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.
Mobile Application, Front-end.
(тема)

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

Середа І. А.
(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 Середа Іллі Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для онлайн-тренінгу ментального здоров'я. Mobile Application, Front-end.
Затверджена наказом по університету від 20.05. 2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024
3. Вихідні дані до роботи Розробити інтерфейс програмної системи, яка забезпечує онлайн-тренінги та курси з питань ментального здоров'я, використовуючи бібліотеку React та мову програмування TypeScript.
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	10.04.2024	*виконано*
2	Створення специфікації ПЗ	12.04.2024	*виконано*
3	Проектування ПЗ	15.04.2024	*виконано*
4	Розробка ПЗ	26.05.2024	*виконано*
5	Тестування ПЗ	28.05.2024	*виконано*
6	Оформлення пояснювальної записки	29.05.2024	*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 08 квітня 2024р.

Студент (ка)_____ Середа І. А.
(підпис)

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

РЕФЕРАТ / ABSTRACT

Звіт з кваліфікаційної роботи, 88 стор., 48 рис., 10 джерел.

АРХІТЕКТУРА ПРОГРАМНИХ СИСТЕМ, ІНТЕРФЕЙС КОРИСТУВАЧА, ВЕБ-ДОДАТОК, МОНЕТИЗАЦІЯ, ОНЛАЙН ТРЕНІНГИ, МЕНТАЛЬНЕ ЗДОРОВ'Я, REACT, TYPESCRIPT, VSCODE, MATERIAL-UI, РОЗРОБКА ПЗ, ФУНКЦІОНАЛЬНІ ВИМОГИ, ЦІЛЬОВА АУДИТОРІЯ

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

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

Метод рішення – середовище розробки VSCode, Rest API, та бібліотека для Front-end React, що використовує Material-UI.

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, USER INTERFACE, WEB APPLICATION, MONETIZATION, ONLINE TRAINING, MENTAL HEALTH, REACT, TYPESCRIPT, VSCODE, MATERIAL-UI, SOFTWARE DEVELOPMENT, FUNCTIONAL REQUIREMENTS, TARGET AUDIENCE

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

The purpose of the work is to create a software system that provides online training and courses in mental health topics.

Solution method – VSCode development environment, Rest API, and React frontend library using Material-UI.

As a result of the development, a software system that provides online courses and training in mental health topics, including calendar integration for scheduling with a mentor.

Я, Середа Ілля Андрійович, 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. Mobile Application, Front-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

ВСТУП

У наш час питання ментального здоров'я стає дедалі важливішим, оскільки багато людей зіткнулися зі стресом, емоційним виснаженням та нестабільним психічним станом. Доступність професійної психологічної допомоги обмежена географічними та фінансовими чинниками, тому онлайн-платформи стають важливим інструментом для отримання якісних ресурсів та підтримки. Суспільство потребує програмної системи, що дозволить підвищувати психічне благополуччя, розвивати навички саморегуляції та користуватися цінними тренінгами і порадами менторів в будь-якому місці і в зручний час.

Онлайн-тренінги з ментального здоров'я можуть допомогти у багатьох ситуаціях, коли звичайна терапія не доступна. Важливо, щоб користувачі могли спілкуватися з ментором у зручний для них час і отримувати індивідуальні поради з поліпшення психічного здоров'я. Відстеження особистого прогресу та розробка індивідуальних планів допомагає систематично працювати над своїм станом, концентруючись на конкретних результатах. Планування та запис до ментора в календар дозволяє краще управляти часом і підвищує ефективність тренінгів.

Головна мета цього дослідження полягає в розробці програмної системи для онлайн-тренінгу ментального здоров'я, що дозволить користувачам самостійно подолати труднощі або отримати підтримку ментора. Система має допомагати людям зі стресом та іншими емоційними викликами, сприяючи покращенню психологічного благополуччя. З метою досягнення поставленої задачі потрібно детально дослідити існуючий матеріал та різні способи створення візуально приємного та інтуїтивно зрозумілого інтерфейсу користувача, який буде заохочувати клієнтів користуватися програмною системою. Також важливо розробити систему відстеження прогресу, яка надаватиме користувачам об'єктивну оцінку власного розвитку.

У кінцевому результаті буде отримано програмну систему, яка поєднає всі необхідні інструменти для онлайн-тренінгу ментального здоров'я. Ця система дозволить користувачам самостійно планувати заняття, отримувати зворотній зв'язок від менторів та відстежувати прогрес у поліпшенні психічного стану.

Front-end частина розроблятиметься з використанням мови програмування TypeScript, сучасної бібліотеки React та таких бібліотек, як react-router-dom та zod для покращення роботи додатку і виконання всіх необхідних функцій. Інтерфейс системи буде створений за допомогою сучасної технології Material-UI, що містить в собі готові дизайнерські рішення, та CSS для модернізації їх візуальної частини. Такий підхід дозволить створити функціональний та доступний продукт, здатний надавати всебічну підтримку користувачам у поліпшенні ментального здоров'я.

Використання таких бібліотек, як Material-UI допоможе набагато швидше розробляти компоненти системи. Це пов’язано з тим, що Material-UI це таке сховище готових дизайнерських та функціональних рішень, які можна використати, наслідувати та змінювати, як потрібно в кожній окремій частині проекту. Це можна порівняти з корисністю використання класів в об’єктно-орієнтованих мовах програмування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної галузі та постановка задачі

Психічне здоров'я – це не просто відсутність проблем, а стан благополуччя, що дозволяє людині повноцінно жити, працювати та спілкуватися. Сьогодні, коли стрес, поспіх, економічні негаразди та соціальні проблеми стають все більш поширеними, питання психічного здоров'я виходить на перший план. За даними Всесвітньої організації охорони здоров'я (ВООЗ), понад 450 мільйонів людей у світі страждають від психічних розладів.

Пандемія COVID-19 та військові конфлікти лише загострили ситуацію. Люди стикаються з тривогою, депресією, втратою та фінансовими труднощами, що негативно впливає на їхнє психічне здоров'я. Соціальна дистанція та обмежений доступ до традиційних методів лікування роблять нові підходи до підтримки психічного здоров'я більш актуальними, ніж будь-коли.

Онлайн-тренінги з психічного здоров'я стають доступним та зручним способом отримати допомогу. Ці програми пропонують психологічну підтримку та навчання 24/7, незалежно від вашого місцезнаходження. Вони включають різні методи, такі як когнітивно-поведінкова терапія, медитація, релаксаційні практики та багато іншого. Онлайн-тренінги можуть бути як самостійним інструментом, так і доповненням до традиційної терапії.

Однак існують певні проблеми та виклики, які пов’язані з використанням онлайн-тренінгів з покращення ментального здоров’я. Деякі програми можуть містити неякісний контент або використовувати неперевірені методики, що може призвести до неефективності або навіть погіршення стану. Багато програм не пропонують індивідуального підходу, що не враховує унікальні потреби та проблеми кожного користувача. Не всі мають доступ до Інтернету або можуть дозволити собі платні програми, що обмежує доступ до допомоги.

З огляду на ці проблеми, розробка нової програмної системи для онлайн-тренінгів психічного здоров'я стає нагальною потребою. Така система повинна

ґрунтуватися на наукових даних, пропонувати індивідуальний підхід та бути доступною для всіх.

Впровадження такої системи може зробити допомогу з психічного здоров'я доступною для мільйонів людей, покращити їхнє життя та зробити світ кращим. Для досягнення цієї мети необхідно активно працювати над розробкою та впровадженням ефективних інструментів, що забезпечать всебічну та доступну підтримку психічного здоров'я.

1.2 Цільова аудиторія

Основна цільова аудиторія складається з наступних груп:

а) Ця група включає людей різного віку і соціального статусу, які бажають підвищити свою стійкість до стресу та навчитися методів саморегуляції для поліпшення емоційного благополуччя. Наприклад, студенти, які зазнають навчальних та емоційних навантажень, можуть здобути користь від онлайн-тренінгів з ментального здоров'я, щоб краще управляти стресом і зберігати психологічну стійкість. Також до цієї категорії можуть входити працівники, які стикаються зі стресом через проблеми на роботі та бажають засвоїти техніки саморегуляції для покращення ефективності роботи. Також зацікавитись онлайн-тренінгами може і дуже велика кількість літніх людей, які в такому віці особливо турбуються про своє ментальне та фізичне здоров’я. Через все це для таких людей онлайн-платформа, яка допоможе прибрати стрес та нервування буде просто необхідною мірою.

б) Люди, які переживають складні життєві обставини через соціальні або економічні проблеми, також становлять важливу аудиторію для онлайн-тренінгів з психічного здоров'я. Наприклад, під час економічних криз або соціальних нестабільностей, рівень тривожності та депресії значно зростає, що підтверджується даними American Psychological Association (APA). Онлайн-тренінги можуть надати цим людям доступ до психологічної підтримки та навчання методам самодопомоги, щоб

полегшити їхній психологічний стан під час важливих життєвих перипетій.

в) Фахівці сфери психічного здоров'я, такі як психологи та психотерапевти, можуть отримати потужний інструмент завдяки онлайн-тренінгам. Ця система дозволяє їм ефективно відслідковувати прогрес клієнтів, відстежуючи їх психоемоційний стан та збираючи дані про думки, емоції та поведінку. Аналізуючи цю інформацію, фахівці можуть створювати персоналізовані плани лікування, враховуючи унікальні потреби кожного клієнта. Дослідження з Journal of Medical Internet Research підтверджують, що використання цифрових інструментів у психотерапії значно підвищує її ефективність. До переваг онлайн-тренінгів для цієї категорії людей можна віднести доступність даних, значну економію часу та більшу мотивацію клієнтів.

г) Компанії та некомерційні організації можуть знайти застосування онлайн-тренінгам з психічного здоров'я для підтримки своїх співробітників чи членів спільноти. Інвестування у програми психічного здоров'я приносить організації низку переваг. По-перше, доступ до ресурсів з психічного здоров'я робить працівників щасливішими, здоровішими та більш задоволеними життям. Це, у свою чергу, покращує моральний дух, знижує рівень стресу та підвищує мотивацію. По-друге, онлайн-тренінги допомагають боротися із серйозною проблемою вигорання, що знижує продуктивність та призводить до відходу з роботи. Навчаючи співробітників справлятися зі стресом, встановлювати межі та дбати про себе, тренінги зменшують ризик вигорання. Зрештою, здорові та щасливі працівники більш продуктивні та креативні. Дослідження показують, що програми психічного здоров'я можуть призвести до значного зростання продуктивності та зменшення витрат на медичне обслуговування. Організації можуть використовувати онлайн-тренінги кількома способами. Систему можна застосовувати для проведення регулярних тренінгів з управління стресом, когнітивно-поведінкової

терапії, mindfulness та інших тем, що сприяють психічному благополуччю. Крім того, система дозволяє проводити анонімні опитування для оцінки психоемоційного стану працівників. Ця інформація допомагає визначити проблемні зони та розробити цільові програми підтримки. Отримані дані також дозволять організаціям розробити стратегії створення більш сприятливого робочого середовища, яке сприяє психічному здоров'ю та благополуччю співробітників.

Важливо, щоб програмна система відповідала потребам кожної із зазначених вище груп та надавала користувач інтуїтивно зрозумілий та приємний оку інтерфейс. Також додаток має надавати всі необхідні засоби та інструменти для ефективного навчання, підтримки або покращення ментального здоров’я. Є необхідним зробити систему адаптивною до будь-яких пристроїв та сучасних браузерів, що б дозволило клієнтам легко користуватись програмною системою, не витрачаючи час та сили на те, щоб розібратись, як почати роботу з додатком.

1.3 Огляд систем аналогів

Сучасний ринок онлайн-платформ для ментального здоров'я пропонує різноманітні сервіси, які можна умовно поділити на декілька категорій.

Для самостійної роботи користувачам доступні мобільні додатки, на кшталт Calm та Headspace. Вони пропонують інструменти для відстеження настрою, виконання медитацій, ведення щоденників емоційного стану та багато іншого.

Для тих, хто потребує професійної допомоги, існують платформи відеоконсультацій, як Talkspace або BetterHelp. Вони дозволяють зв'язуватися з ліцензованими психологами та терапевтами для онлайн-сеансів.

Якщо людина прагне більше дізнатися про психічне здоров'я, їй допоможуть освітні веб-сайти, наприклад Verywell Mind. Вони надають інформаційні ресурси, статті, відеоуроки та курси з управління стресом, тривогою, депресією тощо.

Останнім часом набирають популярності інтерактивні терапевтичні інструменти. Вони використовують ігрові механіки та завдання для надання

допомоги у доступній формі. Наприклад, Woebot пропонує підтримку у вигляді чат-бота, який дає поради на основі принципів когнітивно-поведінкової терапії.

Завдяки такому різноманіттю сервісів, люди можуть обрати найбільш зручний та дієвий спосіб покращити своє психічне благополуччя.

Headspace — це популярний мобільний додаток, який спеціалізується на медитації та підтримці ментального здоров'я. Він пропонує користувачам структуровані курси медитації, що допомагають освоїти основні техніки саморегуляції та стрес-менеджменту (див. рис. 1.1).

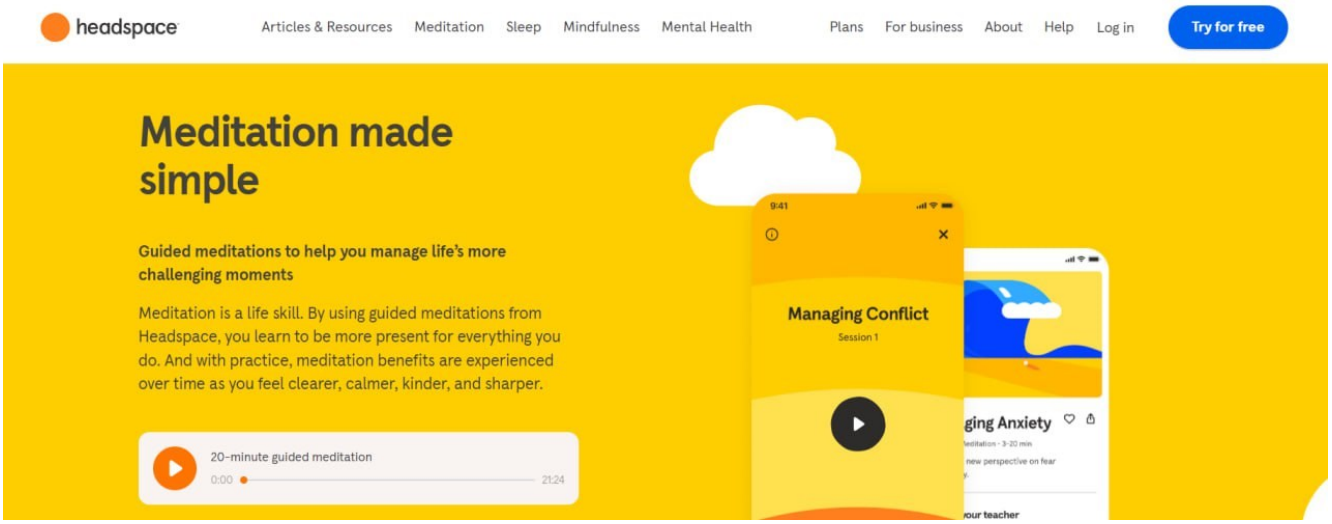


Рисунок 1.1 – Головна сторінка Headspace (за даними [1])

Переваги:

- Чітка і зрозуміла структура курсів;
- Великий вибір медитацій, допомагаючих від різних ментальних проблем;
- Велика кількість курсів та різних статей на тематику психології;
- Індивідуальні вправи для кожного;
- Наявність аудіо-записів, які краще допомагають заспокоїтись.

Недоліки:

- Надто висока ціна на преміум-підписку;
- Майже весь контент відкривається лише за гроші;
- Не буде ефективним для людей з серйозними ментальними проблемами.

BetterHelp — це ще один додаток для відеоконсультацій з ліцензованими спеціалістами та лікарями. На цій платформі реалізована можливість зв'язку з лікарем через відеоконференції або текстові повідомлення (див. рис. 1.2).

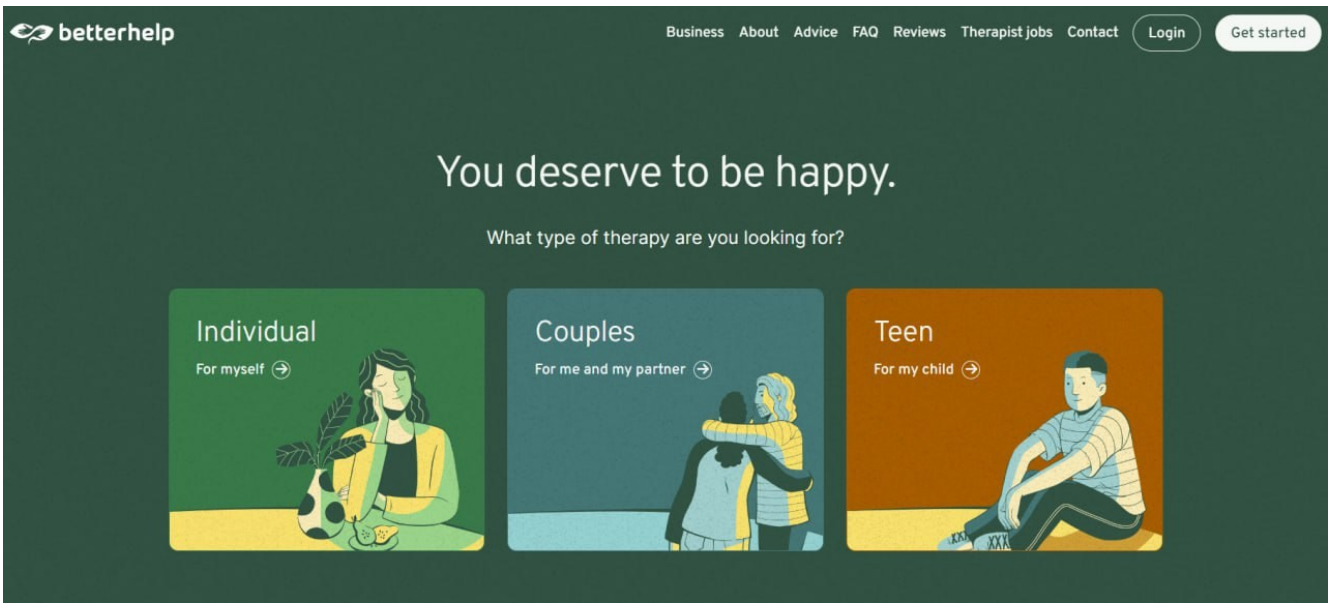


Рисунок 1.2 – Головна сторінка BetterHelp (за даними [2])

Переваги:

- Багато різних спеціалістів в будь-якій з галузей;
- Можливість звернутися до іншого лікаря в будь-який момент;
- Лікарі пропонуються дуже гнучкий графік зустрічей та консультацій;
- Всі дані користувачів надійно захищені.

Недоліки:

- Надто висока ціна на преміум-підписку;
- Труднощі з якістю допомоги через онлайн-формат зустрічей;
- Деякі послуги недоступні для користувачів з певних країн.

Verywell Mind є одним з найкращих освітніх веб-сайтів, що пропонує різноманітні інформаційні ресурси, статті, відеоуроки та курси з ментального здоров'я. Цей сайт охоплює широкий спектр тем, зокрема управління стресом, тривожність, депресія та інші психологічні стани, і вважається надійним джерелом знань у цій сфері (див. рис. 1.3).

