Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для онлайн-тренінгу ментального здоров'я. DevOps, Back-end
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-5

Дедюкін О. К.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Побіженко І. О.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Дедюкіну Олексію Костянтиновичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для онлайн-тренінгу ментального здоров’я. DevOps, Back-end
Затверджена наказом по університету від 20.05. 2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 17.06.2024
3. Вихідні дані до роботи Розробити програмну систему, яка забезпечує онлайн-тренінги та курси з питань ментального здоров'я, використовуючи фреймворк Django, база даних PostgreSQL, та Rest API.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.05.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	28.05.2024	*виконано*
5	Тестування ПЗ	30.05.2024	*виконано*
6	Оформлення пояснювальної записки	05.06.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Попередній захист	09.06.2024	*виконано*
9	Нормоконтроль, рецензування	06.06.2024	*виконано*
10	Здача роботи у електронний архів	10.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	11.06.2024	*виконано*

Дата видачі завдання 16 травня 2024р.

Студент (ка)_____ Дедюкін О. К.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Побіженко І. О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 92 стор., 12 рис., 10 джерел.

АРХІТЕКТУРА ПРОГРАМНИХ СИСТЕМ, БЕЗПЕКА ДАНИХ, ВЕБ-ДОДАТОК, МОНЕТИЗАЦІЯ, ОНЛАЙН ТРЕНІНГИ, ПСИХОЛОГІЧНЕ ЗДОРОВ’Я, DOCKER, PYTHON, PYCHARM, REST API, РОЗРОБКА ПЗ, ФУНКЦІОНАЛЬНІ ВИМОГИ, ЦІЛЬОВА АУДИТОРІЯ

Об'єкт розробки – програмна система для онлайн-тренінгу ментального здоров'я.

Мета розробки – створення серверної частини програмної системи, яка забезпечує онлайн-тренінги та курси з питань ментального здоров'я.

Метод рішення – середовище розробки PyCharm, фреймворк Django, база даних PostgreSQL, Rest API.

У результаті розробки створено програмну систему, яка забезпечує онлайн-курси та тренінги з питань ментального здоров'я, з інтеграцією календаря для запису до ментора.

SOFTWARE SYSTEM ARCHITECTURE, DATA SECURITY, WEB APPLICATION, MONETIZATION, ONLINE TRAINING, MENTAL HEALTH, DOCKER, PYTHON, PYCHARM, REST API, SOFTWARE DEVELOPMENT, FUNCTIONAL REQUIREMENTS, TARGET AUDIENCE

The object of development is a software system for online training in mental health.

The purpose of the development is to create a server-side software system that provides online training and courses on mental health issues.

Solution method - PyCharm development environment, Django framework, PostgreSQL database, Rest API.

As a result of the development, a software system was created that provides online courses and trainings on mental health issues, with the integration of a calendar for making an appointment with a mentor.

Я, Дедюкін Олексій Костянтинович, студент гр. ПЗПІ-20-5, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для онлайн-тренінгу ментального здоров'я. Dev-op, Back-end», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

CI/CD – Continuous Integration/Continuous Deployment

CSRF – Cross-Site Request Forgery

Docker – платформа для контейнеризації додатків

FTP – File Transfer Protocol

JWT – JSON Web Token

ORM – Object-Relational Mapping

PaaS – Platform as a Service

REST – Representational State Transfer

SQL – Structured Query Language

XSS – Cross-Site Scripting

ВСТУП

Питання ментального здоров’я в сучасному світі набуло особливого значення, оскільки дедалі більше людей стикаються з труднощами в подоланні стресу, емоційного виснаження та нестабільного психічного стану. У той час, як доступність фахової психологічної допомоги залишається обмеженою через географічні та фінансові фактори, онлайн-платформи набувають все більшого значення як засіб для отримання якісних ресурсів та підтримки. Сучасне суспільство потребує програмної системи, яка допоможе людям в будь-якому місці та в зручний час покращувати своє психічне благополуччя, зміцнювати навички саморегуляції та отримувати доступ до цінних тренінгів та порад менторів.

Онлайн-тренінги з ментального здоров’я можуть вирішити низку проблем, які виникають при звичайному форматі терапії. Користувачам необхідна можливість взаємодіяти з ментором в зручний для них час і отримувати індивідуальні рекомендації щодо того, як покращити своє психічне здоров'я. Крім того, відстеження особистого прогресу та створення індивідуальних планів дозволяє систематично працювати над своїм станом, орієнтуючись на конкретні результати. Особливо важливою є можливість планування та запису до ментора в календарі, що підвищує ефективність тренінгів і дозволяє користувачам краще керувати своїм часом. Доступ до ресурсів для самостійного навчання сприяє систематичному підходу до покращення ментального здоров'я.

Головною метою цього дослідження є створення програмної системи для онлайн-тренінгу ментального здоров’я, яка надаватиме користувачам можливість долати труднощі самостійно або за підтримки ментора. Ця система має допомогти людям впоратися зі стресом та іншими емоційними викликами, дозволяючи їм поліпшувати своє психологічне благополуччя у зручний спосіб. Щоб досягти цієї мети, слід детально вивчити доступні освітні ресурси, проаналізувати різноманітні підходи до створення індивідуальних планів і знайти оптимальний спосіб реалізації ефективного та зрозумілого інтерфейсу. Потрібно також розробити систему відстеження прогресу, яка допоможе користувачам отримувати об'єктивну оцінку власного розвитку.

Зрештою, результатом має стати програмна система, яка об'єднає всі необхідні інструменти для онлайн-тренінгу ментального здоров’я. Ця система допоможе користувачам самостійно планувати свої заняття, отримувати зворотній зв'язок від менторів та відстежувати прогрес у покращенні психічного стану.

Для серверної частини буде використано Python, фреймворк Django та базу даних PostgreSQL, що забезпечить надійну та гнучку взаємодію з даними. Інтерфейс системи буде розроблений за допомогою сучасних технологій Material-UI та React, щоб надати користувачам простий та інтуїтивно зрозумілий досвід. Це дозволить створити функціональний та доступний продукт, здатний забезпечити всебічну підтримку користувачам на шляху до покращення ментального здоров'я.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної галузі та постановка задачі

Ментальне здоров'я є однією з найважливіших складових загального добробуту людини, яка впливає на якість життя, продуктивність та соціальну інтеграцію. В сучасному світі проблеми ментального здоров'я стали ще більш актуальними через швидкий темп життя, постійний стрес, економічні нестабільності та соціальні виклики. За даними Всесвітньої організації охорони здоров'я (ВООЗ), близько 450 мільйонів людей у світі страждають від психічних або поведінкових розладів, що складає значну частину глобального населення.

Останні події такі як пандемія COVID-19 та ланка військових конфліктів по всьому світу ще більше загострила питання ментального здоров'я. Люди переживають підвищений рівень тривожності, депресії та інших психічних розладів через ізоляцію, невизначеність, втрату близьких та фінансові труднощі. В умовах постійної соціальної дистанції та обмеженого доступу до традиційних медичних послуг виникла потреба в нових формах підтримки ментального здоров'я.

Онлайн-тренінги ментального здоров'я представляють собою сучасний підхід до вирішення цих проблем. Вони дозволяють забезпечити безперервний доступ до психологічної підтримки та навчання незалежно від місця перебування користувача. Такі системи можуть включати різноманітні методики та техніки, такі як когнітивно-поведінкова терапія, медитація, техніки релаксації та інші. Важливо зазначити, що онлайн-тренінги можуть бути як самостійним інструментом, так і доповненням до традиційних форм терапії.

На сьогоднішній день існує кілька проблем, пов'язаних з використанням онлайн-тренінгів для ментального здоров'я. По-перше, не всі існуючі програми є достатньо ефективними та науково обґрунтованими. Деякі з них можуть містити неякісний контент або використовувати методики, які не мають доказової бази. Це може призводити до недостатньої ефективності або навіть погіршення стану користувачів.

По-друге, багато програм мають обмежений функціонал і не можуть забезпечити індивідуальний підхід до кожного користувача. Це особливо важливо, оскільки кожна людина має свої унікальні потреби та проблеми, які потребують індивідуального підходу та адаптації методик.

По-третє, доступ до онлайн-тренінгів може бути обмежений через фінансові бар'єри або технічні труднощі. Не всі люди мають доступ до інтернету або можуть дозволити собі платні програми, що створює додаткові перешкоди на шляху до отримання необхідної допомоги.

Зважаючи на вище зазначені проблеми, розробка нової програмної системи для онлайн-тренінгу ментального здоров'я є надзвичайно актуальною. Така система повинна бути науково обґрунтованою, ефективною, доступною та здатною забезпечити індивідуальний підхід до кожного користувача. Важливо також забезпечити простий та зручний інтерфейс, щоб користувачі різних вікових категорій та з різним рівнем технічної підготовки могли легко користуватися програмою.

У підсумку, актуальність теми розробки програмної системи для онлайн-тренінгу ментального здоров'я обумовлена зростаючою потребою у забезпеченні доступної та якісної психологічної підтримки у сучасному світі. Така система може значно покращити якість життя людей, допомогти їм впоратися зі стресом, тривожністю та іншими психічними розладами, а також сприяти загальному добробуту суспільства.

1.2 Цільова аудиторія

Основна цільова аудиторія складається з наступних груп:

а) люди різного віку та соціального статусу, які мають на меті підвищити свою стійкість до стресу, навчитися методів саморегуляції та покращити емоційне благополуччя. За даними ВООЗ, стрес є однією з найбільш поширених причин проблем зі здоров'ям, і близько 75% дорослих людей у світі відчувають стрес у своєму житті. Ця група може включати студентів, які стикаються з навчальними навантаженнями, працівників,

які переживають професійний стрес, та літніх людей, які хочуть зберегти своє ментальне здоров'я. Вони можуть бути зацікавлені в індивідуальних тренінгах з ментором або в самостійних заняттях, використовуючи мобільні додатки чи онлайн-платформи;

б) соціальні та політичні виклики, такі як економічні кризи, соціальна нестабільність, або інші кризові ситуації, можуть викликати тривогу, стрес та пригнічення. Люди, які переживають складні періоди, прагнуть знайти професійну підтримку для полегшення психологічного стану. Згідно з даними American Psychological Association (APA), у періоди соціальних і економічних криз рівень тривожності та депресії значно зростає. Онлайн-тренінги можуть стати важливим інструментом для таких людей, забезпечуючи доступ до методів самодопомоги та професійної підтримки в будь-який час;

в) психологи, психотерапевти та інші професіонали в цій галузі можуть використовувати систему як інструмент для моніторингу прогресу клієнтів і планування індивідуальних програм роботи. Система може включати функції для відстеження психоемоційного стану клієнтів, аналізу даних та адаптації програм під конкретні потреби кожного клієнта. Згідно з дослідженням, опублікованим у Journal of Medical Internet Research, використання цифрових інструментів у психотерапії може значно підвищити ефективність терапевтичного процесу;

г) компанії та некомерційні організації можуть пропонувати доступ до тренінгів як частину програм корпоративного здоров'я для своїх працівників або членів спільнот. За даними Global Wellness Institute, інвестиції в програми ментального здоров'я на робочих місцях можуть знизити рівень вигоряння серед працівників на 30%, а також підвищити продуктивність праці. Організації можуть використовувати такі системи для проведення регулярних тренінгів, оцінки психоемоційного стану співробітників та розробки стратегій для поліпшення робочого клімату.

Система повинна відповідати потребам кожної групи, надаючи користувачам зручний інтерфейс та інструменти для ефективного навчання, підтримки та планування сесій. Важливо забезпечити адаптивність системи, що дозволить користувачам отримувати персоналізовані рекомендації та доступ до необхідних ресурсів у зручний для них час.

1.3 Огляд систем аналогів

На сучасному ринку онлайн-платформ для ментального здоров'я існують кілька напрямів, у яких активно розвиваються різні сервіси. По-перше, це мобільні додатки для самодопомоги, такі як Calm і Headspace, які забезпечують користувачам інструменти для відстеження настрою, виконання медитаційних вправ, ведення журналів емоційного самопочуття та інших подібних функцій. По-друге, є платформи для відеоконсультацій, на кшталт Talkspace і BetterHelp, які дозволяють користувачам зв'язуватися з ліцензованими психологами та терапевтами для особистих консультацій через відео. Третім напрямом є освітні веб-сайти, такі як Verywell Mind, що надають інформаційні ресурси, статті, відеоуроки та курси. Ці матеріали допомагають людям отримати більше знань про управління стресом, тривожність, депресію та інші психологічні стани. І, нарешті, існують інтерактивні терапевтичні інструменти, які використовують ігрові механіки та інтерактивні завдання для надання терапевтичної допомоги. Як приклад, Woebot пропонує підтримку у вигляді розмовного робота, який дає поради, ґрунтуючись на когнітивно-поведінковій терапії.

Headspace є популярним мобільним додатком, що спеціалізується на медитації та ментальному здоров'ї. Він пропонує користувачам структуровані курси медитації, які допомагають навчитися основним технікам саморегуляції та управління стресом (див. рис. 1.1).

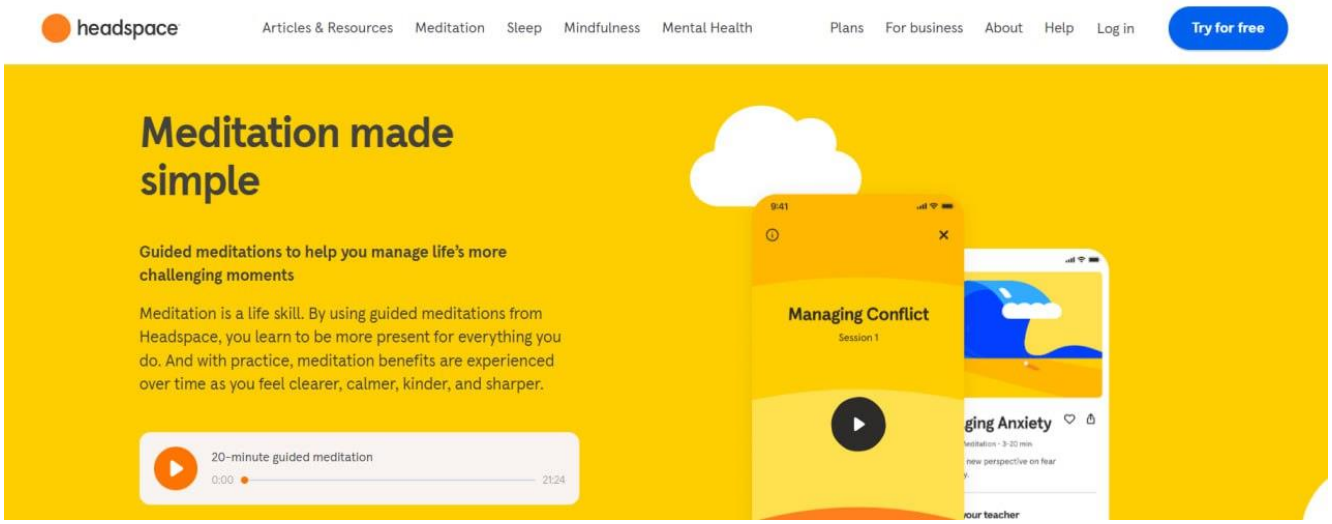


Рисунок 1.1 – Сторінка Headspace (за даними [1])

Переваги:

- чітко структуровані курси медитації для різних рівнів підготовки;
- великий вибір тематичних медитацій (від стресу, тривожності, сну тощо);
- доступ до додаткових ресурсів, таких як відеоуроки та статті;
- інтерактивні вправи та трекери прогресу.

Недоліки:

- висока вартість підписки;
- більшість контенту доступна лише в преміум-версії;
- неефективний для серйозних ментальних проблем.

BetterHelp є однією з найбільших онлайн-платформ для відеоконсультацій з ліцензованими психологами та терапевтами. Платформа надає можливість отримувати консультації через відео, текстові повідомлення та телефонні дзвінки (див. рис. 1.2).

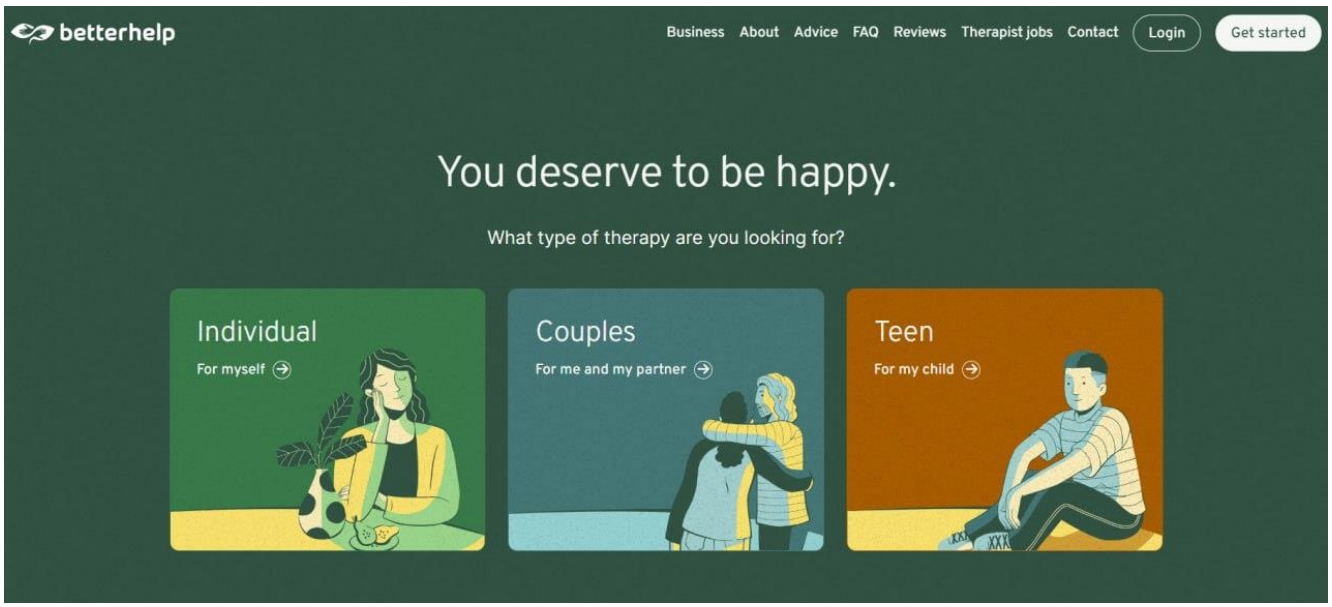


Рисунок 1.2 – Сторінка BetterHelp (за даними [2])

Переваги:

- широкий вибір спеціалістів у різних галузях психічного здоров'я;
- можливість змінити терапевта в будь-який момент;
- гнучкий графік консультацій, зручний для користувачів;
- високий рівень конфіденційності та безпеки даних.

Недоліки:

- висока вартість підписки;
- труднощі з налагодженням емоційного зв'язку через онлайн-формат;
- обмеження деяких послуг для користувачів з певних регіонів.

Verywell Mind є одним з найбільш популярних освітніх веб-сайтів, що надає інформаційні ресурси, статті, відеоуроки та курси з тематики ментального здоров'я. Сайт охоплює широкий спектр питань, таких як управління стресом, тривожність, депресія та інші психологічні стани (див. рис. 1.3).

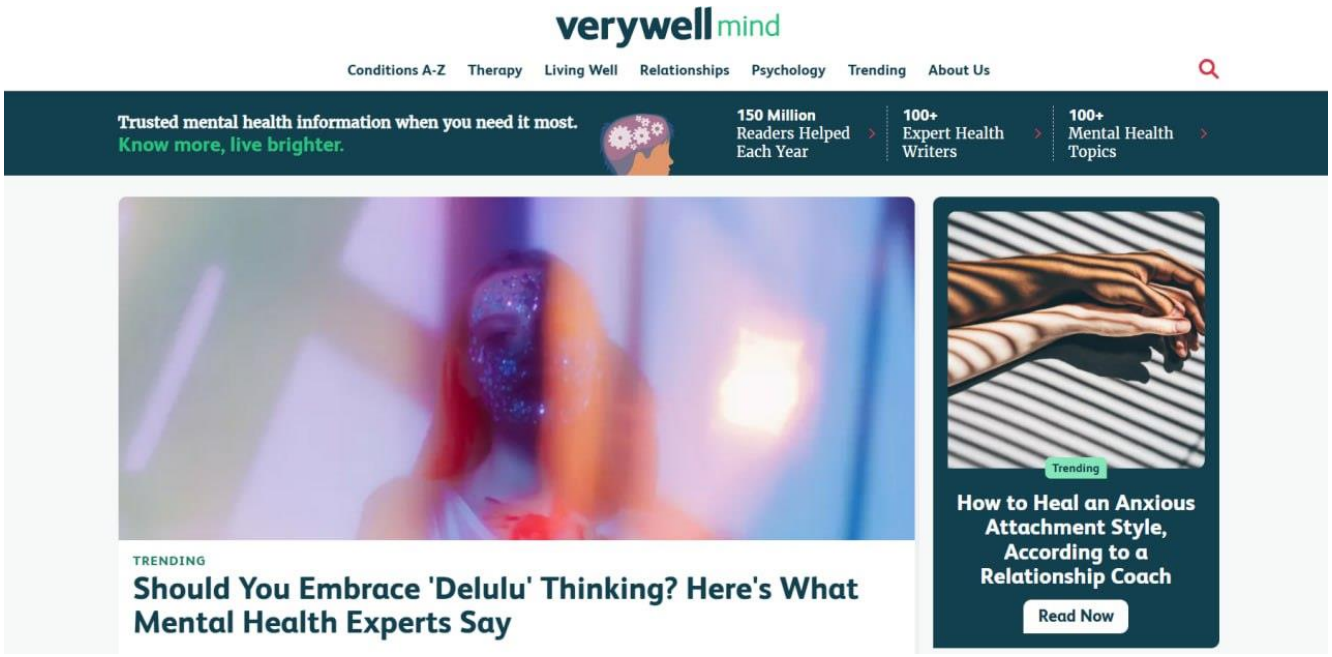


Рисунок 1.3 – Сторінка Verywell Mind (за даними [3])

Переваги:

- велика кількість якісних статей та ресурсів на різні теми;
- доступ до інформаційних матеріалів від експертів у галузі психології;
- безкоштовний доступ до більшості контенту;
- регулярне оновлення та доповнення матеріалів.

Недоліки:

- відсутність інтерактивних елементів та індивідуального підходу;
- може бути складно знайти інформацію через великий обсяг матеріалів;
- відсутність можливості безпосереднього зв'язку з фахівцем.

Враховуючи ці переваги і недоліки наведених аналогів, розроблена програмна система повинна поєднувати найкращі риси цих рішень, забезпечуючи користувачам більш комплексний та адаптивний підхід до покращення ментального здоров'я. Основні аспекти, які будуть враховані під час розробки, включають інтерактивність і персоналізацію. Система буде включати інтерактивні вправи та трекери прогресу, подібно до Headspace.

Ще один важливий аспект – це доступність і гнучкість. Враховуючи переваги BetterHelp, розроблена платформа забезпечить доступ менторів через

відеоконсультації та текстові повідомлення. Це дозволить користувачам вибирати зручний формат спілкування та отримувати допомогу в будь-який час.

Освітні ресурси також відіграватимуть значну роль. Відповідно до моделі Verywell Mind, розроблена система повинна надавати широкий спектр освітніх матеріалів, таких як статті, відеоуроки та курси, що допоможуть користувачам отримати необхідні знання та навички для управління своїм ментальним здоров'ям.

Система має бути економічно доступною для широкого кола користувачів, включаючи безкоштовний базовий доступ та платні курси, що надаватимуть додаткові можливості.

Безпека та конфіденційність даних користувачів буде головним пріоритетом. Високий рівень безпеки та конфіденційності буде забезпечувати захист особистої інформації та безпечне зберігання даних, подібно до BetterHelp.

