Вінницький національний технічний університет)
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))
Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))
(повна назва

кафедри (предметної, циклової комісії)

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методів і програмних засобів реалізації веб-порталу психологічної підтримки»

Виконала: студентка II курсу
групи 2ПІ-23м спеціальності
121 – Інженерія програмного забезпечення
Озерова Катерина Олександрівна

Керівник: к.т.н., доц. каф. ПЗ Войтко В.В.
«06» грудня 2024 р.
Опонент: к.т.н., доц. каф. ОТ Колесник І.С.
«06» грудня 2024 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«06» грудня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
к.т.н., професор Романюк О. Н.
« 18 » вересня 2024 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Озеровій Катерині Олександрівні

1. Тема роботи – Розробка методів і програмних засобів реалізації веб-порталу психологічної підтримки.

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від « 17 » вересня 2024 р. № 310.

2. Строк подання студентом роботи 3 грудня 2024р.

3. Вихідні дані до роботи: методи побудови архітектури – багаторівнева архітектура з використанням REST API; методи обчислення рейтингу спеціалістів – метод вагових коефіцієнтів та нормалізації; алгоритми інтеграції Telegram-боту для надсилання динамічних сповіщень; система планування подій – Node Schedule; середовище розробки – PHPStorm; мова розробки – JavaScript (Node.js, React); управління реляційними базами даних – SQLite.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку веб-порталів психологічної підтримки та постановка задач дослідження; розробка методу та моделей веб-порталу; розробка програмних модулів реалізації веб-порталу психологічної підтримки; тестування програми;

економічна частина; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: інтерфейс аналогів веб-порталу психологічної підтримки; блок-схеми алгоритмів роботи веб-порталу; моде роботи системи; тестування веб-порталу психологічної підтримки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Войтко В.В., к.т.н., доц. кафедри ПЗ	19.09.2024	02.12.2024
5	Причепа І.В., к.е.н., доцент кафедри ЕПВМ	22.11.2024	30.11.2024

7. Дата видачі завдання 19 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку веб-порталів психологічної підтримки	20.09.2024 07.10.2024	виконано
2	Розробка методу та моделей веб-порталу	08.10.2024 21.10.2024	виконано
3	Розробка програмних модулів реалізації веб-порталу психологічної підтримки	22.10.2024 16.11.2024	виконано
4	Тестування програми	17.11.2024 23.11.2024	виконано
5.	Економічна частина	24.11.2024 30.11.2024	виконано

Студент (підпис) **Озерова К. О.** (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи (підпис) **Войтко В.В.** (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.912.032.26

Озерова К.О. Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 142 с.

На укр. мові. Бібліогр.: назв: 40; рисунки: 65; таблиць: 11.

У магістерській кваліфікаційній роботі розроблено методи і програмні засоби реалізації вебпорталу психологічної підтримки. Було проведено аналіз існуючих рішень, що дозволило визначити їхні переваги та недоліки. У роботі обґрунтовано доцільність створення порталу, визначено його основні функціональні можливості, а також специфічні особливості, що відрізняють його від аналогів.

Розроблено метод формування рейтингу спеціалістів, що включає вагові коефіцієнти для різних критеріїв та нормалізацію оцінок з урахуванням кількості відгуків. Запропоновані методи забезпечують адаптивність системи під потреби різних груп користувачів та гарантують надійність і об'єктивність оцінок.

У роботі реалізовано програмні модулі управління даними користувачів і спеціалістів, календар для відображення доступних годин, систему оцінювання та відгуків, а також функціонал підписки на оновлення. Окремим модулем розроблено інтеграцію з Telegram-ботом, що дозволяє користувачам підписуватися на сповіщення про нові доступні дати консультацій та отримувати нагадування, що значно підвищує зручність користування платформою.

Ключові слова: вебпортал, психологічна підтримка, рейтинг спеціалістів, вагові коефіцієнти, нормалізація, React, Node.js, Telegram-бот.

ABSTRACT

Ozerova K.O. Development of methods and software tools for implementing a web portal for psychological support. Master's qualification thesis in specialty 121 – Software Engineering, Educational Program – Software Engineering. Vinnytsia: VNTU, 2024. 142 pages.

In Ukrainian. Bibliography: 40 titles; Figures: 65; Tables: 11.

The master's qualification thesis focuses on the development of methods and software tools for implementing a web portal for psychological support. An analysis of existing solutions was conducted to identify their advantages and disadvantages. The study substantiates the need for the portal's creation, defines its core functionalities, and outlines the specific features distinguishing it from similar platforms.

A method for forming a specialist rating system was developed, which incorporates weighted coefficients for various evaluation criteria and normalizes scores considering the number of reviews. These proposed methods ensure the system's adaptability to the needs of different user groups and guarantee the reliability and objectivity of evaluations.

The thesis implements software modules for managing user and specialist data, a calendar for displaying available time slots, a rating and feedback system, as well as a subscription functionality. A separate module was developed to integrate with a Telegram bot, allowing users to subscribe to notifications about newly available consultation dates and receive reminders, significantly enhancing the platform's usability.

Keywords: web portal, psychological support, specialist rating, weighted coefficients, normalization, React, Node.js, Telegram bot.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Психологічне здоров'я стає все більш актуальною темою в сучасному суспільстві, особливо в умовах швидкого темпу життя, економічних та соціальних потрясінь. Це вимагає від суспільства та фахівців переходу на новий рівень надання психологічної підтримки.

Зростання стресу, пов'язане з економічними і соціальними проблемами, такими як пандемії, війни та інші кризові ситуації, які призводять до збільшення випадків тривожних розладів, депресії та інших психічних проблем. Традиційні методи підтримки часто є недостатніми, оскільки вони не можуть забезпечити доступність і своєчасність допомоги у необхідному обсязі [1].

Така ситуація вимагає розробки сучасних підходів до надання психологічної підтримки, зокрема через використання інноваційних цифрових технологій. Використання вебтехнологій у сфері психічного здоров'я набуло популярності у багатьох країнах, і вже доведено, що інтерактивні онлайн-інструменти можуть підвищувати ефективність психологічної допомоги. За даними міжнародних досліджень, понад 70% респондентів зазначили, що використовували онлайн-ресурси для отримання психологічної підтримки або самодопомоги під час кризових ситуацій.

І звичайно використання сучасних вебтехнологій дозволяє значно розширити доступ до психологічних послуг. Розробка порталу з функціями онлайн-консультацій, тестів для оцінки психічного стану, курсів самодопомоги та персоналізованих рекомендацій підвищує якість підтримки, а більше того, робить її доступнішою для широкого кола користувачів.

Повний перегляд традиційних методів надання підтримки дозволяє інтегрувати новітні технології, що забезпечують можливість створення інтерактивних, адаптивних та персоналізованих сервісів. Індустрія надання психологічної підтримки більше не може ігнорувати виклики часу і змушена адаптуватися до сучасних умов [2].

Важливими аспектами для користувачів є:

– можливість проходження тестування для оцінки психічного стану;

– отримання персональних рекомендацій щодо психологічних методів самодопомоги;

– онлайн-консультації зі спеціалістами;

– збереження конфіденційності та безпечного доступу до даних.

Існуючі вебсистеми часто мають обмежену функціональність, не забезпечуючи достатньої інтерактивності та персоналізації для користувачів. Це створює проблему, коли навіть ті, хто звертається за допомогою, не отримують комплексних інструментів для оцінки свого стану, вибору відповідного спеціаліста та отримання ефективних рекомендацій.

Отже, розробка сучасного інтерактивного вебпорталу з розширеною функціональністю, включаючи рейтингову систему спеціалістів, а також інструменти для самодопомоги, є важливим кроком у напрямку підвищення якості та ефективності психологічної підтримки.

Така платформа дозволить користувачам отримувати повноцінний спектр послуг, що відповідають їхнім індивідуальним потребам та очікуванням. А найголовніше збереже конфіденційність та безпеку доступу до даних.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей для надання психологічної підтримки за рахунок розробки методів і програмних засобів для інтерактивного вебпорталу, що забезпечить персоналізовану взаємодію з користувачами, підвищить точність рекомендацій та оптимізує процеси вибору спеціалістів.

Основними задачами роботи є:

– визначити засоби реалізації вебпорталу для психологічної підтримки;

– розробити методи визначення загального стану пацієнта та

персоналізованого підбору спеціалістів на основі результатів;

– розробити методи розрахунку рейтингу спеціалістів;

– розробити програмні модулі для інтеграції телеграм-боту з функціями нагадувань та динамічних сповіщень;

– провести тестування вебпорталу для перевірки його функціональності, надійності та відповідності заданим вимогам.

Об’єктом дослідження є процес розробки вебпорталу психологічної підтримки.

Предметом дослідження є методи та засоби розробки вебпорталу психологічної підтримки.

Методи дослідження. У процесі досліджень використовувались методи дослідження:

– метод побудови клієнт-серверної архітектури – REST API для взаємодії між клієнтською та серверною частинами вебпорталу;

– метод структурної організації компонентів системи на основі архітектурного патерну Layered Architecture для оптимізації роботи з даними;

– методи розробки алгоритмів для створення модулів тестування та персоналізованого підбору спеціалістів;

– метод вагового оцінювання для створення рейтингової системи спеціалістів, що враховує різні аспекти діяльності спеціалістів через застосування вагових коефіцієнтів до окремих критеріїв оцінки;

– метод нормалізації рейтингових оцінок для забезпечення об’єктивності порівняння спеціалістів із різною кількістю відгуків;

– метод інтерактивної візуалізації для побудови календаря завантаженості спеціалістів, що дозволяє відображати робочий графік, зайняті та вільні часові проміжки для консультацій;

– метод інтеграції API для реалізації телеграм-боту з функціями нагадувань, сповіщень та керування записами на консультації;

– метод планування подій на основі Node Schedule для автоматичного оновлення статусів записів та надсилання сповіщень користувачам у встановлені

часові проміжки;

– методи тестування програмного забезпечення для перевірки функціональності, продуктивності та надійності вебпорталу.

Наукова новизна отриманих результатів:

1. Подальшого розвитку отримав метод побудови рейтингової системи спеціалістів, який, на відміну від існуючих, використовує вагові коефіцієнти для кожного критерію оцінки, а також застосовує нормалізацію рейтингових оцінок для стабільності рейтингу, що забезпечує адаптивність до потреб користувачів, враховує важливість окремих аспектів, таких як ефективність консультування та задоволеність клієнтів, дозволяє об'єктивно порівнювати спеціалістів із різною кількістю відгуків, що розширює функціональність застосунку та підвищує надійність роботи системи.

2. Подальшого розвитку отримав метод інтеграції телеграм-боту з функціями динамічних сповіщень і аналітичних звітів, який, на відміну від існуючих, базується на системі планування подій Node Schedule, що забезпечує не лише динамічні повідомлення у відповідь на різні зміни, але й регулярні нагадування, а також реалізує автоматизоване формування й відправку персоналізованих звітів з історією взаємодій користувача та рекомендаціями для подальших дій, що підвищує гнучкість планування, інформованість користувачів та якість взаємодії з платформою.

Практичне значення одержаних результатів. Практична цінність результатів магістерської кваліфікаційної роботи полягає у можливості використання розробленого вебпорталу для надання ефективної та доступної психологічної підтримки. Система може застосовуватися для оптимізації роботи спеціалістів, спрощення процесу планування консультацій та забезпечення користувачів необхідними ресурсами для підтримки психічного здоров’я.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У науковій роботі [3], опублікованій у співавторстві, автору належать такі результати: аналіз функціоналу вебпорталу та розробка засобів його реалізації.

Апробація матеріалів магістерської кваліфікаційної роботи. Результати магістерської кваліфікаційної роботи обговорювалися на міжнародній науково-практичній конференції «Інформаційні технології та автоматизація – 2024».

Публікації. Результати дослідження опубліковані в тезах доповіді міжнародної науково-практичної конференції «Інформаційні технології та автоматизація – 2024» [3].

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел, що містить 40 найменувань, 4 додатки. Робота містить 65 ілюстрації, 11 таблиць.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБПОРТАЛІВ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

1.1 Аналіз стану розвитку вебпорталів психологічної підтримки

Психологічна підтримка через інтернет набуває все більшого значення, особливо в умовах зростання стресових факторів і психологічних проблем серед населення. Сучасний ритм життя, соціальна ізоляція, викликана пандемією COVID-19, та інші кризи суттєво вплинули на психічне здоров'я багатьох людей. У такій ситуації доступ до психологічної допомоги стає критично важливим, але традиційні форми взаємодії, такі як офлайн-консультації, не завжди є доступними через географічні або економічні обмеження.

Вебпортали психологічної підтримки дозволяють подолати ці бар'єри, забезпечуючи можливість отримувати допомогу дистанційно, у зручний час та місце. Це особливо важливо для людей, які відчувають труднощі з особистими візитами до психолога через стигматизацію, тривожність або фізичні обмеження. Важливим аспектом є доступність підтримки 24/7, що дозволяє надавати екстрену допомогу у кризових ситуаціях, коли важливо миттєво зв'язатися з фахівцем [4].

На ринку зараз існує декілька рішень, які надають психологічну підтримку через інтернет. Це можуть бути платформи для онлайн-консультацій з психологами, ресурси з інтерактивними вправами та техніками самодопомоги, а також форуми і спільноти підтримки, де користувачі можуть обмінюватися досвідом і отримувати допомогу від однолітків. Деякі вебпортали пропонують комплексні рішення, поєднуючи можливості індивідуальних консультацій з доступом до навчальних матеріалів та групової підтримки.

Популярність таких сервісів зумовлена їхньою зручністю та ефективністю. Вебпортали психологічної підтримки надають можливість користувачам зберігати анонімність, що знижує бар'єри для звернення за допомогою. Крім того, такі платформи дозволяють людям знайти психолога з урахуванням спеціалізації та мовних уподобань, що сприяє більш персоналізованому підходу

до вирішення психологічних проблем.

Актуальність розробки нових вебпорталів психологічної підтримки полягає в необхідності забезпечення більш доступної та різноманітної підтримки, особливо для тих, хто стикається з обмеженим доступом до традиційних форм допомоги. Такі портали допомагають знизити рівень тривожності, депресії та інших психологічних проблем серед населення, надаючи не лише індивідуальні консультації, але й доступ до інформаційних ресурсів, що сприяють розвитку навичок самодопомоги [5].

Серед переваг вебпорталів психологічної підтримки виділимо:

1. Доступність та зручність. Вебпортали забезпечують доступ до психологічної допомоги 24/7, що особливо важливо в кризових ситуаціях, коли допомога потрібна негайно. Користувачі можуть отримувати консультації з будь-якої точки світу, заощаджуючи час та ресурси на поїздки до фахівця.

2. Анонімність. Багато користувачів відчувають страх чи сором щодо звернення до психолога в традиційному форматі. Вебпортали дозволяють зберегти анонімність, що допомагає знизити психологічний бар'єр та збільшує готовність до отримання допомоги.

3. Різноманіття форм допомоги. Портали можуть пропонувати індивідуальні консультації, групові сеанси, форуми підтримки, а також доступ до ресурсів із самодопомоги. Це дозволяє користувачам обирати ту форму взаємодії, яка найбільше відповідає їхнім потребам та зручності.

4. Персоналізований підхід. Користувачі мають можливість обирати фахівця за спеціалізацією, мовою спілкування та іншими критеріями, що підвищує якість допомоги та задоволення від співпраці. Багато платформ використовують попереднє анкетування, щоб підібрати найбільш підходящого спеціаліста.

Головними недоліками використання вебпорталів психологічної підтримки є:

1. Брак особистого контакту. Незважаючи на розвиток технологій, онлайн-консультації не завжди можуть замінити очний контакт, який для

багатьох клієнтів є важливим аспектом терапії. Деякі люди відчувають недостатню емоційну взаємодію в онлайн-форматі, що може знижувати ефективність консультацій.

2. Залежність від технологій. Для використання таких платформ необхідний доступ до стабільного інтернету та базові навички користування цифровими технологіями, що може бути проблемою для людей старшого віку або тих, хто проживає у віддалених районах.

3. Питання безпеки даних. Захист конфіденційності та безпека персональних даних є критично важливими для таких платформ. Недостатня увага до цього аспекту може призвести до витоку конфіденційної інформації або втрати довіри з боку користувачів. Це вимагає впровадження високих стандартів шифрування та зберігання даних.

4. Обмежені можливості для кризових втручань. У ситуаціях, що вимагають негайної фізичної допомоги (наприклад, суїцидальна криза), онлайн-платформи не завжди можуть забезпечити достатню підтримку. У таких випадках фізична присутність фахівця може бути більш ефективною та необхідною.

Таким чином, вебпортали психологічної підтримки є важливим інструментом у забезпеченні доступу до психологічної допомоги та підтримки, особливо для людей, які стикаються з бар'єрами у доступі до традиційних форм терапії. Вони мають значний потенціал для зменшення рівня тривожності та інших психологічних проблем, надаючи широкий спектр можливостей для індивідуального підходу та самодопомоги. Однак для їх ефективного функціонування важливо забезпечити належний рівень безпеки та розуміти обмеження онлайн-формату.

1.2 Порівняльний аналіз аналогів

Для визначення ефективного підходу до розробки вебпорталу психологічної підтримки необхідно провести ґрунтовний порівняльний аналіз існуючих аналогів, що дозволить оцінити їхні функціональні можливості,

технічні рішення та досвід користувачів.

Такий аналіз допоможе виявити сильні та слабкі сторони конкурентів, що, у свою чергу, дозволить визначити оптимальні шляхи для створення порталу, який відповідатиме потребам користувачів та буде конкурентоспроможним на ринку.

У рамках цього аналізу розглянемо чотири популярні вебплатформи для надання психологічної підтримки: BetterHelp, Talkspace, 7 Cups та Calmerry.

BetterHelp одна з найбільших і найвідоміших онлайн-платформ для психологічної підтримки, яка з'явилася на ринку у 2013 році. Її метою є забезпечення користувачів доступом до кваліфікованих психологів і терапевтів через інтернет, що робить консультації більш доступними для людей з різних регіонів та умов. Платформа обслуговує користувачів із понад 150 країн і надає консультації через відео, аудіо та текстові чати [6].

Платформа пропонує індивідуальні консультації через відео, аудіо та текстові чати. Платформа дозволяє користувачам заповнити анкету, на основі якої підбирається відповідний фахівець, але також є можливість обрати спеціаліста самостійно або отримати рекомендацію на основі анкети.

Інтерфейс вебсистеми психологічної підтримки BetterHelp зображено на рисунку 1.1.

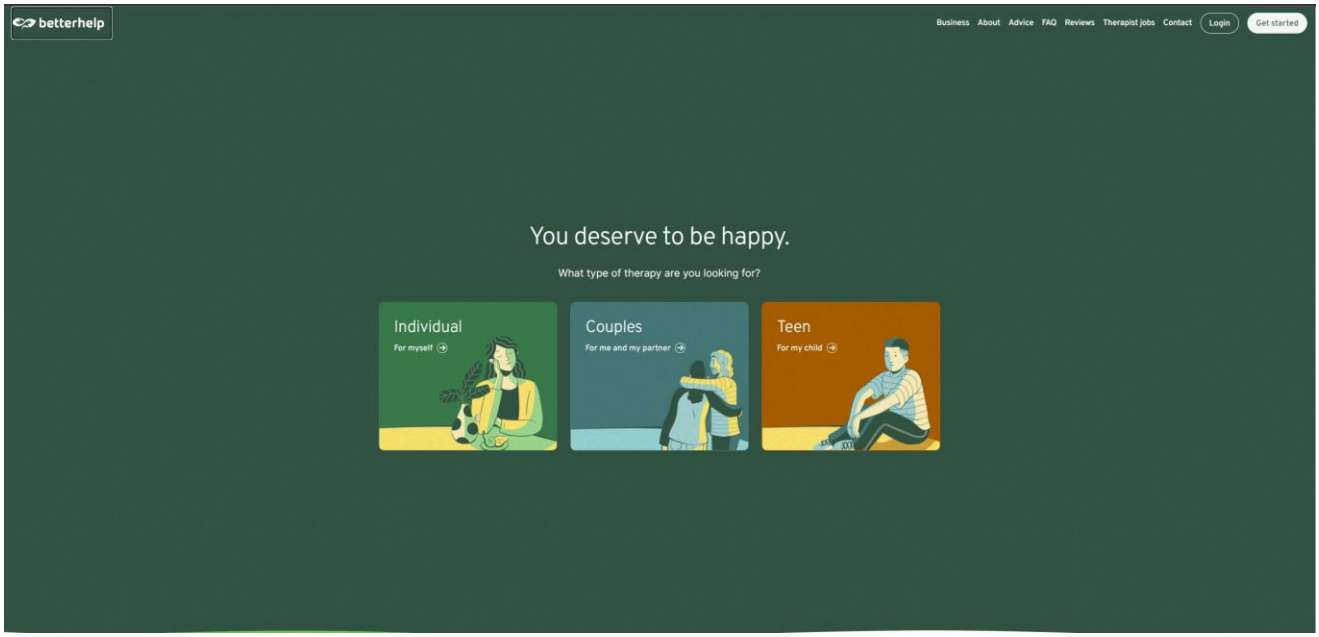

Рисунок 1.1 – Портал психологічної підтримки BetterHelp

Перевагами порталу BetterHelp є велика кількість фахівців, можливість вибору формату консультації, зручний мобільний додаток.

Недоліками порталу BetterHelp є відносно висока вартість послуг, відсутність безкоштовних ресурсів із самодопомоги, консультації доступні лише за попереднім записом.

Talkspace популярна онлайн-платформа для надання психологічної підтримки, заснована у 2012 році. Метою платформи є надання доступу до ліцензованих терапевтів через зручний формат текстових, відео- та аудіоконсультацій.

Talkspace пропонує користувачам гнучкі можливості для отримання професійної допомоги, що дозволяє людям звертатися за підтримкою у зручний для них час, незалежно від місця їхнього перебування [7].

Платформа застосовую консультації з ліцензованими терапевтами через текстові та відеочати. А також дозволяє користувачам безперервно спілкуватися з терапевтом у текстовому форматі, а також отримувати відеоконсультації за попереднім записом. Є можливість обирати фахівця, а також змінювати його у випадку незадоволення (рисунок 1.2).

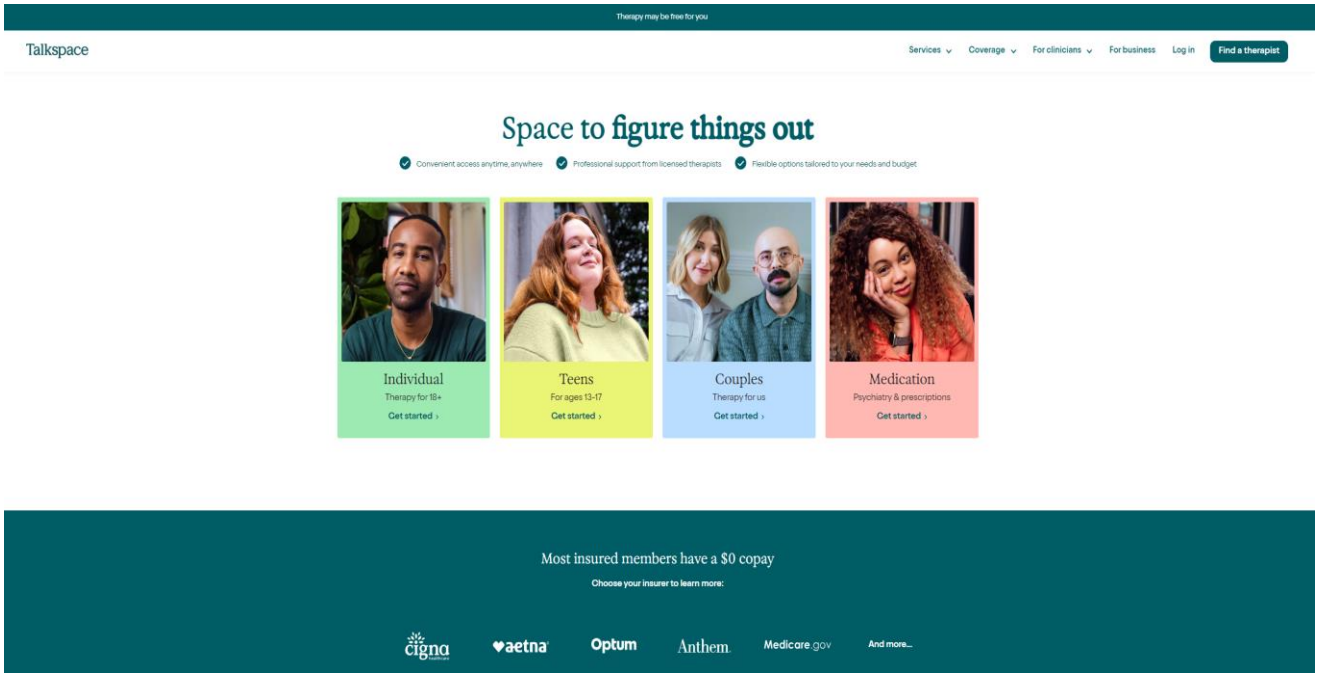

Рисунок 1.2 – Портал психологічної підтримки Talkspace

Перевагами порталу Talkspace є можливість безперервного текстового спілкування з терапевтом; широкий вибір спеціалістів, висока конфіденційність даних.

Недоліками порталу Talkspace деякі користувачі відзначають тривалу відповідь від терапевта в текстовому форматі, високу вартість підписки, обмежений доступ до додаткових ресурсів без додаткової оплати.

7 Cups – онлайн-платформа, що поєднує безкоштовну підтримку від навчених волонтерів та платні консультації з ліцензованими психологами. Платформа була заснована у 2013 році з метою забезпечити доступну емоційну підтримку для всіх, хто цього потребує, особливо у випадках, коли відсутня можливість звернутися за професійною допомогою через фінансові або географічні обмеження [8].

7 Cups відрізняється своїм акцентом на спільноту, що сприяє взаємній підтримці серед користувачів, а також має велику базу ресурсів із самодопомоги, доступну для всіх зареєстрованих користувачів. Включає велику базу ресурсів для самодопомоги (рисунок 1.3).

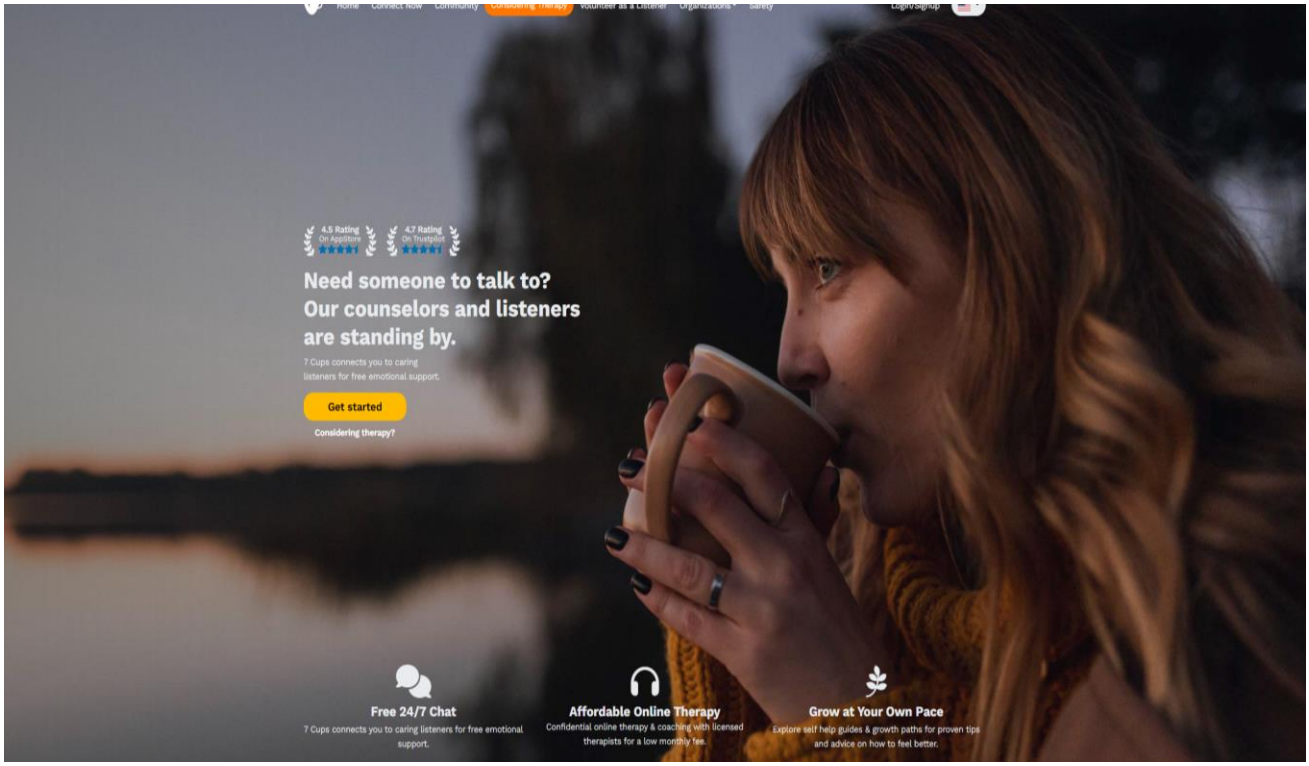

Рисунок 1.3 – Портал психологічної підтримки 7 Cups

Перевагами порталу є безкоштовний доступ до волонтерської підтримки; багато ресурсів для самодопомоги; підтримка в режимі 24/7.

З недоліків можна виділити те, що якість підтримки від волонтерів може варіюватися; ліцензовані консультації доступні лише за плату; деякі користувачі відзначають обмежену глибину допомоги в безкоштовному форматі.

