Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ІНФОРМАЦІЙНА СИСТЕМА ПСИХОЛОГІЧНОЇ РЕАБІЛІТАЦІЇ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент 2 курсу, групи 2КІ-23м
Спеціальності 123 — Комп'ютерна інженерія
_____ Фічковський Д. А.
Керівник: к.т.н., доц. каф. ОТ
_____ Добровольська Н. В.
«____» _____ 2024 р.
Опонент: к.т.н., доц. каф. ПІ
_____ Коваленко О. О.
«____» _____ 2024 р.

Допущено до захисту
Завідувач кафедри ОТ
_____ д.т.н., проф Азаров О. Д.
«____» _____ 2024 р.

ВНТУ 2024

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційниї технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Галузь знань — 12 Інформаційні технології

Освітній рівень — магістр

Спеціальність — 123 Комп'ютерна інженерія

Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

_____ проф., д.т.н. О. Д. Азаров

«_18_» _вересня_ 2024 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

Фічковському Дмитру Анатолійовичу

1 Тема роботи: «Інформаційна система психологічної реабілітації з використанням штучного інтелекту», керівник роботи к.т.н., доц. каф. ОТ Добровольська Н. В., затверджені наказом вищого навчального закладу від 17.09.24 року № 310.

2 Строк подання студентом роботи 17.12.24.

3 Вихідні дані до роботи: дані для навчання моделі, дані для рекомендацій, користувацькі дані, моделі машинного навчання.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, дослідження предметної області, аналіз програмних рішень та підходів для психологічної реабілітації, моделювання та проектування системи, інтеграція зі штучним інтелектом, програмна реалізація та тестування системи.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): діаграма прецедентів системи, діаграма діяльності системи, графічна структура бази даних.

6 Консультанти розділів роботи представлено в табл. 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 - 4	Добровольська Н. В. к.т.н., доц. каф. ОТ		
5	Небава М. І., к.е.н., проф. каф. ЕП і ВМ	Небава	Небава

7 Дата видачі завдання 18.09.2024.

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Примітка
1	Постановка задачі	18.09.24	виконано
2	Аналіз існуючих рішень	18.09-20.09.24	виконано
3	Аналіз та вибір технологій розробки	23.09-25.09.24	виконано
4	Моделювання та проектування системи	26.09-02.10.24	виконано
5	Розробка моделі	03.10-09.10.24	виконано
6	Розробка системи	10.10-01.11.24	виконано
7	Тестування системи	04.11-08.11-24	виконано
8	Розрахунок економічної частини	09.11-13.11.24	виконано
9	Оформлення пояснювальної записки	14.11-22.11.24	виконано
10	Аналіз виконання роботи, висновки, додатки	27.11-29.11.24	виконано
11	Перевірка якості виконання магістерської роботи та усунення недоліків	02.12-09.12.24	виконано

Студент _____ Фічковський Д. А.

Керівник роботи _____ Добровольська Н. В.

АНОТАЦІЯ

УДК 004.8

Фічковський Д. А. Інформаційна система психологічної реабілітації з використанням штучного інтелекту. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, Вінниця: ВНТУ, 2024. Загальний обсяг роботи 108 сторінок, 26 рисунків, 18 таблиць, 7 лістиннгів.

У магістерській кваліфікаційній роботі розроблено інформаційну систему психологічної реабілітації з використанням штучного інтелекту, яка дає змогу дає змогу ефективно виявляти психологічні проблеми, надавати рекомендації щодо їх вирішення та забезпечувати взаємодію користувачів із фахівцями. У загальній частині роботи проведено аналіз предметної області, аналіз існуючих систем де було виявлено їх переваги та недоліки, та технологій розробки. Описано методи створення системи та результати. У роботі проведено навчання моделі штучного інтелекту, та подальше її використання в системі що використовує клієнт-серверну архітектуру. Були проведені економічні розрахунки для визначення доцільності та конкурентоспроможності.

Ключові слова: інформаційна система, психологічна реабілітація, штучний інтелект, аналіз емоційного стану, веб-додаток.

ABSTRACT

Fichkovskyi D. A. Information system of psychological rehabilitation using artificial intelligence. Master's qualification work in specialty 123 - Computer Engineering, Vinnytsia: VNTU, 2024. The total amount of work is 108 pages, 26 figures, 18 tables, 7 listings.

In the master's qualification work, an information system of psychological rehabilitation using artificial intelligence was developed, which allows to effectively identify psychological problems, provide recommendations for their solution and ensure the interaction of users with specialists. In the general part of the paper, the subject area is analyzed, existing systems are analyzed, their advantages and disadvantages are identified, and development technologies are described. The methods of creating the system and the results are described. The paper includes training of the AI model and its further use in a system using client-server architecture. Economic calculations were made to determine the feasibility and competitiveness.

Keywords: information system, psychological rehabilitation, artificial intelligence, analysis of emotional state, web application.

ЗМІСТ

ВСТУП

В наш час, етап розвитку інформаційних технологій характеризується активним використанням цифрових платформ для підтримки різноманітних аспектів людського життя. Одним з аспектів людського життя є психологічне здоров'я, а вирішення проблем психологічного здоров'я позитивно вплине на соціальні, економічні та суспільні результати [1]. Створення інформаційної системи, для психологічної реабілітації, здатно не лише підвищити ефективність надання психологічної допомоги, але й зробити її більш доступною для людей.

Актуальність розробки подібних систем зумовлена зростанням кількості людей, які потребують психологічної підтримки, але не завжди можуть отримати її своєчасно, наприклад, через обмежений доступ до фахівців. Інформаційна система психологічної реабілітації може надавати індивідуальні рекомендації, проводити попередній аналіз психоемоційного стану та допомагати у вирішенні особистих проблем не великої важкості, завдяки інтеграції технологій штучного інтелекту.

Штучний інтелект стрімко прогресує у різних галузях, оскільки дозволяє ефективно оптимізувати й автоматизувати будь-які процеси. Очікується що з 2023 по 2030 рр, впровадження технологій штучного інтелекту щороку буде збільшуватись на 37,3% [2]. Галузь психології не є виключенням, згідно статистики понад 14 млн українців потребують психологічної допомоги [3].

Мета дослідження — створення системи, з високою ефективностю та швидкістю виявлення психологічних проблем, з використанням штучного інтелекту, що надає рекомендації для покращення психологічного стану, та забезпечує візуалізацію прогресу.

Об'єкт дослідження — загальновідомі методи психологічної реабілітації і підтримки, та процеси їх впровадження в інформаційні системи, включаючи використання технологій штучного інтелекту.

Предмет дослідження — конкретні програмні засоби та технології, що використовуються для розробки інформаційної системи, для аналізу психологічного стану та надання рекомендацій.

Завдання дослідження:

— провести аналіз сучасних інформаційних систем психологічної реабілітації;

— розробити архітектуру інформаційної системи для надання психологічної допомоги;

— створити алгоритми для попередньої діагностики психічного стану користувачів;

— навчити модель штучного інтелекту для аналізу текстових запитів користувачів;

— здійснити програмну реалізацію системи;

— провести тестування розробленої інформаційної системи;

— оцінити ефективність та практичність використання системи.

Наукова новизна отриманих результатів магістерської роботи полягає в покращенні якості та швидкості виялення психологічних проблем, використовуючи існуючі методики, шляхом впровадження штучного інтелекту, що дозволяє людям з нескладним діагнозом самостійно справитись з проблемою, а також полегшити процес роботи спеціаліста з клієнтом.

Практичне значення роботи полягає у створенні програмного забезпечення, що характеризується швидкою обробкою даних, зручністю доступу та низькою вартістю, що робить його придатним для використання в сфері психологічної реабілітації користувачів.

Апробація. За результатами дослідження опубліковано тези доповіді: «Інтеграція штучного інтелекту в інформаційні системи для психологічної реабілітації», на Всеукраїнську науково-практичну інтернет-конференцію «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [4].

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ, АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ ТА ПІДХОДІВ ДЛЯ ПСИХОЛОГІЧНОЇ РЕАБІЛІТАЦІЇ

1.1 Розгляд стану проблеми

Проблема доступу до психологічної реабілітації є актуальною для багатьох людей у сучасному світі. Традиційні підходи до надання психологічної допомоги, такі як особисті консультації з психологами чи психотерапевтами, вимагають безпосередньої присутності клієнта та фахівця. Географічна віддаленість, високі витрати на послуги, брак фахівців у певних регіонах, є основними чинниками, що ускладнюють доступ до терапії.

Оскільки, психологічний стан людини має велике значення, так як часто впливає на її фізичне здоров'я, у сучасному суспільстві, де люди постійно стикаються зі стресовими ситуаціями, застосунок на основі штучного інтелекту для реабілітації і підтримки ментального здоров'я стане важливим засобом у боротьбі з психоемоційним вигоранням, внутрішніми конфліктами, соціальною ізоляцією та їхніми негативними наслідками.

Однією з ключових проблем є географічна доступність послуг. Люди, які проживають у віддалених сільських районах або країнах, де доступ до медичних послуг є обмеженим, часто не мають можливості регулярно звертатися за допомогою до фахівців через відсутність клінік або спеціалізованих центрів в межах доступної відстані.

Ще однією проблемою є вартість послуг. Консультації з кваліфікованими психологами та психотерапевтами можуть бути досить дорогими, що робить їх недоступними для багатьох людей із низьким доходом. Без належного фінансування багато людей залишаються без необхідної допомоги, що може призводити до загострення психічних проблем.

Суттєвою перепоною є також страх осуду та упередження з боку суспільства. Це призводить до того, що люди не звертаються за допомогою на ранніх етапах розвитку психічних проблем, що ускладнює подальше лікування.

Відсутність належної інтеграції технологій у галузь психічного здоров'я також є проблемою. Хоча існують додатки та платформи для підтримки психічного здоров’я, більшість із них мають обмежену функціональність або не забезпечують достатньо якісну підтримку.

У зв'язку з цим виникає потреба у створенні інформаційної системи для автоматизації процесів психологічної реабілітації, яка б забезпечувала зручний, доступний та ефективний інструмент для користувачів. Така система може використовувати штучний інтелект для надання первинної консультації, моніторингу емоційного стану користувачів та рекомендувати подальші дії, зокрема пропонувати онлайн-консультації з фахівцями або надавати рекомендації щодо самодопомоги.

Важливо визначити аудиторію, яка буде користуватися системою. В цілому, це можуть бути люди різних категорій, серед яких:

— звичайні користувачі — люди, які відчувають стрес, тривожність, депресію або інші психоемоційні проблеми та потребують підтримки в повсякденному житті, вони можуть використовувати систему для самодіагностики, отримання рекомендацій щодо самодопомоги та консультацій з фахівцями;

— працівники компаній — система може бути інтегрована в корпоративне середовище для підтримки психологічного здоров’я працівників, знижуючи рівень професійного вигорання та підвищуючи продуктивність, користувачі можуть анонімно звертатися за допомогою, отримуючи поради та консультації;

— студенти — молоді люди, які стикаються з емоційним навантаженням через навчання, соціальні проблеми або невизначеність майбутнього, система може допомогти їм підтримувати психічний баланс та забезпечити емоційну стабільність;

— психологи та психотерапевти — фахівці можуть використовувати систему для надання онлайн-консультацій, отримання інформації про емоційний стан пацієнтів та контролю за динамікою їхнього психологічного здоров'я;

— люди з обмеженим доступом до фахівців — система стане особливо корисною для мешканців віддалених або малонаселених регіонів, де доступ до кваліфікованої допомоги обмежений;

— люди, які бажають зберегти анонімність — користувачі, які не хочуть звертатися за традиційною допомогою через страх осуду або стигматизації, можуть анонімно користуватися системою для отримання необхідної психологічної підтримки.

Другим важливим кроком є визначення основних функцій системи. Це може включати надання дистанційних консультацій, автоматичну діагностику психологічного стану, рекомендації щодо терапії, а також інтерактивні техніки для самопідтримки (наприклад, медитація, тренування зниження стресу). Важливим аспектом є надання конфіденційності та безпеки даних користувачів.

Також необхідно визначити технологічні рішення, які будуть використовуватися для розробки інформаційної системи. Це може включати вибір мови програмування, фреймворків, бази даних та сервісів хмарного зберігання, а також інструменти для забезпечення безпеки передачі та зберігання персональних даних пацієнтів. Важливим є також інтеграція систем штучного інтелекту, що можуть використовуватись для аналізу даних користувачів та персоналізації рекомендацій.

Також слід враховувати кілька важливих аспектів при розробці системи:

— адаптивність — система має коректно працювати на різних пристроях – від смартфонів до комп’ютерів;

— безпека даних — особливу увагу слід приділити забезпеченню захисту особистих даних користувачів;

— інтерактивність — швидка та ефективна взаємодія користувача з системою підвищує рівень довіри та зручність користування;

Впровадження таких рішень сприятиме підвищенню якості життя людей, які потребують психологічної підтримки.

1.2 Розгляд існуючих аналогів

У процесі розробки системи важливим кроком є аналіз існуючих рішень на ринку, які надають подібні функції. Вивчення аналогів дозволяє зрозуміти, які технології та підходи використовуються в сучасних системах, визначити їхні переваги та недоліки, а також виявити унікальні можливості для вдосконалення нової системи. У цьому розділі розглянуто найбільш популярні платформи, що пропонують інструменти для підтримки та реабілітації психологічного здоров'я, їхні переваги та недоліки.

1.2.1 Wysa

Wysa — це AI-асистент у сфері психологіічного здоров'я, який пропонує користувачам підтримку та ресурси для управління емоційним станом. Він був розроблений, щоб допомогти людям у боротьбі з тривогою, депресією та стресом через інтерактивні методи.

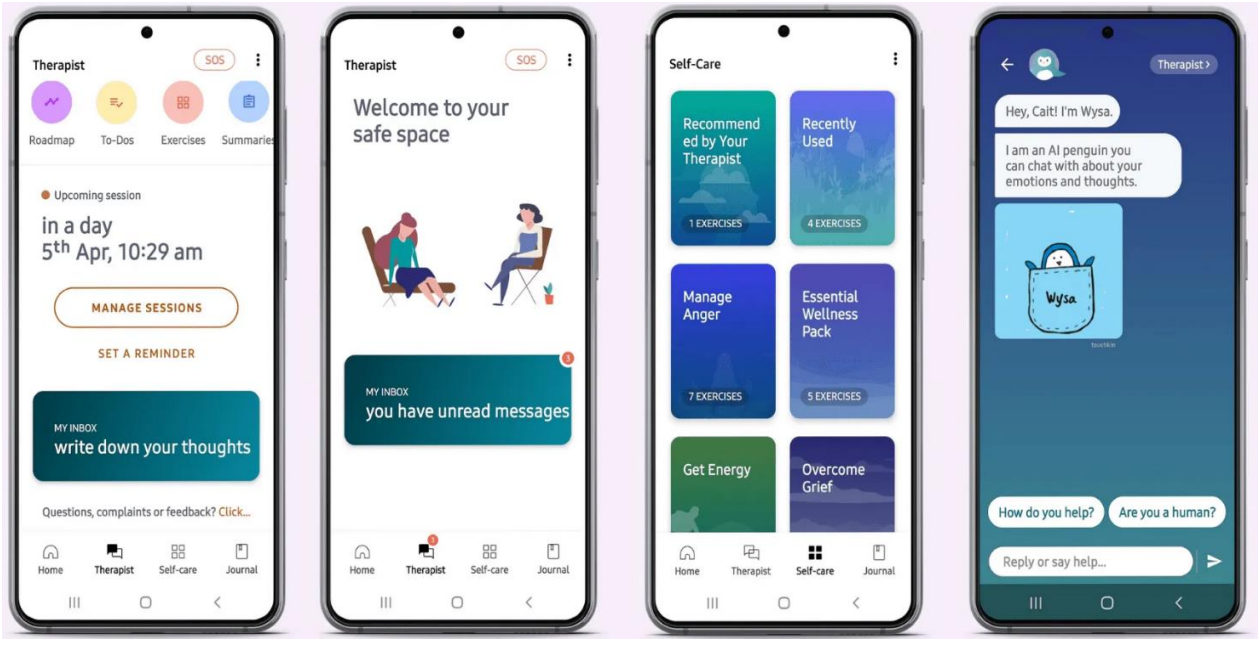

Рисунок 1.1 — Інтерфейс застосунку Wysa

Розглянемо деякі переваги та недоліки Wysa.

Переваги:

— анонімність — користувачі можуть використовувати Wysa без необхідності надання особистих даних, що забезпечує конфіденційність;

— широкий спектр методів — пропонує різноманітні інструменти для роботи з емоціями, включаючи медитацію, вправи на зниження стресу та когнітивно-поведінкову терапію;

— зручність використання — має інтуїтивний інтерфейс, що дозволяє легко записувати настрій і емоції.

Недоліки:

— ціна — висока вартість як для застосунку де використовується тільки AI, без надання послуг реальних терапевтів;

— обмеження в терапії — може бути недостатньо ефективною для користувачів з більш запущеними психологічними проблемами, яким потрібна допомога спеціаліста;

1.2.2 Calmerry

Calmerry — це онлайн-платформа для терапії, що пропонує користувачам можливість спілкуватися з сертифікованими терапевтами через текстові повідомлення або відеозв'язок.

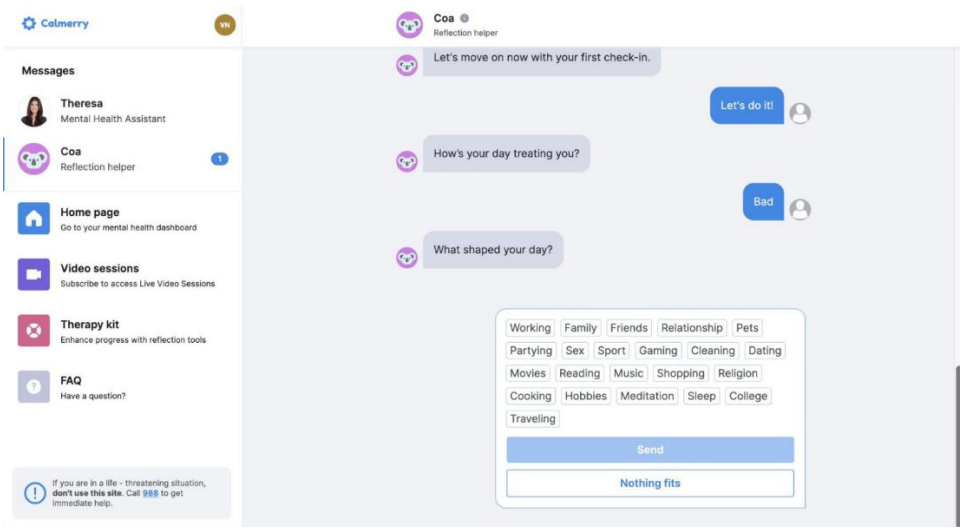

Рисунок 1.2 — Інтерфейс системи Calmerry

Розглянемо деякі переваги та недоліки Calmerry:

Переваги:

— консультації спеціаліста — Calmerry пропонує консультації з сертифікованими терапевтами, що може забезпечити більш якісну підтримку для користувачів з серйозними потребами;

— персоналізований підхід — штучний інтелект допомагає підбирати терапевтів відповідно до потреб користувачів, що робить процес підтримки більш персоналізованим;

Недоліки:

— ціна — оскільки користувачі платять за консультації з терапевтами ціна є дорожчою порівняно з іншими системами;

— обмежена доступність — для користувачів, які не мають стабільного інтернет-з'єднання, використання платформи може бути ускладнене.

1.2.3 Youper

Youper — це інтерактивний веб-додаток і мобільний додаток, який використовує штучний інтелект для управління емоційним станом користувача. Додаток пропонує психологічні інструменти, такі як щоденники настрою, вправи на саморозвиток та медитацію.

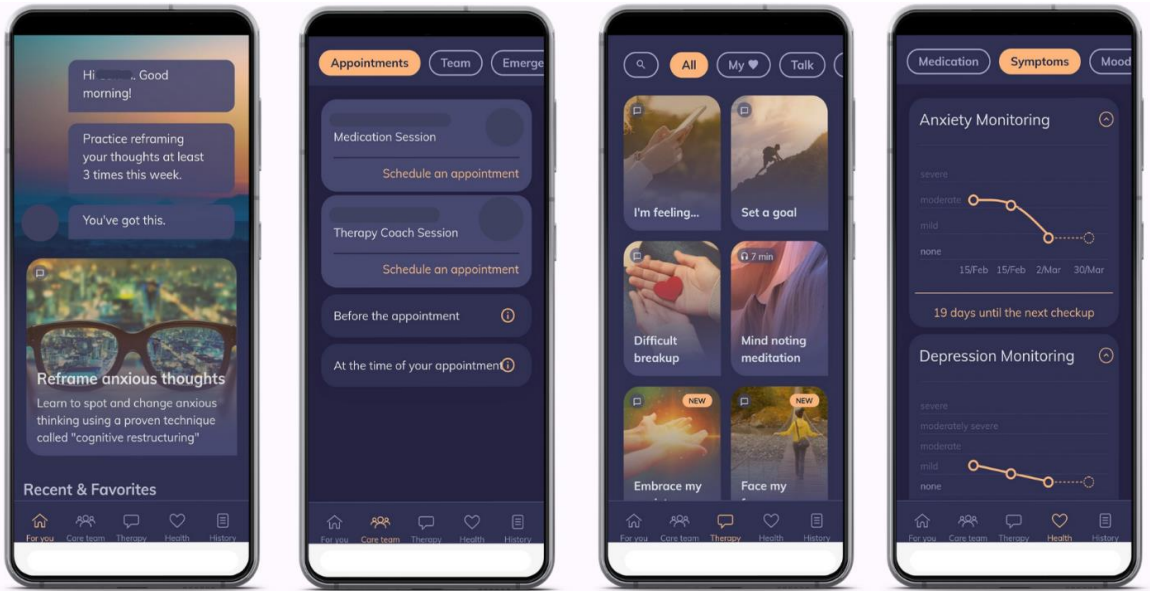

Рисунок 1.3 — Інтерфейс застосунку Youper

Розглянемо деякі переваги та недоліки Youper:

Переваги:

— адаптивність — Youper використовує штучний інтелект для аналізу емоцій та надання персоналізованих рекомендацій, що може допомогти користувачам у розвитку емоційної стійкості;

— науково обгрунтовані методи — Youper використовує перевірені психотерапевтичні техніки для підтримки користувачів;

— щоденник настрою — дозволяє користувачам відстежувати свій настрій та емоції, що може сприяти самоусвідомленню.

Недоліки:

— обмеження в терапії — може бути недостатньо ефективною для користувачів з серйозними психологічними розладами, яким потрібна допомога спеціаліста;

— щоденне заповнення — для досягнення максимального ефекту Youper вимагає активного введення даних від користувача, включаючи щоденне заповнення настрою та взаємодію з додатком.