1.4 Монетизація

Монетизація системи для онлайн-тренінгів ментального здоров'я базується на концепції оплати за доступ до курсів та індивідуальних тренінгів. Цей підхід дозволяє користувачам вибирати конкретні послуги відповідно до їхніх потреб. Кожен курс може включати різноманітні навчальні матеріали, такі як відео, інфографіку та текстові пояснення, а також тести для самоперевірки. Оплата за курси відбувається одноразово при їх придбанні або через систему передплат, яка надає доступ до всього ряду курсів за фіксовану суму.

Окрім курсів, система також пропонує можливість запису на індивідуальні онлайн-сесії з менторами та тренерами. Ці сесії дають користувачам змогу отримати персоналізовану допомогу, відповіді на специфічні запитання та розробку індивідуальних планів розвитку. Платіж за такі сесії організований за кожне заняття окремо.

1.5 Постановка задачі

Розробка програмної системи для онлайн-тренінгу ментального здоров'я має на меті вирішення низки проблем, виявлених під час аналізу предметної галузі.

Система повинна забезпечити користувачам структуровані курси та інші психологічні методики. Важливою складовою є адаптація під індивідуальні потреби кожного користувача через оцінку їхнього психоемоційного стану та підбір відповідних методик.

Додатково, система повинна бути зручною для користувачів різних вікових категорій та з різним рівнем технічної підготовки, забезпечуючи простий та інтуїтивний інтерфейс. Доступність системи також є ключовим аспектом, оскільки вона повинна бути економічно доступною для широкого кола користувачів, включаючи безкоштовний базовий доступ та платні курси з додатковими можливостями.

Ефективність системи потребує підтвердження через тестування та валідацію, що включає збір даних про користувачів, аналіз їхнього прогресу та внесення корективів до методик за потреби. Важливо також забезпечити можливість моніторингу стану користувачів та надання їм необхідної підтримки, включаючи автоматичні нагадування та доступ до консультантів або менторів.

Інтеграція з іншими онлайн-сервісами та платформами дозволить підвищити функціональність та зручність використання системи. Основна задача полягає у створенні ефективної, доступної та науково обґрунтованої системи для онлайн-тренінгів ментального здоров'я, яка зможе забезпечити користувачам необхідну підтримку та допомогу у покращенні їхнього психічного стану та загального добробуту.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Ідея проєкту полягає у створенні програмної системи для онлайн-тренінгів і курсів з питань ментального здоров’я. Мета проєкту – надати зручний і простий спосіб навчання для підвищення обізнаності в галузі психологічного здоров’я та підтримки особистого добробуту. Ціль переддипломної роботи полягає в управлінні проєктом від етапу формування вимог до готового програмного продукту. Визначимо головні задачі:

- аналіз потреб цільової аудиторії;
- визначення вимог до продукту;
- визначення інструментів розробки;
- розробку плану проєкту;
- управління проєктом;
- тестування.

Перед початком формування вимог до продукту треба визначити потреби цільової аудиторії. Цільова аудиторія програмної системи для онлайн-навчання з ментального здоров’я включає різні групи користувачів, кожна зі своїми специфічними потребами. Професійні психологи та тренери потребують функцій для створення курсів і тренінгів, керування розкладом. Учасники навчання, з іншого боку, потребують інтуїтивного інтерфейсу для запису на курси, перегляду матеріалів, спілкування з менторами та відстеження прогресу.

Для подальшого моніторингу роботи команди та дотримання термінів розробки продукту треба чітко сформулювати вимоги до системи, які будуть відповідати потребам та очікуванням користувачів, а саме:

- вимоги до функціональності системи;
- вимоги до безпеки та конфіденційності;
- вимоги до локалізації;
- вимоги до масштабованості системи.

Функціональні вимоги до системи онлайн-навчання з ментального здоров'я враховують необхідність підтримки повного циклу взаємодії між користувачами та

платформою. Кожен користувач повинен мати можливість легко зареєструватися, увійти в систему, переглядати та редагувати свій профіль. Курси можуть бути як безкоштовними, так і платними. Щоб приєднатися до платного курсу, користувач спочатку повинен придбати його за допомогою системи онлайн-платежів на сайті.

Після успішного запису на курс учасники мають отримати доступ до всіх матеріалів та можливість відстежувати свій прогрес. Їм слід надати доступ до інформації про курси та менторів, зокрема, із можливістю бронювання часу для відеоконсультацій. Крім того, користувачам необхідно переглядати список пройдених курсів для оцінювання своїх досягнень та планування подальшого навчання. Такий підхід забезпечить комплексне охоплення всіх функцій для зручного користування платформою як тренерами, так і учасниками.

Так як додаток будемо зберігати конфіденційну інформацію о користувачах необхідно гарантувати безпеку особистих та чутливих даних. Аутентифікація користувачів повинна бути безпечною завдяки шифруванню даних, що передаються.

Доступ до різних типів даних повинен бути обмежений відповідно до ролі користувача, з можливістю налаштування прав доступу адміністратором.

Система повинна бути здатною підтримувати зростаючу кількість користувачів та обробляти великі обсяги даних. Технологія контейнеризації, така як Docker, може бути використана для забезпечення горизонтального масштабування. Це дозволяє легко додавати нові компоненти або модулі, а також розподіляти навантаження між кількома інстанціями серверів, підвищуючи загальну продуктивність системи.

Система повинна підтримувати принаймні дві основні мови: англійську та українську. Для цього інтерфейс повинні мати можливість перекладу. Крім того, формат дат, часу та валют має бути адаптований під кожну конкретну мову та регіон, що дозволить користувачам зручно взаємодіяти з інтерфейсом та отримувати релевантну інформацію без зайвих конверсій.

Також не менш важливим етапом, для успішного створення програмної системи, перед початком створення плану, проєкту є вибір інструментів розробки. З огляду на вимоги до проєкту було обрано такі інструменти розробки:
для бекенду було обрано Python і фреймворк Django, так як він надає повний набір інструментів для розробки масштабованого та гнучкого веб-додатку [4];

а) з метою забезпечення ефективної розробки та управління кодом доцільно використовувати середовище PyCharm, яке має повну підтримку Python і допомагає підвищити продуктивність;
б) для керування проєктом потрібні системи контролю версій, такі як Git, а також хостинг коду на GitHub або GitLab;
в) база даних PostgreSQL стане оптимальним вибором для зберігання інформації про користувачів, курси, прогрес та інші дані.

Так як для фронтенду програмного продукту буде використовуватися React у поєднанні з Material-UI, які забезпечать створення зручного та привабливого інтерфейсу, необхідно подбати про правильне подання даних з серверної частини, для подальшої її обробки на стороні клієнту.

При розробці програмного продукту планується використовувати методологію Scrum, яка є популярним гнучким підходом до управління проєктами, орієнтованим на командну роботу, адаптивне планування та постійне вдосконалення. Основною метою Scrum є створення цінності за допомогою ітеративних процесів, в яких команди можуть швидко реагувати на зміни та розробляти найактуальніші рішення.

Для координації проєкту та взаємодії між розробниками використовуватиметься система Jira, яка не тільки дозволяє відстежувати помилки, але й надає функціонал для організації спілкування, моніторингу завдань і створення ефективного беклогу продукту [5].

Треба подбати про декілька етапів:

а) формування списку завдань продукту (Product Backlog): визначення ключових вимог і функцій продукту, із забезпеченням можливості його корекції під час розробки;

б) планування спринтів (Sprint Planning): постановка завдань для виконання в наступному спринті та оцінка часу, потрібного для їх реалізації;
в) щоденні наради (Daily Scrum): організація коротких щоденних зустрічей для аналізу прогресу та виявлення можливих перешкод;
г) огляди й ретроспективи спринтів (Sprint Review & Sprint Retrospective): перегляд виконаних завдань і формування плану дій для майбутніх спринтів.

Для забезпечення надійності та стабільності програмної системи, тестування є невід'ємною частиною процесу розробки. Тестування буде включати два основних типи: юніт тестування та мануальне тестування.

Юніт тестування буде використовуватися для перевірки найменших частин коду — окремих модулів або компонентів, щоб забезпечити їх коректну роботу окремо від усієї системи. Це дозволить виявити та усунути помилки на ранніх етапах розробки, перш ніж код буде інтегровано з іншими частинами системи. Для написання юніт тестів буде використовуватися бібліотеки, такі як pytest для Python, які дозволяють автоматизувати процес тестування та забезпечити швидке виконання тестів.

Мануальне тестування важливе для перевірки системи у цілісному вигляді, зокрема інтерфейсу користувача та взаємодії між різними компонентами системи, такими як REST API [6]. Це дозволяє оцінити зручність користування, відповідність дизайну та функціональності вимогам користувачів. Відповідальними за мануальне тестування повинні будуть виконати різні сценарії використання системи, ввід різних типів даних, перевірити реакцію системи на крайові випадки і взагалі оцінюють загальне відчуття від взаємодії з програмою.

Мобільний застосунок має бути сумісним із платформами Android та iOS. Для Android мінімальна версія API 23 (Android 6.0 Marshmallow), а для iOS - не нижче версії 10, що забезпечить сумісність з більшістю сучасних пристроїв. Веб-версія має підтримувати сучасні браузери, зокрема Google Chrome (версія 124.0.0.0 або пізніше), Firefox, Safari та Edge.

Система повинна працювати з кукі для зберігання сесій та персоналізації користувацького досвіду. Це допоможе користувачам продовжувати навчання з того місця, де вони зупинилися, та отримувати рекомендації на основі попередніх дій.

Для забезпечення безперебійної роботи на різних пристроях, інтерфейс має бути адаптивним і відповідати принципам UI/UX-дизайну, гарантуючи, що кожен користувач отримає зручний та інтуїтивно зрозумілий досвід взаємодії з платформою [7].

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування ПЗСпочатку створимо Use-case діаграму (див. рис. 3.1), яка відображатиме функціональні можливості програмної системи з точки зору користувача, забезпечуючи візуальне уявлення про різні дії та взаємодії між користувачем і системою.

На діаграмі буде представлено декілька типів користувачів (акторів), кожен з яких має свій набір взаємодій із системою. Ключові актори включають учасників та адміністраторів. Їхні дії описують основні функції, які може виконувати кожна роль.

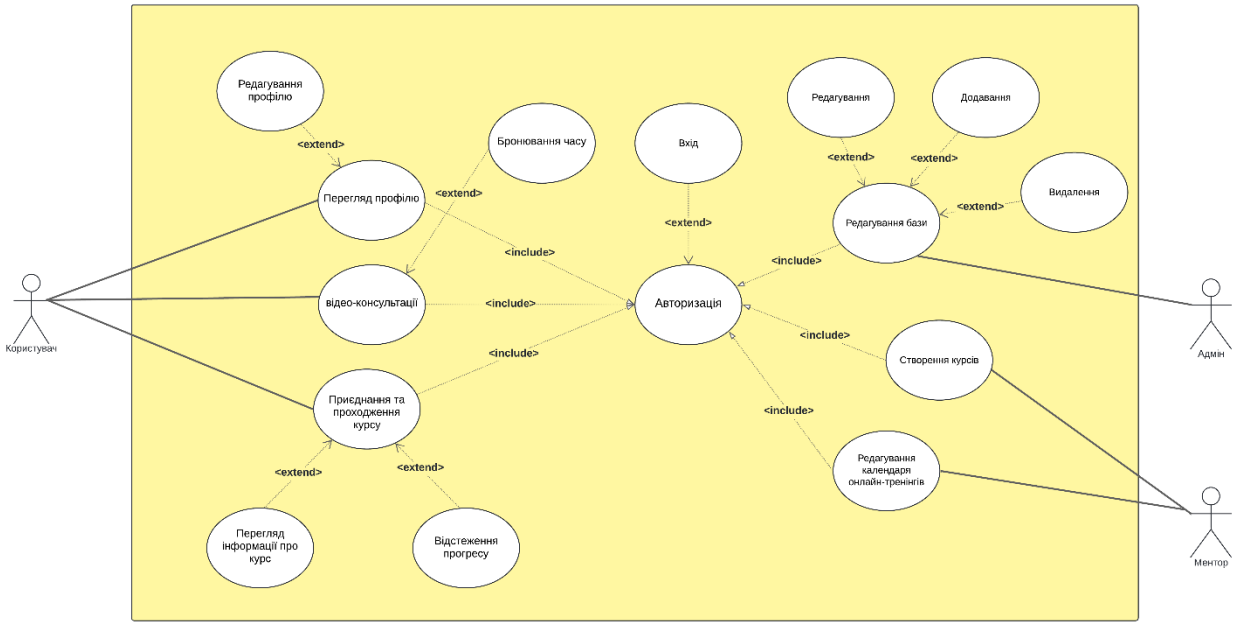


Рисунок 3.1 – Діаграма прецедентів програмної системи (рисунок виконано самостійно)

Користувачі мають можливість реєструватися на платформі та переглядати інформаційні матеріали, але без можливості взаємодії чи проходження курсів. Авторизовані користувачі, які входять у систему, отримують доступ до своїх профілів, можуть записуватися на курси, проходити навчальні матеріали та відстежувати свій особистий прогрес. Вони також можуть взаємодіяти з менторами, бронюючи час для консультацій.

Ментори в системі відповідають за створення та управління курсами, завантаження навчальних матеріалів і тестів, моніторинг прогресу учасників та видачу зворотного зв'язку. Вони також проводять відеоконсультації та аналізують ефективність навчальних програм для подальшої оптимізації.

Адміністратори системи відповідають за керування користувацькими акаунтами, налаштування параметрів системи, забезпечення безпеки даних і вирішення технічних проблем користувачів. Вони також мають можливість моніторити активність в системі для запобігання несанкціонованому доступу та управління доступом до різних ресурсів і функцій.

3.2 Архітектура системи

Архітектура системи для онлайн-навчання з ментального здоров'я побудована за клієнт-серверною моделлю (див. рис. 3.2), яка дозволяє централізовано управляти даними та бізнес-логікою через серверні застосунки, обслуговуючи клієнтські застосунки. Вибір такої архітектури зумовлений потребою в масштабованості та гнучкості, оскільки система повинна підтримувати зростання кількості користувачів та курсів. Серверна частина може бути розширена без змін у клієнтській частині, що дозволяє легко додавати ресурси або сервіси.

Централізоване управління забезпечує спрощення оновлень та обслуговування системи, забезпечуючи консистентний досвід для всіх користувачів на різних пристроях та швидке впровадження змін. Безпека критично важлива, і централізоване управління дозволяє ефективно забезпечувати аутентифікацію та авторизацію, мінімізуючи ризики несанкціонованого доступу. Інформація захищена, а критичні дані не зберігаються на клієнтських пристроях.

Стандартизація комунікації між клієнтом і сервером через добре визначені API сприяє уніфікації інтерфейсів, що покращує користувацький досвід і взаємодію з системою. Це дозволяє регулярно оновлювати інтерфейси, підтримуючи їхню актуальність і функціональність.

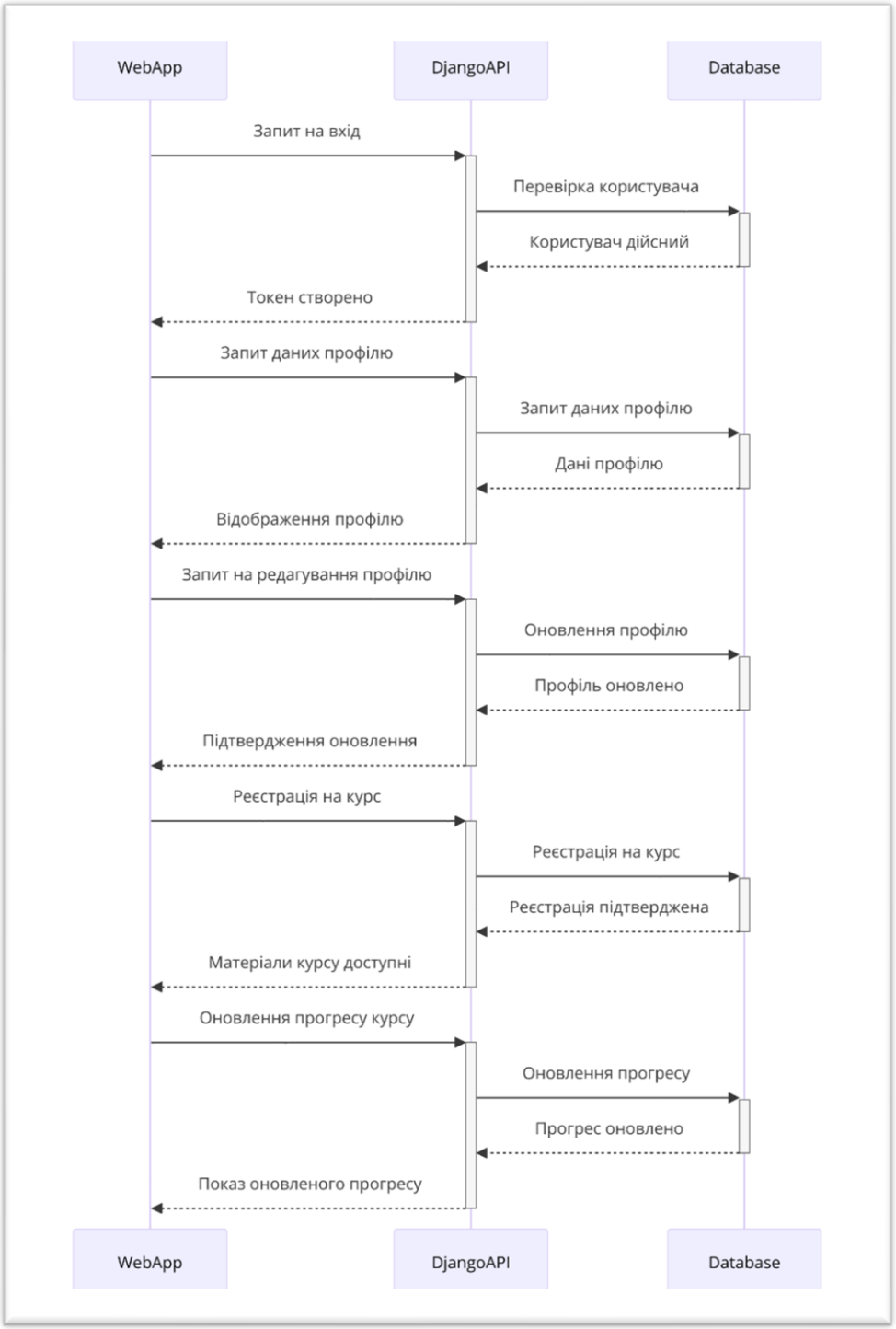


Рисунок 3.2 – Діаграма обраної моделі архітектури системи (рисунок виконано самостійно)

3.3 Обмеження та вимоги

При виборі технологій для розробки додатку особливу увагу необхідно приділити тривалості застосування обраної технології та її спроможності

адаптуватися до потенційних майбутніх оновлень і розширень проєкту. Важливо провести глибокий аналіз можливих обмежень, які можуть бути встановлені обраними технологіями, враховуючи специфічні вимоги та можливості проєкту, а також потенційний вплив на розширюваність і підтримку системи в довгостроковій перспективі.

Проєкт реалізується з використанням клієнт-серверної архітектури, де серверна частина відповідає за обробку запитів від клієнтської частини, яка забезпечує безпосередню взаємодію з користувачем. Архітектура серверної частини побудована на основі мікросервісів, що дозволяє ізолювати окремі функціональні блоки для незалежної роботи. Такий підхід значно спрощує процеси оновлення та масштабування кожної частини системи окремо, забезпечуючи кращу підтримку та гнучкість в управлінні змінами.

Для впровадження та розгортання проєкту активно використовується технологія Docker, яка надає можливості контейнеризації, дозволяючи стандартизувати середовище виконання по всьому пайплайну розробки [8]. Контейнеризація з Docker забезпечує легкість перенесення додатку між різними виробничими та тестовими середовищами без необхідності зміни конфігурації та залежностей, що значно знижує витрати на технічне обслуговування та налаштування інфраструктури.

Як основу для зберігання даних в проєкті використовується система управління базами даних PostgreSQL, що інтегрується з фреймворком Django та мовою програмування Python. Використання об'єктно-реляційного відображення (ORM) в Django спрощує взаємодію з базою даних, перетворюючи складні SQL-запити на більш зрозумілі та легкі для розробки високорівневі конструкції Python. Це не тільки полегшує процес розробки, але й покращує ефективність роботи з даними, забезпечуючи кращу продуктивність та масштабованість проєкту в цілому.

Однією з ключових переваг Django є вбудований адміністративний інтерфейс, який автоматично створюється на основі моделей даних. Це дозволяє легко керувати даними через веб-інтерфейс без необхідності писати додатковий код для адміністрування бази даних.

Бекенд, який взаємодіє з клієнтською частиною, повинен відповідати ряду вимог, щоб забезпечити ефективну та безпечну взаємодію. Основні вимоги до бекенду включають наявність API з підтримкою аутентифікації та сесій, валідації вхідних даних, а також ефективної обробки помилок і забезпечення безпеки даних. Важливою є також оптимізація відгуків сервера, щоб забезпечити швидкість обробки запитів, важливу для зручності користувачів.

Бекенд повинен підтримувати можливість легкої інтеграції з різними клієнтськими платформами, включаючи мобільні пристрої, забезпечуючи адаптивність. Також бекенд може включати компоненти для управління станом сесії користувачів, що дозволяє відстежувати авторизацію та збереження користувацьких налаштувань між сесіями.

Окрім технічних аспектів, бекенд має бути спроєктований з урахуванням можливості масштабування для підтримки зростаючої кількості користувачів та даних. Система повинна забезпечувати високу доступність та надійність, мінімізувати час простою та оптимізувати ресурси для обробки пікових навантажень.

Вимоги до безпеки бекенду мають включати захист від загроз типу SQL ін'єкцій, XSS атак та інших загроз. Це вимагає використання захищених методів програмування, ретельне тестування та аудит безпеки компонентів.

Також бекенд повинен мати можливість підтримувати різноманітні адміністративні функції через веб-інтерфейс, що спрощує управління даними та налаштуваннями системи без необхідності безпосереднього доступу до сервера. Важливою є здатність системи адаптуватися та інтегруватися з іншими системами та сервісами, що можуть знадобитися для розширення функціоналу або інтеграції зі сторонніми рішеннями.

3.4 Забезпечення якості

Для підтримки високої якості програмного продукту Minder, критично важливо, щоб усі члени команди мали єдину думку щодо внесення будь-яких змін в проєкт. Під час обговорення потенційних змін, важливо добитися консенсусу,

дозволяючи кожному члену команди висловити свої ідеї або заперечення. Такий діалог продовжується до моменту, коли всі знайдуть спільне рішення, яке задовольнить потреби всіх учасників.

Цей процес сприяє згуртованості команди, забезпечує включення всіх у важливі рішення і допомагає уникнути можливих конфліктів у майбутньому. В результаті отримуємо виважене рішення, яке підтримала уся команда та яке сприяє успішній реалізації проєкту.

Забезпечення постійного зв'язку між членами команди є ключовим елементом для підвищення якості розробленого продукту. Це передбачає проведення регулярних щоденних зустрічей та комунікацію через електронну пошту та месенджери. Мета такого спілкування полягає у вирішенні поточних проблем, обміні ідеями, обговоренні можливих рішень та діленні досвідом для досягнення спільних цілей.

Щоденні зустрічі зазвичай тривають від 10 до 20 хвилин і проводяться за допомогою платформи Google Meet, що дозволяє членам команди швидко обмінюватися інформацією та підтримувати високий рівень взаєморозуміння.

Така регулярність зв'язку сприяє ефективному керуванню проєктом, швидкому вирішенню проблем та прийняттю рішень, що позитивно впливає на якість та успішність розробки продукту.

У команді передбачено проведення обов'язкових перевірок, які включають дискусії та технічні огляди. Ці перевірки проводяться онлайн кожні два тижні з використанням Google Meet або Discord і тривають до години. Вони спрямовані на виявлення та обговорення помилок, відповідності програми вимогам, а також на удосконалення процесів управління проєктом.