Calmerry – це онлайн-платформа для надання психологічної підтримки, створена з акцентом на доступності та гнучкості консультацій [9]. Заснована у 2020 році, платформа швидко здобула популярність завдяки своєму індивідуалізованому підходу до кожного користувача та зручності використання.

Calmerry пропонує відео- та текстові консультації з ліцензованими психологами, орієнтуючись на забезпечення якісної допомоги у зручному для клієнта форматі. Платформа є відмінним варіантом для людей, які шукають доступну та якісну психологічну підтримку через інтернет. Платформа відрізняється своїм індивідуалізованим підходом, зручністю використання та гнучкими тарифними планами, що робить її більш доступною для широкого кола користувачів (рисунок 1.4).

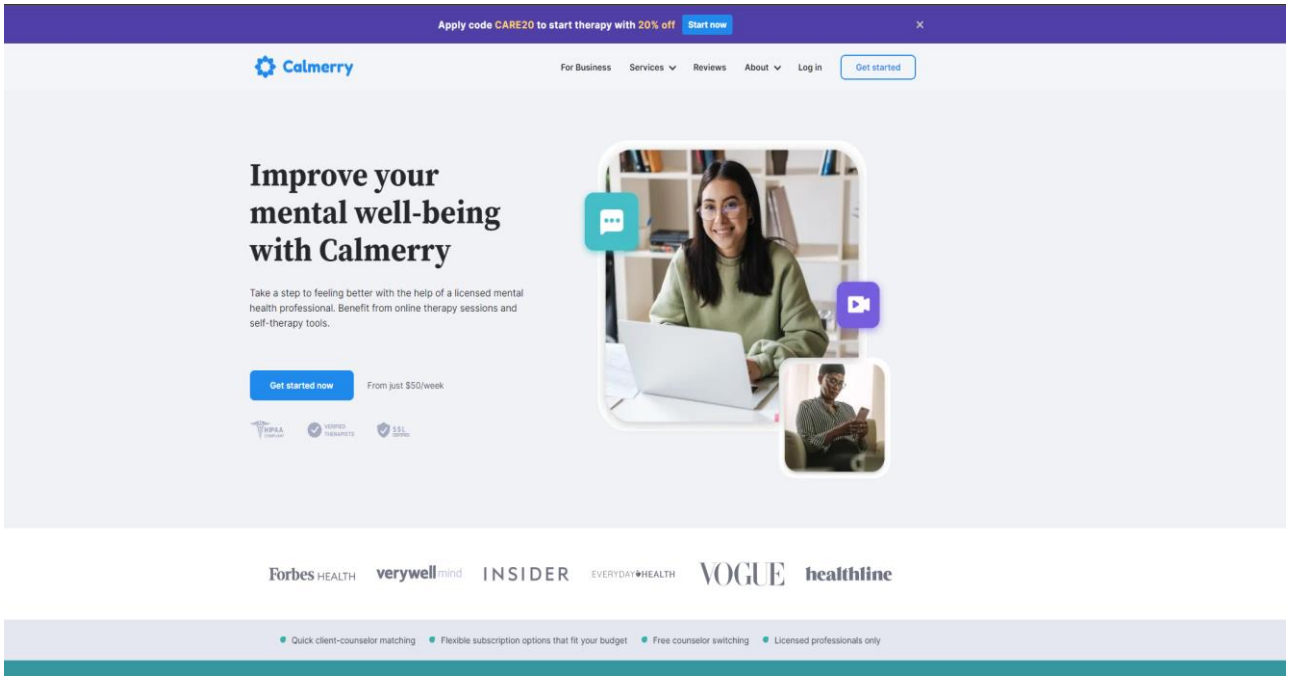

Рисунок 1.4 – Портал психологічної підтримки Calmerry

Однак відсутність безкоштовних ресурсів та обмежені можливості для групової підтримки можуть стати перешкодою для тих, хто шукає додаткову підтримку поза межами індивідуальних сесій.

Перевагами порталу Calmerry є гнучкі тарифи на послуги, висока якість консультацій та можливість швидко змінити психолога.

Недоліками є відсутність безкоштовного контенту, обмежені можливості для групової підтримки і платний доступ до повного спектра функцій.

Розглянемо порівняльний аналіз аналогів разом із запропонованим нами вебпорталом у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз вебпорталів психологічної підтримки

Критерій	BetterHelp	Talkspace	7 Cups	Calmerry	Soul Talk
Різні формати консультацій	+	+	+	+	+
Можливість вибору психолога	+	+	+	+	+
Наявність безкоштовної підтримки	-	-	+	-	+
Додаткові ресурси	-	-	+	-	+
Конфіденційність даних	+	+	-	+	+
Вартість послуг	-	-	-	+	+
Доступність 24/7	-	-	+	-	+
Швидкість зміни психолога	+	+	+	+	+
Підсумковий результат	4	4	6	5	8

Результати порівняльного аналізу показали, що всі розглянуті платформи пропонують індивідуальні консультації з ліцензованими психологами, забезпечують високий рівень конфіденційності та зручність у використанні. Однак, платформи BetterHelp та Talkspace мають вищу вартість, що може обмежувати доступ до їхніх послуг для деяких користувачів. Натомість 7 Cups пропонує безкоштовну підтримку від волонтерів, що робить її більш доступною,

але якість такої допомоги може бути непостійною. Calmerry займає проміжне положення, забезпечуючи гнучкі тарифи та швидку можливість змінити психолога, проте не надає безкоштовного контенту для самодопомоги.

На основі аналізу можна зробити висновок, що жодна з платформ не є ідеальною для всіх категорій користувачів. Розробка нового вебпорталу має враховувати досвід існуючих рішень та поєднувати їхні сильні сторони: забезпечення доступної підтримки, широкі можливості для самодопомоги та високий рівень конфіденційності. Це дозволить створити конкурентоспроможний продукт, орієнтований на потреби різних категорій користувачів.

1.3 Аналіз методів розв’язання задачі

Існує кілька основних підходів до розробки вебпорталів психологічної підтримки, які включають як серверну, так і клієнтську частини. Для реалізації вебпорталу психологічної підтримки було обрано клієнт-серверну архітектуру, яка передбачає розподіл обчислювальних завдань між клієнтською частиною, що відповідає за взаємодію з користувачем, та серверною частиною, що забезпечує обробку запитів, управління даними та виконання бізнес-логіки.

REST API (Representational State Transfer) є одним із найбільш ефективних способів забезпечення взаємодії між клієнтською та серверною частинами. Ця архітектура базується на використанні стандартних HTTP-запитів (GET, POST, PUT, DELETE) для обміну даними у форматі JSON, що робить її зрозумілою та простою в реалізації. REST API дозволяє розробникам легко інтегрувати нові функції, а також забезпечує масштабованість та незалежність окремих частин системи. Це особливо важливо для вебпорталів психологічної підтримки, де важливо забезпечити швидку обробку запитів користувачів та доступ до різних ресурсів [10].

І оскільки планується використання технології Node.js для реалізації серверної частини, а також сучасних фреймворків для клієнтської частини, розглянемо декілька популярних інструментів та бібліотек, які можуть бути

застосовані для створення ефективної та швидкодіючої системи.

Для створення серверної частини вебпорталу психологічної підтримки важливо забезпечити високу продуктивність, асинхронну обробку запитів та зручність розробки. Node.js є чудовим вибором для реалізації таких рішень, оскільки підтримує асинхронний ввід/вивід та дозволяє створювати масштабовані вебсистеми. Розглянемо декілька популярних фреймворків на базі Node.js [11], які можна використовувати для реалізації серверної логіки:

Express.js – це найпопулярніший фреймворк для розробки серверних додатків на Node.js. Express забезпечує широкий набір інструментів для маршрутизації, обробки HTTP-запитів, роботи з сесіями, кукі та шаблонами. Він підтримує численні middleware-пакети, що дозволяє розширювати можливості системи. Express.js підходить для створення серверних додатків із середньою складністю, де важливі як гнучкість, так і швидкість розробки [12].

Koa.js – це мінімалістичний фреймворк для Node.js, створений командою розробників Express. Він надає розробникам більшу гнучкість у налаштуванні middleware та використовує вбудовану підтримку async/await для роботи з асинхронним кодом, що підвищує його зрозумілість та читабельність. Koa.js підходить для створення масштабованих вебдодатків, де потрібна оптимізація продуктивності та простота коду [13].

Hapi.js – це фреймворк для Node.js, який використовує структуру, подібну до Angular, та підтримує TypeScript. Hapi.js зосереджений на модульному підході до розробки, що робить його хорошим вибором для створення великих проектів. Завдяки вбудованій підтримці TypeScript, Hapi.js забезпечує кращу типізацію та полегшує підтримку коду в довгостроковій перспективі. Він підходить для розробки складних проектів з розподіленою архітектурою та багатошаровими системами [14].

Socket.io – це бібліотека для роботи з WebSockets, яка забезпечує можливість створення реального часу оновлення даних. У контексті вебпорталу психологічної підтримки ця бібліотека може бути використана для реалізації чату між користувачами та психологами в реальному часі, що є важливим

елементом системи для забезпечення швидкого зв'язку між користувачами [15].

Серед усіх цих підходів було обрано Koa.js для реалізації серверної частини, оскільки цей фреймворк забезпечує легку інтеграцію з middleware та підтримку сучасних стандартів JavaScript (ECMAScript 6). Він дозволяє створювати мінімалістичні та продуктивні серверні додатки, що є особливо важливим для забезпечення стабільної роботи вебпорталу з високим рівнем навантаження.

Для розробки клієнтської частини вебпорталу психологічної підтримки важливо забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Сучасні фронтенд-фреймворки дозволяють створювати інтерфейси, що добре реагують на дії користувачів та забезпечують швидке оновлення даних. Розглянемо декілька основних фреймворків для розробки клієнтської частини:

React – це один із найпопулярніших фреймворків для розробки інтерфейсів користувача. React використовує компонентний підхід до створення інтерфейсу, що дозволяє розробникам легко створювати повторно використовувані компоненти. Завдяки віртуальному DOM React забезпечує швидке оновлення даних, що робить його оптимальним вибором для створення динамічних інтерфейсів. React добре підходить для вебпорталів, де важливо забезпечити плавну взаємодію користувачів із системою [16].

Vue.js – це ще один фреймворк для створення інтерактивних інтерфейсів, відомий своєю легкістю у вивченні та використанні. Vue.js дозволяє створювати як прості, так і складні інтерфейси, що робить його хорошим вибором для проектів будь-якого масштабу. Завдяки двосторонньому зв’язку даних Vue.js полегшує синхронізацію між даними та інтерфейсом користувача. Це робить Vue.js привабливим варіантом для швидкого прототипування та розробки зручних інтерфейсів [17].

Angular – це потужний фреймворк від Google, що забезпечує повний набір інструментів для розробки складних клієнтських додатків. Angular використовує структуру на основі компонентів та модулів, що сприяє зручному управлінню

великими проектами. Завдяки використанню TypeScript Angular забезпечує високу типізацію та інструменти для налагодження, що полегшує підтримку коду. Angular підходить для проектів, де важлива масштабованість та розподіл відповідальності між різними компонентами системи [18].

Серед усіх розглянутих фреймворків було обрано React для реалізації клієнтської частини вебпорталу, оскільки цей фреймворк забезпечує високу швидкість роботи інтерфейсу та зручний компонентний підхід. Це дозволяє створювати масштабовані та інтерактивні інтерфейси, що забезпечують зручну взаємодію користувачів із системою [19]. Крім того, велика кількість готових бібліотек та компонентів для React дозволяє прискорити процес розробки та зосередитися на ключових функціях порталу.

1.4 Постановка задач дослідження

Проаналізувавши питання розробки методів і програмних засобів реалізації вебпорталу психологічної підтримки, було визначено завдання, які необхідно виконати для розробки програмного застосунку.

Основними задачами роботи є:

– визначити засоби реалізації вебпорталу для психологічної підтримки;

– розробити методи визначення загального стану пацієнта та персоналізованого підбору спеціалістів на основі результатів;

– розробити методи розрахунку рейтингу спеціалістів;

– розробити програмні модулі для інтеграції телеграм-боту з функціями нагадувань, сповіщень та керування записами на консультації;

– провести тестування вебпорталу для перевірки його функціональності, надійності та відповідності заданим вимогам.

1.5 Висновки

У першому розділі було розглянуто стан розвитку вебпорталів психологічної підтримки та проведено аналіз існуючих методів і підходів до їх

розробки. Під час аналізу були розглянуті популярні вебплатформи, такі як BetterHelp, Talkspace, 7 Cups і Calmerry, та проведено їх порівняння з власним застосунком. У результаті цього порівняння обґрунтовано необхідність створення нового вебпорталу, який поєднує в собі передові функції, такі як тестування, підбір спеціалістів, інтеграцію з телеграм-ботом для нагадувань та динамічних сповіщень, а також інтерактивний календар для планування консультацій.

Основні переваги розроблюваного вебпорталу включають його комплексний підхід до надання послуг психологічної підтримки, інтерактивний і адаптивний інтерфейс, а також використання сучасних технологій для підвищення ефективності та зручності користувачів.

Проведений аналіз методів розробки та архітектурних підходів дозволив обрати клієнт-серверну архітектуру з використанням REST API для забезпечення швидкої та надійної взаємодії між клієнтською і серверною частинами.

Сформульовано основні завдання, які необхідно виконати для успішної реалізації вебпорталу, що підвищує його конкурентоспроможність та спрямованість на задоволення потреб користувачів у сфері психологічної підтримки.

2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ ВЕБПОРТАЛУ

2.1 Архітектурне проєктування вебпорталу

Архітектурне проектування є ключовим етапом у створенні програмного забезпечення, адже саме від вибору архітектурного підходу залежить продуктивність, масштабованість, гнучкість та підтримка системи. Вебпортал психологічної підтримки, який розробляється, повинен не тільки відповідати сучасним вимогам щодо продуктивності та надійності, але й забезпечувати можливість подальшого розвитку та інтеграції з іншими системами.

Існує кілька підходів до архітектури програмних систем. Монолітна архітектура є традиційним рішенням, де всі компоненти програми інтегровані в єдине ціле, що забезпечує швидку розробку та розгортання, але з часом ускладнює підтримку та розширення через взаємозалежність усіх частин. Мікросервісна архітектура дозволяє розділити систему на незалежні сервіси, що забезпечує гнучкість та можливість масштабування окремих частин системи. Проте управління мікросервісами може бути складним і вимагати належної інфраструктури.

Найбільш підходящим рішенням для проєкту вебпорталу психологічної підтримки є багаторівнева архітектура (Layered Architecture) [20], яка забезпечує баланс між складністю розробки та підтримкою коду. Ця архітектура розділяє додаток на кілька логічних шарів, кожен з яких відповідає за певну функцію, що дозволяє чітко структурувати код і підвищити його підтримуваність та модульність.

Layered Architecture передбачає чітке розділення системи на кілька основних шарів:

1. Презентаційний шар (Presentation Layer) відповідає за взаємодію з користувачем. У вебпорталі психологічної підтримки цей шар реалізується за допомогою React, що дозволяє створювати адаптивні, динамічні та інтуїтивно зрозумілі інтерфейси користувача.
2. Шар бізнес-логіки (Business Logic Layer) містить основні правила та

алгоритми обробки даних. Використання менеджерів і сервісів дозволяє чітко відокремити бізнес-логіку від інших частин системи, що спрощує тестування та підтримку.

3. Шар доступу до даних (Data Access Layer) відповідає за взаємодію з базою даних. Виконує всі операції доступу до даних, такі як збереження, оновлення, видалення та отримання даних. Для зручності роботи з базою даних використовуються ORM-бібліотеки, наприклад, Sequelize.

4. База даних (Database Layer) фізичний шар, що зберігає інформацію про користувачів, результати тестувань, розклад консультацій та інші дані.

Для організації коду серверної частини було застосовано модульний підхід, що відповідає принципам Layered Architecture:

1. Файл index.js виконує роль вхідного файлу або головного скрипта проєкту. Він відповідає за ініціалізацію додатку, підключення до бази даних та запуск серверу.

2. Файл routes.js відповідає за маршрутизацію (routing) запитів. У ньому визначаються шляхи (URL-шаблони) та відповідні контролери, що обробляють запити.

3. Директорія constants містить файли з константами та змінними, які використовуються у проєкті. Це забезпечує централізоване управління важливими параметрами та налаштуваннями.

4. Директорії managers, controllers, helpers:

- директорія managers містить файли, що відповідають за виконання бізнес-логіки, керування даними та обробку запитів до бази даних;
- директорія controllers містить файли, що обробляють HTTP-запити, викликаючи відповідні функції з менеджерів для отримання або обробки даних;
- директорія helpers містить допоміжні утиліти та функції, що використовуються для виконання загальних операцій, наприклад, форматування або валідації даних.

Візуально підхід Layered Architecture можна побачити у структурі файлів серверної частини вебпорталу (рисунок 2.1).

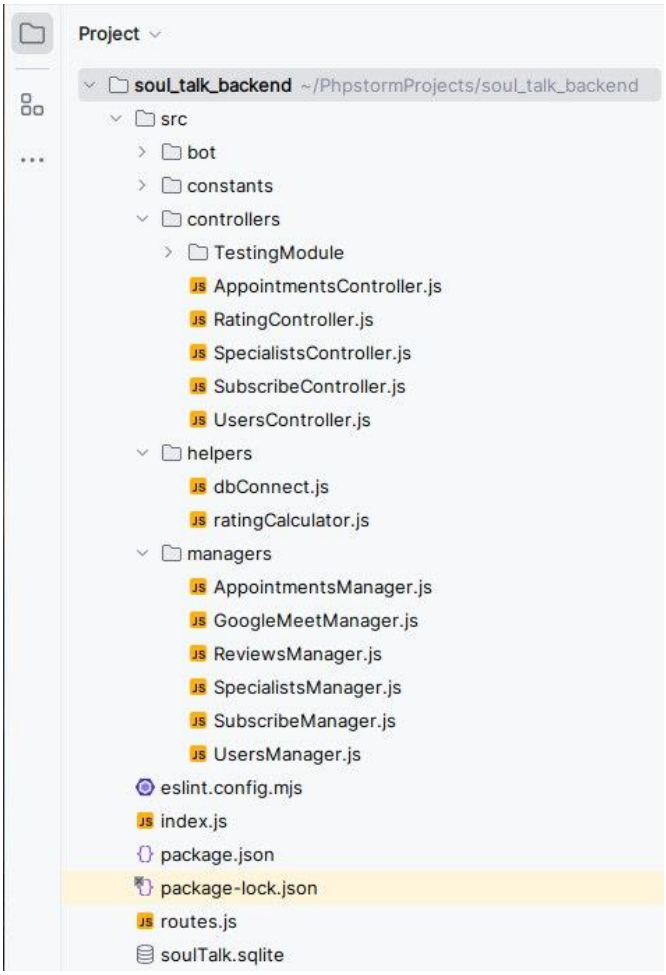

Рисунок 2.1 – Структура файлі серверної частини вебпорталу

Використання багаторівневої архітектури забезпечує низку значних переваг для вебпорталу психологічної підтримки. Однією з головних переваг є модульність, яка дозволяє легко замінювати або змінювати окремі компоненти системи без впливу на інші частини [21]. Це сприяє зручності в підтримці та розширенні, оскільки чітке розділення обов'язків між шарами спрощує процес додавання нового функціоналу. Масштабованість системи також стає простішою, адже кожен шар може розвиватися та оптимізуватися незалежно від інших. Важливою перевагою є й зручність тестування: ізольовані шари можна перевіряти окремо, що значно полегшує процес виявлення та усунення помилок.

Таким чином, вибір багаторівневої архітектури для розробки вебпорталу забезпечує не лише ефективне створення системи, але й гарантує її гнучкість, масштабованість та надійність у подальшій експлуатації. Завдяки структурованому підходу та чіткому розподілу обов'язків, ця архітектура

дозволяє створювати програмні продукти, готові до майбутніх змін та інтеграцій, забезпечуючи надійну основу для розвитку системи.

2.2 Розробка загальної моделі вебпорталу

Розробка вебпорталу психологічної підтримки передбачає створення комплексного інтерактивного середовища, яке надає користувачам доступ до якісних психологічних послуг у зручний для них спосіб. Основною метою вебпорталу є забезпечення безперебійного зв’язку між користувачами та спеціалістами, включаючи можливість онлайн-запису на консультації, управління підписками на спеціалістів та отримання зворотного зв’язку у вигляді відгуків та оцінок.

З технічної точки зору, портал розроблено з використанням сучасних технологій, що забезпечують надійність, масштабованість та безпеку даних користувачів. Основна архітектура платформи базується на принципах багатошарової архітектури, де frontend (інтерфейс користувача) взаємодіє із backend (серверною частиною) через HTTP-запити. Серверну частину реалізовано на базі Node.js із використанням фреймворку Koa, що дозволяє створювати гнучкі й ефективні серверні додатки.

Крім того, система містить функціонал Telegram-боту для додаткової інтеграції з месенджером, що підвищує доступність інформації для користувачів. Telegram-бот інтегрований у систему як окремий модуль, що забезпечує взаємодію з користувачами через популярний месенджер. Він дозволяє користувачам підписуватися на оновлення, переглядати свої записи та здійснювати нові записи на прийоми. Це підвищує доступність та зручність використання порталу.

На рисунку 2.2 показано загальну модель взаємодії компонентів порталу. Користувачі взаємодіють з вебпорталом через інтерфейс, відправляючи запити, які обробляються контролерами. Контролери передають запити до менеджерів, які виконують обробку даних та взаємодіють з базою даних SQLite для збереження та отримання інформації. Додатково система взаємодіє з модулем

Telegram-боту для оповіщення користувачів та надання інформації про записи та підписки.

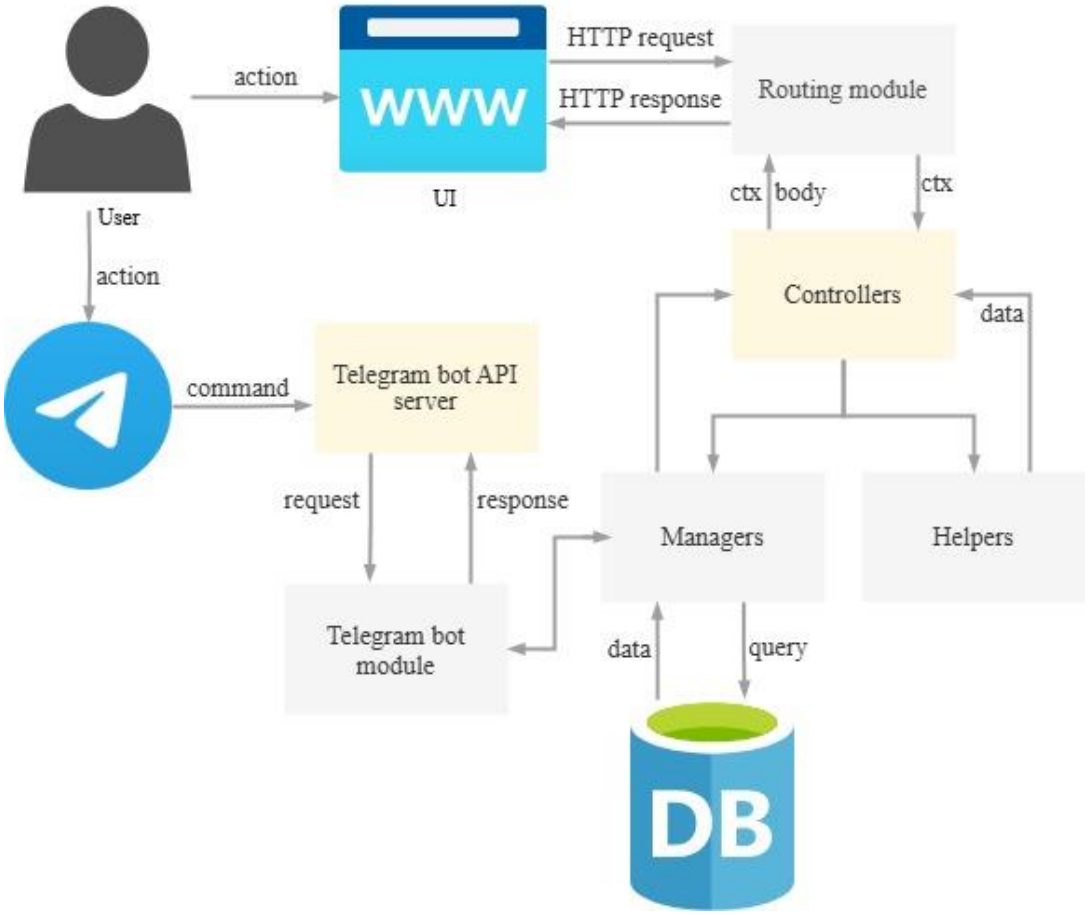

Рисунок 2.2 – Загальна модель роботи вебпорталу

Для забезпечення ефективної роботи порталу було розроблено структуру бази даних, яка підтримує всі необхідні функції системи, включаючи зберігання даних користувачів, спеціалістів, записів на консультації, відгуків та підписок на оновлення. База даних реалізована на основі системи SQLite, що забезпечує простоту використання та ефективну обробку даних у невеликих і середніх проєктах.

Для візуалізації структури бази даних було створено EER (діаграму сутність-зв'язок) [22], яка відображає взаємозв'язок між основними таблицями бази даних, такими як users, specialists, appointments, reviews та telegram_subscribers. Ця діаграма дозволяє чітко побачити, як дані структуровані

та які зв'язки існують між сутностями (рисунок 2.3).

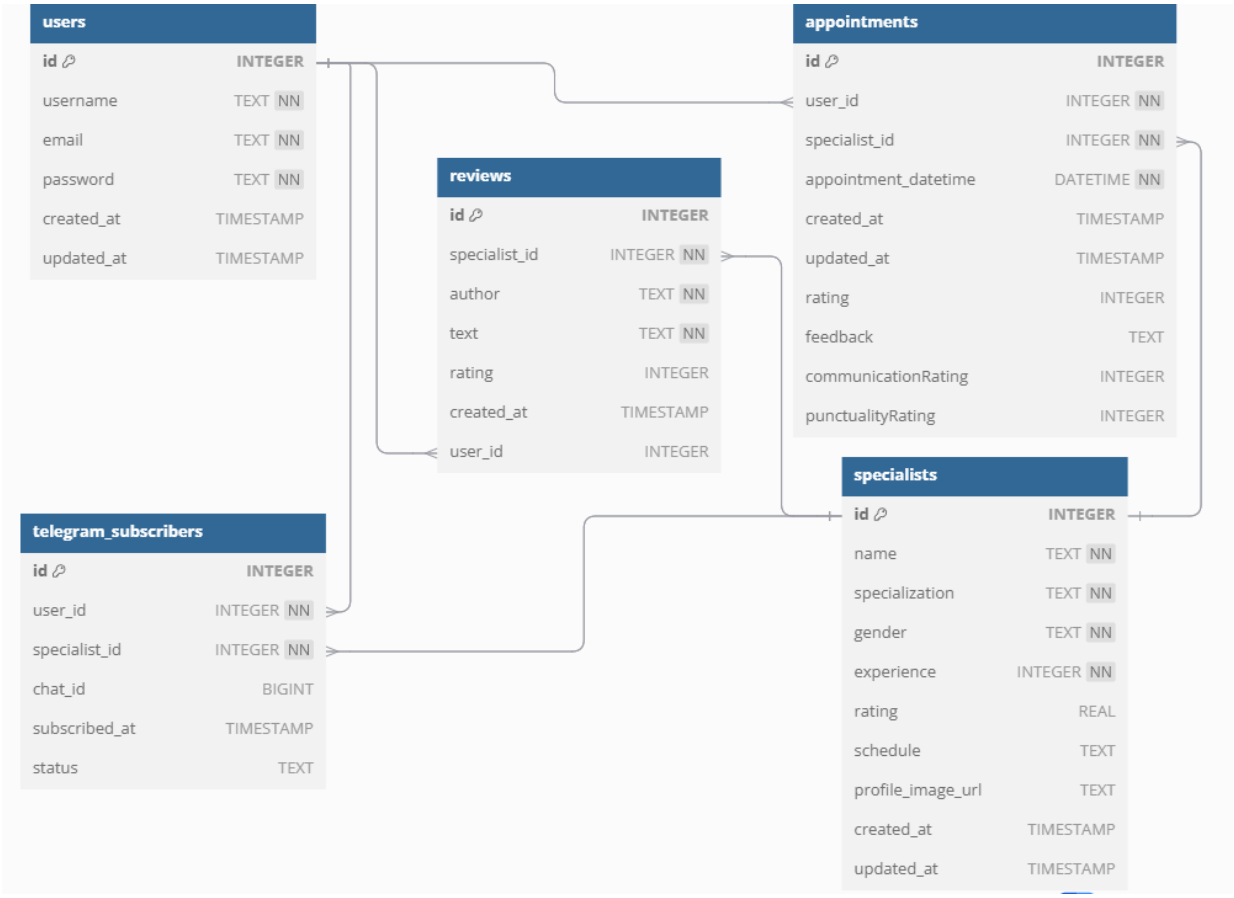

Рисунок 2.3 – Узагальнена модель бази даних вебпорталу

Отже, у цьому розділі було розроблено комплексну модель вебпорталу психологічної підтримки, яка включає багатошарову архітектуру із розподілом на клієнтську, серверну частину та базу даних. Також була створена EER-модель бази даних, яка чітко відображає взаємозв'язки між основними сутностями системи, такими як користувачі, спеціалісти, записи на консультації, відгуки та підписки.