1.3 Загальновідомі методи оцінки та підтримки психологічного здоров’я

Тестування та оцінювання є двома пов’язаними компонентами в психології. Спеціалісти використовують ці інструменти, щоб оцінити стан та зрозуміти шляхи вирішення проблем.

Тестування передбачає використання стандартизованих інструментів, які називають «тестами з посиланням на норму». Це означає, що всі учасники проходять тест за однакових умов, незалежно від місця проживання чи спеціаліста, який проводить тест.

Такі тести неодноразово були перевірені та проявили свою ефективність у вимірюванні певних рис чи розладів. Приклади тестів включають:

— MMPI (Міннесотський багатопрофільний опитувальник): визначає емоційні, поведінкові та особистісні характеристики;

— Beck Depression Inventory (BDI): оцінює рівень депресії;

— Шкала тривоги Гамільтона: вимірює рівень тривоги;

— Тест САН (самопочуття, активність, настрій): оцінює поточний емоційний стан;

— Тест Роршаха: використовує асоціації зі зображеннями для аналізу особистості.

Ще одним методом дослідження психологічного стану людини який можливо реалізувати у вигляді AI чату, є клінічні інтерв'ю, вони можуть бути структурованими, напівструктурованими або вільними, залежно від мети та особливостей пацієнта. Під час інтерв'ю спеціаліст ставить запитання про емоційний стан, поведінкові реакції, життєві ситуації та історію виникнення проблеми. Клінічне інтерв'ю є основою для формування терапевтичного плану, та забезпечення індивідуального підходу до лікування.

За допомогою різноманітних датчиків можливо проводити психофізіологічні дослідження, та отримані дані передавати в систему для моніторингу взаємозв'язку між психологічним станом людини та її фізіологічними показниками. Наприклад, до таких досліджень належить вимірювання серцевого ритму, артеріального тиску, тощо. Ці показники дозволяють оцінити рівень стресу, тривоги чи збудження, які можуть бути пов'язані з психологічними розладами. Наприклад, підвищений серцевий ритм і тиск часто супроводжують панічні атаки. Такі вимірювання дають змогу об'єктивно оцінити вплив психологічного стану на організм, що допомагає у діагностиці, моніторингу динаміки лікування та розробці персоналізованих терапевтичних підходів.

Щоденник емоцій — це інструмент, який допомагає людині краще розуміти свої емоційні стани, їхні причини та наслідки. У щоденнику записуються переживання, події, що їх викликали, а також фізичні та психологічні реакції на них. Така практика сприяє усвідомленню власних емоційних тригерів і способів реагування на них. Щоденники емоцій широко використовуються як у самостійному розвитку, так і в психотерапії для аналізу змін емоційного стану протягом часу. Вони допомагають ідентифікувати

деструктивні моделі поведінки, знижувати рівень стресу та покращувати навички емоційної регуляції. Регулярне ведення щоденника може стати важливим кроком до гармонійного психічного здоров’я.

Стосовно методів підтримки психологічного здоров’я можна сказати наступне — це сукупність підходів і практик, спрямованих на збереження емоційної рівноваги, подолання стресу та поліпшення якості життя. Вони включають як профілактичні заходи для попередження проблем, так і терапевтичні методи для їхнього вирішення.

Найпоширеніші методи в психотерапії:

— кгнітивно-поведінкова терапія: допомагає змінювати негативні моделі мислення;

— гештальт-терапія: спрямована на усвідомлення власних почуттів;

— психоаналіз: дослідження несвідомих конфліктів;

— сімейна терапія: вирішення проблем у родинному середовищі.

Медикаментозна терапія є важливим методом підтримки психічного здоров’я, особливо в разі серйозних психічних розладів, таких як депресія, тривожні розлади, біполярний афективний розлад чи шизофренія, проте такий метод можна застосовувати тільки за рекомендацією спеціаліста.

Медитація та релаксація — це ефективні методи підтримки психічного здоров’я, які допомагають знижувати рівень стресу, покращувати емоційне самопочуття та сприяють загальному розслабленню. Медитація, зокрема техніки усвідомленості (mindfulness), фокусується на присутності в даному моменті, дозволяючи людині відпустити негативні думки та емоції, що виникають у повсякденному житті. Релаксаційні вправи, такі як глибоке дихання або прогресивна м'язова релаксація, сприяють зниженню фізіологічних ознак стресу, таких як підвищене серцебиття чи напруга в м’язах. Ці методи мають на меті активувати парасимпатичну нервову систему, що допомагає відновити внутрішній баланс, покращити концентрацію та загальний емоційний стан. Регулярна практика медитації та релаксації може стати важливим інструментом у боротьбі з тривожністю, депресією та іншими психоемоційними розладами.

Фізична активність є важливим компонентом підтримки психічного здоров’я, оскільки вона допомагає знижувати рівень стресу, покращувати настрій і підвищувати загальний рівень енергії. Регулярні фізичні вправи стимулюють вироблення ендорфінів — гормонів радості, які сприяють покращенню емоційного стану та зменшують симптоми депресії і тривоги. Фізична активність також покращує якість сну, збільшує витривалість і знижує ризик розвитку багатьох хронічних захворювань, таких як серцево-судинні захворювання та ожиріння. Крім того, активність на свіжому повітрі та в групах, наприклад, заняття спортом з друзями, сприяє зміцненню соціальних зв’язків і підвищує почуття задоволення від життя. Регулярні фізичні вправи можуть стати потужним інструментом не тільки для фізичного здоров’я, але й для підтримки емоційної рівноваги та психічної стійкості.

Навчання управлінню стресом є важливим методом підтримки психічного здоров’я, оскільки стрес є одним з основних факторів, що впливає на емоційний стан і загальне самопочуття. Освоєння технік управління стресом дозволяє людині ефективно справлятися з життєвими труднощами, знижувати рівень тривоги та уникати емоційних перевантажень. До основних методів належать тайм-менеджмент для ефективного розподілу часу, що зменшує відчуття перевантаженості. Крім того, важливими є стратегії позитивного мислення та постановка реалістичних цілей, які дозволяють уникати відчуття безвиході.

Профілактичні заходи є ключовими для підтримки психічного здоров’я, оскільки вони дозволяють уникнути розвитку психологічних розладів та зберегти емоційну рівновагу. До основних профілактичних заходів належать підтримка здорового способу життя, зокрема регулярна фізична активність, збалансоване харчування та достатній сон.

Інтеграція більшості методів в інформаційні системи, може автоматизувати процеси відслідковування емоційного стану. Такі інтеграції можуть стати основою для розвитку персоналізованих, доступних і ефективних інструментів підтримки психічного здоров’я в майбутньому.

1.4 Особливості, переваги та недоліки веб-додатків

Веб-додаток — додаток, в якому клієнтом є оглядач Інтернета, а сервером — веб-сервер. Логіка додатка зосереджена на сервері, а відображення відбувається в браузері користувача [10].

Веб-додатки можна класифікувати на кілька категорій, такі як:

— статичні веб-додатки — це додатки, які складаються зі статичних сторінок HTML, CSS та JavaScript, вони зазвичай не мають зв'язку з базою даних і зберігаються на веб-сервері без змін;

— динамічні веб-додатки, ці додатки використовують серверний код, такий як PHP, Python або Ruby, для генерації динамічного контенту, вони можуть отримувати дані з бази даних, обробляти користувацький ввід та створювати персоналізований контент для користувачів;

— односторінкові — це додатки, які працюють на одній сторінці, асинхронно завантажуючи контент за допомогою AJAX, вони забезпечують швидку та плавну взаємодію з користувачем, оновлюючи тільки необхідну інформацію без перезавантаження сторінки;

— веб-портали — це додатки, які надають централізований доступ до різноманітних ресурсів, послуг та інформації, вони можуть містити різні функціональні модулі, такі як календар, електронну пошту, сповіщення, аналітичні звіти тощо;

— системи управління контентом (CMS), ці додатки дозволяють легко створювати, редагувати та керувати веб-контентом, вони надають інтерфейс для створення та оновлення сторінок, управління медіафайлами, керування правами доступу та іншими функціями, що спрощують процес керування веб-сайтом.

Кожен тип веб-додатків має свої особливості, переваги та недоліки, і вибір підходящого типу залежить від конкретних потреб та цілей проекту.

Основними особливостями є те що веб-додатки є доступними з будь-якого пристрою, що має доступ до Інтернету та веб-браузеру. Таким чином користувачі

мають можливість використовувати комп’ютери, смартфони, планшети та інші пристрої.

Веб-додатки розробляються з використання веб-технологій, таких як HTML, CSS і JavaScript, що дозволяє працювати на різних операційних системах, таких як Windows, macOS, Linux та інші. Це знижує зусилля для розробок та підтримки додатків на різних платформах.

Оновлення відбувається безпосередньо на сервері що забезпечує користувачам останні версії, без необхідності встановлювати оновлення на своїх пристроях.

Завдяки використанню серверів, веб-додатки можуть обробляти велику кількість користувачів одночасно. Це дозволяє забезпечити високу продуктивність та доступність навіть при значному навантаженні.

Веб-додатки можуть взаємодіяти з різними веб-сервісами та API, що дозволяє їм використовувати широкий спектр функціональних можливостей. Вони можуть взаємодіяти з соціальними мережами, оплатою онлайн, геолокацією, електронною поштою та багатьма іншими сервісами.

Безпека є важливим аспектом веб-додатків, оскільки вони зазвичай передають чутливу інформацію через Інтернет. Серед найпоширеніших загроз можна виділити:

— SQL-ін'єкції — зловмисники можуть вставити шкідливий SQL-код в запити до бази даних;

— крос-сайтові скрипти (XSS) — зловмисник може ввести шкідливий JavaScript-код, що дозволяє викрасти інформацію користувачів;

— управління сесією — недостатня безпека може призвести до викрадення сесії користувача, що дозволить зловмиснику отримати доступ до облікових записів.

Продуктивність веб-додатків є критично важливою для забезпечення комфортного досвіду користувачів. Основні стратегії оптимізації включають:

— кешування — зберігання часто запитуваних даних у пам'яті для швидшого доступу;

— мінімізація запитів — зменшення кількості HTTP-запитів, об'єднуючи файли CSS та JavaScript;

— оптимізація зображень — використання зображень з меншим розміром або у форматах, що швидше завантажуються, таких як WebP.

Важливим аспектом є те що веб-додатки можуть бути інтегровані з мобільними додатками, дозволяючи користувачам отримувати доступ до функцій як через веб-браузер, так і через спеціалізовані додатки. Це дозволяє розширити функціональність, надаючи користувачам зручний доступ до тих самих послуг на різних платформах.

Веб-додатки продовжують розвиватися, і завдяки новим технологіям, таким як Progressive Web Apps (PWA), які поєднують переваги веб-додатків і мобільних додатків, що дозволяє розробникам створювати швидкі, надійні та інтерактивні програми, що можуть працювати офлайн і відправляти сповіщення.

Нижче описані недоліки притаманні веб-додаткам.

Відсутність підтримки стану сеансу роботи і затримка при перезавантаженні кожної сторінки. Кожне перевантаження (або оновлення сторінки) викликає помітну затримку, викликану необхідністю встановити HTTP-з'єднання, обробити запит на сервері, передати по мережі у відповідь HTTP-повідомлення і перезавантажити сторінку браузером [11].

Хоча розробка веб-додатків дає велику свободу творчості дизайнерам і розробникам, це може призвести до різноманітності в інтерфейсах та способах взаємодії часто створюють проблеми для користувачів. Користувачі зіштовхуються з різними стилями візуальних інтерфейсів та способами взаємодії, кожен з яких має свій власний словник об'єктів, дій і візуальних концепцій, які змішуються в одному додатку.

Щоб вирішити ці проблеми, багато компаній розробляють керівництва з проектування користувацького інтерфейсу та стилю, щоб навести порядок у дизайні та функціональності веб-додатків. Ці керівництва встановлюють спільні стандарти і рекомендації щодо вигляду, поведінки та взаємодії з елементами інтерфейсу.

Ці керівництва допомагають забезпечити більш однорідний і сприятливий досвід користувача, де веб-додатки мають спільні елементи дизайну, логіку взаємодії та поведінку. Це спрощує навігацію та розуміння для користувачів, що працюють з різними веб-додатками.

Не зважаючи на ці недоліки, веб-додатки залишаються дуже популярними та ефективними інструментами для автоматизації бізнес-процесів, забезпечення спільної роботи та надання послуг користувачам. Вони використовуються в різних сферах, від електронної комерції та соціальних мереж до управління проектами та хмарних обчислень.

Загалом, веб-додатки є гнучкими, доступними та масштабованими інструментами, які забезпечують користувачам широкі можливості безпосередньо з веб-браузера.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ СИСТЕМИ, ІНТЕГРАЦІЯ ЗІ ШТУЧНИМ ІНТЕЛЕКТОМ

2.1 Архітектура веб-додатку

Архітектура веб-додатку є важливим аспектом його розробки, оскільки визначає структуру, компоненти та взаємодію між ними. Вибір архітектурного стилю впливає на продуктивність, масштабованість, безпеку та зручність підтримки додатку. Сучасні веб-додатки можуть реалізовуватися в різних архітектурних підходах, таких як монолітна, мікросервісна, серверлес і клієнт-серверна архітектура, кожен з яких має свої переваги та недоліки. Розуміння цих архітектур допомагає розробникам вибрати оптимальне рішення для досягнення вимог бізнесу та забезпечення високої якості продукту.

Монолітна архітектура — це підхід до розробки веб-додатків, при якому вся функціональність програми об'єднана в єдиний кодовий базі (рис. 2.1). Усі компоненти, включаючи серверну логіку, базу даних і клієнтський інтерфейс, існують в одному проєкті, що спрощує розгортання та підтримку на початкових етапах розробки. Монолітні додатки легко зрозуміти, оскільки їхня структура є єдиною, але з ростом проекту можуть виникати проблеми з масштабованістю, складністю управління та швидкістю внесення змін. Зміни в одному компоненті можуть вимагати повторного тестування всього додатку, що може уповільнити процес розробки.

Серверлес архітектура передбачає, що розробники створюють функції, які виконуються в хмарі, без необхідності управління серверами (рис 2.2). У цьому підході провайдер хмарних послуг автоматично виділяє ресурси, обробляє запити та масштабує навантаження за потреби. Розробники фокусуються на написанні коду та логіки, не турбуючись про серверну інфраструктуру, що спрощує управління і зменшує витрати. Серверлес архітектура ідеально підходить для проектів з непередбачуваним навантаженням, оскільки оплата стягується лише за фактично використані ресурси. Однак можуть виникати виклики, пов’язані з управлінням станом додатку, а також з безпекою.

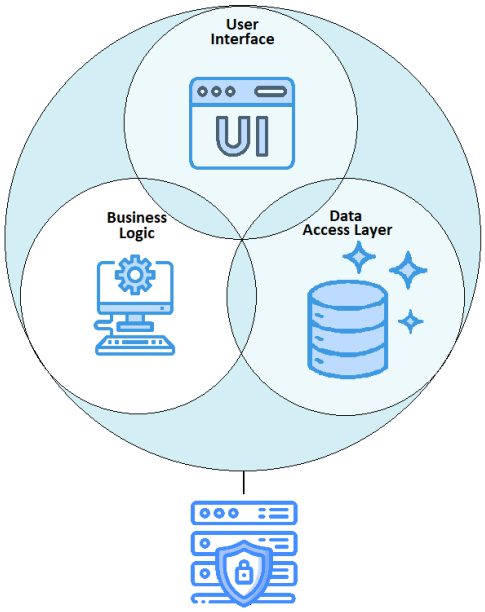

Рисунок 2.1 — Монолітна архітектура

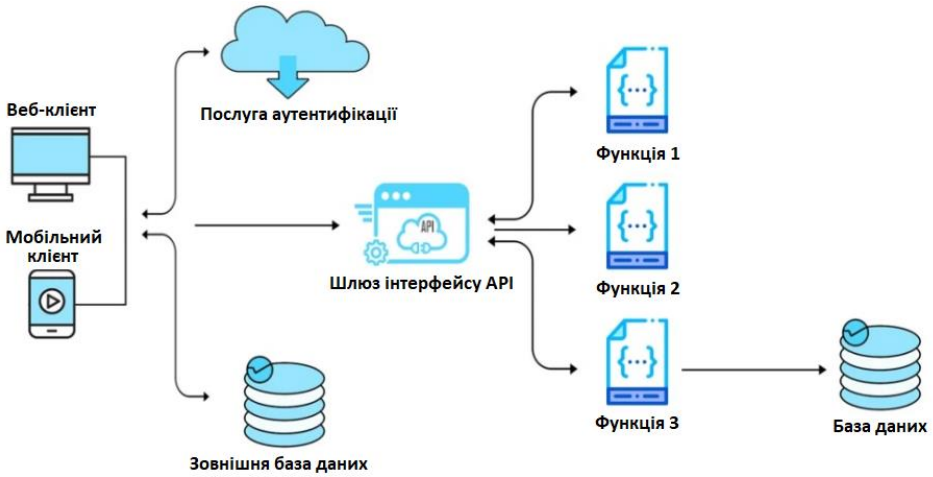

Рисунок 2.2 — Серверлес архітектура

Мікросервісна архітектура є підходом, який базується на створенні додатків як набору незалежних, взаємопов'язаних сервісів, кожен з яких виконує конкретну функцію (рисунок 2.3). Кожен мікросервіс має свою логіку, базу даних і може бути розгорнутий та масштабований окремо від інших мікросервісів. Це дозволяє командам працювати над різними частинами додатку незалежно, швидше впроваджувати нові функції та швидше реагувати на зміни в бізнес-вимогах. Мікросервісна архітектура також сприяє використанню різних технологій та мов програмування для різних сервісів, що дозволяє оптимізувати продуктивність та ресурсозатрати. Однак така архітектура може створювати

складнощі в управлінні, тестуванні та налагодженні взаємодії між сервісами, оскільки потрібно враховувати мережеві виклики та забезпечувати належну безпеку даних.

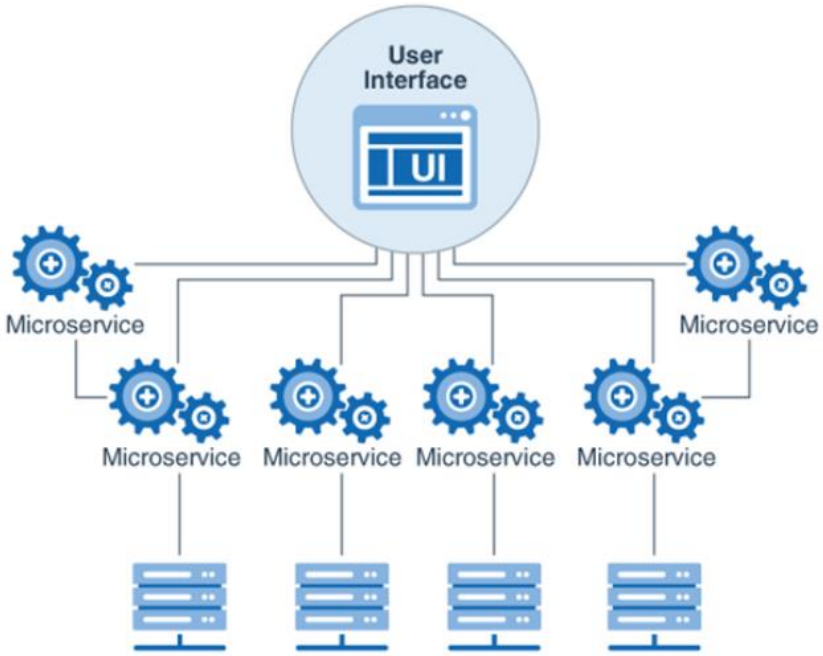

Рисунок 2.3 — Мікросервісна архітектура

Я віддаю перевагу використанню клієнт-серверної архітектури, яка є поширеним підходом до розробки веб-додатків, при якому система розділена на дві основні частини: клієнт і сервер (рис 2.4). Клієнтська частина, яка зазвичай виконується у веб-браузері або мобільному додатку, відповідає за взаємодію з користувачем і надсилає запити до сервера для отримання даних. Сервер, в свою чергу, обробляє ці запити, виконує бізнес-логіку та повертає результати клієнту.

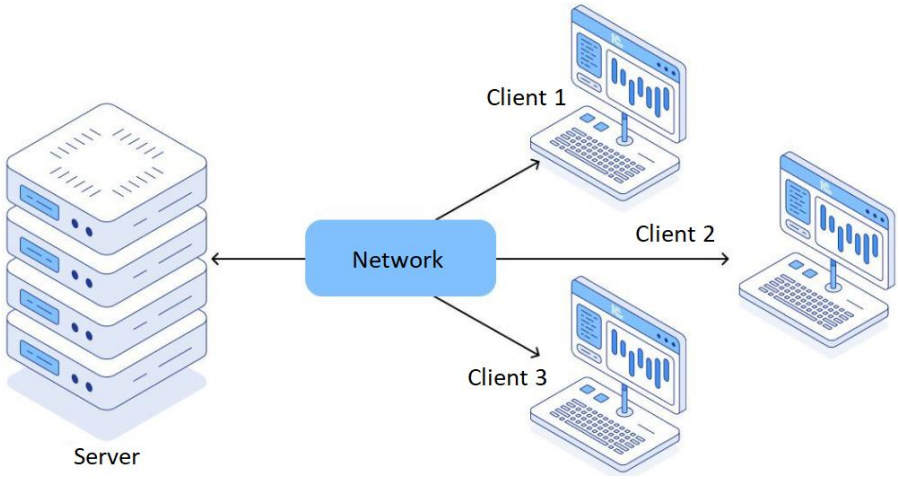

Рисунок 2.4 — Клієнт-серверна архітектура

Клієнт-серверна архітектура має кілька переваг, які роблять її популярним вибором для розробки веб-додатків. Ось основні з них:

— підвищується загальна продуктивність системи, оскільки клієнти та сервер перебувають на різних комп'ютерах, їхні процесори можуть виконувати різні завдання паралельно;

— знижується вартість апаратного забезпечення; досить потужний комп'ютер з більшим пристроєм зберігання потрібний тільки серверу;

— скорочуються комунікаційні витрати;

— підвищується рівень несуперечливості даних, оскільки кожній клієнтській програмі не доведеться виконувати власну перевірку їх цілісності [9].

Архітектура клієнт-сервер складається з рівнів, які включають користувальницький інтерфейс, бізнес-логіку та рівень даних, що взаємодіють між собою для забезпечення коректного функціонування системи.

Користувальницький інтерфейс зазвичай реалізується на клієнтських пристроях і надає можливість користувачам взаємодіяти з додатком. Складність таких інтерфейсів може варіюватися: від простих програм, що відображають результати, до більш комплексних систем з розширеними можливостями.

Бізнес-логіка визначає правила та принципи взаємодії об'єктів у системі. Вона втілюється у вигляді алгоритмів, правил, моделей бізнес-процесів та інших елементів. Бізнес-логіка взаємодіє з інфраструктурними сервісами на нижньому рівні та з користувальницьким інтерфейсом або зовнішніми системами на вищому рівні.

