НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«На правах рукопису»
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2025 р.

Магістерська дисертація

на здобуття ступеня другого (магістерського) рівня вищої освіти
за освітньою програмою «Цифрові технології в енергетиці»
зі спеціальності 122 «Комп’ютерні науки»

на тему Web-платформа підтримки ментального здоров’я з інтеграцією технологій машинного навчання

Виконав студент _2_ курсу, групи ТР-43мп
___Хороших Олександр Леонідович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент к.е.н. Ірина СЕГЕДА _____
(науковий ступінь, вчене звання, ім’я ПРІЗВИЩЕ) (підпис)

Рецензент _доцент каф. ТАЕ, к.т.н. Олександр СІРИЙ___ _____
(посада, науковий ступінь, вчене звання, ім’я ПРІЗВИЩЕ) (підпис)

Н.контроль _асистент, Володимир РУДИК_____ _____
(посада, ім’я ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ — 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти другий (магістерський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-наукова програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

«___»_____2025 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Хороших Олександру Леонідовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Web-платформа підтримки ментального здоров’я з інтеграцією технологій машинного*

керівник роботи *Сегеда Ірина Василівна, к.е.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 3 листопада 2025 р. № 4779-с

2. Термін подання студентом дисертації 06.12.2025

3. Об’єкт дослідження : процес цифрової підтримки ментального здоров’я користувачів за допомогою інтелектуальних рекомендаційних систем

4. Вихідні дані: програмне забезпечення для моніторингу емоційного стану, ведення щоденника та отримання персоналізованих порад на основі алгоритмів машинного навчання

5. Перелік завдань: провести аналіз сучасних цифрових рішень у сфері ментального здоров’я; дослідити теоретичні основи рекомендаційних алгоритмів TF-IDF і косинусної подібності; спроєкувати архітектуру клієнтсько-серверної веб-

платформи; реалізувати серверну частину з використанням Nest.js і PostgreSQL; створити клієнтську частину на базі Next.js з інтеграцією React Query та Zod; впровадити механізми аналітики, рекомендацій та взаємодії з чат-ботом; виконати тестування системи та перевірку адаптивності інтерфейсу.

6. Орієнтований перелік ілюстративного матеріалу : архітектура програмного забезпечення, інтерфейс застосунку, схема структури даних

7. Орієнтований перелік публікацій: XIII Міжнародна науково-практична конференція «Modern digital technologies and problems of their use» Технічні науки, 24 листопада, Прага, Чехія.

6. Дата видачі завдання «16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів підготовки магістерської дисертації	Термін виконання	Примітка
1	Дослідження предметної області	18.09.2025	виконано
2	Вибір засобів для реалізації веб-платформи	21.09.2025	виконано
3	Розробка структури бази даних та архітектури системи	26.09.2025	виконано
4	Проєктування та розробка веб-платформи	13.10.2025	виконано
5	Тестування розробленої веб-платформи, виправлення недоліків	19.10.2025	виконано
6	Захист створеного програмного забезпечення	20.10.2025	виконано
7	Оформлення магістерської дисертації	25.11.2025	виконано
8	Передзахист	27.11.2025	виконано
9	Подання матеріалів магістерської дисертації на кафедру	04.12.2025	виконано
10	Захист	22.12.2025	виконано

Студент _____ **Олександр ХОРОШИХ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **Ірина СЕГЕДА**
(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Структура та обсяг кваліфікаційної роботи. Магістерська дисертація складається зі вступу, шести розділів, висновків та додатка. Робота містить посилання на 33 джерела, 15 таблиць, 26 ілюстрацій. Основна частина викладена на 110 сторінках, загальний обсяг роботи 122 сторінки.

Актуальність. У сучасному цифровому суспільстві зростає усвідомлення важливості ментального здоров'я як одного з ключових чинників загального добробуту людини. Стрес, тривожність і емоційне вигорання стають поширеними проблемами, особливо серед молоді та осіб, що перебувають у кризових умовах. Однак доступ до кваліфікованої психологічної допомоги часто є обмеженим через брак фахівців, стигматизацію теми або фінансові бар'єри. У зв'язку з цим актуальним є створення цифрових інструментів самодопомоги, які поєднують простоту використання, доступність і персоналізацію.

Одним з перспективних напрямів є розробка веб-платформ, здатних не лише фіксувати психоемоційний стан користувача, а й надавати індивідуальні рекомендації на основі аналізу текстів і поведінкових даних. Застосування алгоритмів машинного навчання, зокрема TF-IDF та cosine similarity, дозволяє сформувати релевантні поради, враховуючи індивідуальні особливості користувача, його емоційні коливання та взаємодію з платформою .

Метою роботи є розробка веб-платформи підтримки ментального здоров'я, яка забезпечує персоналізовану взаємодію з користувачем на основі алгоритмів машинного навчання, дозволяє вести щоденник, фіксувати настрій, переглядати аналітику та отримувати рекомендації й навчальний контент для емоційної самопідтримки.

Для досягнення поставленої мети виконано такі **завдання**:

— проаналізовано сучасні цифрові сервіси у сфері ментального здоров'я;

— досліджено математичні основи NLP-моделей, алгоритмів TF-IDF та cosine similarity;

— спроєктовано архітектуру клієнтсько-серверної веб-платформи;

— розроблено серверну частину на основі Nest.js та базу даних у PostgreSQL;

— реалізовано клієнтську частину з використанням Next.js, Tailwind CSS і React Query;

— створено функціональні модулі: щоденник, трекер настрою, навчальні вправи, чат і система рекомендацій;

— проведено дослідження розробленої системи.

Об'єкт дослідження — процес цифрової підтримки ментального здоров'я користувачів у веб-середовищі.

Предмет дослідження — веб-платформа з модульною архітектурою, що реалізує фіксацію емоційного стану, персоналізовані рекомендації та навчальний контент.

Методи дослідження — аналіз, NLP, векторне подання текстів, проєктування, програмування, валідація, REST API, тестування.

Практичне значення роботи полягає у створенні повнофункціонального веб-застосунку, який може бути використаний як цифровий інструмент самодопомоги для покращення ментального стану користувача. Платформа також є адаптивною до впровадження у навчальні, корпоративні чи медичні середовища для моніторингу емоційного стану та надання рекомендацій.

Апробація результатів дисертації.

Основні положення магістерської роботи були опубліковані та презентовані на: XIII Міжнародна науково-практична конференція «Modern digital technologies and problems of their use» Технічні науки, 24 листопада, Прага, Чехія.

Результати дослідження представлені у 1 науковій публікації.

Ключові слова: ментальне здоров'я, машинне навчання, веб-платформа, Next.js, Nest.js, PostgreSQL, TF-IDF, рекомендаційна система, аналіз настрою, цифрова підтримка.

ABSTRACT

Structure and Volume of the Thesis. The master's thesis consists of an introduction, six chapters, conclusions, and an appendix. The thesis includes references to 33 sources, 15 tables and contains 26 illustrations. The main content is presented on 110 pages, total volume of work 122 pages.

Relevance. In today's digital society, there is a growing awareness of the importance of mental health as a key factor in overall human well-being. Stress, anxiety, and emotional burnout have become increasingly common problems, especially among young people and individuals in crisis situations. However, access to qualified psychological support is often limited due to a shortage of specialists, the stigmatization of mental health topics, or financial barriers. Therefore, the development of digital self-help tools that combine ease of use, accessibility, and personalization is highly relevant.

One of the promising directions is the development of web platforms capable not only of recording the user's emotional state but also of providing individualized recommendations based on the analysis of text and behavioral data. The use of machine learning algorithms, particularly TF-IDF and cosine similarity, makes it possible to generate relevant suggestions that take into account a user's individual characteristics, emotional dynamics, and platform activity.

The aim of the study is to develop a web-based mental health support platform that provides personalized interaction using machine learning algorithms, enables users to keep a journal, track their mood, view analytics, and receive tailored recommendations and educational content for emotional self-regulation.

To achieve this goal, the following **tasks** were completed:

— an analysis of current digital services in the mental health domain was conducted;

— the mathematical foundations of NLP models, TF-IDF and cosine similarity algorithms were studied;

— the client-server architecture of the web platform was designed;

— the backend was developed using Nest.js and a PostgreSQL database;

— the frontend was implemented using Next.js, Tailwind CSS, and React Query;

— functional modules were created: mood tracker, journal, learning exercises, chatbot, and recommendation system;

— a comprehensive evaluation of the implemented system was carried out.

Object of the research: the process of digital mental health support in a web environment.

Subject of the research: a modular web platform that enables mood tracking, personalized recommendations, and access to educational content.

Research methods: analysis, NLP, vector-based text representation, system design, programming, validation, REST API, and testing.

Practical value of the work lies in the creation of a fully functional web application that can serve as a digital self-help tool to improve the user's mental well-being. The platform is also adaptable for integration into educational, corporate, or healthcare environments for mood monitoring and personalized support delivery.

Approbation of the dissertation results.

The key findings of the thesis were presented and published at the XIII International Scientific and Practical Conference «Modern digital technologies and problems of their use» (Technical Sciences), November 24, Prague, Czech Republic.

The research results were published in one scientific paper.

Keywords: mental health, machine learning, web platform, Next.js, Nest.js, PostgreSQL, TF-IDF, recommendation system, mood analysis, digital support.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI (Artificial Intelligence) — штучний інтелект; галузь комп'ютерних наук, що займається створенням систем, здатних виконувати завдання, які потребують людського інтелекту.

API (Application Programming Interface) — інтерфейс прикладного програмування; набір визначень, протоколів і інструментів для взаємодії програмних компонентів.

JWT (JSON Web Token) — формат токена для безпечного обміну інформацією між учасниками системи з використанням цифрового підпису.

LLM (Large Language Model) — велика мовна модель; тип моделі штучного інтелекту, здатної обробляти, генерувати та аналізувати природну мову на основі великих корпусів даних.

ML (Machine Learning) — машинне навчання; підгалузь штучного інтелекту, що дозволяє системам навчатися з даних та покращувати результати без явного програмування.

NLP (Natural Language Processing) — обробка природної мови; технологія, яка дозволяє комп'ютерам аналізувати, розуміти та генерувати мову людини.

ORM (Object-Relational Mapping) — об'єктно-реляційне відображення; спосіб взаємодії між об'єктами програмного коду та реляційною базою даних.

REST (Representational State Transfer) — архітектурний стиль взаємодії між клієнтом і сервером у веб-застосунках на основі HTTP.

SSR (Server Side Rendering) — серверний рендеринг; спосіб генерації HTML-сторінки на сервері перед її відправленням клієнту.

SPA (Single Page Application) — односторінковий веб-застосунок; тип веб-додатку, що динамічно оновлює вміст без повного перезавантаження сторінки.

TF-IDF (Term Frequency — Inverse Document Frequency) — статистичний показник важливості слова в окремому документі відносно всього корпусу.

UI (User Interface) — інтерфейс користувача; сукупність елементів для

взаємодії людини з програмною системою.

UX (User Experience) — користувацький досвід; враження користувача від використання системи.

Zod — TypeScript-бібліотека для валідації та парсингу вхідних даних у веб-застосунках.

React.js — JavaScript-бібліотека для побудови інтерфейсів користувача.

Next.js — фреймворк для React.js, що дозволяє створювати високопродуктивні веб-додатки з підтримкою SSR та SPA.

Nest.js — серверний фреймворк для Node.js, орієнтований на модульну архітектуру та підтримку TypeScript.

PostgreSQL — об’єктно-реляційна система керування базами даних з відкритим кодом.

Sequelize — ORM-бібліотека для Node.js, яка забезпечує взаємодію з реляційними базами даних.

Tailwind CSS — utility-first фреймворк для швидкої стилізації веб-інтерфейсів.

React Query — бібліотека для керування запитами даних та кешуванням у React-додатках.

Chatbot — програма, що імітує спілкування з людиною через текстовий інтерфейс.

Cosine similarity — косинусна міра подібності; математична метрика, яка визначає схожість між векторами у просторі.

Vector space model — модель векторного простору; концепція представлення тексту у вигляді багатовимірного вектора для аналізу подібності.

Gemini AI — мультимодальна мовна модель від Google, яка підтримує обробку тексту, зображень і коду для інтеграції у цифрові сервіси.

ВСТУП

Сучасний світ характеризується стрімким зростанням рівня стресу, тривожності та емоційного виснаження [1], що створює потребу у доступних та ефективних інструментах підтримки ментального здоров'я. Забезпечення своєчасної психологічної допомоги та можливості самостійного моніторингу свого стану стає критично важливим аспектом для людей, які прагнуть підтримувати власний психоемоційний баланс і попереджати розвиток розладів [2]. У цьому контексті розробка веб-платформи для підтримки ментального здоров'я з інтеграцією технологій машинного навчання виступає сучасним рішенням, що дозволяє автоматизувати процес аналізу стану користувача та формувати персоналізовані рекомендації.

Проблема відсутності персоналізованих цифрових інструментів часто призводить до низької ефективності існуючих застосунків, які пропонують універсальний контент без урахування індивідуальних потреб [3]. Це знижує мотивацію користувачів, ускладнює відстеження змін власного стану та не сприяє довгостроковому результату. Дана робота присвячена розробці веб-платформи, яка підвищує якість психологічної самодопомоги, оптимізує взаємодію користувача з ресурсами та забезпечує гнучку персоналізацію завдяки алгоритмам машинного навчання.

Для створення актуального та практично корисного продукту необхідно реалізувати масштабовану та інтерактивну платформу, що включатиме щоденник, трекер настрою, чат-інтерфейс, аналітичний модуль і систему рекомендацій. Таке рішення спрощує процес самоспостереження, зменшує ймовірність помилкової інтерпретації свого стану та забезпечує користувачеві індивідуалізований набір інструментів для покращення ментального благополуччя.

Для розробки клієнтської та серверної частини платформи використано мову програмування TypeScript, а також сучасні фреймворки і бібліотеки: Next.js для створення швидкого та адаптивного інтерфейсу, Nest.js для побудови надійної серверної архітектури, PostgreSQL для зберігання структурованих даних

користувача, Sequelize для зручної ORM-інтеграції. Tailwind CSS використано для створення сучасного інтерфейсу, а Chart.js — для реалізації аналітичних візуалізацій. Для формування персоналізованих рекомендацій застосовано алгоритми TF-IDF та косинусної подібності, які забезпечують точність у визначенні релевантного контенту [4].

Одним із ключових аспектів розробки платформи є забезпечення безпеки персональних психологічних даних. Використання сучасних методів шифрування, механізмів авторизації та захисту доступу гарантує конфіденційність інформації та мінімізує ризики несанкціонованого втручання.

Впровадження веб-платформи дозволить користувачам підвищити усвідомленість свого емоційного стану, отримувати персоналізовані рекомендації в режимі 24/7, зменшити рівень стресу та сформувати більш здорові поведінкові патерни. Використані технології забезпечують масштабованість, стабільність та високу продуктивність системи, що робить її ефективним рішенням для широкого кола користувачів, які прагнуть покращити власне ментальне благополуччя.

Структура та обсяг кваліфікаційної роботи. Магістерська дисертація складається зі вступу, шести розділів, висновків та додатка. Робота містить посилання на 33 джерела, 15 таблиць, 26 ілюстрацій. Основна частина викладена на 110 сторінках, загальний обсяг роботи 122 сторінки.

У першому розділі описано постановку задачі розробки веб-платформи підтримки ментального здоров’я з інтеграцією технологій машинного навчання, сформульовано мету та завдання дослідження, визначено призначення системи, її основні функціональні можливості та ключові вимоги до платформи як інструменту персоналізованої психологічної підтримки.

У другому розділі розглянуто програмні засоби розробки програмного забезпечення. Описано вибір технологічного стеку для клієнтської та серверної частин платформи, зокрема мови TypeScript, фреймворків Next.js і Nest.js, системи керування базами даних PostgreSQL, ORM-бібліотеки Sequelize, а також інструментів для стилізації інтерфейсу та візуалізації аналітичних даних. Обґрунтовано їх придатність для створення масштабованої та надійної веб-

системи.

У третьому розділі розглянуто теоретичні та математичні основи функціонування веб-платформи. Наведено опис алгоритмів TF-IDF і косинусної подібності, підходів NLP до обробки текстових даних, а також принципів організації даних, побудови REST API й архітектурних рішень, що забезпечують роботу рекомендаційного модуля та аналітичних компонентів системи.

У четвертому розділі наведено опис програмної реалізації веб-платформи підтримки ментального здоров'я. Деталізовано структуру серверної частини, модулі авторизації, ведення щоденника, трекера настрою, рекомендаційної системи, аналітики та чат-інтерфейсу, а також описано клієнтську частину, логіку маршрутизації, валідації даних та взаємодії з базою даних.

У п'ятому розділі розглянуто взаємодію користувача з розробленою веб-платформою. Описано системні вимоги, основні сценарії використання, роботу інтерфейсу та модулів з погляду кінцевого користувача, а також подано приклади відображення аналітики, динаміки настрою та персоналізованих рекомендацій у процесі експлуатації системи.

У шостому розділі описано стартап-проєкт, створений на основі розробленої веб-платформи. Проведено аналіз ринкового середовища та цільової аудиторії, розглянуто конкурентні рішення, обґрунтовано унікальну ціннісну пропозицію продукту, наведено елементи техніко-економічного обґрунтування, стратегії монетизації та можливості масштабування платформи як комерційного цифрового сервісу підтримки ментального здоров'я

1 ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ВЕБ-ПЛАТФОРМИ ПІДТРИМКИ МЕНТАЛЬНОГО ЗДОРОВ'Я З ІНТЕГРАЦІЄЮ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ

Метою веб-платформи для підтримки ментального здоров'я є створення ефективного, надійного та доступного інструменту, який забезпечує користувачам персоналізовану психологічну підтримку на основі сучасних цифрових технологій. У світі, де рівень стресу, тривожності та емоційного виснаження стрімко зростає, виникає потреба у сервісах, здатних не лише фіксувати стан користувача, а й аналізувати його, пропонуючи індивідуальні рекомендації та інструменти самодопомоги [5]. Веб-платформа має виконувати функції інтелектуального асистента, що допомагає людям краще розуміти свій психоемоційний стан та підтримувати ментальне благополуччя.

У даний час складно забезпечити систематичну підтримку ментального здоров'я без сучасних цифрових рішень, тому впровадження веб-платформи є ключовим кроком для підвищення рівня доступності та ефективності самодопомоги. Використання алгоритмів машинного навчання дає змогу автоматизувати аналіз текстових і поведінкових даних, підвищити точність рекомендацій та мінімізувати вплив людського фактора. Це дозволяє користувачам своєчасно реагувати на зміни свого стану, формувати здорові звички та знижувати рівень стресу.

Призначення веб-платформи полягає у створенні середовища, яке об'єднує декілька важливих модулів: ведення щоденника, трекер настрою, аналітичний блок, чат-інтерфейс та систему рекомендацій. Така інтеграція забезпечує комплексний підхід до підтримки ментального здоров'я і дає змогу користувачу отримувати як об'єктивні дані про свій стан, так і релевантні поради, сформовані автоматизованими алгоритмами. Розроблена система повинна забезпечувати можливості наведені нижче.

Відстеження емоційного стану в реальному часі: надання актуальної інформації про настрій та динаміку змін на основі щоденникових записів, тегів, оцінок і активності користувача.

Автоматизована генерація рекомендацій: формування персоналізованих порад та вправ за допомогою алгоритмів TF-IDF та косинусної подібності, що аналізують текст і контекст записів.

Аналітика та візуалізація даних: створення графіків, статистики та узагальнень, які дозволяють користувачу відстежувати тенденції, закономірності та можливі тригери змін настрою.

Інтерактивний чат-інтерфейс: забезпечення зручної взаємодії з платформою через чат, що дозволяє отримувати рекомендації, нагадування та базову психологічну підтримку у режимі реального часу.

Підтримка мобільних пристроїв та адаптивність: забезпечення доступу до функціоналу з будь-якого пристрою, що дозволяє користувачу взаємодіяти з платформою у будь-який час і в будь-якому місці.

Реалізація веб-платформи сприятиме підвищенню рівня саморефлексії, зменшенню психологічного навантаження та наданню індивідуальної підтримки кожному користувачеві. Поєднання сучасного технологічного стека та алгоритмів машинного навчання робить платформу ефективним інструментом для покращення ментального благополуччя.

1.1 Огляд методів розробки веб-платформи підтримки ментального здоров’я

Для досягнення мети розробки веб-платформи було досліджено та проаналізовано сучасні методи створення цифрових рішень у сфері ментального здоров’я, а також підходи до обробки психологічних даних та генерації персоналізованих рекомендацій. Вибір методів залежить від вимог до функціональності, точності рекомендацій, безпеки та масштабованості системи.

Одним із ключових напрямів є використання методів аналізу

психоемоційного стану користувачів. Застосування алгоритмів машинного навчання, таких як TF-IDF та косинусна подібність, дозволяє автоматизувати обробку текстових даних щоденника, визначати ключові емоційні маркери та формувати персоналізовані рекомендації в реальному часі [6]. Методи NLP (обробки природної мови) забезпечують можливість виділення важливих фраз, визначення контексту та оцінки емоційного забарвлення записів [7].

Сучасні веб-платформи використовують інтегровані системи для відстеження динаміки стану користувача [8]. Аналітичні модулі дозволяють обчислювати зміни настрою, виявляти закономірності та відображати результати у вигляді графіків і статистики. Це забезпечує більш глибоке розуміння стану користувача та мотивує до регулярного ведення щоденника.

Методи побудови користувацького інтерфейсу включають застосування сучасних фреймворків, таких як Next.js, який надає можливість створювати швидкі та адаптивні веб-додатки. Tailwind CSS забезпечує ефективну стилізацію та дозволяє створювати сучасний інтерфейс з високим рівнем доступності. Для аналізу та візуалізації даних застосовуються такі бібліотеки, як Chart.js, що дає змогу формувати інформативні графіки настрою, активності та виконаних рекомендацій.

Методи управління даними включають використання реляційних баз даних, зокрема PostgreSQL, яка дозволяє гнучко зберігати інформацію про стан користувача, його записи, історію рекомендацій, чат-повідомлення та прогрес у вправах. Використання ORM-бібліотеки Sequelize забезпечує спрощену інтеграцію з базою даних, автоматизацію запитів та перевірку цілісності даних.

Окрему увагу приділено методам забезпечення безпеки. Зберігання психологічних даних користувачів потребує суворих заходів захисту: застосування сучасних протоколів аутентифікації, шифрування конфіденційної інформації, захисту API та контролю доступу. Дотримання цих вимог є критично важливим, оскільки дані користувачів містять чутливу особисту інформацію.

Таким чином, використання сучасних методів аналізу даних, побудови інтерфейсу, масштабування та забезпечення безпеки дозволяє створити надійну,

інтерактивну та корисну веб-платформу для підтримки ментального здоров'я, яка враховує індивідуальні потреби кожного користувача.

1.2 Програмні засоби розробки веб-платформи

Для розробки веб-платформи підтримки ментального здоров'я було обрано сучасний стек технологій, який забезпечує високу продуктивність, масштабованість, безпеку та зручність розширення функціоналу. Ключові програмні засоби, використані у створенні системи наведено нижче.

TypeScript — статично типізована надбудова над JavaScript, яка забезпечує покращену структуру проєкту, зменшує кількість помилок під час розробки та підвищує надійність коду. Використовується як на клієнтській, так і на серверній частині [9].

Next.js — фреймворк для React, який дозволяє створювати високопродуктивні веб-інтерфейси з підтримкою серверного рендерингу, маршрутизації та оптимізації завантаження. Застосовується для реалізації інтерфейсу платформи та інтерактивних модулів (чат, щоденник, аналітика, рекомендації) [10].

Nest.js — сучасний серверний фреймворк для Node.js, побудований на архітектурі модулів. Забезпечує структуровану розробку backend-логіки, включно з аутентифікацією, роботою з базою даних, логікою аналітики та рекомендаційної системи [11].

Swagger — інструмент документування API, що дозволяє автоматично генерувати специфікації ендпоінтів та забезпечує зручність тестування взаємодії між клієнтом і сервером.

Tailwind CSS — CSS-фреймворк, який дає можливість створювати сучасні адаптивні інтерфейси з використанням зручних utility-класів. Забезпечує швидку стилізацію компонентів платформи.

Chart.js — бібліотека для побудови графіків та аналітичних візуалізацій. Використовується в модулі аналітики для відображення графіків настрою,

активності та виконання рекомендацій.

Sequelize — ORM фреймворк для Node.js, що спрощує взаємодію серверної частини з реляційною базою даних. Забезпечує створення моделей, автоматизацію запитів, керування зв'язками між таблицями та міграціями.

PostgreSQL — високопродуктивна реляційна система керування базами даних, використана для зберігання даних користувачів: профілів, записів настрою, журналів, аналітики, рекомендацій та історії чату.

JSON — стандартний формат обміну даними між клієнтом і сервером, який дозволяє передавати структуровану інформацію у зручному вигляді.

Обрані програмні засоби забезпечують ефективну розробку та підтримку веб-платформи, дозволяють легко інтегрувати нові модулі, забезпечують масштабованість, стабільність роботи та високу якість взаємодії користувача з системою. Використання сучасного технологічного стеку також сприяє побудові безпечного середовища для зберігання й аналізу чутливих психологічних даних.

У першому розділі було проведено аналіз сучасного стану цифрових рішень у сфері ментального здоров'я та визначено ключові проблеми, що зумовлюють потребу у створенні персоналізованих веб-платформ психоемоційної підтримки. Здійснений огляд підтвердив, що більшість існуючих сервісів не забезпечують комплексного підходу до моніторингу емоційного стану користувача, мають обмежену функціональність та не використовують засоби машинного навчання для адаптивного надання рекомендацій.

Висновки до розділу 1

У ході дослідження сформульовано мету та завдання розробки веб-платформи, орієнтованої на персоналізацію, інтерактивність та можливість автоматизованої аналітики текстових і поведінкових даних. Визначено основні функціональні модулі системи, серед яких: ведення щоденника, фіксація настрою, формування рекомендацій на основі алгоритмів TF-IDF і косинусної подібності, інтерактивний чат та інструменти візуалізації.

Також окреслено вимоги до платформи як сучасного цифрового інструменту підтримки ментального здоров'я: висока доступність, безпека персональних даних, масштабованість, адаптивний інтерфейс та інтеграція алгоритмів аналізу. Результати проведеного огляду дозволили сформувати чітке технічне та наукове підґрунтя для розробки платформи та визначити архітектурні принципи, що мають бути реалізовані у подальших розділах роботи.

Таким чином, розділ 1 заклав теоретичну та концептуальну основу для створення веб-платформи підтримки ментального здоров'я, обґрунтував актуальність проєкту, визначив предмет, об'єкт, мету і завдання дослідження, а також сформував комплекс функціональних і технологічних вимог до програмної системи.

2 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Одним із найважливіших етапів розробки будь-якого програмного продукту є вибір технологій розробки, за допомогою яких буде реалізовано весь необхідний функціонал. Також важливо зробити процес розробки ефективним, використовуючи сучасні бібліотеки та фреймворки, які спрощують етап програмування.

Звідси, правильний вибір інструментів розробки не лише оптимізує процес взаємодії користувача із системою, але й робить розробку веб-платформи більш ефективною та швидкою.

2.1 Мова програмування TypeScript

У процесі розробки веб-платформи підтримки ментального здоров'я було обрано мову програмування TypeScript, яка є однією з найсучасніших та найзручніших для створення масштабованих веб-систем. TypeScript являє собою статично типізовану надбудову над JavaScript, що поєднує гнучкість традиційного JavaScript з перевагами строгого контролю типів, завдяки чому суттєво підвищується якість та надійність програмного коду. У великих проєктах, які містять багато взаємопов'язаних модулів та складну взаємодію між клієнтською й серверною частинами, використання статичної типізації дозволяє виявляти помилки ще на етапі компіляції, не допускаючи їх у фінальну версію [12]. Це особливо важливо у середовищах, де обробляються персональні психологічні дані, а також виконується розрахунок рекомендацій на основі алгоритмів машинного навчання.