Технічні перевірки включають детальний огляд розроблених компонентів, таких як інтерфейси, форми, бази даних, з метою забезпечення їх коректності та інтеграції. Ці перевірки забезпечують відповідність розробленого продукту встановленим стандартам.

Для контролю якості розробки використовуються різноманітні інструменти та методики, включаючи регулярні огляди, голосування, детальне проектування та

аналіз. Ці заходи націлені на мінімізацію помилок та підвищення якості розробки, забезпечуючи відповідність продукту всім необхідним вимогам і стандартам.

3.5 Керування проєктом

Управління проєктом охоплює різноманітні аспекти, такі як планування, реалізація, моніторинг процесів розробки, а також підтримка і оновлення проєкту. Цей процес вимагає розділення на декілька критичних етапів, кожен з яких спрямований на досягнення конкретних цілей, що стосуються додатку.

На стадії ініціального планування команда здобула загальні відомості про проєкт, сформулювала ключові цілі та проаналізувала цільову аудиторію з урахуванням її основних потреб, які повинен задовольнити продукт. Також було визначено основні функціональні можливості, якими зможуть скористатися користувачі системи.

В ході реалізації проєкту планується постійне спостереження за процесом. Для точного моніторингу прогресу та оцінки успіхів проєкту команда використовує систему управління проєктами Jira, що дозволяє тримати проєкт під контролем і гарантувати досягнення визначених цілей у встановлені терміни.

Jira пропонує численні інструменти для управління проєктами, включаючи складання списку завдань, призначення пріоритетів, відстеження часу, додавання коментарів та зміну статусів завдань. Крім того, система дозволяє генерувати звіти та статистику, які допомагають аналізувати продуктивність та ефективність роботи команди.

На основі встановлених вимог до системи онлайн-тренінгу ментального здоров’я буде сформовано перелік завдань для виконання, щоб забезпечити успішне завершення проєкту. В Jira також можна організувати процес роботи на кількатижневі етапи, які відомі як Story. Кожне завдання може бути класифіковано в категорії, такі як "To Do", "In Progress", "Testing", "Done", та відображено на дошці завдань (див. рис. 3.3).

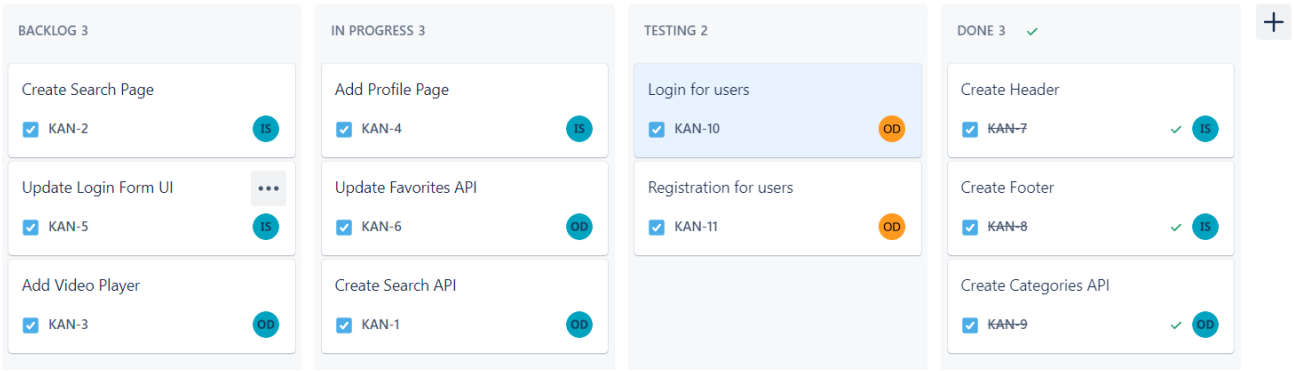


Рисунок 3.3 – Заповнена Story для проєкту на Jira (рисунок виконано самостійно)

3.6 Тестування

Для забезпечення високої якості програмного продукту, призначеного для онлайн-навчання у сфері ментального здоров'я, заплановано виконання двох видів тестувань: мануального та юніт-тестування.

Мануальне тестування пропонує значні переваги для даного проєкту. Цей метод є високо адаптивним, що дозволяє легко вносити зміни у програмне забезпечення і миттєво перевіряти відповідність результатів. Він також уможливлює детальну перевірку правильності обміну даними через Rest API та інші ключові аспекти системної взаємодії. Мануальне тестування виявляє тонкощі функціоналу, які автоматизовані системи можуть пропустити. Цей метод не вимагає глибоких технічних знань для виконання та дозволяє швидко описати та адресувати знайдені помилки. Особливо цінним є його вклад у оцінку реального користувацького досвіду. Для проведення тестування буде застосовано HTTP-клієнт, наприклад Postman, для перевірки API (див. рис. 3.4), а також тестування користувацького інтерфейсу після його інтеграції з серверною частиною.

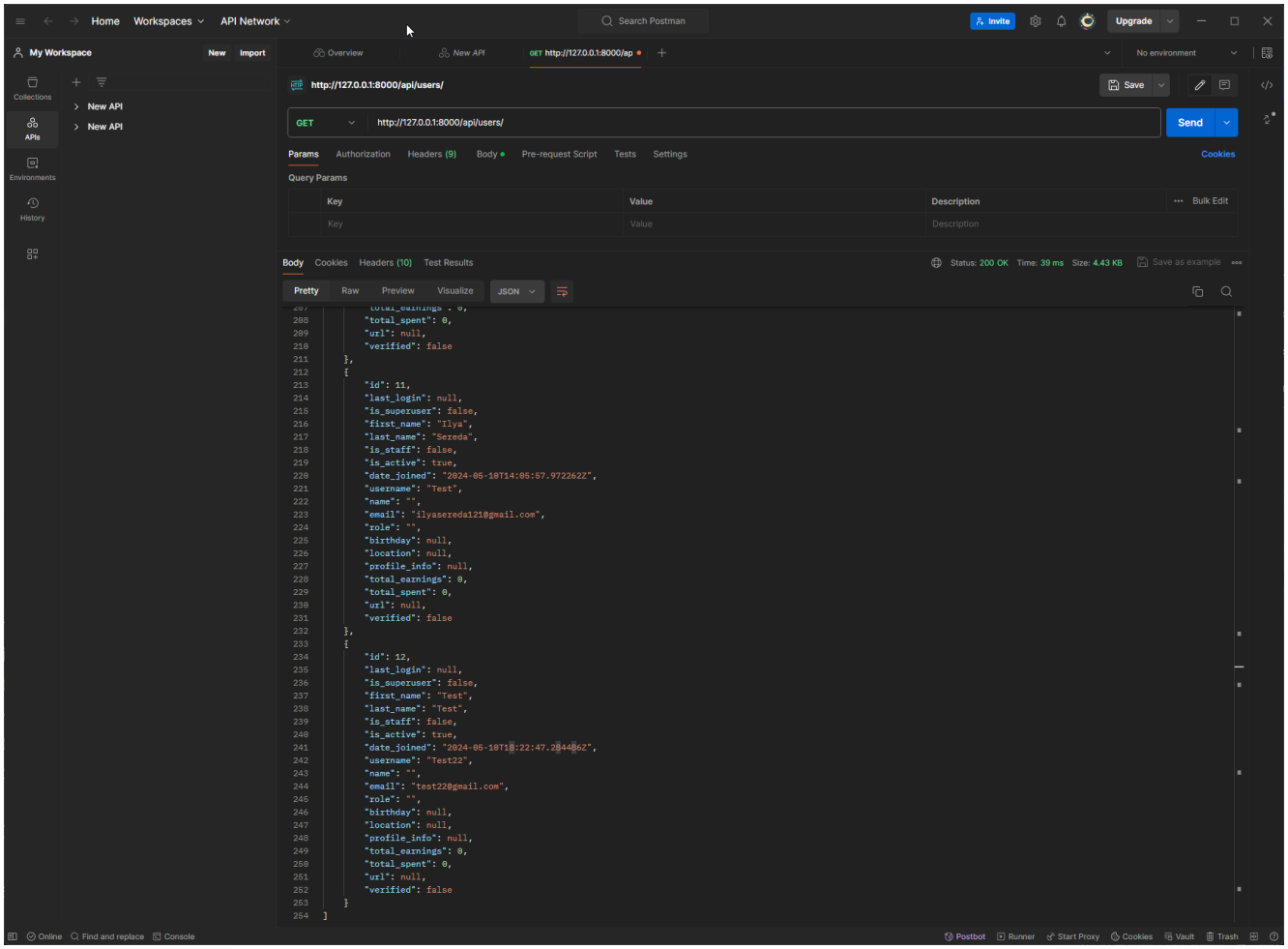


Рисунок 3.4 – Postman (рисунок виконано самостійно)

Юніт-тестування є критично важливим компонентом загальної стратегії тестування. Цей метод тестування спрямований на перевірку індивідуальних компонентів програмного забезпечення, таких як функції, методи та класи, щоб забезпечити їх правильну функціональність. Використання юніт-тестів дозволяє виявляти помилки на ранніх етапах розробки, що значно підвищує якість кінцевого продукту. Ці тести перевіряють роботу модулів окремо від решти системи, що допомагає визначити відповідність кожного компонента встановленим вимогам і стандартам.

Для максимальної ефективності, юніт-тести та мануальні перевірки слід виконувати за допомогою даних, що максимально наближені до реальних. Це дозволяє переконатися, що система буде належним чином функціонувати у реальних умовах і не спричинить неочікуваних помилок для кінцевих користувачів.

Процес тестування організовано у декілька фаз. Перша фаза включає проведення мануальних тестів та юніт-тестування, під час якого фіксуються всі виявлені помилки для подальшої передачі розробникам. Якщо в ході першої фази виявлені помилки, переходимо до другої фази — виправлення помилок. Після того, як проблема виправлена, розробник змінює статус помилки на "потребує повторного тестування".

Ці фази повторюються до того моменту, поки не будуть усунуті всі проблеми. Такий систематичний підхід дозволяє швидко виявляти та усувати помилки, що покращує швидкість реагування на виклики та загальну якість продукту.

Оновлення проєкту та впровадження нових функцій вимагає також оновлення та додавання нових юніт-тестів, а також перегляд мануальних тестів. Це критично важливо для того, щоб переконатися, що нововведення не породжують нових помилок, які можуть негативно вплинути на досвід користувачів або стабільність програмної системи.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Обрання програмних засобів для розробки back-end частини додатку.

Основною мовою програмування для реалізації back-end частини був обраний Python через низку своїх переваг. По-перше, через велику кількість фреймворків, таких як Django та Flask, які значно прискорять процес розробки, дозволивши зосередитися на логіці додатку, а не на рутинних завданнях. Також Python має вбудовані механізми безпеки та можливість легко інтегрувати додаткові засоби захисту, що вкрай важливо для розробки систем, які працюють з конфіденційною інформацією. Нарешті, ця мова програмування добре інтегрується з іншими технологіями та інструментами, що підтримують сучасні методології розробки, такі як DevOps, Continuous Integration/Continuous Deployment (CI/CD), що знадобиться в подальшому для безперервного розвитку та підтримки проєкту.

Django був обраний для реалізації back-end частини проєкту з кількох важливих причин, які роблять його ідеальним вибором для розробки програмного забезпечення. По-перше, Django відомий своєю здатністю пришвидшувати процес розробки завдяки використанню готових компонентів і зручного механізму налаштувань. Це дозволяє створювати повнофункціональні веб-додатки значно швидше, що є критичним для швидкого запуску MVP (мінімально життєздатного продукту). По-друге, Django пропонує рішення, які включають автоматичні адміністративні інтерфейси, систему авторизації та аутентифікації, ORM (Object-Relational Mapping) для роботи з базами даних, а також інструменти для забезпечення безпеки та управління сесіями користувачів. Також однією з ключових причин вибору Django є його високий рівень безпеки. Django забезпечує захист від поширених вразливостей, таких як SQL-ін’єкції, XSS (Cross-site scripting) та CSRF (Cross-site request forgery). Крім того, Django має одну з найбільших та найактивніших спільнот розробників, а також велику кількість доступних бібліотек та розширень, що забезпечує підтримку та можливість швидкого вирішення будь-яких технічних питань, що можуть виникнути в процесі розробки. Масштабованість є ще однією перевагою Django. Він дозволяє легко

масштабувати додаток у міру зростання кількості користувачів та обсягу даних завдяки добре продуманій архітектурі та підтримці різних рівнів кешування, асинхронної обробки запитів та можливості горизонтального масштабування баз даних.

Середа розробки яку було обрано для реалізації проєкту – PyCharm. Він є одним з найпотужніших і функціональних інтегрованих середовищ розробки для мови програмування Python. PyCharm підтримує безліч розширень та плагінів, що дозволяє інтегрувати його з іншими інструментами та сервісами, необхідними для проєкту. Для проєкту, де основною технологією back-end є Django, PyCharm пропонує спеціалізовану підтримку цього фреймворку. Це включає інтегровані інструменти для управління базами даних, а також інтеграцію з веб-серверами та інструментами розгортання.

Docker був обраний для проєкту як інструмент контейнеризації з кількох важливих причин. По-перше, Docker забезпечує упаковку додатків разом з усіма необхідними залежностями в єдиний контейнер, що гарантує стабільність і відтворюваність роботи програми в різних середовищах. Це важливо для проєкту, оскільки це забезпечить безперебійну роботу незалежно від середовища, в якому запускається додаток. По-друге, використання Docker в поєднанні з GitHub дозволить налаштувати безперервну інтеграцію та доставку (CI/CD). Це автоматизує процеси тестування та деплою, підвищуючи ефективність розробки і скорочуючи час на випуск нових версій продукту. Завдяки GitHub Actions і Docker можна реалізовати CI/CD пайплайни, які автоматично будуть запускати тести при будь-яких змінах у кодовій базі, що забезпечує швидке і надійне впровадження нових функцій. Крім того, контейнеризовані додатки можна швидко розгортати на будь-якому сервері, що підтримує Docker, без необхідності встановлювати додаткові залежності чи налаштовувати середовище вручну. Це зменшує час на розгортання і мінімізує ризики помилок.

4.2 Вибір засобів зберігання даних

Для зберігання даних у проєкті було обрано реляційну базу даних PostgreSQL та Azure Storage Account для зберігання мультимедійних файлів [9]. Реляційна база даних була обрана через її здатність забезпечити структуроване зберігання даних з високою інтегрованістю та можливістю виконання складних запитів. Такий підхід дозволяє підтримувати цілісність даних, забезпечуючи їхню консистентність та відмовостійкість, що є критично важливим для системи онлайн навчання з питань ментального здоров'я.

Вибір саме PostgreSQL як реляційної бази даних був обґрунтований її потужними функціональними можливостями та високою продуктивністю. PostgreSQL підтримує різноманітні типи даних, розширені функції індексування та пошуку, а також має потужні механізми для забезпечення транзакційної цілісності. Крім того, PostgreSQL є відкритим програмним забезпеченням з великою спільнотою розробників, що забезпечує постійну підтримку та розвиток. Це робить PostgreSQL ідеальним вибором для проєкту, де критично важливо мати надійну та масштабовану базу даних.

Azure Storage Account був обраний для зберігання мультимедійних файлів через його здатність забезпечити високу надійність та доступність великих обсягів даних. Використання окремого хмарного сховища для мультимедіа дозволяє зменшити навантаження на основну базу даних, забезпечуючи швидкий доступ до медіафайлів без впливу на продуктивність системи в цілому. Такий підхід також спрощує процес управління мультимедійними даними, дозволяючи зберігати великі файли окремо від основних структурованих даних.

Зберігання посилань на мультимедійні файли у базі даних замість самих файлів дозволяє ефективніше керувати обсягами даних та забезпечувати швидший доступ до інформації. Це також сприяє зменшенню витрат на зберігання, оскільки хмарні сховища оптимізовані для зберігання великих обсягів даних з низькою вартістю. Тому вибір PostgreSQL для зберігання структурованих даних і Azure Storage Account для мультимедіа є оптимальним рішенням, що забезпечує надійність, масштабованість та ефективність зберігання даних у розробленій

системі. Нижче приведено структуру зберігання мультимедіа в Azure Storage Account (див. рис. 4.1).

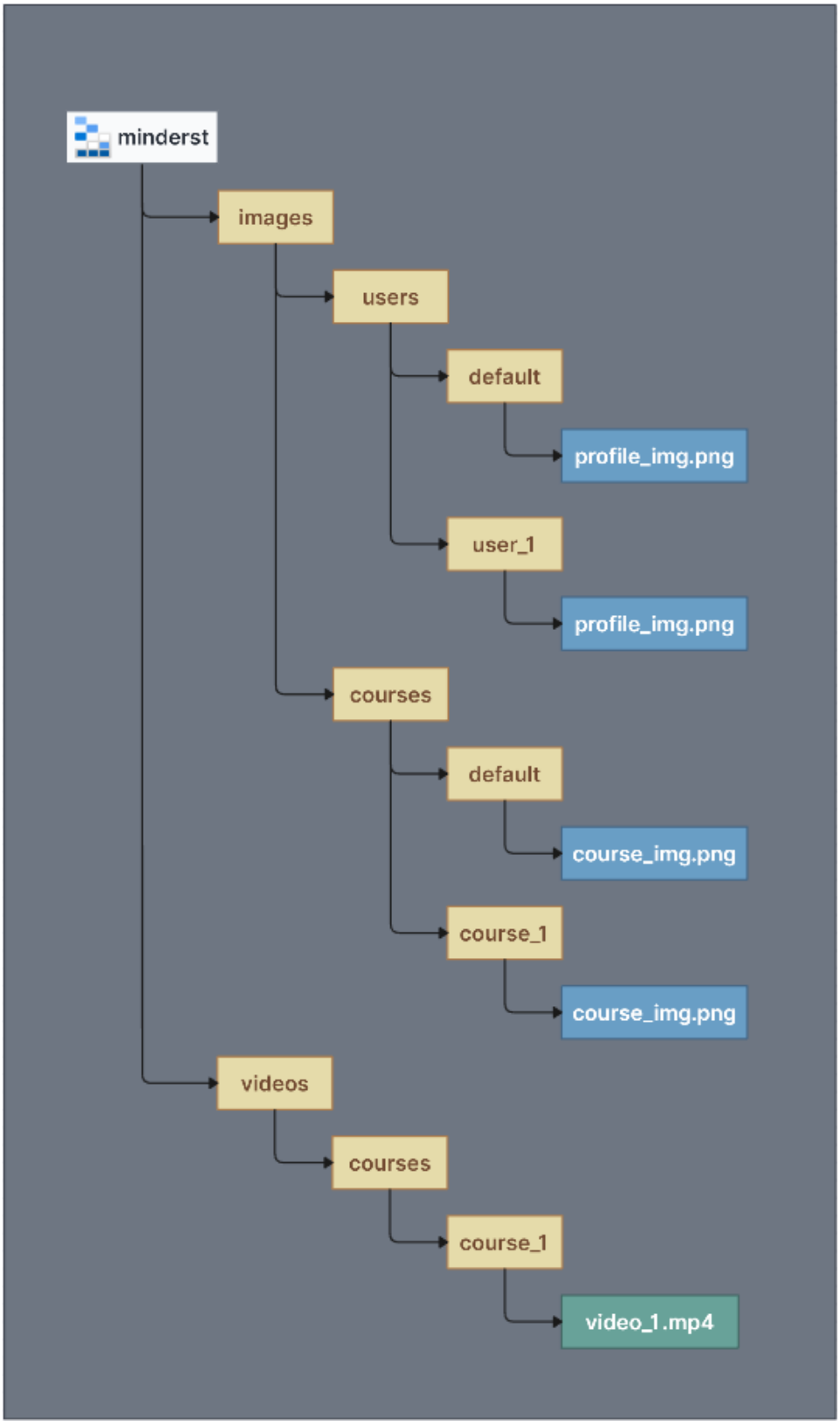


Рисунок 4.1 – Структура Azure Storage Account (рисунок виконано самостійно)

4.3 Структура проєкта

Проєкт складається з кількох ключових компонентів, які відіграють важливу роль у його функціонуванні. Файл ci-cd.yml використовується для налаштування процесів безперервної інтеграції та доставки (CI/CD) на платформі GitHub, що дозволяє автоматизувати тестування та збірку додатку.

Для зберігання мультимедійних файлів призначена директорія azure_storage, яка включає модулі для роботи з Azure Storage Account. Зміни у базі даних, в свою чергу, керуються за допомогою файлів міграцій у папці migrations, які синхронізують структуру бази даних з кодом системи.

Опис структури даних, що зберігаються в базі даних, реалізовано у файлі models.py. Серіалізація даних для API забезпечена у файлі serializers.py, що перетворює складні типи даних у формати, зручні для передачі між клієнтом і сервером. Маршрутизація URL-адрес регулюється файлом urls.py, а обробка запитів від користувачів — за допомогою файлу views.py.

Файл settings.py містить усі критичні налаштування системи, включно з параметрами підключення до бази даних та налаштуваннями безпеки.

Для тестування компонентів системи використовується директорія tests/, яка забезпечує якість та стабільність коду.

Файли docker-compose.yml і Dockerfile використовуються для налаштування та запуску багатоконтейнерних Docker додатків. docker-compose.yml координує збірку та запуск необхідних контейнерів, тоді як Dockerfile містить інструкції для створення Docker контейнера проєкту.

Файл requirements.txt забезпечує можливість швидкої установки всіх необхідних бібліотек та пакетів проєкту.

Файл manage.py це командний інструмент, що дозволяє керувати різними аспектами Django-проєкту. Цей файл автоматично створюється при ініціалізації нового проєкту Django і служить як обгортка над django-admin.py. З його допомогою можна виконувати ряд завдань управління, таких як міграції бази даних, створення нових додатків у проєкті, запуск сервера для розробки та виконання налагоджувальних завдань.

Повна структура проєкта зображена нижче (див. рис. 4.2).