Рівень даних у клієнт-серверній моделі включає програми для обробки інформації в додатках. Зберігання даних може бути організоване через файлову систему або базу даних. Цей рівень забезпечує цілісність даних для різних додатків і містить метадані про додатки. У традиційній моделі клієнт-сервер рівень даних, як правило, розташований на стороні сервера. Використання реляційних баз даних дозволяє розділити обробку даних і їх зберігання, хоча в деяких випадках об'єктно-орієнтовані бази даних можуть бути кращим вибором, особливо для програм з комплексними структурами даних.

На основі проведеного аналізу була створена таблиця, що містить функції, які виконуються в середовищі клієнт-сервер (таблиця 2.1).

Таблиця 2.1 — Функціональність між клієнтом та сервером.

Сервер	Клієнт
Забезпечує цілісність даних	Виконує додаток
Обробляє та виконує запити до бази даних, що надходять від клієнтів	Керує інтерфейсом взаємодії з користувачем
Підтримує роботу системного каталогу	Відображає дані, отримані від сервера
Перевіряє права доступу користувачів	Формує запит до бази даних і передає його серверу для обробки
Обробляє запити й передає результати клієнту	Перевіряє коректність отриманих запитів перед відправкою

Для взаємодії між клієнтом і сервером через протокол HTTP необхідний веб-сервер на стороні сервера та клієнт, наприклад, веб-браузер на комп'ютері користувача. Усе починається з того, що користувач вводить URL-адресу в адресний рядок браузера.

Коли користувач вводить URL сторінки та натискає Enter, браузер перевіряє, чи існує домен з вказаним ім'ям. Якщо домен активний, браузер отримує IP-адресу сервера, пов'язану з цим доменом, і надсилає HTTP-запит до цього сервера.

HTTP-запит містить різну інформацію, таку як тип запиту (наприклад, GET, POST, PUT), шлях до ресурсу (URL), параметри, заголовки та інші дані. Запит передається через Інтернет до сервера за вказаною IP-адресою.

На сервері веб-сервер приймає цей HTTP-запит і обробляє його. В залежності від типу запиту та логіки, реалізованої на сервері, веб-сервер виконує необхідні дії. Наприклад, якщо це запит типу GET, сервер може повернути відповідну веб-сторінку або інший ресурс у відповідь.

Після обробки запиту сервер формує HTTP-відповідь, яка включає статус виконання запиту (наприклад, 200 OK для успішного виконання), дані відповіді та заголовки з додатковою інформацією.

Отримавши HTTP-відповідь, браузер обробляє її та відображає вміст сторінки на екрані користувача. Цей процес триває від моменту введення URL-адреси до відображення сторінки для користувача. Час відповіді може варіюватися в залежності від різних факторів, таких як швидкість мережі, навантаження на сервер, обробка запиту та інші аспекти, які можуть впливати на продуктивність системи.

2.2 Огляд технологій створення веб-додатку

У цьому підрозділі представлено огляд технологій, що використовуються для створення сучасних веб-додатків, із фокусом на серверну та клієнтську частини.

2.2.1 Серверна частина

Серверна частина веб-додатків є ключовою складовою, яка відповідає за обробку запитів від клієнтської сторони, взаємодію з базою даних, та забезпечення логіки додатка. Для створення ефективної та надійної серверної частини було обрано Python.

Python — це потужна та універсальна мова програмування, яка визначається своєю простотою та читабельністю коду. Заснована у 1991 році Гвідо ван Россумом, вона швидко завоювала популярність завдяки своїй лаконічності та великому співтовариству розробників. Python дозволяє вирішувати широкий спектр завдань, починаючи від веб-розробки і закінчуючи штучним інтелектом.

Мова програмування Python широко використовується в компаніях на кшталт Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та багатьох інших для розв'язання різноманітних робочих завдань. Python здобув

популярність серед розробників завдяки простоті у вивченні, ефективності та здатності працювати на різних платформах.

Однією з ключових особливостей Python є динамічна типізація, що дозволяє автоматично визначати тип змінних під час виконання програми. Оскільки Python є інтерпретованою мовою, він може знаходити помилки до моменту повної компіляції або запуску програми, що значно полегшує процес налагодження. Коли виникає помилка, Python надає детальне повідомлення про її місцезнаходження та причину, що допомагає швидше її виправити.

Python підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам під час виконання програми дізнаватися про структуру об'єктів і досліджувати їх внутрішню будову. Додатково, мова відзначається логічним і зрозумілим синтаксисом, що сприяє читабельності коду. Вона забезпечує гнучкість і масштабованість, що дозволяє швидко розробляти високорівневу логіку та легко адаптувати складні програми. Як інтерпретована мова, Python дає змогу писати код у будь-якому текстовому редакторі на будь-якій платформі й виконувати його без складнощів.

Втім, Python має і певний недолік, він може бути повільнішим у порівнянні з деякими іншими мовами програмування і споживати більше ресурсів.

Python також має багатий набір бібліотек, які розширюють можливості мови та спрощують роботу з типовими задачами. Ці бібліотеки надають готові рішення, дозволяючи програмістам використовувати вже написаний код замість створення з нуля. Крім того, Python підтримує велику кількість фреймворків, що робить його ще більш універсальним для розробки програм різної складності.

Фреймворк (framework) — це набір готових компонентів, бібліотек, інструментів і правил, що надають структуру та спрощують процес створення програмного забезпечення. Фреймворки забезпечують загальну архітектуру додатку, надаючи шаблони, класи, функції та інші елементи, які допомагають розробникам ефективно будувати програми, дотримуючись встановлених стандартів та практик.

Основна ідея використання фреймворку в Python полягає в тому, щоб уникнути необхідності написання коду з нуля для кожного нового проєкту. Замість цього, фреймворки надають готові модулі та компоненти для реалізації ключових функцій, як-от маршрутизація URL, обробка запитів, взаємодія з базами даних, автентифікація користувачів тощо.

Мною проаналізовано найпопулярніші фреймворки для створення API, серед них:

— Django — потужний веб-фреймворк, який підтримує розробку як API, так і повноцінних веб-додатків, пропонує вбудовану підтримку ORM, аутентифікації, авторизації та інші корисні функції;

— Flask — легкий і гнучкий мікро-фреймворк для швидкої розробки API з мінімальним набором компонентів, який можна розширити додатковими розширеннями;

— FastAPI — сучасний асинхронний веб-фреймворк, орієнтований на продуктивність, який підтримує типізацію, автоматичну документацію та валідацію даних, дозволяючи створювати високопродуктивні асинхронні API.

Є також інші менш поширені фреймворки, такі як Pyramid та Tornado.

На мою думку, FastAPI є оптимальним вибором для розробки API з кількох причин:

— швидкість і продуктивність (він може обробляти велику кількість запитів одночасно та забезпечує високу ефективність);

— асинхронність (підтримує асинхронне програмування, що дає змогу швидко відповідати на запити та ефективно взаємодіяти з іншими асинхронними сервісами);

— валідація і документація (FastAPI має зручні засоби для валідації вхідних даних і автоматично генерує документацію для API, що прискорює розробку та полегшує виявлення помилок);

— типізація даних (за допомогою Pydantic можна визначати схеми даних, що спрощує роботу з даними та допомагає уникати помилок);

— зручність використання (простий синтаксис дозволяє швидко освоїтися у фреймворку);

— інтеграція з іншими інструментами (FastAPI легко інтегрується з популярними бібліотеками та інструментами Python, такими як SQLAlchemy для баз даних, Celery для асинхронних задач або Pytest для тестування);

— стабільність і активний розвиток (FastAPI постійно оновлюється і вдосконалюється, завдяки чому залишається актуальним і відповідає сучасним вимогам розробки).

Ці особливості роблять FastAPI гнучким і зручним вибором для широкого спектра проєктів.

Перейшовши до вибору бази даних, мій вибір зупинився на реляційній системі управління базами даних. Реляційна система управління базами даних є популярною моделлю організації та зберігання даних, що використовує таблиці для відображення зв'язків між різними об'єктами. У реляційних системах управління базами даних дані структуруються у вигляді рядків і стовпців, де кожна таблиця представляє певний тип сутності, а стовпці визначають її атрибути. Завдяки своїй гнучкості та надійності реляційні системи управління базами даних широко застосовуються в бізнесі, науці, фінансових системах та багатьох інших сферах.

PostgreSQL — це потужна та широко відома реляційна система управління базами даних (СУБД), що відзначається високою продуктивністю, надійністю та можливістю розширення. Вона працює на основі реляційної моделі і підтримує безліч функцій SQL, включаючи складні запити, транзакції, індекси тощо.

Серед основних переваг PostgreSQL виділяється її розширюваність, яка забезпечує можливість додавання нових типів даних, операторів, функцій і індексів, що дає змогу налаштувати систему під конкретні потреби. Система також підтримує мультиверсійну архітектуру, яка дозволяє обробляти кілька транзакцій одночасно без блокування доступу до даних, забезпечуючи надійну роботу в умовах високої завантаженості. Продуктивність PostgreSQL посилюється завдяки оптимізатору запитів, який вибирає найефективніший план

для кожного запиту, та підтримці паралельного виконання запитів на багатоядерних системах. Окрім цього, PostgreSQL має високий рівень безпеки, зокрема підтримує автентифікацію, авторизацію, шифрування даних і аудит, що дозволяє детально налаштовувати права доступу для різних користувачів.

ORM (Object-Relational Mapping) – це технологія, що сприяє інтеграції між об'єктами програмного коду і реляційними базами даних, вирішуючи несумісності між об’єктно-орієнтованою моделлю додатку та структурою бази даних. ORM автоматизує процес перетворення даних між об'єктами коду і таблицями бази даних, що полегшує збереження і вилучення даних. З використанням ORM інструменти бази даних не потребують знання внутрішньої структури програми, а об’єкти додатку не мають знати структуру сховища даних.

SQLAlchemy – один із популярних ORM-інструментів для Python, що забезпечує функціонал SQL разом із можливостями Object Relational Mapper. SQLAlchemy дозволяє розробникам зв'язувати класи об'єктів з таблицями бази даних, що дає змогу чітко формулювати зв’язки між об'єктною моделлю програми і структурою бази даних. Він об’єднує принципи SQL та ORM, поєднуючи зручність роботи з об'єктами і переваги реляційного зберігання даних, адаптуючи їх для розробників Python.

Використання Docker стало вирішальним кроком у спрощенні процесу розробки та розгортання додатків. Ця платформа контейнеризації дозволяє створити ізольоване середовище для запуску програмного забезпечення, забезпечуючи максимальну стабільність та сумісність незалежно від операційної системи чи інших зовнішніх факторів.

Завдяки Docker розробники можуть уникнути проблем з несумісністю програмного забезпечення, оскільки контейнер містить всі необхідні файли для його роботи, включно з бібліотеками, залежностями та іншими компонентами.

Однією з основних переваг Docker є його легкість у використанні та висока швидкість у порівнянні з традиційними віртуальними машинами (VM). Контейнери Docker менш ресурсомісткі та запускаються значно швидше, оскільки вони використовують ядро хостової операційної системи, що дозволяє

уникнути необхідності запуску повноцінної гостьової ОС, як у випадку з віртуальними машинами. Завдяки цьому Docker дозволяє більш ефективно використовувати ресурси і легко масштабувати додатки у великій інфраструктурі.

Docker також надає зручну систему управління образами (images), яка спрощує створення, зберігання та розповсюдження контейнерів. Образи є основою для контейнерів і містять усі необхідні інструкції для запуску додатка. Розробники можуть створювати свої власні образи або використовувати вже готові з Docker Hub — публічного репозиторію, де можна знайти образи для популярних додатків та інструментів. Завдяки цьому Docker значно полегшує процес налаштування середовищ розробки, тестування та продакшену, оскільки весь стек програмного забезпечення можна відтворити за лічені хвилини.

Важливим компонентом є Docker Compose — інструмент, що дозволяє одночасно запускати кілька контейнерів і налаштовувати їхню взаємодію. Це особливо корисно при розробці складних систем, де потрібне одночасне функціонування різних сервісів, таких як бази даних, бекенд і фронтенд. Docker Compose дозволяє легко керувати такими середовищами, що значно прискорює розробку та тестування.

Таким чином, Docker забезпечує простий, швидкий та ефективний підхід до управління додатками та їхніми залежностями. Його широкі можливості зробили його стандартом у сучасному розгортанні та розробці, особливо в середовищах з високими вимогами до надійності, продуктивності та автоматизації.

2.2.2 Клієнтслька частина

JavaScript є однією з найпопулярніших мов програмування, переважно використовуваною для веб-розробки. Завдяки своїм можливостям, вона дозволяє розробникам створювати інтерактивні та динамічні веб-сторінки, які доповнюють статичний контент HTML та CSS. Код JavaScript можна інтегрувати безпосередньо у HTML-сторінки або ж підключати як окремі файли.

Однією з ключових особливостей JavaScript є його здатність взаємодіяти з об'єктною моделлю документа (DOM), яка відображає структуру HTML-документів. JavaScript пропонує методи та властивості, які дозволяють маніпулювати цією структурою, що дає змогу динамічно змінювати вміст, стилі та поведінку веб-сторінок в залежності від дій користувача або інших подій.

Ця мова є інтерпретованою, що означає, що код виконується безпосередньо у веб-браузері без попередньої компіляції. Це забезпечує швидкий процес розробки та тестування, оскільки будь-які зміни можна відразу побачити в браузері. JavaScript також підтримує об'єктно-орієнтоване програмування, що дозволяє створювати та маніпулювати об'єктами, які представляють собою колекції пар ключ-значення. Розробники можуть визначати власні об'єкти, а також використовувати вбудовані, такі як об'єкт Date для роботи з датами чи об'єкт Math для математичних операцій.

Крім того, JavaScript підтримує асинхронне програмування за допомогою механізмів, таких як зворотні виклики, обіцянки та конструкція async/await. Це дозволяє виконувати операції без блокування, що є важливим для мережевих запитів та обробки вводу користувача, не заважаючи при цьому роботі інтерфейсу.

Окрім веб-розробки, JavaScript також знаходить застосування в інших сферах, таких як серверне програмування, розробка мобільних і настільних додатків. Його універсальність та широке використання роблять цю мову важливим інструментом для програмістів. Загалом, JavaScript є потужною та адаптивною мовою програмування, а її постійний розвиток та безліч бібліотек і фреймворків роблять її популярним вибором для реалізації різноманітних проектів в інтернеті.

React — це популярна JavaScript-бібліотека, створена компанією Facebook, що використовується для розробки користувацьких інтерфейсів веб-додатків. Вона надає можливість розробникам створювати складні, динамічні інтерфейси, розбиваючи їх на менші, легко керовані компоненти.

Головним принципом React є впровадження віртуального DOM (Document Object Model), що дозволяє підвищити швидкість і ефективність оновлення інтерфейсу. Ця бібліотека забезпечує декларативний підхід до розробки компонентів, в якому розробники описують, як має виглядати інтерфейс у різних станах, а React автоматично визначає, які частини потрібно оновити при зміні даних.

Крім того, React підтримує JSX (JavaScript XML), що дає змогу описувати структуру UI з використанням синтаксису, схожого на HTML, з можливістю вбудовування JavaScript-виразів. Це спрощує процес створення компонентів і їх інтеграцію з даними.

Однією з найбільших переваг React є його багатогранна екосистема, що включає різноманітні розширення, інструменти та сторонні бібліотеки, які полегшують розробку веб-додатків.

У підсумку, React є потужним інструментом для створення сучасних веб-додатків, пропонуючи високу швидкість реагування на зміни, ефективне управління станом і простоту в розробці компонентів.

2.3 Вибір і навчання моделей для діагностики психологічного стану

Для розробки системи, яка надає точні та надійні рекомендації щодо психологічного стану користувача, необхідно ретельно підійти до вибору моделей машинного навчання. Основною метою цього етапу є створення алгоритмів, здатних аналізувати відповіді користувача, поведінкові дані та інші відповідні показники, а також виявляти потенційні проблеми чи ризики в психологічному стані.

Для задачі діагностики психологічного стану особливо підходять такі моделі як логістична регресія, дерева рішень, метод опорних векторів (SVM) нейронні мережі, рекурентні нейронні мережі (RNN), кластеризація, глибинне навчання.

Логістична регресія, це статистичний метод, що використовується для бінарної класифікації. Логістична регресія прогнозує ймовірність належності

об'єкта до певної категорії. Може використовуватися для оцінки ймовірності наявності певних психологічних розладів.

Дерева рішень, це простий, але потужний метод, який представляє собою модель у формі дерева, де кожен вузол є питанням про певну ознаку. Використовується для класифікації відповідей користувачів на основі різних психологічних аспектів і допомагає виявляти основні фактори, які впливають на психологічний стан.

Метод опорних векторів (SVM), це потужний метод класифікації, який шукає гіперплощину, що найкраще розділяє дані на класи. Може бути застосований для розділення користувачів на різні групи за їх психологічними характеристиками на основі багатовимірних даних.

Нейронні мережі, це складні моделі, що імітують роботу людського мозку, складаються з нейронів, які навчаються на основі даних. Можуть бути використані для аналізу складних наборів даних, таких як текстові описи емоційного стану, виявлення шаблонів у поведінці та прогнозування ризиків розвитку психологічних проблем.

Рекурентні нейронні мережі (RNN), це тип нейронної мережі, що підходить для обробки послідовних даних, та LSTM (Long Short-Term Memory) що є варіантом RNN, який спеціалізується на вивченні довгострокових залежностей. Можуть бути використані для аналізу текстових даних, таких як повідомлення користувачів або щоденники, з метою виявлення емоційних станів або змін у психологічному стані з часом.

Кластеризація, це методи кластеризації, такі як K-means або ієрархічна кластеризація, групують дані на основі подібності. Можуть бути використані для виявлення підгруп користувачів з подібними психологічними характеристиками, що може допомогти в розробці персоналізованих рекомендацій.

Глибинне навчання, це використання глибинних нейронних мереж для виявлення складних патернів у великих наборах даних. Може бути застосовано для аналізу відео або аудіо даних, наприклад, для оцінки невербальної поведінки або інтонацій під час сеансів спілкування.

Для навчання моделей важливо мати набір даних, що містить відповіді користувачів на психологічні опитування, можливі записи сеансів спілкування, дані про рівень стресу, емоційні реакції та інші параметри. Необхідно провести попередню обробку, що включає очищення, нормалізацію та аугментацію даних, щоб забезпечити їх відповідність вимогам моделі. Крім того, важливим етапом є підбір особливостей, що дозволяє зосередитись на ключових показниках для діагностики.

Після обробки даних моделі проходять навчання на підготовленому наборі даних. Використовується крос-валідація та розбиття даних на навчальну та тестову вибірки для оцінки якості роботи моделі та уникнення перенавчання. Метрики, такі як точність, повнота, F-мера та ROC-AUC, допомагають оцінити, наскільки ефективно модель справляється із завданням.

Після початкового навчання моделі проходять серію тестувань та оптимізацій. Зокрема, виконується налаштування гіперпараметрів і вибір оптимальних значень, що максимізують ефективність моделей. Також здійснюється порівняння різних моделей, щоб обрати найбільш ефективні рішення для конкретної задачі.

Отже, вибір моделі залежить від специфіки даних, що використовуються, типу психологічного стану, який потрібно діагностувати, та доступних ресурсів для навчання моделей. Зазвичай, для досягнення найкращих результатів доцільно комбінувати кілька моделей та використовувати ансамблеві методи, що дозволяє отримати більш точні та надійні результати.

2.4 Опис функціональності системи

Функціональність системи охоплює сукупність функцій і можливостей, які вона надає користувачам або іншим системам. Щоб описати функціональні можливості системи, часто використовуються UML-діаграми, які візуалізують ці функції та їх взаємозв'язки. Одним з найпоширеніших видів UML-діаграм є діаграма варіантів використання (use case diagram), яка відображає функціональність системи з точки зору її користувачів.

Основні елементи діаграми варіантів використання включають:

— актори (представляють зовнішніх суб’єктів, що взаємодіють із системою, це можуть бути як люди, так і зовнішні системи чи інтерфейси);

— варіанти використання (відображають конкретні дії або функції, які користувач чи система можуть виконувати).

Для створення діаграми варіантів використання необхідно:

1) визначити акторів системи: ідентифікувати ключових користувачів або зовнішні системи, що взаємодіятимуть із системою;

2) визначити варіанти використання: окреслити основні функції чи дії, які можуть бути виконані;

3) зв’язати акторів з варіантами використання: встановити зв'язки, які показують, які актори мають доступ до яких функцій;

4) додати додаткові деталі: при необхідності описати акторів і варіанти використання для кращого розуміння.

Так як додаток призначений для того щоб допомогти користувачам вирішувати проблеми з психологічним здоров’ям, основними акторами які взаємодіють з функціональністю системи є: користувач (взаємодіє з основними функціями системи для аналізу свого стану, проходить опитування, використовує чат із ШІ, переглядає рекомендації, записує свій настрій, обирає спеціаліста та отримує консультації), спеціаліст (має можливість переглядати статистику користувача який надав доступ, надавати консультації та оцінювати результати проведеного сеансу), адміністратор (відповідає за управління користувачами та контентом, а також за аналіз загальної статистики системи), функції та дії яких зображено у діаграмі прецедентів на рисунку 2.5.

