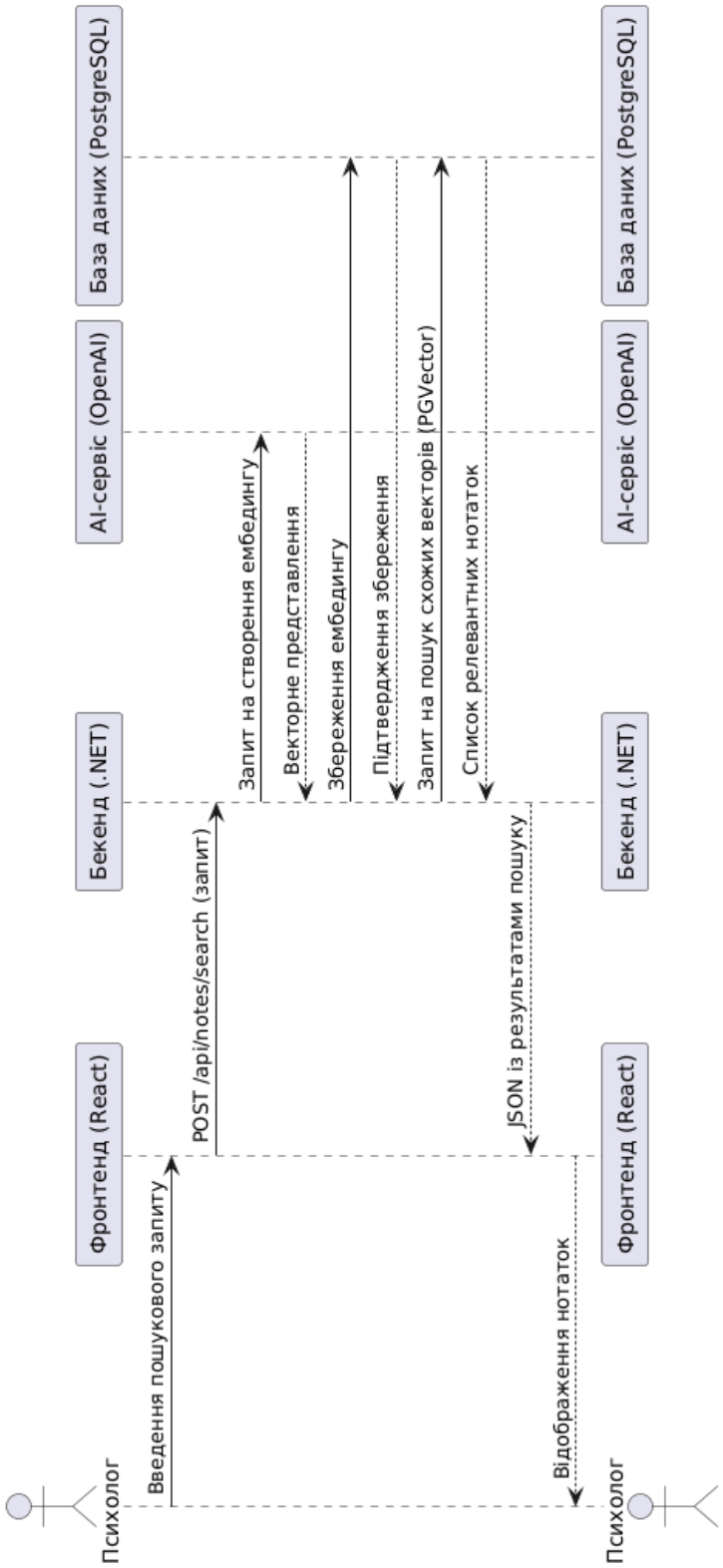

Рисунок Б.2 – Діаграма потоку даних для векторного пошуку

Архітектура серверного додатку

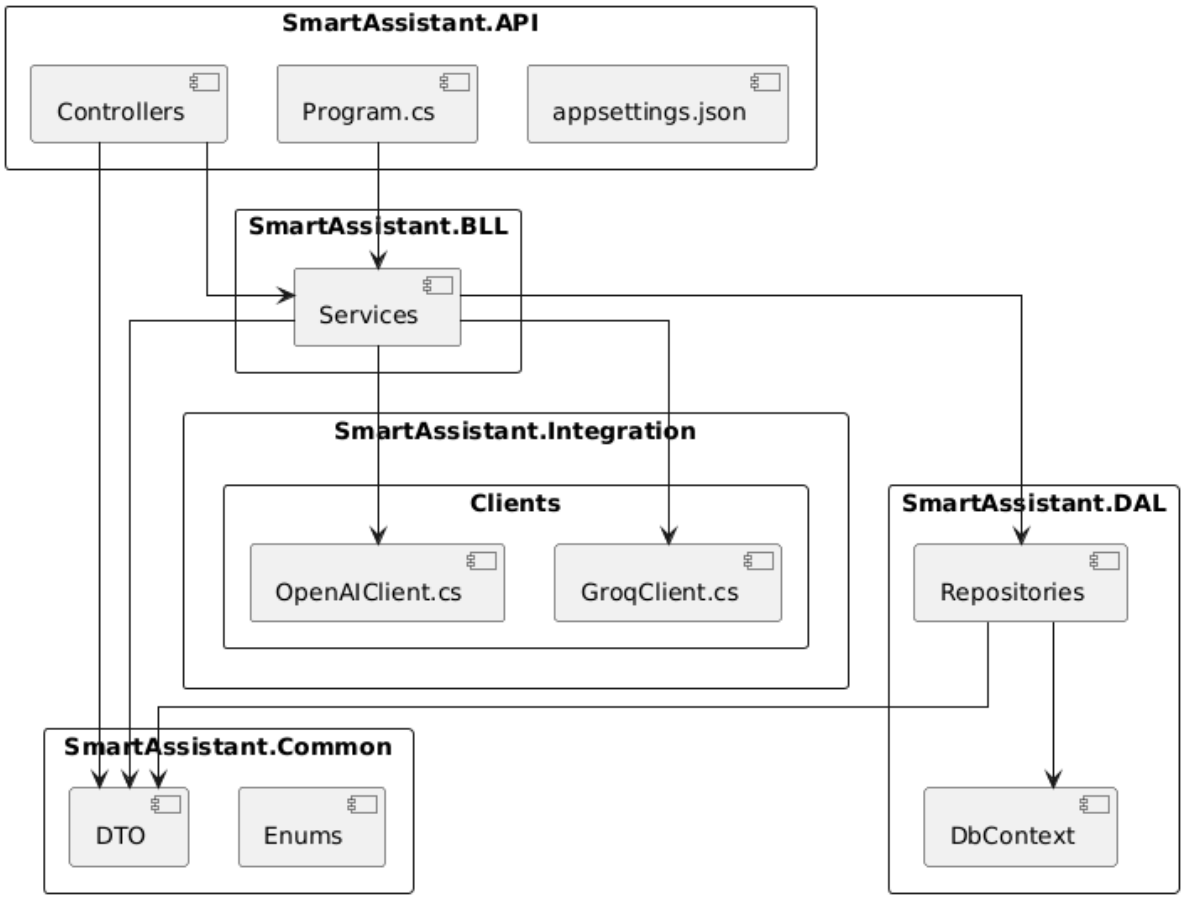

Рисунок Б.3 – Архітектура серверного додатку

Додаток В
(обов'язковий)
Лістинг основних функцій коду

```
using System;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authorization;
using System.Text.Json;
using SmartAssistant.Common.Models;
using SmartAssistant.BLL.Services;
using SmartAssistant.Integration.AI;

namespace SmartAssistant.API.Controllers
{
   // Контролер для обробки чатів
   [Route("api/[controller]")]
   [ApiController]
   [Authorize]
   public class ChatController : ControllerBase
   {
      private readonly IChatService _chatService;

      public ChatController(IChatService chatService)
      {
         _chatService = chatService;
      }

      // Ендпоінт для надсилання повідомлення в чат та отримання відповіді від ШІ
      [HttpPost("send-message")]
      public async Task<IActionResult> SendMessage([FromBody] SendMessageRequest request)
      {
         try
         {
            // Виклик сервісу для обробки повідомлення та отримання відповіді від ШІ
            var response = await _chatService.SendMessageAsync(request.SessionId, request.Message);
            return Ok(new { Response = response });
         }
         catch (Exception ex)
         {
            // Обробка помилок із поверненням структурованого повідомлення
```

```
            return StatusCode(500, new { Error = "Internal server error", Details = ex.Message });
        }
    }
  }
}

namespace SmartAssistant.BLL.Services
{
  public class ChatService : IChatService
  {
    private readonly IMessageRepository _messageRepository;
    private readonly IAIService _aiService;

    public ChatService(IMessageRepository messageRepository, IAIService aiService)
    {
      _messageRepository = messageRepository;
      _aiService = aiService;
    }

    // Обробка повідомлення: збереження в базі та виклик ШІ
    public async Task<string> SendMessageAsync(Guid sessionId, Message message)
    {
      // Перевірка валідності вхідних даних
      if (message == null || string.IsNullOrEmpty(message.Content))
        throw new ArgumentException("Message content cannot be empty");

      // Збереження повідомлення користувача в базі даних