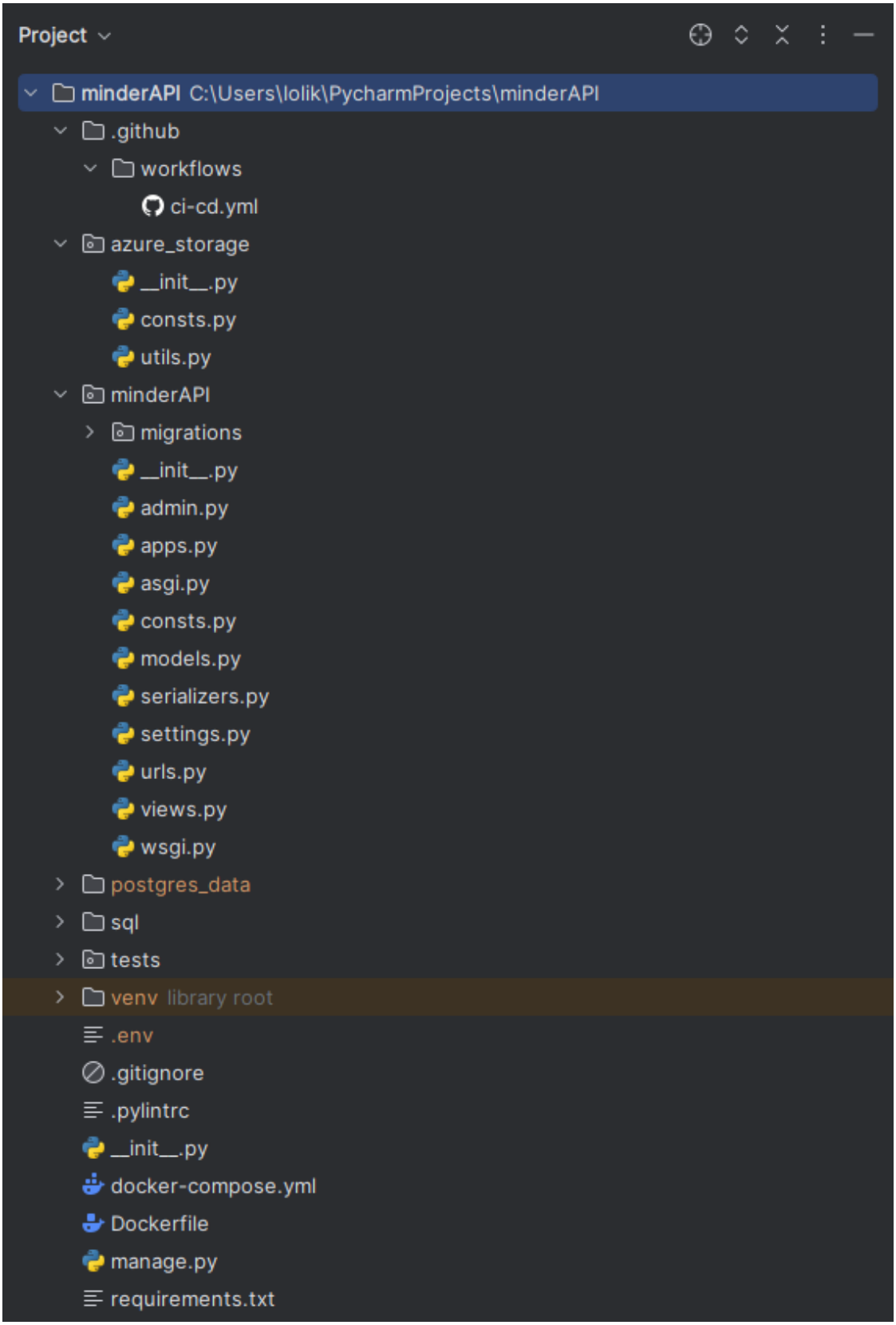


Рисунок 4.2 – Структура проєкту (рисунок виконано самостійно)

4.4 CI/CD потік проєкту

В проєкті потік CI/CD налаштовано через GitHub Actions, що дозволяє автоматизувати тестування та побудову коду безпосередньо з репозиторія на GitHub. Реалізований процес активується при внесенні змін до гілки master через push або pull request, забезпечуючи автоматичну перевірку нового коду та його готовність до розгортання.

Налаштування середовища виконано з використанням Python версії 3.9. Після відбувається автоматичне встановлення необхідних залежностей з файлу requirements.txt, що забезпечує узгодженість умов розробки і тестування. Перевірка на якість коду з використанням pylint підвищує легкість читання коду, виявляючи помилки і недоліки на ранніх стадіях. Запуск тестів за допомогою pytest допомагає виявити проблеми до розгортання проєкту.

Останніми кроками є створення Docker-образу та його відправлення на Docker Hub (див. рис. 4.3), що відбувається лише за умови успішного проходження всіх тестів і перевірок. Це гарантує, що новий код, який розгортається на серверах, є стабільним і пройшов усі необхідні перевірки.

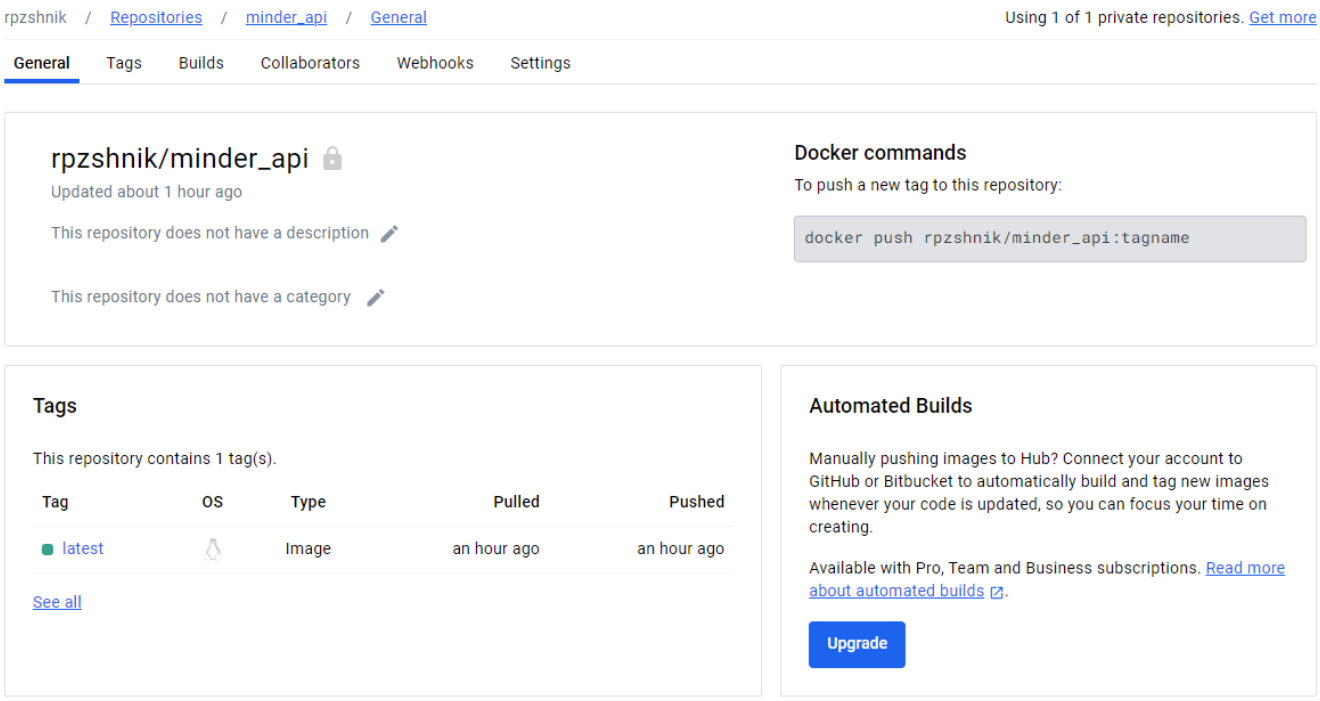


Рисунок 4.3 – Docker Hub репозиторій проєкту (рисунок виконано самостійно)

4.5 Авторизація за допомогою JWT токенів

Для авторизації користувачів в проєкті будуть використані JWT токени. JWT (JSON Web Token) - це безпечний та ефективний спосіб аутентифікації та авторизації користувачів у веб-додатках [10].

Коли користувач успішно проходить аутентифікацію, сервер генерує JWT токен, що містить закодовану інформацію про користувача, зокрема його ідентифікатор та інші необхідні дані. Цей токен підписується за допомогою секретного ключа, що гарантує його цілісність та неможливість підробки. Після отримання токена користувач надсилає його з кожним запитом до захищених ресурсів сервера.

Сервер, отримавши запит з JWT токеном, перевіряє його валідність за допомогою секретного ключа. Якщо токен є дійсним, сервер дозволяє доступ до запитуваного ресурсу. Це забезпечує безпечний доступ до ресурсів, оскільки тільки авторизовані користувачі можуть отримати доступ до захищених даних.

Ще одно перевага використання JWT токенів полягає у частковому полегшенні навантаження на сервер. Бо немає потреби зберігати стан сесії на стороні сервера. Всі потрібні дані про користувача зберігаються в самому токені.

Також було налаштовувано термін дії токенів був обмежений, щоб мінімізувати ризики використання втрачених або вкрадених токенів. Водночас, використання refresh-токенів дозволяє продовжувати сеанси користувачів без необхідності повторної аутентифікації.

Таким чином, впровадження авторизації за допомогою JWT токенів дозволить ефективно управляти доступом до ресурсів веб-додатку.

4.6 Розгортання додатку на платформі Azure

Для розгортання додатку було обрано платформу Microsoft Azure з кількох причин:

а) інтеграція з вже існуючими сервісами, а саме Storage Account який вже використовується для зберігання мультимедійних файлів, що забезпечує інтеграцію та централізоване управління ресурсами;

б) він пропонує високий рівень надійності та можливості масштабування, що є важливим для стабільної роботи та росту додатку;

в) він надає широкий спектр сервісів, які можуть бути використані для різних потреб додатку, таких як бази даних, аналітика, штучний інтелект і багато інших. Це буде корисно для подальшого розвитку програмного продукту.

Для розгортання додатку на платформі Azure було обрано Azure Web App сервіс. Його вибір був обґрунтований декількома перевагами:

а) він дозволяє швидко і легко розгорнути додаток без необхідності управління інфраструктурою;

б) він підтримує розгортання додатків у контейнерах, що дозволяє використовувати Docker-образи. Це саме те що необхідно для інтеграції з CI/CD потоком;

в) платформа дозволяє автоматично масштабувати додаток у відповідь на зміну навантаження, що забезпечує додатку безперебійну роботу навіть під час пікових навантажень;

г) можливість відстежувати час пікових навантажень та оптимізувати подальші витрати для Azure Web App, використовуючи вбудовані інструменти моніторінгу, такі як Azure Monitor або Application Insights.

Також одною з головних переваг Azure Web App є підтримка Continuous Deployment та інтеграція з репозиторієм для зберігання контейнерів, Docker Hub, в якому і зберігається образ.

Отже, в кінцевому результаті отримаємо процес, який складається з декількох етапів. Спочатку зміни вносяться в останню версію коду шляхом злиття з master гілкою. Після успішного проходження всіх перевірок, таких як тестування та оцінка якості коду, створюється Docker-образ, який відправляється в приватний репозиторій Docker. Після цього Web App автоматично підхоплює останню версію образу та розгортає додаток. Нижче показано логування розгортання додатку на Azure, який відображає успішне витягування останньої версії minder_api образу з репозиторію rpzshnik та запуск контейнерів (див. рис. 4.4).

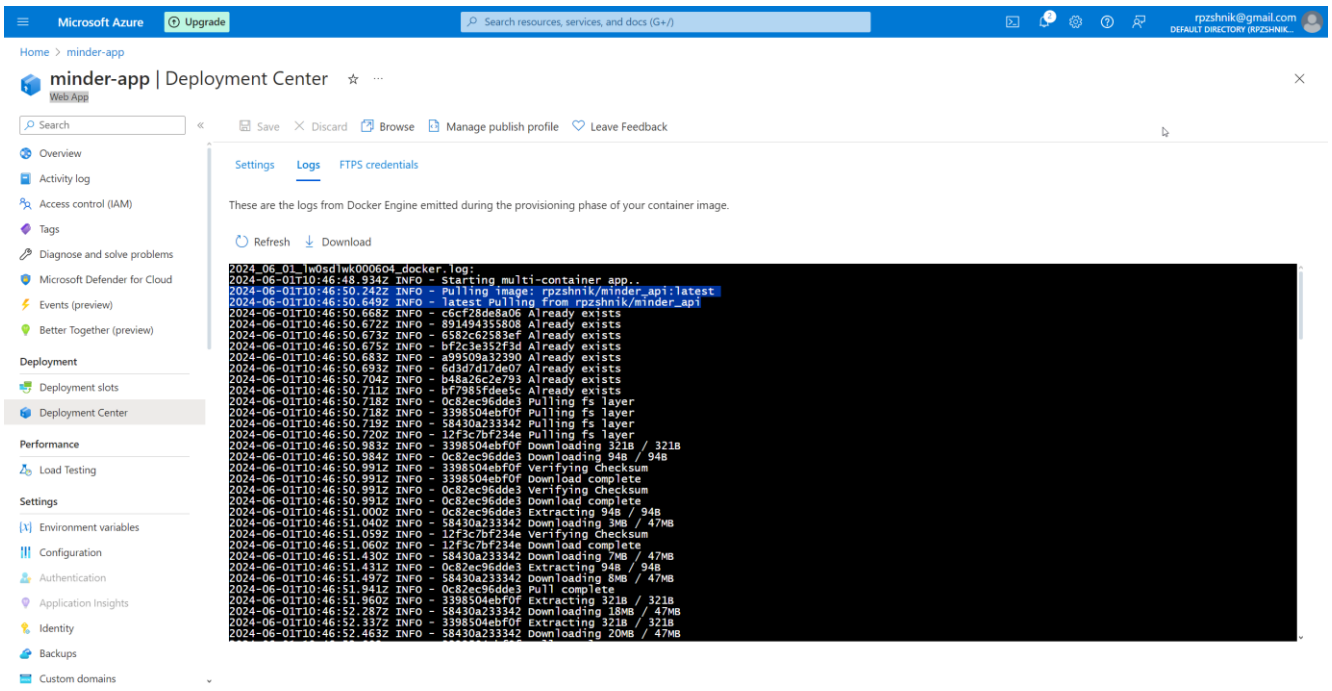


Рисунок 4.4 – Розгортання додатку на Azure (рисунок виконано самостійно)

На рисунку нижче показано кореневий API інтерфейс додатку (див. рис. 4.5), розгорнутого на Azure, доступного за посиланням https://minder-app.azurewebsites.net/.

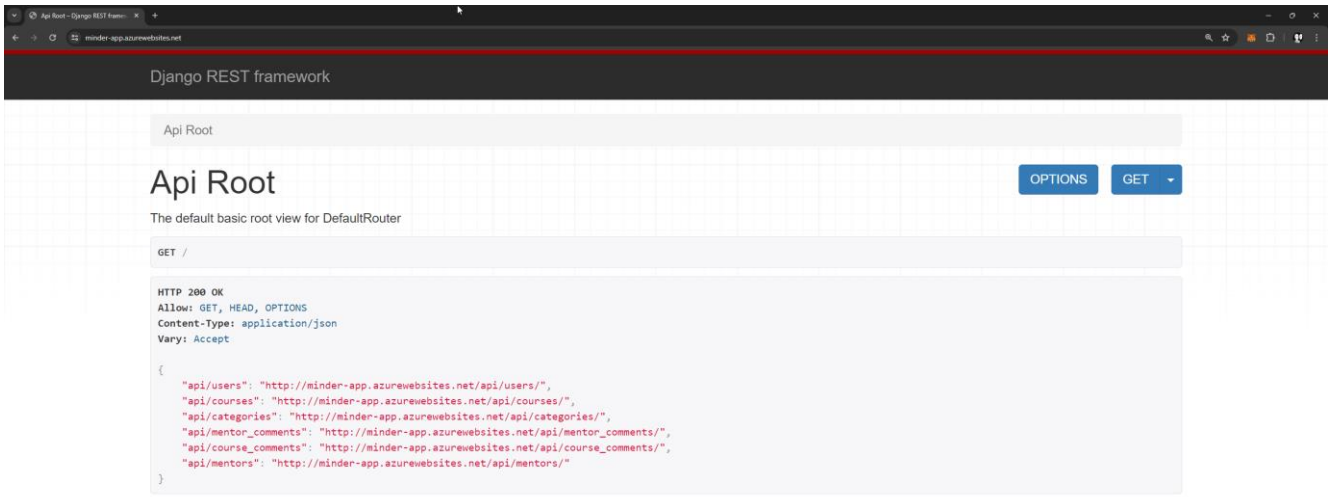


Рисунок 4.5 – Docker Hub репозиторій проєкту (рисунок виконано самостійно)

Цей процес забезпечує надійне та ефективне розгортання додатку, дозволяючи йому працювати в хмарному середовищі з високою доступністю та масштабованістю.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Юніт-тестування

Для окремих частин коду, таких як утиліти для роботи з Azure Storage Account та інші, основним інструментом тестування виступають юніт-тести. Метою цього підходу є виявлення помилок на ранніх етапах розробки, що значно підвищує якість кінцевого продукту. Юніт-тестування дозволяє перевірити коректність роботи кожної функції, методу та класу незалежно від інших частин системи.

У проекті для написання юніт-тестів було використано фреймворк pytest. Вибір цього інструменту обґрунтований його перевагами, такими як простота використання та гнучкість. Pytest забезпечує інтуїтивно зрозумілий синтаксис, що дозволяє швидко створювати тести, не витрачаючи багато часу на вивчення складних інструментів.

Інтеграція pytest з іншими інструментами розробки, такими як CI/CD системи, дозволяє автоматизувати процес тестування. Це означає, що тести запускаються автоматично при кожному оновленні коду, що значно зменшує ризик виникнення помилок у виробничому середовищі. Автоматизовані тести забезпечують високу швидкість і точність перевірки коду, що особливо важливо в умовах швидкого розроблення та впровадження нових функцій.

Прикладом ефективного використання pytest у проекті є тестування API. Зокрема, можна використовувати pytest разом з бібліотекою requests для перевірки коректності відповіді серверу на різні запити. Це включає перевірку статус-кодів, структури та змісту відповіді, а також поведінки API у разі неправильних або неповних запитів.

Ще однією перевагою pytest є його здатність працювати з фікстурами, які дозволяють налаштувати та ініціалізувати тестове середовище перед запуском тестів. Це робить процес тестування більш організованим і структурованим, дозволяючи уникати дублювання коду та знижувати ймовірність помилок у налаштуванні тестового середовища.

На додаток до цього, pytest підтримує параметризацію тестів, що дозволяє запускати один і той самий тест з різними наборами даних. Це особливо корисно для перевірки поведінки функцій з різними вхідними значеннями, що підвищує охоплення тестами і забезпечує більш ґрунтовну перевірку коду.

Загалом, використання pytest для юніт-тестування у проекті забезпечує високу якість коду, зменшує ризик виникнення помилок і прискорює процес розробки, роблячи його більш надійним і ефективним.

5.2 Мануальне тестування

Так як юніт-тестування не може покрити всі випадки, а також не завжди дозволяє виявити проблеми з інтеграцією компонентів та зручністю використання, мануальне тестування є важливим етапом тестування. Мануальне тестування дозволяє перевірити додаток з точки зору користувача та оцінити загальну зручність використання системи.

Процес мануального тестування складається з кількох етапів. Першим етапом є планування тестування, на якому визначаються основні цілі та обсяг мануального тестування. У плані вказуються всі сценарії, які необхідно протестувати, включаючи позитивні та негативні випадки. Наступним етапом є розробка тестових сценаріїв, які розробляються на основі вимог до системи та включають покрокові інструкції для перевірки кожної функціональності.

На етапі проведення тестування виконуються тестові сценарії, документують результати та виявляють дефекти. Важливо ретельно зафіксувати всі помилки та неточності, щоб в подальшому було можливо їх швидко виправити. Після завершення тестування результати аналізуються, і визначаються критичність та пріоритетність виявлених дефектів. Це допомагає зрозуміти, які помилки потрібно виправити в першу чергу. Після виправлення дефектів проводиться повторне тестування для перевірки коректності виправлень та переконання, що нові помилки не були введені в систему.

Для бекенд частини мануальне тестування використовується для перевірки API-запитів. Основним інструментом для цього є Postman, який забезпечує зручний

інтерфейс для створення, відправлення та аналізу запитів. Однією з ключових переваг Postman є легкість передачі bearer token (JWT), що дозволяє ефективно тестувати захищені API.

Таким чином, мануальне тестування в проєкті є невід'ємною частиною процесу забезпечення якості програмного продукту. Воно дозволяє виявити дефекти, які можуть бути пропущені автоматизованими тестами, і забезпечити високу якість кінцевого продукту. Завдяки поєднанню мануального та автоматизованого тестування можна досягти максимальної ефективності та надійності програмного забезпечення, забезпечуючи задоволення потреб користувачів та відповідність їх очікуванням.

ВИСНОВКИ

У ході цього дослідження була розроблена програмна система для онлайн-тренінгу ментального здоров'я. Було визначено основні вимоги до системи, розглянуті необхідні технології та інструменти для створення повноцінної платформи. Проаналізувавши аналогічні системи, були виявлені унікальні функції та особливості, які дозволили спроєктувати систему, що забезпечує користувачам індивідуальні рекомендації та ефективний зворотній зв'язок від менторів.

Протягом роботи над проєктом були пройдені всі заплановані етапи, відповідно до графіку. На початковому етапі було визначено ключові функції та розроблено модель системи. Серверну частину побудовано на базі Python, фреймворку Django та бази даних PostgreSQL для забезпечення надійного та масштабованого зберігання даних. Інтерфейс системи розроблений із використанням технологій Material-UI та React, щоб надати користувачам простий, інтуїтивно зрозумілий досвід.

Особливістю архітектури цієї системи є її гнучкість та масштабованість, що дозволяє легко додавати нові функції. Для оптимізації процесів керування проєктом використовували Jira, а тестування включало як ручне, так і автоматизоване функціональне тестування.

Ця програмна система має на меті допомогти користувачам покращувати своє психічне здоров'я завдяки індивідуальним планам, які можна створювати спільно з менторами, а також за рахунок різноманітних ресурсів для самостійного навчання. Вона пропонує зручні інструменти для планування занять, взаємодії з менторами та відстеження особистого прогресу.

Система стане корисним інструментом для людей, які прагнуть отримати допомогу в покращенні свого ментального здоров'я. Вона сприяє саморозвитку та саморегуляції, допомагаючи людям справлятися зі стресом і підтримувати своє психологічне благополуччя.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Головна Headspace [Електронний ресурс] – URL: https://www.headspace.com (дата звернення: 13.05.2024).

2. Головна BetterHelp [Електронний ресурс] – URL: https://www.betterhelp.com (дата звернення: 13.05.2024).

3. Головна VerywellMind [Електронний ресурс] – URL: https://www.verywellmind.com (дата звернення: 13.05.2024).

4. Srivastava A. K. Django , The Python Web Framework. 2022. Т. 06, № 05. [Електронний ресурс] – URL: https://doi.org/10.55041/ijsrem13183 (дата звернення: 13.05.2024).

5. Jira Software Documentation [Електронний ресурс] – URL: https://confluence.atlassian.com/jirasoftware/jira-software-documentation-774242447.html (дата звернення: 13.05.2024).

6. REST API Tutorial [Електронний ресурс] – URL: https://restfulapi.net (дата звернення: 13.05.2024).

7. UI/UX Design 101: A Comprehensive Guide for Beginners [Електронний ресурс] – URL: https://medium.com/@nile.bits/ui-ux-design-101-a-comprehensive-guide-for-beginners-f6588c86e963 (дата звернення: 13.05.2024).