TypeScript значно спрощує роботу розробника завдяки можливостям сучасних інтегрованих середовищ розробки, які надають автодоповнення, підсвічування типів, контекстні підказки та зручну навігацію кодом. Це дозволяє швидко пересуватися між модулями, уникати помилок у назвах, точно визначати структуру даних і зменшувати ймовірність неправильного виклику методів чи

доступу до властивостей. Для веб-платформи підтримки ментального здоров'я, яка включає модулі авторизації, обробки записів настрою, щоденника, аналітики, чату, рекомендаційної системи та роботи з базою даних, така точність значно полегшує підтримку проєкту та спрощує подальше розширення функціоналу.

Важливою перевагою TypeScript є його повна сумісність із JavaScript, що дозволяє поступово впроваджувати статичну типізацію в проєкти будь-якого розміру. Завдяки цьому підхід до розробки може бути гнучким: розробник не зобов'язаний одразу типізувати всі модулі, а може робити це поступово, що є особливо корисним при роботі з великими проєктами або при перенесенні існуючого JavaScript-коду на TypeScript. Така можливість особливо важлива для систем, які активно розвиваються, оскільки зменшує витрати на підтримку та забезпечує стабільність між різними версіями програми.

TypeScript також підтримує сучасні стандарти ECMAScript, що дає змогу застосовувати найновіші функції мови навіть у тому випадку, якщо браузери чи середовище Node.js не повністю їх підтримують. Це дозволяє використовувати асинхронні конструкції, нові синтаксичні можливості, декоратори, модульну систему та інші інструменти, які роблять код більш зрозумілим, структурованим і простим для підтримки [13]. Така підтримка є особливо важливою для модулів, які працюють з API, обробляють великі обсяги даних або взаємодіють із клієнтською частиною через багатоетапні асинхронні процеси.

У веб-платформі підтримки ментального здоров'я TypeScript використовується для типізації моделей даних, що охоплюють користувацькі профілі, записи настрою, текстові записи щоденника, історію рекомендацій, аналітичні показники та структуру чату. Завдяки системі типів можна чітко описати структуру кожного об'єкта, визначити його обов'язкові та необов'язкові властивості, передбачити типи, які повертаються API, і забезпечити коректність обробки всіх даних на кожному етапі — від взаємодії користувача з інтерфейсом до запису інформації в базу даних і повернення результатів сервером.

TypeScript також дозволяє будувати проєкт за модульною архітектурою, де кожний функціональний блок системи — авторизація, ведення щоденника, робота

з настроєм, рекомендаційний модуль або аналітика — може мати власну чітко визначену структуру [14]. Це спрощує тестування, полегшує додавання нових модулів та дозволяє розробникам працювати над різними частинами проєкту незалежно один від одного, не створюючи конфліктів чи дублювання коду.

Усе це робить TypeScript чудовим інструментом для розробки сучасної веб-платформи, його підвищує якість програмного продукту, зменшує кількість помилок, робить код прозорим і зрозумілим, а також забезпечує надійну основу для подальшого розвитку і масштабування системи. Саме тому TypeScript став базовою мовою для клієнтської та серверної частини розробленої платформи, забезпечивши стабільність роботи, зручність програмування та високу ефективність у реалізації всіх компонентів платформи.

2.2 Фреймворк для створення інтерфейсів Next.js

У процесі розробки клієнтської частини веб-платформи підтримки ментального здоров'я було використано фреймворк Next.js, який є надбудовою над бібліотекою React та забезпечує значно розширені можливості у створенні сучасних, швидких і оптимізованих веб-додатків. Next.js став ключовою технологією для побудови інтерфейсу платформи, оскільки він поєднує високу продуктивність, зручність розробки та гнучку архітектуру, що дозволяє реалізувати масштабовану систему з великою кількістю сторінок, компонентів і модулів.

Однією з головних причин використання Next.js є його підтримка серверного та гібридного рендерингу [15]. Це означає, що частина сторінок може генеруватися на сервері, що значно прискорює завантаження інтерфейсу та покращує взаємодію користувача з додатком. Для платформи підтримки ментального здоров'я це має особливе значення, оскільки користувачі можуть взаємодіяти з додатком на різних пристроях і в умовах нестабільного інтернет-з'єднання. Використання серверного рендерингу забезпечує швидку відповідь інтерфейсу навіть у випадках, коли клієнтське обладнання має низьку продуктивність.

Next.js надає зручну файлову маршрутизацію, в якій структура каталогів

відповідає структурі веб-сторінок. Це дозволило чітко організувати модулі платформи, розділивши публічні сторінки (авторизація, реєстрація, онбординг) і захищені сторінки (щоденник, аналітика настрою, рекомендації, чат). Наявність окремого каталогу для приватних маршрутів спрощує реалізацію middleware-логіки, яка забезпечує перевірку токенів доступу та контроль автентифікації. Це сприяє підвищенню безпеки платформи та створює зрозумілу структурну модель для подальшої підтримки і розвитку системи.

Перевагою використання Next.js є також ефективне завантаження даних і підтримка сучасних механізмів оптимізації, таких як попереднє завантаження сторінок і кешування. У розробленому додатку активно використовується можливість попереднього завантаження даних для тих сторінок, які відображають статистику настрою та рекомендації. Це дозволяє забезпечити плавність роботи інтерфейсу та мінімізувати затримки під час переключення між модулями. З урахуванням того, що користувачі часто працюють зі щоденником, графіками та рекомендаціями, швидкість доступу до цих сторінок має вирішальне значення для зручності та якості взаємодії.

Next.js також підтримує інтеграцію з різними сервісами API та дозволяє легко підключати зовнішні бібліотеки, такі як TanStack Query, яка була використана в цьому проєкті для управління станом даних та оптимізації їх отримання з серверів. Це забезпечує стабільну роботу з API платформи, створеної на Nest.js, та дозволяє автоматично оновлювати дані під час зміни записів настрою, виконання вправ чи отримання нових рекомендацій.

Ще однією суттєвою перевагою Next.js є підтримка сучасної архітектури App Router, яка пропонує більш гнучкий підхід до побудови сторінок, компонентів і логіки маршрутизації [16]. Це дозволило впорядкувати клієнтський код, розділити логіку обробки даних та UI-компоненти, забезпечивши краще масштабування додатку. Таке архітектурне рішення особливо важливе в системах, які мають численні візуальні елементи, велику кількість інтерактивних функцій та роботу з динамічними даними.

У межах розробки веб-платформи Next.js забезпечив можливість створити

інтерфейс, який відповідає вимогам сучасних веб-стандартів: адаптивність, висока продуктивність, оптимізація роботи з великими обсягами даних та інтуїтивно зрозумілий користувацький досвід. Завдяки своїм можливостям цей фреймворк став ключовим елементом у створенні інтерфейсу платформи, що включає різноманітні функціональні модулі, такі як аналіз настрою, генерація рекомендацій, чат-інтерфейс, візуалізація статистики та робота з щоденником.

Використання Next.js дозволило не лише оптимізувати роботу клієнтської частини, але й створити гнучку архітектуру, яка може легко розширюватися з урахуванням потреб користувачів та подальшого розвитку платформи. Завдяки своїй продуктивності, зручності та підтримці сучасних веб-технологій Next.js став ключовим інструментом у реалізації всієї клієнтської логіки веб-платформи підтримки ментального здоров'я.

2.3 Фреймворк для створення серверної архітектури Nest.js

Для реалізації серверної частини веб-платформи підтримки ментального здоров'я було обрано Nest.js — сучасний прогресивний фреймворк для Node.js, який вирізняється високим рівнем структурованості, модульності та підтримкою TypeScript. Завдяки суворому дотриманню принципів об'єктно-орієнтованого програмування та застосуванню архітектурних шаблонів, характерних для корпоративних рішень, Nest.js створює надійний фундамент для побудови складних серверних застосунків, таких як рекомендаційні системи або інтерактивні платформи з великою кількістю взаємопов'язаних модулів.

Однією з ключових переваг Nest.js є модульна архітектура, яка дозволяє логічно розділяти систему на окремі компоненти: авторизацію, управління користувачами, щоденник, відстеження настрою, рекомендаційний модуль, аналітику та чат-інтерфейс. Кожен модуль працює як незалежний блок із власними контролерами, сервісами, моделями та маршрутами. Завдяки цьому до платформи легко додавати новий функціонал, не порушуючи структуру та логіку існуючого коду. Це надзвичайно важливо для систем, які плануються до подальшого

розвитку, а також для платформ, що обробляють індивідуальні психологічні дані й повинні гарантувати стабільність роботи всіх компонентів.

Nest.js використовує контролери для визначення маршрутизації HTTP-запитів та сервіси для реалізації бізнес-логіки. Такий поділ дає можливість відокремлювати логіку обробки даних від їх взаємодії з інтерфейсом, що значно полегшує тестування й підтримку застосунку.

Важливо зазначити, що серверна частина платформи працює з великою кількістю структурованих даних: записами настрою, журналами, історією рекомендацій, повідомленнями чату, поведінковими характеристиками користувачів і результатами аналітики. Використання сервісів дозволяє централізовано реалізовувати обробку цих даних, забезпечуючи повторне використання коду та узгодженість логіки в різних модулях.

Суттєвою перевагою Nest.js є підтримка декораторів і залежностей, що забезпечує гнучке управління залежностями між компонентами. Це дає змогу автоматично підключати сервіси до контролерів, використовувати спільні модулі в різних частинах системи, оптимізувати роботу з базою даних та із зовнішніми API. У межах платформи підтримки ментального здоров’я цей механізм відіграє особливо важливу роль, оскільки забезпечує правильну взаємодію між модулями рекомендацій, аналітики та іншими компонентами, що працюють з великою кількістю обчислень.

Nest.js також має вбудовану підтримку валідації даних та обробки помилок. Використання DTO-класів у поєднанні з декораторами class-validator дозволяє контролювати коректність вхідних даних, що надходять від клієнтських форм, зокрема записів настрою, щоденникових повідомлень або запитів до чату [17]. Такий підхід гарантує, що сервер отримує дані у передбачуваному форматі, а помилки типу або структури виявляються на ранніх етапах. Це знижує ризик некоректного формування рекомендацій або аналітичних графіків, адже алгоритми машинного навчання вимагають строгої відповідності структури даних.

Ще однією важливою можливістю Nest.js є інтеграція зі Swagger, яка забезпечує автоматичну генерацію документації для API. Це значно спрощує

тестування платформи, полегшує взаємодію між клієнтською частиною, сервером та зовнішніми сервісами. Завдяки цьому розробники можуть швидко перевіряти роботу запитів, тестувати передавання даних і контролювати відповідність форматів відповідей. Для платформи, що має десятки різних API-маршрутів — від авторизації до генерації рекомендацій — це є критично важливою функцією.

Nest.js чудово підходить для роботи зі сторонніми бібліотеками, зокрема ORM-інструментом Sequelize, який використовується у проєкті для взаємодії з базою даних PostgreSQL. Завдяки цьому серверна частина має гнучку та масштабовану структуру для зберігання користувацьких профілів, записів настрою, історії рекомендацій, даних зворотного зв'язку та інших структур. Взаємодія між Nest.js і базою даних є швидкою, надійною та захищеною від помилок завдяки валідації, виключенням та строгим правилам типізації.

Також Nest.js забезпечує чудову основу для реалізації рекомендаційної системи. У платформі використовується алгоритм TF-IDF та косинусної подібності, для яких потрібне швидке обчислення векторів текстів, порівняння їх релевантності та генерація підсумкових рекомендацій. Завдяки модульній структурі фреймворку логіка обробки текстів, збереження рекомендацій та їх оцінювання може бути ізольована у власному модулі й легко масштабуватися в майбутньому — наприклад, для переходу на нейронні мережі або інші алгоритми NLP.

У результаті використання Nest.js забезпечило створення стабільної, організованої та легко масштабованої серверної архітектури. Фреймворк дозволяє підтримувати великий обсяг логіки, керувати численними модулями та гарантувати коректну взаємодію всіх елементів платформи, що є необхідним для системи, яка працює з персональними даними користувачів і виконує функції аналітики, рекомендацій та інтерактивної підтримки.

2.4 Інструмент документування та тестування API Swagger

Swagger був використаний у розробці серверної частини веб-платформи для

автоматичного створення документації до всіх API-ендпоінтів системи. Це інструмент, який базується на специфікації OpenAPI та дозволяє описувати структуру запитів, відповідей, параметрів та помилок у стандартизованому форматі. Його застосування значно полегшує процес взаємодії між клієнтською та серверною частинами, оскільки розробники отримують повну та актуальну інформацію про всі маршрути без необхідності вручну описувати їх у зовнішніх документах.

Swagger забезпечує зручний веб-інтерфейс, де можна одразу тестувати ендпоінти: надсилати запити, передавати параметри, переглядати відповіді та аналізувати коди помилок. Це особливо корисно під час створення та вдосконалення модулів платформи, таких як авторизація, реєстрація, робота з настроєм, щоденником, рекомендаціями та аналітикою. Завдяки цьому зменшується кількість технічних помилок, пов’язаних із некоректними даними або неправильними форматами запитів.

Використання Swagger у проєкті дозволило підтримувати API в актуальному стані, забезпечити прозорість взаємодії між командами розробників та спростити тестування функціональних модулів на кожному етапі створення платформи [18]. Це зробило процес розробки більш структурованим та передбачуваним, а серверну логіку — легшою для обслуговування та розширення.

2.5 Інструмент для стилізації Tailwind CSS

Tailwind CSS був використаний у розробці клієнтської частини веб-платформи для створення сучасного, мінімалістичного та адаптивного інтерфейсу [19]. Його головною особливістю є підхід utility-first, який дозволяє будувати дизайн безпосередньо в класах елементів, не створюючи окремих файлів стилів для кожного компонента. Це суттєво прискорює процес розробки та забезпечує високу гнучкість інтерфейсу.

У межах веб-платформи Tailwind CSS використовується під час створення сторінок настрою, щоденника, аналітики та рекомендацій, забезпечуючи

узгодженість стилів і швидку адаптацію елементів до різних розмірів екранів. Tailwind також підтримує створення власних тем і дизайн-систем, що дозволяє дотримуватися єдиного стилю в усьому додатку без надмірного дублювання коду. Завдяки своїй ефективності Tailwind CSS став одним із ключових інструментів побудови інтерфейсу платформи.

2.6 Бібліотека для візуалізації аналітичних даних Chart.js

Для реалізації аналітичного модуля веб-платформи було використано Chart.js — популярну бібліотеку для створення інтерактивних графіків. Вона дозволяє легко відображати статистичні дані, працює з різними типами графіків і забезпечує плавну анімацію та високу продуктивність.

У проєкті Chart.js застосовується для візуалізації динаміки настрою, активності користувача. За допомогою цієї бібліотеки створено лінійні графіки, індикатори та діаграми, які забезпечують зрозумілу й наочну демонстрацію змін психоемоційного стану. Оскільки Chart.js добре інтегрується з React та Next.js, її застосування дозволило швидко створити інформативні візуалізації без складних налаштувань.

2.7 ORM для роботи з реляційними базами даних Sequelize

У розробці серверної частини веб-платформи підтримки ментального здоров’я важливу роль відіграє ORM-бібліотека Sequelize, яка забезпечує зручну, структуровану та безпечну взаємодію з реляційною базою даних PostgreSQL. Використання ORM дозволяє розробникам оперувати даними на рівні моделей та об’єктів, не формуючи вручну складних SQL-запитів, що знижує ризик помилок та прискорює процес розробки [20].

Sequelize підтримує повний набір CRUD-операцій, роботу зі складними зв’язками між таблицями, міграціями та перевіркою даних. Це робить його оптимальним вибором для проєктів, де база даних містить велику кількість

сутностей та сильно пов’язаних структур. У межах веб-платформи до таких сутностей належать користувачі, записи настрою, текстові записи щоденника, рекомендації, зворотний зв’язок, навчальні матеріали та історія повідомлень чату. Використання моделей дає змогу чітко описати структуру кожної сутності, її властивості та зв’язки з іншими таблицями, що спрощує подальший розвиток системи й підтримку цілісності бази даних.

Sequelize забезпечує також механізми для обробки складних реляційних зв’язків: «один-до-багатьох», «багато-до-багатьох» та вкладених асоціацій. У платформі це активно використовується, наприклад, для пов’язування користувача з його записами настрою, журналом, виконаними вправами, сформованими рекомендаціями та діями в аналітичному модулі. Такий підхід дає можливість отримувати дані різного рівня складності одним запитом, що підвищує продуктивність і спрощує обробку інформації на стороні сервера.

Механізм міграцій у Sequelize дозволяє контролювати зміни структури бази даних у часі, зберігаючи історію цих змін. Це необхідно для довготривалих проєктів, де схема даних поступово розширюється, а нові модулі вимагають додавання нових таблиць або полів. Завдяки міграціям база даних залишається узгодженою на всіх середовищах.

Sequelize також підтримує роботу з транзакціями, що є критичним аспектом для функціоналу, який потребує атомності виконання операцій. У платформі це актуально, наприклад, під час обробки послідовних дій у модулі рекомендацій або записів статистики настрою, де важливо, щоб усі операції виконувалися коректно та без втрати даних [21].

Завдяки гнучкості, підтримці TypeScript і широкому функціоналу Sequelize став ефективним інструментом для побудови надійної та масштабованої бази даних веб-платформи.

Його використання забезпечує стабільність роботи серверної частини, спрощує обробку структурованої інформації та дозволяє швидко адаптувати систему до змін і розширення функціоналу.

2.8 Реляційна система керування базами даних PostgreSQL

Під час розробки веб-платформи підтримки ментального здоров'я ключову роль відіграє система керування базами даних PostgreSQL, яка є однією з найбільш надійних і продуктивних реляційних СУБД з відкритим вихідним кодом. PostgreSQL була обрана завдяки своїй стабільності, високій швидкодії, розширюваності та відповідності вимогам сучасних веб-додатків, що працюють з великими обсягами структурованих даних. Для систем, які зберігають персональні психологічні дані користувачів, показники ефективності та текстові записи, PostgreSQL забезпечує необхідний рівень безпеки, цілісності та продуктивності.

Однією з переваг цієї системи є підтримка складних типів даних, включно з JSONB, який дозволяє зберігати структуровану інформацію у зручному форматі. У веб-платформі це використовується для фіксації додаткових атрибутів настрою, збереження параметрів рекомендаційних моделей, історії взаємодій або специфічних аналітичних метаданих. Такий підхід дає змогу гнучко комбінувати реляційні та документні структури, не втрачаючи переваг класичної SQL-парадигми.

PostgreSQL також вирізняється сильними механізмами підтримки транзакцій, що забезпечують надійність і цілісність даних навіть у випадках складних послідовностей операцій. У контексті роботи платформи це є особливо важливим під час запису даних про настрій, збереження результатів аналітики або формування рекомендацій, де будь-яка помилка може призвести до пошкодження даних або некоректної оцінки стану користувача. Транзакційна модель PostgreSQL гарантує, що всі операції виконуються послідовно, надійно та атомарно.

База даних системи містить велику кількість сутностей, серед яких користувацькі профілі, історія настроїв, текстові записи щоденника, сформовані рекомендації, дані про виконані вправи, повідомлення чату та інші логічно пов'язані об'єкти. PostgreSQL дозволяє оптимально будувати між ними реляційні зв'язки, що забезпечує швидке виконання складних запитів та зручність у вибірці пов'язаних даних. Це особливо важливо для модулів аналітики та рекомендацій, де

часто необхідно об’єднувати інформацію з різних таблиць для формування релевантних результатів.

Додатковою перевагою є висока масштабованість PostgreSQL, яка дозволяє адаптувати базу даних до зростання кількості користувачів та збільшення обсягів інформації [22]. Оскільки платформа може з часом розширюватися за рахунок нових модулів, збільшення кількості записів щоденника чи історичних даних настрою, PostgreSQL забезпечує стабільну роботу навіть при інтенсивних навантаженнях. Підтримка індексації, оптимізації запитів та розподілених обчислень робить її придатною для довгострокових проєктів.

У поєднанні з ORM-бібліотекою Sequelize PostgreSQL виступає надійним центральним компонентом серверної частини платформи. Такий тандем забезпечує простоту розробки, високу продуктивність, збереження цілісності даних і можливість подальшого розширення системи. У результаті PostgreSQL є оптимальним рішенням для зберігання та обробки різнопланової інформації у веб-платформі підтримки ментального здоров’я, задаючи фундамент її стабільності та адаптивності.

2.9 Інтеграція Gemini AI у веб-платформу

Gemini AI — це сучасна мультимодальна модель штучного інтелекту, розроблена компанією Google DeepMind, яка поєднує можливості обробки тексту, коду, зображень та інших форматів даних. У контексті розробки веб-платформи для підтримки ментального здоров’я Gemini AI виконує роль розумного помічника у модулі чату, а також може бути використаний як основа для генерації персоналізованих порад, відповіді на запити користувача й адаптації освітнього контенту [23].

Інтеграція Gemini AI здійснюється через API, що дозволяє відправляти текстові запити від користувача на стороні фронтенду до серверної частини, де дані формуються у відповідному форматі та передаються до моделі через HTTP-запит [24]. У відповідь Gemini AI надсилає сформовану відповідь, яка відображається у

чаті у вигляді діалогу. Важливою особливістю є можливість зберігати контекст сесії, що дозволяє моделі аналізувати попередні дії та формувати релевантні відповіді.

У рамках платформи використання Gemini AI дозволяє реалізувати підтримку на природній мові, персоналізовані «швидкі дії» (наприклад, перехід до вправи, яка відповідає емоційному стану), а також динамічне пояснення аналітики настрою. Модель демонструє високу гнучкість і розуміння психоемоційного контексту, що робить її ефективним доповненням до класичних алгоритмів рекомендацій.

З технічної точки зору, реалізація передбачає обробку запитів за допомогою асинхронних функцій, валідацію вхідних даних, захист від небажаного контенту, а також логування активності для подальшого аналізу ефективності взаємодії з користувачем. Такий підхід відповідає сучасним вимогам до інтеграції LLM (Large Language Models) у веб-системи з високими вимогами до безпеки, етичності та точності.

Таким чином, Gemini AI виступає важливою технологічною складовою у побудові інтелектуальної взаємодії між платформою та користувачем, сприяючи створенню більш природного, адаптивного та підтримувального досвіду.

Висновки до розділу 2

У другому розділі здійснено детальний аналіз засобів, технологій і фреймворків, які можуть бути використані під час розробки веб-платформи підтримки ментального здоров’я. Розглянуто доступні інструменти для створення клієнтської та серверної частин системи, зокрема мову програмування TypeScript, фреймворки Next.js і Nest.js, бібліотеки для стилізації інтерфейсу, роботу з базами даних та інструменти аналітики.

Проведений аналіз показав, що TypeScript забезпечує підвищену надійність, зменшення кількості помилок та покращену підтримку масштабованих застосунків порівняно з JavaScript, що є критичним для розробки комплексних платформ. Вибір

Next.js обґрунтовано його здатністю поєднувати серверний рендеринг, статичне генерування сторінок і гнучку маршрутизацію, що підвищує продуктивність клієнтської частини та покращує взаємодію з користувачем. Nest.js визначено як оптимальне рішення для серверної частини завдяки модульній архітектурі, чіткому розділенню логіки, підтримці REST API та високій масштабованості.

Окрему увагу приділено системам роботи з даними: PostgreSQL вибрано як надійну та продуктивну реляційну СУБД, тоді як Sequelize забезпечує зручне управління моделями, міграціями та взаємодією з базою на рівні ORM. Інструменти аналітики і візуалізації, зокрема Chart.js та інші бібліотеки, дозволяють реалізувати інформативні графічні інтерфейси для відстеження стану користувача.

Проведений порівняльний аналіз доступних інструментів підтвердив, що обраний технологічний стек забезпечує необхідні характеристики — продуктивність, гнучкість, масштабованість і безпеку, — що робить його оптимальним для реалізації функціональних вимог платформи.

Таким чином, у межах розділу 2 обґрунтовано вибір засобів розробки програмного забезпечення, що формують технологічну основу майбутньої системи та забезпечують ефективну реалізацію її функціональних можливостей.

3 ТЕОРЕТИЧНІ ТА МАТЕМАТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ПЛАТФОРМИ ПІДТРИМКИ МЕНТАЛЬНОГО ЗДОРОВ'Я

Третій розділ присвячено теоретичним і математичним принципам, що лежать в основі роботи рекомендаційної системи та інших інтелектуальних компонентів веб-платформи підтримки ментального здоров'я. Особливу увагу приділено методам обробки текстових даних, статистичним моделям та алгоритмам, які забезпечують можливість аналізу контенту користувача й формування персоналізованих рекомендацій. У межах цього розділу розглянуто функціонування базових алгоритмів, що складають фундамент аналітичного модуля системи, починаючи з математичних основ TF-IDF.

3.1 Математичні основи алгоритму TF-IDF

Алгоритм TF-IDF (Term Frequency-Inverse Document Frequency) є одним з найпоширеніших і найбільш ефективних методів векторного представлення текстової інформації у задачах інформаційного пошуку, класифікації документів, кластеризації та формування контентно-орієнтованих рекомендацій [25]. Його фундаментальна ідея полягає у визначенні ваги кожного терму в документі таким чином, щоб підкреслити слова, які є значущими для конкретного тексту, та знизити вплив часто вживаних, але неінформативних слів. У контексті розробки веб-платформи підтримки ментального здоров'я TF-IDF виступає базовим механізмом для аналізу щоденникових текстів користувача, визначення ключових елементів психоемоційного контексту та підбору рекомендацій, релевантних до його стану.

Основою алгоритму є подання кожного документа у вигляді вектора, координатами якого є ваги відповідних термів [26]. При цьому документи можуть бути будь-якими текстовими одиницями — записами настрою, щоденниковими фрагментами або описами вправ та курсів платформи. Математична модель TF-IDF дозволяє перетворювати неструктурований текст у числову форму, придатну для

подальших обчислень, таких як визначення схожості документів за допомогою косинусної метрики, кластеризації або ранжування.

Для формування TF-IDF-моделі першочергово здійснюється побудова словника термів, який містить усі унікальні слова з корпусу документів. Нехай корпус складається з N документів, кожен з яких містить певну кількість термів. Для будь-якого терму t та документа d величина TF визначає частоту входження терму в документ і може бути обчислена за формулою:

$$TF(t,d) = \frac{n_{t,d}}{\sum_i n_{i,d}},$$

де $n_{t,d}$ — кількість входжень терму t у документ d,

$\sum_i n_{i,d}$ — загальна кількість термів у документі.

Це нормалізоване значення, яке дає можливість порівнювати різні документи між собою незалежно від їх довжини.

Однак частотний компонент не може повністю відобразити важливість терму, оскільки деякі слова можуть часто зустрічатися у всьому корпусі, але не містити цінної інформації (наприклад, «людина», «сьогодні», «почувався» у контексті щоденника). Тому важливим компонентом моделі є IDF — зворотна частота документів. Вона враховує, у скількох документах корпусу зустрічається терм. Формула для IDF має вигляд:

$$IDF(t) = \ln\left(\frac{N}{df_t}\right),$$

де N — загальна кількість документів корпусу,

df_t — кількість документів, у яких зустрічається терм t.

Чим менше документів містять терм, тим вищою є його інформативність. Логарифмічна функція згладжує різкі перепади значень і робить модель більш стійкою до вибірки.

Після обчислення окремих компонентів TF та IDF отримується фінальна вага TF-IDF, яка дорівнює добутку цих двох величин:

$$TF - IDF(t,d) = TF(t,d) \times IDF(t).$$

У результаті документ перетворюється в високорозмірний вектор, де кожен елемент відповідає певному терму зі словника. Такий підхід дає змогу

використовувати геометричні методи для визначення відстані або схожості між документами. У нашій платформі ці вектори застосовуються в модулі рекомендацій для порівняння текстів користувача з описами психологічних матеріалів: наприклад, якщо користувач пише «відчуваю тривогу через перевтому», алгоритм знаходить ресурси, що містять подібний набір ключових термів («тривожність», «виснаження», «стрес», «робоче навантаження»).