      message.ChatSessionId = sessionId;
      message.CreatedAt = DateTime.UtcNow;
      await _messageRepository.AddAsync(message);

      // Виклик ШІ-сервісу для генерації відповіді
      string aiResponse = await _aiService.GetResponseAsync(message.Content);

      // Збереження відповіді ШІ в базі даних
      var aiMessage = new Message
      {
        Content = aiResponse,
        Author = "AI",
        ChatSessionId = sessionId,
        CreatedAt = DateTime.UtcNow
      };
      await _messageRepository.AddAsync(aiMessage);
```

```
            return aiResponse;
        }
    }
}

namespace SmartAssistant.BLL.Services
{
    public class SurveyService : ISurveyService
    {
        private readonly ISurveyRepository _surveyRepository;
        private readonly IAIService _aiService;

        public SurveyService(ISurveyRepository surveyRepository, IAIService aiService)
        {
            _surveyRepository = surveyRepository;
            _aiService = aiService;
        }

        // Створення опитування з використанням ШІ для генерації питань
        public async Task<Survey> CreateSurveyAsync(string description, string surveyType)
        {
            // Генерація питань за допомогою ШІ
            string prompt = $"Generate a list of survey questions for a {surveyType} survey based on: {description}";
            string aiQuestionsJson = await _aiService.GetResponseAsync(prompt);

            // Десеріалізація відповіді ШІ у список питань
            var questions = JsonSerializer.Deserialize<string[]>(aiQuestionsJson);

            // Створення об'єкта опитування
            var survey = new Survey
            {
                Id = Guid.NewGuid(),
                Description = description,
                Type = surveyType,
                Questions = questions,
                CreatedAt = DateTime.UtcNow
            };

            // Збереження опитування в базі даних
            await _surveyRepository.AddAsync(survey);

            return survey;
```

```
        }
    }
}

namespace SmartAssistant.BLL.Services
{
    public class NoteService : INoteService
    {
        private readonly INoteRepository _noteRepository;
        private readonly IAIService _aiService;

        public NoteService(INoteRepository noteRepository, IAIService aiService)