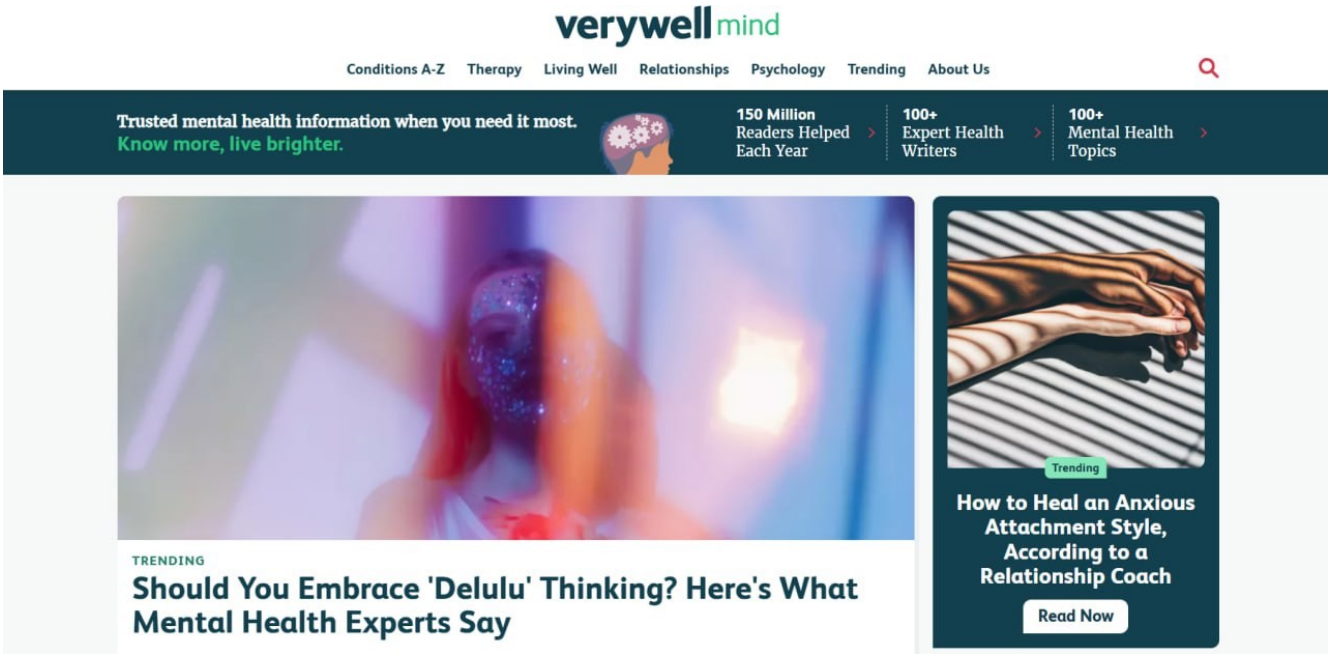


Рисунок 1.3 – Головна сторінка Verywell Mind (за даними [3])

Переваги:

- Різноманітні статті з інформацією майже на будь-яку тему;
- Можливість переглядати інформацію від експертів та психологів;
- Майже весь контент можна переглядати безкоштовно;
- Постійне оновлення матеріалу та статей.

Недоліки:

- Однаковий підхід до кожного користувача і не зрозумілий інтерфейс;
- Додаток виглядає як блог і через це важко шукати інформацію;
- Неможливо зв’язатись з лікарем напряму.

Планується розробити програмну систему, яка враховуватиме найкращі аспекти існуючих аналогів, забезпечуючи більш комплексний та адаптивний підхід до поліпшення ментального здоров'я користувачів. Основні аспекти розробки включатимуть інтерактивність і персоналізацію. Наша система буде включати інтерактивні вправи та механізми відстеження прогресу, аналогічно до функціоналу Headspace.

Другим важливим аспектом буде доступність і гнучкість. Планується забезпечити користувачам можливість спілкування з менторами через відеоконсультації та текстові повідомлення, так само як і BetterHelp. Це дозволить

нашим користувачам вибирати зручний формат спілкування та отримувати підтримку в будь-який зручний час.

Важливий компонент нашої системи - це освітні ресурси. Планується створити широкий спектр освітніх матеріалів, подібних до тих, що пропонує Verywell Mind, таких як статті, відеоуроки та курси. Це допоможе нашим користувачам здобути необхідні знання та навички для ефективного керування своїм ментальним здоров'ям.

Особлива увага буде приділена безпеці та конфіденційності даних користувачів. Система має бути забезпечена високим рівнем захисту особистої інформації та безпечне зберігання даних, так само як це робить BetterHelp.

1.4 Монетизація

Розроблена модель монетизації для нашої системи онлайн-тренінгів ментального здоров'я базується на оплаті за доступ до курсів і індивідуальних тренінгів. Таким чином у клієнтів з’являється можливість обирати тільки ту інформацію, яка їм потрібна. Всі курси складаються з різного виду інформації, а саме відео, в яких користувач може послухати поради від експертів, текстові статті, які допоможуть клієнту дізнатися нову наукову інформацію, а також різноманітні зображення. Кожен курс потребує оплати лише один раз, після чого користувач отримає безкінечний доступ до всієї інформації курсу.

Крім курсів, наша система дозволяє записатися на індивідуальні онлайн-сесії з менторами та тренерами. Така система дозволяє користувача напряму зв’язатися з ментором, щоб обговорити якісь важливі теми або задати питання щодо курсу чи його матеріалу. Оплата за такі сесії здійснюється за кожне заняття окремо.

1.5 Постановка задачі

Створення програмної системи для онлайн-тренінгів ментального здоров'я має на меті вирішення різних проблем у цій сфері. Система повинна надавати доступ до структурованих курсів і методик, щоб адаптуватися під індивідуальні потреби користувачів, враховуючи їхній психоемоційний стан.

Ключовий акцент при розробці системи зосереджений на зручності для користувачів різних вікових категорій і технічних навичок, з простим і інтуїтивним інтерфейсом. Доступність системи є ще однією важливою складовою, оскільки вона має бути економічно доступною для широкого кола користувачів, надаючи безкоштовний базовий доступ та платні курси з додатковими можливостями.

Для підтвердження ефективності системи потрібно провести тестування та валідацію, зібрати дані про користувачів, проаналізувати їхній прогрес та вносити необхідні зміни до методик за потреби. Функція моніторингу психологічно стану наших клієнтів є необхідною. Вона має включати в себе постійний та безперешкодний доступ до зв’язку з ментором та можливість перегляду інформації всіх придбаних курсів.

Інтеграція з іншими онлайн-сервісами та платформами сприятиме підвищенню функціональності та зручності використання системи. Головна мета полягає у створенні ефективної, доступної та науково обґрунтованої системи для онлайн-тренінгів ментального здоров'я.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Головна ідея проекту полягає в створенні програмної системи для навчання онлайн-тренінгів і курсів з ментального здоров'я. Головна ідея програмної системи це забезпечення зручного інструменту для вивчення інформації, пов'язаною з психологією та психологічним здоров'ям. Основна мета кваліфікаційної роботи – управління проектом від етапу формування вимог до готового програмного продукту. Визначимо основні завдання цього проекту:

- аналіз потреб різних груп користувачів;
- визначення ключових потреб продукту;
- визначення інструментів, які будуть потрібні для розробки;
- визначення плану та етапів розробки;
- підтримка та модернізація проекту, а також управління ним;
- проведення якісно та повного тестування системи.

Перед початком формування вимог до продукту необхідно чітко визначити потреби цільової аудиторії програмної системи для онлайн-навчання з ментального здоров'я. Ця аудиторія складається з різних груп користувачів, кожна зі своїми унікальними потребами. Наприклад, професійні психологи та тренери очікують на наявність інструментів для створення курсів і тренінгів та управління розкладом. Всі користувачі мають бути забезпечені привабливим та зручним у використанні інтерфейсом, який дозволить швидко та легко купувати та шукати курси, переглядати їх матеріал і слідкувати за своїми досягненнями. Для ефективного моніторингу роботи команди та дотримання термінів розробки, важливо чітко сформулювати вимоги до системи, які будуть відповідати потребам користувачів, а саме:

- вимоги до основних функцій додатку;
- вимоги до безпеки та захищеності даних користувачів;
- вимоги до можливості зміни мови системи;
- вимоги до горизонтальної масштабованості додатку.

Перед початком формулювання вимог до продукту, необхідно ретельно визначити потреби цільової аудиторії програмної системи для онлайн-навчання з

ментального здоров’я. Сюди входять різні групи користувачів, які мають унікальні вимоги. Наприклад, професійні психологи та тренери очікують наявності інструментів для створення курсів і тренінгів та можливості керування розкладом. Важливо забезпечити всіх користувачів легким та інтуїтивно зрозумілим інтерфейсом, що допоможе користуватись додатком та буде заманювати все більше нових клієнтів.

Функціональні вимоги до системи онлайн-навчання з ментального здоров’я передбачають підтримку повного циклу взаємодії між користувачами та платформою. Такі функції, як авторизації, реєстрація а також зміна персональної інформації в профілі мають бути зрозумілими та доступними для будь-якого клієнту. Курси можуть бути безкоштовними або платними. Для приєднання до платного курсу користувач спочатку повинен придбати його через систему онлайн-платежів на сайті.

Після успішної реєстрації на курс учасники мають отримати доступ до всіх матеріалів і зможуть відстежувати свій прогрес. Кожен користувач повинен мати можливість переглянути свій прогрес по кожному курсу, щоб розуміти, що робити далі, та в якому напрямку потрібно розвиватись наступним кроком.

У зв'язку з необхідністю зберігання конфіденційної інформації про користувачів, маємо гарантувати надійний захист особистих даних. Процедура аутентифікації користувачів має бути виконана у безпечному режимі з шифруванням усіх передач даних. Управління доступом до різних типів інформації повинно бути чітко регульованим залежно від ролі користувача, з можливістю налаштування прав доступу адміністратором.

Система повинна бути готова до росту числа користувачів і ефективної обробки значних обсягів інформації. Використання технології контейнеризації, такої як Docker, дозволяє горизонтально масштабувати систему, що дозволяє з легкістю додавати нові компоненти та модулі і розподіляти навантаження між багатьма серверними інстанціями для підвищення продуктивності та ефективності системи.

Важливо зазначити, що система повинна мати локалізацію мінімум на дві основні мови, а саме англійську та українську. Інтерфейс повинен бути можливим для перекладу, а формат дат, часу та валюти має бути адаптованим під кожну країну, що дозволить клієнтам зручно та швидко користуватись інтерфейсом та отримувати всю актуальну для себе інформацію.

Перед успішним створенням програмної системи необхідно провести вибір інструментів розробки, що відповідатиме потребам і вимогам проекту. З огляду на вимоги до нашого проекту було обрано такі інструменти розробки:

а) для створення стилів додатку було обрану звичайний CSS через його простоту в використанні та ефективність[4];
б) на Front-end використовуватиметься React разом з Material UI для створення зручного та привабливого інтерфейсу [5] [6];
в) для ефективної розробки та керування кодом обрано середовище Visual Studio Code, яке підтримує Javascript та Typescript і підвищує продуктивність;
г) для керування проєктом використовуватимуться системи контролю версій, такі як Git, з хостингом коду на GitHub або GitLab.

Для командної роботи та ефективного створення програмної системи буде задіяна методологія Scrum. Ця методологія надає гнучкий та ефективний підхід до розробки, відстеження часу, а також суттєво допомагає при плануванні командної роботи.

Для налагодження зв’язку всередині команди розробників буде використаний інструмент Jira. Цей потужний інструмент дозволяє ефективно організовувати спілкування та розподіляти завдання між розробниками [7].

Треба подбати про декілька етапів:

а) створення списку завдань продукту: визначення основних вимог і функцій продукту з можливістю їх коригування під час розробки;
б) планування спринтів: визначення завдань, які потрібно виконати протягом наступного періоду та оцінка часу, необхідного для їх виконання;

в) щоденні зустрічі: організація коротких щоденних нарад для огляду прогресу та виявлення можливих перешкод;

г) підсумки та оцінки: огляд виконаних завдань та аналіз досвіду роботи з метою вдосконалення процесу розробки.

Для забезпечення надійності та стабільності програмної системи, тестування відіграє важливу роль у процесі розробки. У нашому випадку будуть використовуватись два основні типи тестування: юніт-тестування та мануальне тестування.

Юніт тести мають бути використанні, коли потрібно протестувати окремі невеликі частини коду або його компоненти. Використання цих тестів дозволить розробникам швидко виправлять майже будь-які помилки або взагалі уникати появи таких помилок. Планується використовувати бібліотеки, такі як Vitest та Playwright для Typescript, щоб автоматизувати процес юніт-тестування та забезпечити швидке виконання тестів. Перевагою цієї бібліотеки є те, що вона дозволяє тестувати кожен компонент системи при різних сценаріях виконання, що значно пришвидшує покриття системи тестами.

Мануальне тестування також є невід'ємною частиною програмної системи, бо воно дозволяє людині перевірити всі наявні функції системи. Такий підхід імітує дії звичайного користувача, що допомагає уникнути неочікуваних проблем для розробників та клієнтів додатку.

Веб-версія повинна підтримувати сучасні браузери, такі як Google Chrome, Firefox, Safari та Edge.

Система також повинна користуватися кукі та локальне сховище (localStorage) для зберігання сесій та персоналізації користувацького досвіду. Використання такого підходу захистить користувачів від втрати свого прогресу.

Для постійної та безперешкодної роботи на будь-яких пристроях буде створено адаптивний під різні розміри інтерфейс. Важливо щоб інтерфейс відповідав всім принципал UI/UX-дизайну, надаючи користувач зручний для користування продукт [8].

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Спочатку буде створена Use-case діаграма (див. рис. 3.1), що буде показувати всі можливості користувача при роботі з додатком. Це забезпечить візуальне уявлення про різні дії та взаємодії між користувачами і системою.

У цій діаграмі будуть зображені різні типи користувачів (акторів), кожен з яких матиме свій набір можливих взаємодій з системою. Основними акторами будуть учасники та адміністратори. Дії кожного актора описуватимуть основні функції, які можуть виконуватися в рамках кожної ролі.

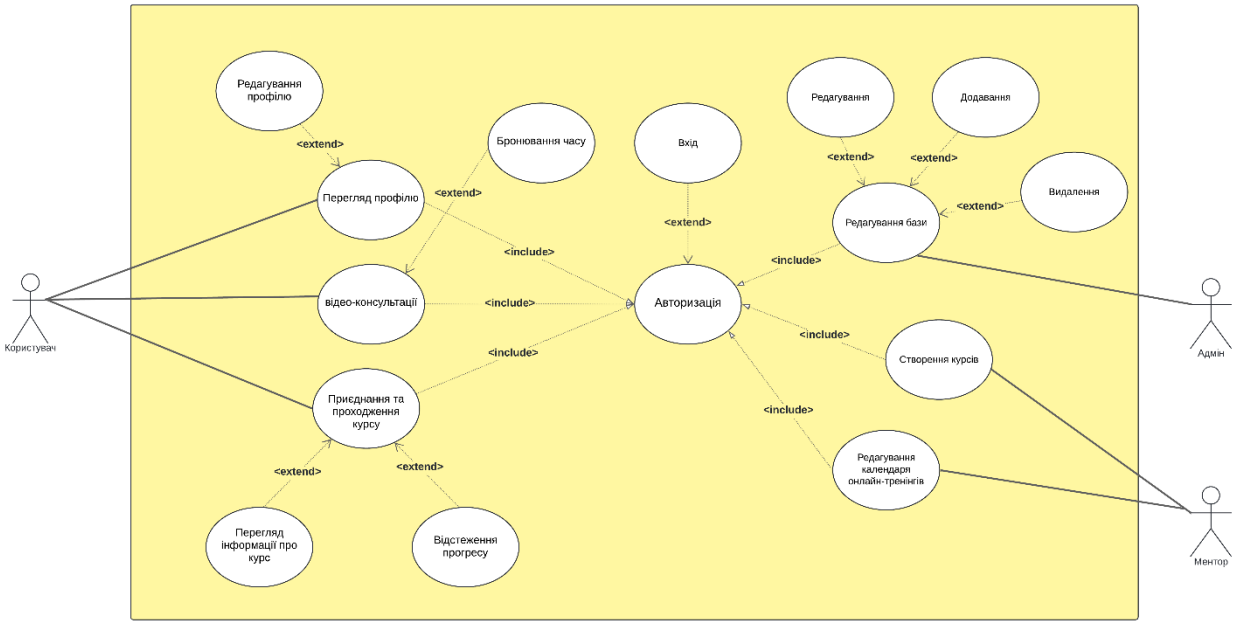


Рисунок 3.1 – Діаграма прецедентів системи (рисунок виконаний самостійно)

На платформі користувачі можуть зареєструватися, щоб переглядати інформаційні матеріали, але не матимуть доступу до курсів. Після авторизації користувачі зможуть записуватися на курси, проходити матеріали і відстежувати свій прогрес. Вони також можуть спілкуватися з менторами і бронювати час для консультацій.

Ментори відповідають за створення та управління курсами, завантаженням матеріалів і тестів, контролем прогресу учасників і наданням зворотного зв'язку.

Крім того, вони проводять відеоконсультації та аналізують ефективність програм для подальшого вдосконалення.

Адміністратори системи керують користувацькими акаунтами, налаштовують параметри системи, забезпечують безпеку даних і вирішують технічні питання користувачів. Вони також відстежують активність в системі для запобігання несанкціонованому доступу та контролю доступу до різних ресурсів і функцій.

3.2 Архітектура системи

Архітектура програмної системи для онлайн-тренінгу ментального здоров'я базується на моделі, що передбачає об’єднане управління даними і бізнес-логікою через серверів, який буде обслуговувати будь-які запити від клієнтської частини. Вибір такої архітектури мотивований потребою у гнучкості та масштабованості, щоб забезпечити підтримку зростаючої кількості користувачів і курсів. Серверна частина може легко розширюватись без впливу на клієнтську частину, забезпечуючи швидке впровадження змін і оновлень.

Забезпечення безпеки є критично важливим завданням, і централізоване управління дозволяє ефективно здійснювати реєстрацію та авторизацію, знижуючи ризики несанкціонованого доступу. Таким чином вся інформація зберігається тільки на сервері та її ніяк не можна дізнатись через клієнтську частину додатку.

Будь-який зв’язок між серверною частиною та клієнтською частиною має бути виконаний лише з використанням чіткої структури API запитів. Це дозволить покращити якість роботи додатку, а також буде позитивно впливати на досвід користувача при роботі з програмною системою. Завдяки цьому інтерфейс можна завжди оновлювати та модифікувати без суттєвих витрат ресурсів та часу. Нижче можна побачити діаграма моделі архітектури програмної системи (див. рис. 3.2).