Важливим етапом формування TF-IDF є попередня обробка тексту, яка включає токенізацію, нормалізацію регістру, видалення стоп-слів, лематизацію або стемінг. Без цього словник може містити дублікати або шумові елементи, що знижує ефективність алгоритму. Наприклад, у психологічному контексті слова «занепокоєння», «тривога», «тривожний» без лематизації будуть вважатися різними термами, хоча насправді відображають один емоційний стан.

Особливістю застосування TF-IDF у тематиці ментального здоров'я є те, що текстові дані часто містять значну кількість емоційно забарвлених слів, метафор або фраз, які можуть мати різну структуру та вживатися у широкому діапазоні контекстів. Для таких випадків TF-IDF залишається ефективним інструментом, оскільки він не намагається інтерпретувати глибинний зміст тексту, а зосереджується на статистичних закономірностях у вживанні термів. Це робить його особливо цінним у початкових етапах формування рекомендацій, коли система має обмежену кількість даних і потребує стабільного механізму порівняння текстів.

У реалізованій платформі TF-IDF використовується як невід'ємний компонент рекомендаційного модуля. На основі векторів користувацьких записів платформа визначає найбільш релевантні вправи, курси або статті, подібні до емоційного стану користувача. Обчислені значення TF-IDF також логуються у спеціальну таблицю, що дозволяє аналізувати роботу рекомендаційних алгоритмів та здійснювати подальше удосконалення моделі. Така інтеграція TF-IDF забезпечує прозорість роботи рекомендаційної системи, легкість налагодження та швидку адаптацію алгоритму у разі збільшення обсягу даних або зміни тематичної структури контенту.

Таким чином, TF-IDF виступає фундаментальною складовою алгоритмічної частини платформи, дозволяючи виконувати математично обґрунтований аналіз текстових записів користувача та формувати персоналізовані рекомендації на основі об'єктивних статистичних характеристик тексту.

3.2 Міра косинусної подібності для оцінювання схожості текстів

Косинусна подібність є одним із фундаментальних математичних інструментів, що застосовуються в задачах обчислення схожості текстових документів, побудови рекомендаційних систем, класифікації текстів та аналізу інформаційного змісту [27]. Метод базується на уявленні текстових даних у вигляді векторів у багатовимірному просторі, де кожен вимір відповідає певному терму словника, а значенням координати є вага цього терму в документі, отримана, як правило, за допомогою TF-IDF. Саме комбінація TF-IDF і косинусної подібності дозволяє точно визначити, наскільки текстові записи користувача відповідають змісту довідкових матеріалів чи психологічних рекомендацій, що є критично важливим для роботи веб-платформи підтримки ментального здоров'я.

Основна ідея методу полягає у тому, що подібність між двома документами визначається не абсолютною різницею в їхніх значеннях, а кутом між відповідними векторами у векторному просторі. Якщо два документи містять однакові ключові терми з подібними вагами, їхні вектори спрямовані приблизно в один бік, а косинус кута між ними наближається до 1, що означає високу схожість. Якщо ж документи містять мало спільних термів або мають суттєво різний зміст, їхні вектори проходитимуть під великим кутом, і значення косинусної подібності наближатиметься до 0.

Формально косинусна подібність між двома векторами $\underset{d_1}{\rightarrow}$ та $\underset{d_2}{\rightarrow}$ визначається за формулою:

$$\cos(\theta) = \frac{\overrightarrow{d_1} \times \overrightarrow{d_2}}{\left\|\overrightarrow{d_1}\right\| \times \left\|\overrightarrow{d_2}\right\|},$$

де $\underset{d_1}{\rightarrow} \times \underset{d_2}{\rightarrow}$ — скалярний добуток векторів,

$\left\|\underset{d_1}{\rightarrow}\right\|$ та $\left\|\underset{d_2}{\rightarrow}\right\|$ — їхні евклідові норми.

Такий підхід дозволяє оцінити саме напрямок векторів, а не їх абсолютні величини, що є надзвичайно важливим для текстів, довжина яких може значно відрізнятися. Наприклад, короткий щоденниковий запис з кількох речень може виявитися семантично ближчим до детальної статті про психологічну підтримку, ніж інший довший, але тематично не пов'язаний текст.

Косинусна подібність має кілька важливих властивостей, які роблять її практично незамінною в системах текстового аналізу. По-перше, вона нечутлива до абсолютної довжини документа, що дає змогу порівнювати як короткі нотатки настрою, так і розгорнуті психологічні статті. Завдяки нормалізації векторів косинусна міра оцінює лише пропорції ваг термів, а не їх загальну кількість. Це робить її особливо корисною у випадках, коли користувачі платформи залишають короткі фрази на кшталт «відчуваю сильну тривогу», які повинні бути коректно інтерпретовані й порівняні з довшими описами вправ або рекомендацій.

По-друге, косинусна подібність добре працює у високовимірних просторах, характерних для моделей TF-IDF, де кількість можливих термів може сягати тисяч. Геометричний підхід дозволяє швидко та ефективно обчислювати схожість навіть для великих корпусів документів, що особливо корисно у платформах, де обсяг даних зростає з часом. У розробленій системі косинусна подібність використовується при порівнянні текстових записів користувача з усіма матеріалами, що доступні в базі даних: це можуть бути описи психологічних практик, вправи для саморегуляції, міні-курси або статті. Завдяки цьому система може кожного разу визначати ті ресурси, які найточніше відповідають поточним емоційним потребам користувача.

У практичній реалізації модулю рекомендацій косинусна подібність відіграє роль механізму ранжування: після обчислення TF-IDF векторів для записів користувача та документів у корпусі система обчислює міру подібності між кожним записом користувача та кожним доступним матеріалом. Результати

впорядковуються за спаданням, і користувач отримує перелік матеріалів, які найбільше відповідають його стану. У випадку низьких значень подібності система може визначити відсутність достатніх збігів у корпусі контенту та запропонувати загальні рекомендації або універсальні вправи для підвищення емоційного балансу.

У психологічній тематиці косинусна подібність проявляє себе як особливо ефективний метод, оскільки текстові записи можуть бути дуже різними за стилем, довжиною та структурою. Людина може описати свої переживання у довільній формі, використовуючи емоційні вирази, метафори або нетипові слова. В таких умовах саме геометрична інтерпретація тексту дозволяє вловити семантичну схожість навіть тоді, коли формальні збіги слів мінімальні. Наприклад, вислови «я виснажений» і «почуваю тривогу через перевантаження» можуть мати різний набір ключових слів, проте завдяки TF-IDF та косинусній подібності система все одно може визначити близькість їхнього змісту і підібрати релевантні рекомендації.

Окремої уваги заслуговують властивості косинусної міри, пов'язані з маштабованістю. На відміну від метрик на кшталт евклідової відстані, косинусна подібність зберігає ефективність навіть при роботі з великими корпусами психологічних матеріалів і зростанням кількості користувацьких записів. У платформі реалізовано оптимізований процес обчислення подібності: вектори TF-IDF попередньо зберігаються та переобчислюються лише в разі оновлення контенту чи додавання суттєвих змін до записів, що значно знижує навантаження на сервер.

У підсумку косинусна подібність виступає центральним математичним інструментом у системі персоналізованих рекомендацій веб-платформи. Вона дозволяє перетворити текстові емоційні описи на числову модель, порівнювати їх з базою психологічних знань та підбирати найбільш доречні матеріали. Завдяки своїм властивостям косинусна подібність забезпечує точність, масштабованість і стійкість рекомендаційної системи, що є критично важливим у сфері ментального здоров'я, де релевантність порад безпосередньо впливає на ефективність підтримки користувача.

3.3 NLP-моделі та методи попередньої обробки текстових даних

Методи обробки природної мови (NLP, Natural Language Processing) є центральним елементом систем, що працюють з текстовими даними. У веб-платформі підтримки ментального здоров'я NLP застосовується для аналізу щоденникових записів, повідомлень у чаті, нотаток про настрій та інших текстових елементів, що відображають психоемоційний стан користувача. Якість рекомендаційної системи безпосередньо залежить від того, наскільки коректно та повно були попередньо оброблені ці тексти. Саме тому NLP-процес виступає фундаментальним шаром, який пов'язує користувацький контент із алгоритмами TF-IDF і косинусної подібності, що реалізовані в модулі рекомендацій платформи.

Обробка природної мови складається з кількох етапів, кожен з яких має своє математичне та лінгвістичне підґрунтя [29]. Першим етапом є токенізація — поділ тексту на окремі одиниці (токени), що зазвичай відповідають словам. Токенізація є критично важливою, адже алгоритми TF-IDF працюють саме з частотами термів, кожен з яких представляє окремий токен. Без коректної токенізації неможливо побудувати векторну модель документів. Текстові записи користувачів у сфері ментального здоров'я часто містять емоційні, розмовні та іноді неформальні конструкції («мене накрила тривога» «повністю вигорів», «відчуваю пустоту»). Тому токенізатор має враховувати ці мовні особливості.

Наступним етапом є нормалізація тексту, яка передбачає приведення всіх термів до нижнього регістру, видалення символів пунктуації, зайвих пробілів, цифр та інших елементів, що не несуть корисного смислового навантаження. Така нормалізація забезпечує узгодженість словника: слова «Тривога», «тривога» та «ТРИВОГА» будуть розглядатися як один і той самий терм. Це особливо важливо для аналізу настрою, оскільки емоційні маркери (тривога, страх, виснаження, апатія) мають зберігати стабільне представлення в корпусі.

Одним із ключових кроків є видалення стоп-слів — слів, що не додають смислової ваги до документа. До стоп-слів належать загальні службові частини мови: «і», «що», «але», «коли», «мене», «це». Їх вилучення дозволяє зосередити

увагу на значущих словах, що несуть емоційне або контекстуальне навантаження. Наприклад, у фразі «Мене охопила сильна тривога через роботу» ключовими будуть «тривога», «робота», «сильна», тоді як решта слів лише розмивають зміст.

Іншою важливою процедурою є лематизація або стемінг. Лематизація — приведення слова до його нормальної словникової форми (леми), тоді як стемінг — грубіше скорочення слова до словотворчої основи. У психологічних текстах лематизація дає точніші результати, оскільки дозволяє об'єднати слова «тривоги», «тривожний» у базову форму «тривога». Це підвищує якість побудови TF-IDF моделі, зменшує розмір словника й покращує точність обчислення косинусної подібності.

Попередня обробка також включає процедури очищення тексту від шуму, наприклад емодзі, символів або фраз, які не є лінгвістично значущими. Хоча емодзі можуть бути непрямими емоційними маркерами, для математичних моделей TF-IDF вони не несуть цінності.

У процесі NLP формується корпус документів — сукупність усіх текстів, що належать як користувачам, так і контентним матеріалам (вправам, поясненням, рекомендаціям). Саме на основі цього корпусу будуються статистичні моделі TF-IDF. Платформа містить різні джерела текстових даних: щоденникові записи, опис настрою, рекомендації, навчальні матеріали. Ці документи утворюють багатошарову текстову базу, яка дозволяє формувати персоналізовані поради, враховуючи індивідуальний контекст користувача.

У психологічному контексті NLP має певні особливості, текстові записи користувачів часто є короткими, емоційно насиченими, іноді фрагментарними. Користувач може писати «погано», «нічого не хочу», «втомився», «паніка», «просто важко». Такі фрази несуть обмежену кількість термів, але форми висловлювань здатні відображати важливі психоемоційні закономірності. Завдяки методам NLP система розпізнає ключові слова, що мають найбільше значення для аналізу настрою, а надалі — для побудови рекомендацій.

Застосування NLP дозволяє виявити емоційні тригери, теми, частоту згадуваних проблем, інтенсивність негативних та позитивних описів. Це дає

можливість побудувати більш інформативні профілі користувачів, які згодом використовуються для підсилення рекомендацій і аналітики стану. У рамках роботи платформи NLP виступає проміжною ланкою між первинними текстовими даними та математичними моделями TF-IDF і косинусної подібності, які формують основу рекомендаційного алгоритму.

Складність обробки посилюється тим, що користувацькі тексти можуть містити багато контекстуальних і психологічно забарвлених фраз, що не завжди однозначно інтерпретуються. Наприклад, слова «мені порожньо» або «відключився емоційно» мають глибоке емоційне значення, але з математичної точки зору вони обробляються як звичайні токени. Проте завдяки комбінації NLP і статистичних алгоритмів платформа здатна визначити, що ці записи наближені до тематики депресивних станів або емоційного вигорання, якщо такі мотиви присутні в описах відповідних психологічних матеріалів.

У підсумку NLP-моделі та методи попередньої обробки тексту дозволяють перетворити різнорідні, емоційні та часто неструктуровані текстові записи на формалізовані вектори ознак, придатні для точного порівняння та аналізу. Завдяки NLP веб-платформа отримує можливість не лише механічно співставляти слова, а й виявляти приховані закономірності у психоемоційному стані користувачів. Саме цей етап забезпечує основу для роботи TF-IDF і косинусної подібності, що надалі дозволяє платформі формувати персоналізовані, релевантні та точні рекомендації у сфері ментального здоров'я.

3.4 Структура та принципи роботи REST API

REST API (Representational State Transfer Application Programming Interface) є найбільш поширеним архітектурним стилем для побудови сучасних веб-сервісів та інтеграції клієнтських і серверних частин програмних систем. Використання REST у веб-платформі підтримки ментального здоров'я забезпечує стандартизовану, передбачувану і масштабовану взаємодію між інтерфейсом користувача, серверною частиною та базою даних. REST API виступає центральним

комунікаційним шаром, через який здійснюються всі операції: реєстрація користувача, додавання записів настрою, робота з щоденником, отримання рекомендацій, виконання вправ і обмін повідомленнями в чаті.

Основна ідея REST полягає в тому, що серверна частина надає набір чітко визначених ресурсів, які клієнт може запитувати, створювати, змінювати або видаляти через стандартні HTTP-методи [28]. Кожен ресурс описується у вигляді унікального URL, використовуючи зрозумілу ієрархічну структуру маршрутів. Наприклад, у системі реалізовано такі REST-шляхи, як /auth/login, /users/profile, /mood, /journal, /recommendations, які відображають логічні сутності, що існують у базі даних та відповідають доменній структурі Nest.js. Така організація дозволяє зрозумілу навігацію між різними частинами системи, а також полегшує тестування та масштабування серверної логіки.

REST API використовує певні семантичні правила для взаємодії з ресурсами. Найпоширенішими HTTP-методами є GET, POST, PUT, PATCH і DELETE. У веб-платформі GET використовується для отримання інформації — наприклад, для завантаження записів настрою чи отримання рекомендацій; POST — для створення нових об’єктів, таких як нові записи щоденника або повідомлення в чаті; PATCH і PUT — для оновлення існуючих ресурсів; DELETE — для їх видалення. Користувацькі запити структуровані таким чином, щоб клієнтська частина Next.js могла отримувати дані у форматі JSON, який є стандартом REST API і забезпечує легку інтеграцію з JavaScript/TypeScript.

Одним з фундаментальних принципів REST є статусність відповідей сервера. Кожен запит супроводжується кодом відповіді HTTP, який відображає результат виконання операції: 200 OK, 201 Created, 400 Bad Request, 401 Unauthorized, 404 Not Found, 500 Internal Server Error тощо. У платформі це реалізовано через Nest.js механізми фільтрації помилок та автоматичне формування відповідей, що забезпечує клієнту точну інформацію про успішність або помилковість операції. Коректне використання статус-кодів є невід’ємною частиною REST API, оскільки дозволяє клієнтському застосунку швидко реагувати на зміни та помилки, забезпечуючи надійність і передбачуваність взаємодії.

Важливим аспектом роботи REST API у цій платформі є аутентифікація та авторизація, реалізовані за допомогою JWT (JSON Web Token). Після успішного входу користувач отримує два токени — access і refresh, які передаються в заголовках HTTP-запитів та дозволяють серверу розпізнавати особу користувача без необхідності зберігати стан сеансу. Доступ до більшості приватних маршрутів, таких як /mood, /journal, /recommendations, захищений через Nest.js Guards, які перевіряють валідність токена. Такий підхід забезпечує високий рівень безпеки й одночасно зберігає масштабованість, оскільки сервер не потребує управління сесіями.

REST API також підтримує ідемпотентність запитів, тобто повторне виконання деяких операцій без зміни результату. Це особливо важливо при нестабільному інтернет-з'єднанні або дублюванні запитів клієнтом. Метод GET за визначенням є ідемпотентним, а PUT повинен виконувати однакову зміну ресурсу незалежно від кількості повторень. Це дозволяє платформі працювати стабільно навіть у стресових умовах або при підвищених навантаженнях.

REST API у платформі забезпечує також підтримку пагінації, фільтрації, сортування та пошуку. Наприклад, у модулі щоденника користувач може запитати записи за певний період або отримати лише останні N записів; у модулі настрою реалізовано фільтрацію за датами; у модулі рекомендацій можуть застосовуватися додаткові параметри для відбору найрелевантніших порад. Це забезпечує гнучкість роботи з великими обсягами інформації та підвищує зручність користування.

У платформі REST API також виконує функцію інтеграційного шару для інших модулів. Наприклад, аналітичний модуль отримує з API всі необхідні дані для побудови графіків настрою за допомогою Chart.js, а чат-модуль взаємодіє з сервером, надсилаючи користувацькі текстові повідомлення для обробки та формування відповіді. Таким чином, API є універсальним каналом передачі як структурованих, так і текстових даних.

Загалом структура й принципи роботи REST API забезпечують прозору, стандартизовану і масштабовану основу для функціонування веб-платформи. Саме завдяки REST клієнтські й серверні компоненти системи взаємодіють узгоджено, а

модулі авторизації, аналітики, рекомендацій і чату працюють як єдина цілісна система, забезпечуючи користувачу стабільний доступ до персоналізованої психологічної підтримки.

3.5 Архітектурні підходи

Архітектурний підхід відіграє ключову роль у побудові програмних систем, особливо коли йдеться про масштабовані веб-платформи, що містять багато функціональних модулів та працюють з великою кількістю різнорідних даних. У контексті розробки веб-платформи підтримки ментального здоров'я важливою задачею було забезпечення структурованості, високої розширюваності та зрозумілості серверної частини застосунку. Для досягнення цих цілей було проаналізовано кілька архітектурних підходів, серед яких класична модель MVC та сучасна модульна архітектура, реалізована фреймворком Nest.js.

Модель MVC (Model-View-Controller) традиційно використовується для створення веб-застосунків завдяки чіткому поділу відповідальностей. Model відповідає за дані та логіку роботи з ними, View відображає інформацію користувачу, а Controller обробляє вхідні запити й координує взаємодію між моделями і поданнями. Такий поділ сприяє зменшенню зв'язаності компонентів, що робить систему гнучкішою та легкою для тестування. У класичному MVC сервер відповідає не лише за обробку даних, а й за формування HTML-подання, що характерно для монолітних веб-застосунків.

Однак із розвитком сучасних веб-технологій, появою односторінкових інтерфейсів (SPA) та відокремленням клієнтської частини від серверної MVC втратив свою універсальність у бекенд-розробці. У таких умовах сервер перестав відповідати за генерацію подань, а фокус зосередився на API, обробці даних, реалізації бізнес-процесів і інтеграції з базами даних. Через це класичний MVC поступився місцем більш гнучким і спеціалізованим архітектурним моделям, які дозволяють будувати модульні, розширювані й високонадійні серверні системи.

Фреймворк Nest.js пропонує модульну архітектуру, яка стала основою

реалізації серверної частини платформи [30]. На відміну від жорстких правил MVC, Nest.js дозволяє будувати систему як сукупність незалежних, ізольованих один від одного модулів, кожен з яких реалізує певний функціональний блок. Така архітектура нагадує структурні принципи великих корпоративних застосунків, оскільки забезпечує високу гнучкість та чіткий розподіл відповідальностей.

У межах Nest.js кожен модуль містить три основні елементи: контролери, сервіси та провайдери. Контролери відповідають за маршрутизацію HTTP-запитів та визначення API-інтерфейсів, сервіси реалізують бізнес-логіку, а провайдери виступають залежностями, що можуть інжектуватися в інші компоненти. Такий підхід значно спрощує інтеграцію між різними частинами системи, оскільки дозволяє легко розширювати функціонал без зміни існуючих компонентів.

Модульність Nest.js ідеально підходить для платформ, які містять багато доменних підсистем. У твоїй веб-платформі модуль Auth відповідає за автентифікацію й авторизацію, модуль Mood за обробку записів настрою, модуль Journal за текстові щоденники, модуль Recommendations за формування рекомендацій на основі TF-IDF та косинусної подібності, модуль Chat за взаємодію з чат-інтерфейсом, а модуль Content — за курси та вправи. Кожен з них функціонує автономно й може розвиватися незалежно від інших.

Суттєвою перевагою модульної архітектури Nest.js є слабка зв’язаність компонентів, яка досягається завдяки механізму Dependency Injection (DI). Сервіси, репозиторії або алгоритмічні компоненти можна легко підключити до будь-якого модуля, не створюючи нових копій або дублювання коду. Завдяки DI система стає не лише чистою з точки зору архітектури, але й простою в тестуванні, оскільки залежності можна замінювати на мок-об’єкти.

Модульність дозволяє платформі працювати як набір мікросервісів (хоча фізично це може бути один моноліт). У разі потреби кожен модуль можна винести в окремий сервіс — наприклад, рекомендаційна система або модуль аналітики можуть працювати як незалежні обчислювальні вузли, якщо у майбутньому система буде масштабуватися під більшу кількість користувачів.

Архітектура Nest.js органічно поєднується з принципами REST API,

забезпечуючи чіткий розподіл маршрутів та логічне розділення відповідальностей між контролерами та сервісами. У той час як MVC обмежується статичним набором правил, модульний підхід Nest.js дозволяє вибудувати архітектуру будь-якої складності, зберігаючи при цьому ясність, структурованість і передбачуваність.

Таким чином, під час розробки було використано саме модульний підхід Nest.js.. Це дозволило створити систему, яка не лише легко підтримується, але й може розширюватися, інтегрувати нові механізми рекомендацій, додаткові види аналітики та нові функціональні модулі без значних змін у загальній структурі.

Висновки до розділу 3

У третьому розділі розглянуто теоретичні та математичні основи, необхідні для побудови веб-платформі підтримки ментального здоров'я з використанням алгоритмів машинного навчання. Детально проаналізовано роботу TF-IDF для визначення значущості слів у текстах користувача та косинусну міру подібності, яка дозволяє оцінювати релевантність рекомендацій. Особливу увагу приділено методам NLP, що забезпечують інтерпретацію текстових описів настрою та формування векторних представлень.

Також описано архітектурні засади побудови REST API та принципи модульної структуризації системи, які забезпечують стабільність обробки запитів і можливість масштабування. Проведений аналіз підтвердив доцільність використання TF-IDF у поєднанні з косинусною подібністю для формування персоналізованих рекомендацій, а також довів ефективність застосування статистичних методів для задач аналізу текстових даних.

Таким чином, розділ 3 сформував математико-теоретичне підґрунтя рекомендаційної системи, визначив базові алгоритми та принципи обробки даних, необхідні для реалізації функціональних можливостей веб-платформи.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Архітектура веб-платформи підтримки ментального здоров'я побудована за принципами багаторівневого поділу відповідальностей, що забезпечує високу масштабованість, надійність та можливість подальшого розвитку системи без необхідності суттєвих змін у її структурі.

Програмна система складається з трьох основних логічних рівнів: клієнтської частини (Frontend), серверної частини (Backend) та рівня зберігання даних (Database). Комунікація між цими рівнями здійснюється виключно через стандартизований REST API, що забезпечує чітке розмежування функцій та незалежність між компонентами.

Клієнтська частина платформи реалізована з використанням фреймворку Next.js, який дозволяє поєднувати можливості серверного рендерингу зі зручністю сучасних односторінкових застосунків. Інтерфейс користувача розроблений таким чином, щоб забезпечити інтуїтивну взаємодію, швидке перемикання між модулями та комфортну роботу навіть на мобільних пристроях. Клієнт відповідає за відображення даних, збір інформації від користувача (настрій, текстові записи, взаємодія в чаті), а також за надсилання відповідних запитів до серверної частини.

Усі операції на фронтенді реалізовані в компонентному вигляді й використовують механізми керування станом та кешування (React Query), що зменшує кількість звернень до сервера та підвищує продуктивність.

Серверна частина системи побудована на основі Nest.js — фреймворку, який підтримує модульну архітектуру та принципи ефективної організації коду. Backend відповідає за всю логіку обробки даних: від авторизації користувача та роботи з профілями до обчислення рекомендацій за допомогою математичних алгоритмів TF-IDF і косинусної подібності.

Усі функції серверної частини організовані у вигляді окремих модулів (Auth, Users, Mood, Journal, Recommendations, Content, Chat), кожен з яких включає контролери, сервіси, DTO-структури та взаємодію з базою даних. Така розділена структура дозволяє легко оновлювати або розширювати будь-який модуль, не

порушуючи роботу всієї системи. Комунікація з клієнтом здійснюється через REST API, де кожен маршрут має чітко визначені методи й формати даних.

Рівень зберігання даних представлений реляційною базою PostgreSQL, яка обрана через свою надійність, підтримку складних зв'язків, високу продуктивність та відповідність вимогам проєкту. У базі зберігаються всі сутності платформи: користувачі, їхні профілі, записи настрою, текстові щоденники, рекомендації, навчальний контент та історія повідомлень чату. Взаємодія серверної частини з базою здійснюється за допомогою ORM Sequelize, що забезпечує об'єктно-реляційне відображення та значно спрощує роботу зі складними структурами даних. Наявність чітких зв'язків між таблицями дозволяє ефективно обробляти складні запити, які виникають, наприклад, у процесі формування рекомендацій або побудови аналітики настрою.

Загалом архітектура системи є логічно структурованою, масштабованою та гнучкою. Кожен з її рівнів виконує свою окрему функцію: клієнтська частина відповідає за взаємодію з користувачем, серверна — за обробку логіки та обчислення, база даних — за надійне зберігання інформації.

Такий підхід дозволяє платформі працювати стабільно навіть при збільшенні кількості користувачів або обсягу даних, а також створює можливості для подальшого розвитку — наприклад, додавання нових модулів рекомендацій, розширення функціоналу чату або впровадження алгоритмів машинного навчання вищого рівня.

4.1 Реалізація серверної частини

Серверна частина платформи реалізована на основі фреймворку Nest.js, який використовує модульно-орієнтовану архітектуру та принципи інверсії залежностей. Такий підхід забезпечує адаптивність, логічне розділення частин системи та простоту подальшого масштабування. Фізично структура проєкту організована у вигляді декількох основних каталогів, кожен з яких відповідає певній функціональній області.

У кореневому каталозі src містяться всі програмні компоненти бекенду. Кожен функціональний елемент згрупований у модулі — наприклад, auth, users, mood, journal, recommendations, content, chat (рис 4.1.).

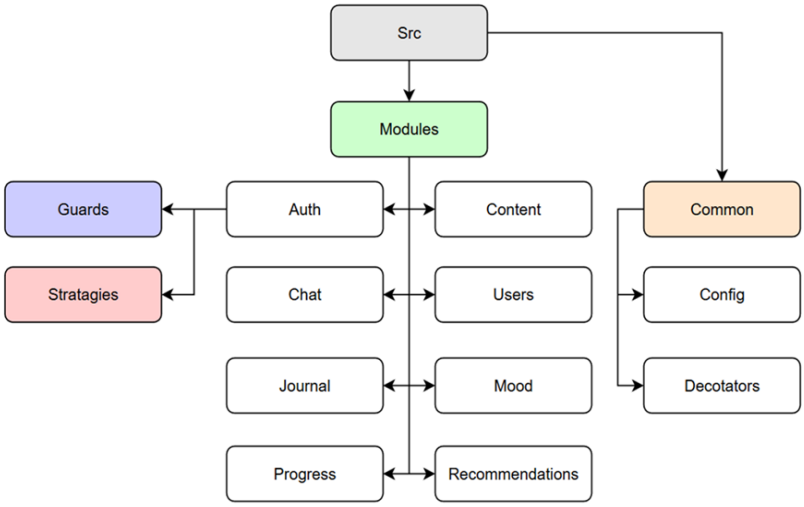

Рисунок 4.1 — Файлова структура проєкту.