Основні прецеденти для користувача:

— проходження опитування (користувач може проходити різні опитування, що допоможуть виявити його психологічний стан);

— чат з AI (користувач взаємодіє з AI-чатом, який аналізує відповіді для виявлення потенційних психологічних проблем, та надання рекомендацій);

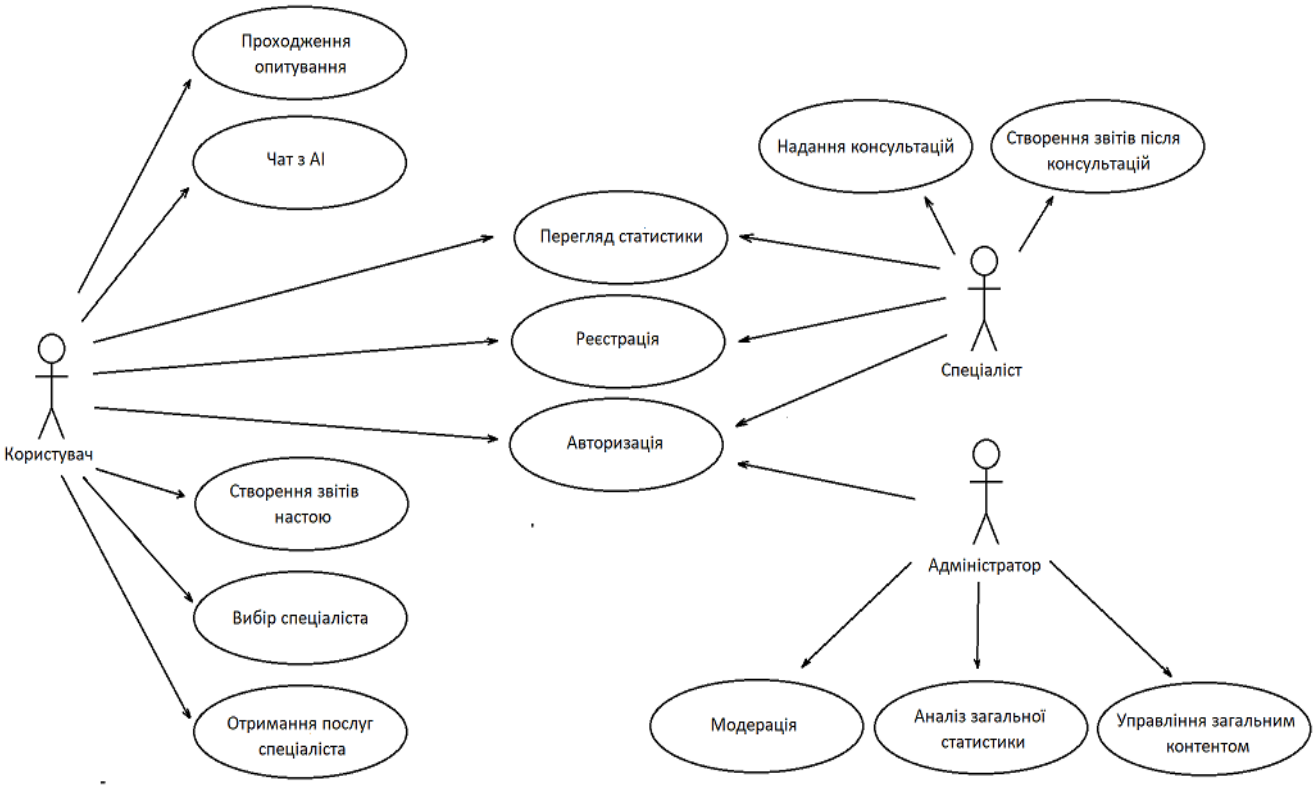

Рисунок 2.5 — Діаграма прецедентів системи

— перегляд статистики (на основі результатів опитувань, чату або консультацій з спеціалістом, та інш., користувач має можливість переглянути статистику власного стану.

— створення звітів настрою (користувач може реєструвати свій настрій, відзначаючи, як він себе почуває в певні дні та час);

— вибір спеціаліста (користувач може обирати спеціаліста на основі його рейтингу та успішного досвіду роботи з подібними випадками);

— отримання послуг спеціаліста (користувач має можливість звернутися до обраного спеціаліста для проведення консультацій).

Основні прецеденти для спеціаліста:

— огляд статистики клієнта (спеціаліст має доступ до даних, зібраних щодо психологічного стану клієнта, для більш детального аналізу);

— надання консультацій (спеціаліст проводить консультації, підтримує клієнта та може коригувати рекомендації, якщо це необхідно);

— створення звітів після консультацій (сспеціаліст може переглядати зібрану інформацію та аналізувати свій прогрес у роботі з клієнтами).

Основні прецеденти для адміністратора:

— модерація (адміністратор має можливість блокувати користувачів та спеціалістів за порушення правил);

— управління загальним контентом (опитуваннями, рекомендаціями. Адміністратор може оновлювати та змінювати опитування та рекомендації, які використовуються в системі);

— аналіз загальної статистики (адміністратор переглядає загальну статистику використання системи, щоб оцінювати її ефективність).

Також, одним із інструментів опису функціональності системи є діаграма діяльності, яка використовується для опису функціонування системи та порядку виконання її процесів. Вона демонструє послідовність дій або операцій і складається з різних графічних елементів, які відображають стани, дії, рішення та взаємозв’язки між ними. Основні компоненти таких діаграм включають:

— стан (відображає певний стан системи або об'єкта);

— дія (вказує конкретну дію або операцію, яка виконується в системі);

— рішення (представляє точку розгалуження, де система приймає рішення на основі певних умов);

— злиття (показує об'єднання шляхів після виконання різних альтернативних дій);

— вибір (визначає варіанти вибору або альтернативні шляхи, які можуть бути обрані системою);

— сигнал (представляє подію або сигнал, який може спричинити перехід до іншого стану або дії).

Дане моделювання дозволяє уявити послідовність дій в системі, що представлено на рисунку 2.6.

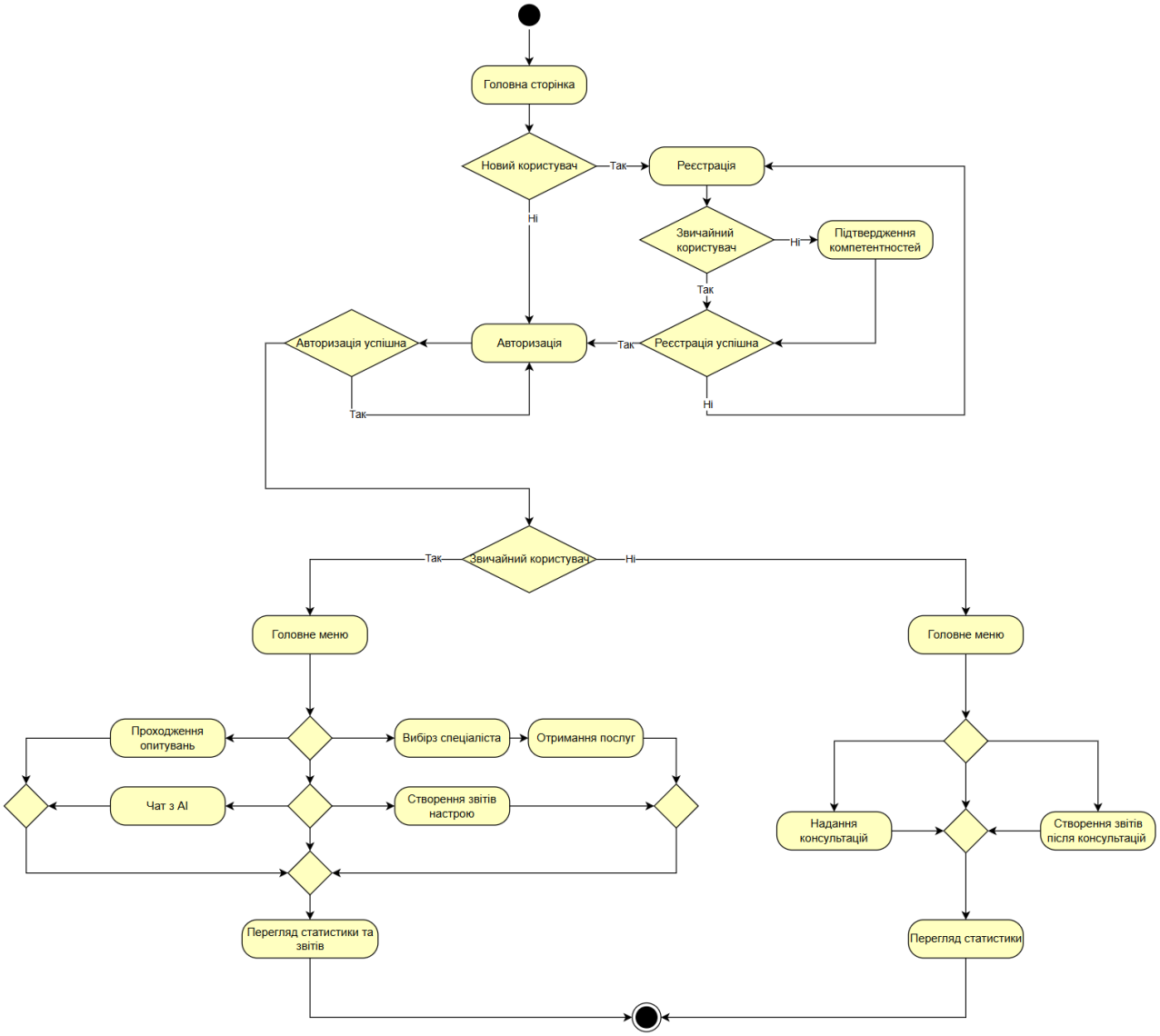

Рисунок 2.6 — Діаграма діяльності системи

2.5 Проектування бази даних

Проектування бази даних є ключовим кроком для забезпечення ефективної роботи системи. База даних слугує сховищем для всіх даних, що використовуються в системі, зокрема інформації про користувачів, оцінки психологічного стану, опитування, рекомендації та інші сутності. Розробка бази даних включає опис структури даних та зв'язків між ними для забезпечення збереження, обробки й аналізу інформації.

Для ефективного проектування бази даних існує ряд кроків.

Проектування бази даних починається з визначення потреб системи: це аналіз того, які дані потрібно зберігати, як вони повинні бути структуровані, а також які операції будуть виконуватись над ними. Важливо врахувати типи даних, взаємозв’язки між ними, а також основні операції, які система повинна виконувати з цими даними.

Наступним етапом є визначення сутностей — базових об’єктів, які зберігатимуть дані в системі. Кожна сутність відповідає певній таблиці в базі, що містить атрибути, необхідні для зберігання конкретної інформації.

Для кожної сутності встановлюються її атрибути, тобто характеристики, що описують дані у таблиці. Наприклад, сутність «User» може мати такі атрибути, як ім'я, електронна пошта, та пароль. Це основні властивості, що визначають дані, які необхідно зберігати для кожного користувача.

На цьому етапі також визначаються первинні ключі. Кожна таблиця потребує унікального ідентифікатора для однозначної ідентифікації записів, що дозволяє швидко знайти чи оновити потрібний запис.

Важливо також встановити зв'язки між сутностями. Це означає, що слід визначити, які сутності пов’язані одна з одною, і як саме: відносини можуть бути один до одного, один до багатьох або багато до багатьох.

Коли зв’язок між сутностями визначений, слід встановити зовнішні ключі. Вони створюють посилання на первинні ключі інших таблиць, дозволяючи пов’язати записи між собою.

Крім цього, можуть бути визначені додаткові обмеження, такі як унікальність значень, цілісність даних або вимоги валідації, щоб забезпечити достовірність даних у системі.

Завершальний етап — розробка схеми бази даних, яка візуалізує структуру всіх таблиць, їх атрибутів, зв’язків між ними та інших елементів бази, що забезпечує наочне уявлення структури та взаємодії даних у системі.

Дотримуючись вище описаних кроків було створено концептуальну модель бази даних, що зображена на рисунку 2.7, що містить наступні наступні таблиці: «Users», «MentalHealthAssessments», «Surveys», «SurveyResponses»,

«Recommendations», «DailyMetrics», «Specialists», «SpecialistMatches», «AIChatLogs».

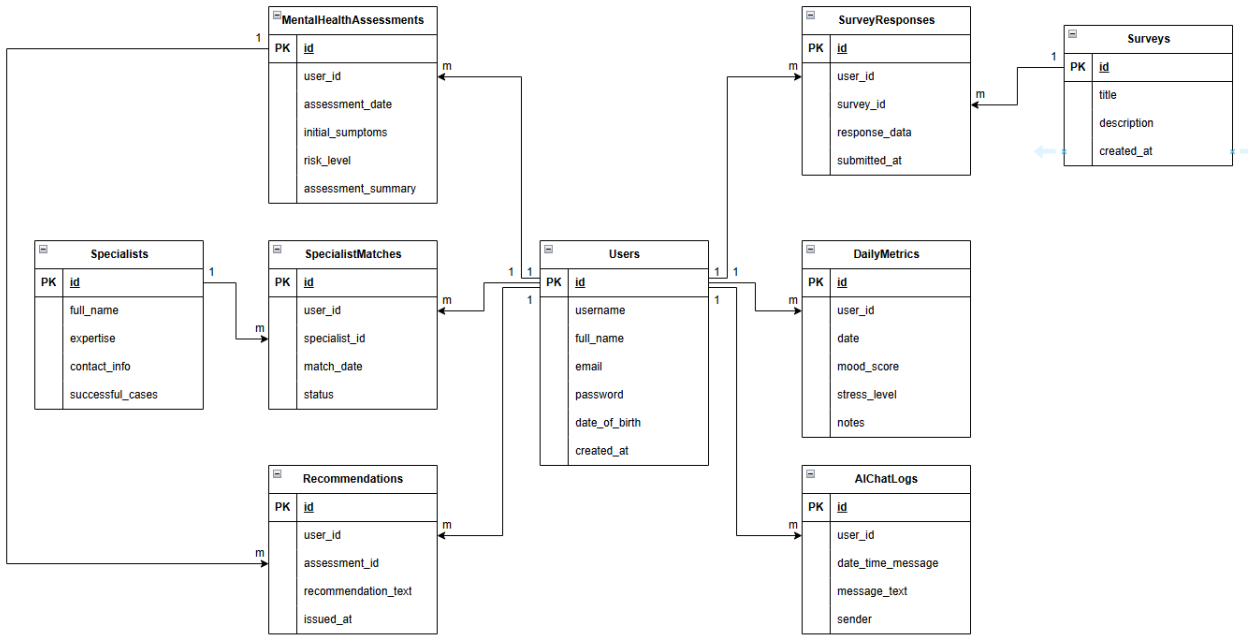

Рисунок 2.7 — Концептуальна модель бази даних

Структура таблиці «Users» (зберігає основну інформацію про кожного користувача) подана в таблиці має зв’язки One-to-Many з таблицями: «MentalHealthAssessments», «SurveyResponses», «DailyMetrics», «SpecialistMatches», «AIChatLogs», «Recommendation» (таблиця 2.2).

Таблиця 2.2 — Структура таблиці «Users»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первиний ключ
username	String	Ім’я для логіну
full_name	String	Повне ім’я користувача
email	String	Електрона адреса
password	String	Пароль
date_of_birth	Date	Дата народження
created_at	DateTime	Дата створення облікового запису

Таблиця «MentalHealthAssessments» (зберігає початкові оцінки, проведені через чат на основі ШІ, щоб виявити будь-які потенційні психологічні проблеми) має зв’язок Many-to-One з таблицею «Users», та One-to-Many з таблицею «Recommendations» (таблиця 2.3).

Таблиця 2.3 — Структура таблиці «MentalHealthAssessments»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первиний ключ
user_id	Int	Зовнішній ключ таблиці «Users»
assessment_date	DateTime	Дата отримання оцінки
initial_sumptoms	Text	Виявлені симптоми під час початкової взаємодії ШІ
risk_level	Enum	Рівень ризику на основі аналізу AI (наприклад, низький, середній, високий)
assessment_summary	Text	Резюме початкової оцінки на основі ШІ

Таблиця «Surveys» (зберігає інформацію про опитування, які надаються користувачам для виявлення конкретних проблем із психічним здоров’ям) має зв’язок One-to-Many з «SurveyResponses» (таблиця 2.4).

Таблиця 2.4 — Структура таблиці «Surveys»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первиний ключ
title	String	Назва опитування
description	Text	Опис опитування
created_at	DateTime	Дата створення опитування

Таблиця «SurveyResponses» (зберігає відповіді користувачів на кожне опитування) має зв’язки Many-to-One з таблицями «Users» та «Surveys» (таблиця 2.5).

Таблиця 2.5 — Структура таблиці «SurveyResponses»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первинний ключ
user_id	Int	Зовнішній ключ таблиці «Users»
survey_id	Int	Зовнішній ключ таблиці «Surveys»
response_data	JSON	Відповіді користувачів на опитування
submitted_at	DateTime	Дата та час проходження

Таблиця «Recommendations» (містить персоналізовані рекомендації, створені для користувачів на основі їхніх оцінок і відповідей на опитування) має зв’язки Many-to-One з таблицями «Users» та «MentalHealthAssessments» (таблиця 2.6).

Таблиця 2.6 — Структура таблиці «Recommendations»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первинний ключ
user_id	Int	Зовнішній ключ таблиці «Users»
assessment_id	Int	Зовнішній ключ таблиці «MentalHealthAssessments»
recommendation_text	Text	Текст рекомендації
issued_at	DateTime	Дата та час отримання рекомендації

Таблиця «DailyMetrics» (містить дані про самооцінку користувача, як настрій та інші емоційні показники, для відстеження з часом) має зв’язок Many-to-One з «Users» (таблиця 2.7).

Таблиця 2.7 — Структура таблиці «DailyMetrics»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первинний ключ
user_id	Int	Зовнішній ключ таблиці «Users»
date	Date	Дата запису
mood_score	Int	Самостійна оцінка настрою
stress_level	Int	Самостійна оцінка стресу
notes	Text	Додаткові примітки

Таблиця «Specialists» (зберігає інформацію про спеціалістів, доступних для консультації) має зв’язок One-to-Many з «SpecialistMatches» (таблиця 2.8).

Таблиця 2.8 — Структура таблиці «Specialists»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первинний ключ
full_name	String	Повне ім’я спеціаліста
expertise	String	Сфера знань
contact_info	String	Контактні дані
successful_cases	Int	Кількість успішно розглянутих справ

Таблиця «SpecialistMatches» (реєструє випадки відповідності користувача та спеціаліста, що дозволяє користувачам бачити, які спеціалісти вирішували подібні випадки) має зв’язки Many-to-One з таблицями «Users» та «Specialists» (таблиця 2.9).

Таблиця 2.9 — Структура таблиці «SpecialistMatches»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первинний ключ
user_id	Int	Зовнішній ключ таблиці «Users»

Продовження таблиці 2.9

specialist_id	Int	Зовнішній ключ таблиці «Specialists»
match_date	DateTime	Дата підбору користувача до фахівця
status	Enum	Поточний статус консультації (наприклад, Виконується, Завершено)

Таблиця «AIChatLogs» (зберігає взаємодію між користувачем і системою чату на основі ШІ) має зв’язок Many-to-One з «SpecialistMatches» (таблиця 2.10).

Таблиця 2.10 — Структура таблиці «AIChatLogs»

Ім’я поля	**Тип поля**	**Опис поля**
id	Int	Первинний ключ
user_id	Int	Зовнішній ключ таблиці «Users»
date_time_message	DateTime	Дата та час повідомлення чату
message_text	Text	Текст обміну повідомленнями
sender	Enum	Визначає, чи є відправник "Користувач" чи "AI"

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Навчання моделі

Навчання моделі є критичним етапом у створенні системи, яка здатна коректно розпізнавати та обробляти різноманітні запити користувачів. Процес навчання включає підготовку даних, визначення архітектури нейронної мережі, налаштування параметрів оптимізації та запуск процесу навчання.

Нейронні мережі бувають різних видів, і кожен вид підходить для певних завдань і типів даних. Нижче наведені основні види нейронних мереж і їх особливості:

1) персептрон (Perceptron);

— найпростіший вид нейронної мережі, який складається лише з одного нейрона або одного шару нейронів;

— здатен розв’язувати тільки лінійно роздільні задачі, тобто такі, де можна провести пряму лінію між класами.

2) багатошаровий персептрон (Multilayer Perceptron, MLP);

— складається з вхідного, одного або кількох прихованих шарів, і вихідного шару;

— належать до повнозв'язних нейронних мереж, де кожен нейрон одного шару під’єднаний до кожного нейрона наступного шару;

— може розв'язувати як лінійно, так і нелінійно роздільні задачі завдяки наявності прихованих шарів і нелінійних активаційних функцій.

3) згорткові нейронні мережі (Convolutional Neural Networks, CNN);

— ефективні для роботи з зображеннями, оскільки здатні розпізнавати складні патерни;

— складаються зі згорткових шарів (convolutional layers), шарів підсумовування (pooling layers) і повнозв'язних шарів;

— згорткові шари виділяють суттєві ознаки (feature extraction), що дозволяє мережі визначати зображення на основі складних взаємозв’язків між пікселями.

4) рекурентні нейронні мережі (Recurrent Neural Networks, RNN);

— підходять для роботи з послідовними даними, наприклад, з текстом або тимчасовими рядами, оскільки здатні запам'ятовувати попередні значення в послідовності;

— класичний RNN має проблеми з короткочасною пам'яттю, через що виникають труднощі із запам'ятовуванням довгих послідовностей.

5) двонаправлені рекурентні нейронні мережі (Bidirectional RNN);

— дозволяють обробляти послідовності з обох напрямків (з початку до кінця і з кінця до початку);

— застосовуються для задач обробки природної мови (NLP), коли важливий контекст як з попередніх, так і з наступних слів.

6) графові нейронні мережі (Graph Neural Networks, GNN);

— використовуються для задач, де дані представлені у вигляді графів;

— моделі GNN навчаються враховувати взаємозв’язки між вузлами графу, що дозволяє моделювати взаємодію між об’єктами.

Ці види нейронних мереж мають різні архітектурні особливості, що дозволяє ефективно використовувати їх у відповідних задачах.

Для ефективного навчання моделі, потрібно підготувати набір даних. В моєму випадку дані зберігаються в файлі «dataset.json», де визначено як чат-бот має відповідати на різні типи запитань (рисунок 3.1).

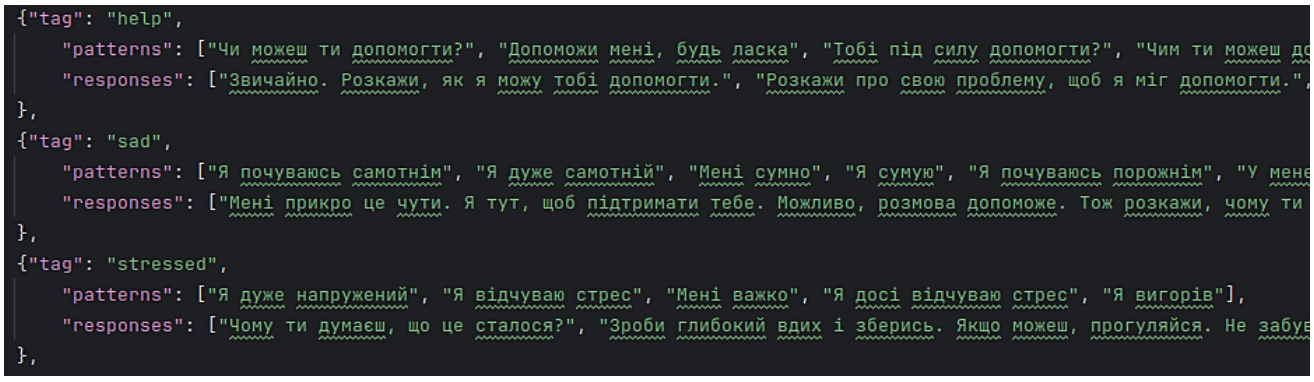

Рисунок 3.1 — Дані для навчання моделі

Кожен об’єкт в списку даних містить: tag (категорія або тип запиту), patterns (можливі варіанти, як користувач може формулювати питання), responses (варіанти відповідей).

Під час підготовки даних код проходить по кожному шаблону (patterns), щоб розбити текст на окремі слова (tokenize) і привести їх до кореневої форми (stem). Цей процес допомагає підготувати модель до розпізнавання шаблонів незалежно від форми слів.

Для кожного шаблону (наприклад, "Hello", "Hi there") створюється "мішок слів" — масив, що показує, які слова із загального списку слів присутні в конкретному запитанні. Цей "мішок слів" є входом для нейронної мережі, що навчається відповідати, знаходячи відповідний tag.