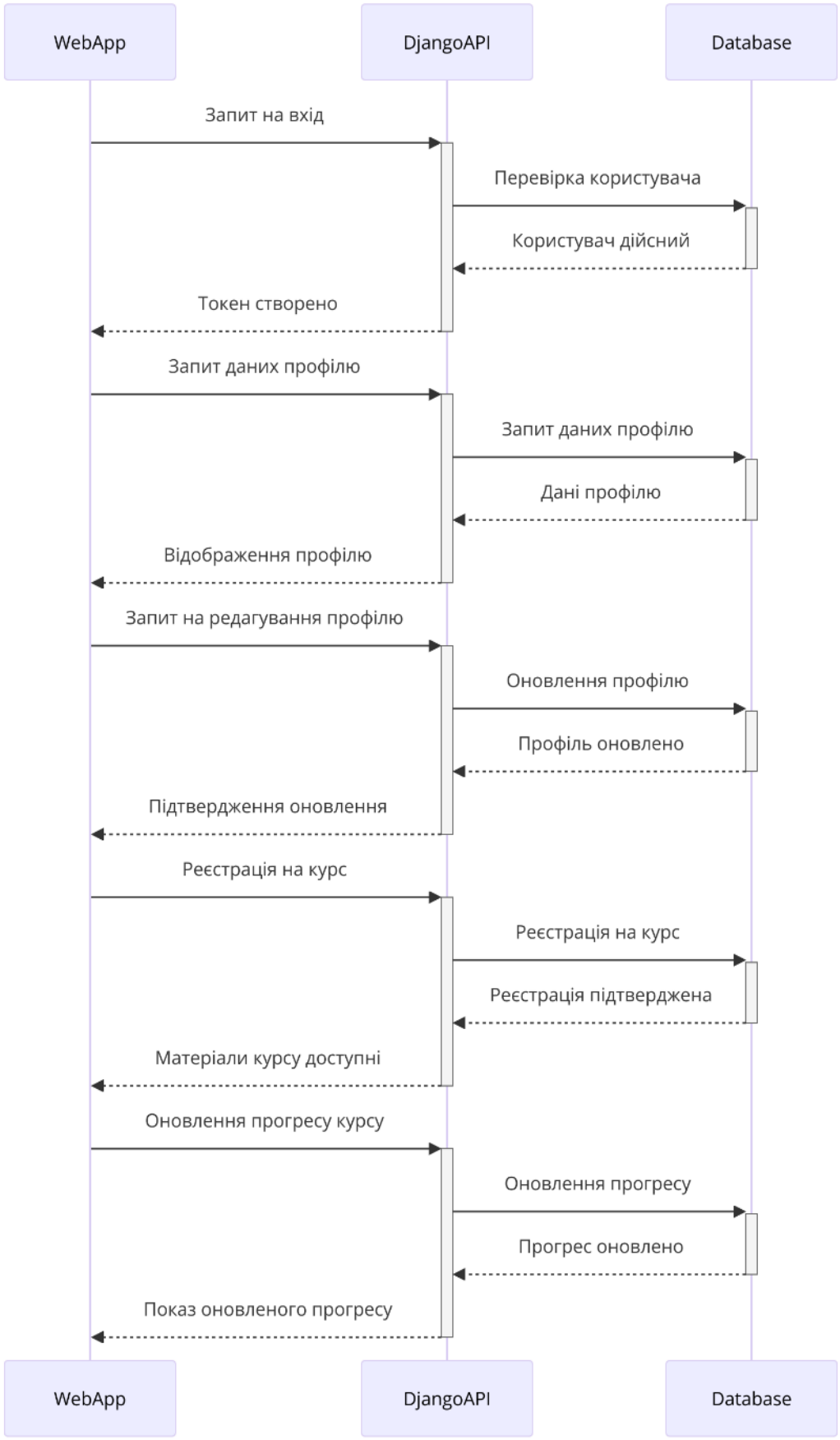


Рисунок 3.2 – Діаграма моделі архітектури (рисунок виконаний самостійно)

3.3 Обмеження та вимоги

При виборі інструментів для розробки додатку важливо врахувати, наскільки довго технологія буде актуальною і як вона підтримуватиме майбутню модернізацію нашого проекту. Ретельно розглянемо обмеження, які технології можуть накласти, виходячи з потреб і можливостей.

Для організації якісної роботи з запитами з серверу буде використана допоміжна бібліотека axios, яка дозволяє набагато простіше відправляти складні запити на сервер. З переваг цієї бібліотеки можна виділити її швидкість роботи та велику кількість вбудованих функцій для модернізації та видозміни формату заголовків та відповідей [9].

Для локалізації додатку буде використана відома та досить потужна бібліотека i18n. Ця бібліотека автоматично зберігає в сесії обрану мову, що дозволить розробникам витрачати менше часу на створення функції зміни мови додатку. I18n потребує від розробника створення мінімум трьох файлів, а саме файла конфігурації, файла з перекладами всіх фраз на одну мову, наприклад, англійську та такого ж файлу з перекладами на іншу мову, наприклад, українську.

Для розробки клієнтських інтерфейсів проекту потрібно використовувати бібліотеку JavaScript React. React дозволяє ефективно розділити інтерфейс на компоненти, що дозволяє побудувати складні інтерфейси з незалежними частинами, які можна повторно використовувати. Це спрощує розробку і полегшує підтримку коду.

Основні переваги використання React включають вбудовану систему тестування, яка дозволяє перевіряти працездатність додатку на різних пристроях та браузерах, що покращує якість продукту і забезпечує користувачів безперебійним досвідом.

Для створення та налаштування стилів веб-додатку потрібно використовувати бібліотеку Material-UI. Material-UI заснована на концепції "component-based" дизайну, що дозволяє легко створювати стилізовані компоненти

і використовувати їх для розробки проекту. Вона містить багато готових компонентів з налаштованим дизайном, такими як кнопки, таблиці, форми, які можна легко і швидко використовувати для побудови інтерфейсу. Використання Material-UI дозволить ефективно створювати стилі для додатку за допомогою готових компонентів і стилів, що спростить процес розробки і полегшить підтримку дизайну.

При розробці додатку також важливо не забувати, що користувацький інтерфейс має певні обмеження. На меню навігації, яке є однаковим для всіх сторінок мають бути наступні елементи: логотип сервісу Minder (див. рис. 3.3), перелік різних частин додатку з можливістю переходу між ними, стрічка для пошуку певної інформації або сторінки проекту, а також кнопки для авторизації або реєстрації користувача, якщо користувач ще не увійшов в свій обліковий запис. Коли користувач входить в свій обліковий запис, то кнопки для авторизації та реєстрації заміняються на зображення профілю з можливість переходу в налаштування облікового запису чи виходу з нього.

Рисунок 3.3 – Логотип сервісу (рисунок виконаний самостійно)

Для авторизації користувача потрібно зареєструвати його обліковий запис та вказати правильні дані в форму для входу. Форма авторизації потребує вказати нікнейм користувача та його пароль, якщо все вказано вірно, то буде виконано вхід в обліковий запис. Для реєстрації користувача потрібно також заповнити форму, в якій треба вказати ім’я, фамілію, нікнейм, пошту та пароль. Якщо все вказано вірно, то користувача буде зареєстровано в системі та автоматично авторизовано.

Програмне забезпечення для перегляду веб-сторінок має бути доступне користувачам для використання веб-системи, і воно повинне підтримувати Javascript, HTML5 та CSS3. Щоб поліпшити досвід роботи з сервісом на мобільних

телефонах, важливо забезпечити можливість перегляду веб-додатку на будь-якому мобільному пристрої та для будь-якого розміру екрану пристрою

Час переходу між вкладками веб-додатку має бути дуже швидким - менше 1 секунди. Система повинна бути надійною і стійкою до помилок. У випадку проблеми користувач повинен отримати зрозуміле повідомлення із описом проблеми, що дозволяє зрозуміти суть без потреби в спеціалізованому коді чи інформації, яку може зрозуміти лише розробник.

Система має бути легкою для оновлення та підтримання, щоб будь-які зміни можна було зробити швидко та легко. Також додаток має підтримувати можливість зміни мови сайту між англійською та українською. Це дозволить збільшити кількість користувачів, а також покращити досвід роботи з додаток для клієнтів. Інтерфейс повинен бути інтуїтивно зрозумілий і приємний у використанні для людей будь-яких вікових категорій.

3.4 Забезпечення якості

Для забезпечення якості додатку Minder, потрібно, щоб команда була єдина в своїх рішеннях щодо будь-яких змін в нашому проекті.

При прийнятті рішень про зміни у проекті проводиться обговорення, щоб узгодити позиції всіх членів команди. Кожен має можливість висловити свою думку щодо запропонованої зміни, інші члени команди можуть підтримати це рішення або висловити свої зауваження чи обгрунтовані протистояння. Обговорення триває до тих пір, поки не буде досягнуто консенсусу і прийняте єдине рішення, яке задовольняє усіх учасників команди.

Цей процес сприяє зміцненню командної взаємодії, забезпечує участь кожного у прийнятті важливих рішень та дозволяє уникнути конфліктів у майбутньому. В результаті отримуємо збалансоване та згодоване рішення, яке підтримує весь колектив та сприяє успішному виконанню проекту.

Підтримка зв’язку між членами команди є важливим механізмом для покращення якості розроблюваного продукту. Цей механізм включає регулярні щоденні зустрічі, а також листування через електронну пошту та месенджери.

Основна мета таких зустрічей і спілкування - обговорити проблеми, висловити ідеї і варіанти рішень, поділитися досвідом та знайти спільне рішення для команди.

Кожна щоденна зустріч проходить протягом 10-20 хвилин і зазвичай проводиться через платформу Google Meet. Це дозволяє членам команди швидко обмінюватися інформацією, вирішувати поточні питання та забезпечувати взаєморозуміння між усіма учасниками процесу розробки.

Такий регулярний зв'язок сприяє ефективному управлінню проектом, вирішенню проблем та швидкому прийняттю рішень, що впливають на якість та успішність розробки продукту.

У нашій команді передбачається проведення обов'язкових оглядів, що включають різні етапи, такі як обговорення чи технічні огляди.

Обговорення існуючих процесів команди, встановлення ключових показників успішності, оцінка поточних процесів, їх перегляд та вибір нових процесів. Ці огляди будуть проводитися онлайн раз на два тижні через платформу Google Meet з тривалістю близько 40-60 хвилин. Основні цілі огляду включають виявлення помилок у функціональності, логіки чи реалізації програмного забезпечення, перевірку відповідності програми її вимогам, переконання у відповідності програми встановленим стандартам та удосконалення управління проектом.

Технічні огляди під час розробки будуть включати покрокові інструкції та перевірки. Покрокові інструкції спрямовані на інтеграцію розроблених компонентів, таких як інтерфейси, форми та база даних, за участю відповідального програміста. Перевірки зосереджуються на правильності розроблених деталей, і проводяться без участі відповідального розробника.

Для забезпечення контролю над якістю розробки програмної системи будуть використовуватись різні інструменти та методи, такі як голосування, зустрічі, детальне проектування та дослідження теми з метою мінімізації помилок. Також планується проведення регулярних оглядів програмного забезпечення для виявлення проблем та відстеження помилок.

Ці заходи спрямовані на покращення якості та ефективності розробки продукту, забезпечуючи високий рівень відповідності вимогам та стандартам усіх етапів процесу розробки.

3.5 Керування проектом

До керування проектом відносяться різні види планування, виконання, контролю за процесом розробки, підтримки або оновленням проекту. Таке керування має проводитись в декілька етапі, кожен з яких має певну важливу для додатку мету.

На етапі початкового планування було зібрано загальну інформацію про проект, сформулювали чіткі цілі і провели аналіз цільової аудиторії разом з її основними потребами, які має вирішувати наш продукт. Також був розроблений перелік ключових функцій, які користувачі зможуть використовувати у системі.

Під час фази виконання проекту планується здійснювати постійний моніторинг. Для ефективного контролю за прогресом та вимірювання успіхів проекту буде використовуватись система управління проектами Jira. Це забезпечить нам впевненість у тому, що проект розвивається згідно з планом і досягає поставлених цілей вчасно

Jira має багато корисних функцій для керування проектами, включаючи створення списку завдань, надання пріоритетів завданням, відстеження часу, додавання коментарів та зміну стану завдань. Крім того, Jira дозволяє створювати звіти та статистику проекту для аналізу продуктивності та ефективності роботи команди.

На основі вимог до програмна система для онлайн-тренінгу ментального здоров'я, буде створено список завдань, які потрібно виконати для успішного завершення проекту. Jira також дозволяє розбивати процес на етапи по кілька тижнів, такі етапи називаються Story. Також є можливість розбити будь-які завдання на категорії, такі як “In Progress”, “Done”, “To Do” тощо. У Jira це буде представлено у вигляді заповненої Story (див. рис. 3.4).

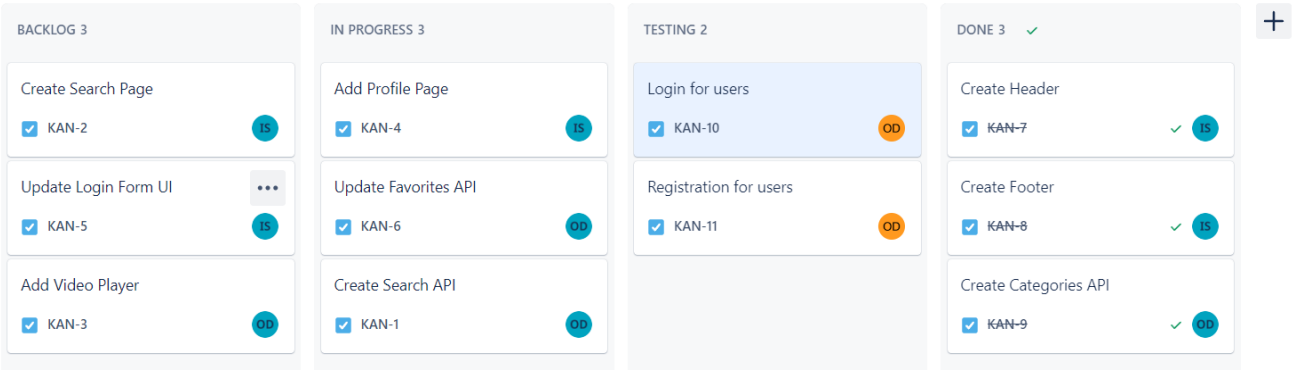


Рисунок 3.4 – Заповнена Story проекту на Jira (рисунок виконаний самостійно)

3.6 Тестування

Для забезпечення якості програмної системи для онлайн-тренінгу ментального здоров'я, планується проведення двох типів тестування: мануального та юніт-тестування.

Мануальне тестування має кілька переваг для нашого проекту. Воно надає гнучкість, оскільки може швидко адаптуватися до змін у програмному забезпеченні. Також мануальне тестування дозволяє ефективно перевіряти візуальну коректність інтерфейсу та інші аспекти користувацького досвіду. Крім того, воно дозволяє проводити більш глибокий аналіз функціональності програми і виявляти потенційні проблеми, які можуть бути пропущені автоматизованими тестами. Мануальне тестування легше виконувати і не вимагає великої кількості технічних знань і досвіду. Також знайдені проблеми можна легко описати і передати розробникам для виправлення. Крім того, воно дозволяє перевіряти реальний користувацький досвід.

Юніт-тестування є ще одним важливим етапом нашого тестування. Цей вид тестування призначений для перевірки окремих компонентів програмного забезпечення (функцій, методів, класів) на коректність їх роботи. Використання юніт-тестів дозволяє виявляти помилки на ранніх стадіях розробки, що сприяє покращенню якості програмного продукту. Ці тести дозволяють перевіряти правильність роботи окремих модулів без їх взаємодії з іншими частинами системи. Результати юніт-тестів допомагають підтвердити відповідність окремих компонентів вимогам та стандартам програмної системи. Використовувати юніт-

тести дуже корисно при розробці з використанням React або Next.js. React має вбудовану та обов’язкову компоненту архітектуру, тобто кожна сторінка розбита на багато маленьких компонентів, які можна перевикористати. Через це використовувати юніт-тести для роботи з ним просто необхідно, бо це дозволяє тестувати кожен компонент окремо від всієї іншої логіки і це набагато легше робити з React, аніж з будь-яким іншим Front-end фреймворком чи бібліотекою.

Будь-які мануальні чи юніт тести мають виконуватись з даними, близькими до реальних. Це робиться для того, щоб запевнитись, що у користувача, дані якого будуть реальні, все буде працювати, як треба і не виникне жодних неочікуваних помилок.

Стадія тестування буде проходити в декілька етапів. Першим етапом буде виконання мануальних та юніт тестів та запис всіх знайдених помилок для подальшої передачі цієї інформації розробникам. В разі виявлення помилок буде виконуватись другий етап, а саме усунення знайдених помилок. Після вирішення проблеми розробник змінює статус помилку з активної на ту, що потребує повторного тестування.

Ці два етапи будуть виконуватись до тих пір, поки всі проблеми не будуть усунені. Такий підхід дозволяє забезпечити швидкий пошук проблем, миттєве реагування на знайдені помилки, а в подальшому і їх усунення.

Також варто зазначити, що не можна зупинятись в створенні юніт тестів або виконанні мануального тестування, бо разом з оновленням проекту та додаванням нових функцій мають бути доданні та оновленні існуючі юніт-тести та перероблено мануальне тестування. Це робиться для того, щоб запевнитись, що на жодному з етапів не була створено нових помилок, які могли б негативно впливати на досвід користувачів чи роботи програмної системи.

Таким чином, мануальне та юніт-тестування разом дозволять забезпечити високу якість та надійність програмна система для онлайн-тренінгу ментального здоров'я.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір програмних засобів реалізації для Front-end частини

Для того, щоб реалізувати Front-end частину додатку було обрано мову програмування TypeScript та фреймворк React. Така комбінація технологій дозволить швидко та легко створювати майже будь-які програмні системи. За допомогою TypeScript написана більшість сайтів за останні кілька років, тому можна сміливо казати, що ця мова програмування займає одну з найголовніших позицій в сфері веб-розробки. Але, навіть, попри це його дуже часто порівнюють з такою мовою програмування, як JavaScript. Пов'язано це з тим, що TypeScript є нащадком JavaScript'а, але містить в собі багато додаткових функцій та можливостей, яких не було до цього. Мова програмування TypeScript має багато переваг над JavaScript. Визначимо кілька найважливіших переваг:

а) типізація: саме це є одним з тих аспектів, яких не вистачало JavaScript для конкуренції з мовами програмування, такими як C#, Python чи C++. Для розробників, звиклих до роботи з типами, може бути важко зрозуміти код, написаний на JavaScript;

б) покращена підтримка IDE: це одна з найвагоміших переваг TypeScript над JavaScript. Сучасні IDE пропонують широкі можливості, що значно спрощують написання коду. Проте, це не завжди повністю використовується з JavaScript, оскільки IDE не завжди може виявити помилки або надати підказку про неправильний тип значення. У TypeScript ця проблема вирішена завдяки статичній типізації, що дозволяє IDE працювати ефективніше, оскільки TypeScript підтримує більшість патернів розробки, що характерні для сучасних мов програмування. Таким чином, TypeScript сприяє більш ефективній роботі з будь-якою інтегрованою середою розробки, що робить його бажаним вибором для багатьох розробників;

в) підтримка сучасних можливостей JavaScript: це ще одна перевага TypeScript. Ця мова програмування швидше адаптується до нових функцій

та синтаксису, що з'являються у нових версіях ECMAScript. Це дозволяє розробникам використовувати останні інновації мови програмування без затримок, що можуть виникати при роботі зі звичайним JavaScript;

г) легкість інтеграції в проєкт: при заміні JavaScript на TypeScript в існуючому проекті не потрібно робити багато міграцій та завантажень пакетів. Це все можна зробити однією командою, завдяки npm (node package manager). Все, що залишиться розробнику – зробити типізацію змінних та функцій в проєкті. А якщо потрібно створити новий проєкт на TypeScript замість JavaScript, то це робиться однією консольною командою, в якій змінюється лише мова програмування, завдяки чому не потрібно витрачати багато часу на інтеграцію TypeScript в проєкт.