Кожен модуль містить власні файли з контролерами, сервісами, DTO-об'єктами, а також логіку взаємодії з моделями бази даних. Такий розподіл робить систему структурованою: контролери відповідають за маршрутизацію та приймання HTTP-запитів, сервіси містять бізнес-логіку, а DTO гарантують коректність і типізованість переданих даних.

Механізм Dependency Injection, реалізований у Nest.js, дозволяє сервісам бути повторно використаними в межах різних модулів без дублювання коду. Наприклад, рекомендаційний сервіс або сервіс користувача можуть інжектуватися в інші модулі, якщо це необхідно для виконання перехресної логіки. Це зменшує зв'язність компонентів і підвищує гнучкість системи.

Взаємодія із базою даних організована через каталог database, який містить моделі Sequelize для всіх сутностей системи: користувачів, профілів, записів настрою, текстових щоденникових записів, рекомендацій, уроків, курсів і повідомлень. Такий підхід дозволяє централізовано керувати ORM-моделями, а модулі можуть імпортувати лише ті моделі, що потрібні для їхньої роботи.

Кореневий модуль app.module.ts об'єднує всі функціональні модулі в єдину систему, налаштовує підключення до бази даних, реєструє глобальні фільтри, middleware та конфігураційні параметри. У підсумку структура проєкту Nest.js забезпечує високу зрозумілість коду, мінімальну зв'язаність частин системи та можливість незалежного розвитку кожного модуля, що особливо важливо для такої великої платформи, як система підтримки ментального здоров'я.

4.1.1 Модуль авторизації

Модуль авторизації є одним із ключових компонентів серверної частини платформи, оскільки забезпечує управління доступом користувачів до персоналізованих даних та приватних функцій системи. Реалізація Auth-модуля в Nest.js побудована на основі JSON Web Token (JWT), що дозволяє організувати безстанну (stateless) автентифікацію та гарантує високу масштабованість без необхідності зберігати сесії на сервері (рис. 4.2).

```
@Post( path: 'login')  no usages  PixelWinner
@HttpCode(HttpStatus.OK)
@ApiOperation({ summary: 'Login user' })
@ApiResponse({
  status: 200,
  description: 'User successfully logged in',
  type: AuthResponseDto,
})
@ApiResponse({ status: 401, description: 'Invalid credentials' })
async login(@Body() loginDto: LoginDto): Promise<AuthResponseDto> {
  return this.authService.login(loginDto);
}
```

Рисунок 4.2 — Приклад методу авторизації з контроллеру «auth».

Основна логіка модуля зосереджена у контролері auth.controller.ts та сервісі auth.service.ts. Контролер відповідає за обробку маршрутів реєстрації (/auth/register), входу (/auth/login) та оновлення токена (/auth/refresh). Усі запити приймаються у вигляді типізованих DTO-об'єктів — наприклад, RegisterDto та LoginDto, де визначені необхідні поля (email, пароль) та правила їхньої валідації. Це забезпечує правильність структури даних ще до їх потрапляння в бізнес-логіку.

У сервісі реалізовано процес хешування пароля за допомогою бібліотеки bcrypt, що гарантує безпечне зберігання облікової інформації. Після успішної авторизації система генерує пару токенів: access-токен з коротким часом життя та refresh-токен тривалішої дії. Access-токен використовується для забезпечення доступу до захищених API-маршрутів, тоді як refresh-токен дозволяє оновлювати пару токенів без повторного введення логіна та пароля.

Для захисту приватних маршрутів використовується Guard JwtAuthGuard, який перехоплює запити та перевіряє дійсність токена. При успішній верифікації Guard додає до запиту інформацію про користувача, що дозволяє іншим модулям працювати з його ідентифікатором та профілем.

4.1.2 Модуль користувача

Модуль користувача є основним компонентом серверної частини платформи, відповідальним за роботу з персональними даними, профілями та сервісною інформацією користувачів. Оскільки система підтримки ментального здоров'я взаємодіє з великою кількістю індивідуальних даних — від особистих налаштувань до історії активності — Users Module виступає центральною точкою доступу до даних користувача та забезпечує їх коректну обробку (рис. 4.3).

```
import { UsersController } from './users.controller';
import { UsersService } from './users.service';
import { AuthModule } from '../auth/auth.module';

@Module({  Show usages   PixelWinner
  imports: [AuthModule],
  controllers: [UsersController],
  providers: [UsersService],
  exports: [UsersService],
})
export class UsersModule {}
```

Рисунок 4.3 — Реалізація модулю «Users».

Модуль складається з контролера (users.controller.ts), сервісу (users.service.ts) та набору DTO-класів, які визначають структуру вхідних і вихідних даних.

Контролер приймає запити на отримання профілю, оновлення інформації або зміну параметрів користувача, передає їх у сервісний шар і повертає результат у стандартизованому форматі JSON. Завдяки використанню декораторів Nest.js (@Get, @Patch, @UseGuards) кожен маршрут має чітко визначений доступ та логіку виконання.

Основна частина бізнес-логіки розміщена у сервісі користувачів. Він відповідає за взаємодію з моделями бази даних — User та Profile, які зберігаються через ORM Sequelize. Під час обробки запиту сервіс може виконувати перевірки наявності користувача, оновлювати профіль, змінювати параметри (ім'я, аватар, часовий пояс, мову інтерфейсу), а також надавати дані, необхідні іншим модулям, зокрема рекомендаційній системі або модулю аналітики настрою.

Усі маршрути Users Module захищені JWT Guard, що гарантує доступ до профільних даних лише автентифікованим користувачам (рис. 4.4).

```
@ApiTags( ...tags: 'Users') Show usages  PixelWinner
@Controller( prefix: 'users')
@UseGuards(JwtAuthGuard)
@ApiBearerAuth()
export class UsersController {
  constructor(private readonly usersService: UsersService) {} no usages  PixelWinner

  @Get( path: 'me') no usages  PixelWinner
  @ApiOperation({ summary: 'Get current user profile' })
  @ApiResponse({
    status: 200,
    description: 'Current user profile',
    type: UserResponseDto,
  })
  @ApiResponse({ status: 401, description: 'Unauthorized' })
  async getMe(@GetUser() user: User): Promise<UserResponseDto> {
    return this.usersService.getMe(user.id);
  }
```

Рисунок 4.4 — Використання JWT Guard для захисту маршруту «me».

Це важливо з точки зору інформаційної безпеки, оскільки профіль може містити конфіденційні дані про психологічний стан або історію взаємодій із платформою.

Модуль користувача також забезпечує важливий функціонал, пов'язаний з інтеграцією між іншими частинами системи. Наприклад, модуль рекомендацій

отримує дані про активність користувача для точнішого формування рекомендацій, а модуль аналітики використовує інформацію профілю для групування та узагальнення даних настрою. Усі ці взаємодії відбуваються завдяки чітко структурованій сервісній логіці Users Module.

4.1.3 Модуль настрою

Модуль настрою є одним із центральних функціональних модулів серверної частини платформи, оскільки саме він забезпечує реєстрацію психоемоційних станів користувачів і формує базову інформацію, на основі якої працюють аналітичні та рекомендаційні механізми системи. Mood Module реалізує CRUD-операції для записів настрою, обробляє часові дані, надає агреговану статистику та взаємодіє з кількома іншими модулями, зокрема аналітикою та рекомендаціями (рис 4.5).

```
@Post()  no usages  PixelWinner
@HttpCode(HttpStatus.CREATED)
@ApiOperation({ summary: 'Create a new mood entry' })
@ApiResponse({
  status: 201,
  description: 'Mood entry created successfully',
  type: MoodResponseDto,
})
@ApiResponse({ status: 400, description: 'Invalid input' })
@ApiResponse({ status: 401, description: 'Unauthorized' })
async create(
  @GetUser() user: User,
  @Body() createMoodDto: CreateMoodDto,
): Promise<MoodResponseDto> {
  return this.moodService.create(user.id, createMoodDto);
}
```

Рисунок 4.5 — Приклад реалізації CRUD-операції для записів настрою.

Структурно модуль складається з mood.controller.ts, mood.service.ts та набору DTO-класів (CreateMoodDto, FilterMoodDto). Контролер визначає маршрути для створення, отримання та фільтрації записів настрою. Усі маршрути захищені механізмом JWT Guard, що гарантує доступ лише зареєстрованим користувачам. Запити проходять валідацію за допомогою декораторів class-validator, що

забезпечує коректність переданих даних (наприклад, допустимі значення рівня настрою, формат дати тощо).

Основна логіка модуля реалізована у сервісі MoodService. Сервіс відповідає за взаємодію з ORM-моделлю MoodEntry, яка зберігає такі дані, як оцінка настрою, дата запису, емоційні теги та коментарі. При створенні нового запису сервіс використовує ідентифікатор користувача, отриманий із JWT-токена, що забезпечує прив'язку всіх записів до конкретного профілю.

Окрім збереження даних, модуль настрою надає функції аналітики. Сервіс може виконувати фільтрацію записів за часовими діапазонами, що дозволяє будувати графіки змін настрою на фронтенді та визначати психологічні тенденції. Ці можливості інтегруються з клієнтською частиною через Chart.js для створення наочних візуалізацій.

Важливою частиною модуля є передача інформації рекомендаційному модулю. Дані про настрій користувача, зокрема різкі зміни, низькі показники або тривалі періоди погіршення, можуть впливати на процес формування персоналізованих рекомендацій. Наприклад, при низькому середньому значенні настрою система може пропонувати вправи для зниження тривоги або матеріали з емоційної саморегуляції.

4.1.4 Модуль щоденника

Модуль щоденника є важливим функціональним компонентом веб-платформи, який забезпечує можливість користувачам зберігати текстові записи, пов'язані з їхнім психоемоційним станом, подіями дня або особистими рефлексіями. На відміну від модуля настрою, де фіксуються короткі числові оцінки, щоденникові записи містять повноцінний текст, що дозволяє глибше аналізувати емоційний контекст користувача та використовувати ці дані в системі персоналізованих рекомендацій.

Архітектурно Journal Module складається з контролера (journal.controller.ts), сервісу (journal.service.ts) і набору DTO-класів, які забезпечують структуру й валідацію даних. Контролер обробляє запити для створення нових записів, отримання списку записів, фільтрації за датами та плагінації. Усі маршрути

захищені JWT Guard, що обмежує доступ лише для автентифікованих користувачів, оскільки записи щоденника містять персональні та конфіденційні текстові дані.

Основна бізнес-логіка реалізована у JournalService (рис 4.6).

```
@Injectable()  Show usages  PixelWinner
export class JournalService {
  private readonly logger : Logger  = new Logger(JournalService.name);

  constructor(private readonly sentimentService: SentimentService) {}  no usages  PixelWinner

  /**
   * Create a new journal entry with sentiment analysis
   */
  async create(  Show usages  PixelWinner
    userId: string,
    createJournalDto: CreateJournalDto,
  ): Promise<JournalResponseDto> {
    const { title, content, tags = [], isPrivate = false } = createJournalDto;

    // Analyze sentiment of content
    const sentimentResult : SentimentResult  = this.sentimentService.analyzeSentiment(content);

    // Create journal entry
    const entry : JournalEntry  = await JournalEntry.create({
      userId,
      title,
      content,
      tags,
      isPrivate,
      context: {
        sentiment: sentimentResult.sentiment,
        score: sentimentResult.score,
      },
    });

    this.logger.log(
      message: `Journal entry created: ${entry.id} by user ${userId}, sentiment: ${sentimentResult.sentiment}`,
    );

    return this.mapToResponse(entry);
  }
```

Рисунок 4.6 — Метод створення запису у щоденник.

Сервіс виконує взаємодію з моделлю JournalEntry, яка зберігає текст запису, дату створення, позначки користувача та прив'язку до конкретного профілю. Збереження даних відбувається через ORM Sequelize, а використання DTO забезпечує коректність структури введеного тексту та додаткових параметрів. Сервіс підтримує фільтрацію за інтервалами дат, що дозволяє користувачам переглядати записи за певний період або аналізувати динаміку змін їхнього психоемоційного стану.

Дані зі щоденника також застосовуються в рекомендаційному модулі. Під час обчислення TF-IDF і cosine similarity текстові записи користувача включаються у корпус документів, що значно підвищує точність підбору рекомендацій. Наприклад, якщо користувач часто згадує у записах слова, пов'язані зі стресом, вигоранням або тривогою, система може пропонувати вправи та статті, релевантні саме цим темам.

4.1.5 Модуль рекомендацій

Модуль рекомендацій забезпечує формування персоналізованих психологічних порад на основі текстових даних користувача та описів навчальних матеріалів. У межах Nest.js він реалізований як окремий модуль, що включає контролер, сервісну логіку, допоміжний TF-IDF-сервіс та моделі збереження рекомендацій і зворотного зв'язку (рис 4.7).

```
export class TfIdfService {
  private readonly logger : Logger  = new Logger(TfIdfService.name);

  /**
   * Tokenize text into words (lowercase, filter short words)
   */
  private tokenize(text: string): string[] {  Show usages  PixelWinner
    return text
      .toLowerCase()
      .replace( searchValue: /[^\wа-яіїєґ\s]/gi,  replaceValue: ' ')
      .split( splitter: /\s+/)
      .filter((word : string ) : boolean  => word.length > 2);
  }

  /**
   * Calculate term frequency for a document
   */
  private calculateTf(tokens: string[]): Map<string, number> {  Show usages  PixelWinner
    const tf = new Map<string, number>();
    const total : number  = tokens.length;

    for (const token of tokens) {
      tf.set(token, (tf.get(token) || 0) + 1);
    }

    // Normalize by total tokens
    for (const [term, count] of tf.entries()) {
      tf.set(term, count / total);
    }

    return tf;
  }
```

Рисунок 4.7 — Реалізація методів допоміжного сервісу TF-IDF.

Цей модуль складається з контролера recommendations.controller.ts, який обробляє запити користувача на отримання рекомендацій та передачу оцінок ефективності запропонованих матеріалів. Всі маршрути захищені JWT Guard, оскільки рекомендації формуються на основі персональних записів настрою та щоденника. Для обміну даними використовуються спеціалізовані DTO, які забезпечують правильність структури введених текстів і числових параметрів.

У модулі ключову роль відіграє сервіс RecommendationsService, який взаємодіє з контентними матеріалами, записами настрою, журналом і модельними структурами бази даних. Під час обробки запиту сервіс отримує історію останніх текстових записів користувача — щоденникових нотаток та описів настрою — і перетворює їх у векторні представлення за допомогою TF-IDF. Для цього використовується допоміжний компонент TfidfService, який виконує токенізацію, нормалізацію тексту, обчислення ваг термів і формування векторів документів. Математична обробка здійснюється на основі підходу, описаного в теоретичному розділі, включаючи побудову термінового словника та визначення інформаційної важливості кожного терму.

Після формування TF-IDF-векторів сервіс виконує порівняння текстів користувача з корпусом навчальних матеріалів (уроків, статей, вправ) за допомогою міри косинусної подібності. На основі отриманих значень формується рейтинговий список документів, відсортований від найбільш релевантних. Для кожного матеріалу модуль обчислює індивідуальний бал релевантності та повертає користувачу перелік рекомендацій у структурованому форматі.

Усі виконані розрахунки можуть зберігатись у таблиці PredictionLog, що дозволяє системі накопичувати інформацію про точність рекомендацій та аналізувати їх ефективність у майбутньому. Крім цього, користувач може залишати фідбек для кожної рекомендації — позитивний або негативний. Ці дані зберігаються у моделі RecommendationFeedback та використовуються для подальшого вдосконалення рекомендаційного алгоритму, наприклад, шляхом корекції ваг або винятку матеріалів, які не принесли користувачу користі.

Модуль рекомендацій також взаємодіє з модулем настрою: якщо система

виявляє значні коливання настрою або довготривале погіршення, вона може підбирати матеріали певного типу, які історично показували ефективність для користувача або подібних профілів. Хоча в поточній версії реалізовано лише контентно-орієнтований підхід, структура модуля дозволяє легко інтегрувати більш складні моделі NLP або машинного навчання у майбутніх розширеннях.

У підсумку модуль рекомендацій забезпечує персоналізовану обґрунтовану взаємодію з користувачем, поєднуючи алгоритмічні методи з текстовими даними та забезпечуючи індивідуальне вдосконалення психологічного стану через релевантний контент.

4.1.6 Модуль навчального контенту

Модуль навчального контенту відповідає за управління освітніми матеріалами, що використовуються платформою для підтримки ментального здоров'я користувачів. Контент включає курси, уроки та вправи, які формують структурований навчальний процес, орієнтований на розвиток навичок саморегуляції, емоційної стійкості та усвідомленості. Модуль також забезпечує можливість відстеження прогресу користувача та інтегрується з рекомендаційною системою, яка може пропонувати певні уроки або вправи відповідно до емоційного стану.

Основними компонентами Content Module є контролери (courses.controller.ts, lessons.controller.ts, exercises.controller.ts) та відповідні сервіси (courses.service.ts, lessons.service.ts, exercises.service.ts) (рис 4.8).

```
@Module({  Show usages  PixelWinner
  imports: [AuthModule, ProgressModule],
  controllers: [ContentController, CourseController],
  providers: [ContentService, CourseService],
  exports: [ContentService, CourseService],
})
export class ContentModule {}
```

Рисунок 4.8 — Реалізація модулю навчального контенту, з використанням інших модулів та сервісів, контролерів.

Контролери відповідають за обробку запитів на отримання списків курсів, перегляд конкретних уроків або вправ, а також за оновлення даних адміністраторами платформи. Сервіси містять бізнес-логіку для завантаження навчального контенту з бази даних та його структуризації перед передачею клієнту.

У модулі використовуються ORM-моделі Course, Lesson і Exercise, що описують структуру навчальних матеріалів. Курс є найвищою сутністю та включає набір уроків, кожен з яких може містити одну або декілька вправ. Така ієрархічна структура дозволяє формувати цілісні навчальні блоки, які можуть бути орієнтовані на різні психологічні потреби, зокрема боротьбу з тривогою, розвиток усвідомленості або покращення емоційного інтелекту.

Важливою частиною модуля є підтримка прогресу користувача, що реалізована через моделі UserCourse та UserExercise. Під час взаємодії користувача з контентом сервіс може реєструвати переглянуті уроки чи виконані вправи. Це дозволяє системі зберігати інформацію про ступінь проходження матеріалів, а також автоматично пропонувати наступні уроки. Завдяки цьому навчальні модулі можуть виступати як базою для саморозвитку, так і частиною рекомендаційної логіки, що адаптується під стан користувача.

Content Module також інтегрується з рекомендаційним модулем. Описи уроків та вправ входять до корпусу документів, що використовуються при обробці TF-IDF. Це дозволяє системі підбирати найбільш релевантні навчальні матеріали відповідно до текстових записів користувача у щоденнику або даних із модуля настрою. Наприклад, якщо у записах часто зустрічаються теми перевтоми чи тривоги, алгоритм може рекомендувати вправи зі зниження стресу або уроки з технік глибокого дихання.

4.1.7 Модуль чату

Модуль чату забезпечує інтерактивну взаємодію користувача з системою в режимі реального часу, виступаючи зручним інструментом комунікації між користувачем та платформою. Основна його мета полягає у створенні можливості ставити запитання, отримувати оперативні поради та вести короткі текстові діалоги, що доповнюють роботу інших компонентів — зокрема, щоденника та

рекомендаційного модуля. Така функціональність дозволяє формувати більш цілісний досвід використання платформи та підсилює її практичну цінність.

На відміну від класичних чат-ботів, орієнтованих переважно на інформаційні або сервісні запити, модуль чату у платформі спрямований на підтримку психоемоційного стану користувача. Завдяки цьому передбачається можливість його подальшого розширення алгоритмами NLP, які зможуть аналізувати зміст повідомлень та надавати більш чутливі, персоналізовані й контекстно релевантні відповіді.

Архітектурно модуль складається з chat.controller.ts, chat.service.ts та моделі Message, яка описує формат і структуру збережених повідомлень. Контролер опрацьовує маршрути надсилання повідомлень, отримання історії діалогу та взаємодії з відповідями системи. Усі маршрути є приватними й захищені JWT Guard, що гарантує доступ до історії чату лише автентифікованим користувачам.

Основна логіка обробки повідомлень реалізована у ChatService. Під час надсилання повідомлення сервіс створює новий запис у базі даних, включаючи текст повідомлення, тип відправника (користувач чи система), часову мітку та ідентифікатор користувача (рис 4.9).

Модель Message представляє ці дані у структурованій формі та дозволяє легко запитувати історію діалогів. Збереження повідомлень дає змогу не лише переглядати контекст попередніх взаємодій, а й використовувати вміст чату для побудови майбутніх моделей персоналізованих відповідей.

Модуль також містить логіку формування відповідей від системи. У поточній реалізації відповіді формуються на основі узагальнених шаблонів або інтеграції з допоміжним сервісом, який може генерувати текстові реакції на повідомлення користувача. Поведінка модуля легко розширюється — у майбутньому можливе підключення моделей NLP, що дозволять аналізувати емоційний тон, зміст повідомлень і пропонувати релевантні поради або вправи.

Модуль інтегрується з іншими частинами платформи. Наприклад, записані повідомлення можуть використовуватися у рекомендаційній системі як додаткове джерело текстової інформації.

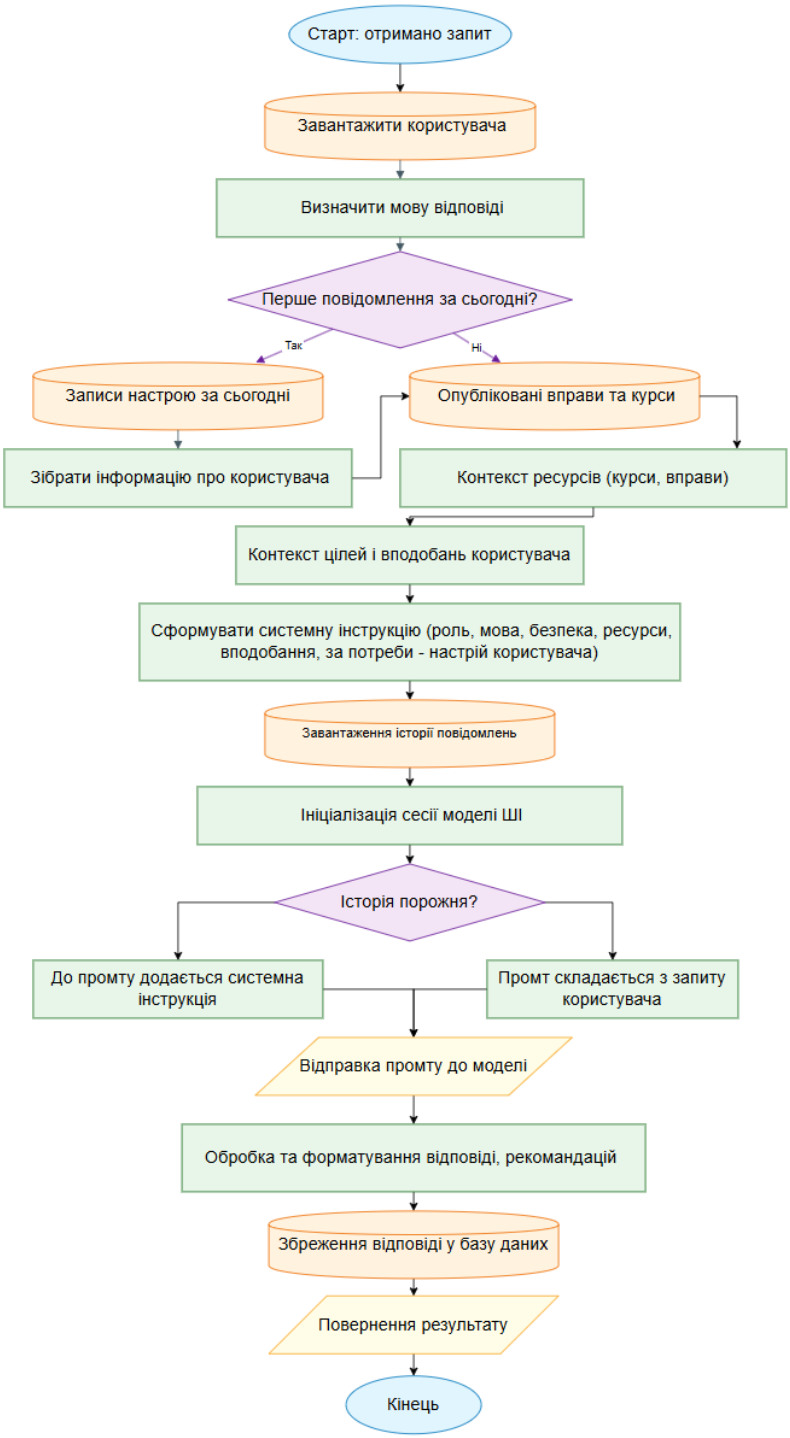

Рисунок 4.9 — Алгоритм роботи модулю «Chat»

Якщо користувач часто згадує у чаті певні емоційні тригери, аналітичні модулі можуть ураховувати це при підборі рекомендацій. Також чат дозволяє користувачу звертатися по швидку підтримку, не відкриваючи щоденник чи модуль настрою.

4.2 Реалізація клієнтської частини (Frontend, Next.js)

Клієнтська частина веб-платформи реалізована з використанням фреймворку Next.js, який поєднує можливості серверного рендерингу та сучасної компонентної архітектури React. Такий підхід забезпечує високу продуктивність, оптимізацію часу завантаження сторінок і створення інтерфейсу, який залишається стабільним і швидким навіть при активній взаємодії з даними. Основним призначенням клієнтської частини є забезпечення зручної інтерактивної взаємодії користувача з платформою, включно з введенням текстових записів, переглядом статистики настрою, отриманням рекомендацій та роботою з навчальними матеріалами.

Архітектура фронтенду побудована на основі App Router, що дозволяє організовувати маршрути за принципом файлової структури. У проєкті виділено дві основні групи маршрутів: публічні й захищені (рис 4.10).

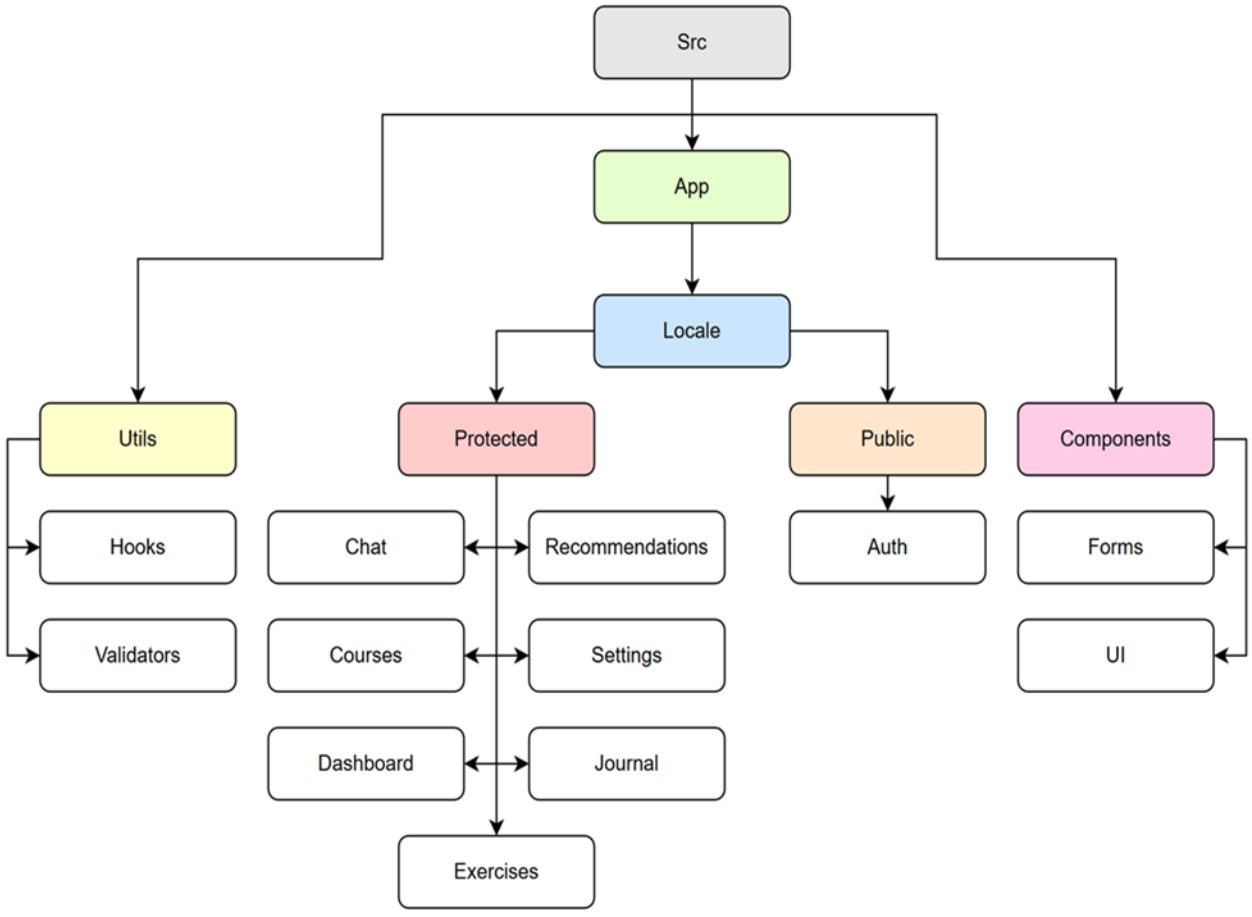

Рисунок 4.10 — Структура веб-додатку

Публічні маршрути охоплюють сторінки входу, реєстрації та онбордингу, тоді як доступ до захищених забезпечується через middleware, яке перевіряє наявність валідного JWT-токена. Це гарантує безпеку користувацьких даних та контроль доступу до персоналізованих функцій платформи.