Під час навчання модель створює зв’язок між "мішками слів" і відповідними тегами (tags). Після навчання вона може класифікувати нові запити, які підходять до певних шаблонів.

Коли користувач вводить запит, бот передає його через модель, яка передбачає відповідний тег. Якщо впевненість у передбаченні висока (понад 0.75), бот надає відповідь. Якщо модель не впевнена у відповіді, вона продовжить ставити питання.

Для попередньої реалізації нейронної мережі та навчання моделі, використовувалась безкоштовна онлайн-платформа Google Colab, яку надає Google і яка дозволяє писати та виконувати код на Python без потреби встановлювати спеціальне середовище на своєму комп'ютері. Colab особливо популярний серед дослідників і розробників у галузі машинного навчання, оскільки пропонує доступ до графічних процесорів (GPU) та тензорних процесорів (TPU) для пришвидшення обчислень.

Colab надає безкоштовний доступ до апаратних прискорювачів (GPU, TPU), що робить його корисним для обчислювально інтенсивних задач, таких як тренування моделей машинного навчання.

Colab має багато заздалегідь встановлених бібліотек, що полегшує початок роботи. А також підтримує режим спільного редагування, що дозволяє кільком людям працювати над одним ноутбуком у реальному часі.

Коли є набір даних, середовище для розробки, та розуміння як використовувати ці дані, можна переходити до написання коду який реалізує нейронну мережу для класифікації запитань користувача і надання відповідних відповідей.

В розробці використовуються бібліотеки: torch (для створення нейронної мережі), nltk (для токенізації тексту), json (для завантаження навчальних даних).

Створюється клас NeuralNet, який містить трирівневу повнозв'язну мережу (Linear шари), що приймає вхідні дані, передає їх через приховані шари, і на виході видає ймовірності класів (лістинг 3.1).

Лістинг 3.1 — Клас NeuralNet

```
class NeuralNet(nn.Module):
  def __init__(self, input_size, hidden_size, num_classes):
      super(NeuralNet, self).__init__()
      self.l1 = nn.Linear(input_size, hidden_size)
      self.l2 = nn.Linear(hidden_size, hidden_size)
      self.l3 = nn.Linear(hidden_size, num_classes)
      self.relu = nn.ReLU()
  def forward(self, x):
      out = self.l1(x)
      out = self.relu(out)
      out = self.l2(out)
      out = self.relu(out)
      out = self.l3(out)
      return out
```

Далі відбувається обробка даних, де функції: tokenize (розбиває речення на слова), stem (зменшує слова до їх коренів), bag_of_words (створює “мішок

слів” - масив, де 1 означає присутність слова у запитання, а 0 - його відсутність) (лістинг 3.2).

Лістинг 3.2 — Функції обробки даних

```
def tokenize(sentence):
  return nltk.word_tokenize(sentence)
def stem(word):
  return stemmer.stem(word.lower())
def bag_of_words(tokenized_sentence, words):
  sentence_words = [stem(word) for word in tokenized_sentence]
  bag = np.zeros(len(words), dtype=np.float32)
  for idx, w in enumerate(words):
    if w in sentence_words:
      bag[idx] = 1
  return bag
```

Вібувається токенізація слів з dataset і створюються масиви X_train і y_train, де X_train містить "мішки слів" для кожного запитання, а y_train — індекси відповідних тегів (лістинг 3.3).

Лістинг 3.3 — Токенізація слів та створення навчальних масивів

```
all_words = []
tags = []
xy = []
for intent in intents['intents']:
  tag = intent['tag']
  tags.append(tag)
  for pattern in intent['patterns']:
    w = tokenize(pattern)
    all_words.extend(w)
    xy.append((w, tag))
ignore_words = ['?', '.', '!']
```

Продовження. Лістинг 3.3

```
all_words = [stem(w) for w in all_words if w not in ignore_words]
all_words = sorted(set(all_words))
tags = sorted(set(tags))
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
  bag = bag_of_words(pattern_sentence, all_words)
  X_train.append(bag)
  label = tags.index(tag)
  y_train.append(label)
X_train = np.array(X_train)
y_train = np.array(y_train)
```

Далі відбувається навчання моделі, де визначаються параметри навчання, зокрема кількість епох (num_epochs), розмір пакету (batch_size), і швидкість навчання (learning_rate) (лістинг 3.4).

Лістинг 3.4 — Навчання моделі

```
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
class ChatDataset(Dataset):
  def __init__(self):
    self.n_samples = len(X_train)
    self.x_data = X_train
    self.y_data = y_train
  def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]
  def __len__(self):
    return self.n_samples
```

Продовження. Лістинг 3.4

```
dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset,
                          batch_size=batch_size,
                          shuffle=True,
                          num_workers=0)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = NeuralNet(input_size, hidden_size, output_size).to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
for epoch in range(num_epochs):
  for (words, labels) in train_loader:
      words = words.to(device)
      labels = labels.to(dtype=torch.long).to(device)
      outputs = model(words)
      loss = criterion(outputs, labels)
      optimizer.zero_grad()
      loss.backward()
      optimizer.step()
  if (epoch+1) % 100 == 0:
      print (f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
print(f'final loss: {loss.item():.4f}')
```

Після навчання модель та важливі параметри зберігаються у файлі data.pth. Цей файл потім можна завантажити, і використовувати вже навчену модель.

Отже у цьому коді використовується багатошарова нейронна мережа. Це мережа, яка складається з вхідного шару, одного прихованого шару, і вихідного шару. Тому модель має наступні компоненти:

— вихідний шар (приймає вхідний вектор input_size, який відповідає кількості унікальних слів, визначених у наборі навчальних даних, цей шар передає інформацію до прихованого шару;

— прихований шар (шар з кількістю нейронів hidden_size, визначеного гіперпараметром);

— вихідний шар (має num_classes нейронів, що відповідає кількості можливих класів (tags), вихід цього шару представляє ймовірності для кожного класу, з якими модель передбачає, що вхід належить до конкретного класу).

3.2 Структурування проєкту

Так як при розробці проекту використовується клієнт-серверна архітектура, важливо розділити back-end і front-end частини, що сприяє впорядкованості та зручності в підтримці. Дотримання "best practices", це набори методів і технік, забезпечує оптимальні результати, підвищує ефективність та допомагає створити впорядковану структуру.

Чітко структурований проект полегшує орієнтацію, адже всі файли, код і ресурси розташовані логічно, що дозволяє швидше знаходити потрібні елементи. У розробці програмного забезпечення зазвичай використовуються різні середовища, такі як середовища розробки, тестування і продакшну. Структура папок допомагає чітко розмежувати код додатка від тестів і скриптів для розгортання, що робить процес розгортання більш безпечним та ефективним.

Загалом, грамотна структура полегшує командну роботу, спрощує розгортання та управління проектом. Однак, залежно від особливостей проекту, таких як обсяг, вибір фреймворка, мова програмування, можуть використовуватися різні підходи до організації папок. Далі буде наведено огляд структури, яка використовуються для поточного проєкту, зокрема для back-end і front-end частин.

Структура back-end частини зображена на рисунку 3.2.

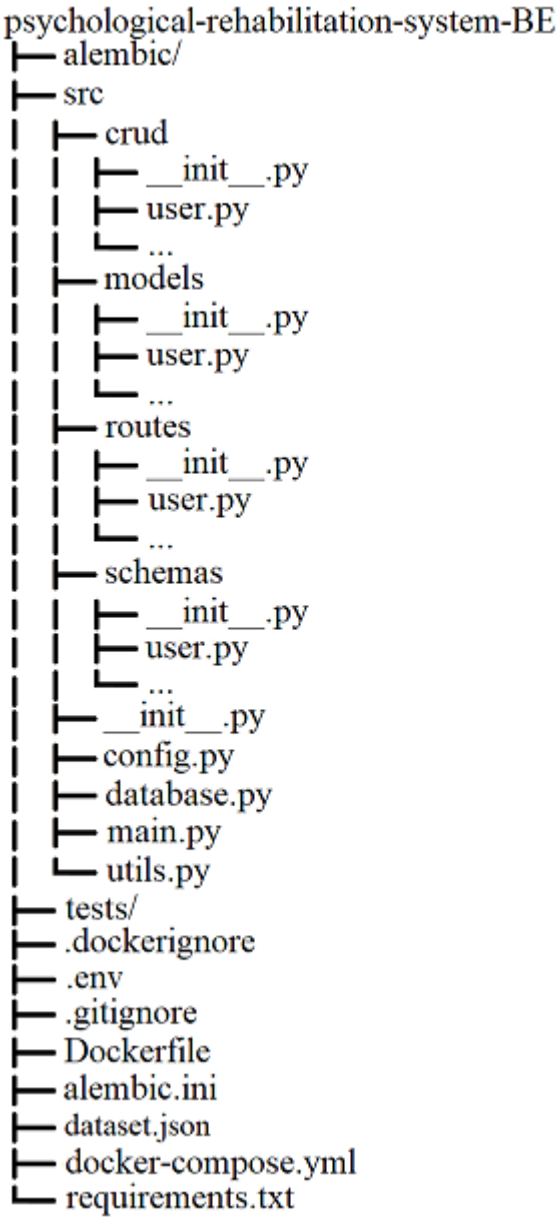

Рисунок 3.2 — Файлова структура back-end частини проєкту

Всі каталоги містяться в src папці:

— src/ — найвищий рівень програми, що містить загальні моделі, конфігурації, константи тощо.

— src/main.py — корінь проекту, тут ініціалізується додаток FastAPI.

Всі реалізації кінцевих точок, схем, моделей тощо, мають власний пакет:

— routers/ — пакет, що містить всі кінцеві точки;

— crud/ — в цьому пакеті реалізована бізнес-логіка, а саме запити до бази даних;

— models/ — призначений для створення моделей бази даних;

— schemas/ — для опису pydantic моделей;

Файли та які розміщуються в каталозі src:

— config.py — для збереження конфігураційних параметрів або налаштувань;

— database.py — речі пов’язані з підключенням до бази даних;

— utils.py — функції які виконують не бізнес-логіку;

— Файли та папки які за межами src:

— alembic/ — тут відбувається налаштування міграцій, та зберігання версії змін, які застосовуються до схем бази даних;

— tests/ — містить додаткові підпапки, такі як «auth», «gets», «posts», «patchs», «deletes» які відповідають за різні категорії тестів відповідно до функціональності, або типу HTTP-запиту;

— Dockerfile і docker-composeu.yml — використовуються для контейнеризації проекту;

— .env — використовується для збереження конфіденційних змінних середовища проекту;

requirements.txt — файл, в якому міститься список залежностей для проекту Python.

Структура front-end частини зображена на рисунку 3.3.

Дана структура розроблена з метою забезпечити масштабованість, зручність обслуговування та простоту навігації.

— src — папка, що містить код програми React;

— assets — містить зображення які використовуються в проекті;

— components — містить компоненти програми;

— styles — містить CSS стилі;

— utils — містить службові функції

— views — містить компоненти вищого рівня, відповідальні за відтворення певних сторінок програми;

— App.js — файл що містить кореневий компонент програми;

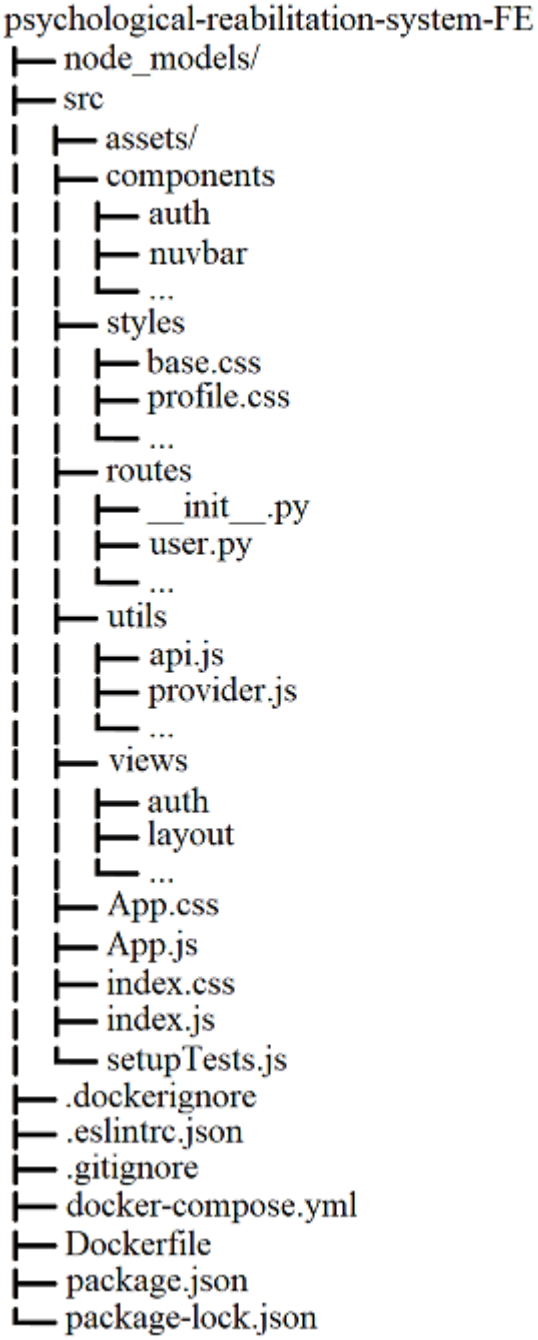

Рисунок 3.3 — Файлова структура front-end частини проєкту

— index.js — файл який відповідає за рендеринг корневого компонента та його монтування до DOM;

— setupTests.js — файл в якому налаштовується тестове середовище;

— docker-composeu.yml і Dockerfile — використовуються для контейнеризації проекту;

— package.json — файл в якому перераховані залежності для проекту.

Загалом ці структури допомагають забезпечити організацію та підтримку розробки великомасштабних програм зі складним набором функціональностей і значною кодовою базою. Розділення на рівні back-end та front-end дозволяє працювати паралельно над різними частинами проекту.

3.3 Реалізація back-end частини

Для розробки back-end частини цього проєкту використовується Python разом із фреймворком FastAPI. Важливою практикою в Python-проєктах є створення віртуального середовища, яке забезпечує зберігання всіх необхідних бібліотек та інструментів, що використовуються під час розробки. Залежності вказуються у файлі requirements.txt (рисунок 3.4), і можуть бути встановлені однією командою: pip install -r requirements.txt.

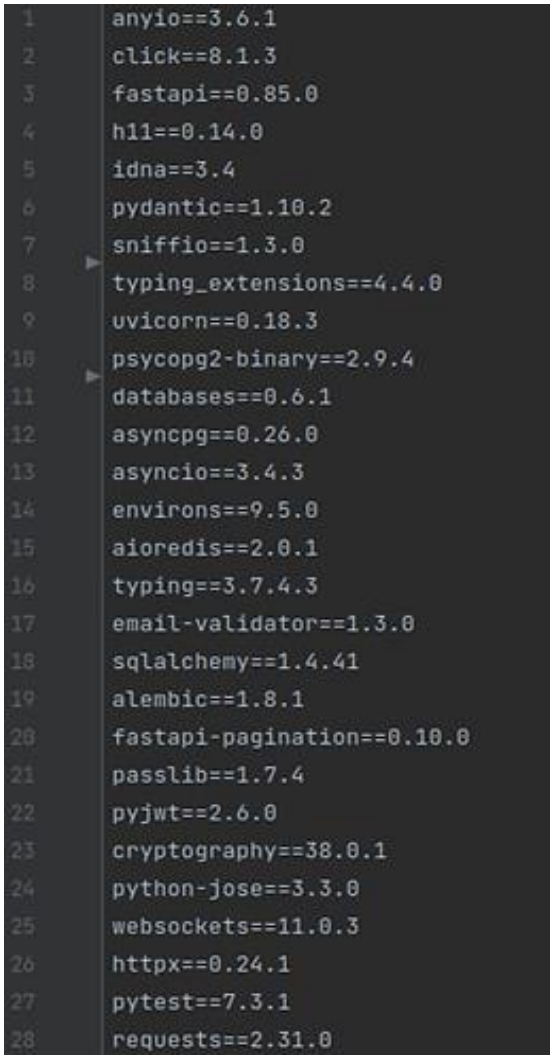

Рисунок 3.4 — Зовнішній вигляд файла requirements.txt, з переліком бібліотек

Для запуску back-end частини додатку застосовується команда: uvicorn src.main:app. Щоб спростити процес розгортання додатку на сервері, а також для зручності роботи з різними мікросервісами, такими як база даних, було вирішено використовувати Docker. Docker дозволяє автоматизувати розгортання й керування додатками завдяки контейнеризації.

В Dockerfile прописано базовий образ (у нашому випадку - Python), змінні середовища, залежності та команда для запуску додатку. У файлі docker-compose.yml визначені та з’єднані контейнери, зокрема сервіси «web» та «database».

Проєкт використовує PostgreSQL як базу даних, а також бібліотеку SQLAlchemy, яка дозволяє описувати структуру бази даних та методи взаємодії з нею на Python, без використання SQL.

У процесі розробки виникає необхідність вносити зміни до структури бази даних: додавати нові таблиці, змінювати або видаляти існуючі. Для управління версіями бази даних у цьому проєкті використовується інструмент для міграцій Alembic. Щоб забезпечити коректну роботу Alembic, було налаштовано файли alembic/env.py та alembic.ini, які автоматично створюються після виконання команди ініціалізації: alembic init alembic.

На прикладі моделі таблиці «User», зображеному у лістингу 3.5, показано, як налаштовуються такі моделі в проєкті.

Лістинг 3.5 — Модель таблиці «User»

```
class User(Base):
    __tablename__ = "users"
    id = Column(Integer, primary_key=True)
    username = Column(String, default=None)
    full_name = Column(String, default=None)
    email = Column(String, unique=True, index=True, nullable=False)
    password = Column(String)
```

Продовження. Лістинг 3.5

```
date_of_birth = Column(Date)
created_at = Column(DateTime(timezone=True), default=func.now())
```

Після в терміналі потрібно виконати команди: *alembic revision — autogenerate -m "created users table"* і *alembic upgrade head*. І автоматично створюється «.py» файл в alembic/versions/

Сам додаток створюється за допомогою app = FastAPI(). Об'єкт app представляє центральну частину додатку, яка обробляє HTTP-запити, та відповідає на них, забезпечує основні функціональні можливості, такі як маршрутизація, обробка запитів та відповідей тощо.

Щоб визначити всі маршрутизатори, їх потрібно додати до app за допомогою методу app.include_router(router). router — це окремий маршрут, який створюється за допомогою APIRouter() і налаштовується з певними параметрами, одним із головних параметрів є префікс, що відповідає за певний маршрут.

Для зручності розробки клієнтської частини потрібна документація, яка описує інформацію про доступні маршрути, їх параметри, типи даних, схему запиту та відповіді. Така як використовується FastAPI, то він надає підтримку для генерації документації API з використанням специфікації OpenAPI (раніше відомої як Swagger) (рисунок 3.5).

Рисунок 3.5 — Сторінка документації для API

Під час розробки було реалізовано ряд кінцевих точок. Перелік деяких маршрутів API:

— auth — відповідає за реєстрацію, вхід, вихід, а також отримання інформації про поточного користувача;

— users — відповідає за взаємодію та управління користувачами;

— chat — забезпечують доступ до чат-бота;

— surveys — відповідає за проведення опитувань серед користувачів для ідентифікації потенційних психологічних проблем;

— recommendations — відповідає за отримання рекомендацій на основі результатів опитувань та історії взаємодій користувача;

— specialists — відповідає за взаємодію з доступними психологічними спеціалістами;

— analytics — отримання статистичних даних щодо активності та настрою користувачів;

— admin — відповідає за дії адміністратора.

3.4 Реалізація front-end частини

Розробка front-end частини здійснюється мовою програмування JavaScript із використанням бібліотеки React.

Для початку розробки використовується бібліотека «create-react-app», яка полегшує створення додатків, а також забезпечує виконання завдань, таких як тестування, збірка та розгортання. Виконуючи команду «npx create-react-app», створюється новий проєкт із усіма необхідними залежностями. Для запуску додатка використовується команда «npm start». У разі внесення змін до файлів проєкту, додаток автоматично перебудовується.

Проте, як зазначалося раніше, для зручності у розгортанні та керуванні додатком використовується програмне забезпечення Docker, яке забезпечує автоматизацію цих процесів завдяки підтримці контейнеризації.

Для взаємодії з серверною частиною застосовується бібліотека axios, яка дозволяє виконувати HTTP-запити безпосередньо з браузера.

Після налаштування всіх залежностей було розпочато розробку сторінок системи. Зокрема, реалізовано сторінку реєстрації зображену на рисунку 3.6 та авторизації зображену на рисунку 3.7, що забезпечують доступ до системи лише для авторизованих користувачів.