Можна багато говорити про переваги TypeScript над JavaScript, але не варто забувати про те, що всі мови програмування мають і свої суттєві недоліки. Визначимо кілька основних недоліків TypeScript:

а) додаткова складність через статичну типізацію: статична типізація допомагає уникнути багатьох помилок на етапі компіляції, але вона може також збільшити складність написання коду, особливо для тих, хто не звик працювати з типами або для менших проектів, де це може здатися зайвим. Найчастіше з такого проблемою стикаються саме новачки у програмуванні, які обрали своєю першою мовою програмування JavaScript і ще не мають достатньо досвіду, щоб працювати зі статичною типізацією;

б) додатковий етап компіляції: будь-який браузер інтерпретує JavaScript, через це перед кожним завантаженням змін в коді виконується компіляція TypeScript той код, який може обробити браузер, а саме JavaScript. Це може призводити до невеликої затримки в процесі розробки та розгортання, але не матиме ніякого впливу на досвід користувача та швидкість роботи додатку. Тому, цей недолік не є таким суттєвим, як інші;

в) не всі бібліотеки придатні до використання: хоча TypeScript активно розвивається, не всі сторонні JavaScript бібліотеки мають визначені типи

або підтримують TypeScript. Це може призвести до того, що розробникам доведеться писати або підтримувати власні оголошення типів для цих бібліотек, щоб отримати переваги типізації.

Хоча мова програмування TypeScript має свої проблеми, такі як складність для новачків або додатковий етап компіляції, ці недоліки не можуть стояти на одному рівні з тими суттєвими перевагами, які пропонує TypeScript. Навіть одна типізація перекриває всі недоліки і робить його бажаним вибором для будь-якого досвідченого веб-розробника. Ця інформація підтверджується і статистикою авторитетного сервісу Dou.ua, на якому проголосували більше ніж 500 веб-розробників, з яких 74% обрали саме TypeScript і лише 26% обрали JavaScript як свою основу мову програмування [10].

Вибір між TypeScript і JavaScript є важливим, але вибір фреймворку для Front-end розробки є найбільш значимою частиною етапу вибору технологій. Серед сучасних фреймворків та бібліотек можна виділити три основні, такі як React, Vue та Angular. Всі вони є потужними інструментами, які можуть суттєво допомогти в розробці будь-якого додатку. Кінцевий вибір все ж таки пав на React через його переваги над Vue та Angular. Серед таких переваг можна виділити основні:

а) React виділяється своєю гнучкістю й простотою та має дуже простий API, який дозволяє розробникам швидко розпочинати роботу. Він дає можливість зосередити увагу на розробці компонентів та забезпечує більшу гнучкість при виборі інструментів розробки та допоміжних інструментів та бібліотек для будь-якого типу завдання;

б) велика спільнота розробників: React є найпопулярнішою сучасною Front-end бібліотекою, завдяки чому в інтернеті можна знайти багато допоміжної інформації на різні типи запитань. Сьогодні вірогідність, що розробник натрапить на проблему чи баг, який раніше ніхто не бачив та не обговорював в інтернеті, близиться до нуля. Варто зазначити, що завдяки великій спільноті розробників все частіше з’являється нові функції та додатки, такі як Redux для керування станом або Next.js для рендерингу на стороні сервера;

в) швидкість рендерингу: напевно кожен веб-розробник знає про React та його неймовірну швидкість для майже будь-якої задачі. React відомий своєю ефективністю у роботі з відображенням великих даних завдяки віртуальному DOM і механізмам оптимізації рендерингу. Його механізм роботи суттєво відрізняється від всіх інших фреймворків та бібліотек. Такі інструменти, як Vue чи Angular перерисовують всю сторінку при кожній зміни її компонентів, але React працює зовсім інакше, бо використовує віртуальне DOM-дерево. Цей механізм дозволяє відстежувати зміни і перерисовувати лише ту частину сторінки, яка була змінена, що значно пришвидшує роботи веб-додатку при роботі з даними.

Ще багато всього можна сказати про переваги React та про те, який він необхідний для світу веб-розробки, але в жодному разі не можна забувати про такі потужні інструменти, як Vue чи Angular. Звісно є чимало аспектів, в яких Vue або Angular будуть кращими за React. Нижче визначимо, які переваги мають фреймоворки Vue або Angular над бібліотекою React:

а) React є бібліотекою, а не повноцінним фреймворком. Це означає, що він містить в собі основні інструменти, які можуть знадобитись при розробці інтерфейсів, але будь-який інший функціонал по типу маршрутизації або керуванням станом можуть вимагати використання додаткових бібліотек, таких як react-router-dom для маршрутизації чи axios для API-запитів. Vue та Angular, на відміну від React, містять в собі вбудовані функції та інструменти для більшості цих завдань, що дозволяє швидше розв’язувати поставлені задачі. Встановлення великої кількості сторонніх бібліотек може зробити розмір проєкту більшим, швидкість завантаження сторінок меншою та загалом суттєво знизити ефективність використання додатку;

б) проєкти на React не мають чіткої структури та стилю програмування. Через цю проблему можуть виникати суттєві розбіжності в якості коду та його архітектурі. Це особливо помітно, коли на проєкт приходить нова людина, яка раніше працювала з React в іншій компанії або команді. Такому розробнику на початкових етапах важко звикнути до нової

архітектури та патернів розробки, які використовує команда. Vue та Angular вимагає від розробників більш чіткої архітектури проєкту, що дозволяє швидше звикати до розробки нового проєкту та розвиватись, як програмісту.

Вище були зазначені всі найсуттєвіші переваги та недоліки бібліотеки React над фреймворками Vue та Angular. Завдяки ним можна зрозуміти, що всі ці інструменти розробки можуть використовуватись в комерційній розробці й це не буде помилкою, але React все ж таки є вибором для розробки програмної системи для онлайн-тренінгу ментального здоров'я. Такий додаток потребує великої кількості даних, які будуть постійно видалятись, створюватись та оновлятись. Саме з такої задачею React справляється найкраще. Також варто відзначити, що React має широкий вибір бібліотек та інструментів, що сприяє подальшому розвитку та розширенню функціоналу програми.

Для стилів був обраний CSS (англ. Cascading Style Sheets – каскадні таблиці стилів) – це така мова для опису зовнішнього вигляду компонентів системи, написаних за допомогою HTML. Але, варто відзначити, що в програмній системі для онлайн-тренінгу ментального здоров'я не будуть використовуватись звичайні файли з CSS кодом, замість них буде використана бібліотека Material-UI, яка містить в собі безліч готових варіантів зовнішнього вигляду компонентів додатку. Для модернізації таких компонентів використовується Inline-CSS, це такий стиль написання коду, при якому стилі прописуються прямо в HTML коді, що дозволяє робити структуру проєкту більш зрозумілою та простою.

Для написання розмітки сторінки та написання функціоналу було обрано використання .tsx файлів замість .ts та .html файлів. Таке розширення файлів дозволяє писати в одному місці TypeScript код для компонентів React, HTML код, який використовується для розмітки веб-сторінки, а також Inline-CSS стилі, щоб модернізувати зовнішній вигляд HTML-розмітки.

Не варто також забувати про важливість валідації форм та будь-яких значень, які вводить клієнт під час користування додатком. Це важливо тому, що відсутність якісної валідації має ризик викликати суттєві проблеми, які можуть забрати в

розробників чимало часу, а в деяких випадках навіть повністю порушити роботу додатку. На жаль, TypeScript не має в собі вбудованої якісної технології для валідації, він може лише перевіряти типи, а всі інші перевірки мають бути написані власноруч розробником. Саме для таких функцій була обрана бібліотека Zod, яка працює по принципу схем, тобто розробник пише схему для очікуваного результату та за допомогою функцій бібліотеки зрівнює, чи підходить результат по схемі. Це може бути дуже корисно, наприклад, коли потрібно перевірити чи підходить текст по різним умовам, таким як довжина, кількість слів та інші різні умови, або коли потрібно перевірити чи правильний вид має об’єкт, який буде відправлений на сервер.

4.2 Вибір програмних засобів реалізації для Mobile частини

Для написання Mobile частини буде також використана мова програмування TypeScript та бібліотека React. Вище було описано, чому були обрані саме ці технології для розробки програмної системи для онлайн-тренувань ментального здоров'я. На рівні команди було прийняте рішення в якості мобільної частини розробити адаптивну під будь-які пристрої версію додатку. Такий підхід має багато плюс на початкових етапах існування продукту. Визначимо основні плюс адаптивної версії сайту на повноцінним мобільним застосунком:

а) доступність додатку: мобільна версію сайту легко доступна через будь-який сучасний браузер за умовою підключення до мережі Інтернет. Таким чином користувачам не потрібно встановлювати додаткові програми, щоб користуватися продуктом. На відміну від мобільної версії сайту мобільний додаток потребує від користувача додатково витрачених часу та зусиль. Використання мобільної версії сайту дозволить зменшити поріг входу для користувачів на початкових етапах проєкту;

б) витрати та труднощі розробки: зазвичай мобільна версія сайту вимагає набагато менше витрат, ніж створення повноцінного мобільного додатку. Мобільна версія сайту використовує один код для всіх пристроїв, а мобільний застосунок потребує створення нового окремого застосунку,

тобто потребує вдвічі більших витрат часу та зусиль, ніж створення адаптивної версії сайту

в) оновлення та зміни: в адаптивну версію додатку можна вносити зміни без перешкод для користувачів. При кожному оновленні повноцінного мобільного додатку користувачам потрібно завантажувати оновлення, що вимагає додаткових зусиль, а для оновлення адаптивної версії додатку користувачу треба лише оновити сторінку;

г) пошукова видимість: мобільна версія сайту індексується пошуковими системами, що сприяє підвищенню видимості у пошукових результатах. Повноцінний мобільний додаток потребує додаткових зусиль для просування в магазинах по типу Play Market або App Store.

Проаналізувавши всі основні переваги адаптивної версії сайту над повноцінним мобільним додатком можна зробити висновок, що на початкових етапах існування проєкту немає особливого сенсу створювати окремий мобільний додаток. Такий застосунок буде мати сенс після певного етапу успіху проєкт, що допоможе ще більшій кількості людей узнати про програмну систему для онлайн-тренінгу ментального.

4.3 Опис структури проєкту

Раніше було зазначено, що в якості IDE для розробки проєкту буде використано Visual Studio Code, яке є найкращим безкоштовним середовищем для розробки під TypeScript та React.

Вся Front-end частина програмної системи складається з трьох головних директорій та файлу конфігурації package.json.

Перша папка має назву node_modules, в ній зберігаються все зовнішні модулі та бібліотеки необхідні для роботи React та проєкту загалом. Ця папка створюється автоматично разом з проєктом і доповнюється кожен раз, коли розробник завантажує новий зовнішній інструмент.

Другою папкою є public, яка містить в собі файл index.html. Цей файл є найголовнішим в усьому проєкті, це пов’язано з тим, що браузери сприймають

тільки HTML файли, тому весь код проєкту при компіляції перетворюється на HTML та JavaScript і відображається через цей файл. Без нього робота додатку неможлива. Крім того, в директорії public є ще дві інші папки, а саме locales та assets. Папка locales потрібна для правильної роботи локалізації додатку. В ній зберігаються переклади всього тексту, присутнього в додатку, на різні мови. Для кожної мови своя папка, наприклад, uk для української або en для англійської. Папка assets в свою чергу використовується для зберігання будь-яких зображень, які використовуються в проєкті. Ця папка зберігається саме в директорії public через те, що вона є головною для правильної роботи додатку.

Третьою та основною для розробника папкою є src. Ця директорія складається з багатьох різних частин. Вона містить в собі папки utils та helpers, в яких зберігаються основні допоміжні функції проєкту. Крім папок utils та helpers у директорії src присутні інші папки та файли:

а) hooks – директорія, в якій зберігаються кастомні хуки, які можна використовувати по всьому додатку. Ці хуки дуже схожі на звичайні функції, але мають трохи іншу структуру;

б) types – папка, в якій зберігаються абсолютно всі типи та об’явлення класів. Без цієї папки було б дуже важко працювати з TypeScript та використовувати статичну типізацію;

в) constants – директорія, в якій знаходяться всі об’явлені константи, що використовуються по всьому додатку;

г) pages – папка, що містить в собі абсолютно всі сторінки, які використовуються в проєкті;

д) components – найбільша папка в цій директорії. В ній зберігаються всі невеличкі частини проєкту, які можна всюди перевикористовувати. Саме такі частини в React називаються компонентами. В проєкті було дотримано принцип організації компонентів у різні папки в залежності від того, на яких сторінках вони використовуються. Це дозволяє зберігати код структуровано та зрозуміло. Таким чином, якщо компонент використовується лише на одній сторінці, тоді вона знаходиться у

відповідній папці цієї сторінки. Проте, якщо компонент використовується на різних сторінках, вона додається до спільної папки shared у розділі компонентів. Такий підхід дозволяє уникнути дублювання коду та забезпечити легке управління компонентами проєкту;

е) App.tsx – файл, в якому прописується вся маршрутизація проєкту. Вона має вигляд списку, в якому ключ це URL сторінки, а значення це .tsx файл необхідної для відображення сторінки з папки pages.

Нижче можна побачити, як виглядає в IDE структура директорії src для Front-end частини розробленої програмної системи (див. рис. 4.1).

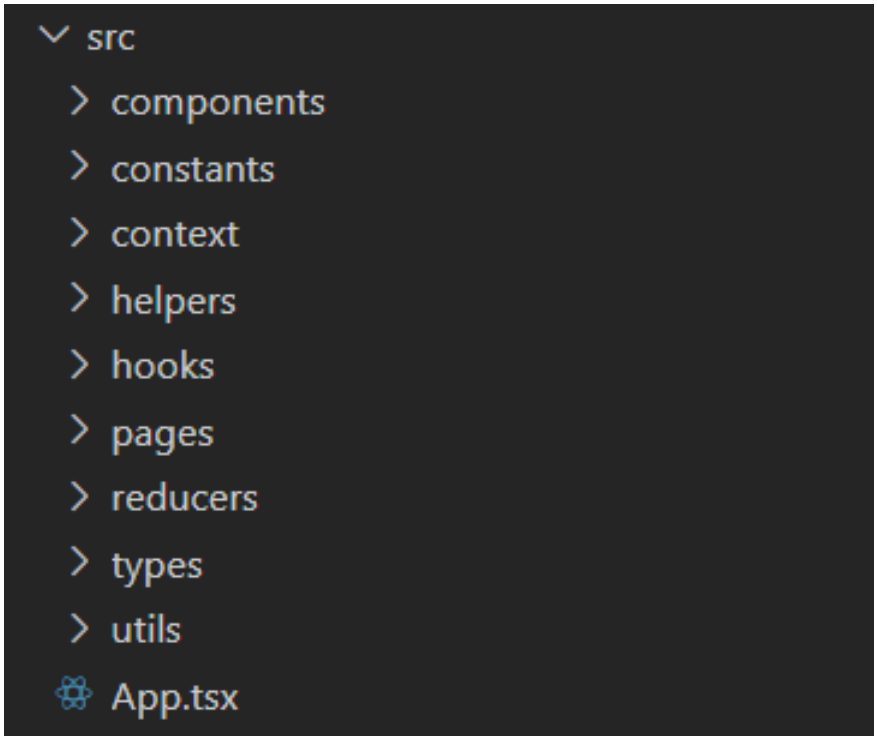


Рисунок 4.1 – Структура папки src (рисунок виконаний самостійно)

Варто зазначити, що в проєкті також є директорія __tests__, в якій зберігається код для всіх юніт-тестів, що є в проєкті.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Мануальне тестування

Одним з двох видів тестування, які будуть використанні в програмній системі для онлайн-тренувань ментального здоров'я, є мануальне тестування. При використанні мануального тестування розробник чи команда тестувальників власноруч перевіряють всі наявні в проєкті функції. Такий вид тестування є досить корисним для маленьких команд, а також має певні переваги та недоліки. Визначимо основні переваги мануального тестування:

а) такий вид тестування дозволяє команді тестувальників швидко адаптуватися до змін у вимогах чи функціоналі програми. Вони можуть виявляти недоліки, які автоматизовані засоби тестування можуть пропустити через обмежену функціональність. Майже завжди людина буде уважніша за комп’ютер, тому побачить ті проблеми, які не зможу знайти автоматизована система;

б) людський фактор в тестуванні дає можливість застосовувати інтуїцію та творчість для виявлення проблем у програмному забезпеченні та дослідження різних сценаріїв використання, які не завжди передбачаються автоматизованими тестами. Інтеграція цього фактору допомагає відтворювати реальні умови використання продукту, оскільки користувачі можуть взаємодіяти з програмним забезпеченням у непередбачуваних чи незвичайних способах. Таким чином, людський фактор у тестуванні дозволяє виявити проблеми та вразливості, які можуть залишитися невиявленими в інших методах тестування.

На перший погляд, мануальне тестування може здаватися якісним підходом, оскільки дозволяє тестувальникам виявляти проблеми та вразливості, які можуть залишитися невиявленими в інших методах тестування. Однак цей підхід також має дуже суттєві недоліки, які можуть впливати на вибір виду тестування для проєкту. Це пов’язано з тим, що на кожному проєкті різна команда, різний бюджет та різні

потреби в об’ємі тестування. Визначимо такі суттєві недоліки мануального тестування:

а) мануальне тестування може бути часо- та ресурсоємким процесом, особливо при великих проєктах. Вимагає багато робочих годин від команди тестувальників. В цьому аспекті юніт-тести будуть набагато кращими, бо їх потрібно створити один раз і далі запускати командою в терміналі. Мануальне тестування потрібно повторювати кожен раз в повному обсязі;

б) результати мануального тестування можуть бути сильно залежними від індивідуальних оцінок інспекторів. Такий підхід може бути, як корисним, так і шкідливим. Це може призвести до пропуску деяких проблем або надмірної уваги до інших, менш важливих аспектів. Тому в комерційних проєктах майже ніколи не можна обмежуватись лише мануальним тестуванням, потрібно створювати юніт-тести або end-to-end тести;

в) збільшенні обсягу програми або частоти випусків нових версій, об'єм ручного тестування може стати надто великим для ефективного управління. З ростом обсягу програми чи збільшенням частоти випусків нових версій, цей процес може стати неможливим для ефективного виконання. Також важливо враховувати, що при масштабуванні мануального тестування збільшується ймовірність помилок та непроглядних проблем через залучення більшої кількості людей до процесу. Координація та забезпечення однорідності у виконанні тестів може стати значним викликом, особливо якщо тестування виконується на різних майданчиках або командами. Отже, обмеженість масштабованості мануального тестування може стати серйозною перешкодою для компаній, які швидко розвиваються чи мають великі програмні продукти зі складною функціональністю.

Враховуючи все зазначене вище, можна сміливо казати, що мануальне тестування в жодному разі не може бути єдиним видом тестування на проєкті. Воно є дуже корисним на початкових етапах та поки проєкт не став надто великим.

Навіть попри всі недоліки, жоден комерційний проєкт не може обійтись без мануального тестування, бо його користь неможливо переоцінити. Хоча мануальне тестування і є корисним для невеликих проєктів, його потрібно підстраховувати використовуючи юніт-тести або end-to-end тести для складних частин додатку.