Для побудови інтерфейсу використовується компонентна модель React, яка дозволяє створювати вкладені, повторно використовувані елементи: форми введення, модальні вікна, графіки, списки рекомендацій і блоки чату. Стан даних та взаємодію з серверною частиною реалізовано за допомогою бібліотеки TanStack Query (React Query). Вона відповідає за кешування запитів, автоматичне оновлення даних при їх зміні на сервері, оптимістичні оновлення та повторні спроби у разі помилок. Це дозволяє мінімізувати кількість мережевих запитів і підвищує продуктивність інтерфейсу. Наприклад, при створенні запису настрою дані миттєво оновлюються у локальному кеші, що робить інтерфейс більш плавним і швидким.

Для валідації форм використовується бібліотека Zod, яка дозволяє точно визначити структуру даних та гарантувати правильність введеної інформації перед відправкою на сервер. Кожна форма — реєстрації, створення запису настрою, додавання щоденникової нотатки або відгуку на рекомендацію — має власну схему валідації, що забезпечує захист від некоректних даних та покращує загальну надійність системи.

Стилізація інтерфейсу виконується з використанням Tailwind CSS, що дозволяє створювати адаптивні компоненти без необхідності використання зовнішніх CSS-файлів. Завдяки цьому інтерфейс платформи є мінімалістичним, інтуїтивним і легко масштабованим. Tailwind також полегшує розробку «responsive»-дизайну, що дозволяє ефективно працювати з платформою на мобільних пристроях.

Для відображення статистики та графіків настрою використовується бібліотека Chart.js, інтегрована через React-адаптери. Модуль аналітики дозволяє користувачам переглядати динаміку їхнього емоційного стану у вигляді лінійних графіків, що генеруються на основі даних, отриманих із серверної частини через

REST API. Це забезпечує наочність і дозволяє користувачу краще розуміти тенденції свого настрою.

Структура клієнтської частини також включає модулі для відображення рекомендацій, перегляду уроків та вправ, а також роботи з чат-інтерфейсом. Кожен з цих елементів реалізований як окрема сторінка або компонент, що взаємодіє з API та отримує дані у форматі JSON. Завдяки цьому платформа дотримується принципів модульності й забезпечує простоту масштабування — у разі потреби можна без труднощів додавати нові функціональні блоки.

4.2.1 Структура App Router

Архітектура App Router забезпечує файлово-орієнтовану побудову маршрутизації та чіткий поділ інтерфейсу на публічні і захищені області. Структура директорій у каталозі app безпосередньо визначає маршрути, які доступні у веб-застосунку, що спрощує навігацію, підтримку та розширення клієнтської частини.

Базова організація маршрутизації побудована навколо динамічного сегмента [locale], який використовується для підтримки мультимовності інтерфейсу. Усі сторінки додатку розміщені в дереві app/[locale]/, де [locale] відповідає коду мови (наприклад, en, uk). Це дозволяє Next.js автоматично формувати маршрути виду /en/mood або /uk/journal, а також інтегрувати локалізацію контенту через бібліотеку локалізації.

Всередині app/[locale]/ реалізовано логічний поділ на дві основні групи маршрутів — публічні та захищені. Публічні сторінки (група public) включають маршрути авторизації та реєстрації, наприклад auth/login, auth/register, а також початкові сторінки, які не потребують автентифікації. Захищена група (protected) містить основний функціонал платформи: панель користувача (dashboard), модулі настрою (mood, mood/analytics), щоденник (journal), рекомендації (recommendations), чат (chat) і навчальний контент (courses, exercises). Кожен із цих маршрутів визначається через окремі папки з файлами page.tsx, що відповідають за рендеринг сторінки, та, за потреби, layout.tsx, які задають спільну структуру для групи підсторінок.

Доступ до захищених маршрутів контролюється за допомогою middleware, яке розташоване на рівні кореня застосунку (middleware.ts). Воно перехоплює запити до /(protected)-маршрутів, перевіряє наявність і валідність токена авторизації (JWT) та, у разі відсутності або помилки, перенаправляє користувача на сторінку входу. Такий підхід дозволяє централізовано реалізувати логіку захисту, не дублюючи перевірки в кожному окремому компоненті.

App Router також забезпечує можливість використання layout-компонентів, які визначають загальний каркас для різних областей застосунку. Для захищеної частини платформи використовується спільний layout із навігаційним меню, заголовками, блоками аналітики та основною робочою зоною, тоді як публічні сторінки мають спрощений макет з формами входу або реєстрації. Така багаторівнева структура дозволяє легко підтримувати єдиний стиль інтерфейсу та уникати дублювання верстки.

4.2.2 Інтерфейс користувача

Інтерфейс користувача веб-платформи розроблений у відповідності до сучасних принципів UX/UI, що передбачають простоту взаємодії, інтуїтивність навігації та доступність функцій незалежно від типу пристрою. Використання фреймворку Next.js дозволило поєднати високопродуктивний рендеринг із компонентною архітектурою React, створивши масштабований і логічно структурований інтерфейс, що відповідає потребам користувачів, які працюють з особистими психологічними даними.

Інтерфейс поділяється на дві основні зони: публічну та захищену. Публічна зона містить сторінки авторизації, реєстрації та першого налаштування профілю. Вони виконані у мінімалістичному стилі та містять форми введення, валідовані за допомогою React Hook Form та Zod, що дозволяє забезпечити коректність даних ще на рівні інтерфейсу. Після входу користувач отримує доступ до головної частини платформи — dashboard, який виступає центральною точкою навігації між модулями.

Захищена частина інтерфейсу включає декілька ключових модулів. Сторінка Mood дозволяє користувачеві швидко додавати записи настрою через компактну

форму з вибором рівня настрою, емоційних тегів та коментаря. Візуалізація історії настрою реалізована за допомогою графіків Chart.js, що дозволяє переглядати динаміку емоційного стану у зручній формі.

Модуль Journal надає інтерфейс для створення та перегляду текстових щоденникових записів. Записи відображаються у вигляді списку із зазначенням дати, а детальний перегляд дозволяє відкривати повний текст. Форма створення нового запису підтримує валідацію та мінімалістичний дизайн для комфортного набору тексту.

Сторінка Recommendations містить перелік персоналізованих порад, сформованих алгоритмом на основі поведінки та текстових даних користувача. Кожна рекомендація представлена карткою з коротким описом і можливістю залишити зворотний зв'язок. Інтерфейс підтримує динамічне оновлення списку рекомендацій через React Query.

Модуль Chat реалізує діалогову взаємодію користувача із системою. Інтерфейс побудований за аналогією з месенджерами — історія повідомлень відображається у вертикальному потоці, нові повідомлення надсилаються через компактну форму введення тексту. Система автоматично підвантажує відповіді сервера та синхронізує історію через API.

Модуль Content представлений сторінками курсів, уроків і вправ. Інтерфейс дозволяє переглядати доступні курси, відкривати уроки, виконувати вправи та слідкувати за прогресом. Вся інформація відображається у вигляді карток або інтерактивних блоків, забезпечуючи зручність навігації та використання.

Усі компоненти інтерфейсу мають адаптивний дизайн завдяки Tailwind CSS, що гарантує коректне відображення на мобільних пристроях. Крім того, використання компонентів Next.js Server/Client Components дозволило оптимізувати час завантаження сторінок та покращити загальний користувацький досвід.

4.2.3 Використання React Query

У клієнтській частині веб-платформи важливу роль відіграє механізм взаємодії з API та управління станом отриманих даних. Для цього в проєкті

використовується бібліотека React Query (TanStack Query), яка забезпечує ефективне кешування, синхронізацію, оновлення та повторне використання даних, отриманих від серверної частини. Застосування React Query значно спрощує логіку взаємодії з REST API та мінімізує кількість дубльованих запитів, що особливо важливо для системи з великою кількістю користувацьких дій та частими зверненнями до серверу.

Основна концепція React Query — автоматичне керування станом даних, які отримуються через асинхронні запити. Коли користувач переходить між сторінками або виконує дії, що потребують отримання інформації (наприклад, завантаження записів настрою або рекомендацій), бібліотека перевіряє, чи існують ці дані в кеші. Якщо вони актуальні, React Query повертає кешовану версію без повторного звернення до серверу; якщо ж дані застарілі — автоматично виконується їх оновлення. Це дозволяє забезпечити високу продуктивність інтерфейсу та швидку реакцію на дії користувачів.

У платформі активно використовуються query-хуки (useQuery) та mutation-хуки (useMutation) (рис 4.11).

```
// Send feedback for recommendation
export function useSendFeedback() : UseMutationResult<void, Error, { recommen... { Show usages  PixelWinner
  const queryClient : QueryClient = useQueryClient();

  return useMutation<
    void,
    Error,
    { recommendationId: string; data: RecommendationFeedbackInput }
  >({
    mutationFn: async ({ recommendationId, data } : { recommendationId: string; data: { actio... } : Promise<void> => {
      await api.post( url: `/recommendations/${recommendationId}/feedback`, data);
    },
    onSuccess: () : void => {
      // Invalidate recommendations to refetch with updated feedback
      queryClient.invalidateQueries({ queryKey: recommendationKeys.lists() });
    },
    onError: (error : Error ) : never => {
      throw handleApiError(error);
    },
  });
}
```

Рисунок 4.11 — Приклад реалізації хуку useMutation.

Запити useQuery застосовуються для отримання списків записів настрою,

даних щоденника, навчальних матеріалів, рекомендацій та іншої інформації, що не змінюється миттєво. Наприклад, при відкритті сторінки настрою React Query автоматично завантажує останні записи та оновлює їх фоні, забезпечуючи актуальність даних без втручання користувача.

Операції, що змінюють дані — створення нового запису, відправлення повідомлення в чат, додавання щоденникової нотатки або надсилання фідбеку до рекомендації — реалізовані через useMutation. Ці мутації можуть автоматично оновлювати кеш після успішного запиту. Наприклад, після створення нового запису настрою React Query автоматично викликає повторний запит до /mood для оновлення списку записів, забезпечуючи синхронність UI з серверними даними.

Ще однією важливою перевагою використання React Query є підтримка обробки помилок, повторних спроб, фонового рефетчингу та паралельних запитів. Для платформи це має особливе значення, оскільки стабільність взаємодії користувача з системою впливає на якість роботи сервісу, особливо у випадках слабкого інтернет-з'єднання або високого навантаження.

4.2.4 Валідація форм

Для забезпечення коректності даних, що надсилаються користувачем із клієнтської частини платформи, у проєкті використовується бібліотека Zod — сучасний інструмент для схематичної валідації даних у TypeScript- та React-застосунках. Zod дозволяє створювати типізовані схеми, які одночасно працюють як валідатори форм і як джерело типів для всієї системи (рис 4.12).

```
// Mood entry input schema
export const moodInputSchema : ZodObject<{ score: ZodNumber; tags: ZodOp... = z.object({
  score: z
    .number()
    .min( value: 1, params: "mood.errors.scoreMin")
    .max( value: 10, params: "mood.errors.scoreMax")
    .int(),
  tags: z.array(z.string()).optional(),
  note: z.string().max( maxLength: 2000, params: "mood.errors.noteMax").optional(),
});

export type MoodInput = z.infer<typeof moodInputSchema>;
```

Рисунок 4.12 — Реалізація схеми валідації та типу.

Такий підхід забезпечує єдність структур даних між фронтендом і бекендом, зменшує ймовірність помилок і підвищує стабільність взаємодії з API. У платформі форми активно використовуються в модулях авторизації, запису настрою, щоденника, чату та рекомендацій. Кожна форма супроводжується Zod-схемою, яка визначає структуру вхідних даних, їхні обмеження, допустимі значення та правила валідації. Наприклад, у формі авторизації схема перевіряє, що email має правильний формат, а пароль містить достатню кількість символів. У модулі настрою Zod контролює правильність переданої числової оцінки та формат текстового коментаря.

Інтеграція Zod із React Hook Form дозволяє виконувати валідацію безпосередньо під час введення даних, забезпечуючи користувачу миттєвий зворотний зв'язок. Помилки відображаються в інтерфейсі у вигляді текстових підказок або змін стилізації компонентів, що значно покращує зручність використання системи. Крім того, Zod-схеми можуть повторно використовуватися як типи даних, що передаються до API, що зменшує ризик розбіжностей між фронтендом і бекендом.

Ще однією перевагою Zod є можливість формувати вкладені й складні структури даних, що є особливо корисним для модулів з великою кількістю полів, наприклад, під час створення щоденникового запису або відправлення повідомлення в чаті. Завдяки цьому всі дані, що надходять у запитах до серверної частини, проходять чітку схематичну перевірку перед відправленням, що мінімізує можливість помилок API.

4.2.5 Стилізація Tailwind CSS

Для реалізації зовнішнього вигляду та адаптивної верстки клієнтської частини веб-платформи використовується фреймворк Tailwind CSS, який ґрунтується на підході utility-first. На відміну від класичних CSS-фреймворків, де основний акцент робиться на готових компонентах, Tailwind надає велику кількість дрібних, добре продуманих утилітарних класів, з яких конструюються інтерфейсні елементи (рис 4.13).

```
theme: {
    extend: {
        colors: {
            background: "hsl(var(--background))",
            foreground: "hsl(var(--foreground))",
            card: {
                DEFAULT: "hsl(var(--card))",
                foreground: "hsl(var(--card-foreground))",
            },
            popover: {
                DEFAULT: "hsl(var(--popover))",
                foreground: "hsl(var(--popover-foreground))",
            },
            primary: {
                DEFAULT: "hsl(var(--primary))",
                foreground: "hsl(var(--primary-foreground))",
            },
```

Рисунок 4.13 — Опис стилів з використанням Tailwind CSS.

Такий підхід дозволяє створювати гнучкий дизайн, не перевантажуючи проєкт кастомними таблицями стилів та уникаючи дублювання CSS-коду.

У межах платформи Tailwind CSS використовується для стилізації всіх основних екранів: авторизації, реєстрації, dashboard, сторінок настрою, щоденника, рекомендацій, чату та навчального контенту. Кожен інтерфейсний блок — форма, панель, картка рекомендацій, елемент списку або графік — описується набором Tailwind-класів безпосередньо у JSX-розмітці компонентів. Це дозволяє одночасно бачити структуру елемента й правила його відображення, що підвищує читабельність і спрощує підтримку коду.

Важливою перевагою Tailwind є вбудована підтримка адаптивної верстки. За допомогою breakpoint-класів (sm:, md:, lg: тощо) інтерфейс автоматично підлаштовується під різні розміри екранів. Для платформи підтримки ментального здоров'я це особливо актуально, оскільки користувачі можуть взаємодіяти з системою з мобільних пристроїв, планшетів та настільних комп'ютерів. Наприклад, блоки аналітики настрою й рекомендацій будуються як сітка, що на малих екранах перетворюється на вертикальний список, а на великих — відображається в кілька колонок.

Tailwind також використовується для формування єдиної дизайн-системи:

через конфігураційний файл задаються базові кольори, типографіка, відступи та радіуси заокруглення. Це забезпечує стилістичну цілісність усіх екранів платформи, а зміна глобальних налаштувань дозволяє швидко адаптувати візуальний стиль без редагування кожного компонента окремо. Такий підхід зручний у контексті довгострокової підтримки проєкту та можливого ребрендингу.

З технічної точки зору Tailwind позитивно впливає на продуктивність: завдяки механізму tree-shaking у фінальну збірку потрапляють лише ті класи, які реально використовуються в компонентному коді. Це дозволяє зменшити розмір CSS-файлів і прискорити завантаження сторінок, що важливо для комфортної роботи користувачів, особливо при нестабільному інтернет-з'єднанні.

4.3 Структура бази даних PostgreSQL

База даних платформи реалізована на основі реляційної СУБД PostgreSQL та побудована таким чином, щоб забезпечити надійне зберігання персональних даних користувачів, їхніх психоемоційних записів, навчального контенту й службової інформації, пов'язаної з рекомендаціями та історією взаємодій. Структура бази даних спроєктована відповідно до принципів нормалізації, що дозволяє мінімізувати дублювання інформації, зберігати цілісність даних і підтримувати чіткі реляційні зв'язки між сутностями.

Центральне місце в моделі даних займає таблиця users, у якій зберігаються основні облікові дані користувача: ідентифікатор, email, хешований пароль, роль, ім'я користувача, мова інтерфейсу, часовий пояс та інші налаштування (рис 4.14).

Для фіксації психоемоційного стану користувача використовується таблиця mood_entries, що містить посилання на користувача (userId), числову оцінку настрою, дату запису та, за потреби, текстовий коментар або набір тегів. Ці дані утворюють часовий ряд настрою, який надалі використовується як для побудови графіків, так і для формування рекомендацій. Текстові щоденникові записи зберігаються в таблиці journal_entries, де для кожного запису фіксуються ідентифікатор користувача, текст, дата створення та додаткові параметри. Обидві

таблиці (mood_entries і journal_entries) логічно пов’язані з користувачем через зовнішні ключі й слугують основним джерелом даних для модулів аналітики та рекомендацій.

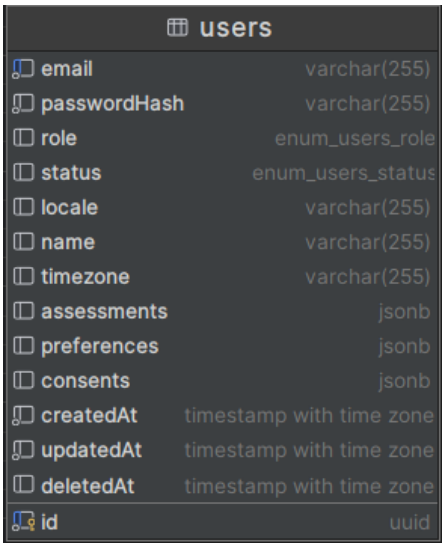

Рисунок 4.14 — Структура таблциці «users».

Навчальний контент моделюється через таблиці courses, lessons та exercises. Таблиця courses містить загальну інформацію про курси, lessons — про окремі уроки, прив’язані до курсу, а exercises — про конкретні вправи, які можуть входити до уроків. Структура побудована ієрархічно: курси пов’язані з уроками, а уроки — з вправами, що дозволяє формувати цілісний навчальний маршрут. Прогрес користувача відстежується у таблицях user_courses та user_exercises, де фіксується, які курси й вправи були розпочаті, виконані чи переглянуті користувачем.

Окремий блок структури пов’язаний із рекомендаційною системою. Таблиця recommendations зберігає інформацію про сформовані рекомендації, прив’язані до користувача, матеріалу та часу створення. Для оцінки якості та релевантності порад використовується таблиця recommendation_feedback, у якій зберігаються оцінки та відгуки користувача щодо запропонованих рекомендацій. Додатково, у таблиці prediction_logs можуть зберігатися службові дані про виконані прогнозні обчислення — наприклад, значення релевантності або параметри TF-IDF, що

дозволяє аналізувати роботу алгоритмів та вдосконалювати їх у майбутньому.

Модуль чату використовує таблицю messages, де для кожного повідомлення зберігаються автор (користувач або система), текст, час надсилання та прив'язка до користувача. Це дозволяє відновлювати контекст діалогів, аналізувати історію спілкування та потенційно використовувати ці дані як додаткове джерело текстової інформації для рекомендаційних алгоритмів.

Висновки до розділу 4

У четвертому розділі представлено комплексну програмну реалізацію веб-платформи підтримки ментального здоров'я, що охоплює проєктування архітектури, інтеграцію функціональних модулів та налаштування взаємодії між клієнтською й серверною частинами системи. У межах цього розділу послідовно описано логіку побудови всіх компонентів, їхню внутрішню організацію та способи забезпечення узгодженої роботи платформи як єдиного програмного продукту.

Серверну частину побудовано на основі фреймворку Nest.js, модульна структура якого забезпечує чітке логічне розділення компонентів, спрощує підтримку, підвищує масштабованість і дозволяє гнучко розширювати функціонал. Реалізовано ключові модулі платформи, серед яких: авторизація, щоденник, трекер настрою, рекомендаційна система, аналітика та чат-інтерфейс, кожен з яких виконує окрему функцію в межах цілісної системи.

Клієнтську частину створено за допомогою Next.js, що забезпечує високу швидкодію, оптимізацію рендерингу та ефективну маршрутизацію сторінок. У розділі детально описано принципи роботи інтерфейсних компонентів, підходи до валідації форм, організації стану додатку та інтеграції з REST API. Використання Tailwind CSS сприяло побудові адаптивного інтерфейсу, орієнтованого на зручність взаємодії користувача та доступність функціональних можливостей на різних пристроях.

Значну увагу приділено взаємодії з базою даних PostgreSQL через ORM

Sequelize. Реалізовано моделі, міграції та обробку запитів, що гарантує узгодженість даних та стабільність роботи системи. Проведені технічні рішення забезпечили захищений доступ до даних, оптимальний обмін інформацією між сервером і клієнтом та коректну обробку запитів у режимі реального часу.

Таким чином, у рамках розділу 4 завершено формування повноцінної програмної архітектури веб-платформи, інтегровано всі ключові модулі та реалізовано технологічні механізми, що забезпечують функціональність, надійність і масштабованість системи. Створена програмна інфраструктура є основою для подальшого тестування платформи, оцінювання її ефективності та адаптації до практичного використання.

5 РОБОТА КОРИСТУВАЧА З ДОДАТКОМ

У цьому розділі наведені системні вимоги для роботи з веб-додатком та сценарії роботи користувача з ним. У пункті 4.1 наведені системні вимоги, у пункті 4.2 описується взаємодія користувача з веб-додатком.

5.1 Системні вимоги

Для забезпечення коректної роботи веб-додатку комп'ютер повинен відповідати наступним вимогам:

— процесор — не гірше, ніж Intel® Core™ i3 або аналогічний від AMD. Це забезпечить достатню продуктивність для обробки сучасних веб-технологій;

— оперативна пам'ять — менше 4 GB оперативної пам'яті. Це дозволить плавно працювати з веб-додатком навіть при виконанні декількох завдань одночасно;

— графічне ядро — графічний адаптер повинен підтримувати технології не гірше, ніж Intel® HD Graphics 4000 або еквівалентний. Це забезпечить належне відображення графічного інтерфейсу додатку;

— операційна система — рекомендується використовувати сучасну операційну систему, таку як Windows 10, macOS 10.14 або новіші версії, або дистрибутиви Linux з актуальними оновленнями;

— браузер — сучасний веб-браузер з підтримкою актуальних веб-стандартів, таких як Google Chrome, Mozilla Firefox, Microsoft Edge або Safari. Важливо, щоб браузер був оновлений до останньої версії;

— інтернет-з'єднання — стабільне інтернет-з'єднання з мінімальною швидкістю 5 Мбіт/с для забезпечення швидкої передачі даних та плавної роботи веб-додатку.

Ці вимоги гарантують стабільну та ефективну роботу веб-додатку, забезпечуючи користувачам оптимальний досвід взаємодії.

5.2 Взаємодія користувача з програмним продуктом

Після відкриття веб-платформи користувач потрапляє на сторінку авторизації, де йому пропонується увійти до свого облікового запису (рис. 5.1). Інтерфейс сторінки входу виконаний у мінімалістичному стилі: містить поля для введення електронної пошти та пароля, кнопку «Увійти», а також посилання для переходу до форми реєстрації. Такий підхід забезпечує швидкий доступ до основного функціоналу та зручність для користувачів, які регулярно використовують платформу.

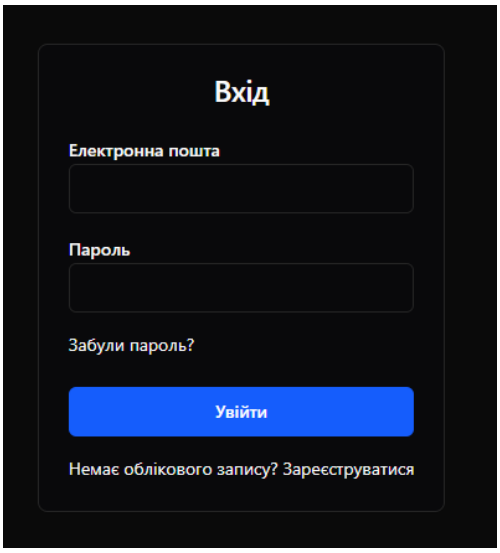

Рисунок 5.1 — Вхід до облікового запису

Якщо користувач ще не має свого акаунта, він може перейти до сторінки реєстрації, натиснувши відповідне посилання внизу форми. На сторінці реєстрації відображається форма з трьома полями — електронна пошта, пароль та підтвердження пароля (рис. 5.2). Для уникнення помилок під час введення даних форма оснащена валідацією, яка повідомляє користувача у разі некоректного формату email або невідповідності паролів. Після успішної реєстрації користувач одразу перенаправляється на головну сторінку платформи.

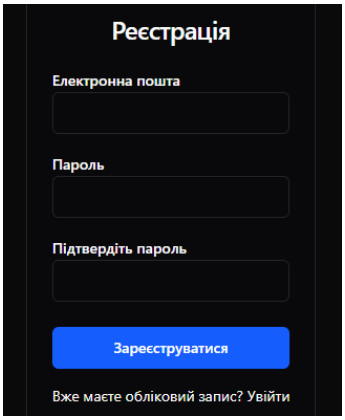

Рисунок 5.2 — Створення облікового запису

Потрапивши на головну сторінку (рис. 5.3), користувач бачить привітання та панель навігації, що дозволяє переходити до основних модулів системи: настрою, щоденника, вправ, курсів, рекомендацій та чату. У центральній частині розміщені картки з коротким підсумком стану користувача — оцінка настрою за останній день, тенденції змін настрою за два тижні та добірка персональних рекомендацій. Якщо записів настрою ще немає, система пропонує додати перший запис через кнопку «Додати запис».