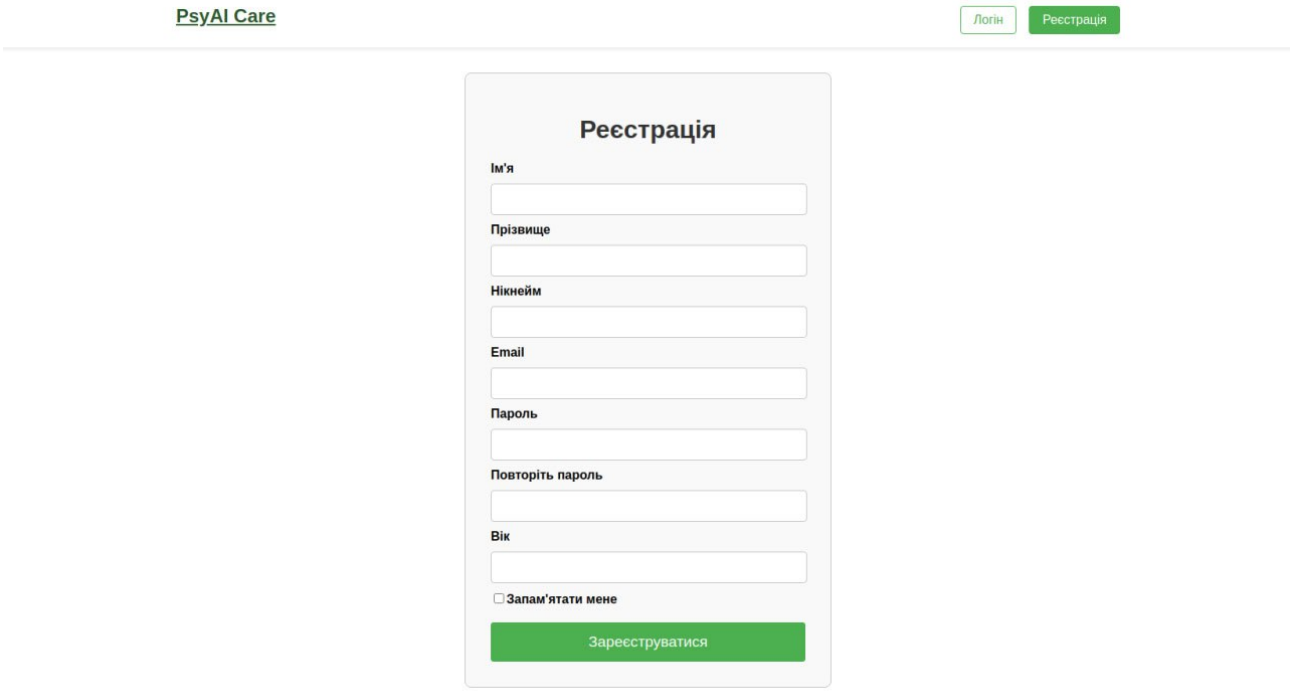

Рисунок 3.6 — Сторінка реєстрації

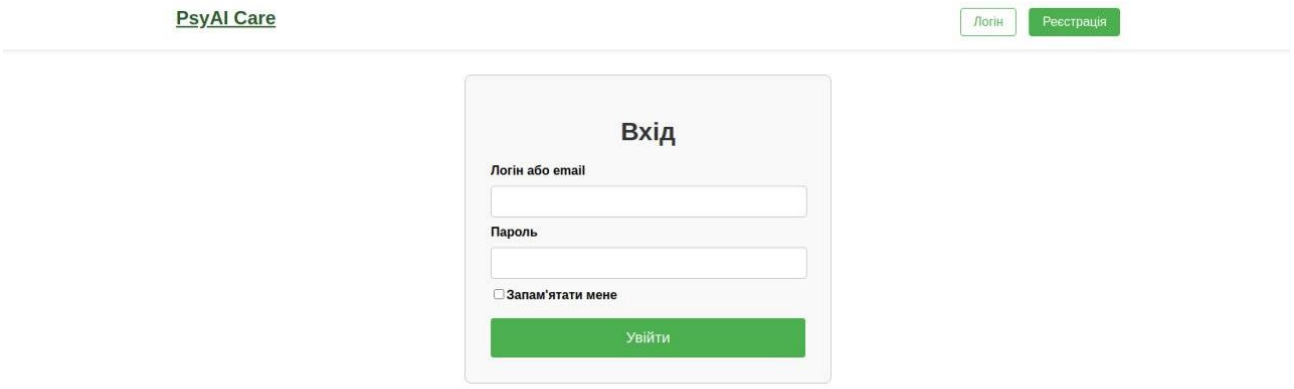

Рисунок 3.7 — Сторінка входу

На головній сторінці програми, що зображена на рисунку 3.8, після успішної авторизації користувач отримує доступ до персоналізованої інформації

про свій прогрес та статистику, пов’язану з психологічним станом і досягненнями. Ця сторінка служить основним інформаційним центром, що відображає ключові дані, корисні для користувача.

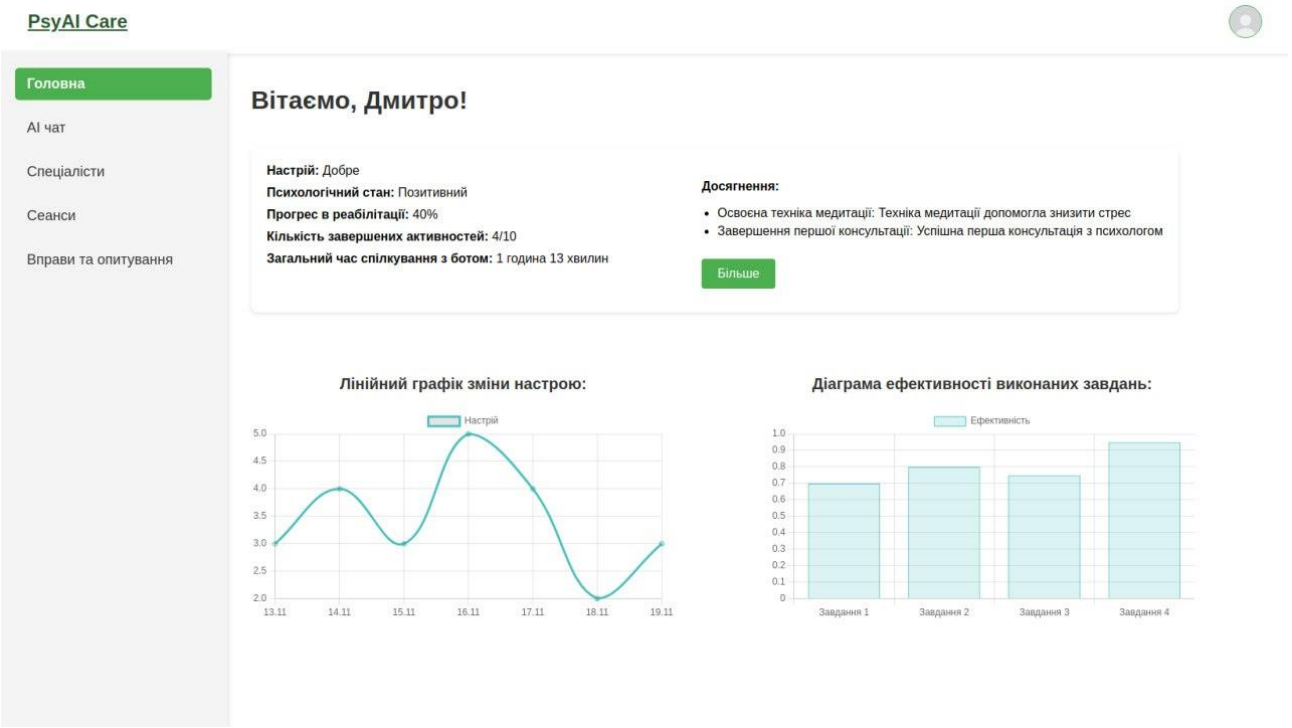

Рисунок 3.8 — Головна сторінка проетку (статистичні дані)

Окрім панелі статистики головний акцент робиться на бічній панелі, яка дозволяє зручно переключатися між сторінками де можна перейти до до таких сторінок, як «AI-чат», «Пошук спеціаліста», «Активності», і т. д.

Перейшовши за кнопкою "AI-чат", користувач відкриває сторінку спілкування з чатботом, який використовує алгоритми штучного інтелекту для надання інформації, порад та рекомендацій, що зображено на рисунку 3.9. Користувач може поставити питання, пов’язані з психологічним здоров’ям, а AI в свою чергу може запропонувати шляхи вирішення проблем.

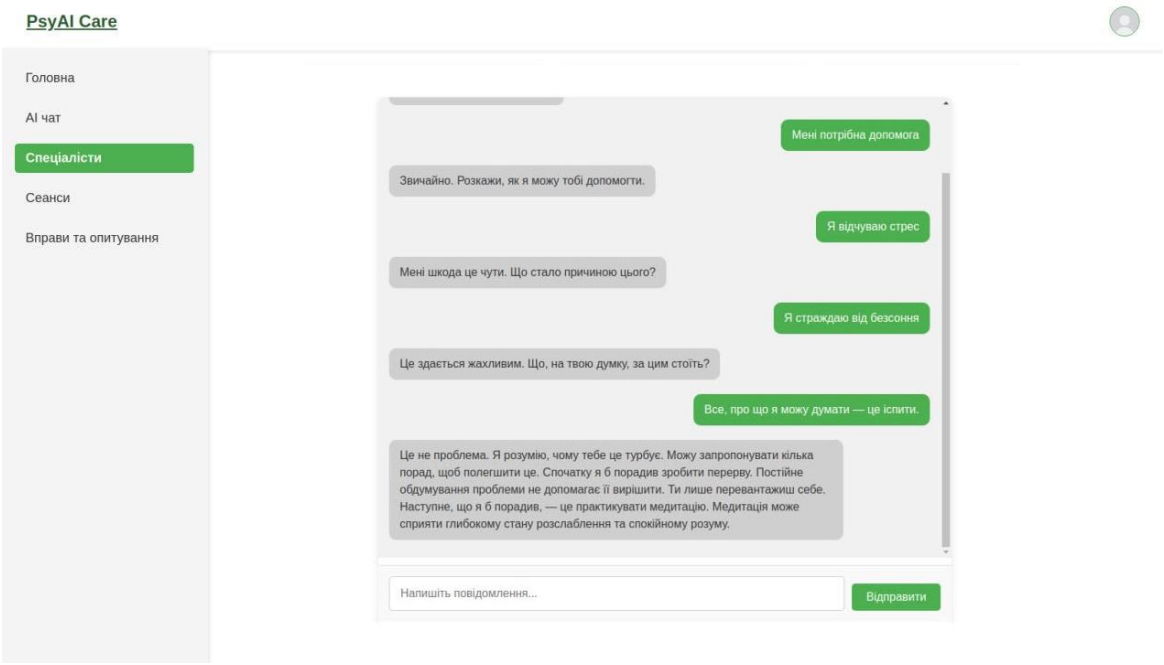

Рисунок 3.9 — Інтерфейс AI-чату

Сторінка «Спеціалісти» зображена на рисунку 3.10, вона є не менш важливим розділом веб-додатку, який допомагає користувачам знайти кваліфікованих психологів для вирішення проблем, пов’язаних із психологічним здоров’ям. Ця сторінка надає можливість переглядати список спеціалістів, шукати їх за різними критеріями, переглядати профілі та записуватись на консультації. Штучний інтелект дає можливість підібрати спеціаліста, який краще відповідає потребам користувача.

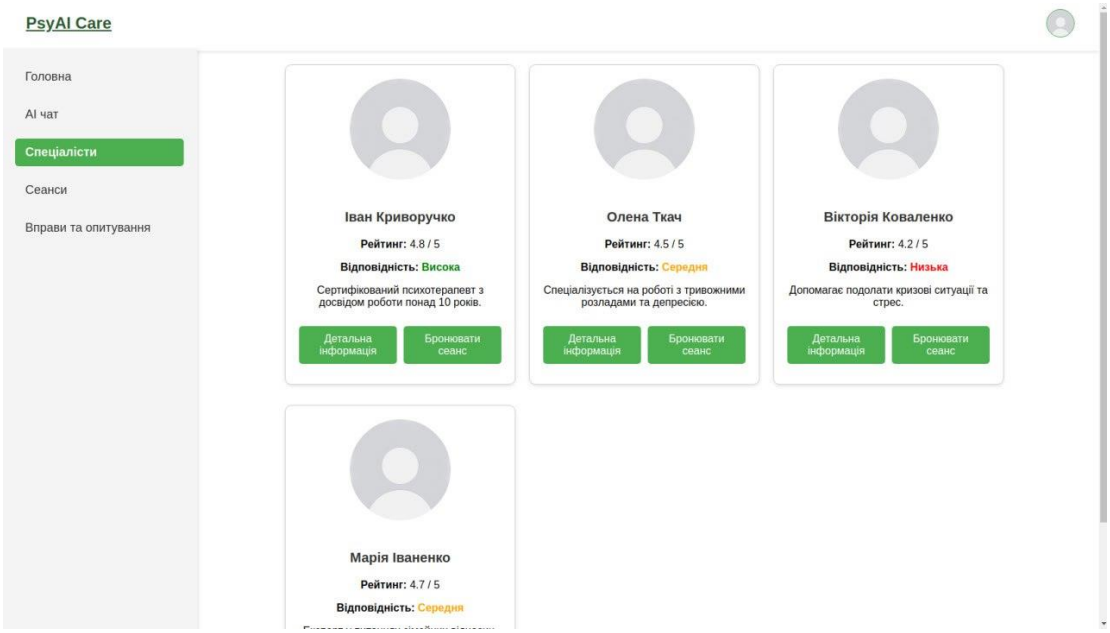

Рисунок 3.10 — Сторінка з переглядом спеціалістів

Для отримання детальної інформації про спеціаліста, можна переглянути профіль, сторінка профіля спеціаліста. Також, є можливість забронювати сеанс, рисунок 3.11.

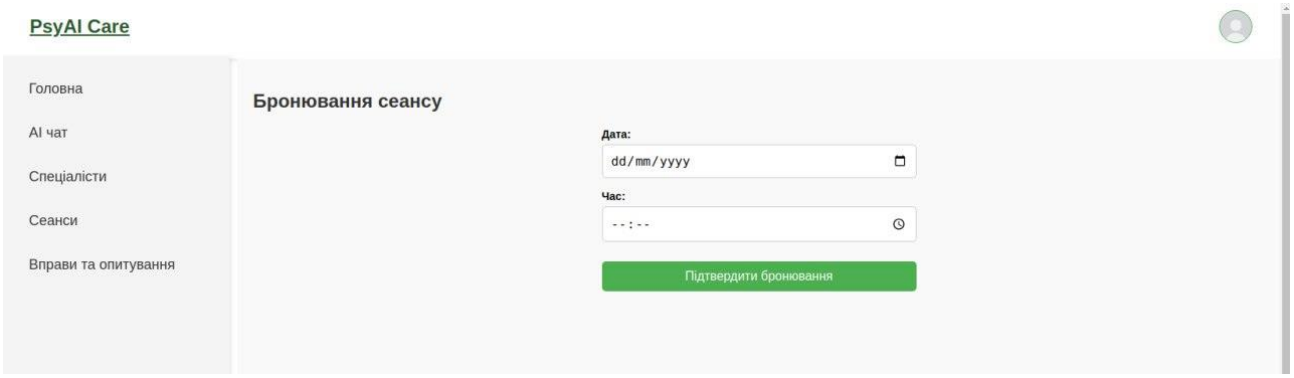

Рисунок 3.11 — Бронювання сеансу

Якщо сеанси було вдало забронювано, їх можна переглянути на сторінці «Сеанси», щоб розпочати розмову, потрібно перейти в Google Meet, кнопка, стає активною, за 10 хвилин до початку сеансу, сторінка з запланованими сеансами зображена на рисунку 3.12.

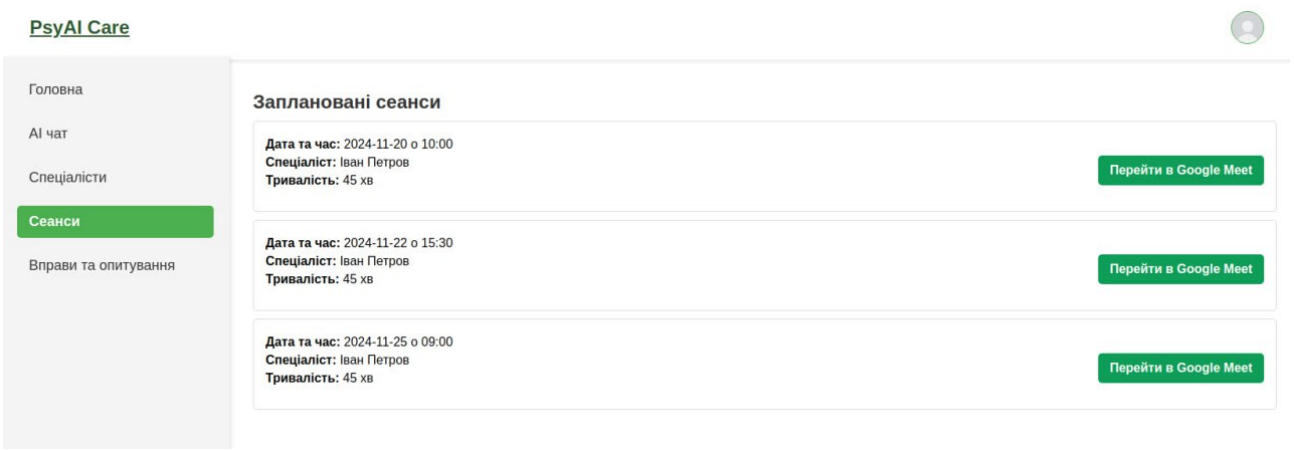

Рисунок 3.12 — Сторінка запланованих сеансів

Сторінка «Активності» відповідає за різні вправи, а також опитування, які є за замовчуванням, так і створені окремими спеціалістами для конктетної особи, інтерфейс сторінки зображено на рисунку 3.13.

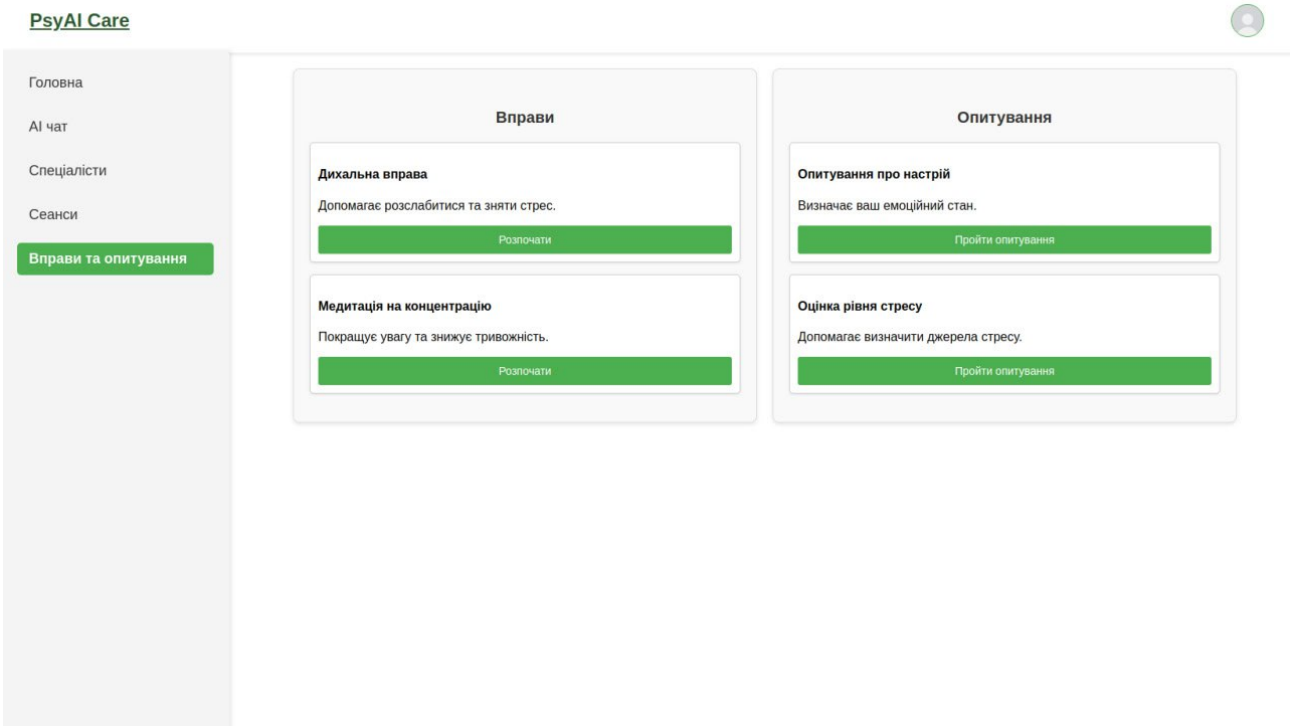

Рисунок 3.13 — Сторінка з активностями

4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Під час розробки програмного продукту є критично важливим перевірити його працездатність. Цей етап включає проведення спеціальних тестів для перевірки працездатності продукту.

4.1 Тесування beck-end частини

У процесі розробки програмного забезпечення використовуються різні види тестування для перевірки якості продукту. В даній системі застосовуються три основні види тестування:

— юніт-тестування (Unit Testing) — перевірка окремих частин або модулів програми для забезпечення їхньої коректної роботи;

— API тестування — перевірка роботи програмного інтерфейсу (API), що включає оцінку його функціональності, надійності та взаємодії з іншими компонентами;

— функціональне тестування (Functional Testing) — тестування функцій та модулів системи, яке визначає, чи відповідає їхня робота заданим специфікаціям.

Для юніт-тестування зазвичай використовуються спеціалізовані фреймворки. У цьому проєкті використовується Pytest, один із найпопулярніших фреймворків для Python. Pytest має простий синтаксис, дозволяє автоматично знаходити тести, створювати звіти про їх виконання, а також підтримує такі функції, як параметризовані тести, фікстури (fixtures) і мокування (mocking), що значно підвищує гнучкість і ефективність тестування.

Розглянемо приклад тестування реєстрації нового користувача (лістинг 4.1).

Лістинг 4.1 — Приклад тестування реєстрації користувача.

```
def test_register_user(app):
    data = {
        "username": "test",
```

Продовження. Лістинг 4.1

```
    "full_name": "user",
    "email": "testuser@gmail.com",
    "date_of_birth": "19.06.2002"
    "password": "testing123"
    "confirm_password": "testing123"
}
response = app.post("/auth/register", json.dumps(data))
assert response.status_code == 200
assert response.json()["token_type"] == "bearer"
```

В змінній data зберігаються дані, які передаються у тілі запиту.

Після виконання POST запиту, результат зберігається в змінну response. Тест має кілька перевірок (assertions), що дозволяють перевірити правильність відповіді сервера:

— рядок assert response.status_code == 200 перевіряє, чи код статусу відповіді дорівнює 200, що означає успішну реєстрацію, якщо код статусу не дорівнює 200, тест видасть помилку;

— рядок assert response.json()["token_type"] == "bearer" перевіряє, чи поле "token_type" у відповіді містить значення "bearer", якщо значення не співпадає, або воно відсутнє, тест видасть помилку.

Далі зображено результат де тест виконано успішно (рис. 4.1) та результат який видає помилку (рисунок 4.2).

До API-тестування належить також перевірка з використанням інструменту Postman. Postman — це один із найпоширеніших інструментів для розробників, що дозволяє виконувати HTTP-запити до веб-серверів і тестувати різні аспекти API. Він забезпечує можливість вручну перевіряти коректність відповіді сервера та оцінювати функціональність API.

API-тестування дає змогу перевірити, чи відповідає API встановленій специфікації, чи повертає очікувані дані, які статус-коди надсилає, а також як

обробляються помилки. Це допомагає переконатися в коректній роботі API та його здатності взаємодіяти з іншими компонентами системи (рисунок 4.3).