2.3 Розробка методу визначення загального стану пацієнта та персоналізованого підбору спеціалістів на основі результатів

Розробка методу підбору спеціалістів є важливим компонентом системи вебпорталу психологічної підтримки, що спрямований на надання користувачам доступу до найбільш релевантних спеціалістів для їх індивідуальних потреб.

Застосування тестування дозволяє здійснювати підбір спеціалістів з урахуванням особистих характеристик та специфічних вимог користувача, що сприяє підвищенню ефективності надання психологічних послуг.

Розглянемо метод визначення загального стану пацієнта та персоналізованого підбору спеціалістів на основі результатів детальніше:

1. Початковий етап. Користувач переходить на сторінку тестування, де йому пропонується пройти спеціальний тест для визначення його потреб та очікувань. Тест включає питання, що стосуються поточного емоційного стану, типу необхідної підтримки та інших важливих аспектів.

2. Введення даних. Користувач заповнює тест, надаючи інформацію, яка допоможе системі визначити його потреби.

3. Надсилання даних на сервер. Заповнені дані надсилаються на сервер для подальшої обробки за допомогою відповідних контролерів. Контролери передають ці дані менеджерам, які відповідають за управління логікою обробки інформації.

4. Аналіз даних та підбір спеціалістів. Менеджери здійснюють аналіз даних, використовуючи алгоритми обробки, які враховують вагові коефіцієнти для різних критеріїв підбору. Це дозволяє підвищити точність підбору, адаптуючи його до конкретних потреб користувача.

5. Якщо система успішно підбирає спеціалістів, користувач перенаправляється на сторінку результатів, де відображається список рекомендованих спеціалістів із зазначенням їхньої спеціалізації, досвіду та рейтингу.

6. Якщо спеціалісти не відповідають критеріям, користувач отримує повідомлення про це та має змогу перейти до загального каталогу спеціалістів для самостійного вибору.

7. Завершення процесу. У випадку успішного підбору або переходу до загального каталогу завершується взаємодія з модулем.

На рисунку 2.4 подана блок-схема алгоритму роботи модуля підбору спеціалістів, яка ілюструє логіку виконання процесу.

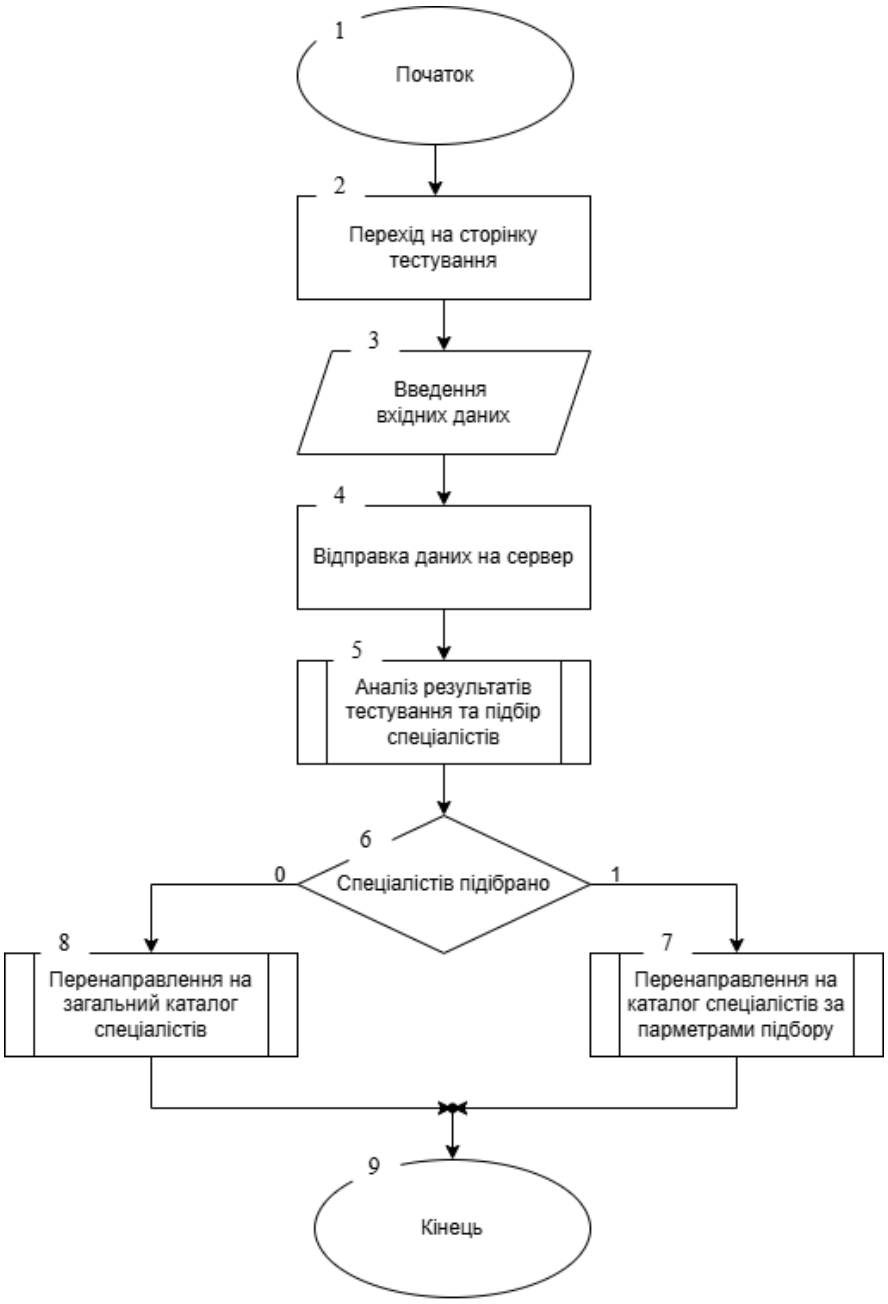

Рисунок 2.4 – Блок-схема алгоритму роботи модуля підбору спеціалістів

Отже, розроблений метод підбору спеціалістів на основі тестування дозволяє підвищити точність та персоналізацію вибору психологів для користувачів вебпорталу.

2.4 Розробка методу розрахунку рейтингу спеціалістів

Метод розрахунку рейтингу спеціалістів є ключовим елементом вебпорталу психологічної підтримки, оскільки він дозволяє користувачам отримувати об'єктивну оцінку кваліфікації спеціалістів на основі різних

критеріїв. Метод базується на отриманні й обробці даних про спеціаліста, які включають його досвід роботи, відгуки клієнтів та кількість успішно завершених записів на консультації.

Найголовнішим в такому методі є обрати підхід до обрахунку загальної оцінки рейтнгу. Звичайно, найпростіший спосіб – це оцінювання спеціалістів за середнім арифметичним всіх пунктів рейтингу, проте він має суттєві недоліки.

1. Насамперед в такому підході не враховується різна важливість критеріїв. Наприклад, "ефективність консультацій" може бути важливішою для користувача, ніж "популярність у Telegram".

2. Чутливість до окремих низьких оцінок. Одна низька оцінка може значно вплинути на середній результат, навіть якщо більшість інших оцінок високі.

3. Недооцінка або переоцінка спеціалістів. Спеціалісти з малою кількістю відгуків можуть отримувати завищений або занижений рейтинг, що робить систему менш справедливою.

Саме тому вирішено використати метод вагових коефіцієнтів та їх нормалізацію. Вагові коефіцієнти – це числові значення, що визначають важливість кожного з критеріїв, які враховуються під час розрахунку рейтингу спеціаліста. Вони дозволяють відображати вплив кожного критерію на підсумкову оцінку відповідно до його значущості [23].

Нормалізація даних у контексті побудови рейтингової системи спеціалістів – це процес масштабування вагових коефіцієнтів критеріїв таким чином, щоб їх сума дорівнювала одиниці. Це забезпечує коректний розподіл впливу кожного критерію на підсумковий рейтинг, запобігаючи дисбалансу між різними аспектами оцінювання [24].

Тобто, вагові коефіцієнти дозволять враховувати важливість кожного критерію, а їх нормалізація в свою чергу гарантує, що всі значення коректно масштабується у межах від 0 до 1, забезпечуючи справедливий внесок кожного критерію у загальний рейтинг.

Розглянемо алгоритм роботи методу розрахунку рейтингу спеціалістів

детальніше:

1. Відкриття сторінки спеціаліста – початок процесу, коли користувач взаємодіє з інтерфейсом вебпорталу.

2. Запит на отримання рейтингу спеціаліста – відправлення запиту до сервера для отримання необхідних даних.

3. Перевірка існування спеціаліста за айді – перевіряється, чи існує такий спеціаліст у базі даних.

4. У разі невдачі повертається повідомлення про відсутність рейтингу.

5. Отримання усіх даних про спеціаліста, відгуків, підписників та записів на консультацію.

6. Обчислення критеріїв рейтингу – оцінюються такі фактори, як досвід, задоволеність користувачів та ефективність. Усі критерії, які мають більше 1 балу, включаються у подальший рейтинг.

7. Застосування методу вагових коефіцієнтів та їх нормалізації, для обчислення загальної оцінки рейтингу. Детальніше про процес:

7.1. Визначення вагових коефіцієнтів.

7.2. Перевірка суми вагових коефіцієнтів. Якщо сума дорівнює 1, нормалізація не потрібна. Якщо сума більше або менше 1 відбувається перехід до наступного кроку.

7.3 Виконується нормалізація коефіцієнтів, щоб їх сума дорівнювала одиниці. Це забезпечує збалансованість впливу ваг критеріїв на загальну оцінку.

7.3 Обчислення загальної оцінки за методом вагових коефіцієнтів. Загальна оцінка розраховується як сума добутків значень критеріїв на їх вагові коефіцієнти.

8. Формування загальної оцінки та її передача на клієнтську частину – підрахунок фінальної оцінки з урахуванням вагових коефіцієнтів.

Алгоритм закінчується відправленням всіх даних на клієнтську частину, де рейтинг спеціаліста відображається для користувача на сторінці конкретного спеціаліста у вигляді блоку з вказаним загальним рейтингом, а також детальним поясненням кожного пункту рейтингу (рисунок 2.5).

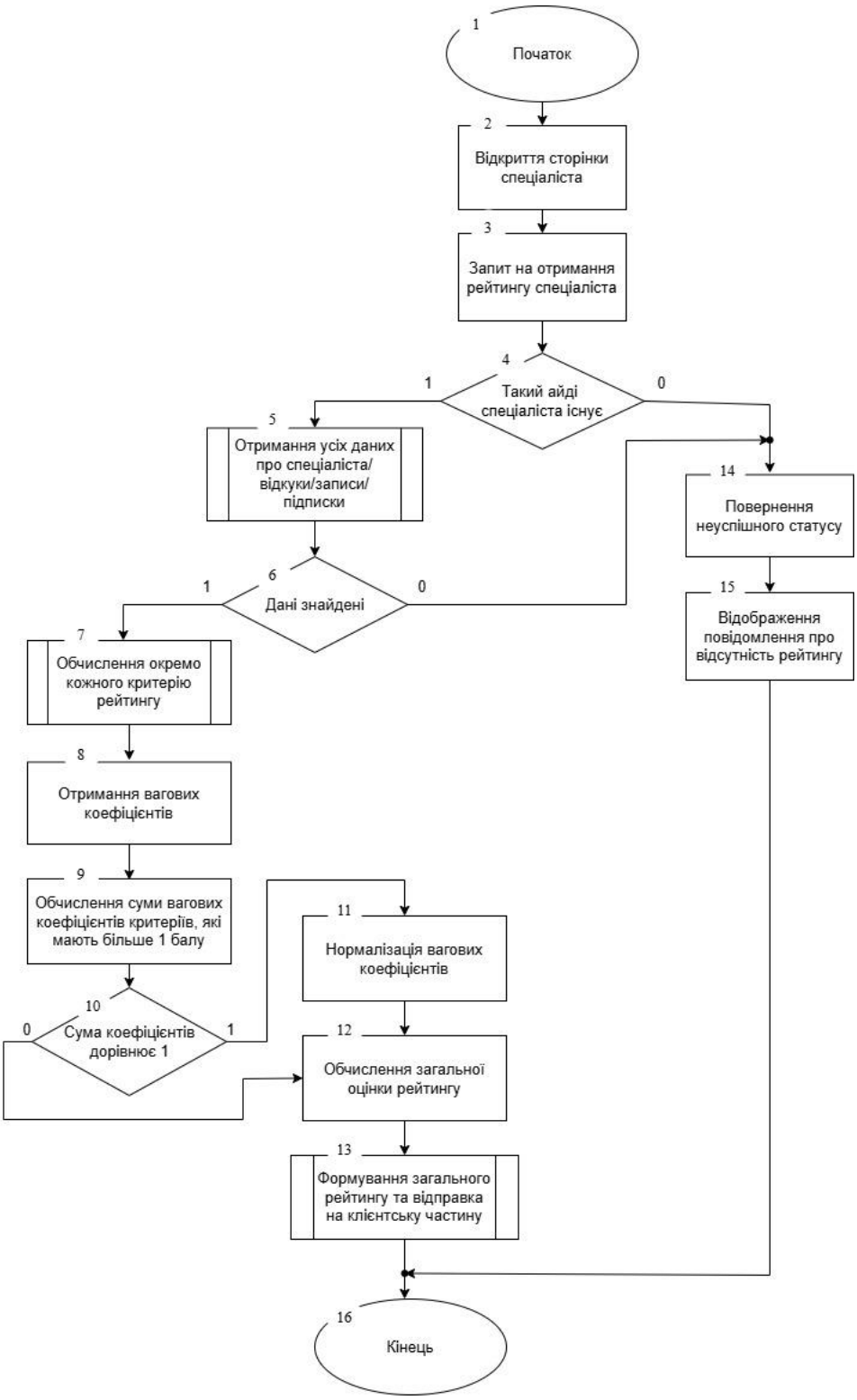

Рисунок 2.5 – Блок-схема роботи модуля рейтингу спеціаліста

Отже, було розроблено метод розрахунку рейтингу спеціалістів, який використовує дані про досвід, відгуки та записи. Важливою особливістю методу розрахунку рейтингу спеціалістів є використання вагових коефіцієнтів для кожного з критеріїв. Це дозволяє надавати більшу вагу найбільш значущим

факторам, наприклад, таким як ефективність консультування та репутація.

2.5 Розробка методу інтеграції телеграм-боту з функціями динамічних сповіщень і аналітичних звітів

Telegram-бот – це ПЗ, що працює всередині месенджера Telegram і дозволяє автоматизувати взаємодію між користувачами та сервісами.

Інтеграція Telegram-боту є важливим компонентом вебпорталу психологічної підтримки, що забезпечує динамічну взаємодію між користувачами, спеціалістами та серверною частиною порталу. Цей метод дозволяє реалізувати автоматичне надсилання нагадувань, отримання зворотного зв’язку про консультації та формування аналітичних звітів. Застосування Telegram-боту сприяє підвищенню зручності користування системою та оптимізації процесів комунікації.

Розглянемо метод інтеграції Telegram-боту з функціями динамічних сповіщень і аналітичних звітів детальніше:

1. Реєстрація Telegram-боту. Реєстрація відбувається через BotFather сервіс. Результатом реєстрації є унікальний токен доступу. За допомогою токену доступу та бібліотеки node-telegram-bot-api бот підключається до Telegram API.

2. Налаштування середовища бекенду. Під час цього процесу налаштовується серверна частина вебпорталу, інтегруючи бібліотеку для роботи з Telegram API. Реалізується базовий функціонал, такий як ініціалізація командного меню, обробка запитів з Telegram API і з’єднання з базою даних.

3. Реалізація функціоналу боту. Створення функцій для обробки команд від користувача, наприклад формування статистики, перегляд персональних підписок та записів. А також налаштування динамічних та запланованих сповіщень, використовуючи систему планування подій Node Schedule. Прикладом таких сповіщень є надсилання нагадувань про запис на консультацію або запити щодо зворотнього зв’язку.

Алгоритм роботи методу завершується відправленням даних на Telegram API та відображенням у боті відповідного результату команди (рисунок 2.6).

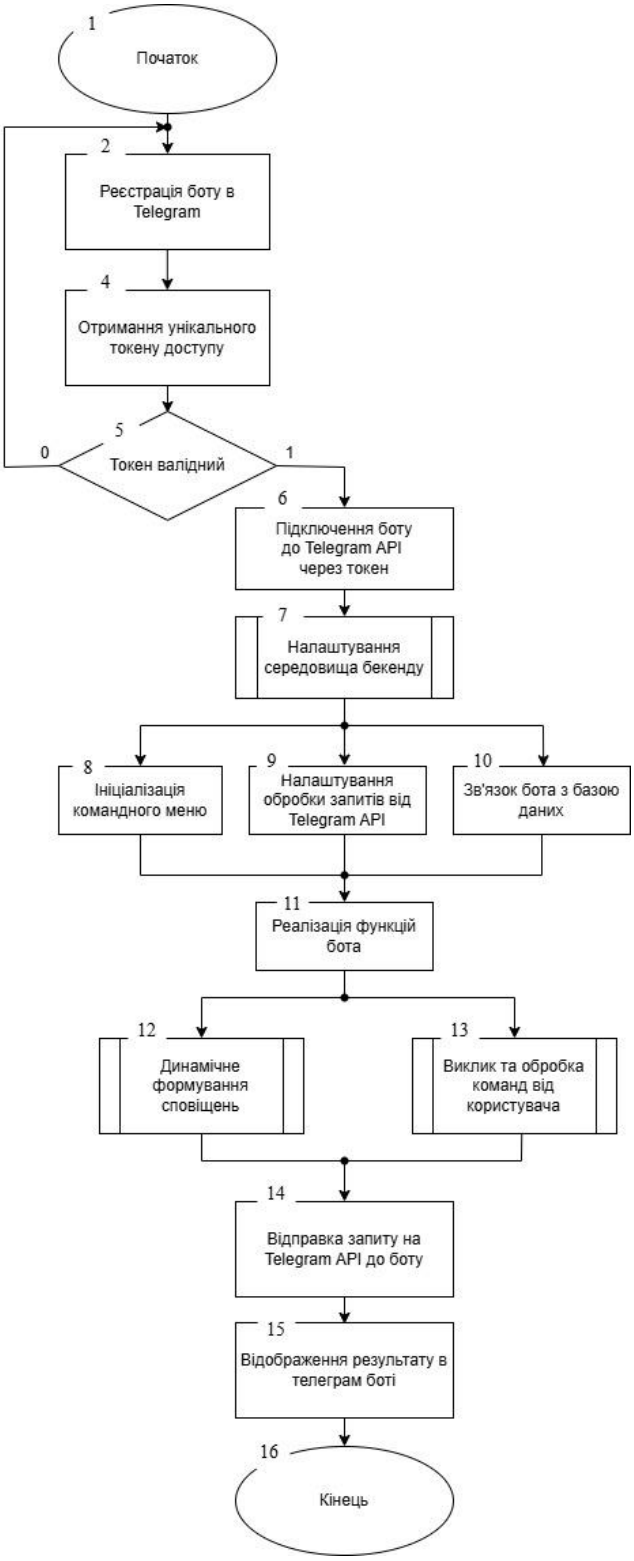

Рисунок 2.6 – Блок-схема алгоритму інтеграції телеграм-боту

Отже, було розроблено метод інтеграції телеграм-боту з функціями динамічних сповіщень і аналітичних звітів. Який на відміну від існуючих рішень, базуються на системі планування подій Node Schedule.

2.6 Розробка моделі вебпорталу психологічної підтримки

Існування зв'язків між елементами системи та їх відносини є однією з ключових характеристик для створення ефективної архітектури програмного забезпечення. Для візуалізації та документування цих зв'язків у процесі проєктування застосовуються діаграми, що описують різні аспекти роботи системи. Одним із найпоширеніших способів моделювання систем є використання уніфікованої мови моделювання (UML).

UML (Unified Modeling Language) – це уніфікована мова моделювання, яка дозволяє створювати діаграми для представлення структури та поведінки програмних систем. Це не мова програмування, а набір інструментів, що використовується для проектування та аналізу системи, її документації та комунікації між учасниками проєкту. UML активно використовується у різних галузях, включаючи розробку програмного забезпечення, бізнес-аналітику, системний дизайн та архітектуру [25].

Основні типи діаграм UML:

1. Діаграми структури, які описують статичну структуру системи, такі як діаграми класів, діаграми компонентів, діаграми пакетів та діаграми розгортання.

2. Діаграми поведінки, які описують динамічну поведінку системи, наприклад, діаграми варіантів використання, діаграми послідовностей та діаграми активностей.

3. Діаграми реалізації, що відображають фізичну реалізацію системи, такі як діаграми компонентів та діаграми розгортання.

UML є потужним інструментом для візуалізації взаємодії між акторами (користувачами) та функціональністю системи. Однією з діаграм, що допомагає відобразити ці зв'язки, є діаграма варіантів використання (Use-case diagram). Вона дозволяє зрозуміти, як різні користувачі взаємодіють із системою, визначаючи функціональні вимоги системи.

Діаграма варіантів використання складається з двох основних елементів:

1. Актор (Actor) – роль, яку виконує користувач або інша система при

взаємодії з прецедентами.

2. Прецедент (Use case) – опис окремого аспекту функціональності системи з точки зору користувача.

На рисунку 2.7 зображено діаграму варіантів використання для вебсистеми психологічної підтримки, що демонструє основні взаємодії користувачів із системою, включаючи користувачів, спеціалістів і Telegram-боту.

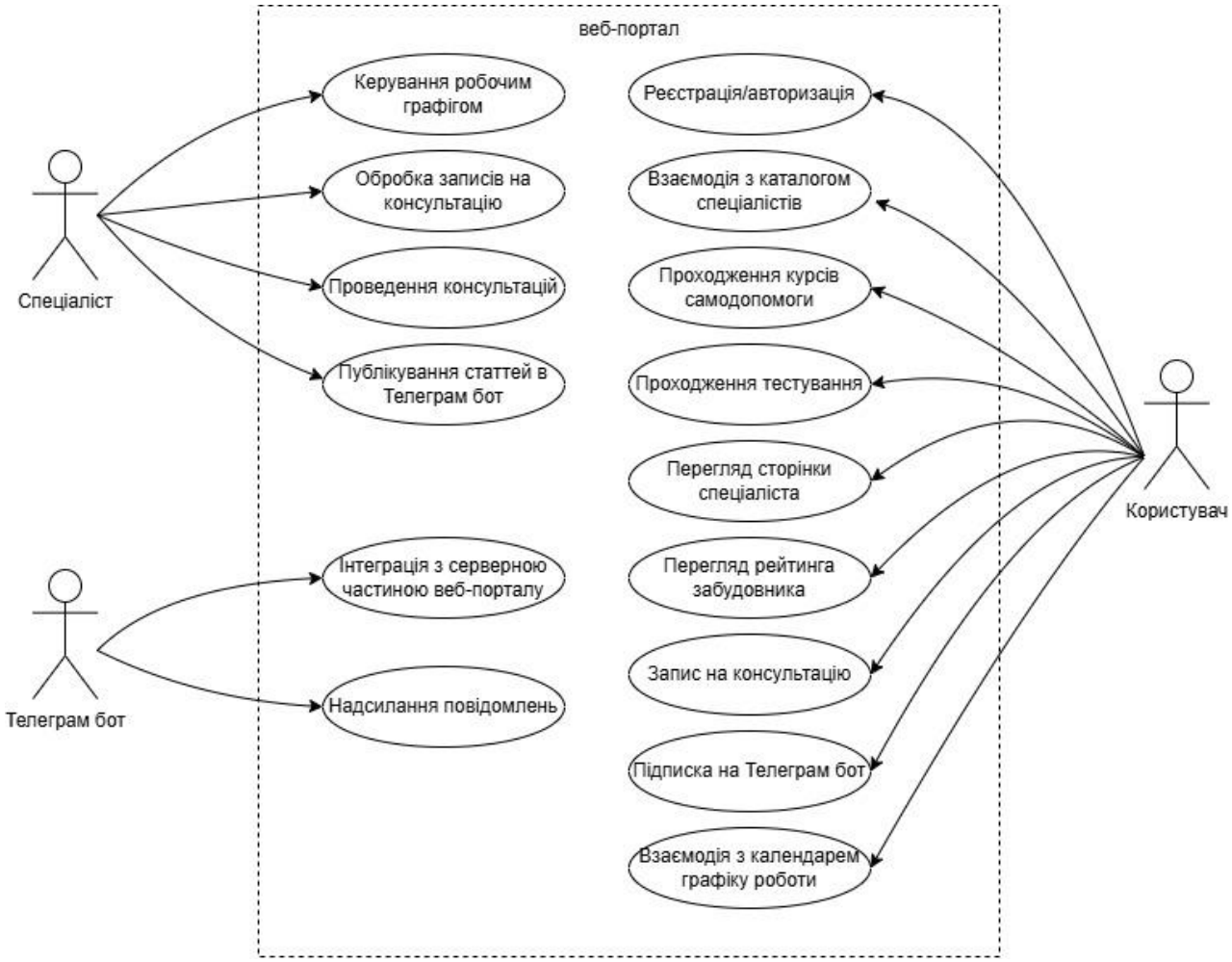

Рисунок 2.7 – Модель вебпорталу у вигляді діаграми варіантів використання

Головна мета вебсистеми полягає у створенні зручного інтерфейсу для взаємодії користувачів зі спеціалістами, підтриманні комунікацій через Telegram-боту та забезпеченні повного циклу роботи системи психологічної підтримки. Перелік варіантів використання вебпорталу та їх короткий опис наведено в таблиці 2.1.

Таблиця 2.1 – Реєстр варіантів використання

Актор	Прецедент	Опис
Користувач	Реєстрація/авторизація	Введення даних для створення облікового запису або авторизації
	Взаємодія з каталогом спеціалістів	Огляд доступних спеціалістів для вибору відповідного спеціаліста
	Проходження тестування	Виконання тестів для визначення поточного психологічного стану
	Проходження курсів самодопомоги	Доступ до курсів для покращення психоемоційного стану
	Перегляд сторінки спеціаліста	Перегляд детальної інформації про спеціаліста
	Взаємодія з календарем графіку роботи спеціаліста	Вибір доступного часу для запису на консультацію
	Перегляд рейтингу спеціаліста	Оцінка рейтингу спеціаліста на основі відгуків і консультацій
	Запис на консультацію	Запис на прийом до спеціаліста
	Підписка на Телеграм бот	Підписка для отримання нагадувань та сповіщень
Спеціаліст	Проведення консультацій	Проведення консультацій та обробка даних клієнтів
	Публікування статей у Телеграм бот	Публікація матеріалів у Telegram боті
	Керування робочим графіком	Зміна та налаштування свого робочого розкладу
	Обробка записів на консультацію	Перевірка та управління записами клієнтів
Telegram Bot	Надсилання повідомлень	Надсилання повідомлень клієнтам у Телеграм бот
	Інтеграція з серверною частиною вебпорталу	Зв’язок із базою даних для отримання інформації про консультації, підписки та рейтинг.

Отже, було розглянуто модель роботи вебсистеми у вигляді діаграми варіантів використання, яка ілюструє ключові взаємодії між акторами та прецедентами, що дозволяє краще уявити основні процеси системи.

2.7 Висновки

У другому розділі було розроблено комплексну модель вебпорталу психологічної підтримки, яка включає багатошарову архітектуру із розподілом на фронтенд, бекенд та базу даних. На основі створеної моделі забезпечено взаємодію користувачів із системою через інтерфейс, обробку запитів за допомогою контролерів та менеджерів, а також управління даними за допомогою бази даних SQLite.

Також була створена EER-модель бази даних, яка чітко відображає взаємозв'язки між основними сутностями системи, такими як користувачі, спеціалісти, записи на консультації, відгуки та підписки. Ця структура сприяє ефективному управлінню даними та забезпечує цілісність і зв'язність інформації у системі.

Крім того, було створено діаграму варіантів використання, яка ілюструє взаємодію між акторами системи, такими як користувачі, спеціалісти та Telegram-бот, та основними функціями вебпорталу. Ця діаграма дозволила визначити ключові функції, що забезпечують основний функціонал порталу, такі як реєстрація, запис на консультацію, управління графіком, публікація матеріалів у Telegram, а також проходження курсів і тестів користувачами.

Також у цьому розділі було розроблено метод розрахунку рейтингу спеціалістів, який включає обробку даних про спеціаліста, відгуки та успішні записи. Використання вагових коефіцієнтів, нормалізації та дисперсійного аналізу забезпечує об'єктивність, адаптивність та стабільність оцінки, що підвищує надійність рейтингової системи вебпорталу.

Усі ці елементи створюють основу для побудови функціональної, адаптивної та надійної системи, яка задовольняє потреби користувачів та спеціалістів у наданні та отриманні психологічної підтримки онлайн.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ВЕБПОРТАЛУ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Розробка вебпорталу психологічної підтримки вимагає вибору відповідних засобів для забезпечення надійності, ефективності та масштабованості системи. Аналіз варіантів мови програмування та середовища розробки дозволяє зробити обґрунтований вибір для реалізації цього програмного засобу.

Для реалізації сучасного вебпорталу психологічної підтримки важливо обрати такі технології, які дозволять забезпечити єдність у розробці клієнтської та серверної частини. Це дозволяє значно спростити процес розробки, тестування та підтримки коду, оскільки розробники можуть використовувати одну і ту ж мову програмування для обох частин проєкту.

Аналіз мов програмування, які підходять для клієнтської та серверної розробки:

1. JavaScript (Node.js + React). JavaScript – це найпопулярніша мова програмування для веброзробки, яка використовується для створення як клієнтської, так і серверної частини вебдодатків [26]. Завдяки фреймворкам і середовищам, таким як React і Node.js, JavaScript дозволяє створювати інтерактивні вебдодатки з використанням однієї мови для фронтенду та бекенду.