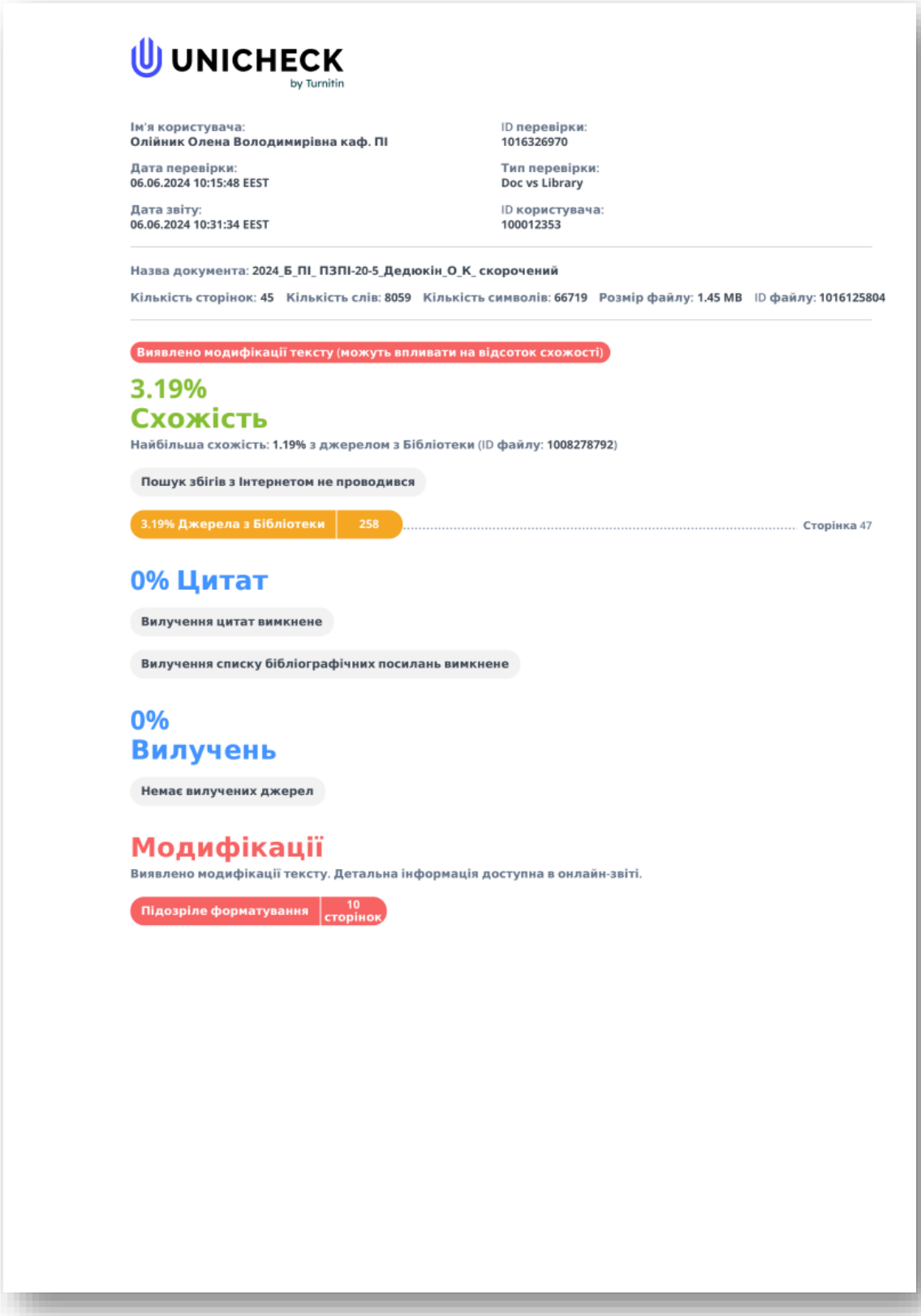
8. Patil S. Study of Container Technology with Docker. International Journal of Advanced Research in Science, Communication and Technology. 2021. С. 504–509. [Електронний ресурс] – URL: https://doi.org/10.48175/ijarsct-1283 (дата звернення: 13.05.2024).

9. Azure Storage Documentation [Електронний ресурс] – URL: https://learn.microsoft.com/en-us/azure/storage/ (дата звернення: 13.05.2024).

10. JWT.io. JSON Web Tokens Introduction [Електронний ресурс] – URL: https://jwt.io/introduction#:~:text=JSON%20Web%20Token%20(JWT)%20is,because%20it%20is%20digitally%20signed. (дата звернення: 13.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016326970

Дата перевірки:
06.06.2024 10:15:48 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
06.06.2024 10:31:34 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ ПЗПІ-20-5_Дедюкін_О_К_ скорочений

Кількість сторінок: 45 Кількість слів: 8059 Кількість символів: 66719 Розмір файлу: 1.45 MB ID файлу: 1016125804

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.19%
Схожість

Найбільша схожість: 1.19% з джерелом з Бібліотеки (ID файлу: 1008278792)

Пошук збігів з Інтернетом не проводився

3.19% Джерела з Бібліотеки 258 ..... Сторінка 47

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

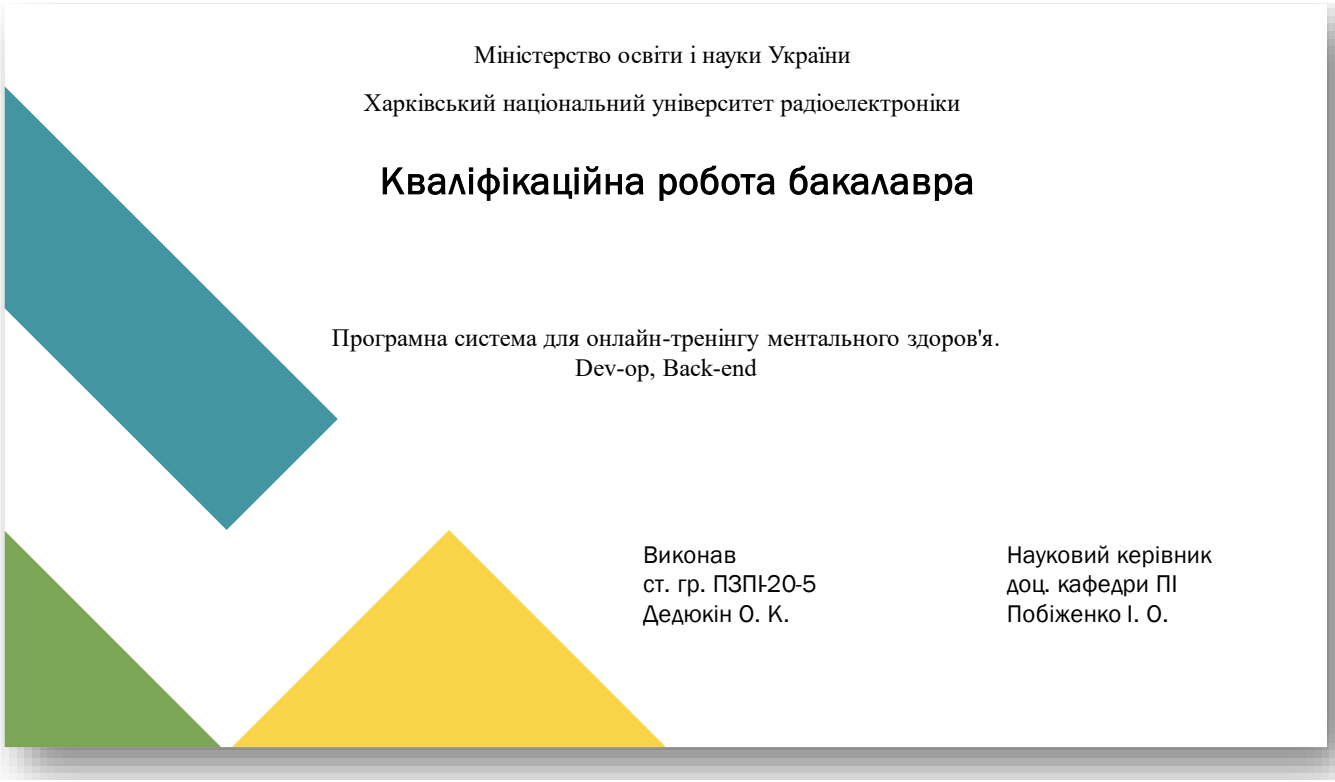
Модифікації

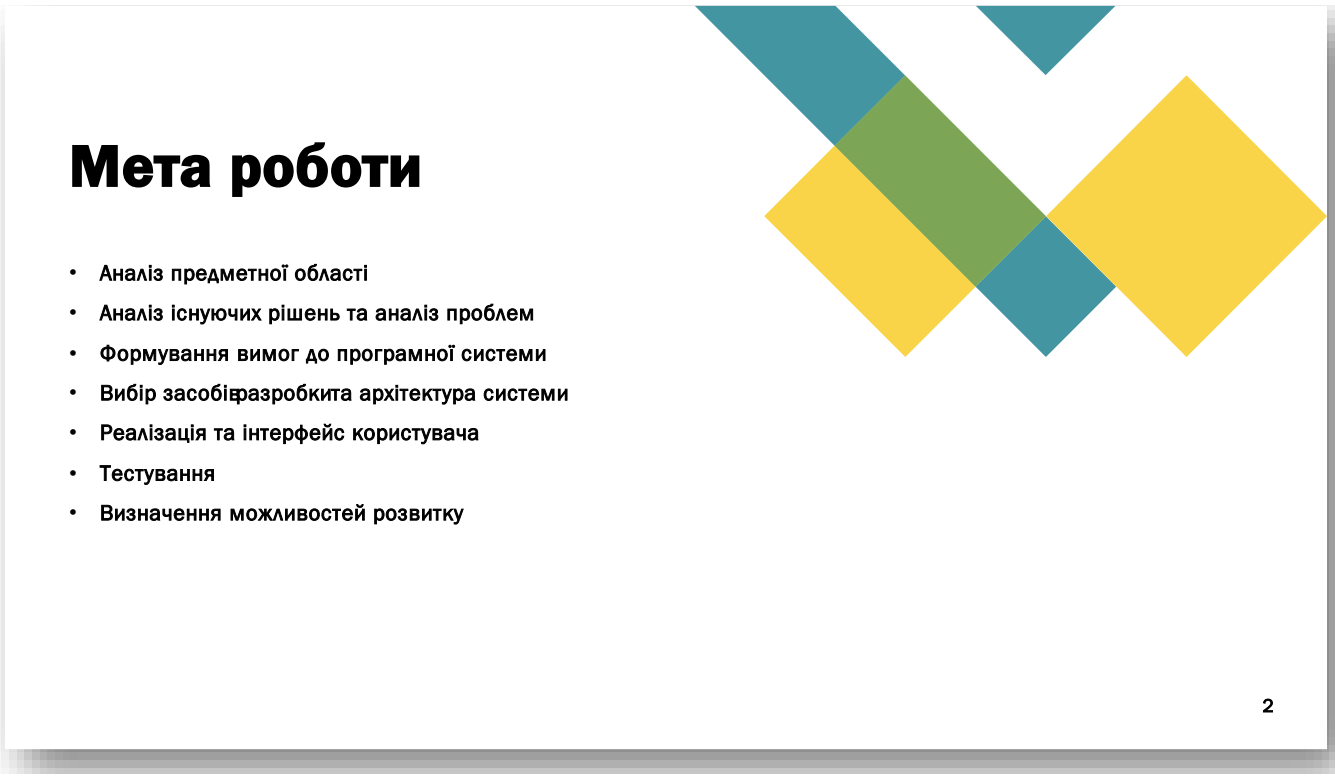
Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 10 сторінок

ДОДАТОК Б

Слайди презентації





Аналіз предметної області

- Ментальне здоров'я впливає на якість життя, продуктивність та соціальну інтеграцію.
- Пандемія COVID-19 та інші соціальні кризи загострили проблеми ментального здоров'я.
- Люди різного віку та соціального статусу, які хочуть підвищити стійкість до стресу та покращити емоційне благополуччя.

3

Існуючі рішення

	Рішення	Переваги	Недоліки
HEADSPACE	Мобільний додаток з медитації	Структуровані курси медитації, великий вибір тематичних медитацій, додаткові ресурси, інтерактивні вправи та трекери прогресу.	Висока вартість підписки, більшість контенту доступна лише в преміум-версії, неефективний для серйозних ментальних проблем.
betterhelp	Онлайн-платформа для відеоконсультацій	Широкий вибір спеціалістів , можливість змінити терапевта в будь-який момент, гнучкий графік консультацій , високий рівень конфіденційності та безпеки даних.	Висока вартість, труднощі з налагодженням емоційного зв'язку онлайн, обмеження деяких послуг для певних регіонів.
verywell mind	Освітній веб-сайт.	Велика кількість якісних статей та ресурсів, доступ до матеріалів від експертів, безкоштовний доступ до більшості контенту, регулярне оновлення матеріалів.	Відсутність інтерактивних елементів та індивідуального підходу, складність пошуку інформації через великий обсяг, відсутність можливості зв'язку з фахівцем.

4

Аналіз проблем

- Фінансові бар’єри
- Обмежений функціонал
- Обмежені можливості взаємодії

5

Формування вимог до програмної системи

- Функціональні вимоги
- Вимоги до безпеки
- Вимоги до масштабованості
- Вимоги до CI\CD

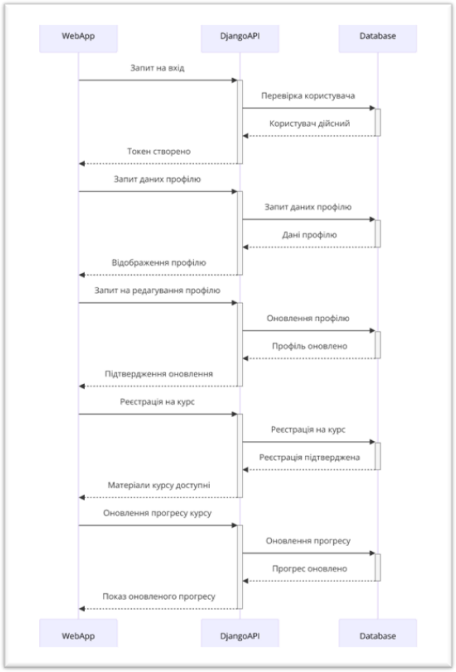
6

Вибір засобів разробки

7

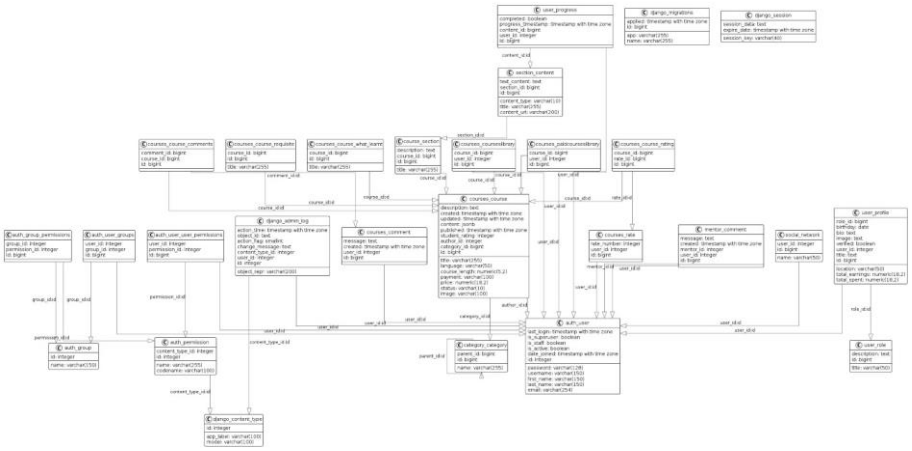
Архітектура системи

Веб-серверну архітектуру було обрано для побудови онлайн системи ментального здоров'я, що складається з додатку, API побудованого на фреймворку Django, та бази даних.



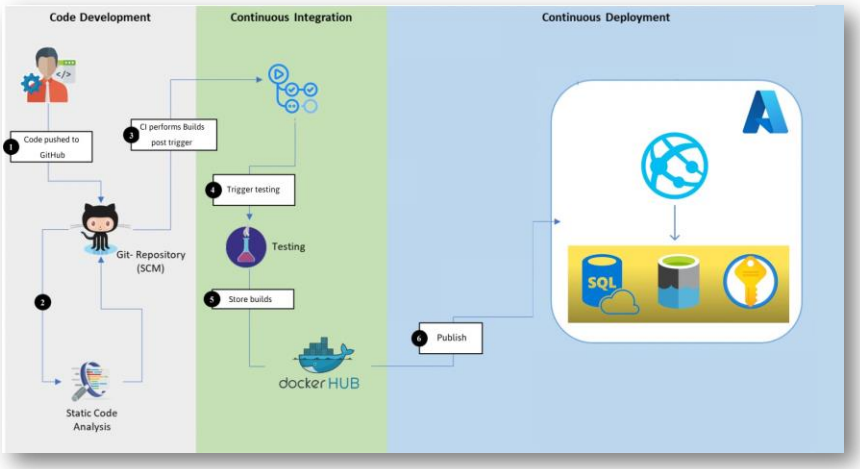
8

Архітектура системи



9

Архітектура системи



10

Приклад реалізації

```python
class CourseSerializer(serializers.ModelSerializer):
    author_name = serializers.CharField(source='author.first_name', read_only=True)
    what_learnt = CourseWhatLearntSerializer(many=True, read_only=True)
    sections = CourseSectionSerializer(many=True, read_only=True)

    ratings_count = serializers.SerializerMethodField()
    num_purchases = serializers.SerializerMethodField()
    purchased = serializers.SerializerMethodField()
    length = serializers.SerializerMethodField()

    class Meta:
        model = CoursesCourse
        fields = '__all__'

    def get_purchased(self, obj):
        user = self.context.get('user')
        if not user or not user.is_authenticated:
            return False
        return CoursesPaidcourseslibrary.objects.filter(user=user, course=obj).exists()

    def get_ratings_count(self, obj):
        return CoursesCourseRating.objects.filter(course=obj).count()

    def get_num_purchases(self, obj):
        return CoursesPaidcourseslibrary.objects.filter(course=obj).count()

    def get_length(self, obj):
```

```python
class CourseApiView(viewsets.ModelViewSet):
    queryset = CoursesCourse.objects.all()
    serializer_class = CourseSerializer
    filter_backends = (filters.DjangoFilterBackend,)
    filterset_class = CourseModelFilter
    http_method_names = ['get', 'post', 'put', 'patch', 'delete']

    def get_queryset(self):
        queryset = super().get_queryset().annotate(
            ratings_count=Count('coursescourserating'),
            num_purchases=Count('coursespaidcourseslibrary')
        )

        ordering = self.request.query_params.get('ordering')
        if ordering:
            queryset = queryset.order_by(ordering)

        return queryset

    def get_serializer_context(self):
        context = super().get_serializer_context()
        context['user'] = self.request.user
        return context
```

11

Приклад реалізації

```python
def upload_image_to_blob(image_file: bytes, path: str) -> str:
    """
    Uploads an image file to Azure Blob Storage and returns the URL of the uploaded image.

    :param image_file: Image file to upload
    :param path: Path in the blob storage where the image will be saved
    :return: URL of the uploaded image
    """
    try:
        blob_service_client = BlobServiceClient.from_connection_string(AZURE_STORAGE_CONNECTION_STRING)
        blob_client = blob_service_client.get_blob_client(
            container=AZURE_IMAGES_STORAGE_CONTAINER_NAME,
            blob=f"{path}/profile_image.png"
        )
        blob_client.upload_blob(image_file, overwrite=True)
        return blob_client.url
    except Exception as e:
        logger.error(f"Failed to upload image to blob storage: {e}")
        raise
```

```python
@lru_cache(maxsize=128)
def get_video_length_from_azure(content_url: str) -> int:
    """
    Retrieves the length of a video stored in Azure Blob Storage.

    :param content_url: URL of the video in Azure Blob Storage
    :return: Length of the video in seconds, or 0 if not available or on error
    """
    if not content_url:
        return 0

    try:
        blob_service_client = BlobServiceClient.from_connection_string(AZURE_STORAGE_CONNECTION_STRING)
        blob_client = blob_service_client.get_blob_client(
            container=AZURE_VIDEOS_STORAGE_CONTAINER_NAME,
            blob=content_url.split(f"/{AZURE_VIDEOS_STORAGE_CONTAINER_NAME}/")[-1]
        )
        properties = blob_client.get_blob_properties()
        return int(properties.metadata.get('video_length', 0))
    except Exception as e:
        logger.error(f"Failed to get video length from Azure Blob Storage: {e}")
        return 0
```

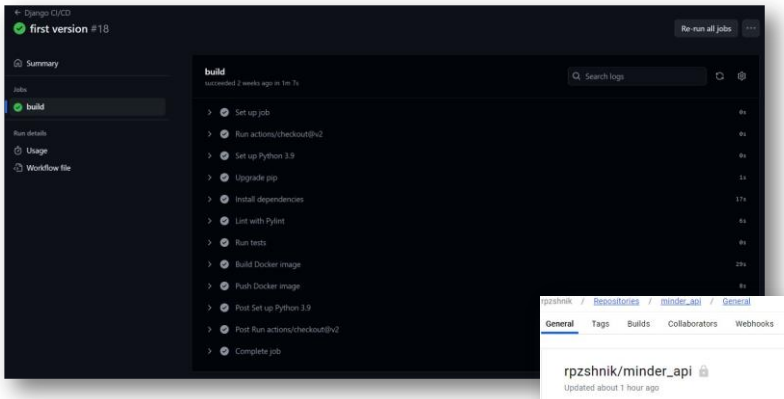
12

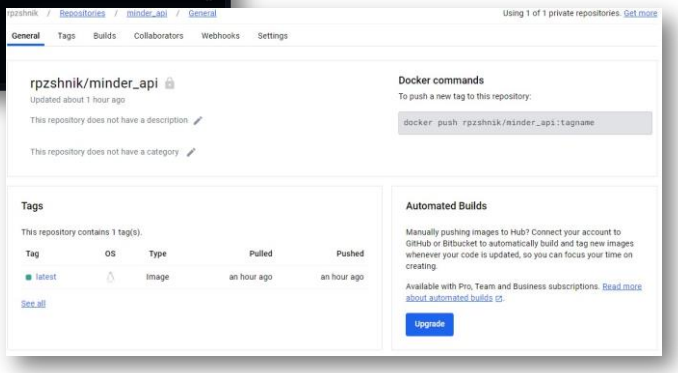
Приклад реалізації

```yaml
services:
  db:
    image: postgres:15
    restart: always
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
      PGUSER: postgres
    ports:
      - "5432:5432"
    volumes:
      - ./postgres_data:/var/lib/postgresql/data
      - ./sql:/docker-entrypoint-initdb.d/
      - ./minderAPI:/code/minderAPI
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U postgres" ]
      interval: 5s
      timeout: 5s
      retries: 5
  web:
    depends_on:
      db:
        condition: service_healthy
    build: .
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    environment:
      DJANGO_DB_NAME: ${POSTGRES_DB}
      DJANGO_SU_NAME: ${SUPERUSER_NAME}
      DJANGO_SU_EMAIL: ${SUPERUSER_EMAIL}
      DJANGO_SU_PASSWORD: ${SUPERUSER_PASSWORD}
      DB_HOST: db
      AZURE_STORAGE_CONNECTION_STRING: ${AZURE_STORAGE_CONNECTION_STRING}
    restart: always
```

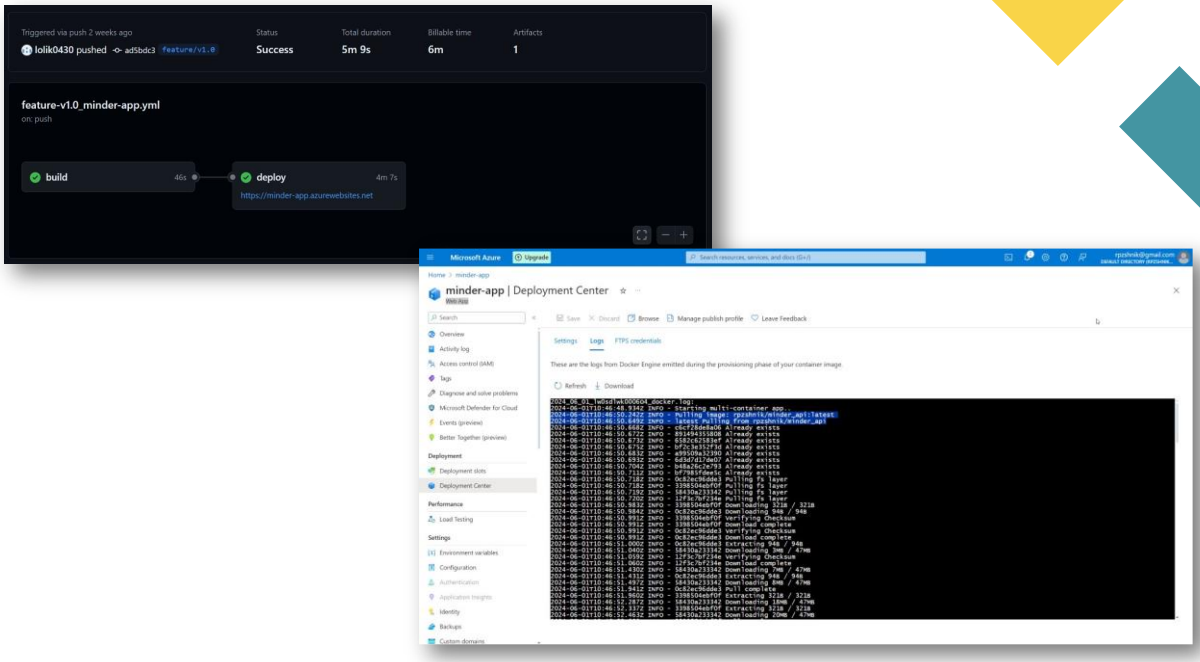
```yaml
name: Django CI/CD

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    - name: Set up Python 3.9
      uses: actions/setup-python@v2
      with:
        python-version: 3.9

    - name: Upgrade pip
      run: |
        pip install --upgrade pip

    - name: Install dependencies
      run: |
        pip install -r requirements.txt

    - name: Lint with Pylint
      run: |
        pylint minderAPI/*.py

    - name: Run tests
      run: |
        pytest ./tests

    - name: Build Docker image
      run: docker build -t rpzshnik/minder_api .

    - name: Push Docker image
      if: github.ref == 'refs/heads/master'
      run: |
        echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{ secrets.DOCKER_USERNAME }}" --password-stdin
        docker push rpzshnik/minder_api
```