5.2 Юніт-тестування

Юніт-тестування є одним з найкращих та одним з найпоширеніших видів тестування. Такі тести є майже в кожному комерційному проєкті, бо внесення змін в проєкт просто неможливо уявити без допомоги цих тестів. Вони мають дуже багато переваг. Визначимо такі переваги юніт-тестування:

а) юніт-тести зазвичай швидко виконуються, оскільки вони тестують невеликі фрагменти коду. Такі тести потребують в середньому приблизно пів секунди на перевірку одного компонента через те, що ці тести перевіряють кожен компонент системи окремо;

б) такий вид тестів допомагає виявити помилки та проблеми на ранніх етапах розробки, коли їх виправлення є дешевшим та менш часовим. Зазвичай, на комерційних проєктах після кожної зміни в терміналі виконується команда, яка запускає виконання юніт-тестів. Якщо розробник помилився в якомусь місці, то юніт-тести це одразу покажуть і не дадуть помилкам залишатись в проєкті;

в) юніт-тести забезпечують високий рівень покриття коду, оскільки кожна одиниця програми тестується окремо. Завдяки цьому можна бути впевненим, що кожен компонент системи виконує свою задачу без помилок;

г) такі тести допомагають фіксувати поточний стан функціональності програми, забезпечуючи безпеку при внесенні змін у код. Як було зазначено вище, юніт-тести виконуються при кожній зміні коду, що дозволяє уникати майже всіх помилок при розробці.

На перший погляд може здаватись, що юніт-тести це ідеальний інструмент для тестування додатків та пошуку помилок в коді. Але це не зовсім так, бо навіть

такий потужний інструмент має чималу кількість недоліків, які теж потрібно враховувати при виборі типу тестування для проєкту. Визначимо такі недоліки:

а) юніт-тести перевіряють кожну частину програми окремо. Тому вони можуть не можуть знайти помилки, які трапляються при взаємодії двох різних компонентів програми між собою. В такій ситуації на допомогу юніт-тестуванню приходять end-to-end тести;

б) тести такого типу потребують постійного оновлення та підтримки, оскільки разом з розвитком програми можуть змінюватися і вимоги до тестів. Через це розробнику не варто забувати про зміну тестів, коли вноситься якась важлива зміна в компонент системи, який тестується;

в) виконання та підтримка юніт-тестів може стати витратною задачею, особливо коли мова йде про великі проекти. Написання та підтримка тестів вимагає значних зусиль та часу від розробників. Крім того, при роботі з великою кодовою базою необхідно враховувати, що з часом кількість тестів може значно зрости, що призведе до збільшення витрат на їх підтримку. Додатково, для виконання тестів може знадобитися спеціалізоване програмне забезпечення або обладнання, що також збільшує загальні витрати на проект;

г) написання та підтримка юніт-тестів вимагає від розробників певних навичок та розуміння принципів тестування. Через це потрібно враховувати, що на проєкті має бути людина, в якої є досвід в написанні юніт-тестів для великих комерційних проєктів.

В розробці веб-сайтів юніт-тести також необхідні, особливо при розробці з використанням сучасних Front-end фреймворків та бібліотек. В нашому проєкті для виконання та написання юніт-тестів буде використана бібліотека Vitest, яка є сучасним аналогом відомої бібліотеки під назвою Jest. Вони досить схожі в багатьох аспектах, але Vitest новіша і містить в собі кілька нових функцій, яких не має в Jest. Також нижче показано приклад одного з тестів для Footer частини нашої програмної системи.

```
describe("Footer", () => {
  it("renders Footer component with correct links", () => {
    const container = render(<Footer />);
    expect(container).toBeDefined();
    const logo = container.getByAltText("Udemy Logo");
    expect(logo).toBeInTheDocument();
    const termsLink = container.getByText("Terms");
    expect(termsLink).toBeInTheDocument();
    expect(termsLink).toHaveAttribute("href", "/terms");
    const aboutUsLink = container.getByText("About Us");
    expect(aboutUsLink).toBeInTheDocument();
    expect(aboutUsLink).toHaveAttribute("href", "/about-us");
    const faqLink = container.getByText("FAQ");
    expect(faqLink).toBeInTheDocument();
    expect(faqLink).toHaveAttribute("href", "/faq");
    const languageSelector = container.getByText("Select Language");
    expect(languageSelector).toBeInTheDocument();
  });
});
```

Юніт-тести є дуже потужним інструментом для виконання тестування в проєкті, але навіть вони не можуть бути використані без іншого додаткового виду тестування, наприклад, мануального або end-to-end тестування. Незважаючи на це, такий вид тестів є найважливішим аспектом тестування будь-якого сучасного проєкту, бо він дозволяє зберігати час та сили розробників і тестувальників.

ВИСНОВКИ

Під час цього дослідження було розроблено програмну систему для онлайн-тренувань ментального здоров'я, визначивши основні вимоги до системи та обрали необхідні технології та інструменти для створення повноцінної платформи. Провівши аналіз аналогічних систем, було виявлено унікальні функції та особливості, які дозволили нам спроєктувати систему з індивідуальними рекомендаціями для користувачів та ефективним зворотнім зв'язком від менторів.

У ході роботи над проектом були успішно виконані всі заплановані етапи відповідно до графіку. Спочатку були визначені ключові функції та розробили модель системи. Front-end або клієнтська частина системи була реалізована з використанням технологій Material-UI та React, щоб забезпечити користувачам простий інтуїтивно зрозумілий досвід.

Для оптимізації процесів керування проектом використовувалась Jira та методологія Scrum. Процес тестування складався з написання юніт-тестів та виконання мануального тестування.

Мета цієї програмної системи полягає в тому, щоб допомогти користувачам покращити своє ментальне здоров'я завдяки індивідуальним планам, які можна розробляти спільно з менторами, а також за рахунок різноманітних ресурсів для самостійного навчання. При використанні додатку користувач отримує можливість перегляду великої кількості інформації, пов’язаної з психологією, а також отримання зв’язку від психологів та менторів.

Ця система стане корисним інструментом для людей, які прагнуть отримати допомогу у покращенні свого ментального здоров'я, сприяючи саморозвитку та саморегуляції, що допомагає справлятися зі стресом і підтримувати психологічне благополуччя.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Головна Headspace URL: https://www.headspace.com (дата звернення: 13.05.2024).

2. Головна BetterHelp URL: https://www.betterhelp.com (дата звернення: 13.05.2024).

3. Головна VerywellMind URL: https://www.verywellmind.com (дата звернення: 13.05.2024).

4. Довідник по CSS властивостям URL: https://css.in.ua/css/properties (дата звернення: 13.05.2024).

5. MaterialUI Learning resources URL: https://mui.com/material-ui/getting-started/learn/ (дата звернення: 13.05.2024).

6. Tutorial: Intro to React URL: https://legacy.reactjs.org/tutorial/tutorial.html (дата звернення: 13.05.2024).

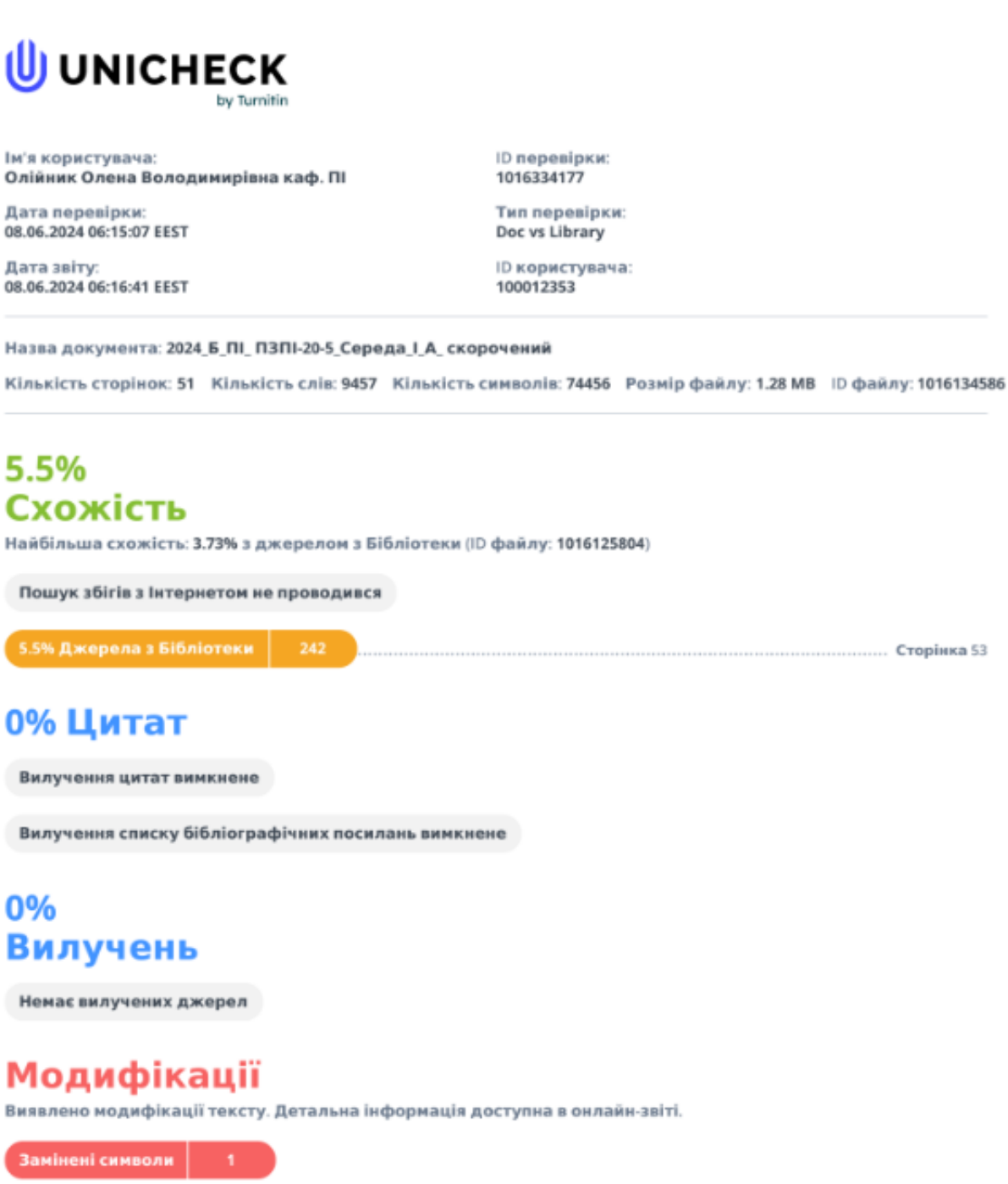
7. Що таке Jira і як з нею працювати | IAMPM. IAMPM. URL: https://iampm.club/ua/blog/shho-take-jira-i-yak-z-neyu-praczyuvati/ (дата звернення: 13.05.2024).

8. Introduction to UI and UX Design URL: https://www.codecademy.com/learn/intro-to-ui-ux (дата звернення: 13.05.2024).

9. Axios API | Axios Docs URL: https://axios-http.com/ru/docs/api_intro (дата звернення: 13.05.2024).

10. JavaScript vs TypeScript: якій мові надаєте перевагу ви? URL: https://dou.ua/forums/topic/48306/ (дата звернення: 03.06.2024).

ДОДАТОК А



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016334177

Дата перевірки:
08.06.2024 06:15:07 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
08.06.2024 06:16:41 EEST

ID користувача:
100012353

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

Кількість сторінок: 51 Кількість слів: 9457 Кількість символів: 74456 Розмір файлу: 1.28 MB ID файлу: 1016134586

5.5%
Схожість

Найбільша схожість: 3.73% з джерелом з Бібліотеки (ID файлу: 1016125804)

Пошук збігів з Інтернетом не проводився

5.5% Джерела з Бібліотеки 242 ..... Сторінка 53

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

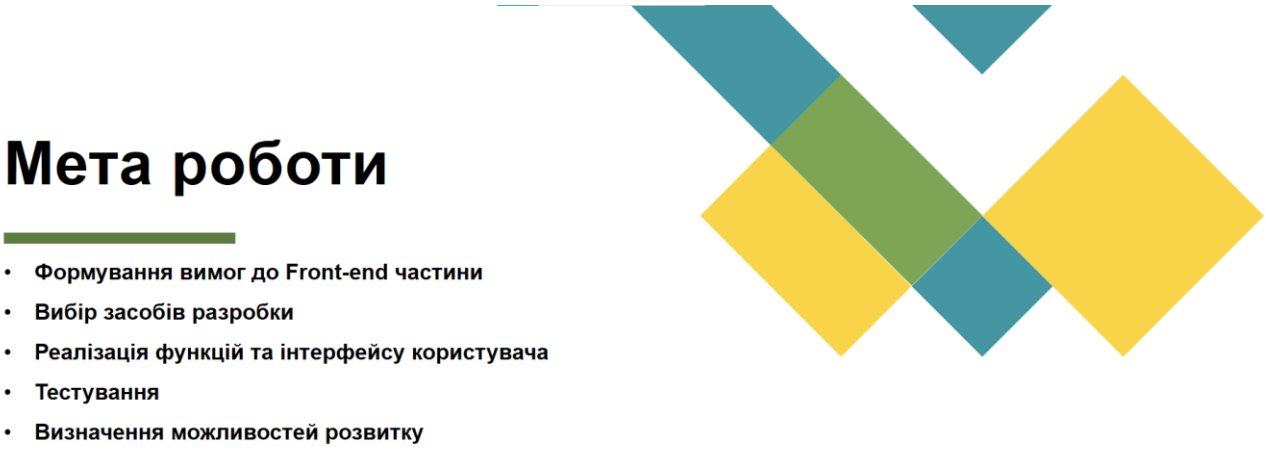
Модифікації

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

Замінені символи 1

ДОДАТОК Б


Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Кваліфікаційна робота бакалавра
Програмна система для онлайн-тренінгу ментального здоров'я.
Mobile Application, Front-end.
Виконав
ст. гр. ПЗПІ-20-5
Середа І. А.
Науковий керівник
доц. кафедри ПІ
Побіженко І. О.


Мета роботи
Формування вимог до Front-end частини
Вибір засобів разробки
Реалізація функцій та інтерфейсу користувача
Тестування
Визначення можливостей розвитку

Формування вимог до Front-end частини

- Функціональні вимоги
- Вимоги до безпеки
- Вимоги до інтерфейсу
- Вимоги до адаптивності

Вибір засобів разробки






Приклад реалізації

```
const cart_reducer = (state: any, action: any) => {
  const userId = getUser()?.id;

  if (action.type === ADD_TO_CART) {
    const tempArr = state.cart[userId].filter(
      (item: CourseType) => item.id === action.payload.id
    );
    if (tempArr.length < 1) {
      return {
        ...state,
        cart: {
          ...state.cart,
          [userId]: [...state.cart[userId], action.payload],
        },
      };
    }
    return {
      ...state,
    };
  }

  if (action.type === REMOVE_CART_ITEM) {
    const tempCart = state.cart[userId].filter(
      (item: CourseType) => item.id !== action.payload
    );
    return {
      ...state,
      cart: {
        ...state.cart,
        [userId]: tempCart,
      },
    };
  }
```

```
  if (action.type === GET_CART_TOTAL) {
    if (localStorage.getItem("cart") === JSON.stringify({})) return 0;
    const total_amount = state.cart[userId].reduce(
      (total: number, cartItem: CourseType) => {
        total += Number(cartItem.price);
        return total;
      },
      0
    );
    return {
      ...state,
      total_items: state.cart[userId].length,
      total_amount,
    };
  }

  if (action.type === BUY_COURSES) {
    fetch(`${API_LINK}/api/purchase`, {
      method: "POST",
      body: state.cart[userId],
    });
    return {
      ...state,
      cart: {
        ...state.cart,
        [userId]: [],
      },
    };
  }

  if (action.type === CLEAR_CART) {
    return {
      ...state,
      cart: {
        ...state.cart,
        [userId]: [],
      },
    };
  }

  throw new Error(`No matching "${action.type}" - action type`);
};

export default cart_reducer;
```

Приклад реалізації

```
import { Message, MessageWithStatus, SidebarChats } from "../types/chat.types";

export function mapConversationStatus(
  messages: Message[]
): MessageWithStatus[] {
  return messages.map((message) => ({ ...message, status: "success" }));
}

export function getActiveConversation(
  conversations: SidebarChats[],
  activeConversationId: number
) {
  return conversations.length > 0
    ? conversations.find((conv) => conv.user.id === activeConversationId) ||
      conversations[0]
    : null;
}
```

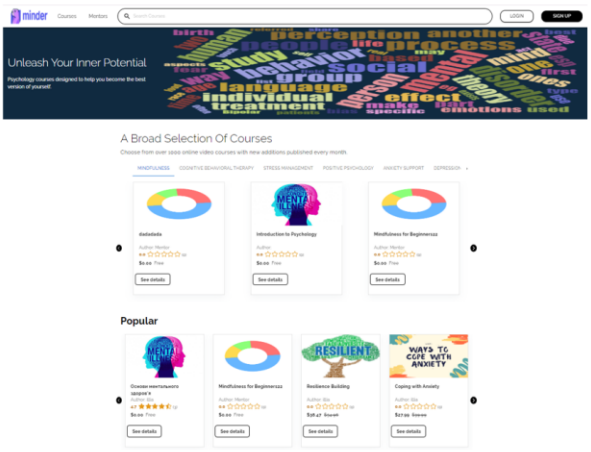
```
export function getCurrentTime(time: string) {
  const date = new Date(time);

  return date;
}

export function fDateToChatTime(date: Date | string | number): string {
  const utc =
    new Date(date + "z").toString() === "Invalid Date"
      ? new Date(date)
      : new Date(date + "z");
  let hours = getHours(utc);
  let minutes: number | string = getMinutes(utc);
  const AM_PM = hours >= 12 ? "PM" : "AM";

  if (hours > 12) hours -= 12;

  if (minutes.toString().length === 1) minutes = `0${minutes}`;

  return `${hours}:${minutes} ${AM_PM}`;
}

export function getChatDate(date: Date | string | number, locale: "uk" | "en") {
  const yearsDif = differenceInYears(new Date(), new Date(date));

  return yearsDif > 0 ? fDate(date) : fDateToFullMonthDay(date, locale);
}

export function areDifferentDays(firstDate: Date, secondDate: Date) {
  const areSameDays = firstDate.getDay() === secondDate.getDay();
  const areSameMonths = firstDate.getMonth() === secondDate.getMonth();
  const areSameYears = firstDate.getFullYear() === secondDate.getFullYear();

  return areSameDays && areSameMonths && areSameYears;
}
```

Приклад реалізації

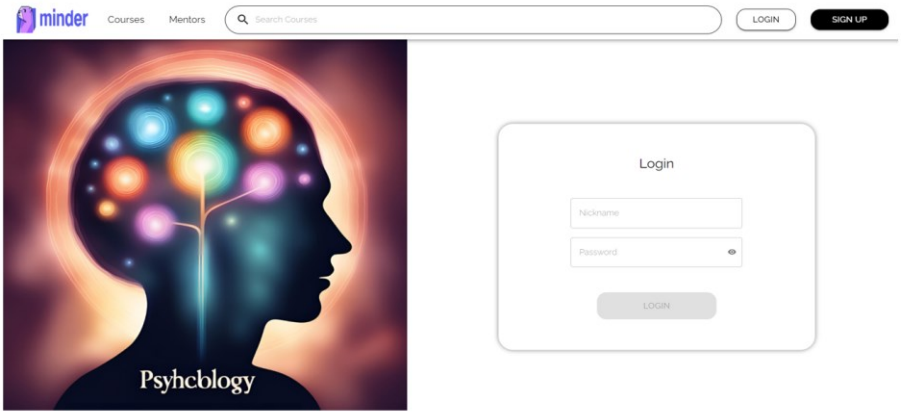
```ts
import { useEffect, useState } from "react";

const useDebounce = (value: string, delay: number): string => {
  const [debouncedValue, setDebouncedValue] = useState(value);

  useEffect(() => {
    const handler = setTimeout(() => {
      setDebouncedValue(value);
    }, delay);

    return () => {
      clearTimeout(handler);
    };
  }, [value, delay]);

  return debouncedValue;
};

export default useDebounce;
```

```ts
export default function useResponsive(
  query: Query,
  key?: Key,
  start?: Start,
  end?: End
) {
  const theme = useTheme();

  const mediaUp = useMediaQuery(theme.breakpoints.up(key as Key));

  const mediaDown = useMediaQuery(theme.breakpoints.down(key as Key));

  const mediaBetween = useMediaQuery(
    theme.breakpoints.between(start as Start, end as End)
  );

  const mediaOnly = useMediaQuery(theme.breakpoints.only(key as Breakpoint));

  if (query === "up") {
    return mediaUp;
  }

  if (query === "down") {
    return mediaDown;
  }

  if (query === "between") {
    return mediaBetween;
  }

  if (query === "only") {
    return mediaOnly;
  }
}
```