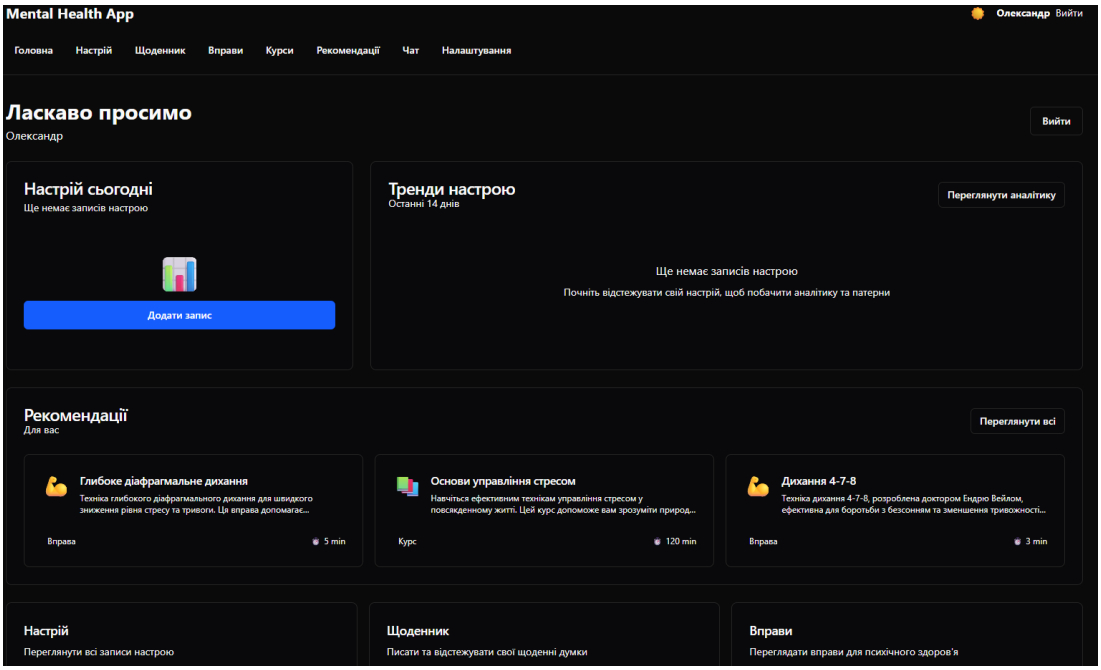

Рисунок 5.3 — Головна сторінка та посилання на інші частини веб-платформи.

При натисканні на цю кнопку відкривається модальне вікно (рис. 5.4), у якому користувач може обрати свою оцінку настрою за шкалою від 1 до 10, додати теги (наприклад, «щасливий», «втомлений», «тривожний») та залишити коротку нотатку.

Ця інформація зберігається у базі даних і використовується як для побудови аналітики, так і для роботи рекомендаційної системи. Після натискання кнопки «Зберегти» модальне вікно закривається, а новий запис миттєво відображається у всіх відповідних розділах інтерфейсу.

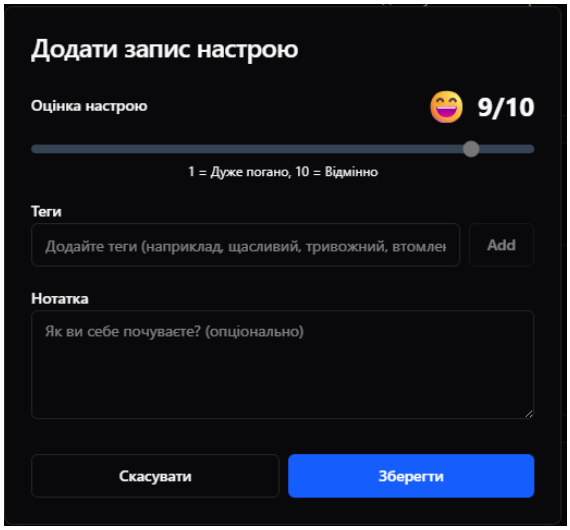

Рисунок 5.4 — Модальне вікно для додавання запису настрою.

У розділі «Настрій» користувач отримує доступ до повного списку створених записів та може переглядати аналітику змін свого психоемоційного стану за вибраний період (рис. 5.5). На сторінці аналітики відображаються середня, максимальна і мінімальна оцінки настрою, а також графік динаміки записів. Графік побудований за допомогою бібліотеки Chart.js, що забезпечує інтерактивність та наочність.

Це дозволяє користувачу бачити загальні тенденції та коригувати свою активність на платформі відповідно до отриманої інформації.

Рисунок 5.5 — Сторінка «Аналітика настрою»

Розділ «Щоденник» надає змогу вести текстові записи, що допомагають користувачеві рефлексувати та фіксувати власний психоемоційний досвід (рис. 5.6). Інтерфейс містить список усіх створених записів та поле пошуку за ключовими словами або тегами.

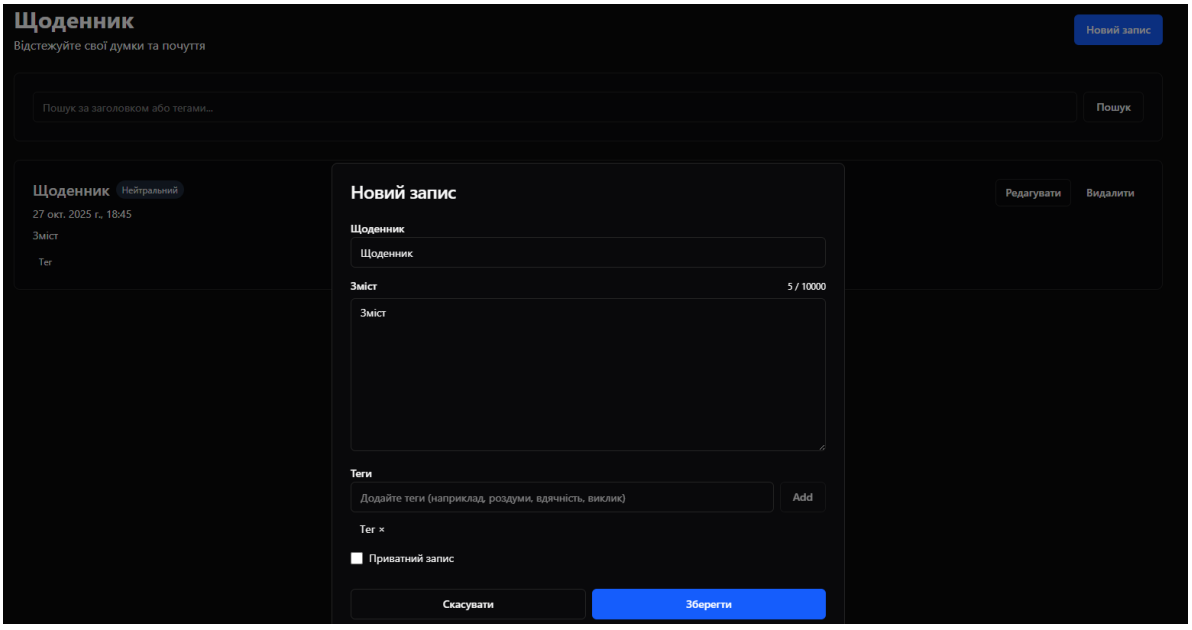

Рисунок 5.6 — Сторінка «Щоденник» та модальне вікно створення нового запису.

Натискання кнопки «Новий запис» відкриває окрему форму, у якій користувач має можливість створити новий елемент щоденника. У цій формі передбачено поле для введення текстового опису події, думки або спостереження, а також інструменти для вибору відповідних тегів, що допомагають структурувати записи та забезпечують зручність їх подальшого пошуку. Окрім цього, користувач може позначити запис як приватний. Приватні записи не враховуються у рекомендаційних алгоритмах, що дає змогу повністю контролювати рівень конфіденційності та обмежувати використання чутливих даних у процесі формування порад або аналітичних висновків.

Розділ «Вправи» представлений у вигляді каталогу вправ з фільтрами за рівнем складності, тематикою чи тривалістю (рис. 5.7).

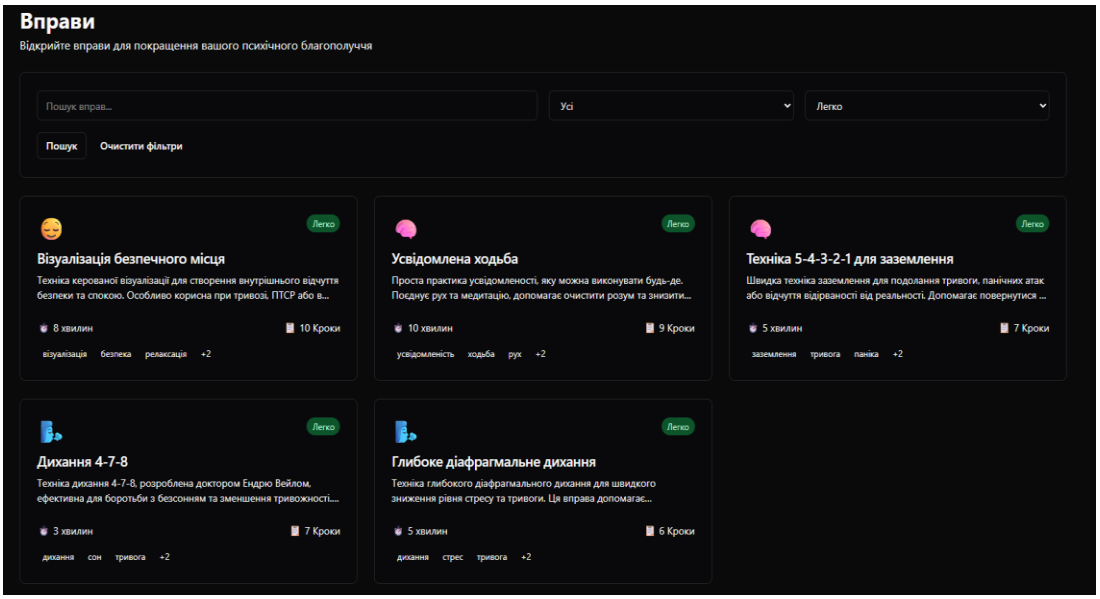

Рисунок 5.7 — Перелік доступних вправ з рівнем важкості «Легко».

Кожна вправа відображена у форматі картки, де міститься короткий опис, кількість кроків, тривалість та відповідні теги.

Після вибору вправи користувач переходить до інтерактивного інтерфейсу її виконання з покроковими інструкціями та таймером (рис. 5.8). Завершивши вправу, користувач може оцінити її ефективність та залишити нотатку, що буде використано для персоналізації наступних рекомендацій.

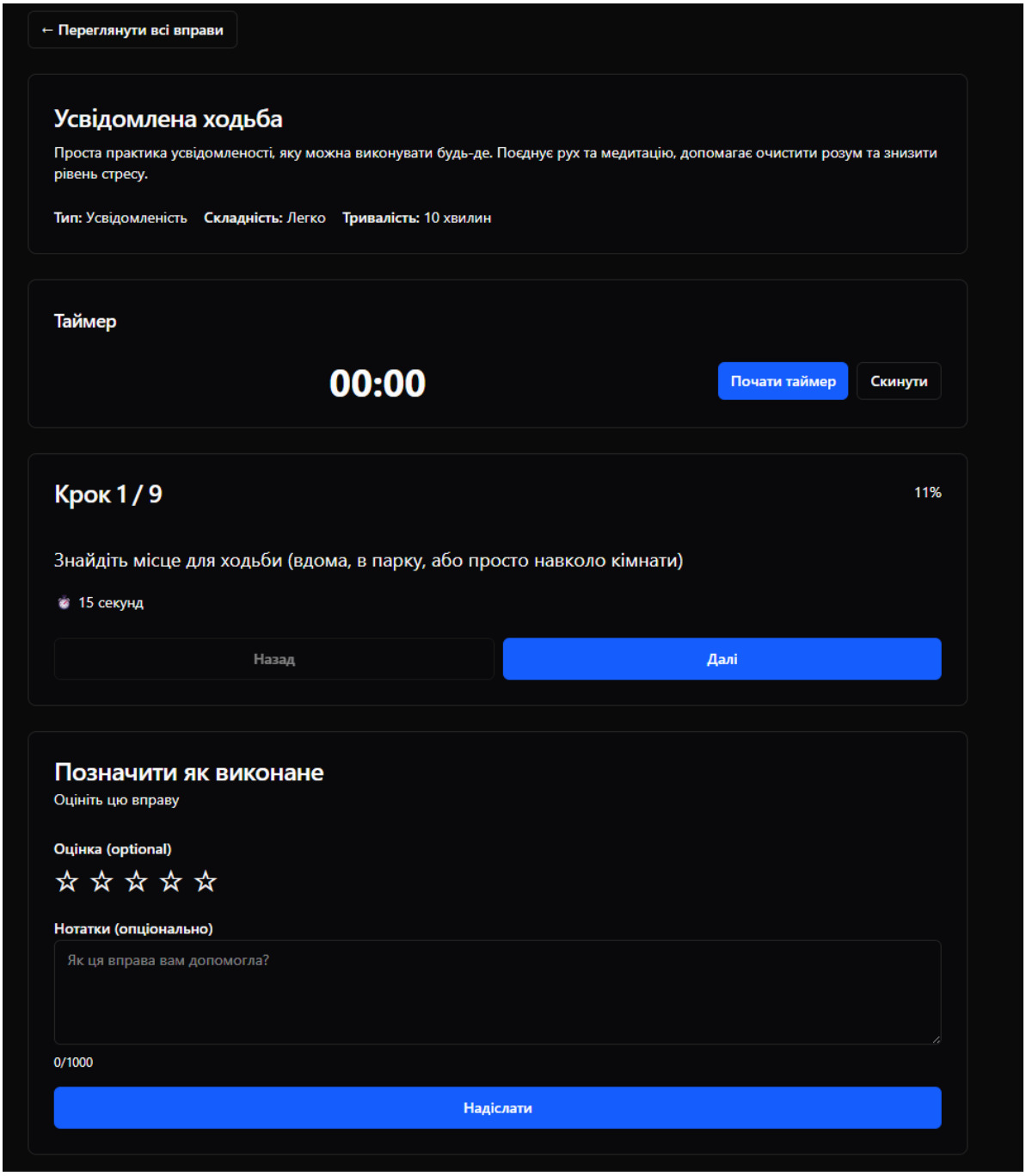

Рисунок 5.8 — Приклад інтерфейсу вправи під час її виконання.

Розділ «Курси» також містить зручний інтерфейс для фільтрації та вибору навчальних програм (рис. 5.9).

Кожен курс складається з набору уроків, організованих у послідовну програму. При відкритті курсу користувач бачить його повну структуру, короткий опис, тривалість та прогрес проходження.

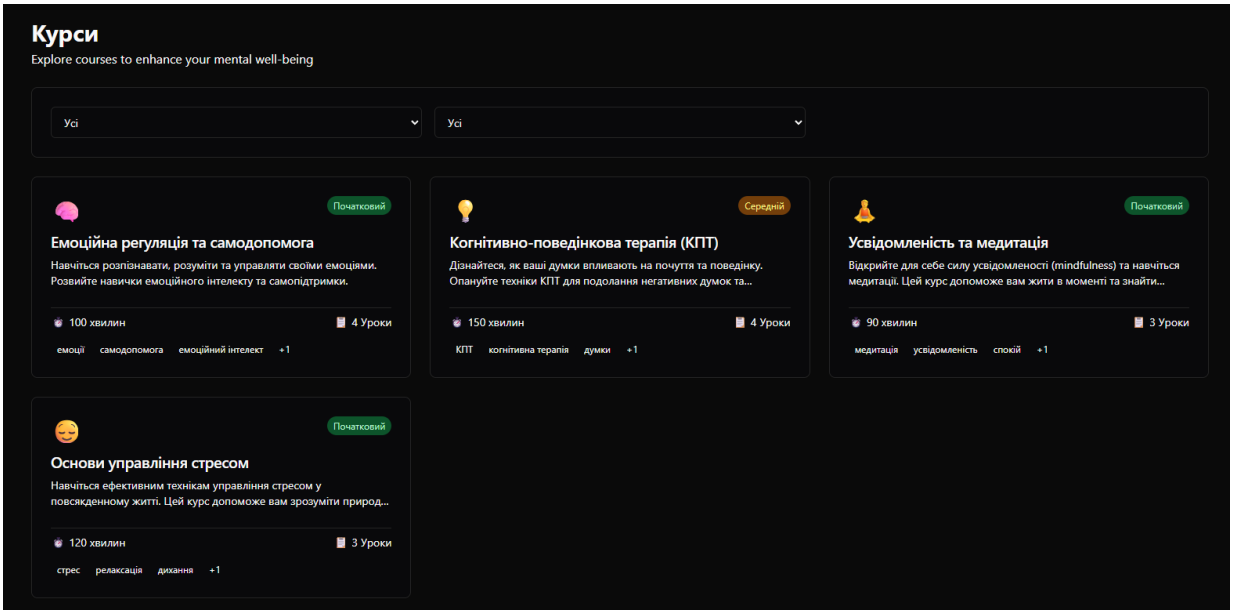

Рисунок 5.9 — Сторінка «Курси»

Переходячи до уроків, користувач отримує структурований навчальний матеріал у вигляді тексту, зображень або інтерактивних елементів (рис. 5.10). Після завершення уроку його можна позначити як виконаний, а система автоматично перенаправить користувача до наступного етапу курсу.

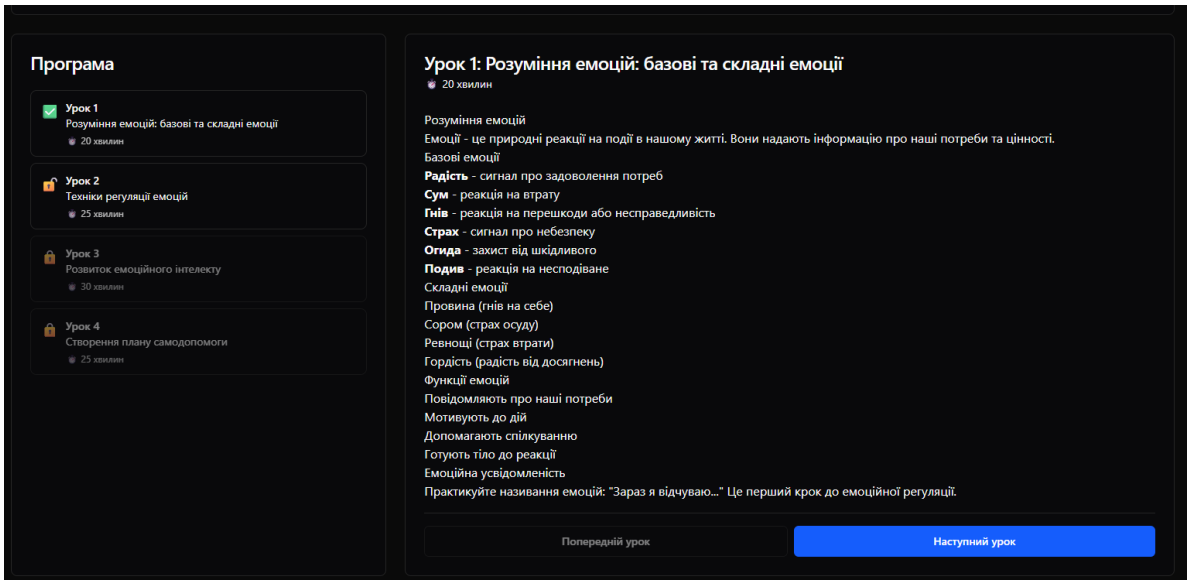

Рисунок 5.10 — Інтерфейс курсу «Емоційна регуляція та самодопомога», та демонстрація його проходження.

На сторінці «Рекомендації» користувач отримує персоналізований перелік курсів і вправ, які система підібрала на основі його психоемоційних записів, активності в щоденнику та взаємодії з навчальним контентом (рис. 5.11). Кожна рекомендація відображається у вигляді окремої картки, де зазначено категорію матеріалу (курс або вправа), рівень складності, короткий опис, тривалість та теги. Користувач може перейти до детального перегляду матеріалу, натиснувши кнопку «Переглянути деталі», або вплинути на якість наступних рекомендацій, оцінивши корисність матеріалу через кнопки «Подобається» та «Не цікавить».

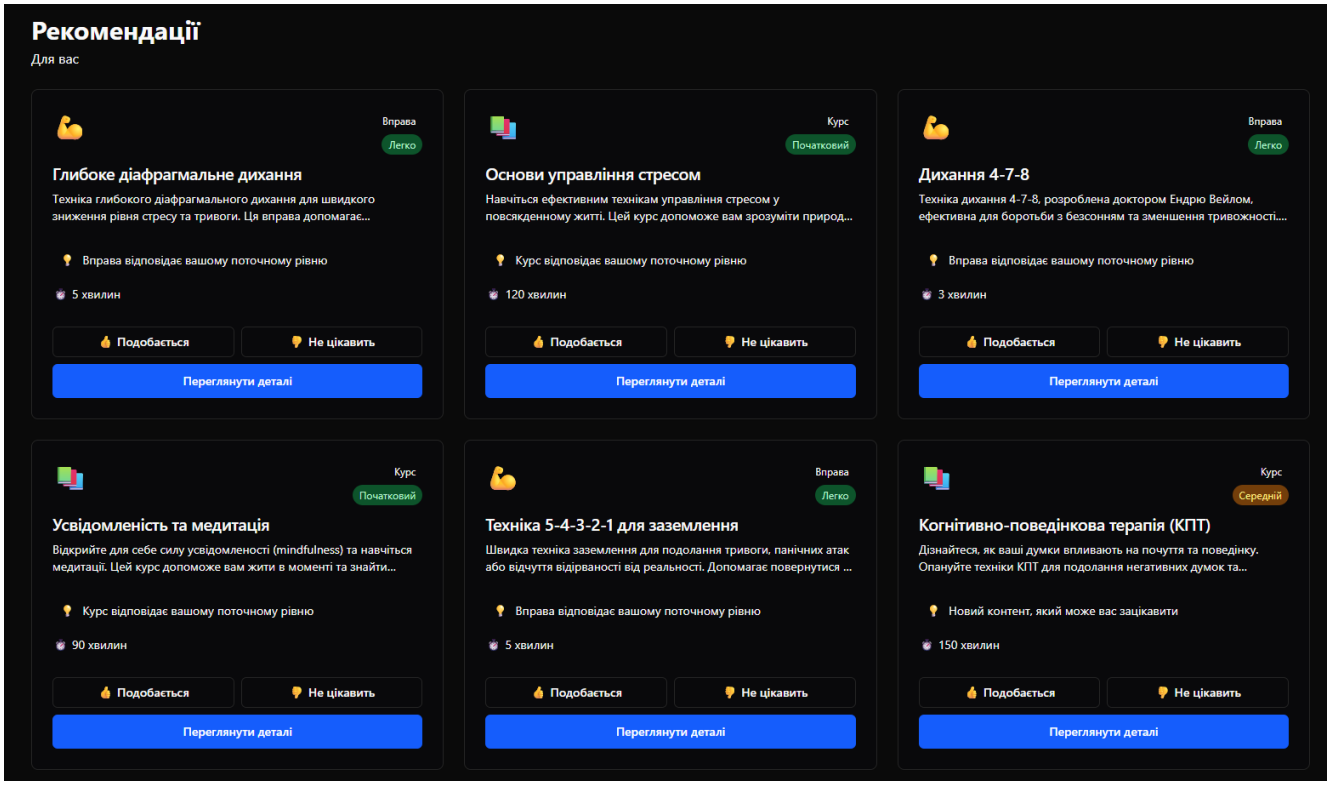

Рисунок 5.11 — Сторінка «Рекомендації».

Такі оцінки відіграють важливу роль у поступовому вдосконаленні рекомендаційного алгоритму. Система зберігає інформацію про вподобання користувача і під час наступних рекомендацій враховує як подібність матеріалу до текстових записів, так і історію взаємодій. Завдяки цьому платформа стає дедалі точнішою, пропонуючи ті вправи та курси, що найбільш відповідають поточним потребам і настрою користувача. Крім того, рекомендації можуть бути динамічно

оновлені, якщо система помітить зміни в емоційному стані або активності користувача.

Сторінка «Чат» надає можливість вести діалог із інтегрованим чат-ботом, який працює як підтримувальний помічник, а також як інтерфейс для швидкого отримання рекомендацій (рис. 5.12). Чат створений у форматі сучасного месенджера: зліва розташовується історія повідомлень, а знизу — поле для введення тексту. Користувач може звертатися до бота з будь-якими запитами, пов'язаними з власним психоемоційним станом, настроєм, техніками релаксації чи порадами щодо самодопомоги.

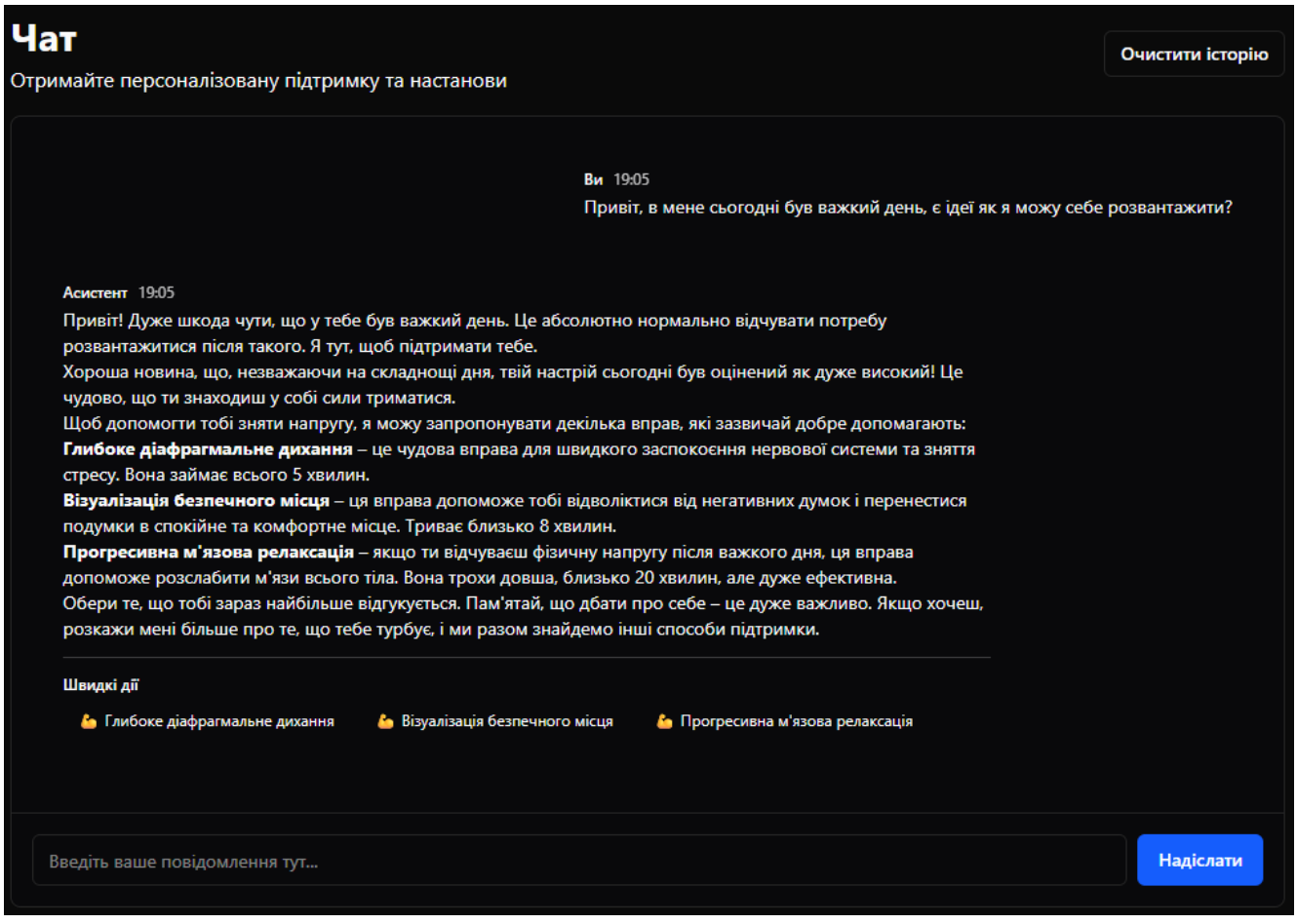

Рисунок 5.12 — Приклад відповіді від боту з використанням ШІ.

Відповідь чат-бота формується на основі інформації про записи настрою, виконані вправи, переглянуті курси та загальну історію взаємодій користувача з

платформою. Це дозволяє адаптувати повідомлення до контексту та робити рекомендації більш точними та корисними. Наприклад, якщо користувач повідомляє про втому чи тривожність, бот може запропонувати заспокійливі вправи, короткі дихальні техніки або відповідні уроки з курсу з управління емоціями.