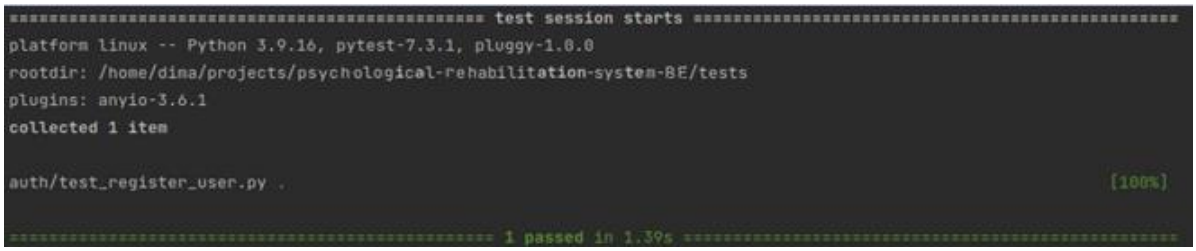

Рисунок 4.1 — Виконання тесту успішне

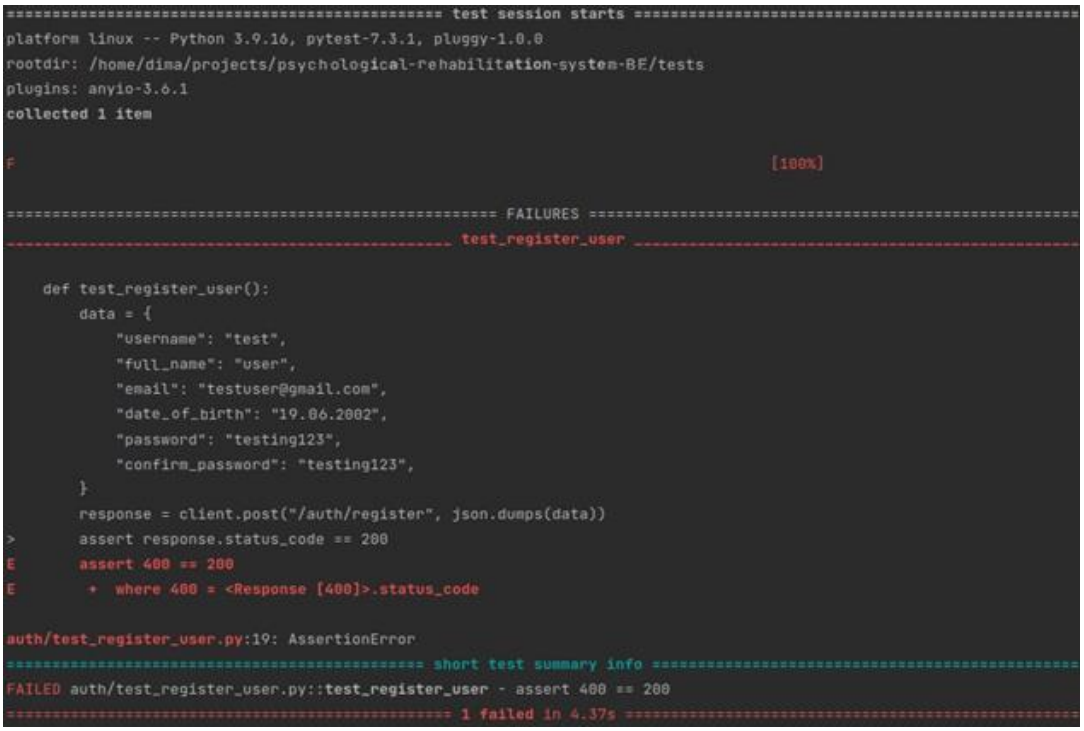

Рисунок 4.2 — Виконання тесту з помилкою

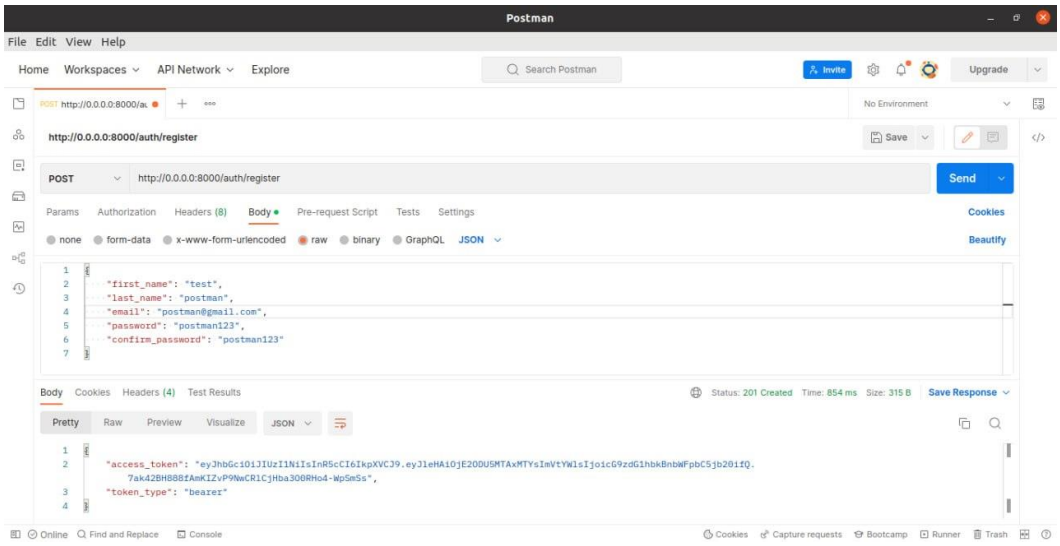

Рисунок 4.3 — Тестування за допомогою Postman

4.2 Тестування front-end частини

Функціональне тестування веб-додатків включає використання бібліотеки React Testing Library для перевірки компонентів, створених із використанням React. Ця бібліотека забезпечує інструменти для тестування компонентів з точки зору їхнього відображення та взаємодії з користувачем. Завдяки React Testing Library можна легко створювати тести, які моделюють дії користувача, такі як натискання кнопок, заповнення форм, навігація між сторінками тощо, щоб перевірити, чи працює додаток коректно.

Основна ідея React Testing Library — це тестування додатка так, як його бачить користувач. Бібліотека фокусується на перевірці взаємодії з компонентами та їхньої функціональності, а не на деталях реалізації чи внутрішніх станах компонентів. Це підхід дозволяє створювати надійні й незалежні тести, які максимально відповідають реальному використанню додатка.

Розглянемо приклад тестування компонента входу (SignIn) (лістинг 4.2).

Лістинг 4.2 — Приклад тестування компонента входу

```
import React from 'react';
import { render, screen, fireEvent } from '@testing-library/react';
import { SignIn } from './SignIn';
describe('SignIn', () => {
  test('renders SignIn component', () => {
    render(<SignIn />);
    const loginOrEmailInput = screen.getByPlaceholderText("Логін або email");
    const passwordInput = screen.getByPlaceholderText("Пароль");
    const submitButton = screen.getByRole('button', { name: /Увійти/i });
    expect(emailInput).toBeInTheDocument();
    expect(passwordInput).toBeInTheDocument();
    expect(submitButton).toBeInTheDocument();
  });
```

Продовження. Лістинг 4.2

```
 test('input fields update state', () => {
   render(<SignIn />);
   const emailInput = screen.getByPlaceholderText("E-mail");
   const passwordInput = screen.getByPlaceholderText("Пароль");
   fireEvent.change(emailInput, { target: { value: 'john.doe@example.com' } });
   fireEvent.change(passwordInput, { target: { value: 'password123' } });
   expect(emailInput.value).toBe('john.doe@example.com');
   expect(passwordInput.value).toBe('password123');
 });
 test('form submission', async () => {
   const mockPost = jest.fn();
   const mockLogin = jest.fn();
   jest.mock('../api/http', () => ({ $api: { post: mockPost } }));
   jest.mock('../provider/TokenProvider', () => ({ login: mockLogin }));
   jest.mock('react-router-dom', () => ({ useNavigate: () => mockNavigate }));
   render(<SignIn />);
   const submitButton = screen.getByRole('button', { name: /Увійти/i });
   fireEvent.click(submitButton);
   expect(mockPost).toHaveBeenCalledWith(
     '/auth/login',
     expect.objectContaining({
       email: '',
       password: '',
     }),
   );
   expect(mockLogin).toHaveBeenCalledTimes(1);
   expect(mockNavigate).toHaveBeenCalledTimes(1);
 });
});
```

Це тестування включає наступні перевірки:

— перевіряє, чи відображаються всі необхідні елементи форми;

— перевіряє, чи оновлюється стан компонента при введенні значень в поля вводу;

— перевіряє, чи викликається функція post з правильними параметрами при натисканні кнопки "Увійти";

— перевіряє, чи викликається функція login після успішного входу;

— перевіряє, чи викликається функція navigate для переадресації користувача після успішного входу.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту науковотехнічної розробки

Мета проведення комерційного і технологічного аудиту полягає у визначенні науково-технічного рівня та комерційного потенціалу розробки, що є результатом науково-технічної діяльності. Комерційний аудит спрямований на оцінку потенціалу застосування методів і засобів, розроблених у рамках системи адаптивного тестування знань.

Комерційний потенціал інвестицій буде оцінюватись відповідно до дванадцяти критеріїв, наведених у таблиці 5.1.

Таблиця 5.1 — Оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірнісь концепції не підтверджеа	Концепція підтверджеа експертними висновками	Концепція підтверджена розрахунками	Концепця перевіреа на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогі	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 5.1

Технічна здійсненність концепції:					
5	Експлуатацій н і витрати значно вищі, ніж в аналогів	Експлуатаційн і витрати дещо вищі, ніж в аналогів	Експлуатаційі витрати на рівні експлуатаційни х витрат аналогів	Експлуатаці йні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансуванн я є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовують ся у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовують ся у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 5.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

На основі таблиці різні експерти, в данному випадку керівник магістерської роботи та інші незалежні експерти з різних профілів, визначають різні результати. Результати цієї оцінки комерційного потенціалу узагальнено у таблиці 5.2.

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Добровольська Н. В.	Коваленко О. О.	Крупельницький Л. В.
1	3	3	2
Ринкові переваги (недоліки):			
2	2	2	2
3	3	3	3
4	3	3	4
5	4	3	4
Ринкові перспективи			
6	3	2	3
7	3	3	3
Практична здійсненність			
8	3	3	2
9	2	2	2
10	2	2	2
11	4	3	4
12	3	3	3
Сума балів	$СБ_1 = 35$	$СБ_2 = 32$	$СБ_3 = 34$
Середньоарифметична сума балів $\overline{СБ}$	$СБ_С = \frac{\sum_1^3 СБ_i}{3} = \frac{35 + 32 + 34}{3} = 33{,}6$		

Відповідно до таблиці 5.2, а також відповідно до рекомендацій, наведених у таблиці 5.3, можна зробити висновок про рівень потенціалу комерційного розвитку.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 — 10	Низький
11 — 20	Нижче середнього
21 — 30	Середній
31 — 40	Вище середнього
41 — 50	Високий

Рівень комерційного потенціалу розробки, становить 33,6 балів, що відповідає рівню «вище середнього».

Серед аналогів можна виділити вебзастосунки Wysa та Youper. Платформа Wysa об’єднує групу фахівців, які пропонують підтримку через чатботи, орієнтовані на психічне здоров'я, що дозволяє користувачам отримувати допомогу за допомогою інтерактивних бесід. Youper використовує штучний інтелект для надання психологічних рекомендацій та проведення психоаналітичних опитувань, що дають можливість отримати висновки на основі відповідей користувачів.

Порівння розробки з її аналогами приведено в таблиці 5.4.

Таблиця 5.4 — Порівняння характеристик розробки із аналогами

Показники	Розробка	Аналог1	Аналог2
Функціонал	9	8	7
Швидкодія	8	9	9
Надійність	7	8	8
Метод розповсюдження	6	8	8
Інтерфейс, простота використання	8	9	8

Продукт буде просуватися за допомогою реклами в соціальних мережах, пошукових системах та багатьох інших джерелах Інтернету. Використовуючи

аналітику цих сервісів, можна буде націлити рекламу на цільову групу захисників інформації.

Наукова новизна дослідження полягає у впровадженні штучного інтелекту в інформаційні системи, для виявлення психологічних проблем користувачів, та наданя рекомендацій їх вирішення, на основі інтерактивного AI-чату з елементами самонавчання.

Даний рівень було досягнуто за рахунок покращення та/або розширення функціональних можливостей нової науково-технічної розробки порівняно з аналогічними розробками, існуючими в цей час на ринку.

5.2 Розрахунок витрат на здійснення науково-технічної розробки

У магістерській роботі розглядається програмне забезпечення для психологічної реабілітації та оцінки стану користувачів за допомогою штучного інтелекту. Основну частину витрат складають витрати пов’язані з розробкою, а не виробництвом та відтворенням. Відповідно розрахунки потребують специфічного підходу.

5.2.1 Витрати на оплату праці

Основна заробітна плата розробників, що працюють над проектом, визначена у формулі 5.1:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (5.1)$$

де k — кількість посад дослідників, залучених до процесу досліджень;

$М_{ni}$ — місячний посадовий оклад конкретного дослідника, грн;

$Т_р$ — середня кількість робочих днів в місяці, $Т_р$ — 20 днів;

t_i— кількість днів роботи конкретного дослідника, днів.

Над створенням розробки працював менеджер проекту та інженер програмного забезпечення, тому ми виконаємо для них усі необхідні розрахунки, і вносимо їх до таблиці 5.5.

$$З_{о.к.} = \frac{14500 \cdot 8}{20} = 5800 \text{ (грн)}.$$

$$З_{о.в.} = \frac{10000 \cdot 32}{20} = 21800 \text{ (грн)}.$$

Додаткова винагорода ($З_{дод.}$) усіх розробників та працівників, які брали участь у цьому етапі роботи, обчислюється як 10 ... 12% від суми основної заробітної плати дослідників та робітників за формулою 5.2:

$$З_{дод.} = (З_о + З_р) \cdot \frac{Н_{дод.}}{100\%}, \quad (5.2)$$

де $Н_{дод.}$ — норма нарахування додаткової заробітної плати.

$$З_{дод.к.} = \frac{12 \cdot 5800}{100} = 696 \text{ (грн)},$$

$$З_{дод.в.} = \frac{12 \cdot 16000}{100} = 1920 \text{ (грн)},$$

$$З_{дод.} = З_{дод.к.} + З_{дод.в.} = 2616 \text{ (грн)}.$$

Витрати на основну заробітну плату робітників за відповідними найменуваннями робіт відсутні, тобто З$р$ = 0.

Таблиця 5.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	14500	725	9	5800
Старший інженер-програміст	10000	500	32	16000
Всього				21800

5.3.2 Соціальні відрахування

Заробітна плата робітників відсутня, тому Зр = 0. Нарахування на заробітну плату дослідників та нарахування на заробітну плату працівників, які брали участь у цьому етапі роботи, розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 5.3:

$$З_{н.} = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (5.3)$$

де $Н_{зп}$ — норма нарахування на заробітну плату.

$$З_{н.} = (21800 + 0 + 2616) \cdot \frac{22\%}{100\%} = 5371{,}52$$

5.2.3 Сировина та матеріали

Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою 5.4:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj} \quad (5.4)$$

де $Н_j$ — кількість матеріалу j-го виду, шт.;

n — кількість видів матеріалу.

$Ц_j$ — ціна матеріалу j-го виду, грн;

$К_j$ — коефіцієнт транспортних витрат, Кj = (1,1…1,15), обираємо Кj 1,15;

B_j — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.

Результати розрахунків занесено до таблиці 5.6.

Таблиця 4.6 — Витрати на матеріали

Найменування комплектуючих	Ціна за 1 штуку, грн	Кількість матеріалу, штук	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір	175,00	1	0,000	0,00	175,00
Канцелярське приладдя	180,00	1	0,000	0,00	180,00
Flesh-пам'ять 16 GB	150,00	1	0,000	0,00	150,00
Всього					505,00

5.2.4 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» включаються витрати на створення та придбання спеціалізованих програмних інструментів і застосунків, які забезпечують виконання досліджень.

Балансову вартість програмного забезпечення р озраховуємо за формулою 5.5:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \cdot С_{прг.і} \cdot К_{і}, \quad (5.13)$$

де $Ц_{іпрг}$ — ціна придбання одиниці програмного засобу даного виду, грн;

$С_{прг.і}$ — кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_{і}$ — коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо;

k — кількість найменувань програмних засобів.

Розробка програмного забезпечення для наукових досліджень потребує значних початкових інвестицій, однак, в основному в роботі використовується безкоштовне ПЗ, що є у вільному розповсюджені, або ж програми надають безкоштовний доступ для студентів.

5.2.5 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою 5.5:

$$А_{обл} = \frac{Ц_{б}}{Т_{в}} \cdot \frac{t_{вик}}{12}, \quad (5.5)$$

$$А_{ноотбук} = \frac{9800}{12} \cdot 1{,}5 = 1225 \text{ (грн)}$$

$$А_{приміщення} = \frac{20000}{240} \cdot 1{,}5 = 125 \text{ (грн)}$$

Таблиця 4.7 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання, місяців.	Амортизаційні відрахування, грн
Ноотбук	9800,00	1	1,5	1225,00
Приміщення	20000	20	1,5	125,00
Всього				1350,00

4.2.6 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) розраховують за формулою 5.6:

$$В_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_і \cdot Ц_е \cdot К_{впі}}{\eta_і}, \quad (4.6)$$

де Wyi — встановлена потужність обладнання на певному етапі розробки, кВт;

$t_і$ — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), $Ц_е$ = 4,32;

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі}$ <1; обираємо $К_{впі}$ 0,7;

$\eta_і$ — коефіцієнт корисної дії обладнання, $\eta_і$ <1.

$$В_е = \sum_{i=1}^{1} \frac{0{,}065 \cdot 256 \cdot 4{,}32 \cdot 0{,}7}{0{,}8} = 62{,}89(грн.)$$

Проведені розрахунки занесено до таблиці 5.8.

Таблиця 4.8 — Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноотбук	0,65	456	62,89
Всього			62,89

4.2.7 Службові відрядження

Під час розробки програмного забезпечення відрядження штатних працівників, працівників організацій, які працюють за договорами цивільноправового характеру, магістрів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень, не плануються.

Витрати за статтею «Службові відрядження» розраховується як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 5.7:

$$В_{св} = \left(З_о + З_p\right) \cdot \frac{Н_{св}}{100\%}, \quad (5.7)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження».

5.2.8 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» не плануються, так як у цьому не має потреби. Проте їх можна розрахувати як 30…45% від суми основної заробітної плати дослідників та робітників за формулою 5.8:

$$В_{сп} = \left(З_о + З_p\right) \cdot \frac{Н_{сп}}{100\%}, \quad (5.8)$$

де $Н_{сп}$ — норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

5.2.9 Інші витрати

«Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 5.9:

$$І_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%}, \quad (5.9)$$

де $Н_{ів}$ — норма нарахування.

$$I_{в.к.} = 5800 \cdot \frac{50\%}{100\%} = 2900\ (грн.),$$

$$I_{в.в.} = 16000 \cdot \frac{50\%}{100\%} = 8000\ (грн.),$$

$$I_{в} = I_{в.к.} + I_{в.в.} = 10900\ (грн.).$$

5.2.10 Накладні (загальновиробничі) витрати

«Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 5.10:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.10)$$

де $Н_{нзв}$ — норма нарахування.

Беремо норму нарахування 150%.

$$В_{нзв.к.} = 5800 \cdot \frac{150\%}{100\%} = 8700\ (грн.),$$

$$В_{нзв.в.} = 16000 \cdot \frac{150\%}{100\%} = 24000\ (грн.),$$

$$В_{нзв} = В_{нзв.к.} + В_{нзв.в.} = 32700\ (грн.).$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою 5.11:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (5.11)$$

$$В_{заг} = 21800 + 0 + 2616 + 5371{,}52 + 505{,}00 + 0 + 0 + 1350{,}00 + 62{,}89 + 0 + 0 + 10900 + 32700 = 75305{,}41\ (грн.)$$

Загальні витрати ЗВ на завершення науково-дослідної (науковотехнічної) роботи та оформлення її результатів розраховуються за формулою 5.12:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (5.12)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науководослідної роботи, обираємо його 0,5.

$$ЗВ = \frac{75305{,}41}{0{,}6} = 125509\ (грн.)$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У цьому підрозділі виконаємо кількісний прогноз, щоб оцінити, яку вигоду або прибуток можна отримати в майбутньому від реалізації результатів проведеної роботи.

У ринкових умовах головним позитивним наслідком для підприємства після впровадження певної розробки є збільшення чистого прибутку. Це зростання може бути виражене у теперішній вартості грошей. Підвищення чистого прибутку сприятиме залученню додаткових фінансових ресурсів, що покращить загальні фінансові показники діяльності компанії.

Виконання роботи та реалізація її результатів займає приблизно один рік. Очікується, що позитивний ефект від впровадження розробки проявиться вже в перші місяці після її впровадження.

Детально проаналізуємо очікувані результати та виконаємо їх кількісну оцінку за роками. Розрахунок приросту чистого прибутку підприємства $\Delta П_і$ для кожного року, протягом якого прогнозується отримання позитивного ефекту від реалізації розробки, проводитимемо за формулою 5.13:

$$\Delta П_і = \sum_{1}^{n} (\Delta П_я \cdot N + П_я \cdot \Delta N)_i, \qquad (5.13)$$

де $\Delta П_я$ — покращення основного якісного показника від впровадження результатів розробки у даному році;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ — основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Припустимо, що впровадження інформаційної системи психологічної реабілітації з використанням штучного інтелекту дозволить збільшити чистий прибуток підприємства на 100 грн за одну послугу. Очікується, що кількість реалізованих послуг зросте:

— протягом першого року — на 400 послуг;

— протягом другого року — ще на 800 послуг;

— протягом третього року — ще на 1200 послуг;

До впровадження системи реалізація становила 1 послугу на рік, а прибуток із однієї послуги складав 80 грн.

Необхідно спрогнозувати приріст чистого прибутку підприємства в кожному році порівняно з базовим рівнем. Приріст чистого прибутку підприємства $\Delta П_1$ у першому році складає:

$$\Delta П_1 = 80 \cdot 1 + (80 + 100) \cdot 400 = 72080 \text{ (грн.)}$$

Обчислимо збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року:

$$\Delta П_2 = 80 \cdot 1 + (80 + 100) \cdot (400 + 800) = 216080 \text{ (грн.)}$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_2 = 80 \cdot 1 + (80 + 100) \cdot (400 + 800 + 1200) = 432080 \text{ (грн.)}$$

Далі розраховується приведена вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації розробки за формулою 5.14:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.14)$$

де ΔПі — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково — технічної розробки, грн;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15. В Україні рівень інфляції за підсумком 2024 року складає 9,7%, прогнозований рівень на 2025 рік — 11%;

t — період часу (в роках) від моменту початку впровадження розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{72080}{(1+0{,}15)^1} + \frac{216080}{(1+0{,}15)^2} + \frac{432080}{(1+0{,}15)^3} = 928020{,}47$$

Отже, розрахунки показують, що відповідно прогнозуванню комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

Основними критеріями, що визначають доцільність інвестування у наукову розробку, є показники абсолютної та відносної ефективності інвестицій, а також термін їх окупності.

Якщо $E_{абс} > 0$, це свідчить про те, що реалізація наукових досліджень і впровадження їх результатів буде прибутковою. Проте це не гарантує автоматичного зацікавлення інвестора у фінансуванні даного проекту.

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію розробки. При цьому ставка дисконтування τ приймається рівною 0,1.