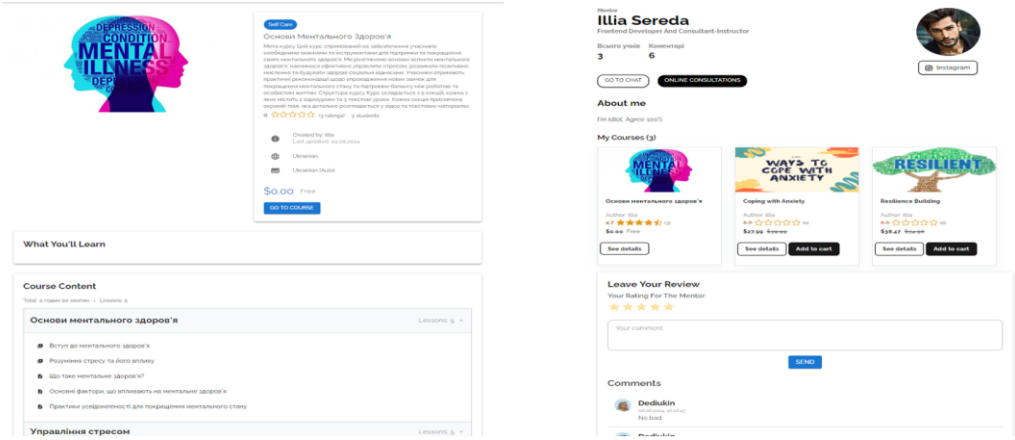
Інтерфейс користувача



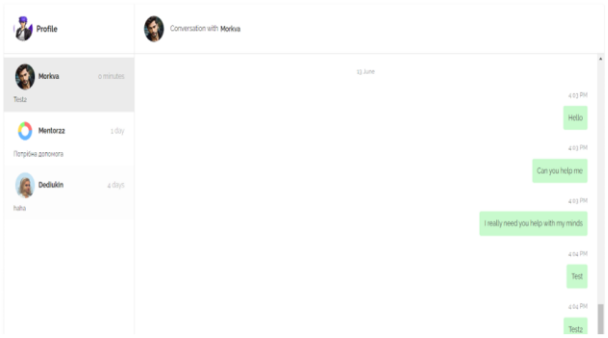
Інтерфейс користувача



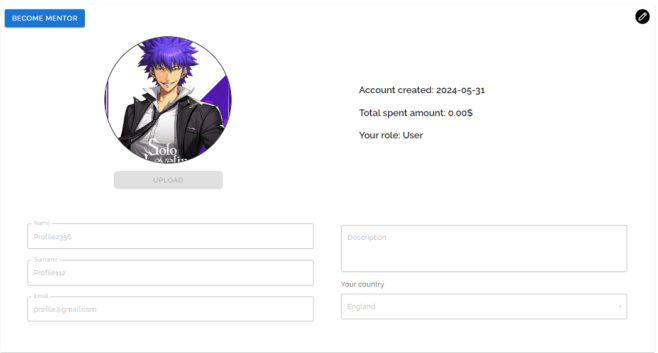
Інтерфейс користувача

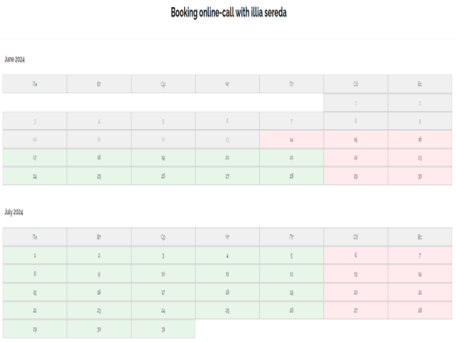


Інтерфейс користувача



Інтерфейс користувача



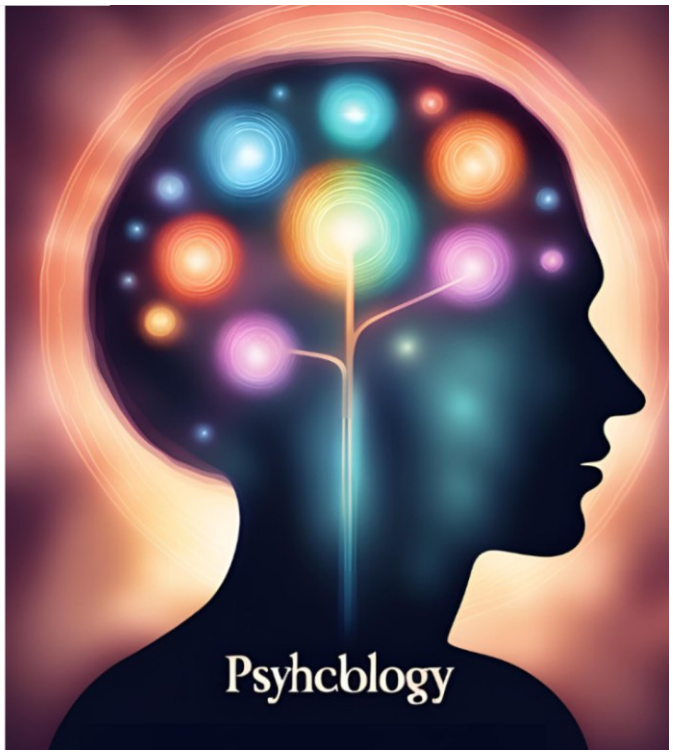


Тестування

ID	Description	Test Space	Input Data	Action	Expected Result	Actual Result	Test Result
1	Відправлення повідомлень в чаті	Локально	Чат між користувачами та текст повідомлення	Ввести повідомлення та натиснути “Enter”	Повідомлення відобразиться локально з правильним часом відправлення	Співпадає	+
2	Бронювання онлайн-зустрічі	Локально	Час та опис броні	Заповнити форму бронювання	Якщо в дні немає вільної години, тоді цей день відмічається червоним	Співпадає	+
3	Реєстрація нового користувача	Локально	Дані користувача	Заповнити та надіслати форму	Дані користувача успішно отримані з бази даних та збережені в сесії	Співпадає	+
4	Створення нового курсу	Локально	Дані про курс	Заповнити форму створення курсу	Курс відобразився локально з можливістю одразу його редагувати	Співпадає	+

Можливості розвитку

- Розширення функціоналу
- Покращення користувацького досвіду
- Додавання тестів до матеріалів курсу
- Збільшення кількості мов для локалізації



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

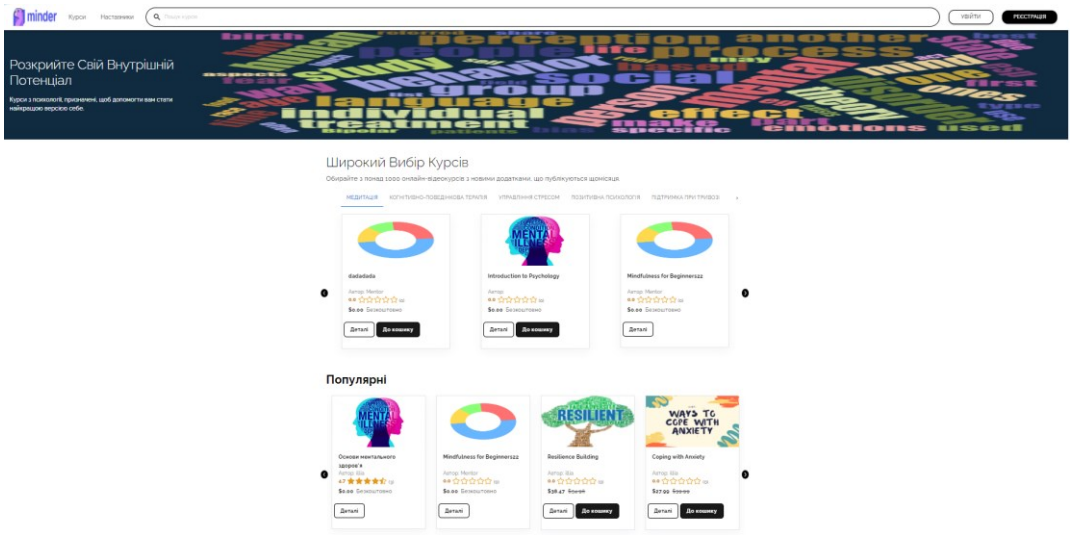
Повноцінне користування програмною системою можливо лише при наявності базових навичок користування браузером та підключення до мережі Internet.

3 СПЕЦИФІЧНІ ВИМОГИ

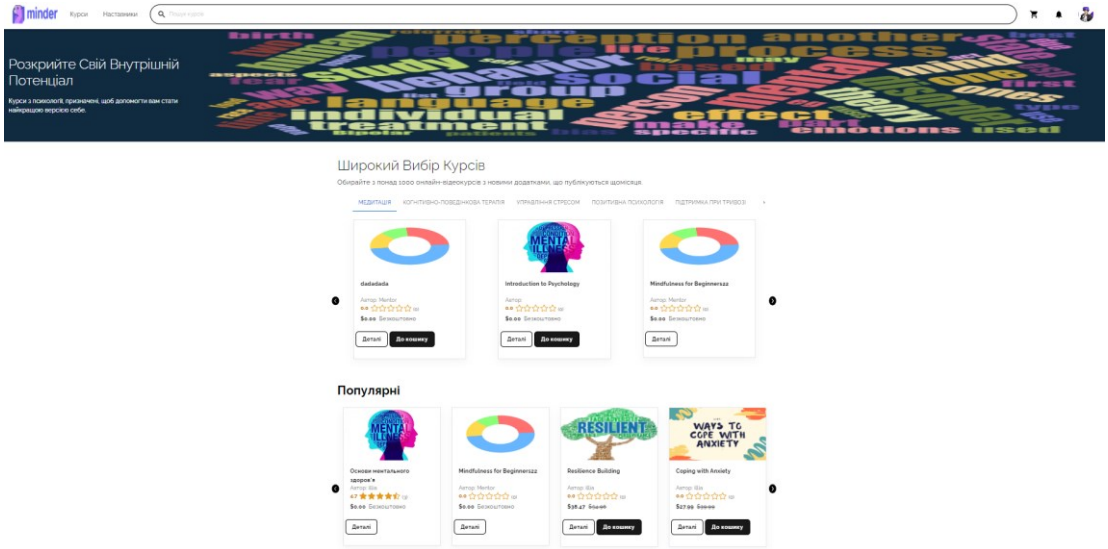
3.1 Вимоги до зовнішнього інтерфейсу

3.1.1 Користувацькі інтерфейси

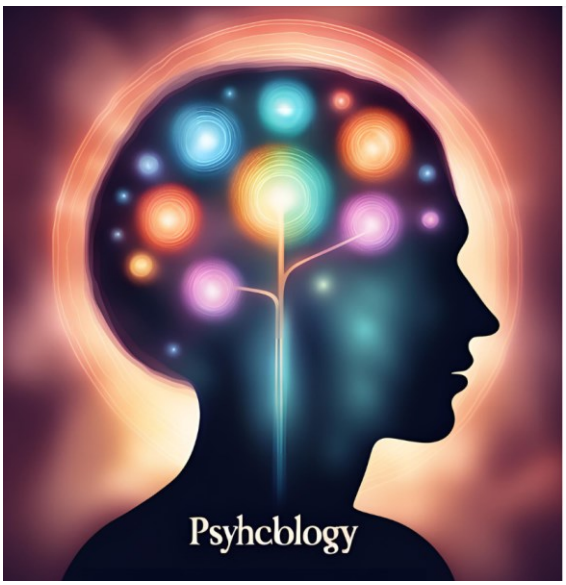
Головна сторінка для незареєстрованого користувача.

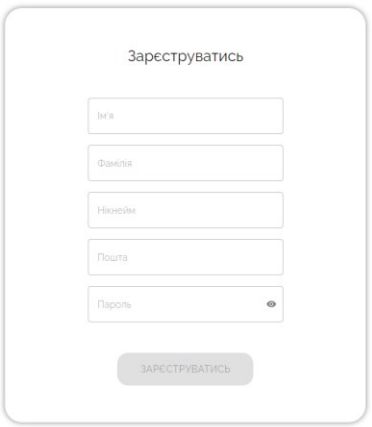


Головна сторінка для авторизованого користувача.

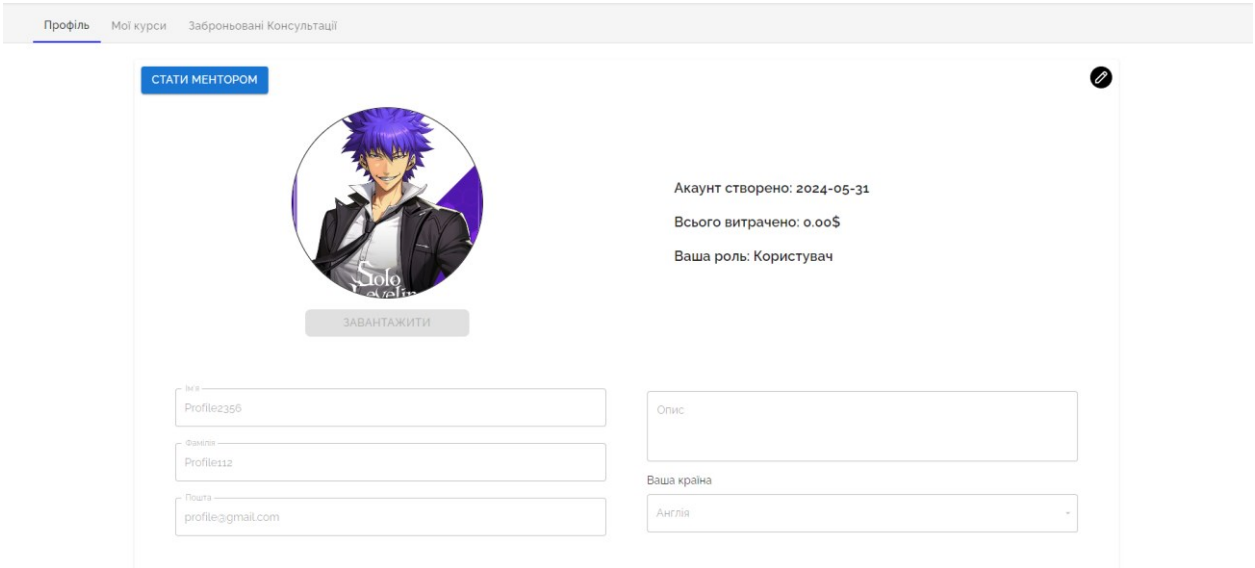


Сторінки для авторизації та реєстрації.

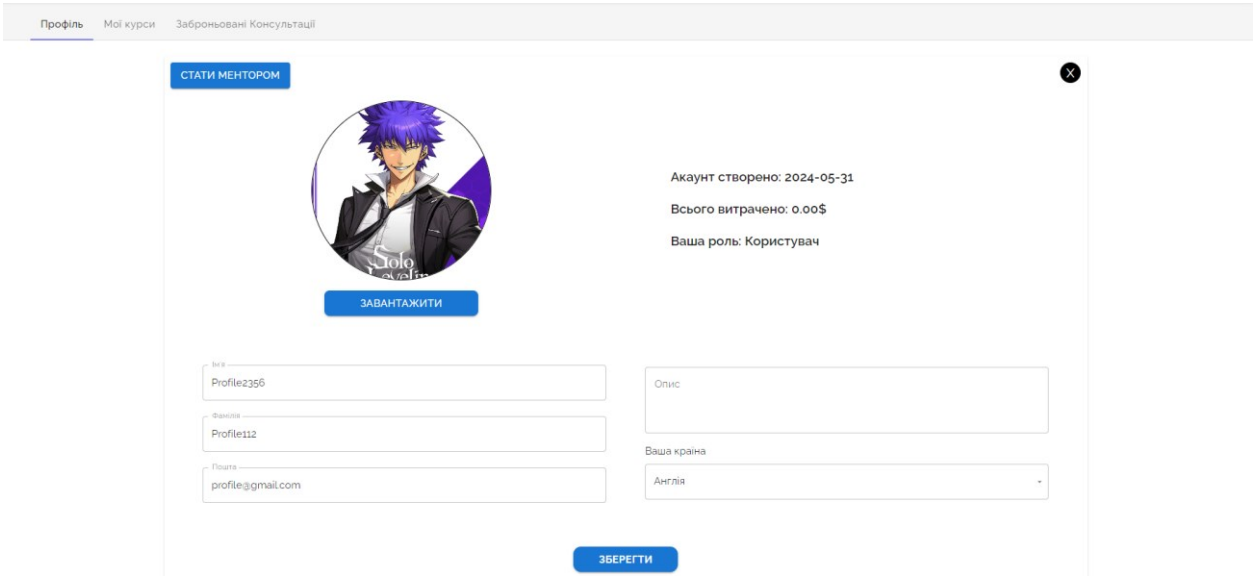




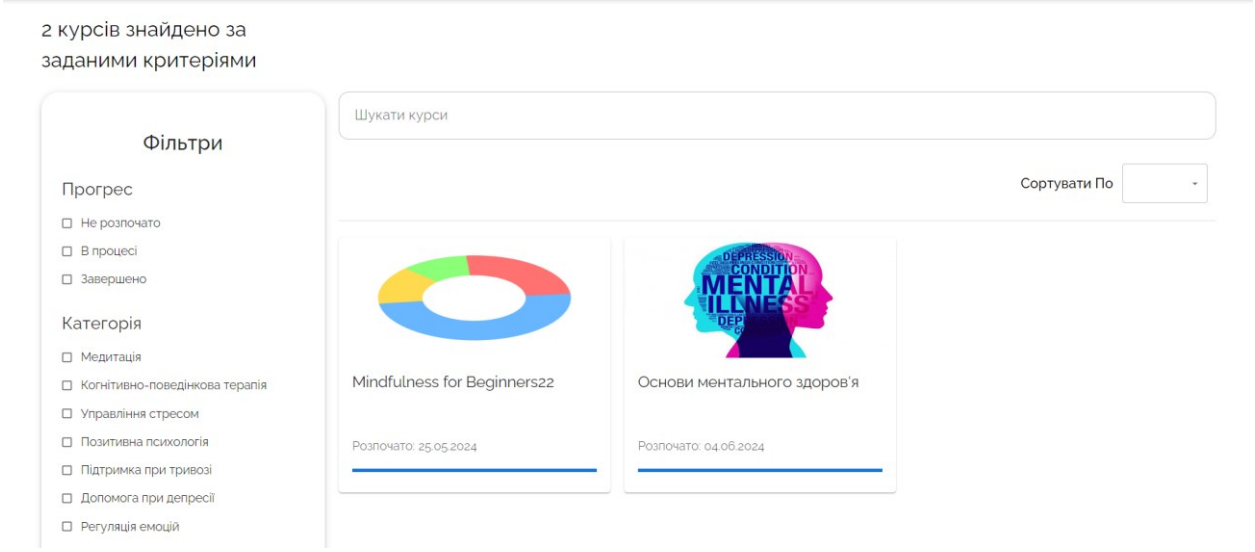
Сторінка профілю аторизованого користувача без режима редагування.