Переваги використання JavaScript:

– єдність мови на фронтенді та бекенді спрощує розробку та підтримку коду;

– велика екосистема, що включає багато бібліотек та фреймворків (React, Express, Next.js);

– швидка розробка завдяки асинхронній обробці та простому інтегруванню нових функцій.

Недоліками використання JavaScript є:

– складність управління асинхронним кодом у великих проєктах;

– потреба в ретельному управлінні залежностями через велику кількість зовнішніх бібліотек.

2. Python (Django/Flask). Python – це інтерпретована, високоуровнева мова програмування, відома своїм простим синтаксисом і читабельністю коду. Використовується для розробки серверної частини вебдодатків завдяки фреймворкам Django та Flask, які забезпечують швидку розробку та мають велику кількість вбудованих функцій [27].

Перевагами використання Python є:

– простота вивчення та написання коду завдяки зрозумілому синтаксису;

– підтримка великої кількості бібліотек для обробки даних, машинного навчання та аналітики;

– Django пропонує готові модулі для аутентифікації, управління користувачами та обробки запитів.

Недоліками використання Python є:

– обмежена продуктивність у випадках високого навантаження;

– більше підходить для серверної частини, що вимагає інтеграції з JavaScript для динамічного фронтенду.

3. Java (Spring Boot). Java – одна з найбільш стабільних і надійних мов програмування, яка використовується для розробки корпоративних систем і вебдодатків. Spring Boot – популярний фреймворк для швидкої розробки додатків на Java з мінімальною конфігурацією [28].

Перевагами використання Java є:

– висока продуктивність, надійність та масштабованість, особливо для великих проєктів;

– підтримка складних архітектурних підходів та багаторівневих систем;

– велика кількість інструментів для тестування, моніторингу та управління безпекою.

Недоліками використання Java є:

– високий поріг входу через складність мови та фреймворку;

– більше часу на розробку через необхідність детальної конфігурації та налаштувань.

4. Ruby (Ruby on Rails). Ruby – це об'єктно-орієнтована мова програмування, відома своєю простотою та елегантністю. Ruby on Rails – фреймворк, який дозволяє швидко створювати вебдодатки з мінімальними витратами на написання коду завдяки принципу «конвенція замість конфігурації» [29].

Перевагами використання Ruby є:

– легкість написання та висока читабельність коду;

– вбудовані інструменти для аутентифікації, управління ресурсами та безпеки;

– швидкий початок розробки та мінімальна кількість конфігурацій.

Недоліками використання Ruby є:

– менш придатна для високонавантажених додатків;

– зменшена популярність та менша кількість розробників у спільноті.

5. PHP (Laravel). PHP – одна з перших мов, розроблених спеціально для вебпрограмування. Laravel є популярним фреймворком для PHP, що дозволяє швидко створювати вебдодатки завдяки інтуїтивному синтаксису та великій кількості вбудованих інструментів [30].

Перевагами використання PHP є:

– легкість у навчанні та використанні для новачків;

– велика кількість готових модулів і бібліотек;

– висока продуктивність при правильній оптимізації коду.

Недоліками використання PHP є:

– менш сучасний підхід до розробки порівняно з новими мовами та фреймворками;

– додаткові налаштування можуть бути необхідними для підвищення безпеки.

6. Go (Golang). Go – це сучасна компільована мова програмування,

розроблена для створення високопродуктивних серверних додатків. Вона використовується в компаніях з високими вимогами до продуктивності завдяки можливостям роботи з багатопоточністю [31].

Перевагами використання Go є:

– висока продуктивність та ефективна робота з багатопоточними додатками;

– строгість типів підвищує надійність коду;

– простота написання та компіляція, що забезпечує швидке виконання.

Недоліками використання Go є:

– менша кількість бібліотек та фреймворків для веброзробки;

– складність інтеграції з фронтендом та розробка складних клієнтських логік.

З огляду на аналіз різних мов програмування для створення вебпорталу психологічної підтримки, JavaScript (Node.js для бекенду та React для фронтенду) обрано як оптимальний варіант. Це дозволяє використовувати єдину мову для всіх частин проєкту, прискорює розробку, забезпечує високу продуктивність та має потужну підтримку спільноти.

Оскільки, обрано JavaScript (Node.js для бекенду та React для фронтенду), то проаналізуємо найпопулярніші середовища для роботи:

1. Visual Studio Code (VS Code). Visual Studio Code – це популярний редактор коду, розроблений Microsoft [32]. Він є відкритим і безкоштовним, підтримує широкий спектр мов програмування та має багату екосистему розширень, що робить його універсальним рішенням для будь-яких проєктів.

Перевагами використання Visual Studio Code є:

– відкритий та безкоштовний, підтримує розширення для JavaScript, Node.js та React;

– вбудований дебагер, автодоповнення та підтримка Git спрощують розробку та тестування;

– підтримка IntelliSense для інтелектуального доповнення коду та перевірки синтаксису.

Недоліками використання Visual Studio Code є:

– може бути важким для запуску на менш потужних комп'ютерах;

– потрібна настройка для використання певних функцій, таких як тестування та відлагодження.

2. WebStorm. WebStorm – це професійна IDE, розроблена компанією JetBrains, яка спеціалізується на розробці JavaScript-додатків. WebStorm має широкі можливості для розробки, включаючи інструменти для налагодження, тестування та рефакторингу коду [33].

Перевагами використання WebStorm є:

– потужні інструменти для роботи з кодом: дебагер, рефакторинг, підказки та підтримка багатьох фреймворків;

– вбудовані інструменти для аналізу коду та інтеграція з популярними системами контролю версій (Git);

– висока продуктивність та інтеграція з іншими продуктами JetBrains.

Недоліками використання WebStorm є:

– платне середовище, що може стати перешкодою для початківців та невеликих команд;

– вимогливість до ресурсів системи порівняно з іншими редакторами.

3. Sublime Text. Sublime Text – це легкий текстовий редактор з підтримкою численних мов програмування, включаючи JavaScript. Його популярність обумовлена швидкодією, простотою використання та великою кількістю розширень [34].

Перевагами використання Sublime Text є:

– висока швидкодія та можливість роботи з великими файлами;

– підтримка різних мов програмування через плагіни та розширення;

– зручний та налаштовуваний інтерфейс.

Недоліками використання Sublime Text є:

– відсутність вбудованого дебагера та підтримки деяких складних функцій;

– потрібно купувати ліцензію для повного використання.

4. Atom. Atom – це відкритий редактор коду, створений GitHub. Він забезпечує гнучкість завдяки великій кількості доступних плагінів і налаштувань, що робить його популярним серед веброзробників [35].

Перевагами використання Atom є:

– безкоштовний та має відкритий код;

– легкий у налаштуванні та гнучкий завдяки плагінам;

– інтуїтивно зрозумілий інтерфейс.

Недоліками використання Atom є:

– порівняно низька продуктивність для великих проєктів;

– відсутність деяких функцій для відлагодження та тестування у стандартній комплектації.

5. PhpStorm. PhpStorm, розроблений JetBrains, є потужним середовищем розробки, яке підтримує численні мови програмування, включаючи JavaScript, PHP та інші. PhpStorm популярний серед розробників, які працюють із повними стеком технологій для вебдодатків [36].

Перевагами використання PhpStorm є:

– підтримка різних мов, інструменти для рефакторингу коду, відлагодження та тестування;

– інтеграція з системами контролю версій, такими як Git;

– зручні інструменти для роботи з базами даних та серверами.

Недоліками використання PhpStorm є:

– потребує ліцензії для повного використання;

– може бути складним у використанні для новачків через велику кількість функцій.

Отже, PhpStorm було обрано як оптимальне середовище для розробки вебпорталу психологічної підтримки завдяки його потужному функціоналу та інтеграції з сучасними технологіями. Середовище підтримує JavaScript, Node.js та має вбудовані інструменти для дебагінгу, рефакторингу та автодоповнення коду.

3.2 Розробка програмного модуля реєстрації та авторизації користувача

Розробка модуля реєстрації та авторизації є ключовим та першим етапом для створення вебпорталу психологічної підтримки. Реєстрація здійснюється за допомогою введення користувачем необхідних даних, таких як ім’я, електронна пошта та пароль.

Важливим елементом є повторне введення пароля для перевірки його збігу з основним паролем, що знижує ймовірність помилок під час авторизації у майбутньому. Після того, як користувач підтвердив правильність введених даних інформація надсилається на сервер для подальшої обробки.

Лістинг функції обробки даних перед відправкою на сервер та безпосередньо їх відправка зображено на рисунку (рисунок 3.1).

```
19    const handleSubmit = async (e) :Promise<void> => {
20        e.preventDefault();
21        setErrorMessage( value: "");
22        setSuccessMessage( value: "");
23        if (!formData.email || !formData.name || !formData.password) {
24            setErrorMessage( value: "Заповніть усі поля");
25            return;
26        }
27        if (formData.password !== formData.confirmPassword) {
28            setErrorMessage( value: "Паролі не співпадають");
29            return;
30        }
31        try {
32            const response :Response = await fetch( input: 'http://localhost:3001/user-register', {method: 'POST'...});
43            const data = await response.json();
44
45            if (response.status === 404) setErrorMessage(data.message);
46            else if (response.ok) {...} else setErrorMessage( value: "Сталася помилка. Спробуйте ще раз.");
50        } catch (error) {...}
54    };
```

Рисунок 3.1 – Лістинг методу відправки даних для реєстрації на сервер

Сервер здійснює перевірку на унікальність електронної пошти. Якщо така електронна адреса вже існує у базі даних, користувач отримує відповідне повідомлення про помилку.

У разі успішної перевірки електронна пошта та хешований пароль користувача зберігаються у базі даних для подальшого використання. Блок-схему алгоритму роботи реєстрації зображено на рисунку 3.2

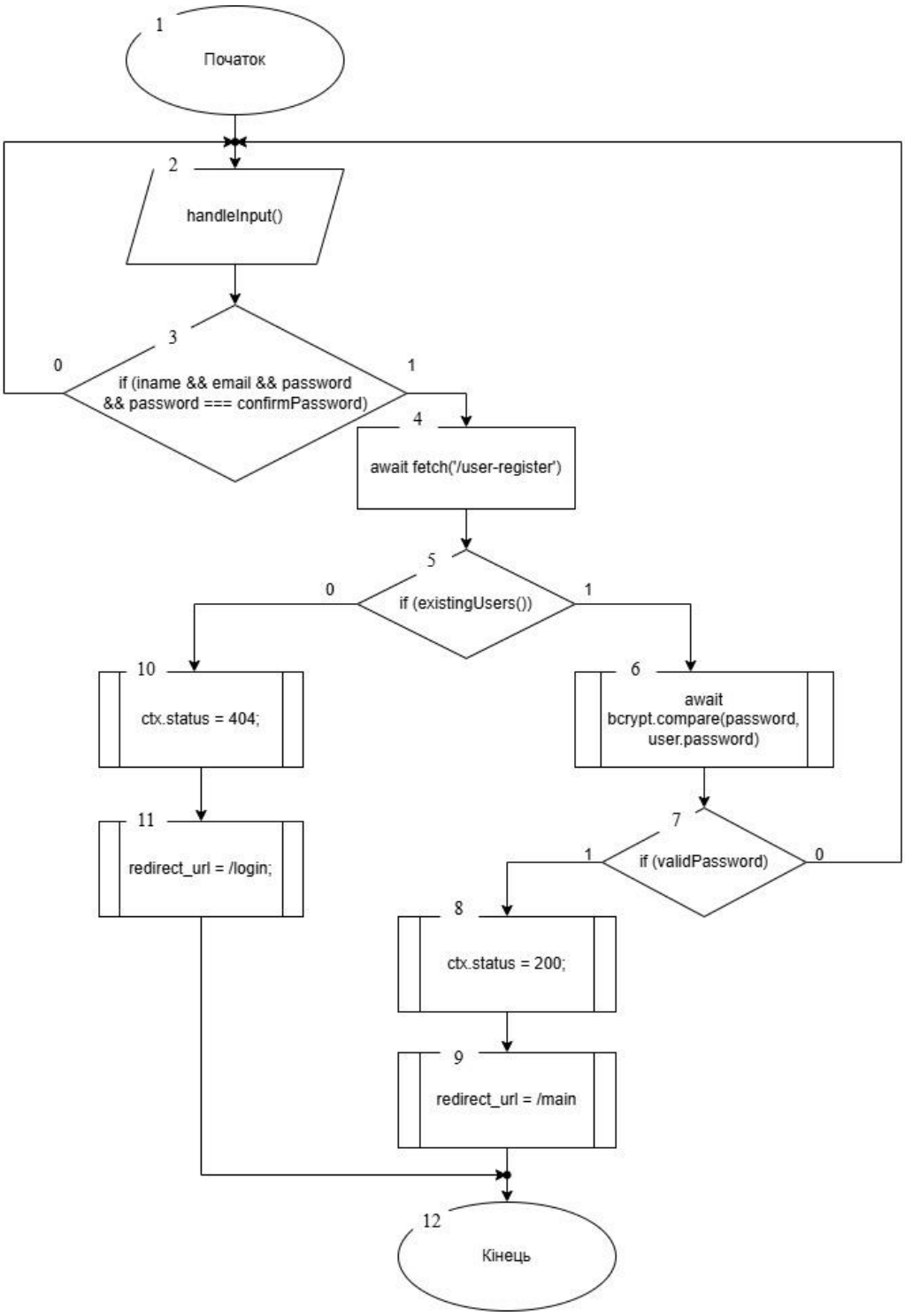

Рисунок 3.2 – Блок-схема алгоритму модуля реєстарції

Для забезпечення безпеки процесу використовується хешування пароля, що гарантує збереження даних у захищеному вигляді. Використання таких механізмів, як bcrypt, дозволяє шифрувати паролі, щоб вони зберігались у базі даних у вигляді хешів, а не відкритого тексту.

Завершальним етапом є створення облікового запису у базі даних, після

чого користувач отримує повідомлення про успішну реєстрацію та може увійти в систему для подальшого використання функціоналу порталу.

Отже, було розроблено програмний модуль реєстрації та авторизації користувача. У процесі розробки використано методи асинхронної обробки запитів, що дозволяє оптимізувати взаємодію з базою даних та підвищити ефективність роботи сервера.

3.3 Розробка програмного модуля розрахунку рейтингу спеціалістів

Наступним етапом розглянемо розробку модуля рейтингу спеціалістів. Метод обчислення рейтингу спеціалістів є важливою складовою системи вебпорталу психологічної підтримки, оскільки він забезпечує користувачів об'єктивною та детальною оцінкою кваліфікації спеціалістів. Рейтинг розраховується на основі сукупності критеріїв, що відображають ефективність, досвід, задоволеність клієнтів та інші аспекти, які є ключовими для вибору спеціаліста.

Для обчислення рейтингу було визначено наступні критерії:

1. Ефективність консультацій – оцінюється на основі середнього рейтингу консультацій.
2. Задоволеність користувачів – розраховується на основі середньої оцінки відгуків клієнтів.
3. Досвід роботи – враховується кількість років професійної діяльності спеціаліста.
4. Комунікація – визначається за спеціальною оцінкою в відгуках користувачів.
5. Пунктуальність – базується на оцінках пунктуальності спеціаліста під час консультацій.
6. Кількість клієнтів – відображає унікальну кількість клієнтів, що взаємодіяли зі спеціалістом.
7. Репутація – середнє значення між оцінками відгуків та консультацій.
8. Популярність у Telegram – враховується кількість активних

підписників спеціаліста.

Оскільки для розрахунку загального рейтингу спеціаліста було обрано метод вагових коефіцієнтів, розглянемо детальніше розробку цієї частини.

Вагові коефіцієнти у рейтинговій системі використовуються для визначення важливості кожного критерію оцінки, що впливає на підсумковий рейтинг. Вони дозволяють надавати пріоритет тим аспектам, які є більш значущими для користувачів або бізнес-логіки системи. У цьому випадку вагові коефіцієнти забезпечують гнучкість і адаптивність рейтингової моделі, дозволяючи коректно враховувати специфіку різних спеціалістів.

Розрахунок рейтингу з використання вагових коефіцієнтів обчислюється формулою 3.1:

$$R = \sum_i C_{(i)} \times W_{(i)}, \tag{3.1}$$

де R – загальний рейтинг спеціаліста;

$C(i)$ – значення критерію i;

$W(i)$ – ваговий коефіцієнт критерію i.

Розподілення кожного пункту рейтингу до їх вагових коефіцієнтів зображено на рисунку 3.3.

```
3   const weights :{…} = {
4       effectiveness: 0.3,
5       userSatisfaction: 0.25,
6       experience: 0.1,
7       communication: 0.3,
8       punctuality: 0.2,
9       clientCount: 0.2,
10      reputation: 0.3,
11      telegramSubscribers: 0.1
12  };
```

Рисунок 3.3 – Розподілення вагових коефіцієнтів

Реалізація розрахунку рейтингу спеціаліста, з використанням формули вагових коефіцієнтів зображено на рисунку 3.4.

```
const calculateRating = (
    { specialist :{}  = {}, reviews :any[]  = [], appointments :any[]  = [], telegramSubscribers :any[]  = [] }
) :{details: {...}, overallRating: string}  => {
    let details :{...}  = {
        effectiveness: calculateEffectiveness(appointments),
        userSatisfaction: calculateUserSatisfaction(reviews),
        experience: calculateExperienceScore(specialist),
        communication: calculateCommunication(appointments),
        punctuality: calculatePunctuality(appointments),
        clientCount: calculateClientCount(appointments),
        reputation: calculateReputation(reviews, appointments),
        telegramSubscribers: calculateTelegramSubscribers(telegramSubscribers)
    };
    details = roundDetails(details);
    const weightedSum :number  = Object.keys(details).reduce(
        (sum :number , key :string ) => sum + details[key] * normalizedWeights[key], 0
    );
```

Рисунок 3.4 – Лістинг використання методу вагових коефіцієнтів

Не важко помітити, що сума всіх вагових коефіцієнтів перевищує одиницю. Це може призвести до некоректний розподіл значень, складність інтерпретації та нестабільність результатів. Адже зміна однієї ваги може суттєво змінити підсумковий рейтинг, порушуючи баланс між критеріями.

Тому є необхідною нормалізація вагових коефіцієнтів – процес масштабування їх значень, щоб сума дорівнювала одиниці. Обчислюється за формулою 3.2

$$W_{normalized\ (i)} = \frac{W_{(i)}}{\sum_j W_{(j)}}, \tag{3.2}$$

де $W_{normalized\ (i)}$ – нормалізована вага критерію *i*;

W(i) – початкова вага критерію *i*;

$\sum_j W_{(j)}$ – сума всіх початкових ваг.

Реалізація функції нормалізації вагових коефіцієнтів зображена на рис. 3.5.

```
const normalizeWeights = (weights) :{}  => {
    const totalWeight = Object.values(weights).reduce((sum, w) => sum + w, 0);
    return Object.keys(weights).reduce((normalized :{} , key :string ) :{}  => {
        normalized[key] = weights[key] / totalWeight;
        return normalized;
    }, {});
};
```

Рисунок 3.5 – Лістинг функції нормалізації вагових коефіцієнтів

Для загального розуміння роботи модуля розрахунку рейтингу спеціалістів, створено діаграму послідовностей (рисунок 3.6).

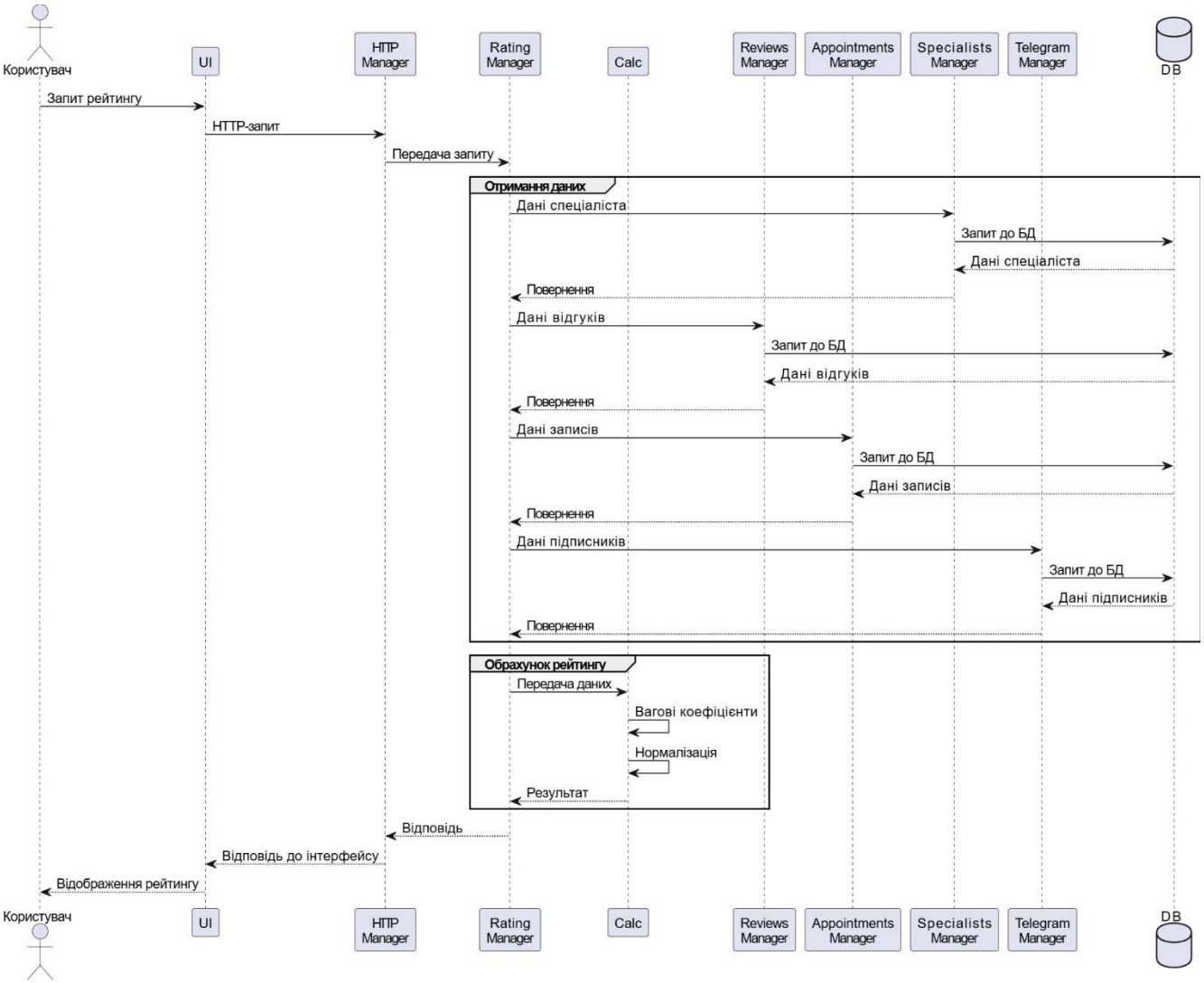

Рисунок 3.6 – Діаграма послідовностей модуля рейтингу спеціалістів

Отже, було розроблено програмний модуль розрахунку рейтингу спеціалістів. У процесі розробки використано методи вагових коефіцієнтів та їх нормалізації, що дозволяє зберегти баланс між впливами різних критеріїв, забезпечуючи точність та об'єктивність рейтингу.

3.4 Розробка модуля інтеграції телеграм-боту

Не менш важливим етапом створення вебпорталу психологічної підтримки є розробка модуля інтеграції телеграм-боту. Саме у випадку вебпорталу

психологічної підтримки, Telegram-бот відіграє роль швидкого каналу комунікації, що:

1. Забезпечує доступ користувачів до функціональності порталу.
2. Підвищує зручність взаємодії завдяки сповіщенням і автоматизації процесів.
3. Забезпечує персоналізовану взаємодію зі спеціалістами.

Розглянемо діаграму розгортання для модуля інтеграції телеграм-боту до вебпорталу психологічної підтримки (рисунок 3.7).

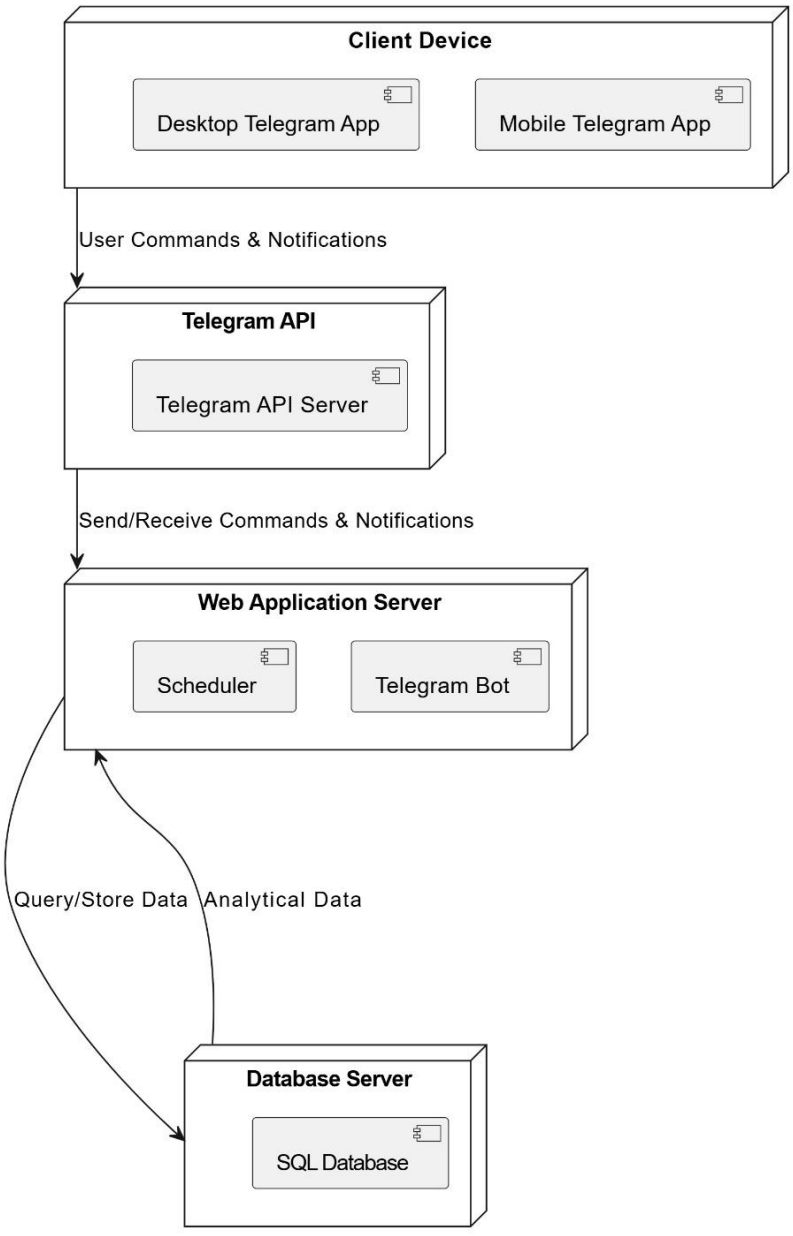

Рисунок 3.7 – Діаграма розгортання інтеграції телеграм-боту

Діаграма ілюструє обмін даними між усіма компонентами, акцентуючи на централізованій обробці команд Telegram-ботом і використанні бази даних для забезпечення актуальності інформації. Складається діаграма розгортання з наступних частин:

1. Клієнтські пристрої (мобільний або вебдодаток Telegram) використовуються користувачами для надсилання команд і отримання повідомлень.

2. Telegram API виступає посередником між клієнтськими пристроями та сервером Telegram-бота, забезпечуючи обробку команд і передачу сповіщень.

3. Сервер вебзастосунку, який містить Telegram-бота і планувальник завдань, обробляє команди користувачів, надсилає заплановані сповіщення та взаємодіє з базою даних.

4. Сервер бази даних використовується для зберігання та отримання даних про користувачів, спеціалістів, підписки та консультації, забезпечуючи основу для аналітичних і операційних запитів.

Реалізація обробки запитів з Telegram API зображено на рисунку 3.8.

```
14  const bot :TelegramBot = new TelegramBot(token, { polling: true });
15
16  bot.setMyCommands( commands: [
17      { command: '/mysubscriptions', description: 'Переглянути свої підписки' },
18      { command: '/appointments', description: 'Переглянути консультації' },
19      { command: '/statistic', description: 'Переглянути статистику за місяць' },
20  ]);
21
22  › bot.onText( regexp: /\/start(?: (.+))?/, callback: async (msg, match) :Promise<...> => {...});
40  › bot.onText( regexp: /\/mysubscriptions/, callback: async (msg) :Promise<...> => {...});
63  › bot.onText( regexp: /\/appointments/, callback: async (msg) :Promise<void> => {...});
79  › bot.onText( regexp: /\/statistic/, callback: async (msg) :Promise<void> => {...});
95  › bot.on( event: 'callback_query', listener: async (callbackQuery) :Promise<...> => {...});
```

Рисунок 3.8 – Лістинг обробки запитів з Telegram API

Проте крім опрацювання виклику команд користувачем модуль взаємодіє з методом планування подій на основі Node Schedule, для автоматичного надсилання сповіщень користувачам у встановлені часові проміжки. Наприклад, відправка посилання на зустріч за годину до події.

Для забезпечення детального розуміння роботи Telegram-боту, його функціоналу та обробки викликів у системі, була створена діаграма послідовностей (рисунок 3.9).