13

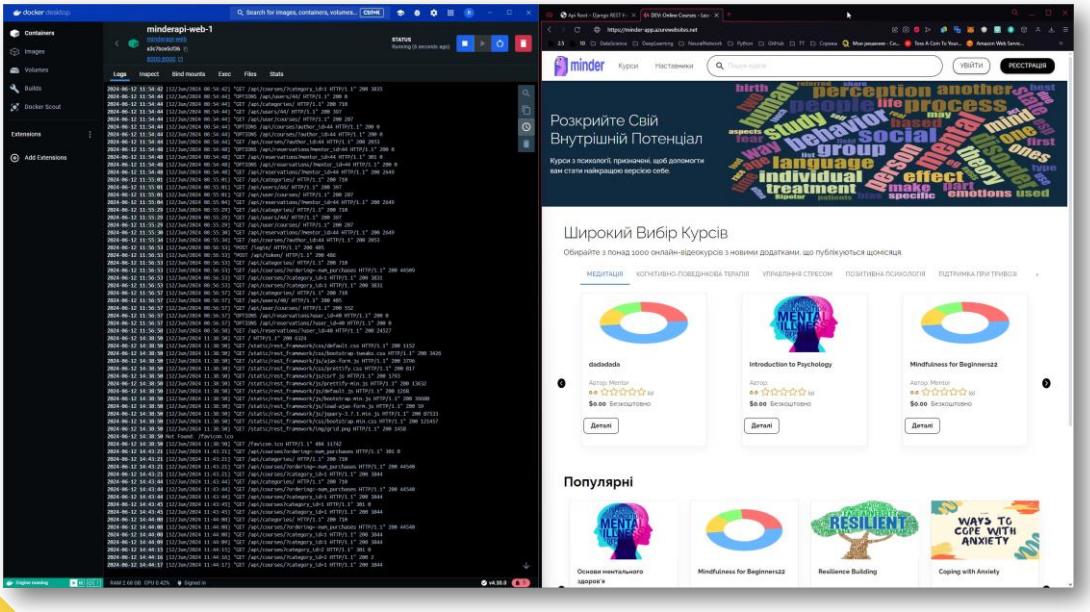




14


Triggered via push 2 weeks ago
lolik0430 pushed -o- ad5bdc3 feature/v1.0
Status
Success
Total duration
5m 9s
Billable time
6m
Artifacts
1
feature-v1.0_minder-app.yml
on: push
build
46s
deploy
4m 7s
https://minder-app.azurewebsites.net
Microsoft Azure
Upgrade
minder-app | Deployment Center
Settings
Logs
FTPS credentials
These are the logs from Docker Engine emitted during the provisioning phase of your container image.
Refresh
Download

15


minderapi-web-1
minder
Розкрийте Свій Внутрішній Потенціал
Курси з психології, призначені, щоб допомогти вам стати найкращою версією себе.
Широкий Вибір Курсів
Обирайте з понад 1000 онлайн-відеокурсів з новими додатками, що публікуються щомісяця.
Популярні
Introduction to Psychology
Mindfulness for Beginners22
Resilience Building
Coping with Anxiety

16

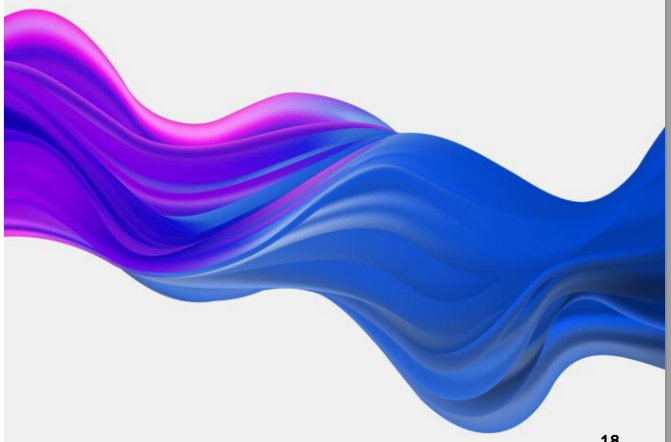
Тестування

ID	Description	Test Space	Input Data	Action	Expected Result	Actual Result	Test Result
1	Відстеження прогресу користувача на курсі	Локально	JWT token, ID контенту курсу	Відзначити контент курсу як завершений	Прогрес користувача оновлено і відсоток перераховано	Співпадає	+
2	Розрахунок довжини курсу		Курс з розділами	Запит на інформацію про курс	Довжина курсу обчислена як сума довжин усіх розділів	Співпадає	+
3	Реєстрація нового користувача		Дані користувача	Надіслати форму реєстрації	Користувач успішно зареєстрований	Співпадає	+
4	Оновлення профілю користувача		Деталі профілю, файл зображення	Оновлення профілю з зображенням	Профіль оновлено з новим отриманим URL зображення	Співпадає	+

17

Можливості розвитку

- Розширення функціоналу
- Інтеграція з іншими сервісами
- Покращення користувацького досвіду
- Розвиток професійних інструментів
- Наукові дослідження та інновації



18

Дякую за увагу

ДОДАТОК В

Специфікація програмного забезпечення

ЗМІСТ

1 ВСТУП

1.1 Мета

Метою цього документа зі специфікації вимог до програмного забезпечення (Software Requirements Specification - SRS) є визначення основних функціональних та нефункціональних вимог до програмної системи для онлайн-тренінгу ментального здоров'я. Документ призначений для команди розробників, проектних менеджерів, а також замовників і користувачів системи, щоб забезпечити чітке розуміння функціоналу, очікуваних характеристик і контексту використання програмного продукту. Він слугуватиме основою для подальшого проектування системи, забезпечуючи деталізацію вимог і сприяючи взаєморозумінню між усіма зацікавленими сторонами у процесі розробки.

Цей документ враховує встановлені цілі та завдання програмної системи, що включають забезпечення доступу до онлайн-тренінгів, моніторинг прогресу користувачів у покращенні ментального здоров'я, інтеграцію з календарем для планування сесій, а також створення індивідуальних планів розвитку.

1.2 Сфера застосування

Програмний продукт, що розробляється, – це "Система онлайн-тренінгу ментального здоров’я". Вона призначена для надання інтерактивних тренінгових модулів, інструментів самооцінки та персоналізованих порад з підтримки ментального здоров'я користувачів. Продукт забезпечить користувачам доступ до курсів, вправ та рекомендацій, спрямованих на покращення ментального благополуччя, але не буде заміною професійної психотерапевтичної допомоги або лікування.

Програма орієнтована на підвищення обізнаності про ментальне здоров'я, навчання методам самодопомоги та розвитку навичок стресостійкості. Вона буде надавати користувачам індивідуальні плани розвитку, інтегровані з календарем для зручності планування, та засоби для відстеження особистого прогресу в покращенні ментального здоров'я.

1.3 Визначення, акроніми та скорочення

Ментальне здоров'я – стан благополуччя, у якому індивід реалізує свої можливості, може справлятися з нормальними життєвими стресами, продуктивно працювати та вносити вклад у життя своєї спільноти.

Онлайн-тренінг – освітній курс або програма, що здійснюється через інтернет, надаючи користувачам можливість вивчати матеріал у віртуальному форматі.

SRS (Software Requirements Specification) – документ, який описує функції та обмеження програмного продукту, які необхідні для його розробки та інтеграції.

UI (User Interface) – інтерфейс, через який користувачі взаємодіють з програмною системою.

UX (User Experience) – загальний досвід користувача при взаємодії з продуктом або системою, включаючи як він сприймає її корисність, зручність та ефективність.

1.4 Список використаних джерел

1) IEEE Recommended Practice for Software Requirements Specifications Approved 30 November 2018

1.5 Огляд

Цей документ містить повну інформацію про проект: його функції, інтерфейс, обмеження. Вони відрізняються аудиторією, яку спрямовано зміст.

Глава «General Description» спрямована на власника проекту та містить огляд функціональності системи. Вона визначає вимоги, які є базою для специфікації у наступному розділі.

Глава «Specific Requirements» описує деталі пов'язані з функціональністю системи, написані технічною мовою розуміння розробниками.

Розділ «Analysis Model» містить різні діаграми та моделі, які полегшують розуміння системи, ілюструючи її роботу та взаємодію з користувачем.

2 ЗАГАЛЬНИЙ ОПИС

Програмний продукт, який складається з бекенд та фронтенд частини, забезпечує онлайн-курси та тренінги з питань ментального здоров'я, з інтеграцією календаря для запису до ментора.

2.1 Перспектива продукту

Система є самостійною і не залежить від інших систем або продуктів. Все, що потрібно клієнту для користування системою – наявність браузера та стабільного підключення до мережі Internet.

Також для доступу до деяких функції, таких, як проходження курсу та онлайн-зустрічі з ментором обов’язково потрібна реєстрація за допомогою акаунту Google.

2.2 Функції продукту

Ця програмна система має надавати користувачу можливість зареєструватися та авторизуватися за допомогою акаунту Google. Кожен зареєстрований клієнт може приєднатися до будь-якого з представлених курсів, а також мати змогу пройти матеріали цього курсу. Курси можуть бути платні або безкоштовні, для приєднання до платного курсу необхідно спочатку придбати цей курс, використовуючи систему оплати на сайті.

FE-1: Реєстрація та авторизація.

FE-2: Перегляд профілю.

FE-3: Редагування профілю.

FE-4: Приєднання до курсу та його проходження.

FE-5: Відстеження прогресу проходження курсу.

FE-6: Перегляд інформації про курс.

FE-7: Перегляд інформації про ментора.

FE-8: Бронювання часу для відео-консультації.

FE-9: Можливість відео-консультації з ментором.

FE-10: Перегляд списку пройдених курсів.

2.3 Характеристики користувача

Психічний стан: Користувачі можуть мати різний рівень психічного благополуччя, включаючи осіб зі стресом, тривогою, депресією або просто бажаючих підтримувати своє ментальне здоров'я.

Технічні вміння: Рівень технічної підготовки користувачів може коливатися від новачків до досвідчених користувачів інтернету та комп'ютерів.

Вік: Користувачі можуть бути різного віку, від підлітків до дорослих і літніх людей.

2.4 Загальні обмеження

Недостатня кількість часу для повної реалізації запланованого функціоналу. Необхідність постійно підтримувати серверну частину системи.

2.5 Припущення та залежності

Повноцінне користування програмною системою можливо лише при наявності базових навичок користування браузером та підключення до мережі Internet.