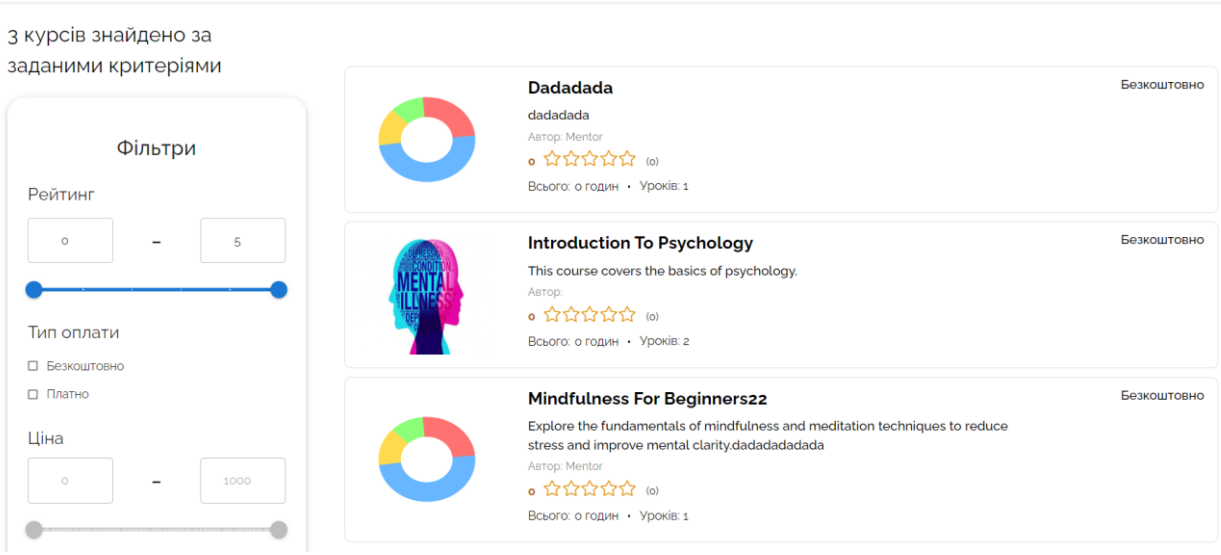
Сторінка профілю авторизованого користувача в режимі редагування.



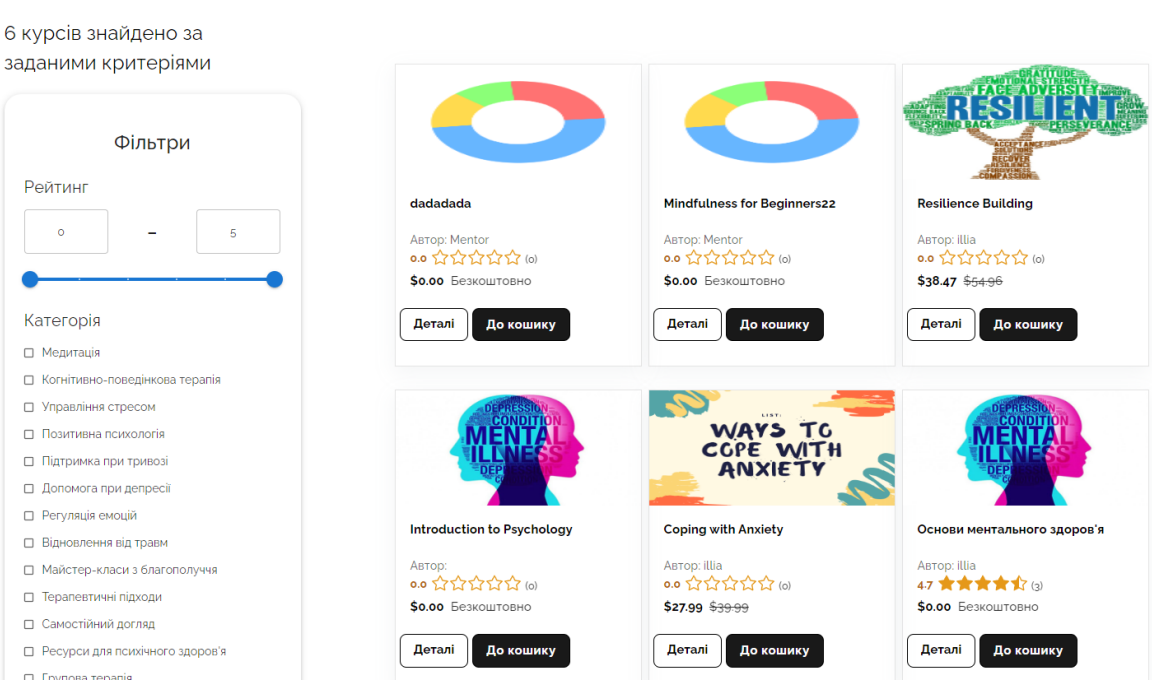
Сторінка прогресу користувача по курсах.



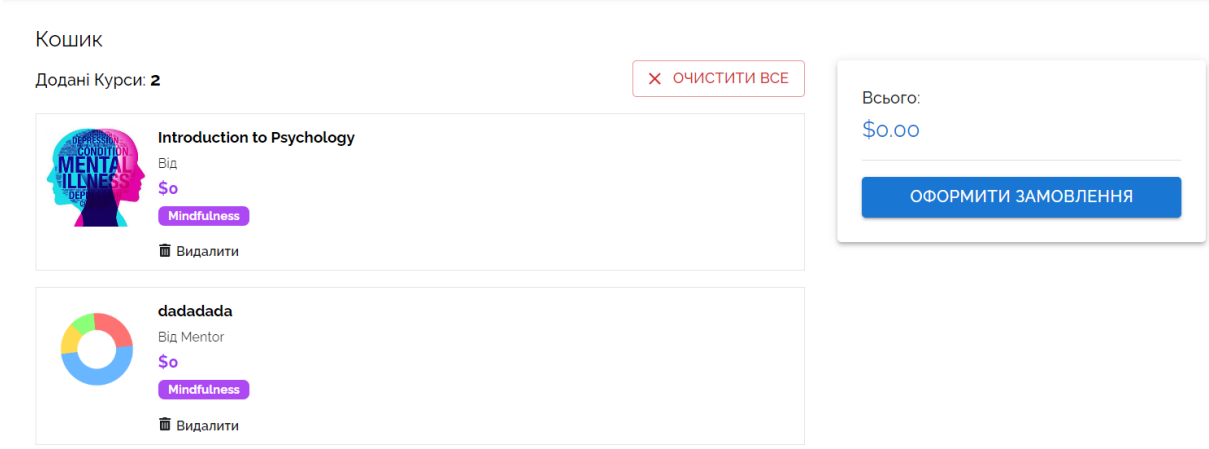
Сторінка курсів за категоріями.



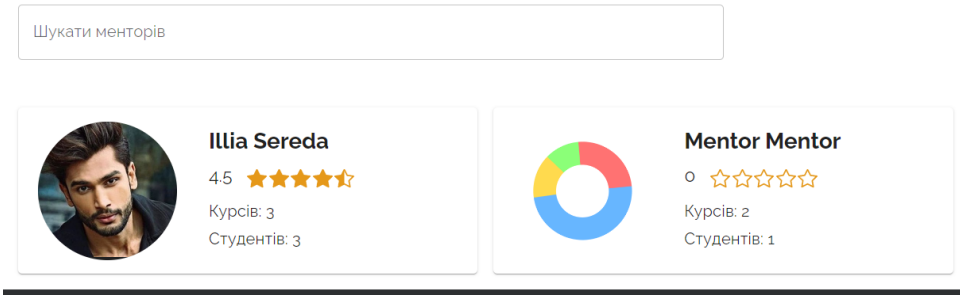
Сторінка пошуку курсів на сайті.



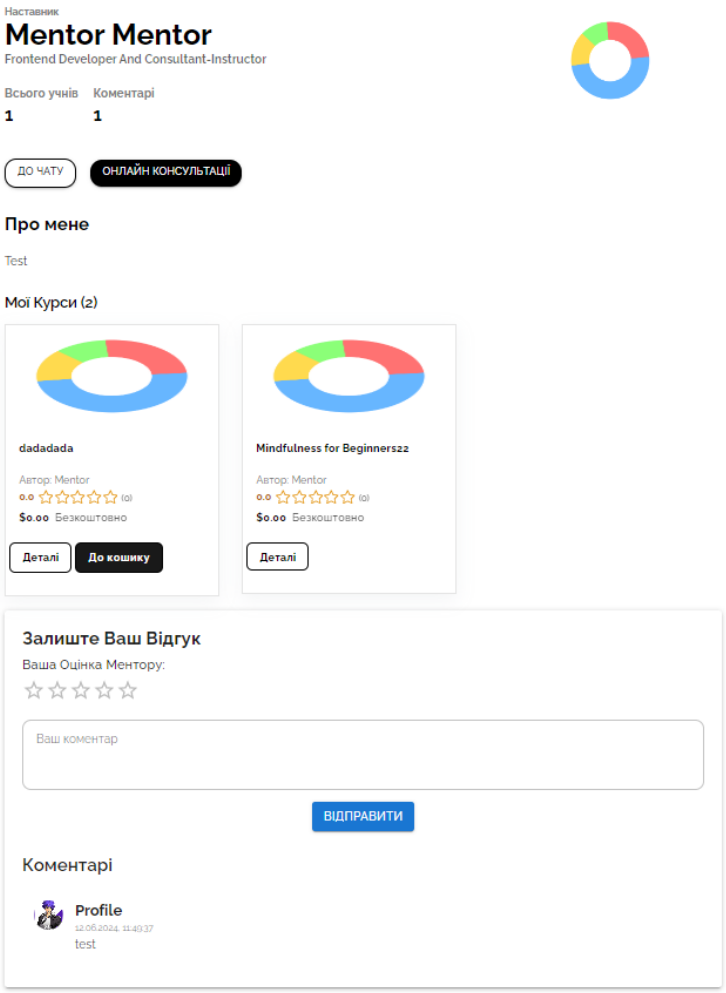
Сторінка кошику.



Сторінка пошуку менторів.

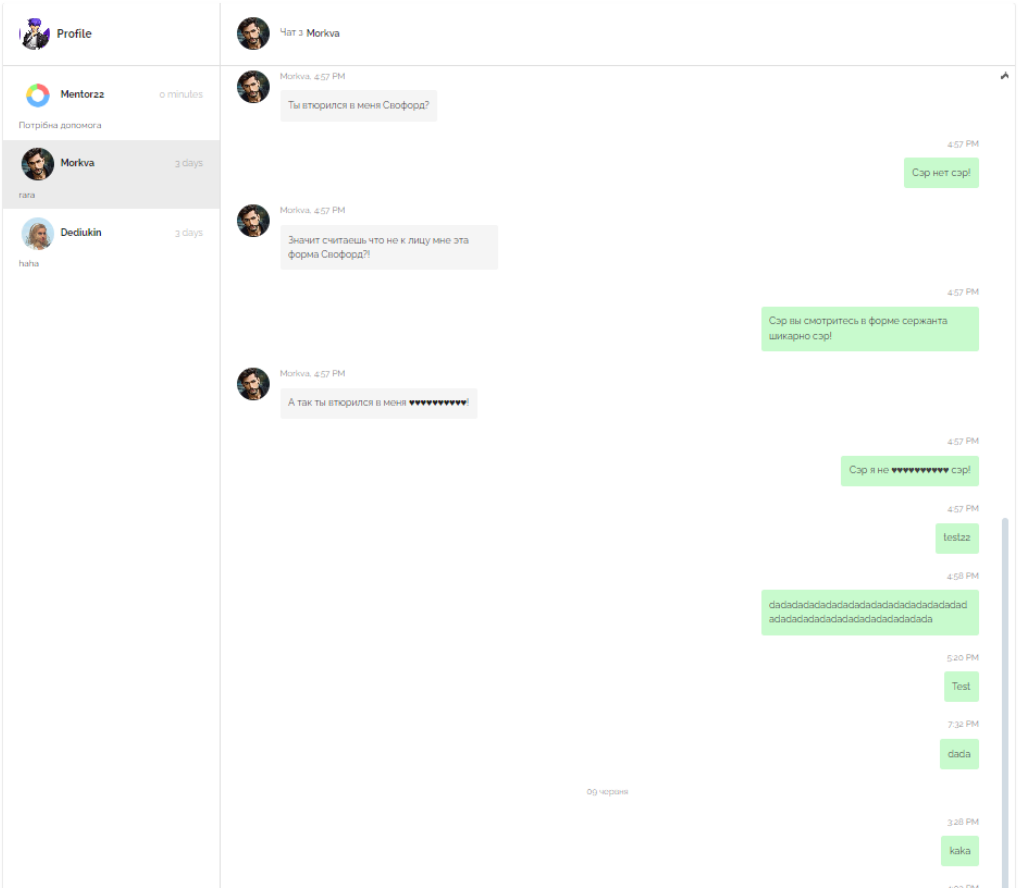


Сторінка профілю ментора



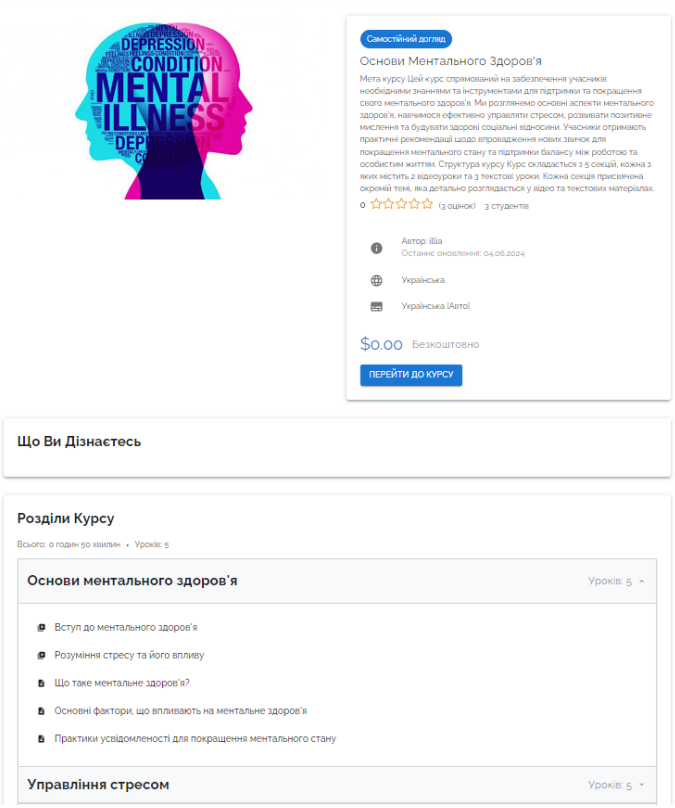
Сторінка чату з ментором.

Сторінка всіх доступних чатів

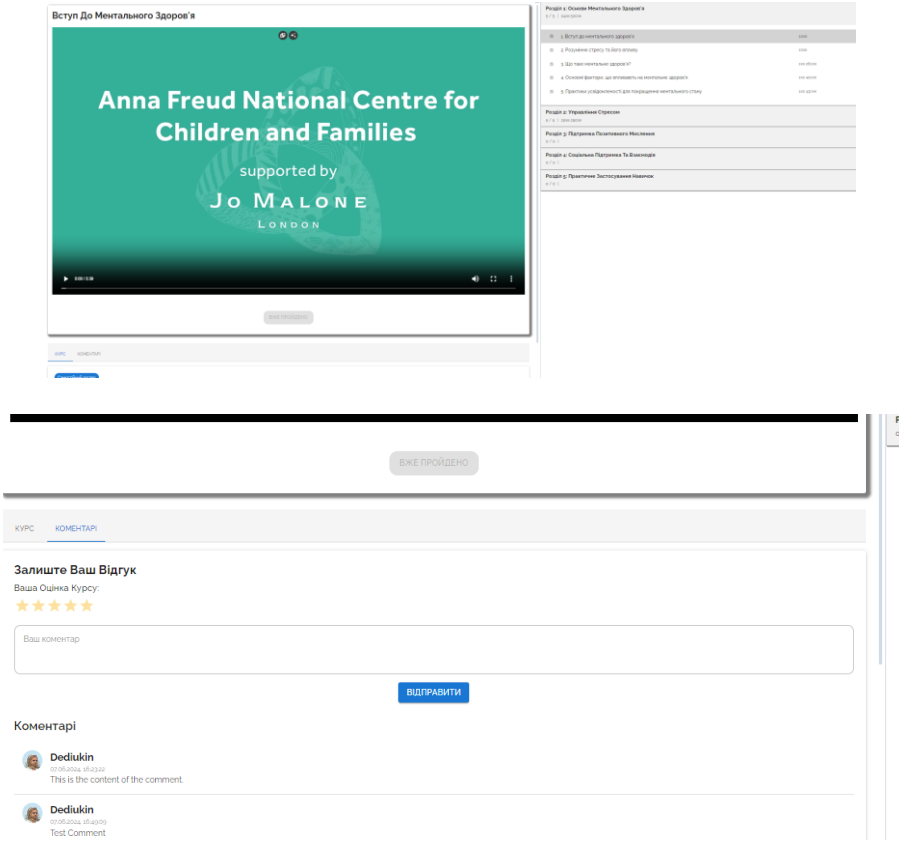


Сторінка бронювання запису до ментора.

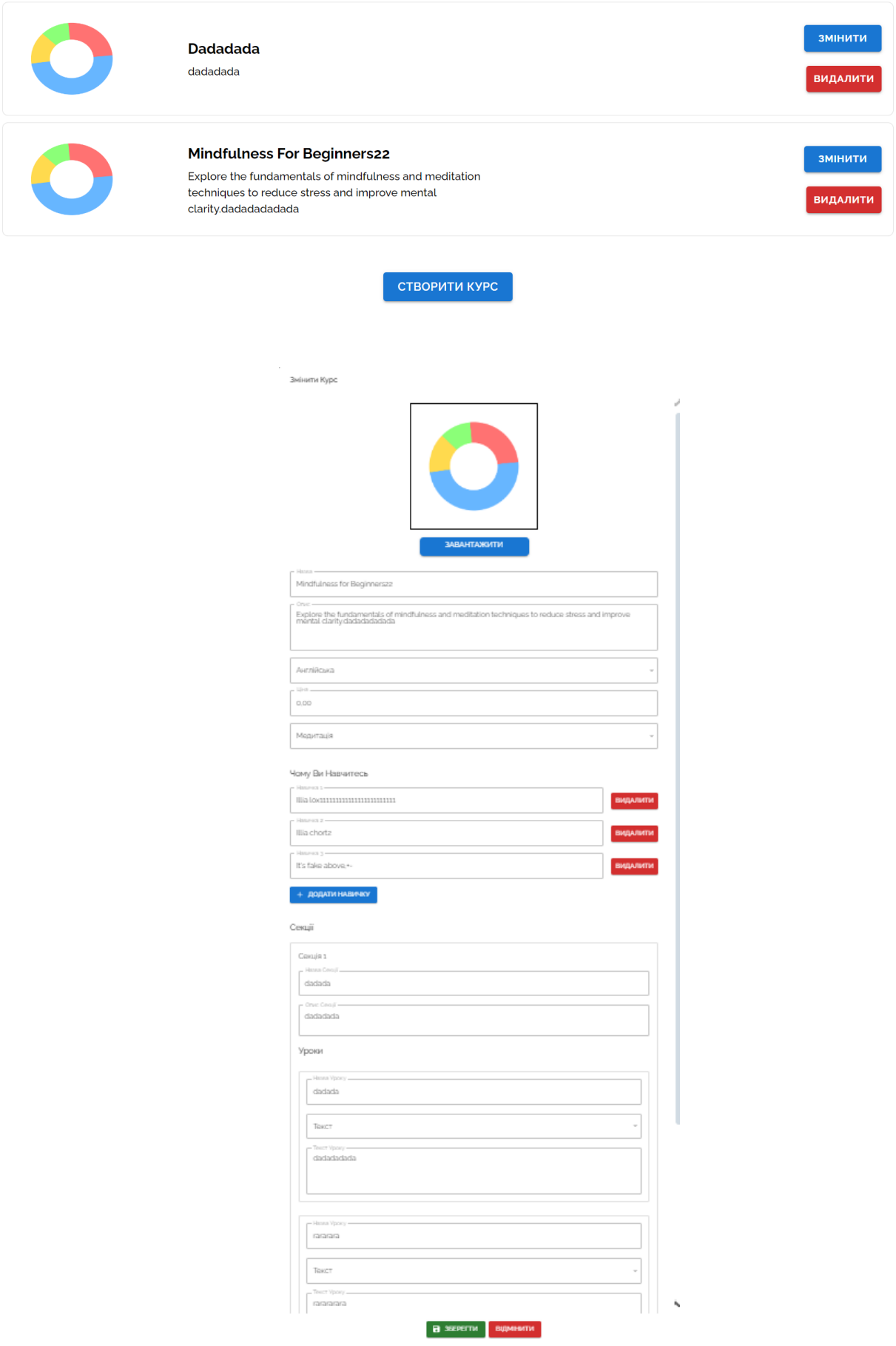
Сторінка обраного курсу.