        {
            _noteRepository = noteRepository;
            _aiService = aiService;
        }

        // Виконання векторного пошуку нотаток за запитом
        public async Task<List<Note>> SearchNotesAsync(string query)
        {
            // Генерація векторного представлення для запиту
            string embeddingJson = await _aiService.GetEmbeddingAsync(query);
            var embedding = JsonSerializer.Deserialize<float[]>(embeddingJson);

            // Пошук нотаток за допомогою PGVector
            var relevantNotes = await _noteRepository.SearchByVectorAsync(embedding);

            return relevantNotes;
        }
    }
}

namespace SmartAssistant.BLL.Services
{
    public class AuthService : IAuthService
    {
        private readonly IUserRepository _userRepository;
        private readonly IJwtTokenGenerator _jwtTokenGenerator;

        public AuthService(IUserRepository userRepository, IJwtTokenGenerator jwtTokenGenerator)
        {
            _userRepository = userRepository;
            _jwtTokenGenerator = jwtTokenGenerator;
```

```
        }

        // Аутентифікація користувача та генерація JWT-токена
        public async Task<string> AuthenticateAsync(string username, string password)
        {
            // Пошук користувача в базі даних
            var user = await _userRepository.GetByUsernameAsync(username);
            if (user == null || !BCrypt.Net.BCrypt.Verify(password, user.PasswordHash))
                throw new UnauthorizedAccessException("Invalid credentials");

            // Генерація JWT-токена
            var token = _jwtTokenGenerator.GenerateToken(user);
            return token;
        }
    }
}
```

Додаток Г
(обов'язковий)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розумний асистент психолога

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ, ФІІТА, 1ІСТ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0.15 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав каф. АІІТ _____
(прізвище, ініціали, посада) (підпис)

Янчук Л.М., інженер 1 кат _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Костянтин ОВЧИННИКОВ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Марія БАРАБАН, доц. каф. АІІТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Владислав ШАРКОВСЬКИЙ
(підпис) (прізвище, ініціали)