Після кожного повідомлення чат може автоматично пропонувати набір варіантів під заголовком «Швидкі дії». Ці рекомендації представлені у вигляді кнопок, натискання на які миттєво перенаправляє користувача на відповідну вправу чи курс. Така інтерактивність дозволяє користувачеві отримати підтримку без необхідності переходити через навігаційне меню, що робить взаємодію з платформою більш природною та швидкою.

Чат-бот виконує не лише функцію рекомендацій, але й допомагає користувачу відчути підтримку у складні моменти. Під час діалогу він може давати емоційно нейтральні або підтримувальні відповіді, підказки щодо технік саморегуляції, або пропонувати переглянути аналітику настрою, якщо поведінкові зміни вказують на пригнічений стан. Такий підхід робить чат не просто технічним інструментом, а повноцінною складовою системи психологічної підтримки.

У цілому взаємодія з розділами «Рекомендації» та «Чат» формує для користувача індивідуальний досвід, де кожен елемент інтерфейсу працює на підтримку психоемоційного благополуччя. Рекомендаційна система пропонує релевантні вправи та курси, тоді як чат надає можливість отримати миттєву підтримку, поради та доступ до корисної інформації, адаптованої до особистих потреб користувача.

Висновки до розділу 5

У п’ятому розділі розглянуто процес взаємодії користувача з веб-платформи та проаналізовано роботу ключових функціональних модулів з позиції кінцевого користувача. На основі опису сценаріїв користування продемонстровано, як інтерфейс, механізми навігації та логіка модулів забезпечують інтуїтивність,

доступність та ефективність використання системи. Визначено, що побудована структура інтерфейсу дозволяє швидко переходити між основними інструментами платформи — щоденником, трекером настрою, аналітикою, чат-інтерфейсом та рекомендаційною системою.

Окрему увагу приділено відображенню аналітичної інформації та методам візуалізації даних, які сприяють формуванню у користувача чіткого розуміння динаміки власного емоційного стану. Вбудовані графіки, історія настрою та аналітичні звіти дозволяють відстежувати зміни у психоемоційному стані, виявляти закономірності та аналізувати вплив різних факторів на самопочуття. Це підсилює практичну цінність платформи й сприяє формуванню більш усвідомленої моделі поведінки.

Детально описано роботу рекомендаційної системи, яка адаптує поради відповідно до історії взаємодії користувача з платформою та контексту його психологічних записів. Реалізація рекомендаційних механізмів демонструє, що обрані алгоритми можуть ефективно застосовуватися для персоналізованої підтримки та стимулювання позитивних змінотворчих процесів у поведінці користувача.

Розглянуто також технічні та функціональні аспекти експлуатації системи, зокрема системні вимоги, захист даних та забезпечення безпечного доступу до особистої інформації. Аналіз показав, що платформа здатна працювати стабільно, забезпечуючи високу якість користувацького досвіду та відповідність функціональних можливостей цільовому призначенню.

Таким чином, у розділі 5 підтверджено, що створена веб-платформа задовольняє потреби користувача у відстеженні емоційного стану, отриманні індивідуальних рекомендацій і веденні психологічних записів, а також забезпечує зручність, зрозумілість та ефективність взаємодії кінцевого користувача з системою.

6 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Розробка стартап-проєкту є невід'ємною частиною дослідження, що передбачає цілісний і стратегічний підхід до створення інноваційного цифрового продукту у сфері ментального здоров'я. Веб-платформа, яка поєднує функції самоспостереження, аналізу психоемоційного стану та персоналізованих рекомендацій із використанням машинного навчання, потребує ретельної оцінки її практичної цінності, технічної доцільності та ринкового потенціалу.

У межах цього розділу буде представлено поетапний підхід до формування стартапу — від валідації проблеми та концепції, аналізу конкурентного середовища, до вибору бізнес-моделі та розробки стратегії виходу на ринок. Особлива увага приділяється технологічному аудиту, визначенню унікальної ціннісної пропозиції, оцінці потреб цільової аудиторії, а також підготовці продукту до потенційної комерціалізації.

Розділ містить техніко-економічне обґрунтування проєкту, аналіз можливостей масштабування, стратегій монетизації та шляхів просування веб-платформи у цифровому середовищі. Такий підхід дозволяє інтегрувати наукові дослідження у прикладне інноваційне рішення з реальним соціальним і ринковим впливом.

6.1 Загальна ідея веб-платформи

Основна ідея проєкту полягає у створенні інтелектуалізованої веб-платформи підтримки ментального здоров'я, яка надає користувачам персоналізовані поради, аналітику та інструменти самодопомоги на основі аналізу емоційного стану. У сучасному темпі життя психоемоційне навантаження стає поширеною проблемою, а традиційні форми психологічної підтримки не завжди доступні. Наявні цифрові рішення здебільшого обмежуються фіксацією настрою або наданням загальних порад, ігноруючи індивідуальні особливості користувачів.

Запропонована система вирішує цю проблему за рахунок інтеграції

алгоритмів машинного навчання (TF-IDF, cosine similarity), моделі NLP для обробки щоденникових записів, а також застосування мовної моделі Gemini AI у модулі чат-помічника. Платформа дозволяє фіксувати зміни настрою, створювати приватні записи, проходити вправи з емоційного саморегулювання та отримувати адаптовані рекомендації на основі поведінки, текстів і прогресу користувача.

Ключовою відмінністю платформи є гнучка система персоналізації, яка враховує контекст попередніх записів, актуальний емоційний стан, відгуки на вправи, а також частоту використання системи. Інтерфейс платформи побудовано з акцентом на зручність, емоційну підтримку та інтерактивність. Система також містить модулі візуалізації аналітики, які дозволяють користувачу бачити тенденції змін настрою, середні оцінки по днях, найбільш вживані теги тощо.

Веб-платформа буде особливо корисною для студентів, працівників інтелектуальної сфери, осіб у стресових станах, а також для освітніх закладів та компаній, які зацікавлені у психологічному благополуччі своїх учасників. У перспективі платформа може бути масштабована як корпоративний інструмент емоційного моніторингу та самодопомоги.

У цьому розділі описано основну ідею стартап-проєкту, потенційні сфери його застосування, а також ключові переваги, які отримають кінцеві користувачі. Узагальнена інформація наведена у таблиці 6.1.

Таблиця 6.1 — Опис ідеї проєкту стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка веб-платформи для підтримки ментального здоров’я з інтеграцією рекомендаційних алгоритмів та AI	Індивідуальне використання людьми, які потребують психологічної підтримки або самоспостереження	Можливість відстеження емоційного стану, отримання персоналізованих порад і саморегуляції настрою

Кінець таблиці 6.1.

	Використання в навчальних закладах для студентів та викладачів	Формування здорового емоційного середовища, підтримка психоемоційного балансу учасників освітнього процесу
	Застосування у корпоративному середовищі HR-відділами	Моніторинг емоційного стану працівників, зменшення ризику вигорання, підвищення продуктивності

Під час виконання дослідження було проведено аналіз техніко-економічних переваг розробленої веб-платформи підтримки ментального здоров'я у порівнянні з наявними цифровими рішеннями у цій сфері. Для порівняння були обрані найбільш поширені сервіси з подібним функціоналом: Woebot Health, Daylio, та Headspace [31, 32, 33]. Порівняння здійснювалося за низкою ключових параметрів, включаючи доступність, ступінь персоналізації, наявність аналітики та рекомендацій, можливість взаємодії з AI, тощо. Результати аналізу наведено у таблиці 6.2.

Таблиця 6.2 — Результати аналізу техніко-економічних переваг

Техніко-економічні характеристики	Розроблена система	Woebot Health	Daylio	Headspace
Призначення	Підтримка ментального здоров'я, щоденник,	Діалоговий бот для CBT-підтримки	Трекер настрою з нотатками	Медитації та вправи для релаксації

Продовження таблиці 6.2.

	аналітика, рекомендації			
Доступність	Веб-платформа, доступ з будь-якого пристрою	Мобільний застосунок	Мобільний застосунок	Веб та мобільний застосунок
Персоналізація порад	Висока, на основі NLP та ML	Середня, адаптація через діалог	Обмежена, вручну	Обмежена до тематичних медитацій
Наявність AI-бота	Так, інтеграція з Gemini AI	Так	Ні	Частково (озвучений гайд)
Фіксація настрою	Так, із гнучкими параметрами та тегами	Так	Так	Ні
Аналітика настрою	Так, візуалізація та графіки	Обмежена	Базова статистика	Відсутня
Навчальний контент	Є вправи та мікрокурси	Немає	Немає	Так, розділено за темами та складністю
Взаємодія користувача з інтерфейсом	Інтуїтивно зрозумілий, багатосторінковий інтерфейс	Чат-інтерфейс	Простий, з фіксацією настрою	Тематичні сесії через меню
Можливість створення профілю	Так, з авторизацією та збереженням сесій	Так	Так	Так

Кінець таблиці 6.2.

Оновлення та підтримка	Активна підтримка	Активна підтримка	Активна підтримка	Активна підтримка

Результати проведеного порівняльного аналізу свідчать про те, що розроблена система поєднує у собі ключові функціональні переваги провідних аналогів, при цьому доповнюючи їх розширеними можливостями персоналізації та аналітики. Завдяки гнучкій архітектурі, інтеграції алгоритмів машинного навчання та наявності інтелектуального чат-помічника на базі Gemini AI, система виходить за межі типових трекерів настрою чи медитативних платформ.

Серед основних конкурентних переваг слід виділити можливість індивідуального підходу до користувача за рахунок аналізу текстових записів і настрою, зручний та інтуїтивно зрозумілий інтерфейс, а також підтримку модулів аналітики й освітнього контенту. На відміну від більшості існуючих рішень, система забезпечує глибоку інтеграцію функцій — від щоденника настрою до рекомендацій та навчальних мікрокурсів — що підвищує її цінність для широкого кола користувачів.

Таким чином, запропонована платформа має високий потенціал для конкурентної реалізації на ринку цифрових сервісів ментального здоров’я. Вона здатна не лише конкурувати з уже наявними рішеннями, а й сформувати новий підхід до підтримки емоційного благополуччя через інтелектуалізовані веб-сервіси.

6.2 Технологічний аудит стартап-проєкту

У процесі проведення технологічного аудиту стартап-проєкту було здійснено аналіз застосованих інструментів та бібліотек, що використовуються в архітектурі веб-платформи підтримки ментального здоров’я. Основною метою є підтвердження відповідності вибраного стеку технологій сучасним вимогам до масштабованості, стабільності, безпеки та зручності розвитку проєкту.

Аудит охоплює клієнтську частину, серверну логіку, базу даних, а також

модулі з машинним навчанням і генеративним AI. Оцінено доступність рішень, їхні переваги та потенційні обмеження. Результати подано у таблиці 6.3.

Таблиця 6.3 — Технології, що використані у проєкті

Компонент проєкту	Використані технології	Доступність	Переваги	Недоліки
Клієнтська частина	Next.js + React	Безкоштовно	SSR + SPA, модульність, гнучкість, висока продуктивність	Потребує належної оптимізації при зростанні масштабів
	Tailwind CSS	Безкоштовно	Швидка адаптивна стилізація, сучасний вигляд	Нетиповий синтаксис, вимагає звикання
	React Query	Безкоштовно	Кешування запитів, просте керування станом, автоматичне оновлення даних	Додаткова абстракція, потребує часу на освоєння
	Zod (валідація форм)	Безкоштовно	Потужна типізація, інтеграція з TypeScript	Обмежена підтримка кастомних схем
	Gemini AI API	Платно	Мультимодальність, підтримка персоналізованої генерації відповідей	Потребує платної підписки, залежність від зовнішнього API
Серверна частина	Nest.js	Безкоштовно	Модульність, підтримка TypeScript, зручна робота з REST API	Вища крива навчання порівняно з Express.js
База даних	PostgreSQL	Безкоштовно	Надійна, розширювана, підтримка складних запитів	Потребує ручної оптимізації при роботі з великими обсягами

Кінець таблиці 6.3.

ORM	Sequelize	Безкоштовно	Абстракція бази даних, простота інтеграції	Вимагає налаштування типів при роботі з TS
Аналітика + Рекомендації	TF-IDF + Cosine Similarity	Безкоштовно	Простота реалізації, ефективність для коротких текстів	Менш ефективна для великих корпусів або контекстуальних запитів

Для реалізації стартап-платформи були обрані такі основні технології:

— Next.js — для клієнтської частини з підтримкою SSR/SPA;

— Nest.js — як основа серверної логіки з чіткою структурою;

— PostgreSQL та Sequelize — для збереження та обробки даних;

— React Query, Zod, Tailwind CSS — для клієнтського комфорту та продуктивності;

— TF-IDF та Cosine Similarity — як основа системи рекомендацій;

— Gemini AI — для побудови інтелектуального чат-помічника.

Застосування саме цього технологічного стеку дозволяє забезпечити як швидку початкову розробку, так і довгострокове масштабування проєкту. Гнучкість, модульність і відповідність сучасним стандартам створюють сприятливі умови для подальшого розвитку системи. Разом із тим, необхідно враховувати потребу в періодичному оновленні залежностей, забезпеченні безпеки даних користувачів та витрати на підтримку зовнішніх сервісів, зокрема API великих мовних моделей.

6.3 Аналіз ринкових можливостей запуску стартап-проєкту

Аналіз ринку є критично важливим етапом для запуску стартап-проєкту, оскільки він дозволяє оцінити масштаб потенційної аудиторії, рівень конкуренції, зрілість цифрової культури в обраній галузі та ймовірність комерційного успіху. У сфері ментального здоров’я попит на інноваційні рішення невпинно зростає через

глобальну діджиталізацію, зміну суспільних цінностей, а також посилення факторів стресу в повсякденному житті.

Особливої актуальності набувають платформи самодопомоги, здатні забезпечити персоналізований підхід, збереження приватності та доступність у будь-який час. Незважаючи на наявність певної кількості конкурентних рішень, ринок ментального добробуту все ще перебуває на етапі активного зростання, що створює сприятливі умови для входження нового продукту.

У таблиці 6.4 наведено основні показники, що характеризують поточний стан цільового ринку для платформи підтримки ментального здоров'я.

Таблиця 6.4 — Характеристика стану ринку продукту

Показник стану ринку	Характеристика
Потреба у продукті	Зростаюча глобальна потреба у цифрових інструментах ментального здоров'я для саморегуляції, профілактики стресу, тривожності та вигорання.
Потенційна кількість користувачів	Від 1 до 3 млн щороку в Україні та значно більше на глобальному ринку; цільова аудиторія — молодь, студенти, працівники сфери освіти та IT.
Динаміка ринку	Ринок активно зростає на 12-20% щороку. В Європі та США розвиваються тренди на ментальну підтримку як частину корпоративної та освітньої політики.
Рівень цифровізації	Високий — більшість молодих користувачів надають перевагу онлайн-форматам. Зростає довіра до цифрових сервісів для самодопомоги.
Показники конкурентного середовища	Існує кілька популярних застосунків (Woebot, Headspace, Daylio), проте багато з них орієнтовані на вузький функціонал або іноземну аудиторію.

Незважаючи на наявність декількох мобільних застосунків у сфері ментального здоров'я, попит на персоналізовані й доступні інструменти

психоемоційної підтримки стабільно зростає. Це пов'язано зі зміною соціального сприйняття ментального благополуччя, підвищенням рівня стресу серед молоді та працівників інтелектуальної сфери, а також з розвитком культури турботи про себе (self-care). У цьому контексті важливо чітко визначити цільові групи користувачів, їхні цифрові звички, очікування від продукту та ключові функціональні запити.

У таблиці 6.5 представлено характеристику потенційних груп користувачів, їх поведінкові особливості та вимоги до функціоналу платформи.

Таблиця 6.5 — Характеристика потенційних користувачів платформи

Потреба ринку	Цільова аудиторія	Особливості поведінки	Вимоги користувачів до продукту
Самоспостереження за емоційним станом	Студенти, молодь, працівники IT-сфери	Регулярне використання мобільних або веб-додатків, високий рівень цифрової грамотності	Простий інтерфейс, швидкий запис настрою, візуалізація динаміки, приватність
Можливість отримання порад та підтримки	Користувачі з підвищеним рівнем тривожності, самотності	Потреба у швидкій допомозі, небажання звертатися до психолога офлайн	Рекомендації на основі стану, діалог з ботом, навчальні вправи
Популяризація турботи про ментальне здоров'я	HR-відділи компаній, викладачі, освітні заклади	Залучення до підтримки персоналу, потреба в корпоративних інструментах	Можливість моніторингу стану колективу, агреговані звіти, доступ до контенту для команд
Навчання технік емоційної саморегуляції	Школярі, студенти, педагоги	Потреба у простих, коротких вправах для щоденного використання	Курси/вправи з рівнями складності, таймер, збереження прогресу
Персоналізовані рекомендації	Усі категорії користувачів	Очікування релевантності контенту до настрою та історії взаємодії	Інтеграція з NLP/ML, можливість фільтрації по тематиках, оновлювані рекомендації

Реалізація стартап-проєкту у сфері цифрової підтримки ментального здоров'я супроводжується низкою потенційних ризиків, які можуть негативно вплинути на ефективність впровадження та масштабування платформи. Серед таких — технічні складнощі, етичні виклики, недостатня довіра користувачів до алгоритмів, а також конкуренція з боку вже наявних рішень. Для забезпечення стабільності продукту на ринку необхідно завчасно врахувати основні загрози та розробити стратегії їх подолання.

У таблиці 6.6 наведено перелік основних ризиків і відповідних дій для їх зниження або уникнення.

Таблиця 6.6 — Потенційні ризики реалізації стартап-проєкту та способи їх уникнення

Ризик / загроза	Характеристика	Шлях мінімізації / усунення
Недовіра до системи зі сторони користувачів	Побоювання щодо конфіденційності даних або ефективності рекомендацій	Забезпечення прозорої політики приватності, шифрування даних, сертифікація безпеки
Конкуренція з популярними додатками	Наявність вже сформованих брендів (Woebot, Headspace, Calm)	Унікальна функціональність: комбінація аналітики, порад, щоденника та навчального контенту
Висока залежність від зовнішніх API (наприклад, Gemini AI)	Можливі зміни тарифів, обмеження в доступі	Розробка резервного режиму функціонування та альтернативних моделей
Низька активність користувачів після реєстрації	Відсутність мотивації повертатися до застосунку	Впровадження гейміфікації, нагадувань, індивідуальних повідомлень від чат-помічника

Кінець таблиці 6.6.

Етичні обмеження щодо надання порад	Система не може замінити консультацію з фахівцем	Чітке юридичне повідомлення в інтерфейсі, рекомендація звертатися до спеціаліста при потребі
Технічні труднощі масштабування системи	Збільшення навантаження на сервери у разі зростання аудиторії	Оптимізація запитів, горизонтальне масштабування інфраструктури, моніторинг продуктивності
Відсутність стійкої бізнес-моделі	Проблеми з монетизацією або фінансуванням	Створення декількох джерел прибутку: freemium, B2B-ліцензії, курси, грантові програми

Більшість ризиків для веб-платформи підтримки ментального здоров'я пов'язані з високим рівнем конкуренції на ринку цифрового добробуту, швидкою динамікою розвитку технологій, а також суворими вимогами щодо збереження приватності та безпеки користувацьких даних. Попри це, впроваджені в проєкті технічні та організаційні рішення дають змогу суттєво зменшити вплив зазначених факторів, забезпечити стабільність функціонування системи та створити надійну основу для її подальшого масштабування.

Успішне просування платформи на ринку підтримується сукупністю сприятливих умов. До них належать стійкий попит на персоналізовану психологічну підтримку, зростання цифрової грамотності та готовності користувачів застосовувати технологічні інструменти для покращення свого емоційного стану, а також унікальна комбінація технологічних рішень, використаних у межах проєкту. Такий комплекс факторів забезпечує платформі стійкі конкурентні переваги та розширює потенціал її подальшого розвитку.

Таблиця 6.7 демонструє ключові чинники, які підвищують конкурентоспроможність та перспективи розвитку платформи.

Таблиця 6.7 — Потенціал розвитку стартап-платформи

Фактор	Опис
Високий суспільний попит	Постійне зростання попиту на сервіси самодопомоги, зниження стигми щодо звернення по психологічну підтримку
Інтеграція сучасних технологій	Використання ML, NLP, TF-IDF, інтеграція Gemini AI для генерації відповідей підвищує інноваційність проєкту
Універсальність цільової аудиторії	Продукт однаково корисний для студентів, працівників, освітніх закладів і корпоративного сегмента
Гнучка архітектура	Система легко адаптується до нових модулів, має REST API, масштабована серверна частина
Відсутність національних аналогів	Більшість схожих рішень — іноземного походження, платні або не локалізовані
Підтримка трендів цифрової психогігієни	Зростаюча популярність медитацій, журналів настрою та чат-підтримки у молодіжному середовищі
Перспективи монетизації	Freemium-модель, можливість B2B-пропозицій, участь у грантових ініціативах

Важливим фактором, що сприяє розвитку веб-платформи підтримки ментального здоров'я, є стрімке зростання попиту на персоналізовані цифрові рішення у сфері психоемоційного благополуччя. Незважаючи на наявність декількох міжнародних конкурентів, український ринок практично вільний від локалізованих продуктів із повним функціональним комплексом — щоденником настрою, аналітикою, вправами, рекомендаціями та AI-помічником.

Конкурентна перевага платформи також полягає у використанні сучасних мовних моделей, можливості швидкої модифікації функціоналу та гнучкої адаптації під освітні або корпоративні потреби. Аналіз конкурентного середовища є ключовим для розробки дієвої маркетингової стратегії та позиціонування

проєкту. У таблиці 6.8 представлено ступеневий аналіз конкуренції з описом викликів і запропонованих дій.

Таблиця 6.8 — Ступеневий аналіз конкуренції на ринку

Рівень	Характеристика конкурентів	Загроза / виклик	Стратегія реагування
Прямі конкуренти	Woebot, Headspace, Daylio — мобільні додатки з фокусом на емоційне здоров'я	Відомі бренди з потужними маркетингами, присутність на глобальному ринку	Фокус на локалізацію, розширення функціоналу, інтеграція з навчальними і HR-платформами
Опосередковані	Психологічні онлайн-сервіси, форуми, YouTube-канали зі схожою тематикою	Альтернативні канали самодопомоги без глибокої інтеграції або структурованого контенту	Наголос на достовірності, науковому підґрунті, інтерактивності і персоналізації
Потенційні гравці	Нові проєкти на основі AI/ML у сфері добробуту	Висока швидкість розвитку технологій, ризик втрати унікальності	Швидке впровадження інновацій, публічний беклог оновлень, користувацьке голосування
Субститути	Класичні офлайн-консультації, гарячі лінії, навчальні курси	Часткове заміщення функцій, вища довіра до «живої» допомоги	Позиціонування як підтримки між сесіями або доступної альтернативи з AI-компонентами

Проведений аналіз конкурентного середовища дозволив визначити ключові виклики, з якими стикається проєкт, а також сформувати релевантні стратегії позиціонування. У свою чергу, переваги розробленої платформи засновані на поєднанні технологічної інноваційності, гнучкої архітектури, високого рівня персоналізації та актуального функціонального наповнення.

Таблиця 6.9 узагальнює основні фактори, які забезпечують високу конкурентоспроможність веб-системи підтримки ментального здоров'я. Серед них — інтеграція машинного навчання, мультифункціональність, простота у використанні та адаптованість до потреб різних категорій користувачів.

Таблиця 6.9 — Фактори конкурентоспроможності розробленої системи

Фактор конкурентоспроможності	Обґрунтування / характеристика
Висока гнучкість системи	Можливість масштабування, підключення модулів, розширення під корпоративні чи освітні потреби
Технологічна новизна	Використання TF-IDF, Cosine Similarity, Nest.js, Next.js, React Query та генеративного ШІ (Gemini)
Інтуїтивно зрозумілий інтерфейс	Простота взаємодії з додатком, що сприяє залученню широкої аудиторії без потреби в навчанні
Локалізація та культурна адаптація	Підтримка української мови, врахування національного контексту, гнучке позиціонування на місцевому ринку
Комплексний функціонал	Щоденник настрою, рекомендації, аналітика, навчальні вправи, курси, чат — усе в межах однієї системи
Мультиплатформенність	Повна підтримка веб-доступу з різних пристроїв без потреби встановлення застосунків
Відкритість до впровадження в інституції	Може використовуватись у закладах освіти, на підприємствах, у медичних або волонтерських програмах

Визначені фактори конкурентоспроможності є критично важливими для досягнення успіху веб-платформи у сфері ментального здоров'я. З огляду на динамічність технологічного середовища, необхідно здійснювати постійний моніторинг сильних та слабких сторін продукту, оперативно реагувати на зміни ринку та адаптувати функціонал відповідно до нових запитів аудиторії.

Спираючись на дані Таблиці 6.9, було здійснено оцінку рівня реалізації ключових характеристик продукту порівняно з конкурентами. Це дозволило визначити поточний стан проєкту, сформулювати стратегії посилення переваг і нейтралізації вразливостей. Результати наведено у Таблиці 6.10.

Таблиця 6.10 — Порівняльний аналіз сильних та слабких сторін проєкту

Фактор конкурентоспроможності	Оцінка (1-10)	Рейтинг
Функціональна відповідність	10	+3
Технологічна інноваційність	9	+2
Стабільність системи	6	0
Простота використання	8	+1
Доступність системи	10	+3

Проведена оцінка підтверджує, що проєкт демонструє високі показники за критеріями доступності, інноваційності та функціональності. Це свідчить про потенціал до успішного виходу на ринок, за умови подальшої технічної стабілізації та розширення маркетингових зусиль.

З метою цілісної оцінки стратегічного становища проєкту було здійснено SWOT-аналіз, який охоплює внутрішні сильні та слабкі сторони, а також зовнішні можливості та загрози. Результати наведено у таблиці 6.11.

SWOT-аналіз підтверджує, що проєкт має високий потенціал завдяки поєднанню зручності, технологічної інноваційності та широкого спектра функціональних можливостей. Простота у використанні й доступність для різних цільових груп створюють базу для залучення як індивідуальних користувачів, так і організаційних клієнтів.

Таблиця 6.11 — SWOT-аналіз стартап-проєкту

Сильні сторони (Strengths)	Слабкі сторони (Weaknesses)
Простий та інтуїтивно зрозумілий інтерфейс	Відсутність брендової впізнаваності
Підтримка мультимодульної архітектури (настрій, щоденник, рекомендації, аналітика, чат)	Залежність від зовнішніх API-провайдерів (зокрема Gemini AI)
Високий рівень персоналізації та гнучкості	Потреба в постійній технічній підтримці та вдосконаленні без комерційного фінансування
Адаптація під мобільні та десктоп-платформи	Обмежений ресурс на маркетинг і просування на стартовому етапі
Можливості (Opportunities)	Загрози (Threats)
Зростання попиту на цифрові сервіси у сфері ментального здоров'я	Посилення конкуренції з боку міжнародних застосунків
Можливість інтеграції у освітні й корпоративні середовища	Швидкі зміни технологій і очікувань користувачів
Отримання фінансування через акселератори або державні програми цифрової трансформації	Складнощі з юридичним позиціонуванням у сфері психологічної допомоги
Партнерство з НУО, університетами та медичними закладами	Потенційне зниження довіри до ML-рекомендацій при відсутності пояснюваності алгоритмів

Однак реалізація повного потенціалу вимагає подолання таких слабких сторін, як обмежені ресурси, потреба в побудові бренду, та технічні залежності. Особливої уваги потребують загрози зовнішнього середовища, пов'язані з динамікою IT-ринку, зміною очікувань користувачів та активністю конкурентів.

З урахуванням можливостей для зростання, зокрема інтеграції у B2B-сектор, участі у грантових програмах та співпраці з освітніми і медичними закладами, було

ухвалено рішення про запуск MVP-продукту та тестування його на локальному ринку з подальшим масштабуванням.