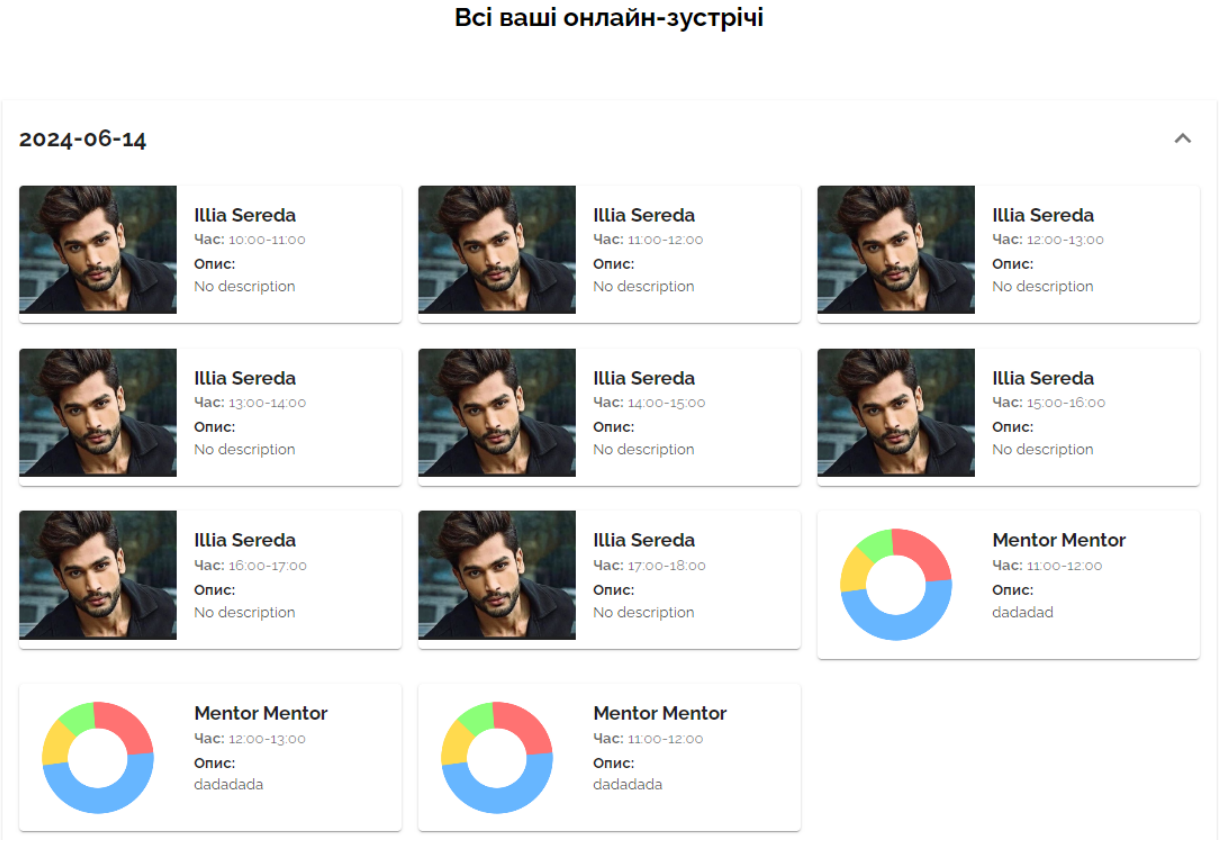
Сторінка проходження курсу.



Сторінка створення курсу.


Dadadada
dadadada
ЗМІНИТИ
ВИДАЛИТИ
Mindfulness For Beginners22
Explore the fundamentals of mindfulness and meditation techniques to reduce stress and improve mental clarity.dadadadadada
ЗМІНИТИ
ВИДАЛИТИ
СТВОРИТИ КУРС
Змінити Курс
ЗАВАНТАЖИТИ
Mindfulness for Beginners22
Explore the fundamentals of mindfulness and meditation techniques to reduce stress and improve mental clarity.dadadadadada
Англійська
0.00
Медитація
Чому Ви Навчитесь
lllia lox
ВИДАЛИТИ
lllia chortz
ВИДАЛИТИ
It's fake above.+-
ВИДАЛИТИ
ДОДАТИ НАВИЧКУ
Секції
Секція 1
dadada
dadadada
Уроки
dadada
Текст
dadadadada
rararara
Текст
rararara
ЗБЕРЕГТИ
ВІДМІНИТИ

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

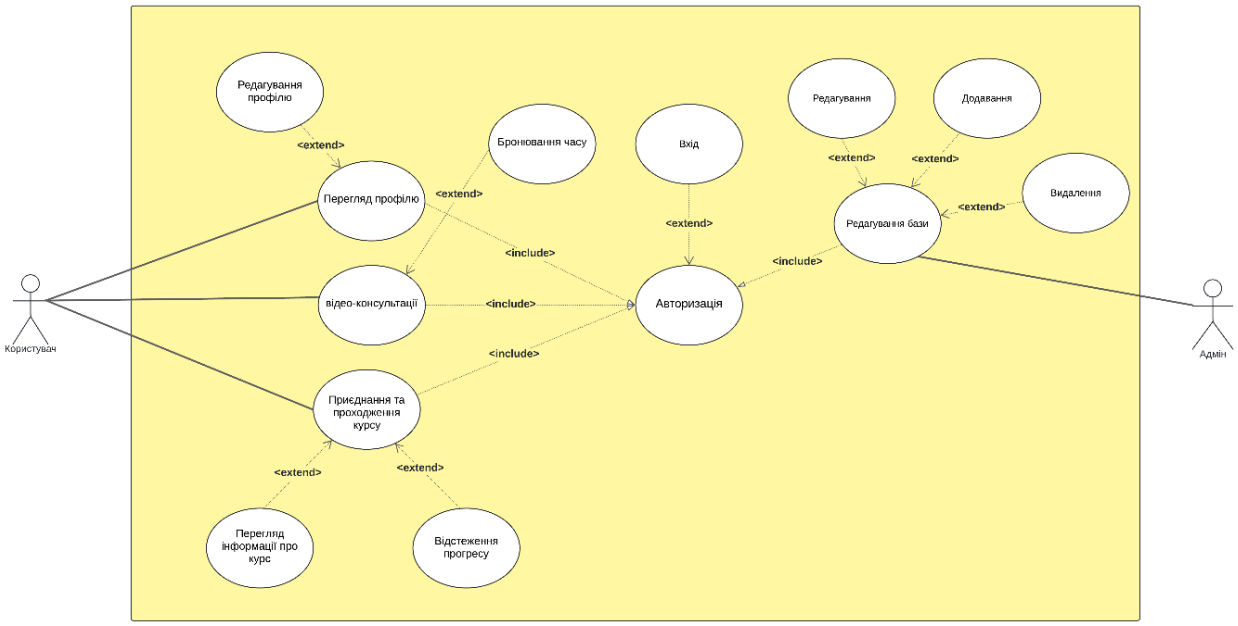


Рисунок 3.1 – Use Case діаграма програмного модулю.

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.

Модель коментарів CoursesComment:

- id;
- user_id;
- message;
- created.

Модель категорій CategoryCategory:

- id;
- name;
- parent.

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 логіруватися.

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

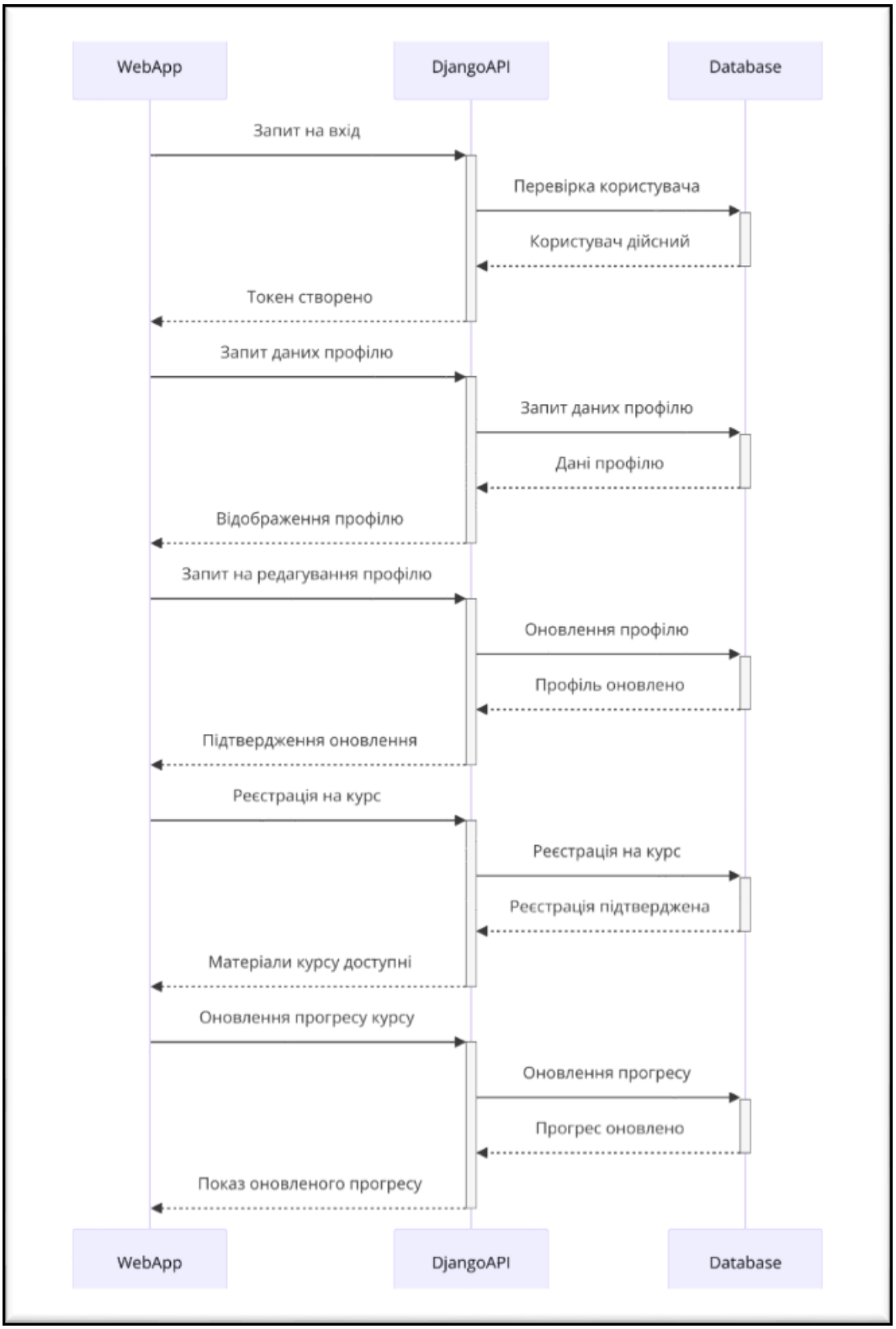


Рисунок 4.1 – Діаграма потоків даних програмного модулю.

4.2 Діаграма переходів станів

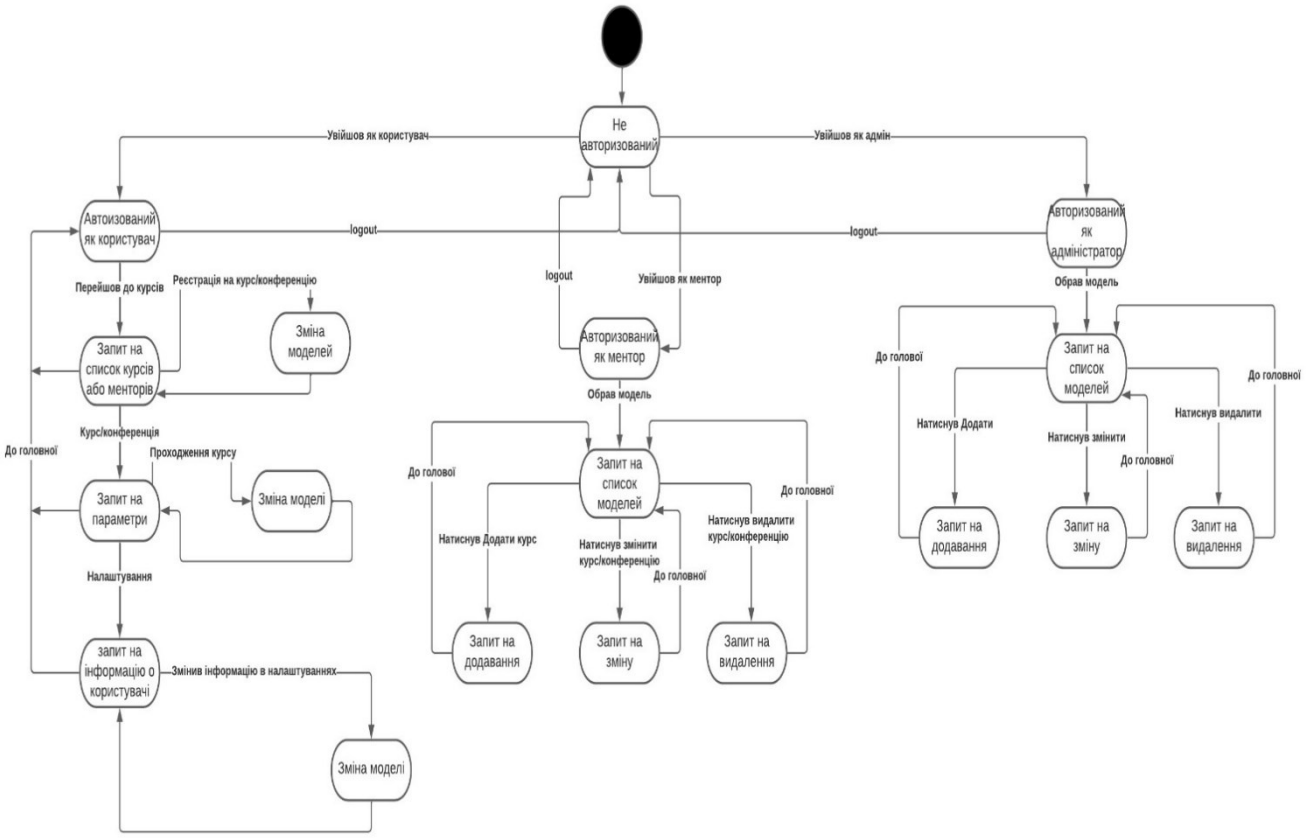


Рисунок 4.2 – Діаграма потоків даних програмного модулю.

4.3 Діаграма потоків даних

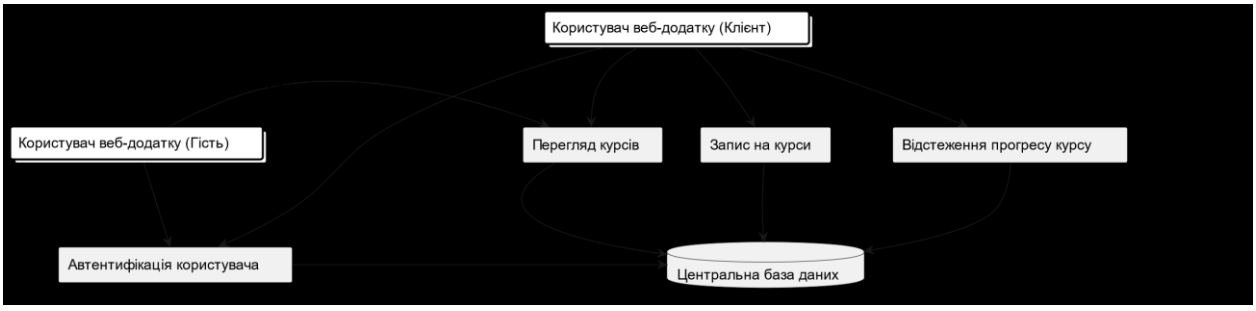


Рисунок 4.3 – Діаграма потоків даних програмного модулю.

ДОДАТОК Г

Наведемо код для додавання курсів в корзину користувача:

```
import {
  ADD_TO_CART,
  REMOVE_CART_ITEM,
  CLEAR_CART,
  GET_CART_TOTAL,
  BUY_COURSES,
} from "../actions";
import { getUser } from "../helpers/locals";
import { CourseType } from "../types/courses.types";
import { API_LINK } from "../utils/constants";

const cart_reducer = (state: any, action: any) => {
  const userId = getUser()?.id;

  if (action.type === ADD_TO_CART) {
    const tempArr = state.cart[userId].filter(
      (item: CourseType) => item.id === action.payload.id
    );
    if (tempArr.length < 1) {
      return {
        ...state,
        cart: {
          ...state.cart,
          [userId]: [...state.cart[userId], action.payload],
        },
      };
    }
    return {
      ...state,
    };
```

```
}

if (action.type === REMOVE_CART_ITEM) {
  const tempCart = state.cart[userId].filter(
    (item: CourseType) => item.id !== action.payload
  );
  return {
    ...state,
    cart: {
      ...state.cart,
      [userId]: tempCart,
    },
  };
}

if (action.type === GET_CART_TOTAL) {
  const total_amount = state.cart[userId].reduce(
    (total: number, cartItem: CourseType) => {
      total += Number(cartItem.price);
      return total;
    },
    0
  );
  return {
    ...state,
    total_items: state.cart[userId].length,
    total_amount,
  };
}

if (action.type === BUY_COURSES) {
  fetch(`${API_LINK}/api/purchase`, {
```

```
      method: "POST",
      body: state.cart[userId],
    });
    return {
      ...state,
      cart: {
        ...state.cart,
        [userId]: [],
      },
    };
  }

  if (action.type === CLEAR_CART) {
    return {
      ...state,
      cart: {
        ...state.cart,
        [userId]: [],
      },
    };
  }

  throw new Error(`No matching "${action.type}" - action type`);
};

export default cart_reducer;
```