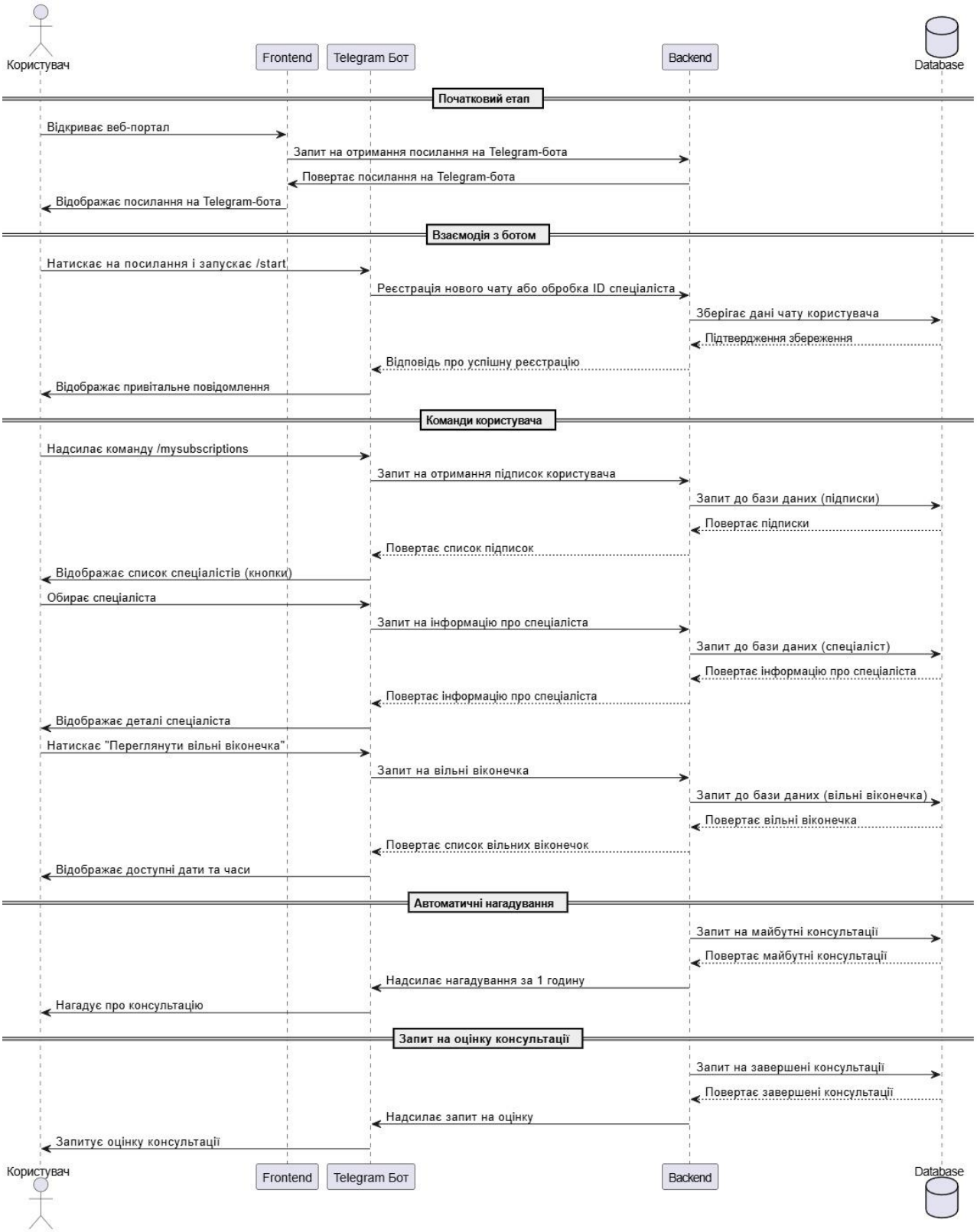

Рисунок 3.9 – Діаграма послідовностей роботи телеграм-боту

Отже, було розроблено модуль інтеграції Telegram-боту до вебпорталу психологічної підтримки, що забезпечує ефективний, швидкий і персоналізований канал взаємодії між користувачами та системою. Розроблений модуль Telegram-боту не лише розширює функціональні можливості порталу, але й підвищує зручність користування завдяки автоматизації сповіщень, обробці запитів у реальному часі та централізованому управлінню даними.

3.5 Висновки

У цьому розділі було представлено процес розробки основних програмних модулів вебпорталу психологічної підтримки, які забезпечують його ключову функціональність та інтеграцію.

Проведено детальний аналіз і обґрунтування вибору технологій, де Node.js і React були обрані як оптимальні інструменти для бекенд і фронтенд частини відповідно. Важливою складовою стала розробка модуля реєстрації та авторизації, який гарантує безпечний і надійний доступ користувачів до системи.

Окрему увагу приділено модулю розрахунку рейтингу спеціалістів. Запропонований підхід із використанням вагових коефіцієнтів та нормалізації дозволяє забезпечити об’єктивність та збалансованість оцінки, враховуючи різні аспекти роботи спеціалістів. Інтеграція Telegram-бота у поєднанні з планувальником задач Node Schedule розширила функціональність порталу, дозволивши автоматизувати сповіщення, покращити комунікацію між користувачами та спеціалістами, а також надати зручний інтерфейс для взаємодії.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення є невід'ємною складовою процесу його розробки, спрямованою на виявлення помилок, перевірку відповідності функціональності вимогам та забезпечення надійності системи. Це дозволяє не лише покращити якість продукту, але й підвищити довіру користувачів до нього. У сучасній розробці широко застосовуються різні підходи до тестування, серед яких варто виокремити методи «чорної скриньки», «білої скриньки» та «сірої скриньки» [37].

1. Метод «чорної скриньки». Метод «чорної скриньки» базується на тестуванні функціональності програми без знання її внутрішньої реалізації. Основна увага приділяється перевірці вхідних і вихідних даних, відповідності системи функціональним вимогам, а також інтеграції з іншими компонентами. Цей метод ефективний для перевірки зовнішньої поведінки системи.

Переваги методу «чорної скриньки»:

- не потребує знання коду програми, що робить його доступним для тестувальників без технічної підготовки;
- орієнтований на перевірку функціональності, що важливо з точки зору кінцевого користувача;
- дає змогу виявляти невідповідності вимогам.

Недоліки методу «чорної скриньки»:

- обмежений у виявленні внутрішніх помилок;
- залежність від точності тестових сценаріїв, створених на основі вимог.

2. Метод «білої скриньки». Метод «білої скриньки» передбачає доступ до вихідного коду та внутрішньої структури програми, що дозволяє глибше аналізувати її роботу. Він застосовується для перевірки логіки, алгоритмів, критичних шляхів виконання та якості коду.

Переваги методу «білої скриньки»:

- детальний аналіз внутрішньої логіки програми;
- можливість виявлення складних помилок, пов’язаних із реалізацією алгоритмів;
- забезпечення високого покриття коду тестами.

Недоліки методу «білої скриньки»:

- вимагає значних знань про структуру програми, що збільшує тривалість тестування;
- не завжди охоплює зовнішні аспекти функціональності програми.

3. Метод «сірої скриньки». Метод «сірої скриньки» є комбінацією попередніх двох підходів. Він передбачає частковий доступ до внутрішньої структури програми при збереженні фокусу на її зовнішній поведінці. Такий підхід ефективний для тестування складних інтеграційних сценаріїв.

Переваги методу «сірої скриньки»:

- поєднує переваги методів «чорної скриньки» та «білої скриньки».
- дозволяє тестувати як внутрішню логіку, так і зовнішню функціональність.
- підходить для тестування модулів із високою залежністю від інших компонентів.

Недоліки методу «сірої скриньки»:

- вимагає додаткового часу на вивчення часткової реалізації програми.
- може бути складним у виконанні для великих систем.

4. Автоматизоване тестування. Автоматизоване тестування використовує спеціальні інструменти для виконання тестів. Воно особливо корисне для регресійного, навантажувального та модульного тестування.

Переваги використання автоматизованого тестування:

- швидкість виконання тестів, особливо при повторюваних завданнях;
- підвищення точності завдяки зменшенню впливу людського фактора;
- легкість масштабування для великих проєктів.

Недоліки використання автоматизованого тестування:

- початкова складність і витрати на впровадження інструментів автоматизації;
- необхідність регулярного оновлення тестових сценаріїв при зміні функціональності.

5. Навантажувальне тестування. Навантажувальне тестування перевіряє продуктивність системи під високими навантаженнями, симулюючи велику кількість одночасних запитів.

Перевагами такого підходу є оцінка стійкості системи в умовах стресу та виявлення вузьких місць у продуктивності. А недоліком є високі вимоги до апаратних ресурсів та потреба в спеціалізованих інструментах.

Оцінюючи ці підходи, можна зробити висновок, що для тестування вебсистеми психологічної підтримки було обрано метод «білої скриньки» як основний, оскільки він дозволяє досягти високого покриття коду та забезпечує надійність критичних шляхів системи. Додатково використовуються елементи «сірої скриньки» для перевірки взаємодії між модулями, таких як інтеграція серверної частини з клієнтським інтерфейсом або Telegram-ботом.

Обраний підхід дозволяє:

- виявляти логічні помилки на ранніх етапах розробки;
- забезпечувати відповідність системи вимогам і сценаріям використання;
- забезпечувати стабільну інтеграцію компонентів та модулів.

Комбінація методів тестування є оптимальним рішенням, що дозволяє забезпечити якість програмного забезпечення, адаптуючи підхід до потреб проєкту.

4.2 Тестування програмного продукту

При тестуванні вебпорталу психологічної підтримки методом «білої скриньки» особливу увагу слід звернути на перевірку внутрішньої логіки програми, коректність алгоритмів та структур даних, які забезпечують роботу системи. Цей підхід дозволяє аналізувати вихідний код і тестувати критичні

ділянки, де помилки можуть призвести до серйозних збоїв у роботі системи. Тестування за цим методом забезпечує високий рівень покриття коду, дозволяючи виявляти складні помилки, які важко виявити при зовнішньому тестуванні.

Одним із перших кроків тестування вебпорталу психологічної підтримки є перевірка модулів реєстрації та входу в систему. Ці функції є базовими для забезпечення доступу користувачів до системи, тому їхня надійність має вирішальне значення.

Взаємодія користувача починається з сторінки входу, де він може ввести пошту та пароль попередньо зареєстрованих на цьому вебпорталі. Якщо ж такого користувача не знайдено, виникає помилка та прохання зареєструватись у системі (рисунок 4.1).

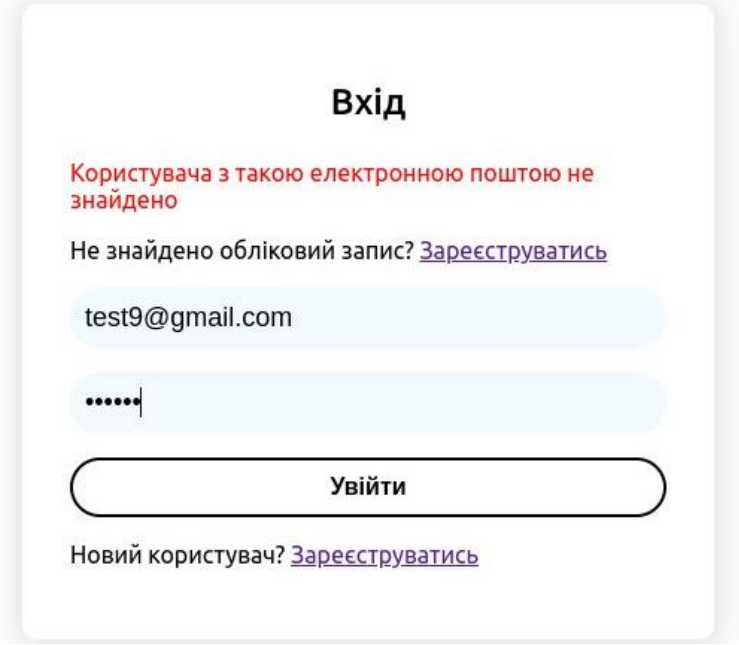

Рисунок 4.1 – Спроба входу неіснуючим користувачем

Тоді наступним кроком спробуємо зареєструвати користувача у системі. Для реєстрації необхідно ввести ім’я, електронну пошту та пароль. При введенні некоректних чи вже існуючих даних користувача виникає повідомлення з текстом помилки та форма не відправляється.

Види помилок при реєстрації можна побачити на рисунку 4.2.

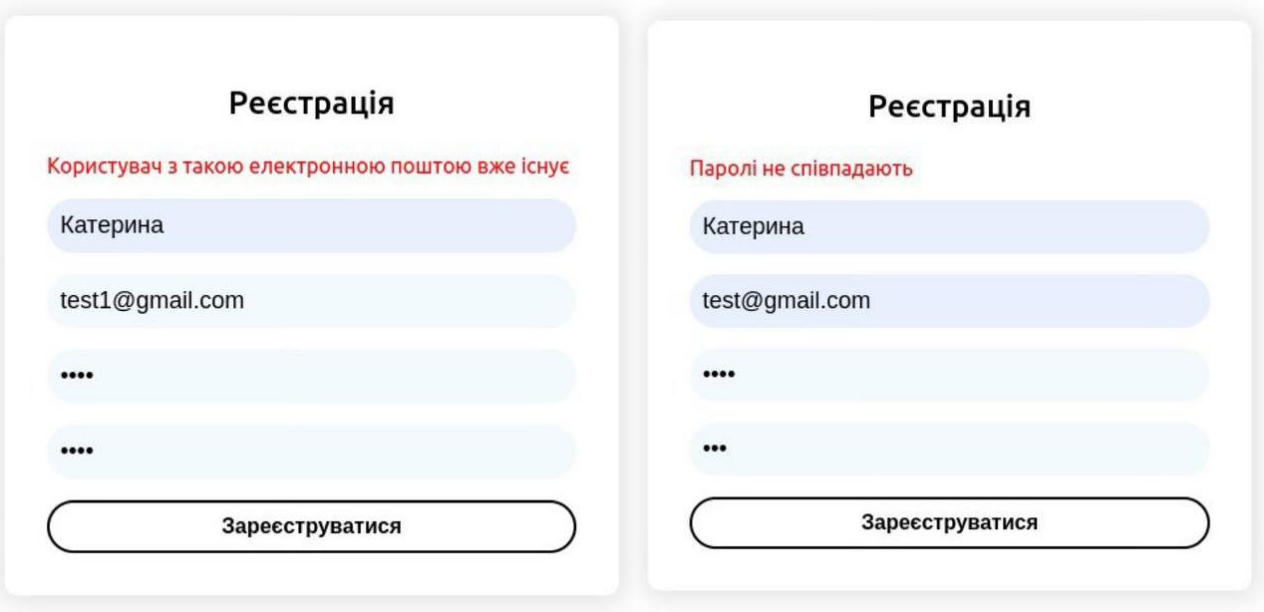

Рисунок 4.2 – Види помилок при реєстрації

Якщо ж всі поля коректно заповнені та такого користувача раніше не було зареєстровано в системі, то реєстрація відбувається успішно. Введений пароль шифрується, інформація про нового користувача додається в базу даних, а його перенаправляє на сторінку входу (рисунок 4.3).

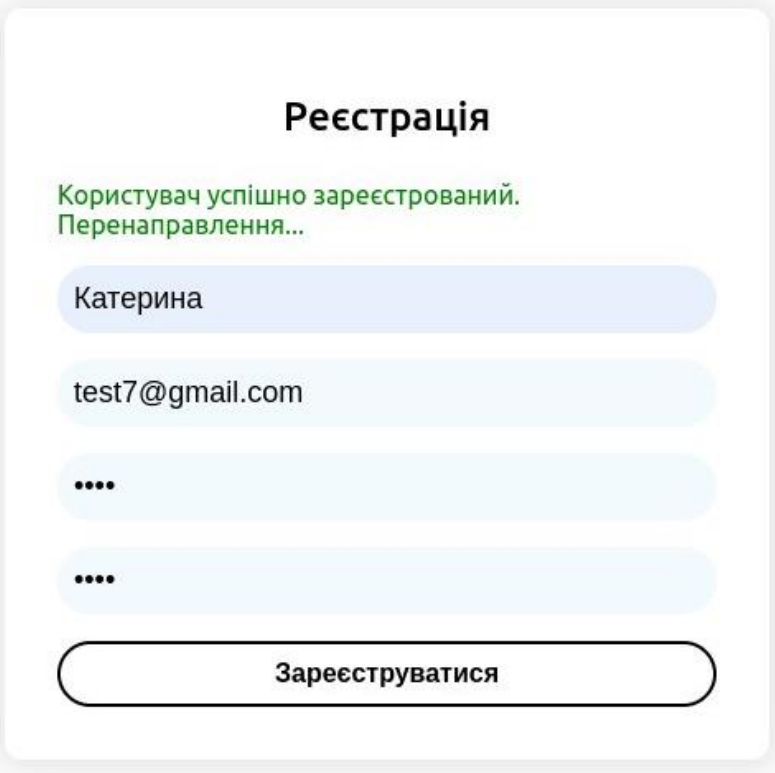

Рисунок 4.3 – Успішна реєстрація користувача в системі

Після успішного входу до вебпорталу психологічної підтримку відкривається головна сторінка, на якій зображено меню, точки входу до тестування психологічного стану та курсів самодопомоги, а також блоки з спеціалістами з психологічного здоров’я (рисунок 4.4).

Кнопка «Зв’язатись» під кожним спеціалістом відкриває модальне вікно із заявкою на консультацію до обраного спеціаліста.

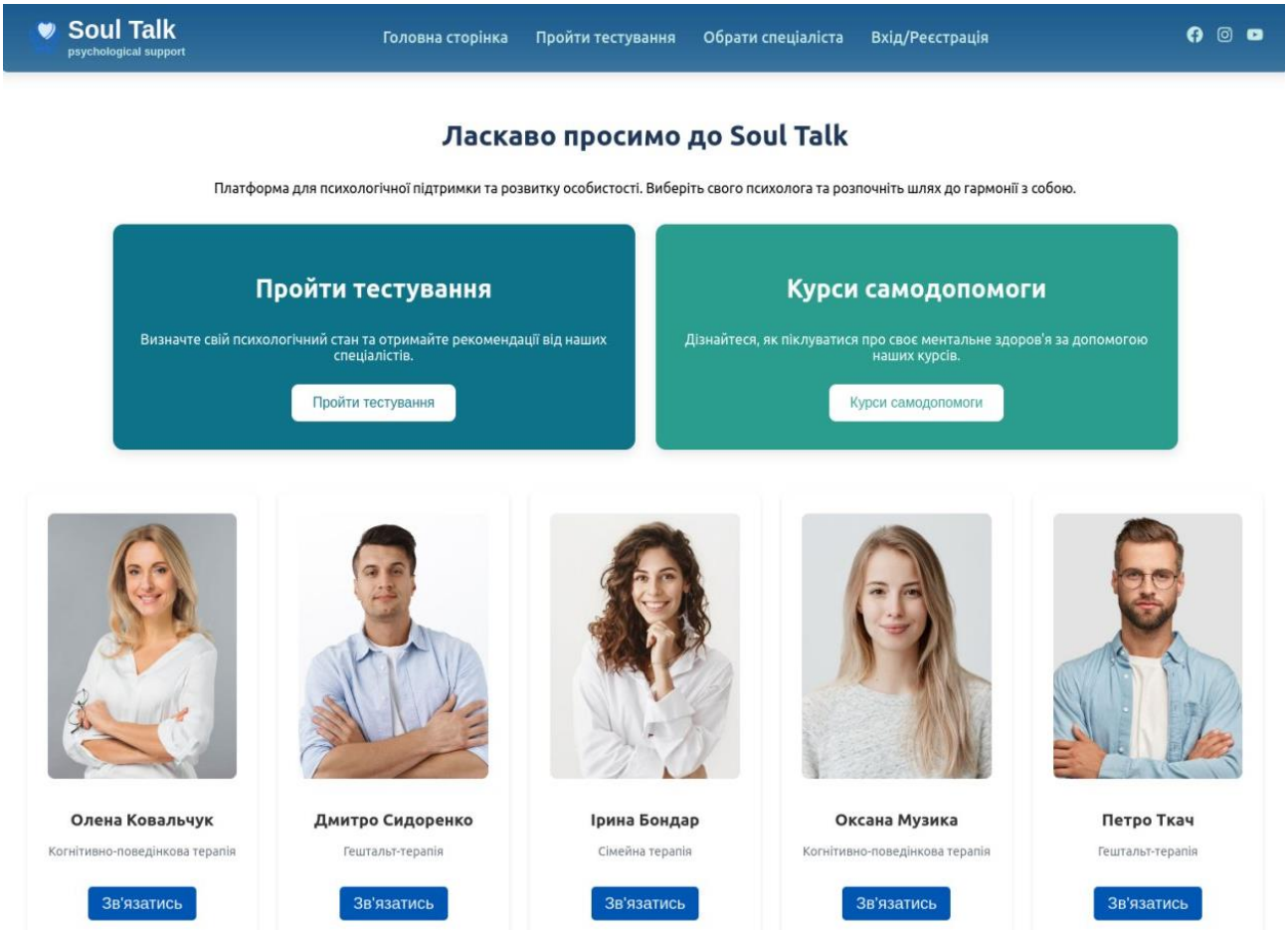

Рисунок 4.4 – Головна сторінка вебпорталу психологічної підтримки

Розпочнемо ознайомлення з головною сторінкою з верхніх блоків «Пройти тестування» та «Курси самодопомоги», оскільки вони привертають найбільше уваги. Натиснувши на кнопку в блоці «Курси самодопомоги», користувача перенаправляє до нової сторінки, де розташовані блоки з найпопулярнішими проблемами. Далі можна обрати розділ, наприклад «Панічна атака» та коротко прочитати, як попередньо можна допомогти собі у такому стані (рисунок 4.5).

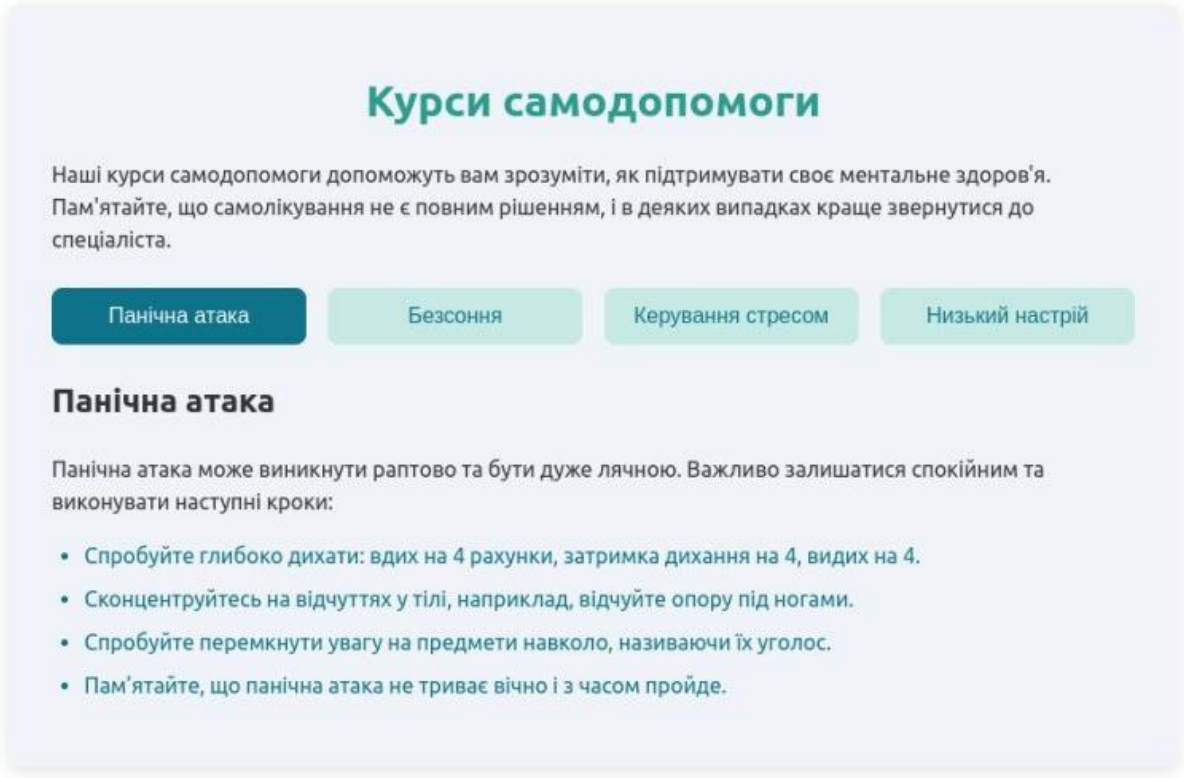

Рисунок 4.5 – Сторінка з курсами самодопомоги

Наступним кроком перейдемо до блоку «пройти тестування», який перенаправляє на сторінку з тестом для визначення психологічного стану користувача (рисунок 4.6).

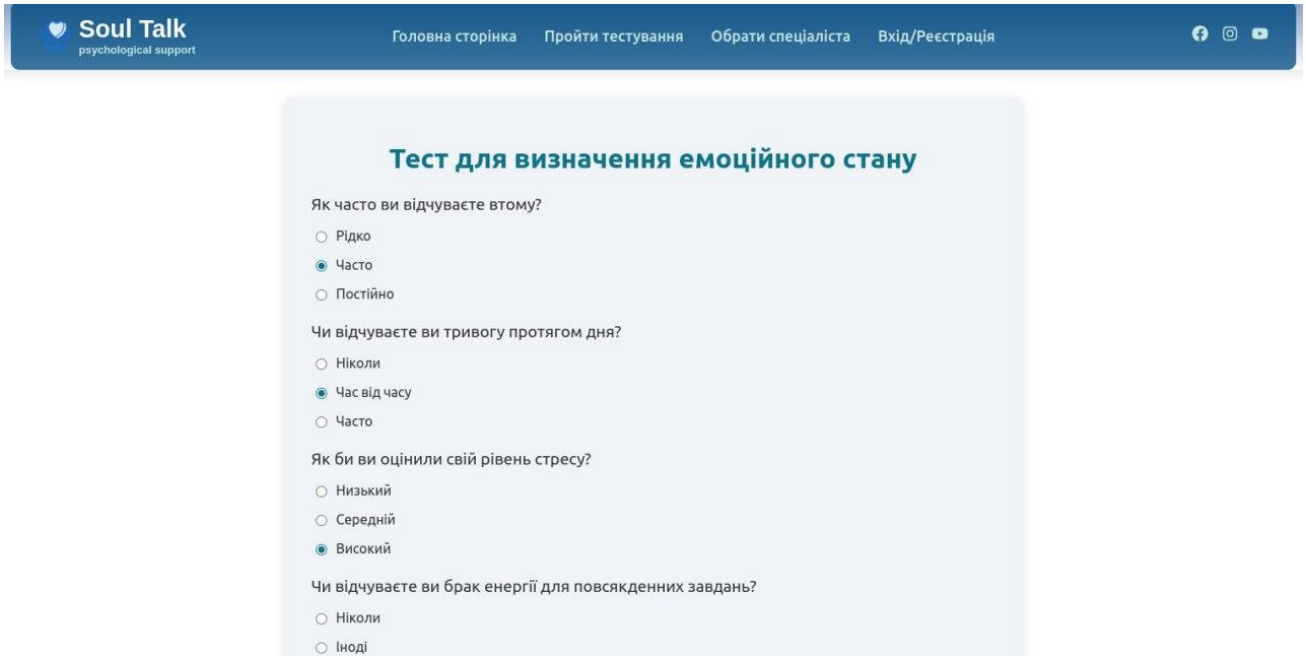

Рисунок 4.6 – Сторінка тестування

Після проходження тесту та відправлення форми можна побачити результат емоційного стану, а також рекомендацію спеціалістів, які можуть допомогти із запитами користувача. На рисунку 4.7 зображено два варіанти проходження тесту.