3 СПЕЦИФІЧНІ ВИМОГИ

3.1 Вимоги до зовнішнього інтерфейсу

3.1.1 Користувацькі інтерфейси

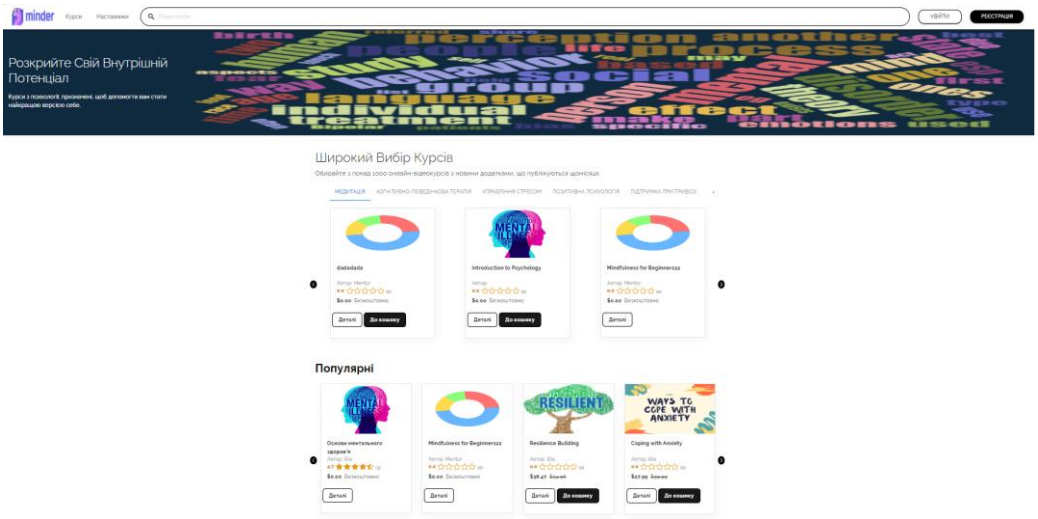


Рисунок 3.1 – Головна сторінка для незареєстрованого користувача

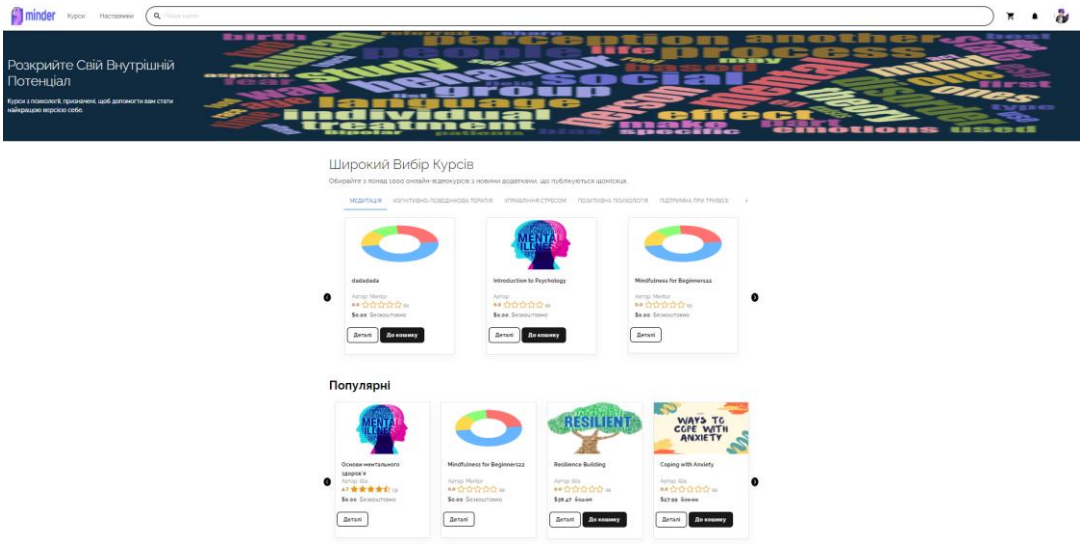


Рисунок 3.2 – Головна сторінка для авторизованого користувача

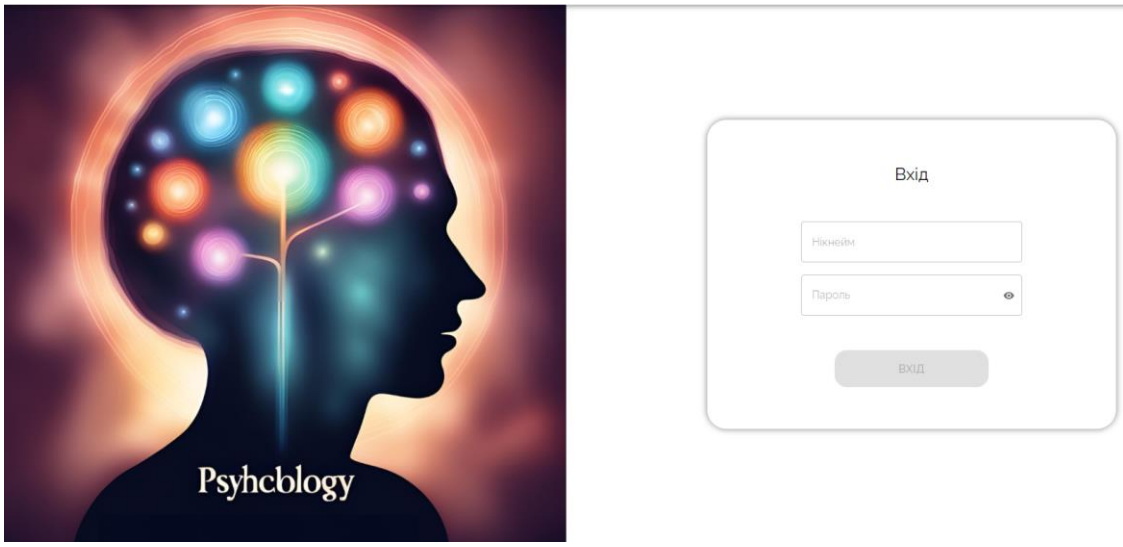


Рисунок 3.3 – Сторінки для авторизації

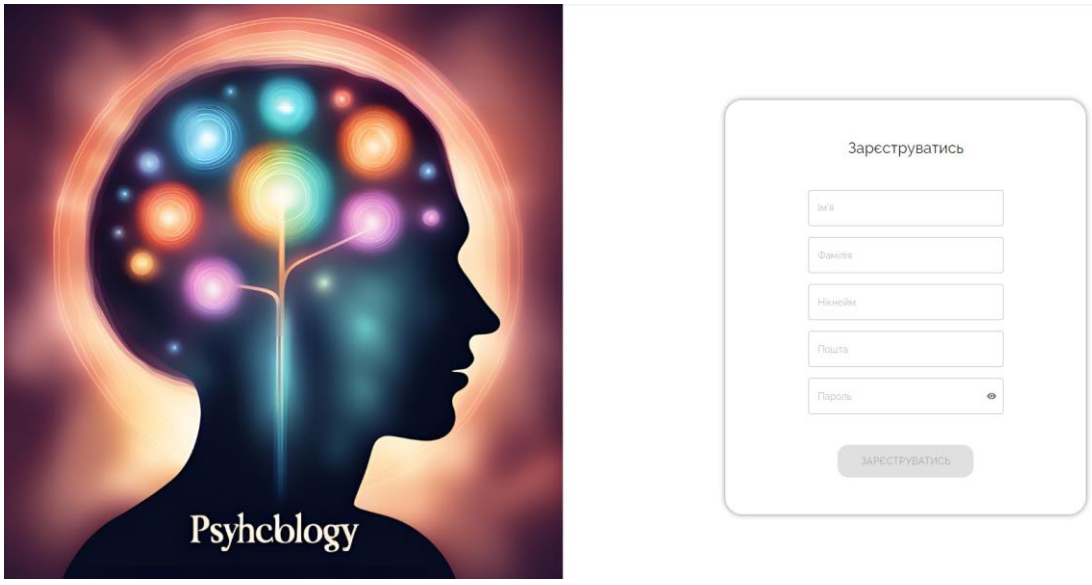


Рисунок 3.4 – Сторінки для реєстрації

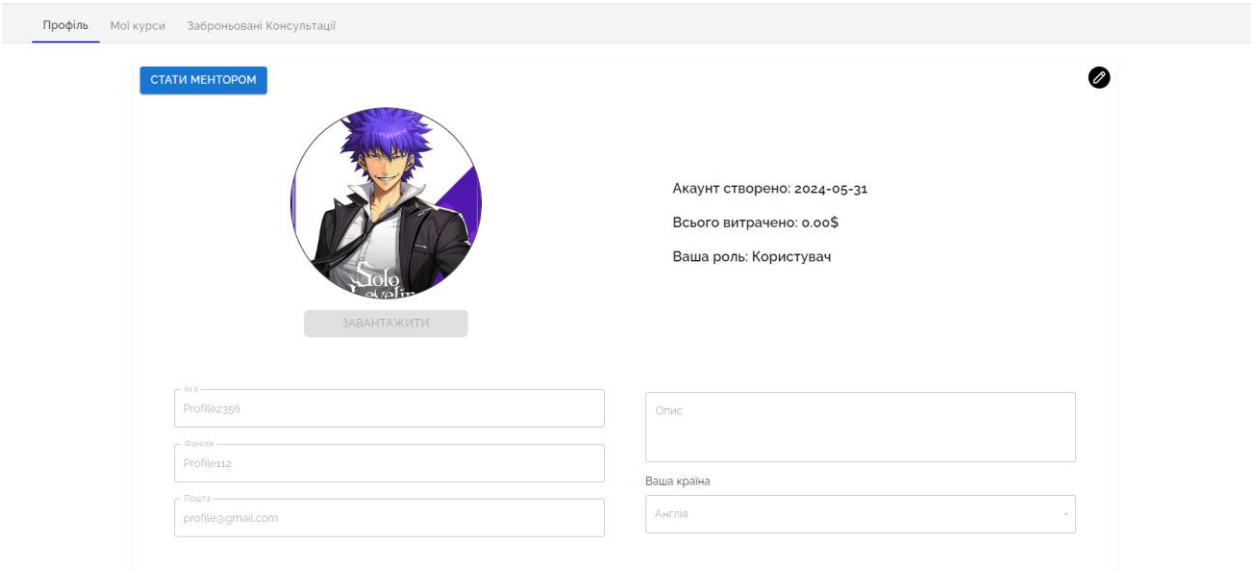


Рисунок 3.5 – Сторінка профілю авторизованого користувача

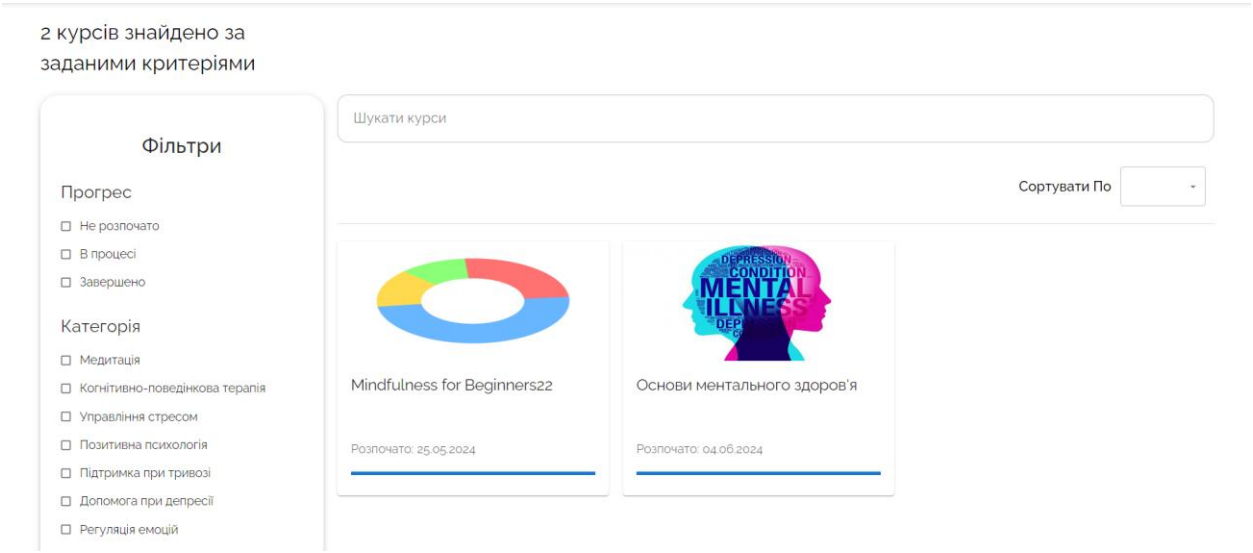


Рисунок 3.6 – Сторінка прогресу користувача по курсах

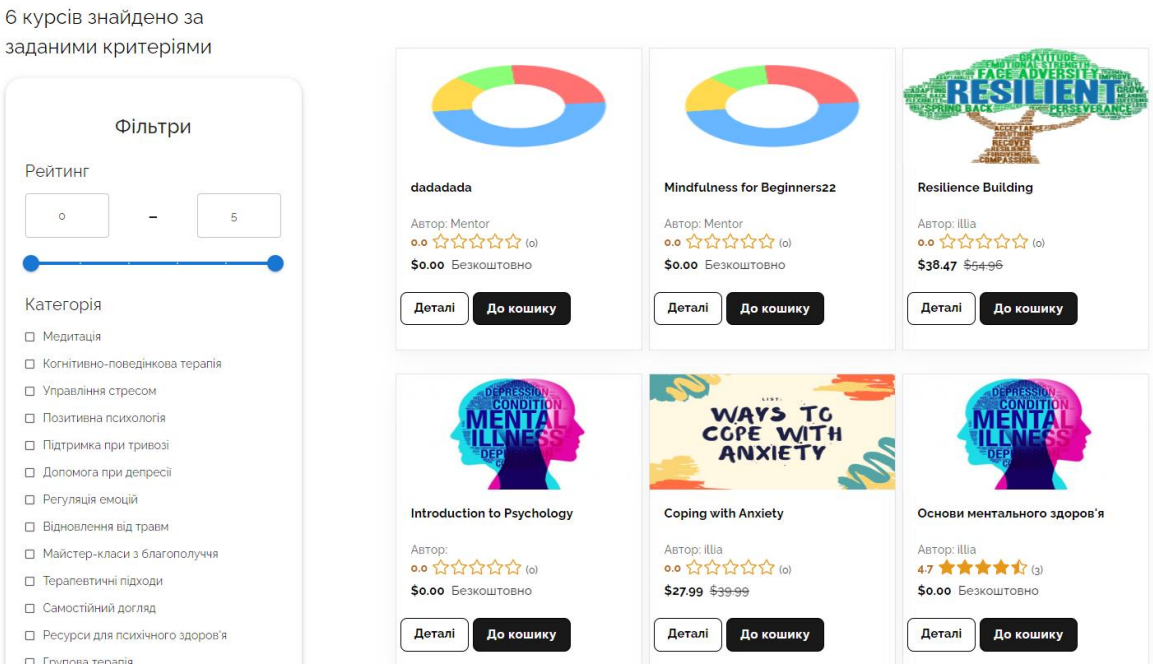


Рисунок 3.7 – Сторінка пошуку курсів на сайті

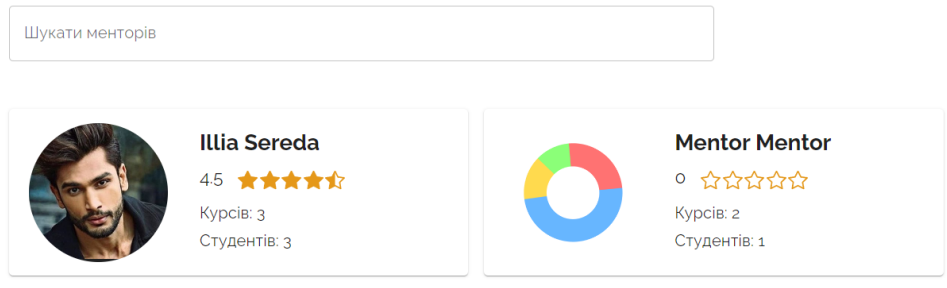


Рисунок 3.8 – Сторінка пошуку менторів

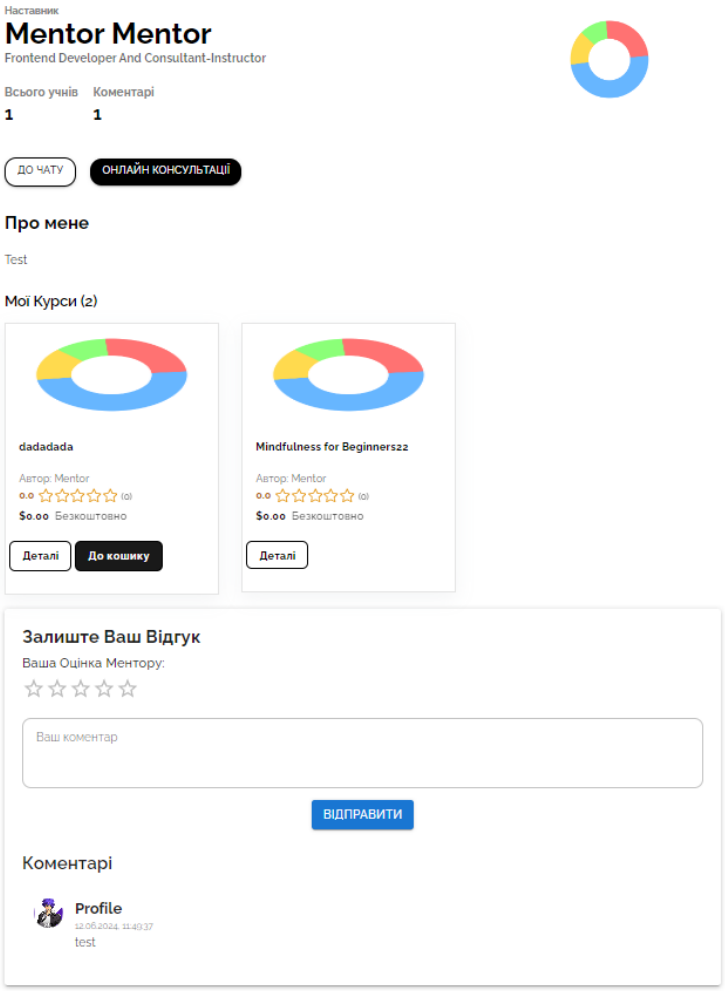


Рисунок 3.9 – Сторінка профілю ментора

Рисунок 3.10 – Сторінка чату з ментором

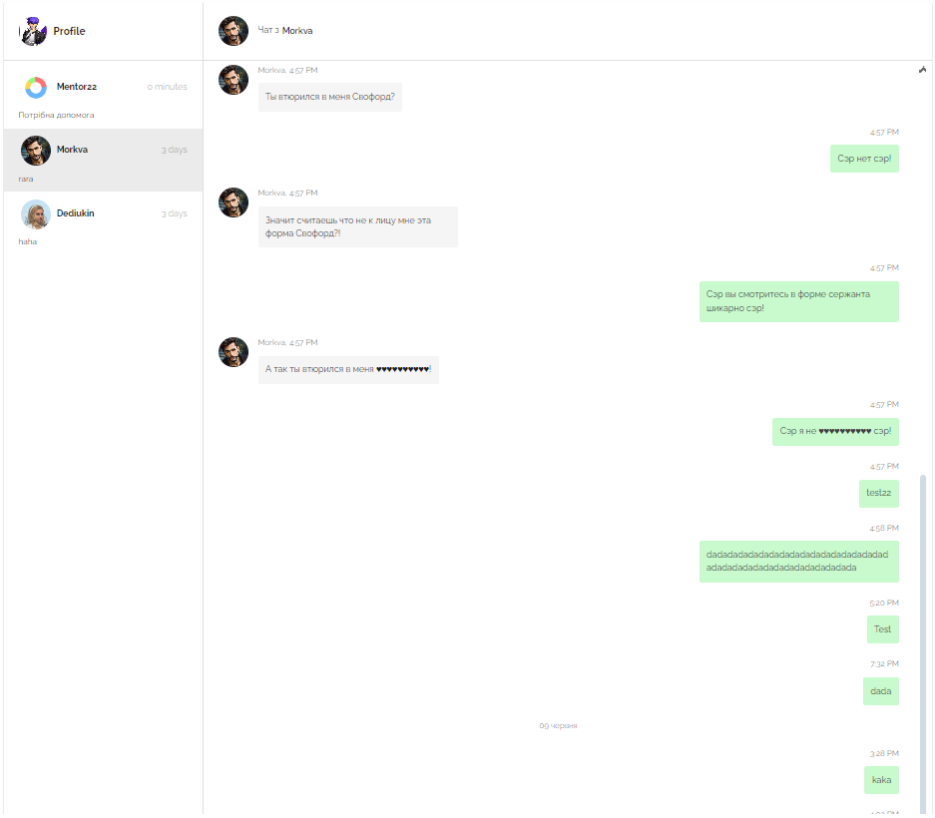


Рисунок 3.11 – Сторінка всіх доступних чатів

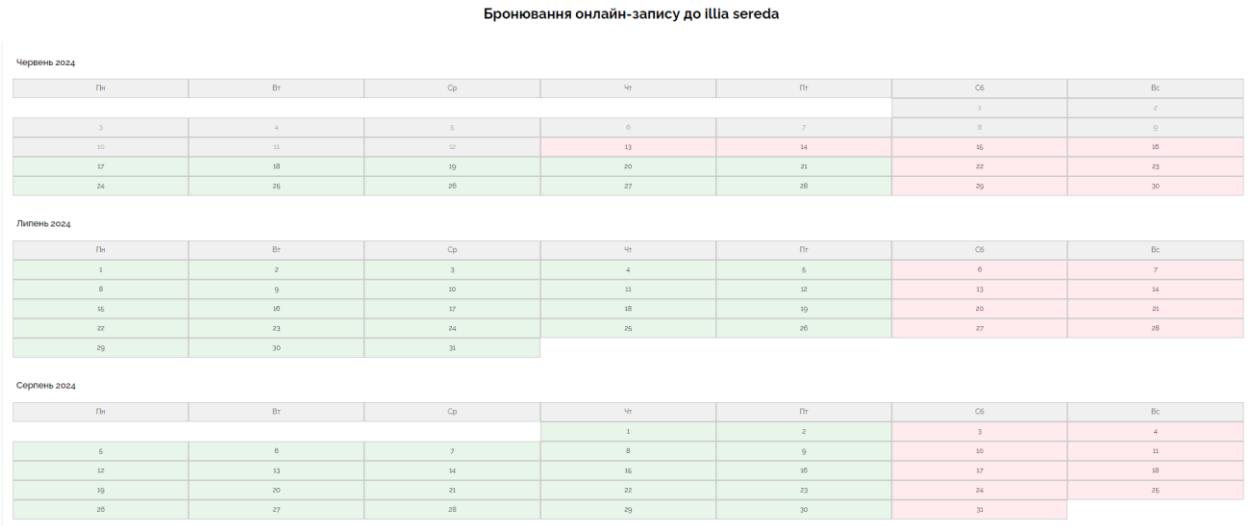


Рисунок 3.12 – Сторінка бронювання запису до ментора

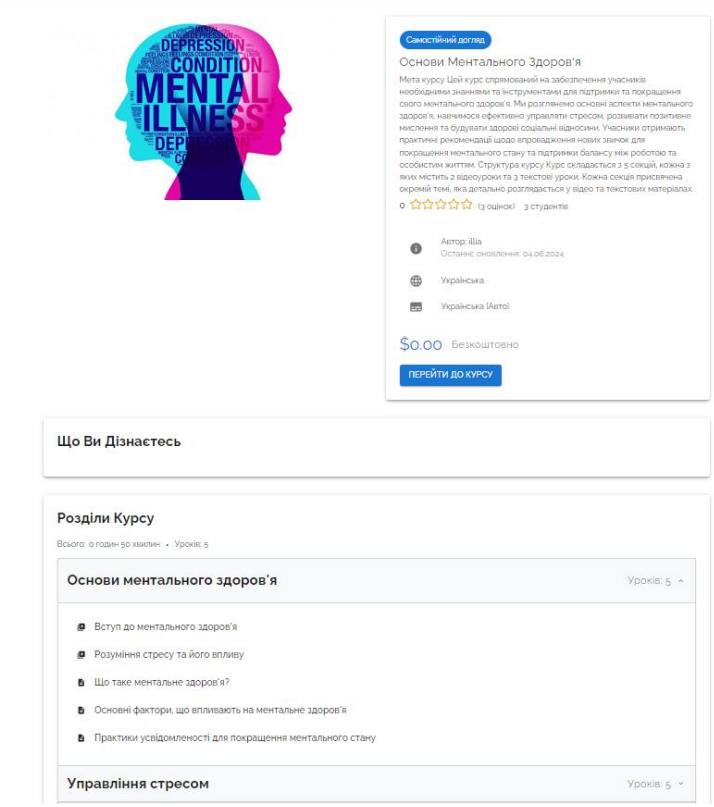


Рисунок 3.13 – Сторінка обраного курсу

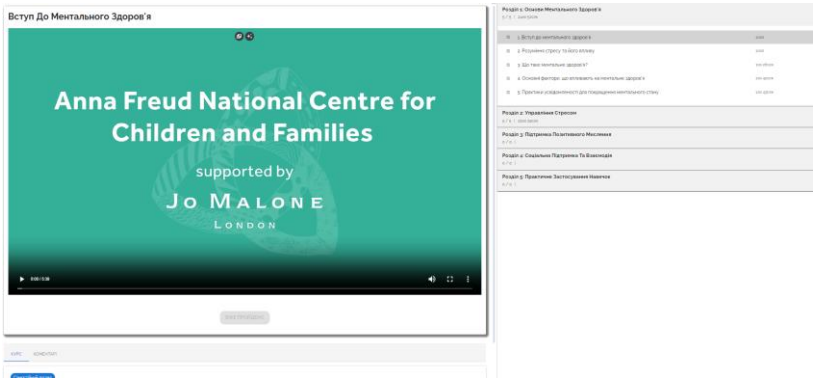


Рисунок 3.14 – Сторінка проходження курсу

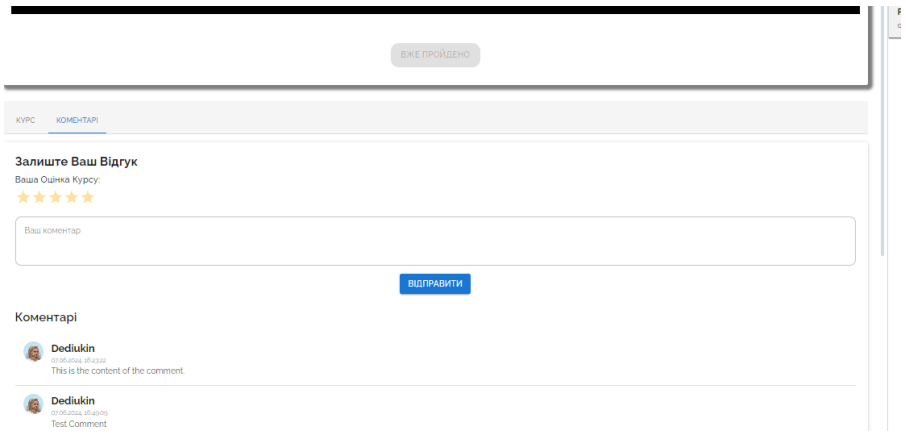


Рисунок 3.15 – Сторінки для авторизації та реєстрації

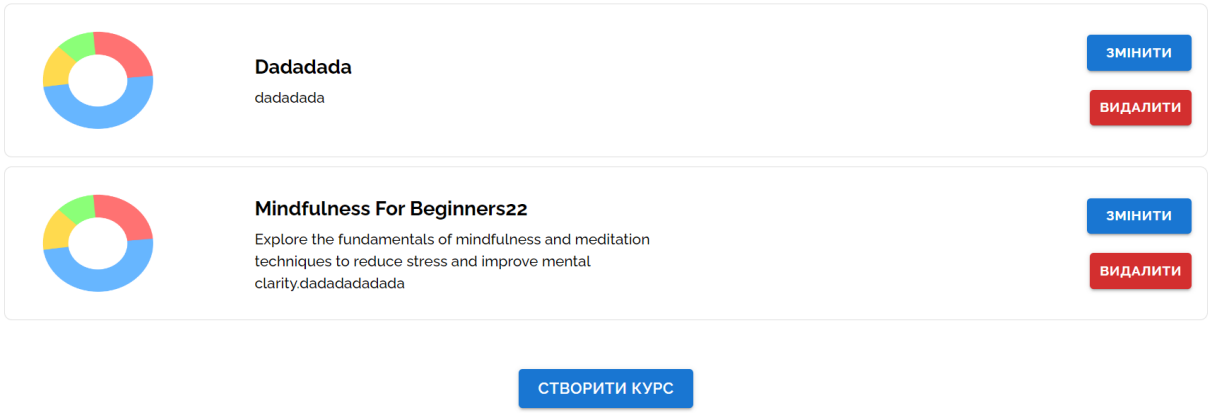


Рисунок 3.16 – Сторінка створення курсу

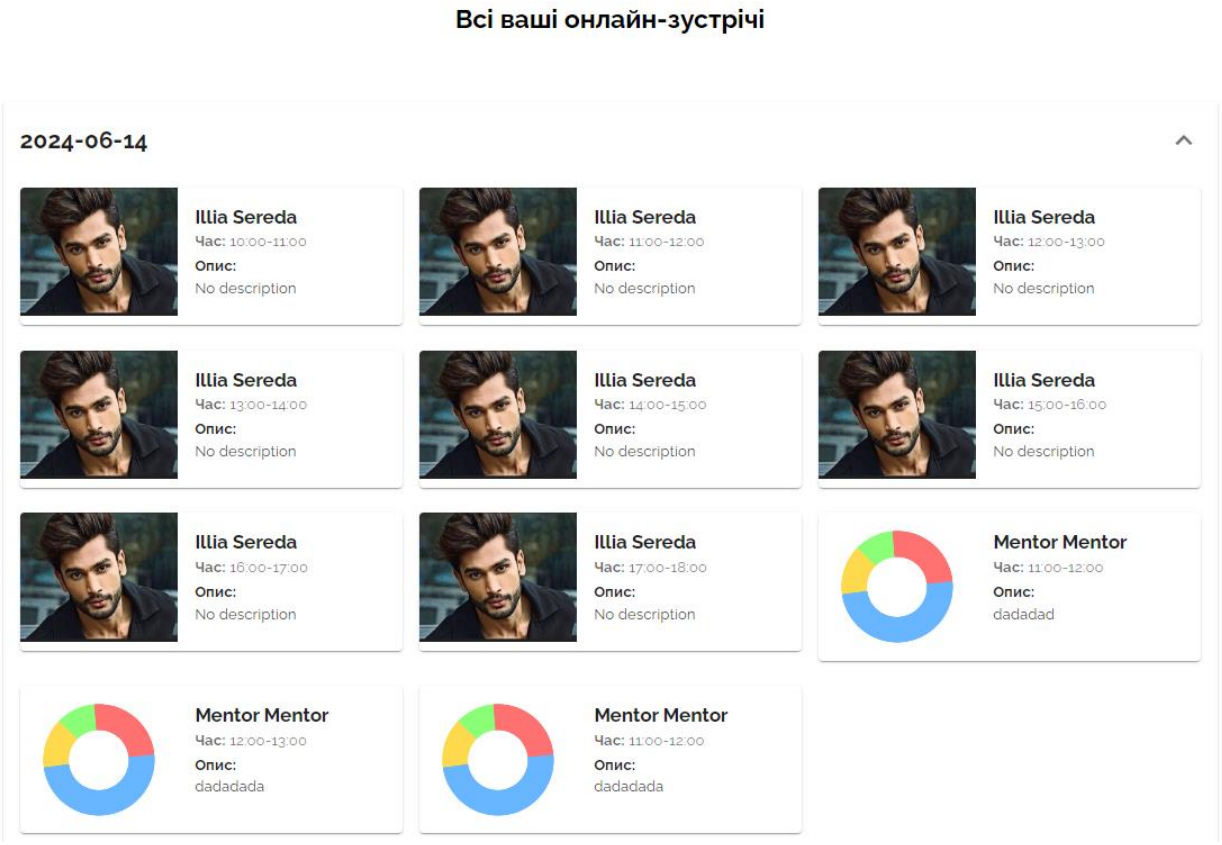


Рисунок 3.17 – Сторінка перегляду заброньованих записів

3.1.2 Апаратні інтерфейси

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36 з дозволом на використання cookie.

3.1.3 Інтерфейси програмного забезпечення

Користувач повинен мати браузер для використання програмної системи.

3.1.4 Комунікаційні інтерфейси

Зв’язок з сервером буде виконуватись за допомогою HTTP та HTTPS запитів.

3.2 Функціональні вимоги

3.2.1 FE-1 Реєстрація та авторизація

Користувачі повинні мати можливість зареєструватися на платформі та авторизуватися для доступу до своїх профілів, курсів та інших функцій.

Користувач вводить свої дані (ім'я, email, пароль) у форму реєстрації або авторизації.

Користувач підтверджує введення даних (наприклад, повторенням пароля або натисканням кнопки "Зареєструватися" або "Ввійти").

Система реєструє нового користувача (якщо користувач реєструється) або перевіряє наявність існуючого користувача (якщо користувач входить).

Система перевіряє правильність та повноту введених даних.

Система генерує токен авторизації для успішно авторизованих користувачів.

У разі успішної реєстрації користувач отримує повідомлення про підтвердження та перенаправлення на сторінку авторизації.

У разі успішного входу користувач отримує доступ до свого профілю та інших функцій платформи.

У разі помилки при реєстрації або вході користувач отримує відповідне повідомлення про помилку (наприклад, невірний пароль, неіснуючий email).

Система повинна чітко вказувати причину помилки при реєстрації або вході.

Користувачу надається можливість виправити помилку та спробувати знову.

Кількість спроб входу може бути обмежена для запобігання зловживанням.

3.2.2 FE-2 Перегляд профілю

Користувачі повинні мати можливість переглядати свій профіль, який містить їхню особисту інформацію, дані про навчання та іншу relevantny інформацію.

Користувач переходить на сторінку свого профілю.

Система завантажує та відображає дані профілю користувача.

Вихідні дані:

- ім'я та прізвище користувача;
- email;
- фотографію профілю (за бажанням);

- контактну інформацію (за бажанням);
- інформацію про освіту та досвід (за бажанням);
- перелік пройдених курсів;
- поточні курси;
- заплановані курси;
- нагороди та досягнення;
- перелік заброньованих консультацій;
- іншу релевантну інформацію.

Система повинна відображати повідомлення про помилку, якщо профіль користувача не може бути завантажений.

3.2.3 FE-3 Редагування профілю

Користувачі повинні мати можливість редагувати свій профіль, оновлюючи свою особисту інформацію, дані про навчання та іншу релевантну інформацію.

Вхідні дані:

- користувач переходить на сторінку редагування профілю;
- користувач змінює дані у відповідних полях;
- користувач підтверджує зміни;
- система валідує внесені зміни;
- система оновлює дані профілю користувача;

У разі успішного оновлення система відображає повідомлення про успіх та оновлений профіль користувача.

У разі помилки валідації система відображає відповідне повідомлення про помилку (наприклад, неправильний формат email, обов'язкове поле не заповнене).

3.2.4 FE-4 Приєднання до курсу та його проходження

Користувачі повинні мати можливість переглядати доступні курси, реєструватися на них та проходити навчальні матеріали.

Вхідні дані:

- користувач переглядає каталог курсів;

- користувач обирає курс, який хоче пройти;
- користувач натискає кнопку "Записатись на курс" або аналогічну.

Обробка:

- система перевіряє, чи відповідає користувач вимогам для реєстрації на курс.
- система реєструє користувача на курс та надає доступ до матеріалів.
- система відстежує прогрес користувача у проходженні курсу.

Після успішної реєстрації користувач отримує доступ до матеріалів курсу, таких як:

- відеолекції;
- текстові матеріали;
- тести;
- практичні завдання;
- система відображає прогрес користувача у курсі.

Система повинна повідомити користувача, якщо він не відповідає вимогам для реєстрації на курс.

Система повинна чітко вказувати на помилки під час проходження курсу (наприклад, незавершені завдання, нескладені тести).