Для визначення величини початкових інвестицій PV, необхідних для здійснення науково-технічної розробки, скористаємося формулою 5.15:

$$PV = k_{розр} \cdot ЗВ, \qquad (5.15)$$

де $k_{розр}$ — коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково— технічної розробки. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай розр k =2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково— технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 125509 = 251018 \text{ (грн.)}$$

Тоді абсолютний економічний ефект Еабс або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації розробки обчислюється за формулою 5.16:

$$E_{абс} = ПП - PV, \quad (5.16)$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково— технічної розробки, грн;

PV — теперішня вартість початкових інвестицій, грн.

$$E_{абс} = 928020{,}47 - 251018 = 677002{,}47 \text{ (грн.)}$$

Оскільки величина економічного ефекту має велике додатне значення, це свідчить про потенційну зацікавленість інвесторів у впровадженні та комерціалізацію розробки.

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Для прийняття остаточного рішення щодо впровадження розробки та її виходу на ринок необхідно обчислити внутрішню економічну рентабельність $E_в$ або показник внутрішньої норми рентабельності (IRR, Internal Rate of Return) вкладених інвестицій. Отримане значення слід порівняти з так званою пороговою ставкою дисконтування, яка встановлює мінімальний рівень рентабельності інвестицій, нижче якого вкладення в проєкт стають економічно недоцільними.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використовується формула 5.17:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.17)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$Т_ж$ — життєвий цикл науково— технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[3]{1 + \frac{677002,47}{251018}} - 1 = 0,55 = 55\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою 5.18:

$$\tau = d + f, \tag{5.18}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d= (0,14…0,2);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0,14 + 0,5 = 0,19$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Далі розраховується період окупності інвестицій, вкладених у реалізацію проекту за формулою 5.19:

$$T_{ок} = \frac{1}{E_в}, \tag{5.19}$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0,55} = 1,81 \text{ роки}$$

Термін окупності складає 1,81 роки, що свідчить про доцільність фінансування даної наукової розробки.

5.5 Результати економічного аналізу

У цьому розділі було проведено аналіз комерційного потенціалу програмного забезпечення для адаптивного тестування знань. Результати показали, що економічний потенціал розробки перевищує середній рівень.

Також виконано прогнозування витрат на проєктування та впровадження системи, які склали 125509 грн.

Розрахунки економічної ефективності та терміну окупності інвестицій підтвердили, що розробка є фінансово вигідною та цікавою для потенційних інвесторів. Термін окупності становить 1,81 року.

ВИСНОВКИ

У результаті виконання дипломної роботи було створено інформаційну систему для психологічної реабілітації з використанням штучного інтелекту. Ця розробка спрямована на забезпечення доступної та ефективної психологічної допомоги для широкого кола користувачів. Система дозволяє проводити первинну діагностику психоемоційного стану, надавати рекомендації та сприяти покращенню психологічного стану завдяки інтеграції інноваційних технологій штучного інтелекту.

У першому розділі було проаналізовано стан сучасних підходів до підтримки психічного здоров'я, оглянуто існуючі аналоги таких систем, зокрема Wysa, Calmerry, Youper, та їхні особливості. Розглянуто загальновідомі методи оцінки психічного здоров'я, а також визначено переваги й недоліки веб-додатків.

У другому розділі виконано проектування інформаційної системи, розроблено архітектуру «клієнт-серверної» взаємодії, а також проведено вибір технологій для створення клієнтської та серверної частин. Спроектовано базу даних. Розроблено алгоритми на основі штучного інтелекту для проведення попередньої діагностики психологічного стану користувачів.

У третьому розділі здійснено програмну реалізацію системи. Було реалізовано серверну частину додатку, а також створено функціональний інтерфейс користувача, та інтегровано AI-чат для діагностики та надання рекомендацій.

У четвертому розділі проведено тестування системи, що включало перевірку функціональності користувацького інтерфейсу та коректної роботи серверної частини. Результати тестування підтвердили стабільність і коректність роботи системи.

У п'ятому розділі проведено економічні розрахунки, що підтвердили ефективність розробки. Термін окупності інвестицій склав 1,81 роки, що свідчить про високу привабливість системи для потенційних інвесторів.

Розроблена інформаційна система психологічної реабілітації з використанням штучного інтелекту має потенціал для впровадження в реабілітаційних центрах, медичних установах, різних компініях та як самостійний продукт для широкого кола користувачів. Її використання дозволить забезпечити доступ до своєчасної психологічної допомоги, знизити бар'єри для її отримання та підвищити ефективність реабілітаційних заходів.

Отже, робота має високий потенціал подальшого розвитку та являється доцільною розробкою в сучасному світі, так як здатно покращити якість надання психологічної допомоги завдяки використанню штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИНАННЯ

1. Система у сфері психічного здоров'я та психосоціальної підтримки в Україні. Всеукраїнська програма ментального здоров'я "Ти як?". 2024.

2. 10 статистичних даних про штучний інтелект, які потрібно знати у 2024 році. SKIM AI. [Електронний ресурс]. Режим доступу: https://skimai.com/uk/10-статистичних-даних-про-штучний-інтел/

3. 14 млн українців через війну потребують психологічної допомоги: як ми реагуємо на стрес. Дніпропетровська обласна державна адміністрація. [Електронний ресурс]. Режим доступу: https://adm.dp.gov.ua/news/14-mln-ukrayinciv-cherez-vijnu-potrebuyut-psihologichnoyi-dopomogi-yak-mi-reaguyemo-na-stres

4. Фічковський Д. А. Добровольська Н. В. «Інтеграція штучного інтелекту в інформаційні системи психологічної реабілітації». Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)», Вінниця, 2024. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22505/18638

5. Impact of artificial intelligence (AI) on psychological and mental health promotion: An opinion piece / K. Oladimeji et al. New Voices in Psychology. 2023. [Електронний ресурс]. Режим доступу: https://doi.org/10.25159/2958-3918/14548

6. Дерев'янко С. П., Примак Ю. В., Ющенко І. М. Штучний інтелект та емоційний штучний інтелект як феномени сучасної когнітивної психології. Наукові записки Національного університету «Острозька академія». Серія «Психологія». 2020. [Електронний ресурс]. Режим доступу: https://doi.org/10.25264/2415-7384-2020-11-115-119

7. Погореленко А. Штучний інтелект: сутність, аналіз застосування, перспективи розвитку / А. Погореленко // Економічні науки. – 2018.

8. Мар'єнко М., Коваленко В. Штучний інтелект та відкрита наука в освіті / М. Мар'єнко, В. Коваленко // Фізико-математична освіта. – 2023.

[Електронний ресурс]. Режим доступу: https://lib.iitta.gov.ua/id/eprint/734475/1/2023-381-marienkokovalenko.pdf

9. Використання можливостей штучного інтелекту для покращення психічного здоров’я. [Електронний ресурс]. Режим доступу: https://betshy.com/uk/2024/01/30/harnessing-the-power-of-artificial-intelligence-to-enhance-mental-health/

10. Сучасні методи веб-програмування. [Електронний ресурс]. Режим доступу: http://sites.znu.edu.ua/webprog/1168.ukr.html

11. Переваги та недоліки web-додатків. [Електронний ресурс]. Режим доступу: https://stud.com.ua/97611/informatika/perevagi_nedoliki_dodatkiv

12. Веб-додаток: що це таке, типи, переваги, принцип роботи. [Електронний ресурс]. Режим доступу: https://megasite.ua/ua/veb-prilogenie-chto-eto-takoe-tipi-preimushchestva-printsip-raboti

13. «Сучасні клієнт-серверні технології та їх застосування при вивченні систем управління базами даних» [працювали: Н.Р. Балик, В.І. Мандзюк].

14. Васильєв О. М. Програмування мовою Python / О. М. Васильєв. - Тернопіль : Навчальна книга - Богдан, 2021. - 504 с

15. FastAPI Документація. [Електронний ресурс]. Режим доступу: https://fastapi.tiangolo.com/.

16. JavaScript - динамічний сценарій на стороні клієнта. [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript.

17. React Документація. [Електронний ресурс]. Режим доступу: https://react.dev/.

18. PostgreSQL Документація. [Електронний ресурс]. Режим доступу: https://www.postgresql.org/about/.

19. SQLAlchemy. Комплексний підручник. [Електронний ресурс]. Режим доступу: https://docs.sqlalchemy.org/en/20/tutorial/index.html.

20. 10 найкращих інструментів розробки та тестування API. [Електронний ресурс]. Режим доступу: http://surl.li/htncc.

21. Система контейнеризації docker. [Електронний ресурс]. Режим доступу: https://www.dut.edu.ua/ua/news-1-626-9313-sistema-konteynerizacii-docker_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy.

22. Мойсеєнко М. І., Кузишин М. М. Великі мовні моделі штучного інтелекту в медицині. [Електронний ресурс]. Режим доступу: https://vspu.net/sit/index.php/sit/article/view/5637/5065

23. Загальна психологія. Модульно-рейтинговий курс для студентів вищих навчальних закладів./ Цимбалюк І. М., Яницька О. Ю. — К.: Професіонал, 2004.

24. Класифікація психологічних тестів [Електронний ресурс]. Режим доступу: https://studopedia.org/13-103872.html

25. Основи проектування баз даних. Навчальний посібник./ Гайна Г. А. — К.: КНУБА, 2005. — 204 с.

26. Крепич С.Я., Співак І.Я. Якість програмного забезпечення та тестування: базовий курс. Навчальний посібник. Тернопіль: ФОП Паляниця В.А., 2020. – 478с.

27. Адлер О.О. Методичні вказівки до підготовки та написання курсової роботи з дисципліни «Економічне обґрунтування інноваційних рішень» / Уклад. О.О.Адлер, І.В.Причепа, Н.М.Тарасюк. — Вінниця: ВНТУ, 2014.

28. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад.: В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
_____ проф., д.т.н. О. Д. Азаров
«_29_» _вересня_ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Інформаційна система психологічної реабілітації з використанням штучного інтелекту»
08-54.МКР.037.00.000 ТЗ

Науковий керівник: к.т.н., доц. каф ОТ
_____ Добровольська Н. В.
Студент групи 2КІ-23м
_____ Фічковський Д. А.

ВНТУ 2024

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність теми дослідження зумовлена зростанням кількості людей, які потребують психологічної підтримки. Постійні стреси, економічна нестабільність, соціальні конфлікти, пандемії та інші глобальні фактори сприяють зростанню рівня депресії, тривожності та інших психологічних розладів. Такий веб-додаток може надавати індивідуальні рекомендації, проводити попередній аналіз психоемоційного стану та допомагати у вирішенні особистих проблем завдяки інтеграції сучасних технологій штучного інтелекту.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — система, яка дозволяє виявити психологічні проблеми, в тому числі з використанням штучного інтелекту, що надає рекомендації для покращення психологічного стану, та забезпечує візуалізацію прогресу.

2.2 Призначення розробки — підвищення ефективності психологічної допомоги, покращення її доступності та впровадження сучасних технологій для підтримки психічного здоров’я користувачів.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих методів та принципів.

3.2 Навчання моделі.

3.4 Розробка веб-додатку.

3.5 Тестування веб-додатку.

3.6 Виконання розрахунків для доведення доцільності розробки з економічної точки зору.

4. Вимоги до виконання МКР

Головна вимога — розробити зручну, легко масштабовану систему, використовуючи усі здобуті навички та знання.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в таблиці А.1

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		Початок	Кінець	
1	Аналіз предметної області	18.09.24	25.09.24	Розділ 1
2	Проектування системи	26.09.24	02.10.24	Розділ 2
3	Розробка функціоналу системи	03.10.24	01.11.24	Розділ 3
4	Тестування системи	04.11.24	08.11.24	Розділ 4
5	Підготовка економічної частини	09.11.24	13.11.24	Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу, презентації	27.11.24	09.12.24	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	10.12.24	16.12.24	Оформленні домументи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлення та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21»

ДОДАТОК Б

Діаграма прецедентів системи

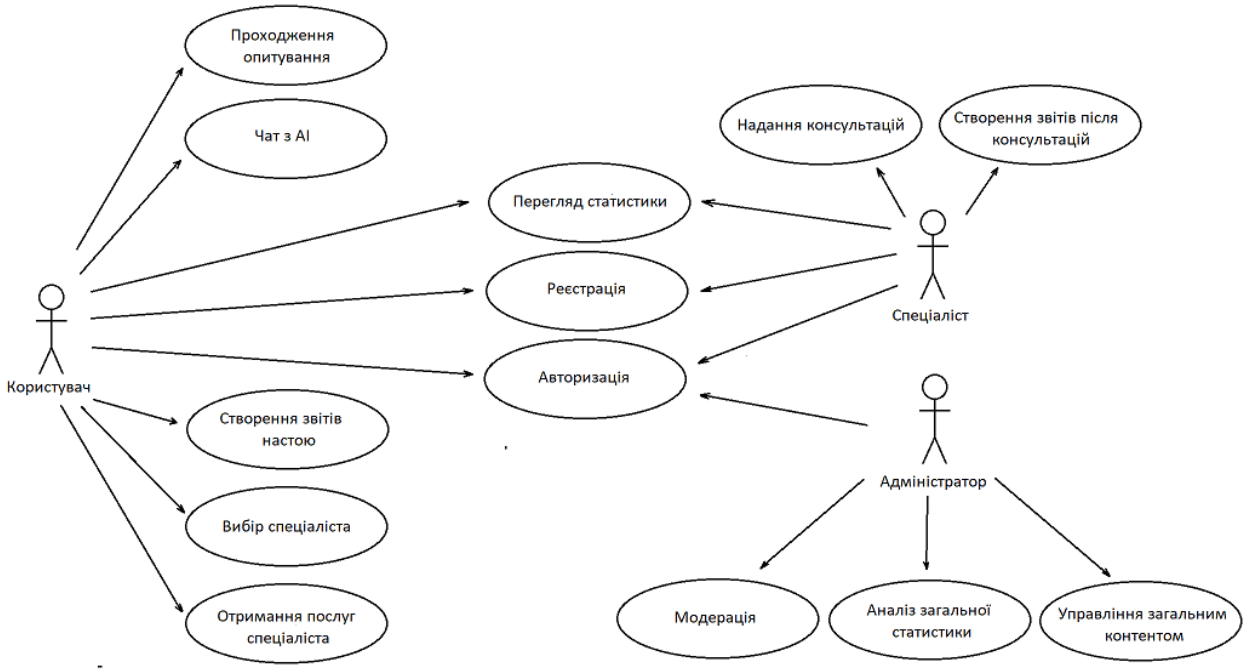

Рисунок Б.1 — Діаграма прецедентів системи

ДОДАТОК В

Діаграма діяльності системи

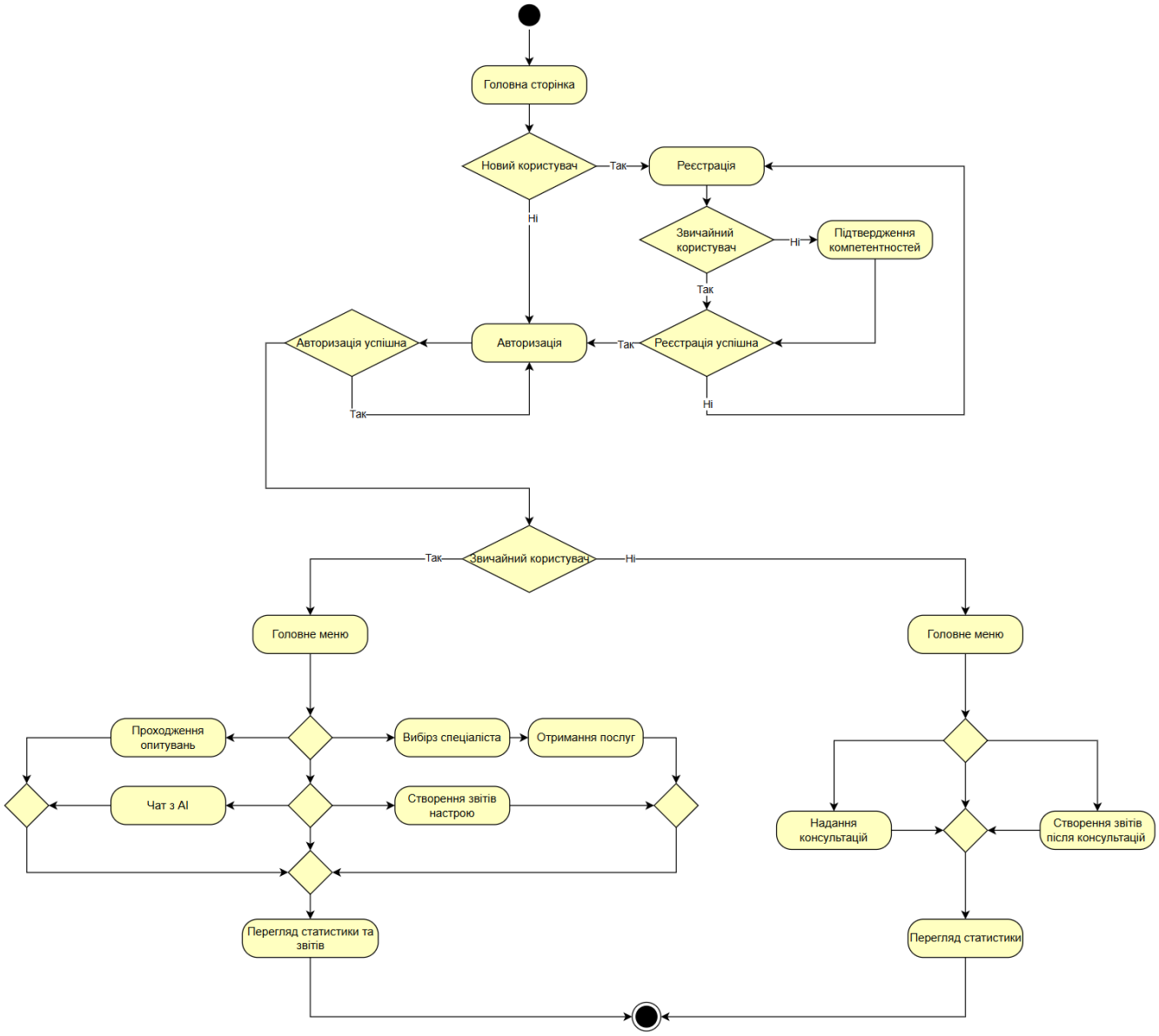

Рисунок В.1 — Діаграма діяльності системи

ДОДАТОК Г

Графічна структура бази даних

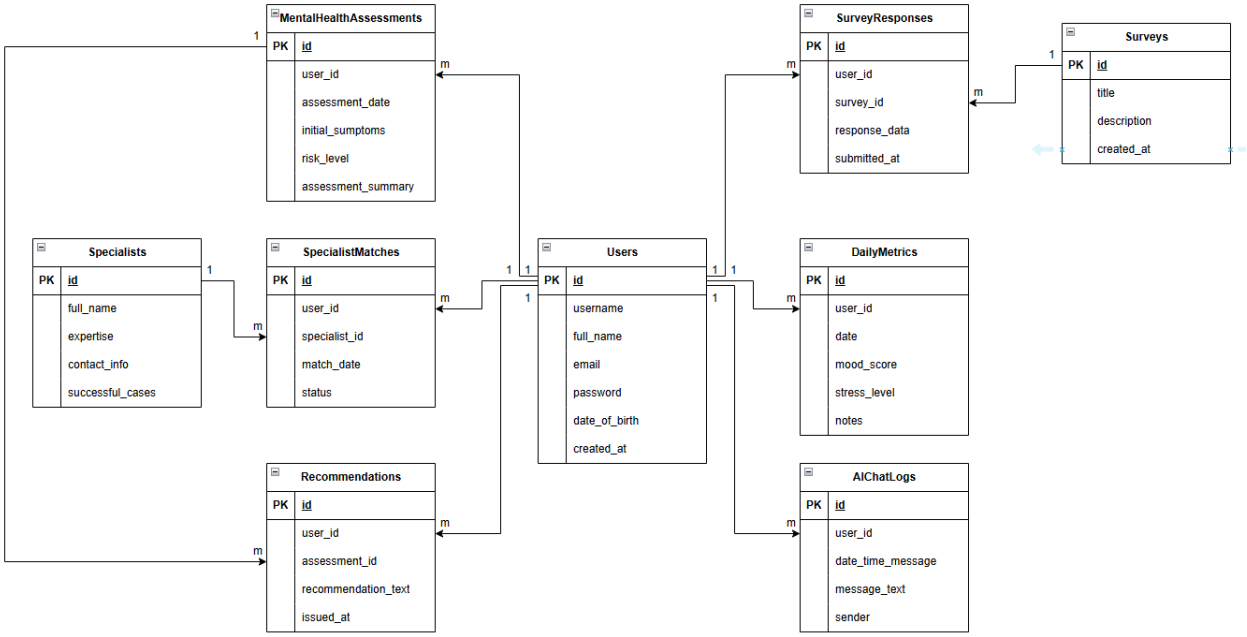

Рисунок Г.1 — Графічна структура бази даних

ДОДАТОК Д

Програмний код. Навчання моделі

```
import numpy as np
import pandas as pd
import random
import json
import torch
import torch.nn as nn
import nltk
nltk.download('punkt')
from torch.utils.data import Dataset, DataLoader
import numpy as np
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()
    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        return out
```

```
with open('intents.json', 'r') as f:
    intents = json.load(f)
def tokenize(sentence):
    return nltk.word_tokenize(sentence)

def stem(word):
    return stemmer.stem(word.lower())

def bag_of_words(tokenized_sentence, words):
    sentence_words = [stem(word) for word in tokenized_sentence]
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1
    return bag

all_words = []
tags = []
xy = []
for intent in intents['intents']:
    tag = intent['tag']
    tags.append(tag)
    for pattern in intent['patterns']:
        w = tokenize(pattern)
        all_words.extend(w)
        xy.append((w, tag))
ignore_words = ['?', '.', '!']
all_words = [stem(w) for w in all_words if w not in ignore_words]
```

```
all_words = sorted(set(all_words))
tags = sorted(set(tags))
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    label = tags.index(tag)
    y_train.append(label)
X_train = np.array(X_train)
y_train = np.array(y_train)
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
print(input_size, output_size)
class ChatDataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]
    def __len__(self):
        return self.n_samples
```

```
ataset = ChatDataset()
train_loader = DataLoader(dataset=dataset,
                          batch_size=batch_size,
                          shuffle=True,
                          num_workers=0)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = NeuralNet(input_size, hidden_size, output_size).to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(dtype=torch.long).to(device)
        outputs = model(words)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    if (epoch+1) % 100 == 0:
        print (f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
print(f'final loss: {loss.item():.4f}')
```

ДОДАТОК Е

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:_____ Інформаційна система психологічної реабілітації з використанням штучного інтелекту

Тип роботи:_____Магістерська кваліфікаційна робота_____

(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____ кафедра обчислювальної техніки_____

(кафедра, факультет (інститут), навчальна група)

Керівник_____Добровольська Н. В., доц. каф. ОТ_____

(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	82%
Загальна схожість	18%

Аналіз звіту подібності (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ ____Фічковський Д.А._____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

_____

_____

_____

Особа, відповідальна за перевірку_____ _Захарченко С.М__

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали, посада)