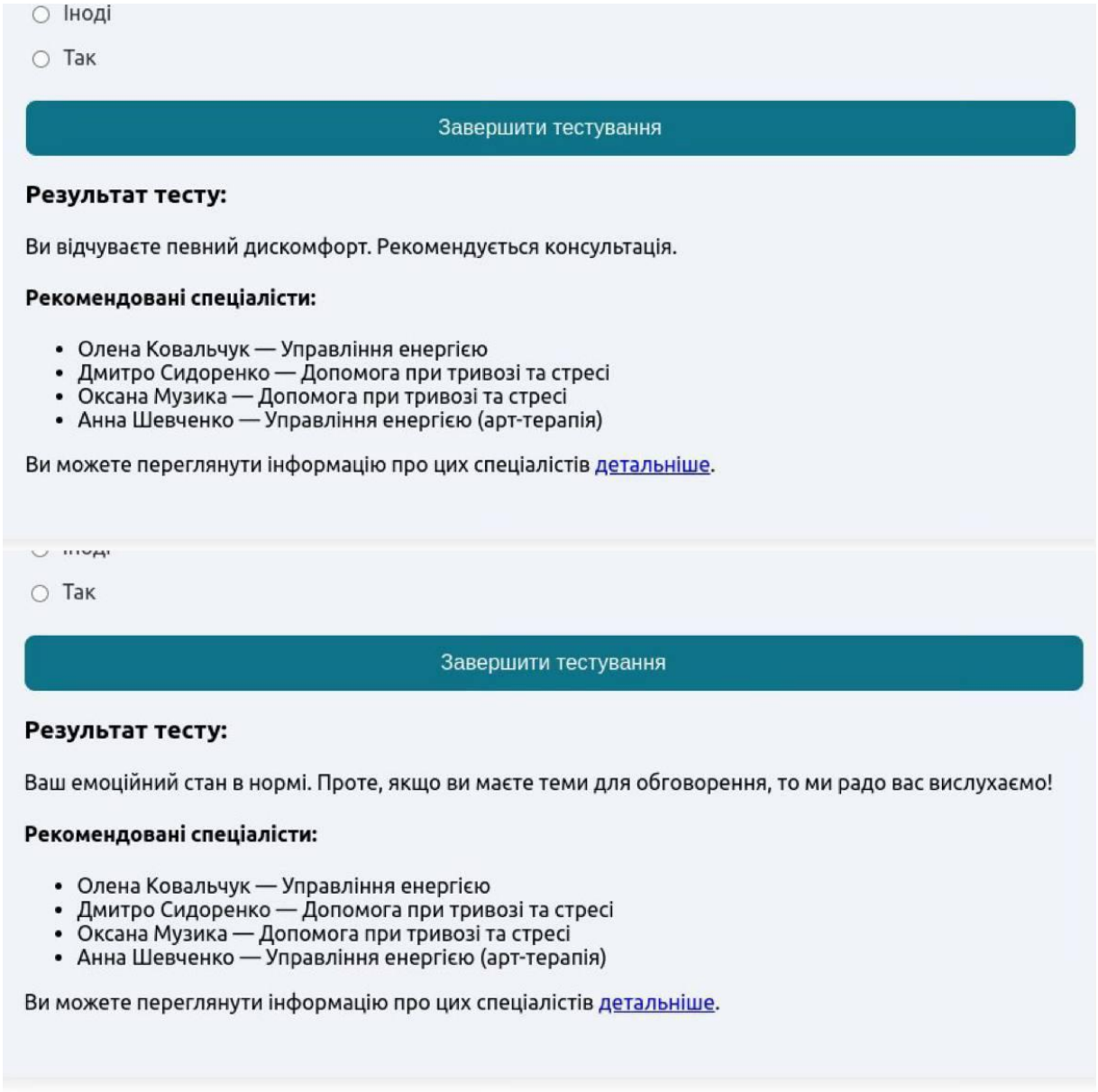

Рисунок 4.7 – Результати тесту з рекомендованими спеціалістами

Посилання «Детальніше» веде користувача на каталог спеціалістів, де відображаються лише ті, які підібрані на основі попереднього тестування. Якщо ж тестування не допомогло підібрати жодного конкретного спеціаліста, то посилання веде на загальний каталог зі всіма спеціалістами (рисунок 4.8).

Рисунок 4.8 – Результати тесту з пустою видачею

Загальний каталог спеціалістів містить перелік усіх спеціалістів, коротку інформацію та відгуки про кожного з них. Також є можливість скористатись фільтрами, для простішого пошуку за доступними критеріями (рисунок 4.9).

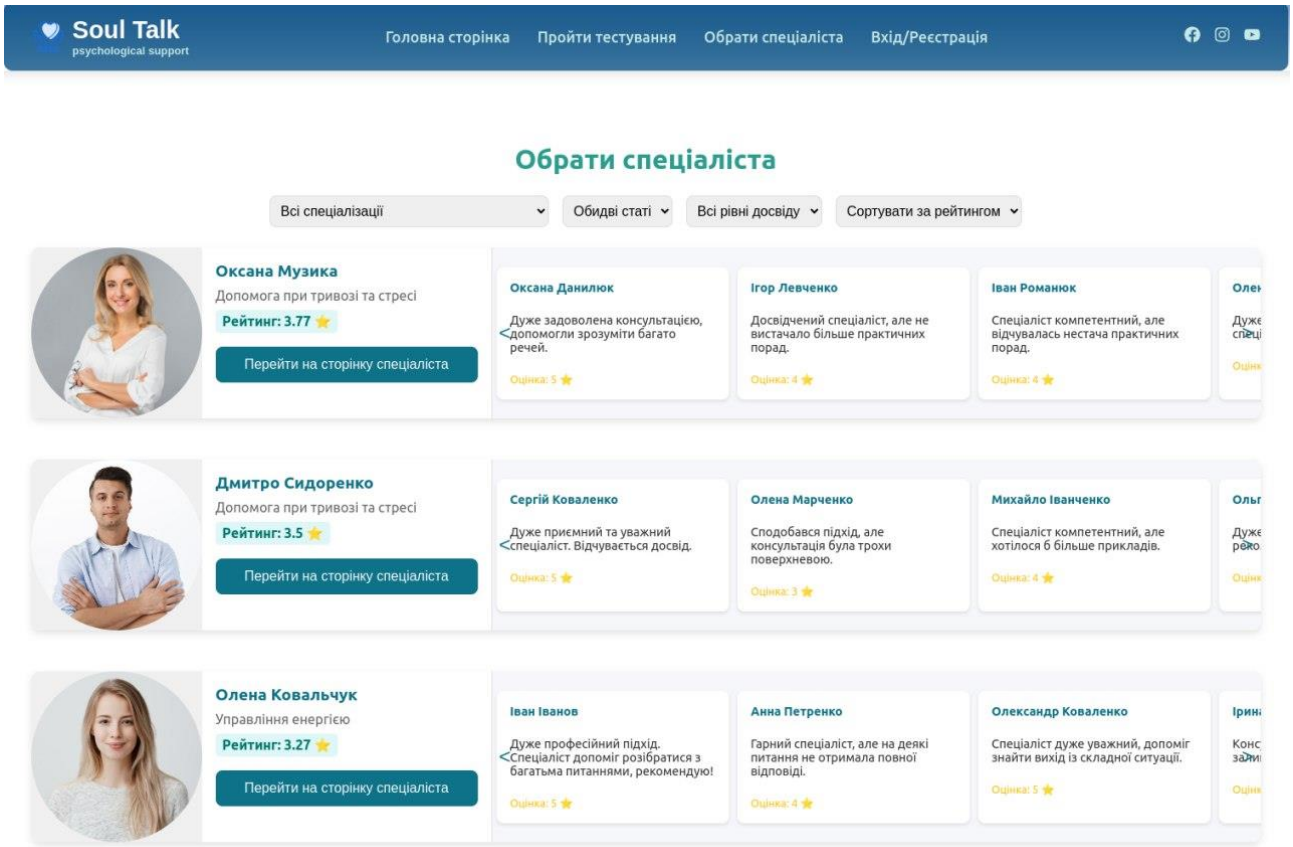

Рисунок 4.9 – Каталог спеціалістів без обраних фільтрів

Відфільтрувати спеціалістів можна за їх спеціалізацією, статтю або ж досвідом роботи. Крім того можливе сортування за рейтингом спеціаліста, або за

алфавітом. Оберемо фільтр спеціалістів за спеціалізацією «Управління енергією», жіночою статтю та досвідом роботи від трьох років.

Результат фільтрування спеціалістів зображено на рисунку 4.10.

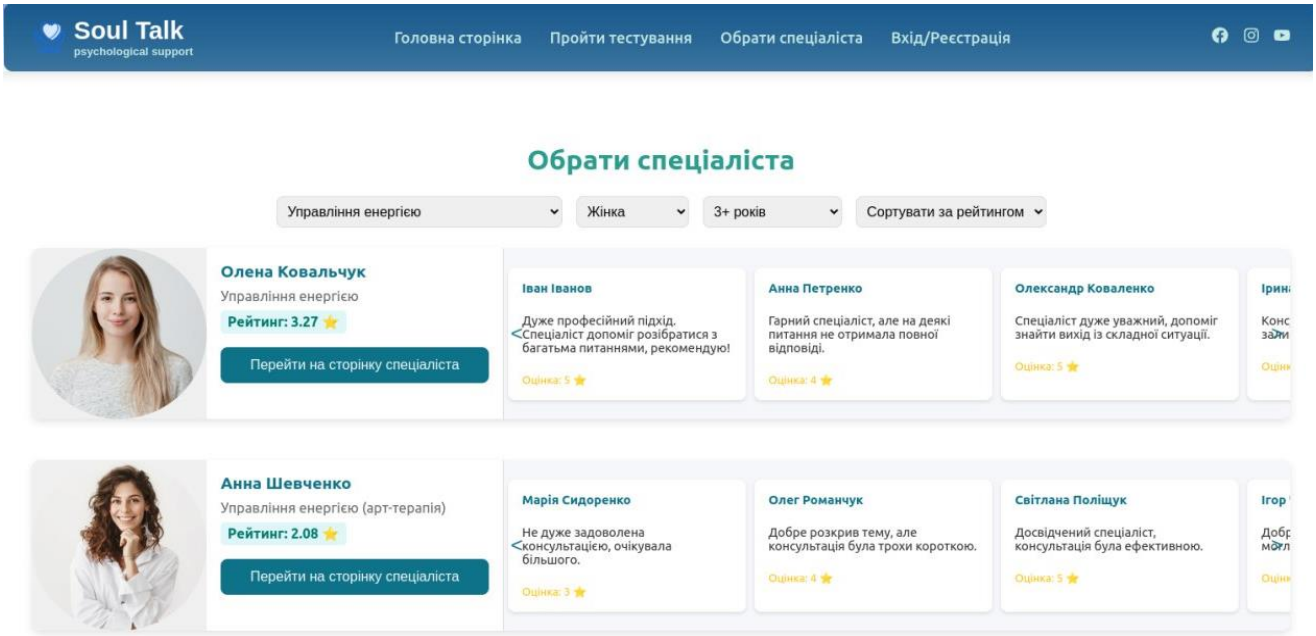

Рисунок 4.10 – Каталог спеціалістів з обраними фільтрами

Оскільки обрані параметри залишили в списку два спеціаліста, виберемо того, у кого кращий рейтинг. Далі натискаємо на «Перейти на сторінку спеціаліста» і переходимо на сторінку конкретного спеціаліста (рисунок 4.11).

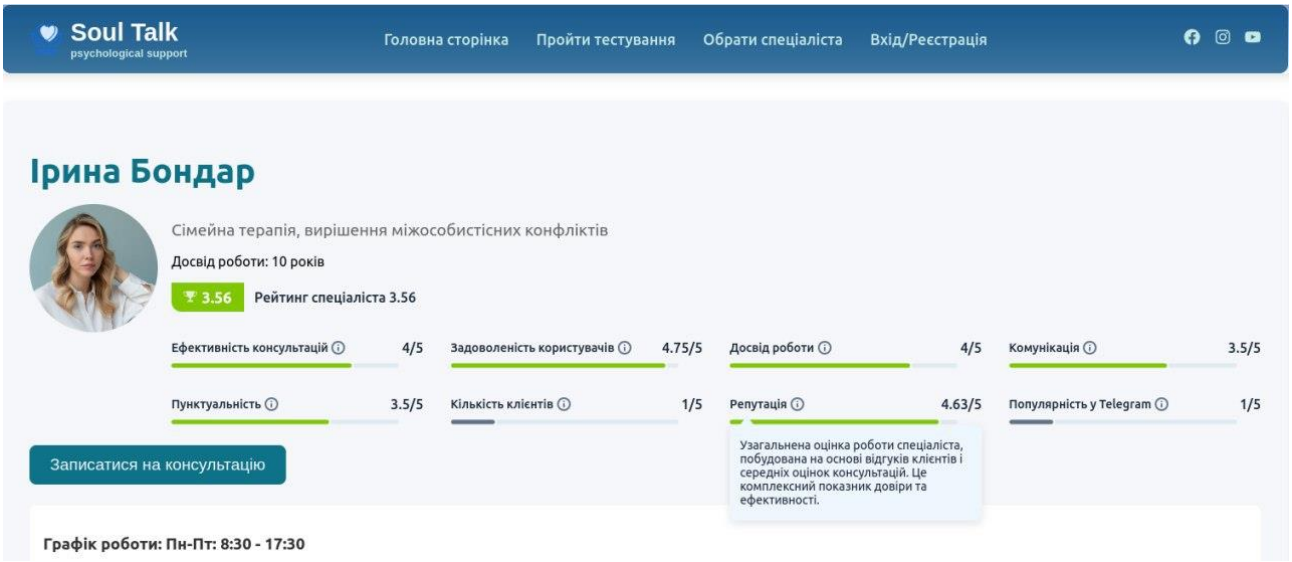

Рисунок 4.11 – Сторінка конкретного спеціаліста

Перше, що привертає увагу це рейтинг спеціаліста. Якщо рейтинг менший ніж три із п’яти, він зафарбовується в сірий колір. Якщо ж більше ніж три із п’яти – в зелений. Це дозволяє користувачу простіше візуально сприймати «репутацію» спеціаліста, а спеціалістів в свою чергу надихає покращувати свої бали (рисунок 4.12).

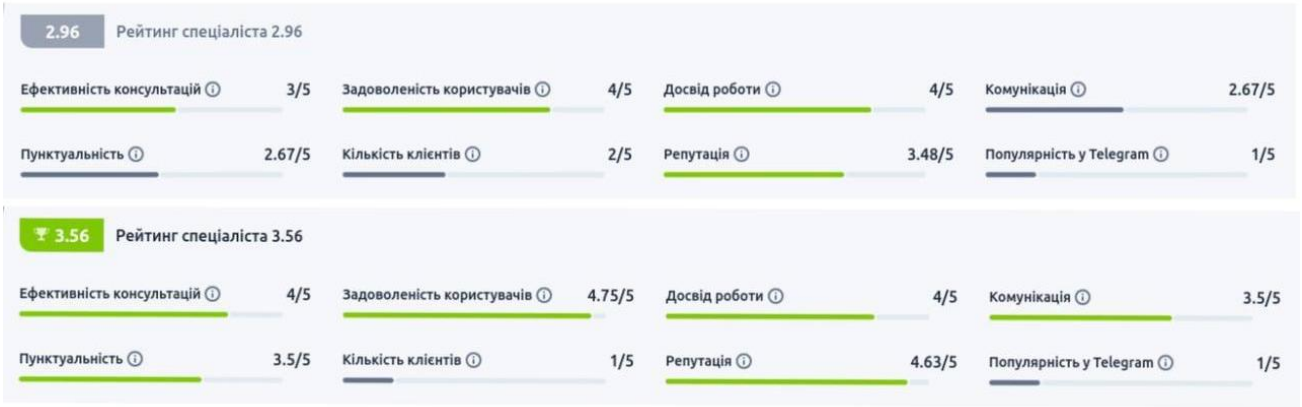

Рисунок 4.12 – Стани рейтингу спеціаліста

Наступним розглянемо інтерактивний календар зайнятості спеціаліста. На якому в залежності від графіку роботи, а також зайнятих віконечок на консультацію на календарі зафарбовуються клітинки (рисунок 4.13).

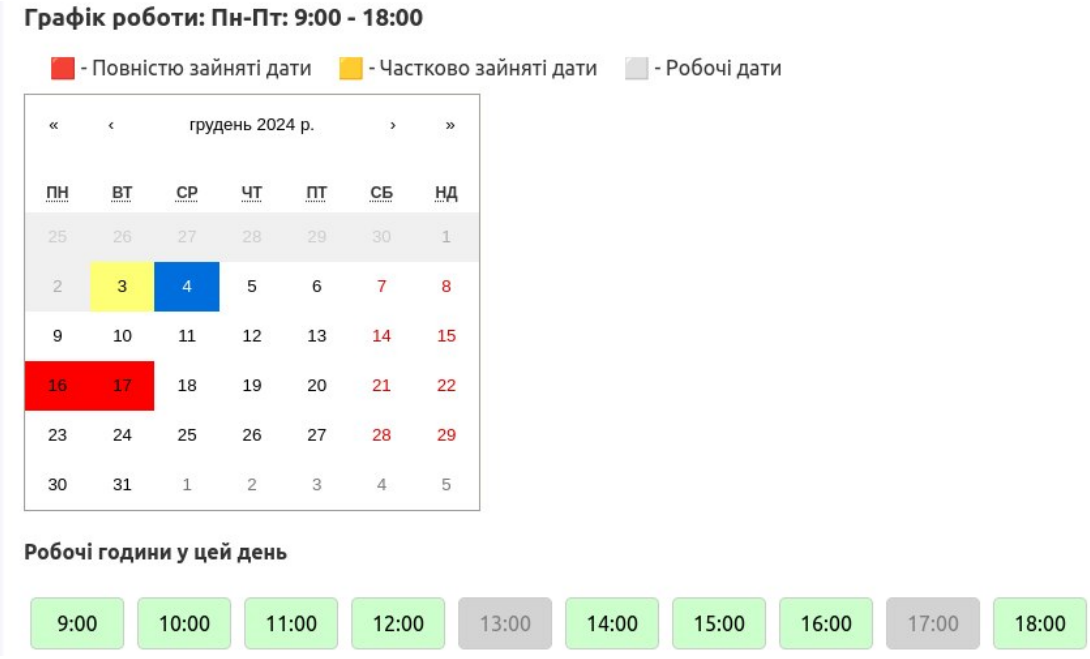

Рисунок 4.13 – Інтерактивний календар

Такий ж календар міститься і у заявці на консультацію, щоб користувачі могли зручно обрати час для зустрічі. Крім інтерактивного календаря заявка містить поля для імені, пошти та коментаря, але обов’язковими для заповнення є лише пошта та час консультації. Якщо ж одне з обов’язкових полів відсутнє, кнопка відправки даних є неактивною (рисунок 4.14).

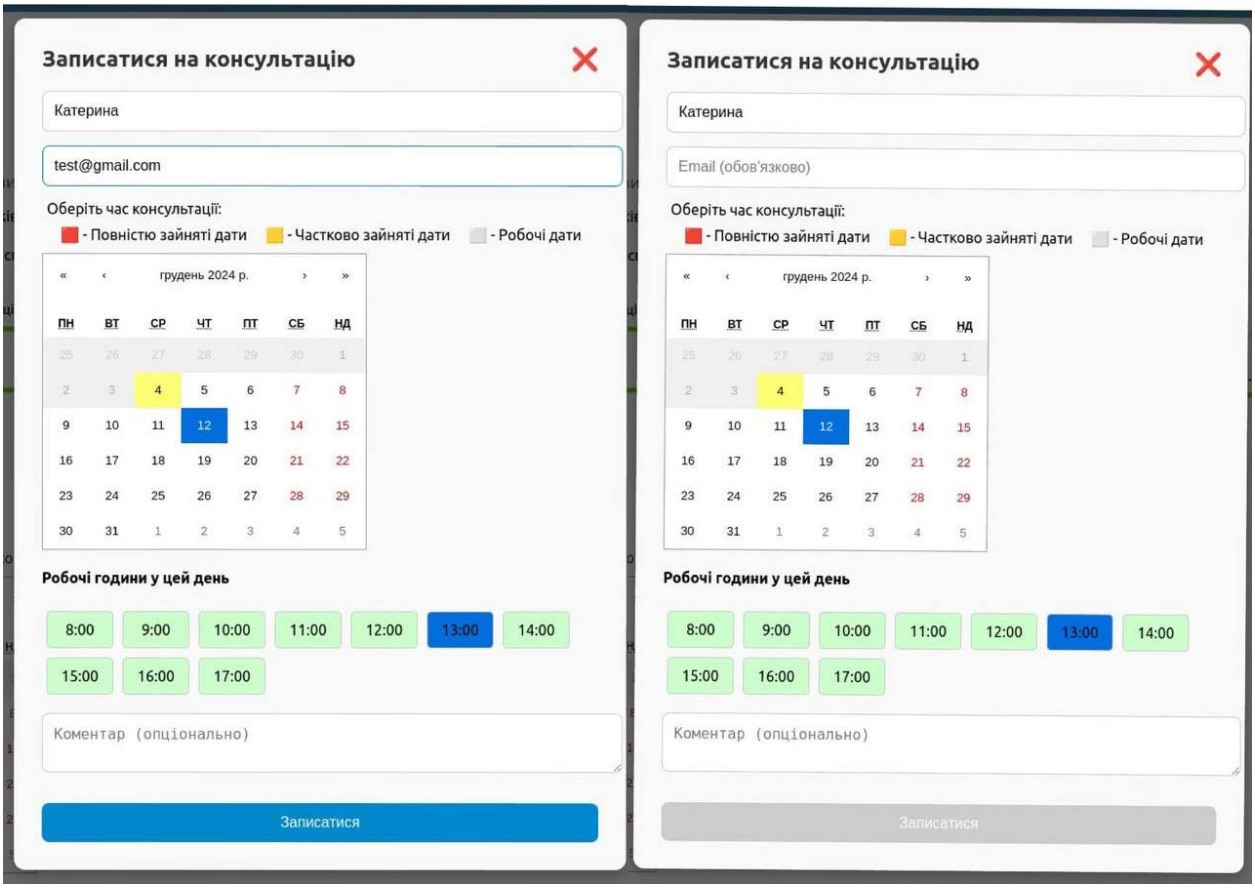

Рисунок 4.14 – Стани заявки на консультацію

Також сторінка спеціаліста містить точку входу, для підписки на Телеграм-бот (рисунок 4.15).

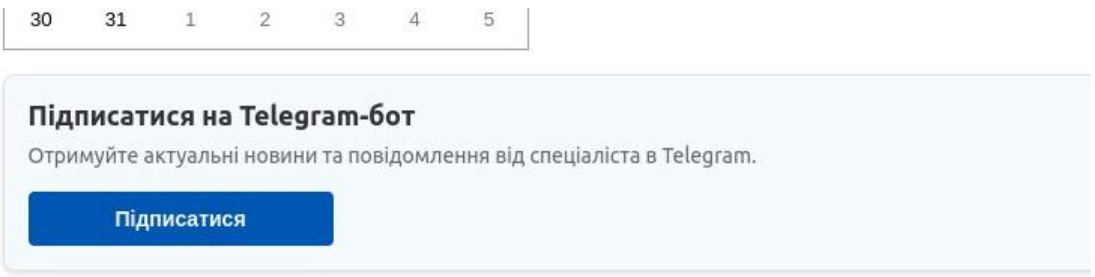

Рисунок 4.15 – Кнопка підписки на Телеграм-бот

При кліку на кнопку «Підписатися» користувача перенаправляє на посилання https://t.me/ozerovaSoulBot/?start=8_3, де в параметрі start=8_3 зашиті айді користувача та айді спеціаліста на якого підписується користувач.

За цим посиланням відкривається екран з Телеграм-ботом та великою кнопкою «Start» (рисунок 4.16).

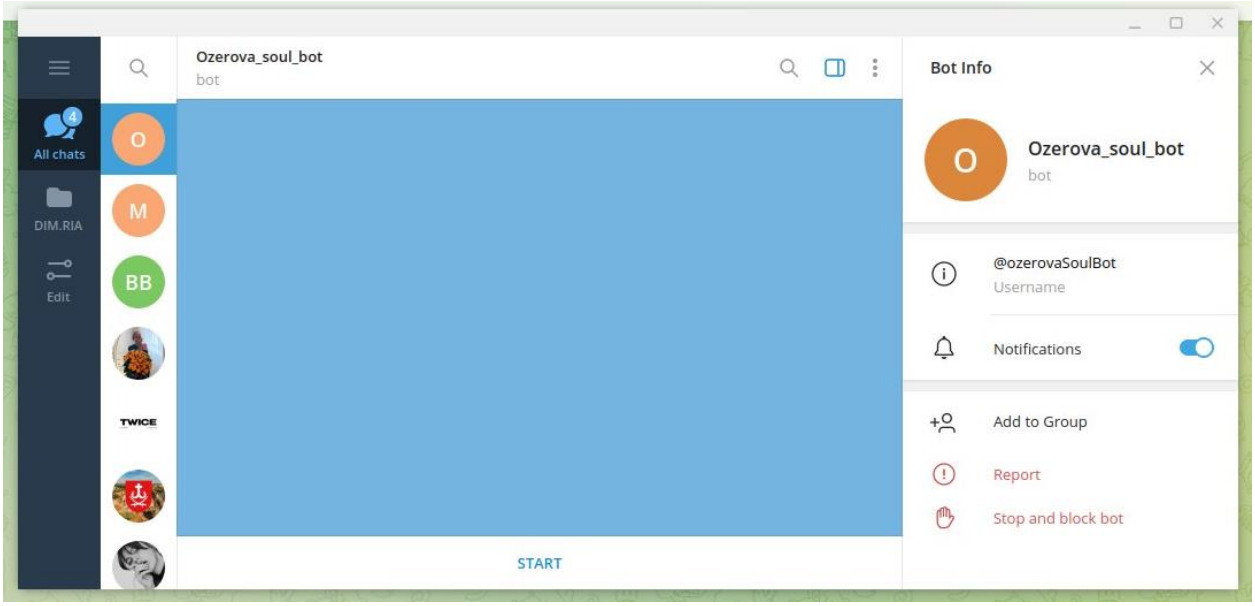

Рисунок 4.16 – Перший екран Телеграм-боту

Після кліку на кнопку «Start»відбувається запит на серверну частину вебпорталу, користувача успішно записує в базу даних, як підписника та повертається меседж з відповіддю (рисунок 4.17).

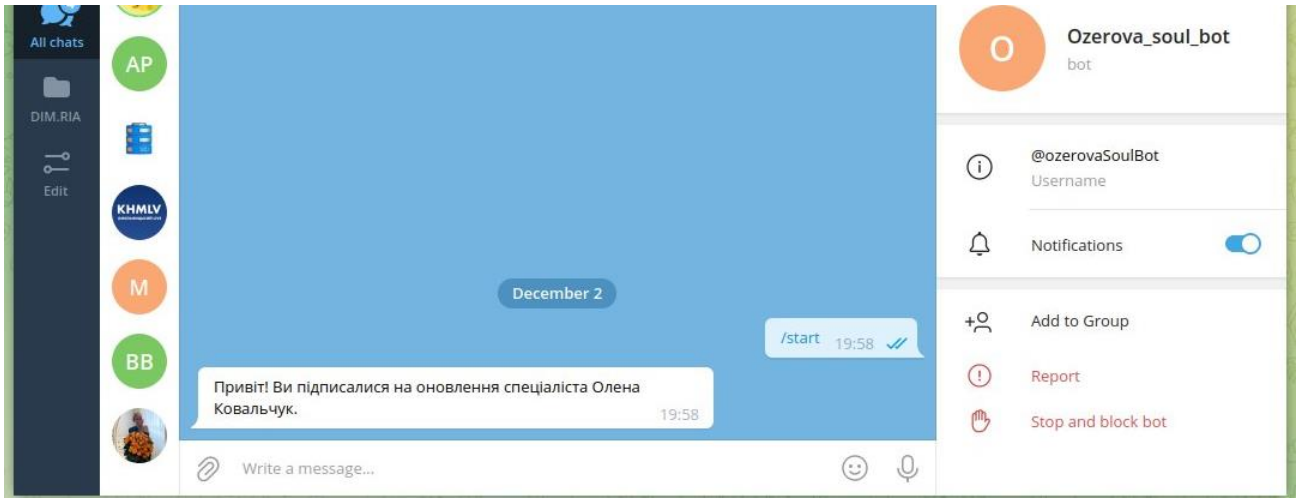

Рисунок 4.17 – Успішна підписка на Телеграм-бот

Після успішної підписки для користувача стає доступне командне меню Телеграм-боту, яке можна побачити на рисунку 4.18. Меню містить три основних команди /mysubscriptions, /appointments та /statistic.

Рисунок 4.18 – Командне меню Телеграм-боту

Розглянемо команду /mysubscriptions, яка повертає всі підписки користувача. Кожна з яких є клікабельною і відкриває детальну інформацію про спеціаліста (рисунок 4.19).

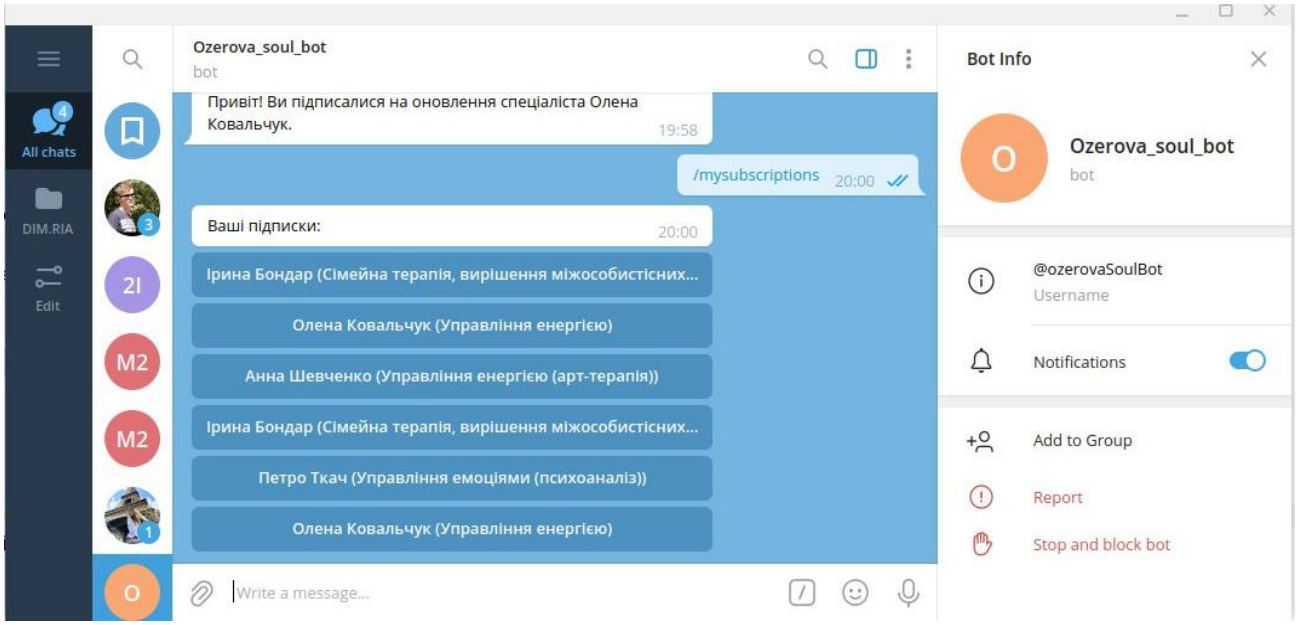

Рисунок 4.19 – Активні підписки користувача на Телеграм

Також розглянемо команду /appointments, яка в залежності від обраної опції повертає майбутні або всі записи на консультацію (рисунок 4.20).