3.2.5 FE-5 Відстеження прогресу проходження курсу

Користувачі повинні мати змогу відстежувати свій прогрес у проходженні курсів.

Користувач переходить на сторінку курсу, який він проходить.

Система завантажує та відображає інформацію про прогрес користувача, включаючи:

- завершені модулі;
- оцінки за пройдені тести та завдання;
- залишені матеріали.

Сторінка прогресу курсу відображає візуальне представлення прогресу (наприклад, смуга виконання, список модулів).

Користувач може легко визначити свій статус проходження курсу.

Система повинна відображати повідомлення про помилку, якщо інформацію про прогрес не може бути завантажено.

3.2.6 FE-6 Перегляд інформації про курс

Користувачі повинні мати можливість переглядати детальну інформацію про курс перед реєстрацією або під час його проходження.

Користувач переходить на сторінку інформації про курс.

Система завантажує та відображає детальну інформацію про курс, включаючи:

- назву курсу;
- опис курсу;
- тематичний план;
- мимоги до попередніх знань;
- тривалість курсу;
- ментора курсу;
- рейтинг курсу;
- секції курсу;
- відгуки користувачів (за бажанням).

Сторінка інформації про курс дозволяє користувачу ознайомитися з цілями, вмістом та структурою курсу.

Система повинна відображати повідомлення про помилку, якщо інформацію про курс не може бути завантажено.

3.2.7 FE-7 Перегляд інформації про ментора

Користувачі повинні мати можливість переглядати інформацію про ментора курсу.

Користувач переходить на профіль ментора або натискає кнопку "Детальніше" біля інформації про ментора в курсі.

Система завантажує та відображає інформацію про ментора, включаючи:

- ім'я та прізвище ментора;
- досвід та експертизу;
- контактну інформацію (за бажанням);
- відгуки користувачів (за бажанням);

Сторінка профілю ментора дозволяє користувачу дізнатися більше про його професійний бекграунд та можливості взаємодії.

Система повинна відображати повідомлення про помилку, якщо інформацію про ментора не може бути завантажено.

3.2.8 FE-8 Бронювання часу для відео-консультації

Користувачі, які навчаються на курсі з ментором, повинні мати можливість бронювати час для відео-консультації.

Вхідні дані:

- користувач переходить на сторінку бронювання консультацій;
- користувач обирає зручний час із розкладу ментора;
- користувач підтверджує бронювання;

Обробка:

- система перевіряє доступність обраного часу;
- система резервує обраний час для консультації;
- система надсилає повідомлення користувачеві та ментору.

Користувач отримує підтвердження бронювання консультації, що містить дату, час та посилання на відео-кімнату (якщо застосовується).

Система повинна повідомити користувача, якщо обраний час вже зайнятий.

Система повинна дозволити користувачу обрати інший час у разі помилки бронювання.

3.2.9 FE-9 Можливість відео-консультації з ментором

Користувачі повинні мати можливість провести відео-консультацію з ментором у запланований час.

Вхідні дані

- Користувач переходить за посиланням на відео-кімнату;
- Користувач підключається до відео-зв'язку.

Система забезпечує двосторонній відео- та аудіо-зв'язок між користувачем та ментором.

Система може дозволяти демонстрацію екрану користувача або ментора для кращої комунікації.

Система може записувати консультацію за згодою обох сторін.

Відео-консультація дозволяє користувачу спілкуватися з ментором віч-на-віч, отримувати персональні поради та роз'яснення.

Система повинна повідомити користувача та ментора у разі виникнення технічних проблем із відео-зв'язком.

Система повинна пропонувати альтернативні способи проведення консультації (наприклад, перехід на телефонний дзвінок).

3.2.10 FE-10 Перегляд списку пройдених курсів

Користувачі повинні мати можливість переглядати список курсів, які вони вже пройшли.

Користувач переходить на сторінку "Мої курси" або аналогічний розділ.

Система завантажує та відображає список курсів, які користувач успішно завершив.

Список пройдених курсів відображає назви курсів, дати завершення, отримані оцінки (за наявності) та сертифікати (за наявності).

Система повинна відображати повідомлення про помилку, якщо список курсів не може бути завантажений.

3.3 Use Cases діаграма

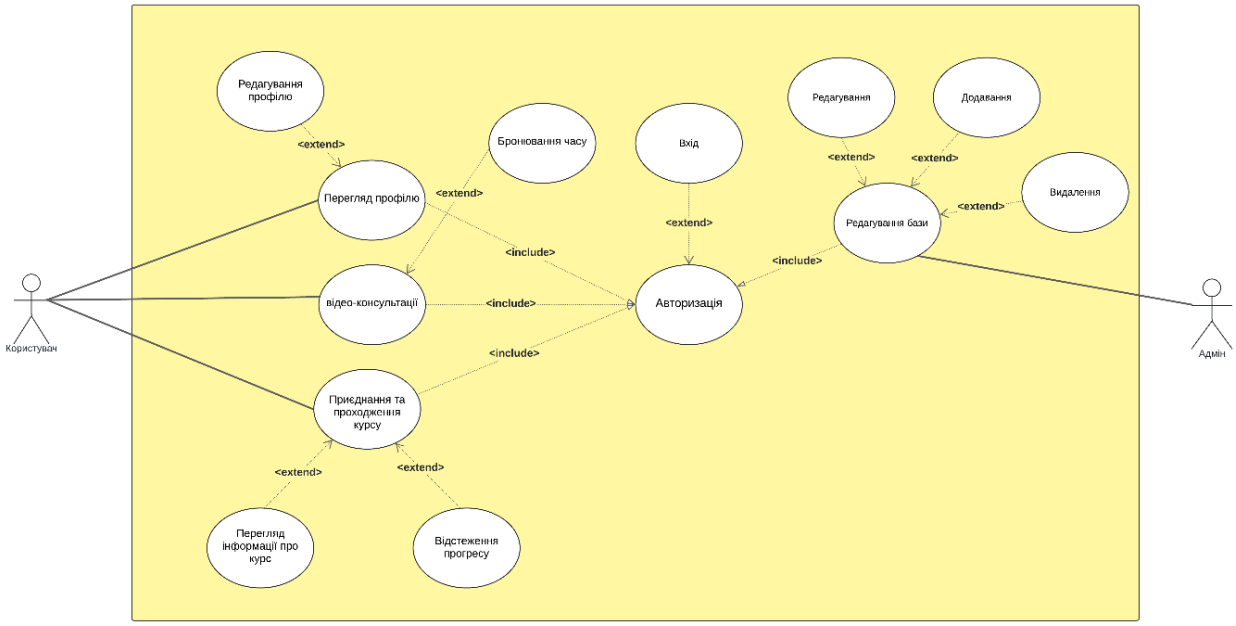


Рисунок 3.1 – Use Case діаграма програмного модулю

3.4 Класи та об’єкти

Модель користувача - User

- id;
- last_login;
- is_superuser;
- first_name;
- last_name;
- is_staff;
- date_joined;
- username;
- name;
- email;
- password;
- role;
- location;
- profile_info;

- total_earnings;
- total_spent;
- verified.

Модель курсів – CoursesCourse:

- id;
- title;
- description;
- created;
- language;
- courth_length;
- payment;
- price;
- content;
- author;
- category;
- student_rating;
- image.

3.5 Нефункціональні вимоги

3.5.1 Продуктивність

Час відповіді сервісу має бути меншим ніж 1 с, щоб його використання користувачем було комфортним.

3.5.2 Надійність

Користувач не повинен бачити повідомлень клієнта про помилки. Усі помилки мають бути виловлені й відображатися у вигляді повідомлення на сторінці сайту.

3.5.3 Доступність

Сервіс має бути доступний у будь-який час доби. Користувач не повинен загубити свою інформацію, оскільки всі дані зберігатимуться на сервері.

3.5.4 Безпека

Усі вхідні дані мають перевірятися на відповідність необхідним типам даних та валідуватись на стороні клієнту та серверу. Усі помилки мають логуватися.

3.5.5 Обслуговуваність

Проєкт має бути розділений на частини: сервер (back-end), клієнтська частина (front-end), а також база даних, де зберігатиметься вся інформація про користувачів, курси та менторів.

3.5.6 Портативність

Сервіс абсолютно портативний, оскільки являє собою веб-сайт, написаний мовою JavaScript за допомогою фреймворка React, який буде доступний на будь-

якому комп'ютері з браузером із підтримкою JavaScript і наявністю підключення до мережі Internet.

3.6 Зворотні вимоги

Програмна система не повинна допускати не зареєстрованого користувача до проходження курсу.

Програмна система не повинна допускати користувача до наступного етапу курсу, якщо користувач не завершив попередній.

3.7 Обмеження проектування

Дизайн сервісу має бути легким у розумінні. Так само дизайн не повинен заважати користувачеві проходити курс і використовувати навігацію.

3.8 Логічні вимоги до бази даних

База даних має зберігати інформацію про користувачів, їхні курси та прогрес. Доступ до неї буде здійснюватися при кожному запиті до сервера.

4 АНАЛІЗ МОДЕЛЕЙ

4.1 Діаграма послідовності

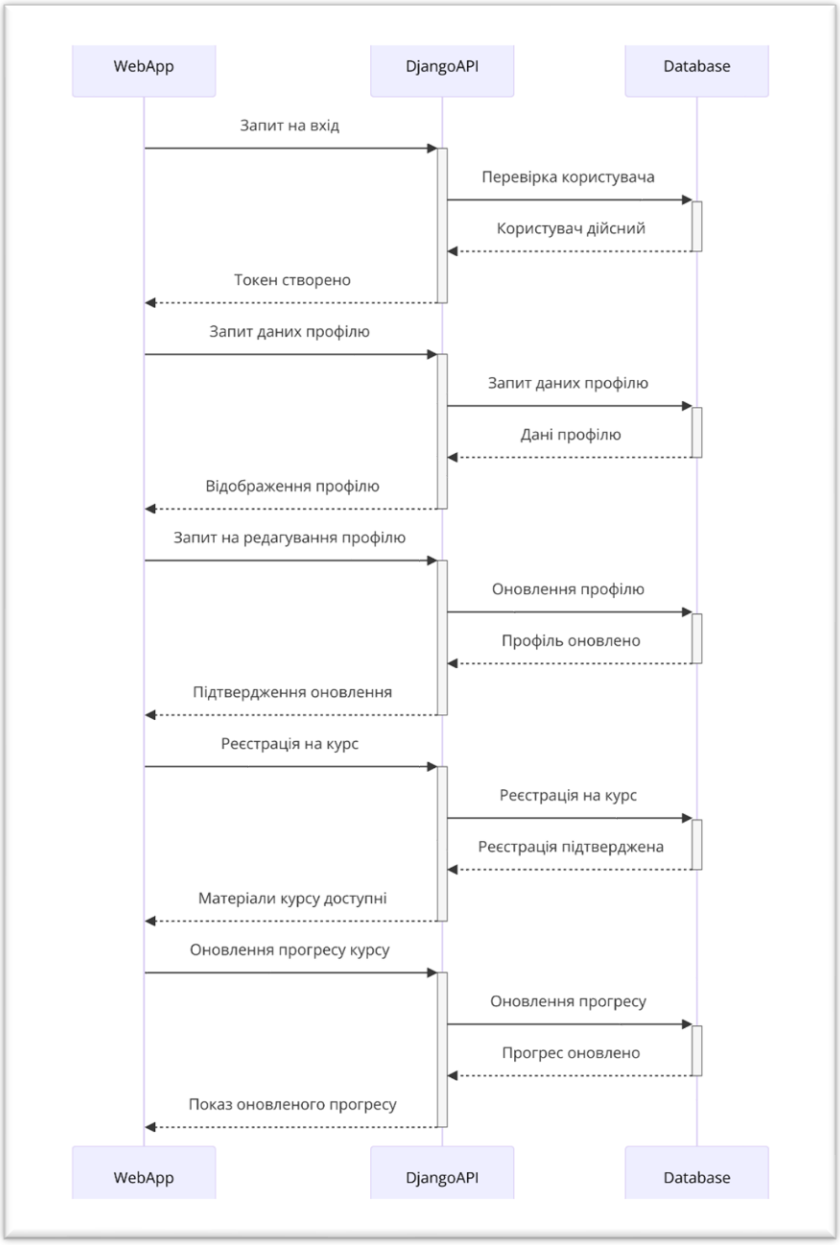


Рисунок 4.1 – Діаграма потоків даних програмного модулю

4.2 Діаграма переходів станів

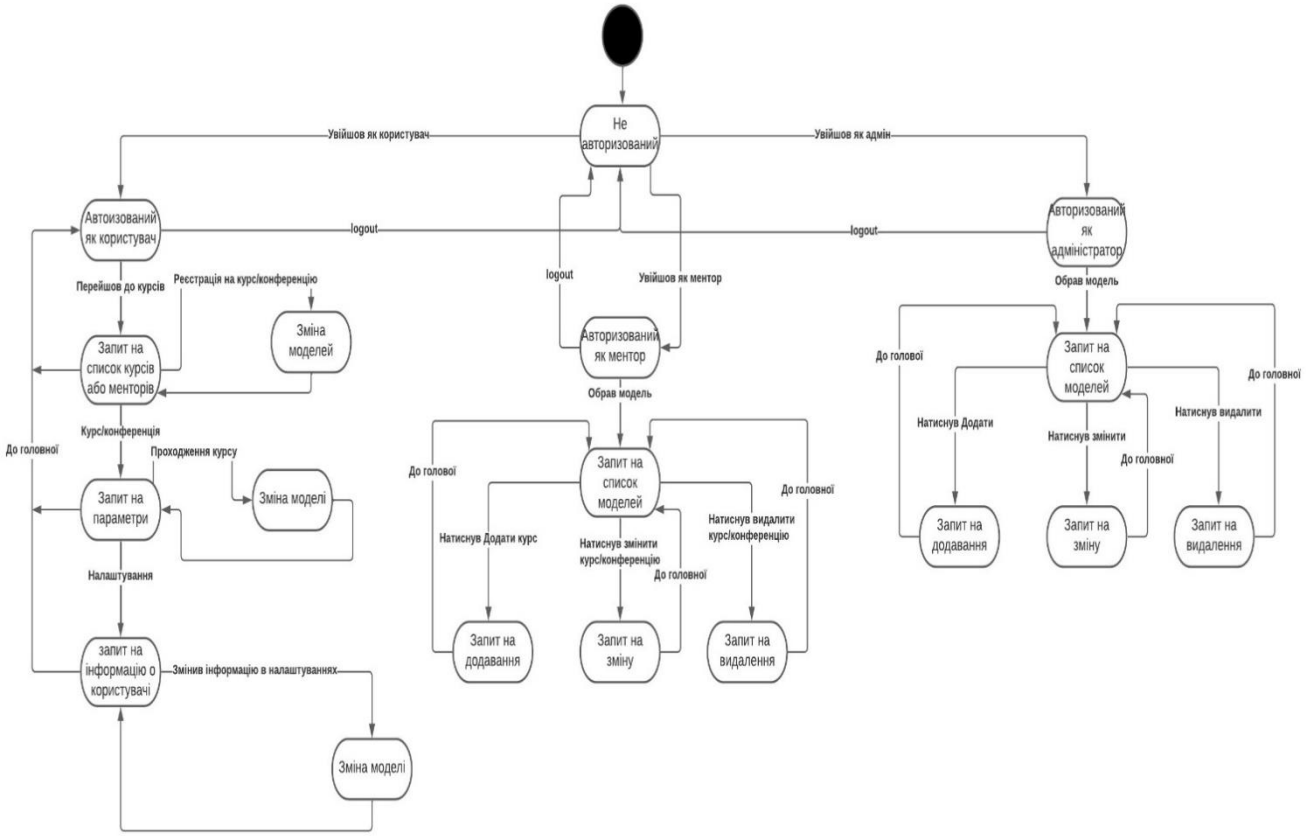


Рисунок 4.2 – Діаграма потоків даних програмного модулю

4.3 Діаграма потоків даних

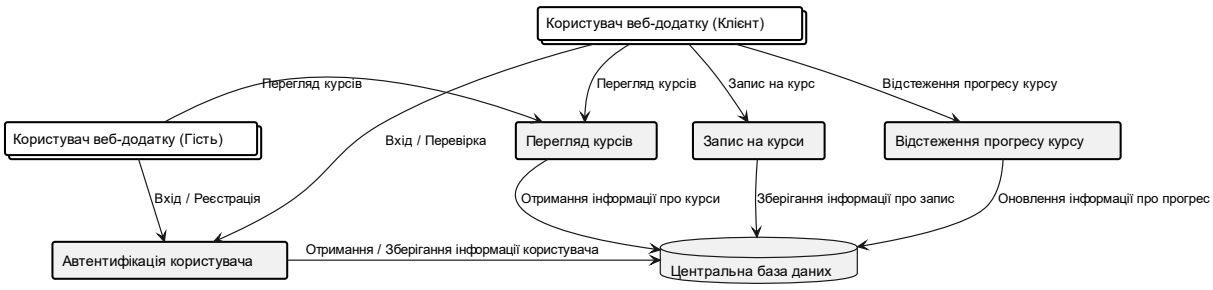


Рисунок 4.3 – Діаграма потоків даних програмного модулю

ДОДАТОК Г

Код класу CourseSerializer

```
class CourseSerializer(serializers.ModelSerializer):
    author_name = serializers.CharField(source='author.first_name',
read_only=True)
    what_learnt = CourseWhatLearntSerializer(many=True, read_only=False)
    sections = CourseSectionSerializer(many=True, required=False)
    ratings_count = serializers.SerializerMethodField()
    num_purchases = serializers.SerializerMethodField()
    purchased = serializers.SerializerMethodField()
    length = serializers.SerializerMethodField()
    current_user_rating = serializers.SerializerMethodField()
    rating = serializers.SerializerMethodField()
    image_content = Base64ImageField(max_length=None, use_url=True,
required=False)
    payment = serializers.SerializerMethodField()

    class Meta:
        model = CoursesCourse
        fields = '__all__'
        extra_kwargs = {
            'image': {'read_only': True},
            'status': {'read_only': True},
        }

    def get_purchased(self, obj):
        user = self.context.get('user')
        if not user or not user.is_authenticated:
            return False
        return CoursesPaidcourseslibrary.objects.filter(user=user,
course=obj).exists()

    def get_ratings_count(self, obj):
        return CoursesCourseRating.objects.filter(course=obj).count()

    def get_num_purchases(self, obj):
        return CoursesPaidcourseslibrary.objects.filter(course=obj).count()

    def get_length(self, obj):
        return sum(section['length'] for section in
CourseSectionSerializer(obj.sections.all(), many=True).data)

    def get_current_user_rating(self, obj):
        user = self.context.get('user')
        if not user or not user.is_authenticated:
            return None
        rating = CoursesCourseRating.objects.filter(user=user,
course=obj).first()

        return rating.rate if rating else 0

    def get_rating(self, obj):
        ratings = CoursesCourseRating.objects.filter(course=obj)
        rating = ratings.aggregate(avg_rating=Avg('rate'))['avg_rating']
        if rating is None:
```

```
            return 0
        return rating

    def create(self, validated_data):
        sections_data = validated_data.pop('sections', [])
        what_learnt_data = validated_data.pop('what_learnt', [])
        if 'image_content' in validated_data:
            image_content = validated_data.pop("image_content")
            path =
f"courses/course_{validated_data.get('id')}/course_image.png"
            image_url = upload_image_to_blob(image_content, path)
            validated_data['image'] = image_url

        course = CoursesCourse.objects.create(**validated_data)

        for section_data in sections_data:
            contents_data = section_data.pop('contents', [])
            section = CourseSection.objects.create(course=course,
**section_data)

            for content_data in contents_data:
                SectionContent.objects.create(section=section,
**content_data)

        for item in what_learnt_data:
            CoursesCourseWhatLearnt.objects.create(course=course, **item)

        return course

    def update(self, instance, validated_data):
        sections_data = validated_data.pop('sections', [])
        what_learnt_data = validated_data.pop('what_learnt', [])
        if 'image_content' in validated_data:
            image_content = validated_data.pop("image_content")
            path =
f"courses/course_{validated_data.get('id')}/course_image.png"
            image_url = upload_image_to_blob(image_content, path)
            validated_data['image'] = image_url
        instance = super().update(instance, validated_data)
        instance.sections.all().delete()
        for section_data in sections_data:
            contents_data = section_data.pop('contents', [])
            section = CourseSection.objects.create(course=instance,
**section_data)
            for content_data in contents_data:
                SectionContent.objects.create(section=section,
**content_data)
        instance.what_learnt.all().delete()
        for item in what_learnt_data:
            CoursesCourseWhatLearnt.objects.create(course=instance, **item)

        return instance

    def get_payment(self, obj):
        return "paid" if obj.price > 0 else "free"
```

ДОДАТОК Д

Код класу CourseModelFilter

```
class CourseModelFilter(filters.FilterSet):
    title = filters.CharFilter(lookup_expr='icontains')
    language = filters.CharFilter(lookup_expr='exact')
    payment = filters.CharFilter(lookup_expr='exact')

    price = filters.NumberFilter(field_name='price')
    price__gte = filters.NumberFilter(field_name='price',
lookup_expr='gte')
    price__lte = filters.NumberFilter(field_name='price',
lookup_expr='lte')

    published_after = filters.DateFilter(field_name='published',
lookup_expr='gte')
    published_before = filters.DateFilter(field_name='published',
lookup_expr='lte')

    status = filters.CharFilter(lookup_expr='exact')

    author_id = filters.NumberFilter(field_name='author_id',
lookup_expr='exact')
    category_id = filters.CharFilter(field_name='category_id',
lookup_expr='exact')
    category_name = filters.CharFilter(field_name='category_id__name',
lookup_expr='exact')

    student_rating__gte = filters.NumberFilter(field_name="student_rating",
lookup_expr='gte')
    student_rating__lte = filters.NumberFilter(field_name="student_rating",
lookup_expr='lte')

    course_length__gte = filters.NumberFilter(field_name="course_length",
lookup_expr='gte')
    course_length__lte = filters.NumberFilter(field_name="course_length",
lookup_expr='lte')

    class Meta:
        model = CoursesCourse
        fields = []
```

ДОДАТОК Д

Код класу CourseApiView

```
class CourseApiView(viewsets.ModelViewSet):
    queryset = CoursesCourse.objects.all()
    serializer_class = CourseSerializer
    filter_backends = (filters.DjangoFilterBackend,)
    filterset_class = CourseModelFilter
    http_method_names = ['get', 'post', 'put', 'patch', 'delete']

    def get_queryset(self):
        queryset = CoursesCourse.objects.annotate(
            ratings_count=Count('coursescourserating'),
            num_purchases=Count('coursespaidcourseslibrary')
        )

        ordering = self.request.query_params.get('ordering', None)

        if ordering:
            queryset = queryset.order_by(ordering)

        return queryset

    def get_serializer_context(self):
        context = super().get_serializer_context()
        context['user'] = self.request.user
        return context
```

ДОДАТОК Ж

Конфігураційний файл docker-compose.yml

```
services:
  db:
    image: postgres:15
    restart: always
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
      PGUSER: postgres
    ports:
      - "5432:5432"
    volumes:
      - ./postgres_data:/var/lib/postgresql/data
      - ./sql:/docker-entrypoint-initdb.d/
      - ./minderAPI:/code/minderAPI
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U postgres" ]
      interval: 5s
      timeout: 5s
      retries: 5
  web:
    depends_on:
      db:
        condition: service_healthy
    build: .
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    environment:
      DJANGO_DB_NAME: ${POSTGRES_DB}
      DJANGO_SU_NAME: ${SUPERUSER_NAME}
      DJANGO_SU_EMAIL: ${SUPERUSER_EMAIL}
      DJANGO_SU_PASSWORD: ${SUPERUSER_PASSWORD}
      DB_HOST: db
      AZURE_STORAGE_CONNECTION_STRING: ${AZURE_STORAGE_CONNECTION_STRING}
    restart: always

volumes:
  postgres_data:
  sql:
```