6.4 Розроблення ринкової стратегії проєкту

Формування ефективної ринкової стратегії для стартап-платформи підтримки ментального здоров'я потребує комплексного підходу, орієнтованого як на початкову адаптацію продукту до запитів цільової аудиторії, так і на довготривале утримання позицій у висококонкурентному середовищі. У рамках цього підходу ключовим завданням є ідентифікація пріоритетних сегментів користувачів і формування адаптивної маркетингової моделі на основі цифрової поведінки, соціального контексту й індивідуальних потреб.

Основна стратегія охоплення ринку полягає в поступовому розширенні охоплення через вибірковий доступ до найбільш зацікавлених груп, які вже сформували попит на цифрові сервіси психоемоційної самопідтримки. Такий підхід дозволяє на старті зосередитися на тих аудиторіях, для яких платформа забезпечує найвищу цінність і релевантність, що підвищує коефіцієнт залучення та знижує витрати на комунікацію.

Ключовим напрямом є формування партнерських моделей з університетами, освітніми платформами, студентськими організаціями, HR-відділами компаній та медичними установами. Саме синергія з цими структурами відкриває канали для масштабування, сприяє організаційному впровадженню системи, забезпечує сталість використання та доступ до релевантної цільової аудиторії.

Ринкова стратегія базується на концепції диференціації, що передбачає акцент на унікальних функціональних і технологічних характеристиках проєкту: інтеграція інструментів NLP, рекомендаційного модуля на основі TF-IDF і Cosine Similarity, наявність чат-помічника на базі AI, а також можливість персонального моніторингу психоемоційного стану. Усі ці елементи забезпечують створення унікальної ціннісної пропозиції, яка відрізняє платформу від наявних рішень на ринку.

Особливу увагу також приділено підтримці зворотного зв'язку: на ранньому етапі передбачено інтеграцію інструментів оцінювання якості рекомендацій, коментарів до навчального контенту та індикаторів ефективності вправ. Це дозволяє оперативно адаптувати продукт до змін у запитах користувачів та посилює рівень лояльності. Також здійснено ідентифікацію потенційних користувачів розроблюваної системи, деталізація яких представлена в таблиці 6.12.

Таблиця 6.12 — Вибір цільових груп потенційних споживачів

Опис цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах сегменту	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Студенти та молодь	Висока	Високий	Висока	Легко
Освітні заклади (школи, університети)	Середня	Високий	Середня	Середньо
HR-відділи компаній	Середня	Помірний	Середня	Середньо
Психологи, коучі, ментори	Висока	Помірний	Низька	Важко
Молоді батьки, які піклуються про добробут дітей	Середня	Зростаючий	Середня	Легко

Ефективна маркетингова стратегія є критичним фактором успішного виведення платформи на ринок у сфері EdTech та психоемоційного добробуту. Завдання маркетингового комплексу — не лише привернути увагу цільової аудиторії, а й сформувати довіру до цифрового продукту у делікатній сфері ментального здоров'я. Основні методи охоплення аудиторії наведено у таблиці 6.13.

Таблиця 6.13 — Маркетингова стратегія веб-платформи

Методи	Опис
Цифровий маркетинг	Контент-маркетинг (статті, блоги, відео), просування через соцмережі (Instagram, TikTok, Telegram), контекстна реклама
Участь у психологічних/освітніх заходах	Презентації продукту на конференціях, форумах, студентських зустрічах, заходах з психологічної підтримки
Демонстраційні сесії	Проведення безкоштовних ознайомчих вебінарів, демо-доступ до платформи для шкіл, університетів і компаній
Реферальна система	Стимулювання запрошень через бонуси або преміальні функції для нових користувачів
Email-кампанії	Надсилання матеріалів про користь платформи, оновлення функцій, поради для підтримки психоемоційного стану
PR-стратегія	Публікації в медіа, залучення експертів з ментального здоров’я, партнерські статті

Для забезпечення життєздатності веб-платформи критично важливими є також компоненти підтримки користувачів та організаційна стратегія продажів. Таблиця 6.14 деталізує механізми реалізації продукту, способи взаємодії з користувачами, а також канали підтримки, які підвищують лояльність та сприяють зростанню активної аудиторії.

Впровадження комплексної стратегії продажів і підтримки користувачів є критично важливим елементом стабільного функціонування та зростання платформи. Такий підхід дозволяє не лише залучити нових користувачів, а й утримати наявну базу за рахунок якісної комунікації, адаптації продукту до потреб цільових груп і швидкого реагування на зміну ринкових запитів.

Таблиця 6.15 — Стратегія реалізації та підтримки користувачів

Компонент	Опис
Пряма реалізація	Запуск MVP для вільного користування з опцією freemium, преміум-функції — аналітика, персоналізовані поради
Партнерські програми	Співпраця з університетами, HR-відділами компаній, психологами, залучення платформи як інструменту самодопомоги
Технічна підтримка	Онлайн-чат, FAQ, email-підтримка, регулярні оновлення функцій та виправлення помилок
Зворотний зв’язок	Інтегровані форми для оцінювання порад, інтерфейсу, впливу контенту, запити на функціональність
Адаптивність системи	Розвиток функцій на основі отриманих даних, підтримка різних мов, кастомізація інтерфейсу

Регулярний аналіз поведінкових патернів, індикаторів задоволеності та активної участі користувачів дозволить постійно вдосконалювати продукт. Своєчасна адаптація та побудова довгострокових відносин із партнерами формують основу для стійкого зростання, відкриваючи можливості масштабування проєкту на нові ринкові сегменти.

Висновки до розділу 6

У шостому розділі здійснено комплексний аналіз стартап-проєкту, спрямованого на впровадження веб-платформи підтримки ментального здоров’я. Розглянуто ринкову готовність продукту, його потенційну цінність для цільових аудиторій та перспективи масштабування. Підтверджено, що концепція платформи відповідає сучасним потребам користувачів і має високий потенціал комерціалізації.

У межах технологічного аудиту встановлено доцільність використання обраного стеку технологій — Nest.js, PostgreSQL, React, AI-модуля на основі

Gemini. Це забезпечує технічну зрілість системи, її безпеку, продуктивність та можливість інтеграції з зовнішніми сервісами. Аналіз ринкових можливостей показав зростаючий попит на цифрові рішення.

SWOT-аналіз виявив сильні сторони стартапу, зокрема персоналізований функціонал, інноваційність, мультифункціональність і зручний інтерфейс. Разом із тим зафіксовано потребу у зміцненні технічної підтримки, підвищенні впізнаваності бренду та розширенні ресурсної бази. На основі ринкової стратегії запропоновано модель диференціації та інструменти маркетингового просування через цифрові канали, партнерські програми та участь у тематичних подіях.

Узагальнюючи результати розділу, можна стверджувати, що розроблений стартап-проєкт має стратегічно зважене підґрунтя, відповідає актуальним ринковим потребам і володіє потенціалом успішної реалізації за умови подальшої адаптації до динаміки ринку.

ВИСНОВКИ

У процесі виконання магістерської роботи було реалізовано комплекс теоретичних і практичних завдань, спрямованих на дослідження сучасних методів веб-розробки, аналізу даних та застосування алгоритмів машинного навчання у контексті ментального здоров'я. Основною метою дослідження було створення сучасної, стабільної та зручної веб-платформи підтримки психоемоційного стану користувачів, яка поєднує можливості самоспостереження, аналітики та формування персоналізованих рекомендацій. У межах роботи ця мета була повністю досягнута.

На початковому етапі було здійснено огляд існуючих цифрових рішень у галузі ментального добробуту та психоемоційного моніторингу. Аналіз показав, що більшість існуючих платформ не використовують персоналізовані підходи, базовані на поведінкових та текстових даних користувачів, що суттєво знижує ефективність таких сервісів. На основі цього було сформульовано вимоги до функціоналу майбутньої системи: реєстрація та авторизація користувачів, можливість фіксації та аналізу настрою, ведення особистого щоденника, перегляд аналітики, доступ до навчального контенту та отримання рекомендацій, адаптованих до реального стану користувача.

У ході проєктування та розробки було використано сучасний та технологічно збалансований стек: серверна частина побудована на Nest.js, що забезпечило модульність, структурованість та зручність у масштабуванні; взаємодія з реляційною базою даних PostgreSQL реалізована через ORM Sequelize; клієнтська частина створена на основі Next.js, що надало переваги гібридного рендерингу та гнучкої маршрутизації; стилізація здійснена за допомогою Tailwind CSS, а управління станом даних — через React Query. Такий вибір інструментів дозволив створити продуктивну, безпечну та зручну у використанні систему.

Особливу увагу було приділено реалізації рекомендаційного модуля. Для аналізу текстів та формування релевантних порад застосовано методи TF-IDF та косинусної подібності, що забезпечують визначення схожості між текстами

користувача та навчальними матеріалами. Попередня обробка тексту виконувалась за допомогою алгоритмів NLP, що дозволило покращити точність рекомендацій. Завдяки такому підходу система адаптується до індивідуальної історії кожного користувача та формує поради, які відповідають його актуальному психоемоційному стану.

Було реалізовано REST API з чітким розмежуванням маршрутів, механізмами автентифікації на основі JWT, інтегрованою документацією через Swagger та структурованою серверною логікою, що охоплює модулі авторизації, користувача, настрою, щоденника, рекомендацій, навчального контенту та чату. Клієнтська частина містить інтуїтивні інтерфейси для взаємодії зі всіма модулями платформи: створення записів настрою, ведення текстового щоденника, проходження вправ і курсів, перегляд аналітики та ведення діалогу з чат-ботом.

Створена платформа дозволяє користувачеві краще розуміти власний психоемоційний стан, відстежувати динаміку змін, отримувати рекомендації, що ґрунтуються на інтелектуальному аналізі його записів, та застосовувати техніки самодопомоги у зручному цифровому форматі. Інтеграція елементів машинного навчання та NLP робить систему адаптивною, а використання сучасних веб-технологій забезпечує високу продуктивність, безпеку та стабільність роботи.

У результаті виконаної роботи було створено повнофункціональний прототип веб-платформи, який може бути використаний як основа для подальших досліджень і впроваджень у сфері цифрової підтримки ментального здоров’я.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Patel V., Saxena S., Lund C. et al. The Lancet Commission on global mental health and sustainable development. *Lancet*. 2018. Vol. 392, No. 10157. pp. 1553–1598.

2. Ткачишина О.Р. Проблема ментального здоров'я в Україні: психологічний аналіз. *Habitus*. 2023. Вип. 53. С. 207–211.

3. Lecomte T., Potvin S., Corbière M. et al. Mobile apps for mental health issues: meta-review of meta-analyses. *JMIR mHealth and uHealth*. 2020. Vol. 8, No. 5. e17458.

4. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge : Cambridge University Press, 2008. 482 p.

5. World Health Organization. World mental health report: Transforming mental health for all. *Geneva: World Health Organization*, 2022. 110 с. URL: https://www.who.int/publications/i/item/9789240049338 (date of access: 02.12.2025).

6. Olawade D.B., Wada Z.W., Odetayo A.O. et al. Enhancing mental health with Artificial Intelligence: Current trends and future prospects. 2024. URL: https://www.sciencedirect.com/science/article/pii/S2949916X24000525 (date of access: 02.12.2025).

7. Ali M. et al. Artificial intelligence for mental health: A narrative review of sentiment analysis and mood detection techniques. 2025. URL: https://journals.sagepub.com/doi/10.1177/20552076251395548 (date of access: 02.12.2025).

8. Ni Y. et al. A Scoping Review of AI-Driven Digital Interventions in Mental Health. 2025. URL: https://www.mdpi.com/2227-9032/13/10/1205 (date of access: 02.12.2025).

9. Jones A. Mastering TypeScript Programming: An In-Depth Exploration of Essential Concepts. 2025. С. 11–13.

10. Rambert D. Advanced Next.js for Everyone: Build Scalable, Fast, and Future-Proof Web Applications with Next.js 15. 2025. С. 8–17.

11. Linjanja P. Scalable Application Development with NestJS: Leverage REST, GraphQL, microservices, testing, and deployment for seamless growth. 2025. С. 5–8.

12. Rauschmayer A. The Benefits of Using TypeScript. Tackling TypeScript. 2020. С. 13–16.

13. Gibbok. TypeScript Introduction — Modern JavaScript Now (Down-leveling). *Github*. URL: https://gibbok.github.io/typescript-book/book/typescript-introduction/ (date of access: 02.12.2025).

14. Rozentals N. Mastering TypeScript: Build enterprise-ready, modular web applications using TypeScript 4 and modern frameworks. 2021. С. 197–233.

15. Riva M. Exploring Different Rendering Strategies. Real-World Next.js. Packt Publishing, 2022. pp. 25–42.

16. Riva M. Next.js Basics and Built-In Components. Real-World Next.js. Packt Publishing, 2022. pp. 33–43.

17. Linjanja P. Setting Up Your NestJS Environment and Exploring NestJS – Building a Robust App. Scalable Application Development with NestJS. Packt Publishing, 2025. pp. 31–50.

18. Ponel J., Rosenstock L. Describing APIs. Designing APIs with Swagger and OpenAPI. Manning, 2021. pp. 1–64.

19. Rappin N. Flexible Styling Without the Fuss. Modern CSS with Tailwind. Pragmatic Programmers, 2022. pp. 40–102.

20. Durante D. Introduction to Sequelize and ORM in Node.js & Defining and Using Sequelize Models. Supercharging Node.js Applications with Sequelize. 2022. pp. 7–12.

21. Durante D. Managing Database Schema with Migrations and Transactions. Supercharging Node.js Applications with Sequelize. 2022. pp. 46–78.

22. Ferrari L., Pirozzi E. Learn PostgreSQL — Use, manage and build secure and scalable databases with PostgreSQL 16. 2nd ed. 2023.

23. Lanham M. Engaging GPT assistants. AI Agents in Action. Manning, 2025. pp. 45–78.

24. Хороших О.Л., Сегеда І.Р. Web-платформа підтримки ментального здоров'я з інтеграцією технологій машинного. *XIII Міжнародна науково-практична конференція «Modern digital technologies and problems of their use».*

Технічні науки. Прага, Чехія, 24 листопада 2025 р. С. 246–250.

25. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge : Cambridge University Press, 2008. pp. 117–121.

26. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge : Cambridge University Press, 2008. pp. 111–112.

27. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge : Cambridge University Press, 2008. pp. 120–123.

28. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. 2nd ed. Sebastopol : O'Reilly Media, 2019. pp. 541–548.

29. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O'Reilly Media, 2007. pp. 5–15.

30. Model-View-Controller. *NestJS Documentation*. URL: https://docs.nestjs.com /techniques/mvc (date of access: 02.12.2025).

31. Chat-based AI mental health support. *Woebot Health*. URL: https://woebothealth.com (date of access: 02.12.2025).

32. Journal, Diary and Mood Tracker. *Daylio*. URL: https://daylio.net (date of access: 02.12.2025).

33. Mental Health App for Meditation & Sleep. *Headspace*. URL: https://www.headspace.com (date of access: 02.12.2025).

ДОДАТОК А

АПРОБАЦІЯ

XIII Міжнародна науково-практична конференція «Modern digital technologies and problems of their use» Технічні науки, 24 листопада, Прага, Чехія.
Аркушів 8

Київ — 2025


EUROPEAN CONFERENCE
Conference Proceedings
XIII International Science Conference
«Modern digital technologies and problems of their use»
November 24-26, 2025
Prague, Czech Republic

WEB-ПЛАТФОРМА ПІДТРИМКИ МЕНТАЛЬНОГО ЗДОРОВ'Я З ІНТЕГРАЦІЄЮ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ

Хороших Олександр
Магістр
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Сегеда І.В
Доцент, к.е.н., доцент
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Проблема психічного здоров'я набула глобальних масштабів, мільйони людей зі всього світу потребують психологічної підтримки. Попит на якісні послуги ментального здоров'я на сьогодні час дуже високий, проте традиційні засоби допомоги часто залишаються недостатніми та малодоступними [1].

Проаналізувавши дане питання, було вирішено розробити веб-платформу для підтримки ментального здоров'я, яка інтегрує технології машинного навчання. Основна ціль додатку забезпечити користувачам персоналізовані рекомендації та інструменти самодопомоги в будь-який необхідний момент, тобто у режимі 24/7. Веб-платформа об'єднує кілька ключових функцій: відстеження настрою користувача, ведення особистого щоденника, генерацію рекомендацій з використанням ШІ, аналітику психоемоційного стану та інтерактивний чат для підтримки.

Необхідно зазначити, що на фоні зростання кількості психоемоційних розладів та стресу останніми роками підвищилася суспільна увага до ментального здоров'я, але нові цифрові рішення можуть частково заповнити прогалину між потребою в допомозі та її доступністю. Типовим недоліком більшості таких платформ є підхід, коли всім користувачам пропонується однаковий контент без урахування індивідуальних особливостей людини. Інтеграція машинного навчання, зокрема методів обробки текстів і даних користувача, надає можливість створити персоналізований досвід взаємодії [2, 3].

Запропонована веб-платформа забезпечує персоналізацію контенту спираючись на алгоритми машинного навчання: кожному користувачу надаються рекомендації, які необхідні та релевантні саме для його стану та потреб. Вона також поєднує певну сукупність функцій (відстеження настрою,

щоденник, рекомендації курсів та вправ, аналітика, чат), створюючи єдину екосистему турботи про себе.

На відміну від більшості рішень, веб-додаток не змушує користувача постійно щось обирати: замість великої кількості матеріалів вона пропонує декілька найбільш доречних рекомендацій, у вигляді вправ та курсів. Також платформа забезпечує інтерактивність і підтримку в реальному часі через чат-інтерфейс, а модуль аналітики надає змогу користувачу відстежувати динаміку свого стану, у вигляді показових графіків.

Розглянуто основні категорії конкурентів: трекери настрою (Daylio) [4], чат-боти для терапії (Woebot) [5], застосунки для медитації (Headspace) [6], форуми підтримки (7 Cups) [7]. Всі ці сервіси мають певні обмеження: вони або не персоналізують контент, або не інтегрують кілька функцій одночасно. Запропонована платформа поєднує функціонал трекера, щоденника, бота і аналітичного помічника в одному продукті. Вона навчається на даних користувача та генерує релевантні рекомендації, чого не забезпечують конкуренти.

Веб-додаток складається з трьох рівнів: клієнт (Next.js), сервер (Nest.js) і база даних (PostgreSQL). Клієнт відповідає за інтерфейс, сервер обробляє запити, зберігає і обробляє дані. Основні модулі: аутентифікація, щоденник, настрій, рекомендації, чат, аналітика. Користувач взаємодіє через інтерфейс, надсилаючи інформацію про свій стан, дані оброблюються на сервері, який взаємодіє з БД, обробляє їх і повертає відповіді у вигляді рекомендацій або візуалізацій. Приклад алгоритму роботи модулю «Chat», після отримання запиту з повідомленням від користувача зображено на рисунку 1.

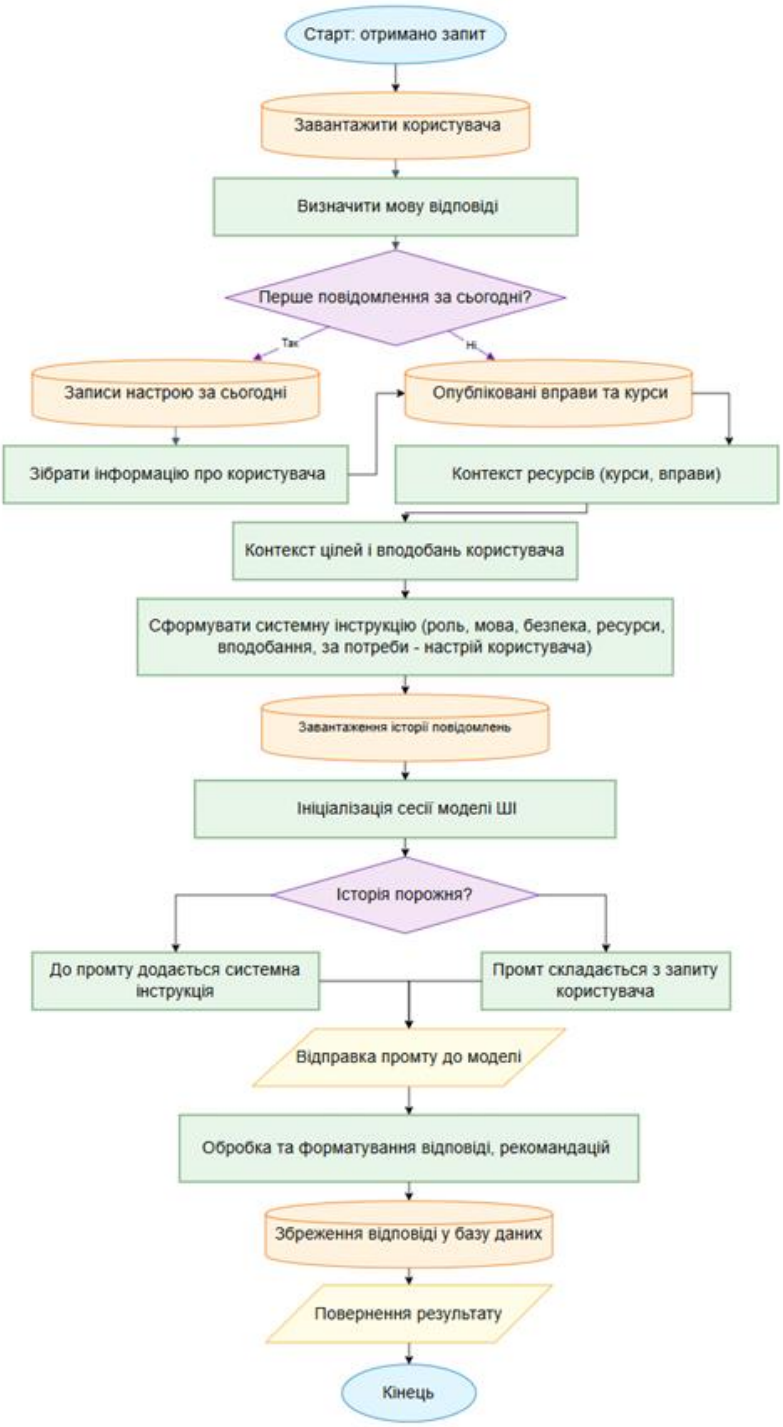

Рисунок 2. Алгоритм роботи модулю «Chat»

Джерело: власна розробка автора

Ядро аналітичного модую - алгоритм рекомендацій на базі TF-IDF і косинусної подібності. Текст та інформація користувача обробляється (токенізація, видалення стоп-слів, стемінг), обчислюється TF-IDF вектор, порівнюється з векторами ресурсів у базі. За значенням подібності обирається перелік релевантих рекомендацій.

Реалізовані компоненти: графік настрою з часом (лінійний), індикатори виконаних рекомендацій та вправ. Для візуалізацій використовуються бібліотеки Chart.js. Аналітичний модуль обчислює середні значення, тренди, зв'язки між станами і активністю користувача, дані передаються через API і виводяться у вигляді графіків. На рисунку 2 представлено сторінка з рекомендаціями, які збираються на основі дій користувача.

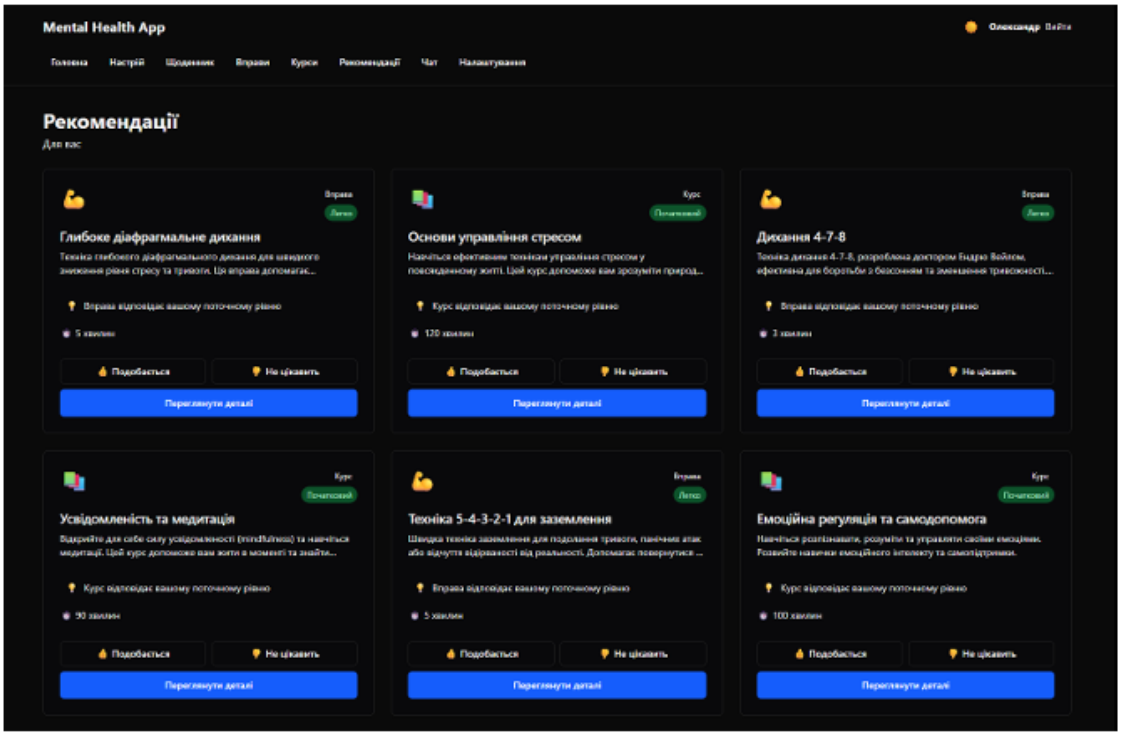

Рисунок 2. Сторінка рекомендацій

Джерело: власна розробка автора

Технології реалізації Next.js (React) - фронтенд фреймворк; Nest.js - серверна логіка; PostgreSQL зберігання даних; Sequelize - ORM; Tailwind CSS - стилізація; Chart.js - графіки; TF-IDF, cosine similarity - основа ML-рекомендацій.

Користувач проходить онбординг, вказує свої цілі. Створює запис у щоденнику: "Сьогодні відчуваю тривогу через роботу". Система обробляє текст, виділяє ключові слова ("тривога", "робота"), обчислює подібність із базою порад і пропонує релевантні ресурси: дихальні вправи, статтю про стрес на роботі тощо.

Через кілька днів користувач бачить графік настрою з позначками рекомендацій, графік стану, аналітику найчастіших проблем. Це сприяє саморефлексії і мотивації.

Платформа забезпечує персоналізовану підтримку ментального здоров'я, поєднуючи машинне навчання, аналітику та інтерактивний інтерфейс. Вона перевершує існуючі рішення завдяки комплексному підходу, інтелектуальним рекомендаціям, а використання сучасного технологічного стеку забезпечує масштабованість і ефективність.

Список літератури

1. World Health Organization. World mental health report: Transforming mental health for all. Geneva : WHO, 2022. 110 c.

2. Valentine L., D'Alfonso S., Lederman R. Recommender systems for mental health apps: advantages and ethical challenges. AI & Society. 2023. T. 38. C. 1627–1638.

3. Chaturvedi A., D'Alfonso S., Lattie E., Mohr D. C. Content Recommendation Systems in Web-Based Mental Health Care: Real-world Application and Formative Evaluation. JMIR Formative Research. 2023. T. 7. C. e38831.

4. Daylio – Mood Tracker & Journal. 2025. [Електронний ресурс]. Режим доступу: https://daylio.net/ (Дата звернення: 17.11.2025).

5. Woebot Health. 2025. [Електронний ресурс]. Режим доступу: https://woebothealth.com/ (Дата звернення: 17.11.2025).

6. Headspace. Mindfulness and Meditation. 2025. [Електронний ресурс]. Режим доступу: https://www.headspace.com/ (Дата звернення: 17.11.2025).

7. 7 Cups: Online Therapy & Free Counseling. 2025. [Електронний ресурс]. Режим доступу: https://www.7cups.com/ (Дата звернення: 17.11.2025).