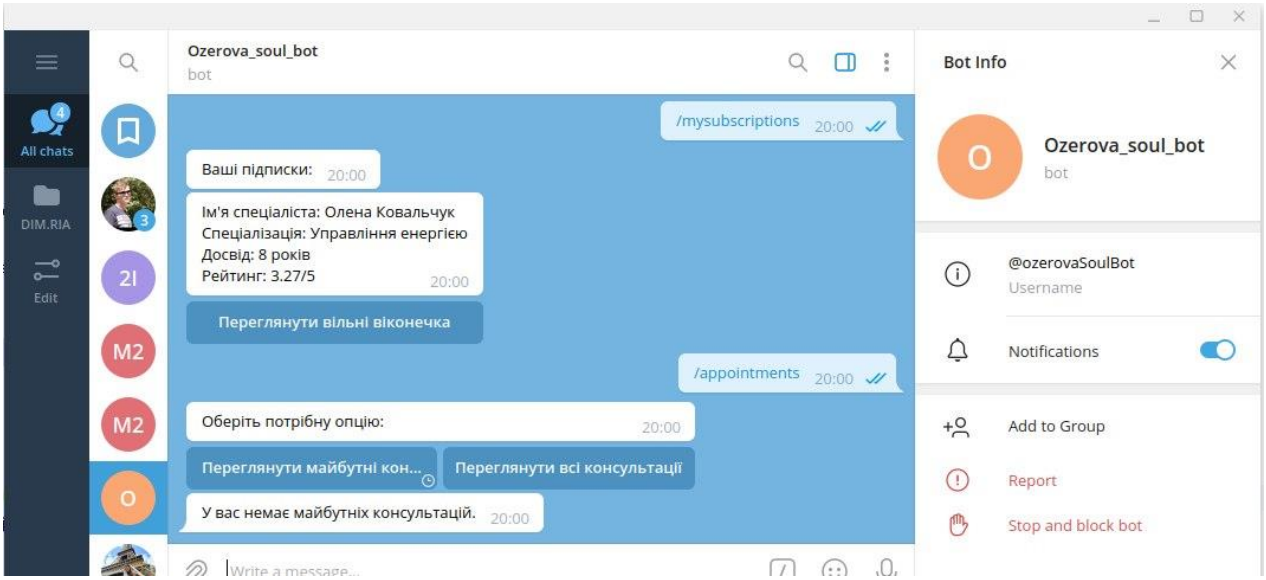

Рисунок 4.20 – Список записів на консультацію в Телеграм-боті

Якщо користувач має завершені консультації, для нього доступна команда /statistic, для перегляду графіку консультацій за останній тиждень (рис.4.21).

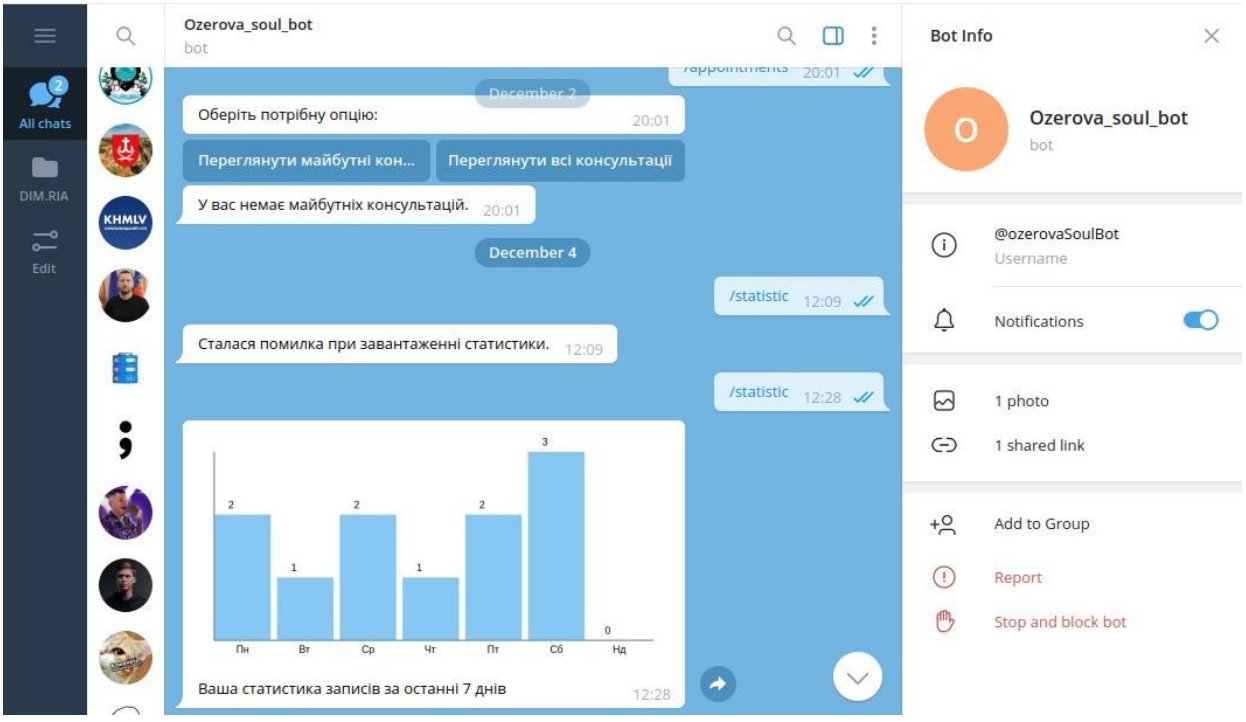

Рисунок 4.21 – Перегляд графік консультацій за останній тиждень

Окрім використання командного меню, розроблений Телеграм-бот має функцію відправки запланованих сповіщень. Розглянемо декілька прикладів таких повідомлень.

За годину до консультації користувач отримує нагадування, вигляд якого зображено на рисунку 4.22.

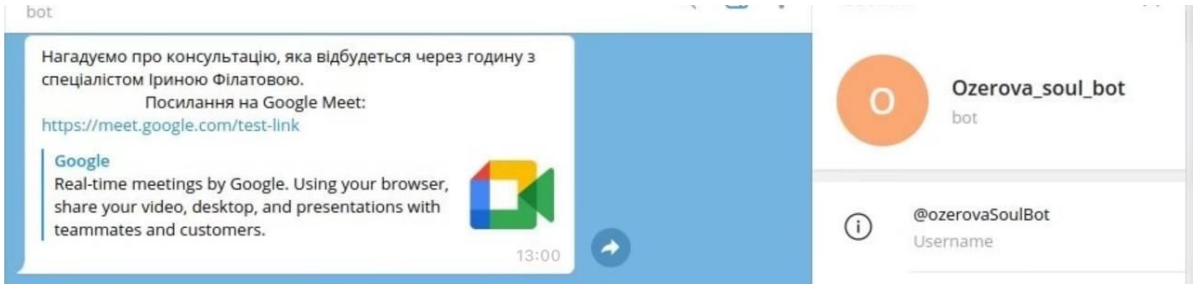

Рисунок 4.22 – Нагадування за годину до початку консультації

А за 5 хвилин після завершення консультації користувач отримує прохання оцінити консультацію з двома кнопками на вибір. Якщо натиснути на кнопку «Пропустити» то відображається повідомлення з подякою і на цьому оцінка консультації завершується. Якщо ж натиснути на кнопку «Оцінити», то поступово починають надходити повідомлення з критеріями оцінки консультації та спеціаліста в цілому.

Обидва сценарії розвитку подій зображені на рисунку 4.23.

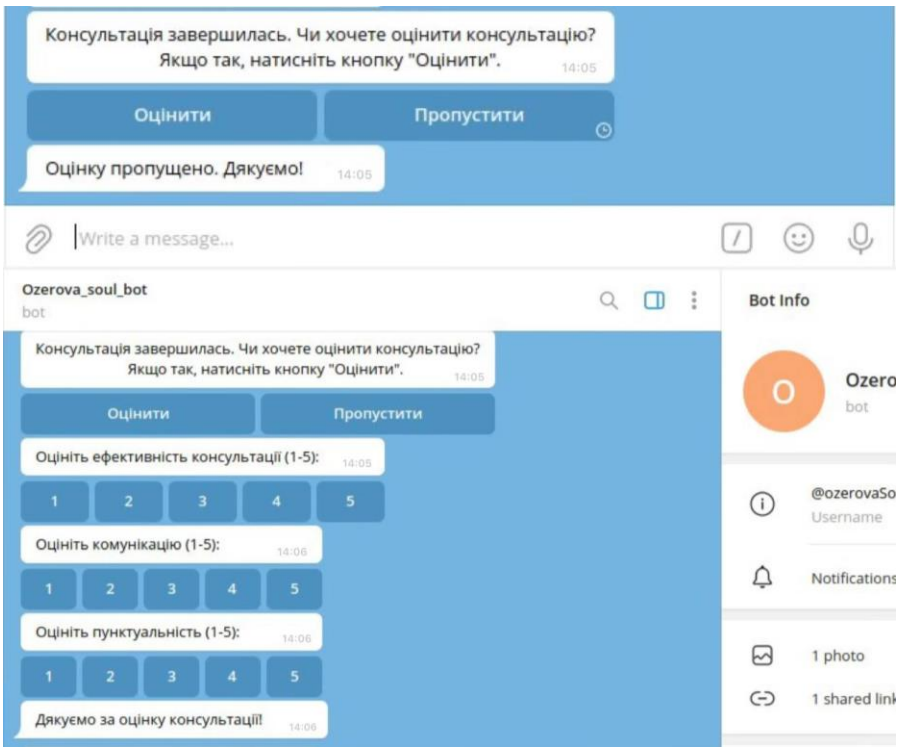

Рисунок 4.23 – Сценарії реакції на прохання оцінити консультацію

Отже, було розглянуто основний функціонал вебпорталу психологічної підтримки, здійснено тестування методом «білої скриньки» для перевірки внутрішньої логіки програми, коректності алгоритмів та структур даних, які забезпечують роботу системи.

4.3 Висновки

У четвертому розділі було детально проаналізовано методики тестування програмного забезпечення, зокрема вебпорталу психологічної підтримки, з метою забезпечення його надійності та відповідності функціональним вимогам. Вибір методу тестування «білої скриньки» дозволив провести глибоке тестування серверної частини системи, зосереджуючись на коректності алгоритмів, логіці роботи модуля розрахунку рейтингу спеціалістів, а також надійності інтеграції з Telegram-ботом.

Проведене тестування підтвердило коректність роботи основних модулів системи, включаючи реєстрацію користувачів, авторизацію, функціонал розрахунку рейтингу спеціалістів, планування консультацій і надсилання сповіщень через Telegram-бот. Отримані результати продемонстрували високу стабільність системи, її відповідність поставленим вимогам та готовність до реального використання.

5 ЕКОНОМІЧНА ЧАСТИНА

З метою успішного впровадження науково-технічної розробки, необхідно забезпечити її відповідність сучасним вимогам як у сфері науково-технічного прогресу, так і з економічної точки зору. Тому під час проведення науково-дослідних робіт важливо оцінювати економічну ефективність отриманих результатів.

Магістерська кваліфікаційна робота на тему «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки» належить до науково-технічних проектів, які спрямовані на вихід продукту на ринок. Це означає, що відбувається процес комерціалізації науково-технічного продукту, що має важливе значення, оскільки результати розробки можуть використовуватись іншими споживачами, приносячи економічний ефект. Для досягнення цього необхідно знайти потенційного інвестора, який зацікавиться проектом, та переконати його в економічній доцільності таких інвестицій.

Для цього потрібно виконати наступні етапи:

1. Провести комерційний аудит науково-технічної розробки, тобто визначити її науково-технічний рівень та комерційний потенціал.

2. Розрахувати витрати на здійснення науково-технічної розробки.

3. Оцінити економічну ефективність впровадження і комерціалізації розробки потенційним інвестором та обґрунтувати економічну доцільність такого кроку.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями [38], наведеними в табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження табл. 5.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і

комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	3	4
2. Ринкові переваги (наявність аналогів)	4	4	3
3. Ринкові переваги (ціна продукту)	3	3	4
4. Ринкові переваги (технічні властивості)	4	3	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	4	3
8. Практична здійсненність (наявність фахівців)	3	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	40	41	42
Середньоарифметична сума балів $СБ_c$	41		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній

Продовження таблиці 5.3

Середньоарифметична сума балів СБ, розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки» становить 41 бал, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою 5.1 [33]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_{o1} = 32000{,}00 \cdot 60 / 22 = 87272{,}73 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	32000	1454,55	60	87272,73
Інженер-розробник програмного забезпечення	28000	1272,73	60	76363,64
Інженер-розробник програмного забезпечення	28000	1272,73	60	76363,64
Вебдизайнер	25000	1136,36	45	51136,36
Фахівець з тестування	25000	1136,36	30	34090,91
Психолог	24000	1090,91	20	21818,18
Всього				291136,36

Витрати на основну заробітну плату робітників ($З_p$) за відповідними найменуваннями робіт НДР розраховуємо за формулою 5.2:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \text{,} \qquad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою 5.3:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати, приймемо *Мм*=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [32];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 8000,00 \cdot 1,35 \cdot 1,65 / (22 \cdot 8) = 101,25$ грн.

$З_{р1} = 101,25 \cdot 12,00 = 1215$ грн.

Величина витрат на основну заробітну плату робітників наведена в табл.5.5.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочих місць дослідників	12	3	1,35	101,25	1215
Встановлення програмного забезпечення	6	2	1,1	82,5	495
Налаштування апаратного забезпечення	16	3	1,35	101,25	1620
Всього					3330

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної

заробітної плати дослідників та робітників за формулою 5.4:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (291136,36 + 3330) · 11 / 100% = 32391,3 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 5.5:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (291136,36 + 3330+ 32391,3) · 22 / 100% = 71908,69 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою 5.6:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}, \quad (5.6)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

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

$M_1 = 6 \cdot 135{,}00 \cdot 1{,}15 - 0{,}000 \cdot 0{,}00 = 931{,}5$ грн.

Проведені розрахунки зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Набір ручок кулькових BIC Round Stic Classic Сині 8 шт х 2 упаковки	135	6	0	0	931,5
Тижневик датований Buromax 2025 Steel A5 зі штучної шкіри 136 сторінок Срібний	199	6	0	0	1373,1
Ящик паперу офісного Zoom Stora Enso A4 80 г/м2 клас C+ 5 пакувань по 500 аркушів Білий	799	1	0	0	918,85
Папка-реєстратор Delta by Axent A4 75 мм Арочна	75	6	0	0	517,5
Файл-кишеня Buromax Job A4 50 мкм глянсовий Прозорий 100 шт	90	5	0	0	517,5
Всього					4258,45

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методів і програмних засобів реалізації вебпорталу

психологічної підтримки» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Витрати на спецустаткування, які використовують при проведенні НДР на тему «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки» відсутні.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою5.8:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \quad , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 7899,00 \cdot 6 \cdot 1,1 = 52133,4$ грн.

Отримані результати зведемо до таблиці 5.7.

Таблиця 5.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	6	7899	52133,4
Прикладний пакет Microsoft Office 2019	6	9999	65993,4
Сервіс розробки інтерфейсів Figma	1	1800	2070
Середовище розробки WebStorm	3	2900	10005
Всього			130201,8

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою 5.9:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (150000{,}00 \cdot 3) / (3 \cdot 12) = 12500$ *грн.*

Проведені розрахунки зведемо до таблиці 5.8.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Lenovo 16IAH8 (6 шт.)	150000	3	3	12500

Продовження таблиці 5.8

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Робоче місце дослідника	14000	7	3	500
Оргтехніка	16500	5	3	825
Середовище розробки PhpStorm (3шт.)	10005	3	3	833,75
Прикладний пакет Microsoft Office 2021 (6 шт.)	52133,4	3	3	4344,45
Сервіс розробки інтерфейсів Figma	1800	1	3	450
Всього				19453,2

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою 5.10:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 11 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_{e1} = 0,12 \cdot 480,0 \cdot 11 \cdot 0,95 / 0,97 \cdot 6 = 3723,22$ *грн.*

Проведені розрахунки зведемо до таблиці 5.9.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер (6 шт)	0,12	480	3723,22
Робоче місце дослідника	0,2	480	1034,23
Оргтехніка	0,7	16	120,65
Всього			4878,10

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 5.11:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 22%.

$B_{св}$ = (291136,36 + 3330) · 22 / 100% = 64782,6 грн.

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

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою 5.12:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 42%.

$В_{сп}$ = (291136,36 + 3330) · 42 / 100% = 123675,87 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 5.13:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 65%.

$І_в$ = (291136,36 + 3330) · 65 / 100% = 191403,14 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 5.14:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі)

витрати», приймемо $Н_{нзв}$ = 110%.

$$В_{нзв} = (291136,36 + 3330) \cdot 100 / 110\% = 323913 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою 5.15:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.15)$$

$$В_{заг}=291136,36 + 3330+32391,3+71908,69+4258,45+0,00+130201,8+19453,2+ 4878,10+64782,6+123675,87+191403,14+323913= 1261332,51 \text{ грн.}$$

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою 5.16:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,7.

$$ЗВ = 1261332,51/ 0,7 = 1801903,59 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що

аналізуються, від покращення його певних характеристик;

1-й рік – 3000 користувачів/рік;

2-й рік – 11000 користувачів/рік;

3-й рік – 18000 користувачів/рік.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 50000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 3000,00 грн /за рік користування;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою 5.17 [39]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 =$ (50000,00·500,00+3500·3000)·0,83·0,3·(1-0,18/100%)=7277208,9 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 =$(50000,00·500,00+3500·(3000+11000))·0,83·0,3·(1-0,18/100%)= 15845791,14 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 =$(50000,00·500,00+3500·(3000+11000+18000))·0,83·0,3·(1-0,18/100%)= 21959521,58 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою 5.18:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,1;

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

$ПП =$ 7277208,9/(1+0,1)^1+ 15845791,14/(1+0,1)^2+ 21959521,58 /(1+0,1)^3= 23272615,95 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки, розраховується за формулою 5.19:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=4;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо *1801903,59* грн.

$PV = k_{інв} \cdot ЗВ$ = 4· *1801903,59* = 7207614,35 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки розраховується за формулою 5.20:

$$E_{абс} = ПП - PV \tag{5.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 23272615,95 грн;

PV – теперішня вартість початкових інвестицій, 7207614,35 грн.

$E_{абс} = ПП - PV$ = 23272615,95 - 7207614,35 = 16065001,60 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 5.21:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.21}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 16065001,60 грн;

PV – теперішня вартість початкових інвестицій, 7207614,35 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+16065001{,}60/7207614{,}35)^{1/3} = 0{,}478.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ розраховується за формулою 5.22:

$$\tau_{мін} = d + f, \tag{5.22}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,096;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,09+0,25 = 0,346 < 0,478 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 5.23:

$$T_{ок} = \frac{1}{E_в}, \tag{5.23}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,796 = 2,09 року.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки» становить 41 бал, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

Також термін окупності становить 2,09 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів і програмних засобів реалізації вебпорталу психологічної підтримки».

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було розроблено інтерактивний вебпортал психологічної підтримки з розширеним функціоналом, що включає рейтингову систему спеціалістів, інструменти тестування психічного стану, персоналізований підбір спеціалістів, а також інтеграцію Telegram-боту для нагадувань і сповіщень.

Для реалізації роботи було використано сучасні технології, зокрема Node.js для серверної частини, React для клієнтської частини, а також SQLite для управління базою даних. Робота оформлена згідно методичних вказівок [40].

Було проведено аналіз сучасного стану проблеми, описано основні аналоги існуючих систем психологічної підтримки, виявлено їхні недоліки та обмеження. На основі цього аналізу обґрунтовано переваги розробленого вебпорталу порівняно з існуючими рішеннями. Розкрито, яким чином розроблена система вирішує проблеми доступності, персоналізації та інтерактивності, забезпечуючи зручність використання та ефективність взаємодії користувачів із платформою.

Було розроблено модулі для реєстрації та авторизації користувачів із застосуванням безпечного хешування паролів. Метод розрахунку рейтингу спеціалістів із використанням вагових коефіцієнтів і нормалізації, що дозволяє забезпечити об’єктивність та стабільність рейтингу.

Інтегровано Telegram-бот, який виконує функції динамічних сповіщень, нагадувань та автоматизованого інформування користувачів. А також розроблено модуль проведення тестування психічного стану та підбору спеціалістів, які найкраще відповідають потребам користувача.

Подальшого розвитку отримав метод розрахунку рейтингу спеціалістів, який, на відміну від існуючих, базується на використанні вагових коефіцієнтів для кожного критерію оцінки та нормалізації результатів, що забезпечує стабільність і об’єктивність порівняння. Це дозволяє адаптувати рейтинг до потреб користувачів, враховувати важливість окремих аспектів роботи спеціалістів і коригувати оцінки залежно від стабільності отриманих відгуків.

Також удосконалено метод інтеграції Telegram-боту, який, на відміну від існуючих, реалізує автоматичне формування персоналізованих сповіщень, аналітичних звітів і нагадувань через систему планування подій Node Schedule. Це підвищує ефективність взаємодії користувачів із платформою, забезпечуючи своєчасну комунікацію та інформування.

Було проведено комплексне тестування системи, яке підтвердило її працездатність і відповідність технічним вимогам. Результати тестування продемонстрували стабільну роботу платформи в умовах реального використання та підтвердили ефективність розробленого функціоналу.

Розроблений вебпортал має значний практичний потенціал і може бути використаний для підвищення доступності психологічної підтримки, оптимізації роботи спеціалістів та надання користувачам ефективних інструментів для підтримки психічного здоров’я.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стрес і психологічна травма [Електронний ресурс]. – Режим доступу до ресурсу: https://nus.org.ua/articles/stres-i-psyhologichna-travma-zaraz/.

2. Чудаєва Н.В Емоційний світ людини. 2-ге вид./ Чудаєва Н.В., Шулдик Г.О. – Умань, 2019. 126 с.

3. Войтко В.В. Особливості розробки програмних модулів вебсистеми психологічної підтримки / В.В. Войтко, Озерова К. О., Барчук Н.Є., Гаврилюк О.В. // Інформаційні технології і автоматизація – 2024 / Матеріали XVII міжнародної науково-практичної конференції. Одеса, 31 жовтня - 1 листопада 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. – С.482-484.

4. Гусак Н. Гусак Психосоціальна підтримка в умовах надзвичайних ситуацій / Н. Гусак, В. Чернобровкіна, В. Чернобровкін, А. Максименко, С. Богданов, О. Бойко; за заг. ред. Н. Гусак / Нац. ун-т «Києво-Могилянська академія». – Київ: НаУКМА, 2017. – 174 с.

5. Психолог онлайн: переваги та недоліки [Електронний ресурс]. – Режим доступу до ресурсу: https://shumskyi.pro/psicholog_plusi_ta_minusi/.

6. BetterHelp [Електронний ресурс]. – Режим доступу до ресурсу: https://www.betterhelp.com/.

7. TalkSpace [Електронний ресурс]. – Режим доступу до ресурсу: https://www.talkspace.com/.

8. 7 Cups [Електронний ресурс]. – Режим доступу до ресурсу: https://www.7cups.com/.

9. Calmerry [Електронний ресурс]. – Режим доступу до ресурсу: https://calmerry.com/.

10. Leonard Richardson «RESTful Web Services» / Leonard Richardson, Sam Ruby // O'Reilly Media – 1st edition, 2017.

11. David Herron «Node.js Web Development» / David Herron // Packt Publishing. – 2020.

12. Express.js [Електронний ресурс] – Режим доступу до ресурсу:

https://expressjs.com/.

13. Koa.js [Електронний ресурс] – Режим доступу до ресурсу: https://koajs.com/.

14. Hapi.js [Електронний ресурс] – Режим доступу до ресурсу: https://hapi.dev/.

15. Socket.io [Електронний ресурс] – Режим доступу до ресурсу: https://socket.io/.

16. React [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/.

17. Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/.

18. Angular [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/.

19. Порівнюємо React, Angular і Vue — найпопулярніші бібліотеки й фреймворки у 2022 році [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/39933/.

20. Software Architecture Patterns – Layered Architecture [Електронний ресурс]. – Режим доступу до ресурсу: https://priyalwalpita.medium.com/software-architecture-patterns-layered-architecture-a3b89b71a057/.

21. Architecture Patterns переваги та недоліки [Електронний ресурс]. – Режим доступу до ресурсу: https://lazypro.medium.com/layered-architecture/.

22. Діаграма сутність-зв'язок [Електронний ресурс]. – Режим доступу до ресурсу: https://www.guru99.com/uk/er-diagram-tutorial-dbms.html.

23. Методика розрахунку системи статистичних ваг [Електронний ресурс]. – Режим доступу до ресурсу: https://ips.ligazakon.net/document/F641.

24. Нормалізація даних [Електронний ресурс]. – Режим доступу до ресурсу: https://www.vpnunlimited.com/ua/help/cybersecurity/data-normalization/.

25. Miles R. Learning UML 2.0: A Pragmatic Introduction to UML. / Miles R, Hamilton K.: O'Reilly Media, Inc., 2016. 286 p.

26. Douglas Croford. JavaScript: The Good Parts / Crockford D, 2013. 176 c.

27. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/python-programming-language/.

28. Java [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/ua/java/technologies/downloads/.

29. Ruby [Електронний ресурс] – Режим доступу до ресурсу: https://www.ruby-lang.org/en/documentation/.

30. Alan Forbes. The Joy of PHP / Forbes A. – 2015. – 128 с.

31. Go lang [Електронний ресурс] – Режим доступу до ресурсу: https://golang-blog.blogspot.com/p/go-specification.html.

32. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/.

33. WebStorm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/.

34. SublimeText [Електронний ресурс] – Режим доступу до ресурсу: https://www.sublimetext.com/.

35. Atom [Електронний ресурс] – Режим доступу до ресурсу: https://flight-manual.atom.io/.

36. PhpStorm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/phpstorm/.

37. Методи тестування [Електронний ресурс] – Режим доступу до ресурсу: https://www.softwaretestinggenius.com/.

38. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

39. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. – Вінниця: ВНТУ, 2016. – 113 с.

40. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад.: О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2024. – 50 с.

ДОДАТКИ

98

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., проф. О. Н. Романюк

«19» вересня 2024 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів реалізації веб-порталу психологічної підтримки»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доц. В.В. Войтко

"19" вересня 2024 р.

Виконала:

_____студентка гр.2ПІ-23м К.О. Озерова

"19" вересня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів реалізації веб-порталу психологічної підтримки».

Галузь застосування – вебтехнології.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №310 від 17 вересня 2024 р. ректора ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей для надання психологічної підтримки за рахунок розробки методів і програмних засобів для інтерактивного вебпорталу, що забезпечить персоналізовану взаємодію з користувачами, підвищить точність рекомендацій та оптимізує процеси вибору спеціалістів.

4. Вихідні дані для проведення НДР

1. Leonard Richardson «RESTful Web Services» / Leonard Richardson, Sam Ruby // O'Reilly Media – 1st edition, 2017.

2. David Herron «Node.js Web Development» / David Herron // Packt Publishing. – 2020.

3. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 50 с.

5. Технічні вимоги

Вхідні дані – користувач системи, електронна пошта, телеграм, браузер.

Вихідні дані – графічний інтерфейс користувача з можливістю взаємодіяти з вебпорталом психологічної підтримки.

6. Конструктивні вимоги.

Інтерфейс вебпорталу повинен відповідати естетичним та ергономічним вимогам, Текстова та ілюстративна документація повинна відповідати стандартам. Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку вебпорталів психологічної підтримки	20.09.2024 – 07.10.2024
2	Розробка методу та моделей вебпорталу	08.10.2024 – 21.10.2024
3	Розробка програмних модулів реалізації вебпорталу психологічної підтримки	22.10.2024 – 16.11.2024
4	Тестування програми	17.11.2024 – 23.11.2024
5	Економічна частина	24.11.2024 – 30.11.2024

10. Порядок контролю та прийняття.

Усі етапи магістерської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

101

ДОДАТОК Б

Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів і програмних засобів реалізації веб-порталу психологічної підтримки.**

Тип роботи: кваліфікаційна робота
Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ-23м
Науковий керівник: : к.т.н., доц. каф. ПЗ Войтко В.В.

Turnitin	
Оригінальність	**97 %**
Схожість	3 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.
Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методів і програмних засобів реалізації веб-порталу психологічної підтримки».

Автор _____ Озерова Катерина Олександрівна

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

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

ДОДАТОК В

Лістинг коду

Backend

```
const Koa = require('koa');
const bodyParser = require('koa-bodyparser');
const routes = require('./routes');
const cors = require('@koa/cors');

const app = new Koa();
app.use(bodyParser());
app.use(cors({
   origin: '*',
   methods: ['GET', 'POST'],
}));


app.use(routes.routes());
app.listen(3001);

const Router = require('koa-router');
const UsersController = require('./src/controllers/UsersController');
const SpecialistsController = require('./src/controllers/SpecialistsController');
const RatingController = require("./src/controllers/RatingController");
const AppointmentsController = require("./src/controllers/AppointmentsController");
const SubscribeController = require("./src/controllers/SubscribeController");
const TestingController = require("./src/controllers/TestingModule/TestingController");

const router = new Router();

router.get('/users', UsersController.getAllUsers);
router.get('/user', UsersController.getUserById);
router.get('/specialists', SpecialistsController.getAllSpecialists)
router.get('/rating', RatingController.getSpecialistRating)
router.get('/calendar', AppointmentsController.getAppointments)

router.post('/user-register', UsersController.saveNewUser)
router.post('/login', UsersController.loginUser)
router.post('/add-appointment', AppointmentsController.addAppointments)
router.post('/submit-test', TestingController.getResult)
router.post('/subscribe', SubscribeController.subscribeToTelegramBot);

module.exports = router;

const bcrypt = require('bcrypt');
const userManager = require("../managers/UsersManager");
const {checkAndRegister, checkUserByEmail} = require("../managers/UsersManager");

const getAllUsers = async (ctx) => {
  try {
    const users = await userManager.getAllUsers();
    ctx.status = 200;
    ctx.body = users;
  } catch (error) {
    ctx.status = 500;
    ctx.body = { error: "Виникла помилка при отриманні користувачів" };
    console.error("Помилка при отриманні користувачів:", error);
  }
};

const getUserById = async (ctx) => {
  const { userId } = ctx.query;
  try {
    const [user] = await userManager.getUserById(Number(userId));
    ctx.status = 200;
```

```
    ctx.body = user;
  } catch (error) {
    ctx.status = 500;
    ctx.body = { error: "Виникла помилка при отриманні даних про користувача" };
    console.error("Помилка при отриманні даних про користувача:", error);
  }
}

const saveNewUser = async (ctx) => {
  const {name, email, password} = ctx.request.body;

  if (!name || !email || !password) {
    ctx.status = 400;
    ctx.body = { message: 'Будь ласка, заповніть усі поля' };
    return;
  }

  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const { existingUsers } = await checkAndRegister({ name, email, hashedPassword });
    console.log(existingUsers, 'UsersController.js', 42);
    if (existingUsers) {
      ctx.status = 409;
      ctx.body = {message: 'Користувач з такою електронною поштою вже існує'};
    } else {
      ctx.status = 201;
      ctx.body = {message: 'Користувач успішно зареєстрований'};
    }
  } catch (error) {
    console.error('Помилка при реєстрації користувача:', error);
    ctx.status = 500;
    ctx.body = {message: 'Помилка сервера'};
  }
};

const loginUser = async (ctx) => {
  const { email, password } = ctx.request.body;

  if (!email || !password) {
    ctx.status = 400;
    ctx.body = { message: 'Будь ласка, заповніть усі поля' };
    return;
  }

  try {
    const user = await checkUserByEmail(email);
    if (!user) {
      ctx.status = 404;
      ctx.body = { message: 'Користувача з такою електронною поштою не знайдено' };
      return;
    }

    const validPassword = await bcrypt.compare(password, user.password);
    if (!validPassword) {
      ctx.status = 401;
      ctx.body = { message: 'Неправильна електронна адреса або пароль' };
      return;
    }

    ctx.status = 200;
    ctx.body = {
      id: user.id,
      username: user.username,
      email: user.email,
    };
  } catch (error) {
    console.error('Помилка при вході користувача:', error);
    ctx.status = 500;
    ctx.body = { message: 'Помилка сервера' };
  }
```

```
}

module.exports = {
  getAllUsers,
  getUserById,
  saveNewUser,
  loginUser,
};

const { findSubscription, addSubscription } = require('../managers/SubscribeManager');

const subscribeToTelegramBot = async (ctx) => {
    const { userId, specialistId } = ctx.request.body;
    if (!userId || !specialistId) return ctx.status = 400;
    try {
        const existingSubscriber = await findSubscription(userId, specialistId);

        if (existingSubscriber.length === 0) {
            await addSubscription(userId, specialistId, null);
            return ctx.status = 201;
        } else {
            return ctx.status = 200;
        }
    } catch (error) {
        console.error('Помилка при підписці на Telegram-бота:', error);
        return ctx.status = 500;
    }
};

module.exports = { subscribeToTelegramBot };

const _ = require("lodash");
const specialistsManager = require("../managers/SpecialistsManager");
const { getReviewsBySpecialistId } = require("../managers/ReviewsManager");

const getAllSpecialists = async (ctx) => {
    try {
        const specialists = await specialistsManager.getSpecialists(ctx.query);
        const specialistIdToReviews = await getReviewsBySpecialistId(_.map(specialists, 'id'));
        ctx.status = 200;
        ctx.body = specialists.map(specialist => ({
            ...specialist,
            reviews: specialistIdToReviews[specialist.id],
        }));
    } catch (error) {
        ctx.status = 500;
        ctx.body = [];
        console.error("Помилка при отриманні спеціалістів:", error);
    }
};

module.exports = {
    getAllSpecialists,
};

const calculateRating = require('../helpers/ratingCalculator');
const {getReviewsBySpecialistId} = require("../managers/ReviewsManager");
const {getSpecialists, updateSpecialistRating} = require("../managers/SpecialistsManager");
const {getAppointmentsBySpecialist} = require("../managers/AppointmentsManager");
const {findSubscriptionsBySpecialistId} = require("../managers/SubscribeManager");

const getSpecialistRating = async (ctx) => {
    const { id } = ctx.query;
    if (!Number(id)) {
        ctx.status = 404;
        ctx.body = {};
    }
    try {
        const [specialist] = await getSpecialists({ id });
        if (!specialist) {
```

```
            ctx.status = 404;
            ctx.body = {};
        }
        const [reviews, appointments, telegramSubscribers] = await Promise.all([
            getReviewsBySpecialistId([id]),
            getAppointmentsBySpecialist(id),
            findSubscriptionsBySpecialistId(id),
        ]);

        const rating = calculateRating(
            {specialist, reviews: reviews[id], appointments, telegramSubscribers}
        );
        await updateSpecialistRating({ rating: rating.overallRating, id });

        return ctx.body = {
            specialist: {
                id: specialist.id,
                name: specialist.name,
                experience: specialist.experience,
                expertise: specialist.expertise,
            },
            rating: rating.overallRating,
            ratingDetails: rating.details
        };
    } catch (error) {
        console.error('Помилка при розрахунку рейтингу:', error);
        return ctx.status = 400;
    }
};

module.exports = { getSpecialistRating };

const { getAppointmentsBySpecialist, createAppointment} = require("../managers/AppointmentsManager");
const {findSubscription, addSubscription} = require("../managers/SubscribeManager");

const getAppointments = async (ctx) => {
    const { specialistId } = ctx.query;
    try {
        ctx.body = await getAppointmentsBySpecialist(Number(specialistId));
    } catch (error) {
        console.error('Помилка при отриманні записів на прийом:', error);
        ctx.status = 500;
        ctx.body = [];
    }
};

const addAppointments = async (ctx) => {
    const { userId, specialistId, appointmentDate  } = ctx.request.body;
    if (!userId || !specialistId  || !appointmentDate) return ctx.status = 400;
    try {
        await createAppointment({ userId, specialistId, appointmentDate });
        return ctx.status = 200;
    } catch (error) {
        console.error('Помилка при створенні запису на консультацію:', error);
        return ctx.status = 500;
    }
};

module.exports = {
    getAppointments,
    addAppointments,
};

const { getSpecialists } = require('../../managers/SpecialistsManager');
const questions = require("../../constants/test");

const getResult = async (ctx) => {
    const answers = ctx.request.body;

    if (!answers || typeof answers !== "object") {
```

```
        ctx.status = 400;
        ctx.body = { message: "Некоректний формат даних" };
        return;
    }

    const valid = questions.every(q => Object.keys(answers).includes(q.id));
    if (!valid) {
        ctx.status = 400;
        ctx.body = { message: "Відсутні відповіді на всі питання" };
        return;
    }

    try {
        // Аналіз відповідей і визначення критеріїв
        const criteria = analyzeCriteria(answers);

        // Визначення загального стану
        const score = calculateScore(answers);
        const overallStatus = getOverallStatus(score);

        // Отримання спеціалістів
        const specialists = await getSpecialists({});
        const recommendedSpecialists = specialists.filter(specialist =>
            filterSpecialistsByCriteria(specialist, criteria)
        );

        ctx.body = {
            analysisResult: {
                score,
                overallStatus,
                criteria,
            },
            recommendedSpecialists,
        };
    } catch (error) {
        console.error("Помилка обробки результатів:", error);
        ctx.status = 500;
        ctx.body = { message: "Помилка сервера" };
    }
};

// Аналіз відповідей і визначення критеріїв
const analyzeCriteria = (answers) => {
    const criteria = {
        needEnergyManagement: false,
        needAnxietySupport: false,
        needEmotionalManagement: false,
        needSocialSupport: false,
        readinessForSpecialist: false,
    };

    for (const question of questions) {
        const answer = answers[question.id];

        switch (answer) {
            case "Постійно":
            case "Часто":
            case "Так":
                // Сильний вплив на критерії
                updateCriteria(criteria, question.category);
                break;
            case "Іноді":
            case "Час від часу":
            case "Середній":
                // Помірний вплив на критерії
                updateCriteria(criteria, question.category);
                break;
            case "Рідко":
            case "Ніколи":
            case "Низький":
```

```
            case "Ні":
                // Мінімальний або нульовий вплив
                break;
            default:
                console.warn(`Невідомий варіант відповіді: ${answer}`);
                break;
        }
    }

    return criteria;
};

// Допоміжна функція для оновлення критеріїв
const updateCriteria = (criteria, category) => {
    switch (category) {
        case "EnergyManagement":
            criteria.needEnergyManagement = true;
            break;
        case "AnxietySupport":
            criteria.needAnxietySupport = true;
            break;
        case "EmotionalManagement":
            criteria.needEmotionalManagement = true;
            break;
        case "SocialSupport":
            criteria.needSocialSupport = true;
            break;
        case "ReadinessForSpecialist":
            criteria.readinessForSpecialist = true;
            break;
        default:
            console.warn(`Невідома категорія: ${category}`);
            break;
    }
};

const calculateScore = (answers) => {
    let score = 0;
    for (const key in answers) {
        switch (answers[key]) {
            case "Рідко":
            case "Ніколи":
            case "Низький":
            case "Ні":
                score += 1;
                break;
            case "Іноді":
            case "Час від часу":
            case "Середній":
                score += 2;
                break;
            case "Часто":
            case "Постійно":
            case "Так":
                score += 3;
                break;
            default:
                console.warn(`Невідомий варіант відповіді: ${answers[key]}`);
                break;
        }
    }
    return score;
};



// Визначення загального стану
const getOverallStatus = (score) => {
    if (score === 0) {
        return "Неможливо визначити ваш стан. Пройдіть спочатку тестування.";
```

```
    } else if (score <= 15) {
        return "Ваш емоційний стан в нормі.";
    } else if (score <= 25) {
        return "Ви відчуваєте певний дискомфорт. Рекомендується консультація.";
    } else {
        return "Високий рівень стресу. Рекомендована допомога спеціаліста.";
    }
};

// Фільтрація спеціалістів за критеріями
const filterSpecialistsByCriteria = (specialist, criteria) => {
    if (criteria.needEnergyManagement && specialist.specialization.toLowerCase().includes("управління енергією")) {
        return true;
    }
    if (criteria.needAnxietySupport && specialist.specialization.toLowerCase().includes("допомога при тривозі")) {
        return true;
    }
    if (criteria.needEmotionalManagement && specialist.specialization.toLowerCase().includes("управління емоціями")) {
        return true;
    }
    if (criteria.needSocialSupport && specialist.specialization.toLowerCase().includes("вирішення міжособистісних конфліктів"))
{
        return true;
    }
    return false;
};

module.exports = { getResult };

const sqlite3 = require('sqlite3').verbose();

const DB_PATH = 'soulTalk.sqlite';
const db = new sqlite3.Database(DB_PATH, (err) => {
    if (err) {
        console.error('Error connecting to database', err.message);
    } else {
        console.log('Connected to database');
    }
});


const runQuery = (query, params = []) => {
    return new Promise((resolve, reject) => {
        db.all(query, params, (err, rows) => {
            if (err) {
                reject(err);
            } else {
                resolve(rows);
            }
        });
    });
};

module.exports = {
    runQuery
};

const _ = require('lodash');

const weights = {
    effectiveness: 0.3,
    userSatisfaction: 0.25,
    experience: 0.1,
    communication: 0.3,
    punctuality: 0.2,
    clientCount: 0.2,
    reputation: 0.3,
    telegramSubscribers: 0.1
};
```

```
const normalizeWeights = (weights) => {
    const totalWeight = Object.values(weights).reduce((sum, w) => sum + w, 0);
    return Object.keys(weights).reduce((normalized, key) => {
        normalized[key] = weights[key] / totalWeight;
        return normalized;
    }, {});
};

const normalizedWeights = normalizeWeights(weights);

/**
 * Округлення значень усіх полів об'єкта до двох знаків після коми
 * @param {object} details
 * @returns {object}
 */
const roundDetails = (details) => {
    return Object.keys(details).reduce((rounded, key) => {
        rounded[key] = Math.round(details[key] * 100) / 100;
        return rounded;
    }, {});
};

/**
 * @param specialist
 * @param reviews
 * @param appointments
 * @param telegramSubscribers
 * @returns {{overallRating: string, details: {}}}
 */
const calculateRating = (
    { specialist = {}, reviews = [], appointments = [], telegramSubscribers = [] }
) => {
    let details = {
        effectiveness: calculateEffectiveness(appointments),
        userSatisfaction: calculateUserSatisfaction(reviews),
        experience: calculateExperienceScore(specialist),
        communication: calculateCommunication(appointments),
        punctuality: calculatePunctuality(appointments),
        clientCount: calculateClientCount(appointments),
        reputation: calculateReputation(reviews, appointments),
        telegramSubscribers: calculateTelegramSubscribers(telegramSubscribers)
    };
    details = roundDetails(details);
    const weightedSum = Object.keys(details).reduce(
        (sum, key) => sum + details[key] * normalizedWeights[key], 0
    );

    return { overallRating: weightedSum.toFixed(2), details };
};

/**
 * Розрахунок пункту "Ефективність консультацій"
 * @param appointments
 * @returns {number}
 */
const calculateEffectiveness = (appointments) => {
    const avgEffectiveness = _.meanBy(appointments, 'rating') || 0;
    return Math.min(5, Math.max(1, avgEffectiveness));
};

/**
 * Розрахунок пункту "Задоволеність користувачів"
 * @param reviews
 * @returns {number}
 */
const calculateUserSatisfaction = (reviews) => {
    const avgRating = _.meanBy(reviews, 'rating') || 0;
    return Math.min(5, Math.max(1, avgRating));
};
```

```
/**
 * Розрахунок пункту "Досвід роботи"
 * @param specialist
 * @returns {number}
 */
const calculateExperienceScore = (specialist) => {
    const yearsExperience = _.get(specialist, 'experience', 0);
    if (yearsExperience < 1) return 1;
    if (yearsExperience <= 3) return 2;
    if (yearsExperience <= 5) return 3;
    if (yearsExperience <= 10) return 4;
    return 5;
};

/**
 * Розрахунок пункту "Комунікація"
 * @param appointments
 * @returns {number|number}
 */
const calculateCommunication = (appointments) => {
    return _.meanBy(appointments, 'communicationRating') || 1;
};

/**
 * Розрахунок пункту "Пунктуальність"
 * @param appointments
 * @returns {number|number}
 */
const calculatePunctuality = (appointments) => {
    return _.meanBy(appointments, 'punctualityRating') || 1;
};

/**
 * Розрахунок пункту "Кількість клієнтів"
 * @param appointments
 * @returns {number}
 */
const calculateClientCount = (appointments) => {
    const uniqueClients = _.uniqBy(appointments, 'user_id').length;
    if (uniqueClients > 15) return 5;
    if (uniqueClients > 9) return 4;
    if (uniqueClients > 6) return 3;
    if (uniqueClients > 3) return 2;
    return 1;
};

/**
 * Розрахунок пункту "Популярність у Telegram"
 * @param subscribers
 * @returns {number}
 */
const calculateTelegramSubscribers = (subscribers) => {
    const activeSubscribers = subscribers.filter(sub => sub.status === 'active').length;

    if (activeSubscribers > 10) return 5;
    if (activeSubscribers > 8) return 4;
    if (activeSubscribers > 4) return 3;
    if (activeSubscribers > 2) return 2;
    return 1;
};

/**
 * Розрахунок пункту "Репутація"
 * @param reviews
 * @param appointments
 * @returns {number}
 */
const calculateReputation = (reviews, appointments) => {
    const weights = [...reviews, ...appointments].map(r => r.rating === 5 ? 1.5 : r.rating === 1 ? 0.5 : 1);
```

```
    const weightedRatings = [...reviews, ...appointments].map((r, i) => r.rating * weights[i]);
    return _.sum(weightedRatings) / _.sum(weights);
};

module.exports = calculateRating;

const { runQuery } = require('../helpers/dbConnect');

const getAppointmentsByUser = async (userId) => {
    const query = `
        SELECT *
        FROM appointments
        WHERE user_id = ?
        ORDER BY appointment_datetime DESC
    `;

    try {
        return runQuery(query, [userId]);
    } catch (error) {
        console.error('Помилка при отриманні записів по користувачу:', error);
        throw error;
    }
};

// Отримати записи користувача для конкретного спеціаліста
const findAppointmentsBySpecialist = async (userId, specialistId) => {
    const query = `
        SELECT appointment_datetime
        FROM appointments
        WHERE user_id = ? AND specialist_id = ?
    `;
    return await runQuery(query, [userId, specialistId]);
};

/**
 * Створення запису на консультацію
 * @param userId
 * @param specialistId
 * @param appointmentDate
 * @returns {Promise<unknown>}
 */
const createAppointment = async ({ userId, specialistId, appointmentDate }) => {
    const query = `
        INSERT INTO appointments (user_id, specialist_id, appointment_datetime, created_at)
        VALUES (?, ?, ?, CURRENT_TIMESTAMP)
    `;
    return await runQuery(query, [userId, specialistId, appointmentDate]);
};

const getAppointmentsBySpecialist = (specialistId) => {
    const query = `
        SELECT *
        FROM appointments
        WHERE specialist_id = ?
        ORDER BY appointment_datetime DESC
    `;
    try {
        return runQuery(query, [specialistId]);
    } catch (error) {
        console.error('Помилка при отриманні записів по спеціалісту:', error);
        throw error;
    }
};

const getUpcomingAppointments = async () => {
    try {
        const query = `
            SELECT
                a.id,
                a.user_id,
```

```
            a.specialist_id,
            ts.chat_id,
            a.appointment_datetime,
            s.name AS specialist_name
        FROM appointments a
                JOIN specialists s ON a.specialist_id = s.id
                JOIN telegram_subscribers ts ON a.user_id = ts.user_id
        WHERE DATETIME(a.appointment_datetime) >= CURRENT_TIMESTAMP
          AND DATETIME(a.appointment_datetime, '-1 hour') <= CURRENT_TIMESTAMP
          AND ts.status = 'active';
      `;
      return await runQuery(query);
    } catch (error) {
      console.error('Помилка при отриманні майбутніх записів:', error);
      throw error;
    }
};

const getCompletedAppointments = async () => {
    try {
      const query = `
        SELECT
          a.id,
          a.user_id,
          a.specialist_id,
          ts.chat_id,
          a.appointment_datetime
        FROM appointments a
                JOIN telegram_subscribers ts
                    ON a.user_id = ts.user_id
        WHERE DATETIME(a.appointment_datetime, '+1 hour') <= CURRENT_TIMESTAMP
          AND DATETIME(a.appointment_datetime, '+1 hour', '+5 minutes') > CURRENT_TIMESTAMP
          AND ts.status = 'active'
          AND (a.rating IS NULL);
      `;
      return await runQuery(query);
    } catch (error) {
      console.error('Помилка при отриманні завершених консультацій:', error);
      throw error;
    }
};

async function updateAppointmentRatings(appointmentId, { rating, communicationRating, punctualityRating }) {
    const query = `
      UPDATE appointments
      SET rating = ?, communicationRating = ?, punctualityRating = ?
      WHERE id = ?;
    `;
    await runQuery(query, [rating, communicationRating, punctualityRating, appointmentId]);
}


module.exports = {
    createAppointment,
    updateAppointmentRatings,
    getAppointmentsByUser,
    getAppointmentsBySpecialist,
    getUpcomingAppointments,
    getCompletedAppointments,
};

const _ = require("lodash");
const { runQuery } = require("../helpers/dbConnect");

/**
 * Дістає відгуки про спеціаліста
 * @param ids - масив айдішок спеціалістів
 * @returns {Promise<unknown>}
 */
const getReviewsBySpecialistId = async (ids = []) => {
```

```
    if (!ids.length) return Promise.resolve({});

    const placeholders = ids.map(() => '?').join(', ');
    const query = `SELECT * FROM reviews WHERE specialist_id IN (${placeholders})`;
    const reviews = await runQuery(query, ids);

    if (!reviews.length) return Promise.resolve({});
    return Object.keys(reviews).length === 1 ? reviews : _.groupBy(reviews, 'specialist_id');
};

module.exports = {
    getReviewsBySpecialistId,
};

const _ = require("lodash");
const { runQuery } = require("../helpers/dbConnect");

/**
 * Дістає всіх спеціалістів
 * @param id
 * @returns {Promise<unknown>}
 */
const getSpecialists = ({ id = 0 }) => {
    let query = 'SELECT * FROM specialists';
    const ids = _.castArray(id).map(Number).filter(Boolean);

    if (ids.length) {
        const placeholders = ids.map(() => '?').join(',');
        query += ` WHERE id IN (${placeholders})`;
    }

    return runQuery(query, ids);
};


/**
 * Оновлюємо рейтинг спеціаліста
 * @param id
 * @param rating
 * @returns {Promise<void>|Promise<unknown>}
 */
const updateSpecialistRating = ({ id, rating }) => {
    if (!id || !rating) return Promise.resolve();
    const query = 'UPDATE specialists SET rating = ? WHERE id = ?';

    return runQuery(query, [rating, id]);
}

module.exports = {
    getSpecialists,
    updateSpecialistRating,
};

const { runQuery } = require('../helpers/dbConnect');

const findSubscription = async (userId, specialistId) => {
    const query = `
        SELECT * FROM telegram_subscribers
        WHERE user_id = ? AND specialist_id = ?
    `;
    return await runQuery(query, [userId, specialistId]);
};

const addSubscription = async (userId, specialistId) => {
    const query = `
        INSERT INTO telegram_subscribers (user_id, specialist_id, status)
        VALUES (?, ?, 'active')
    `;
    return await runQuery(query, [userId, specialistId]);
};
```

```
const updateChatId = async ({ userId, specialistId, chatId }) => {
   const query = `
      UPDATE telegram_subscribers
      SET chat_id = ?
      WHERE user_id = ? AND specialist_id = ?
   `;
   return await runQuery(query, [chatId, userId, specialistId]);
};

const findSubscriptionsByUser = async (userId) => {
   const query = `
      SELECT specialists.name, specialists.specialization
      FROM telegram_subscribers
      JOIN specialists ON telegram_subscribers.specialist_id = specialists.id
      WHERE telegram_subscribers.user_id = ? AND telegram_subscribers.status = 'active'
   `;
   return await runQuery(query, [userId]);
};


const findSubscriptionsByChatId = async (chatId) => {
   const query = `
      SELECT specialists.name, specialists.specialization, specialists.id
      FROM telegram_subscribers
      JOIN specialists ON telegram_subscribers.specialist_id = specialists.id
      WHERE telegram_subscribers.chat_id = ? AND telegram_subscribers.status = 'active'
   `;
   return await runQuery(query, [chatId]);
};

const findSubscriptionsBySpecialistId = async (specialistId) => {
   const query = `
      SELECT *
      FROM telegram_subscribers
      WHERE telegram_subscribers.specialist_id = ? AND telegram_subscribers.status = 'active'
   `;
   return await runQuery(query, [specialistId]);
};

module.exports = {
   findSubscription,
   findSubscriptionsByUser,
   findSubscriptionsByChatId,
   findSubscriptionsBySpecialistId,
   addSubscription,
   updateChatId
};

const { runQuery } = require("../helpers/dbConnect");

/**
 * Дістає всіх зареєстрованих юзерів
 * @returns {Promise<unknown>}
 */
const getAllUsers = () => {
  const query = "SELECT * FROM users";
  return runQuery(query);
};

const getUserById = (userId) => {
  if (!(Number(userId))) return Promise.resolve([]);
  const query = "SELECT * FROM users WHERE id = ?";
  return runQuery(query, [userId]);
}

/**
 * Додавання нового юзера
 * @param user.userName - ім`я
 * @param user.email - емейл
```

```
 * @param user.password - пароль
 * @returns {Promise<unknown>}
 */
const registerUser = async (user) => {
  const { name, email, hashedPassword } = user;
  const registerUserSQL = 'INSERT INTO users (username, created_at, email, password) VALUES (?, CURRENT_TIMESTAMP, ?,
?)';
  return runQuery(registerUserSQL, [name, email, hashedPassword]);
};

/**
 * Перевірка на наявність користувача за email
 * @param email
 * @returns {Promise<boolean>}
 */
const checkUserByEmail = async (email) => {
  const user = await runQuery('SELECT * FROM users WHERE email = ?', [email]);
  return user.length > 0 ? user[0] : null;
};

/**
 * Перевірка та реєстрація нового користувача
 * @param user
 * @returns {Promise<{existingUsers: boolean}>}
 */
const checkAndRegister = async (user) => {
  const existingUser = await checkUserByEmail(user.email);
  if (existingUser) {
    return { existingUsers: true };
  }
  await registerUser(user);
  return { existingUsers: false };
};


const findUserByChatId = async (chatId) => {
  const query = `
        SELECT user_id, username
        FROM telegram_subscribers
        JOIN users ON telegram_subscribers.user_id = users.id
        WHERE telegram_subscribers.chat_id = ?
    `;
  return await runQuery(query, [chatId]);
};

module.exports = {
  getAllUsers,
  getUserById,
  findUserByChatId,
  checkAndRegister,
  checkUserByEmail,
};

];


module.exports = questions;
const fs = require('fs');
const dayjs = require("dayjs");
const TelegramBot = require('node-telegram-bot-api');
const isSameOrAfter = require('dayjs/plugin/isSameOrAfter');
const { updateChatId, findSubscriptionsByChatId} = require('../managers/SubscribeManager');
const {getSpecialists} = require("../managers/SpecialistsManager");
const {getAppointmentsByUser} = require("../managers/AppointmentsManager");
const {findUserByChatId} = require("../managers/UsersManager");
const {generateStatisticsChart} = require("./statistic");
const schedule = require("node-schedule");

dayjs.extend(isSameOrAfter);
```

```
const token = super secret;

const bot = new TelegramBot(token, { polling: true });

bot.setMyCommands([
    { command: '/mysubscriptions', description: 'Переглянути свої підписки' },
    { command: '/appointments', description: 'Переглянути консультації' },
    { command: '/statistic', description: 'Переглянути статистику за тиждень' },
]);

bot.onText(/\/start(?: (.+))?/, async (msg, match) => {
    const chatId = msg.chat.id;

    if (!match[1]) return bot.sendMessage(chatId, 'Ви успішно почали роботу з ботом');
    const [userId, specialistId] = match[1].split('_');

    try {
        const [specialist] = await getSpecialists({ id: specialistId });
        console.log(specialist, 'bot.js', 26);
        const text = `Привіт! Ви підписалися на оновлення спеціаліста ${specialist.name}.`;

        await updateChatId({ userId, specialistId, chatId });
        await bot.sendMessage(chatId, text);
    } catch (error) {
        await bot.sendMessage(chatId, 'Сталася помилка при підписці. Спробуйте ще раз.');
        console.error('Помилка при обробці підписки:', error);
    }
});
bot.onText(/\/mysubscriptions/, async (msg) => {
    const chatId = Number(msg.chat.id);
    try {
        const subscriptions = await findSubscriptionsByChatId(chatId);

        if (subscriptions.length === 0) {
            return bot.sendMessage(chatId, 'У вас немає активних підписок.');
        }
        // Створення повідомлення з кнопками для кожного спеціаліста
        const inlineButtons = subscriptions.map(sub => [
            { text: `${sub.name} (${sub.specialization})`, callback_data: `view_${sub.id}` }
        ]);
        const options = {
            reply_markup: {
                inline_keyboard: inlineButtons
            }
        };
        bot.sendMessage(chatId, 'Ваші підписки:', options);
    } catch (error) {
        await bot.sendMessage(chatId, 'Помилка при завантаженні підписок.');
        console.error(error);
    }
});
bot.onText(/\/appointments/, async (msg) => {
    const chatId = msg.chat.id;

    const options = {
        reply_markup: {
            inline_keyboard: [
                [
                    { text: 'Переглянути майбутні консультації', callback_data: 'upcoming_appointments' },
                    { text: 'Переглянути всі консультації', callback_data: 'all_appointments' }
                ]
            ]
        }
    };

    bot.sendMessage(chatId, 'Оберіть потрібну опцію:', options);
});

bot.onText(/\/statistic/, async (msg) => {
```

```
    const chatId = msg.chat.id;

    try {
        const [user] = await findUserByChatId(chatId);
        if (!user) {
            return bot.sendMessage(chatId, 'Не вдалося знайти користувача.');
        }

        const appointments = await getAppointmentsByUser(user.user_id);

        const chartBuffer = await generateStatisticsChart(appointments);

        const tempFilePath = './chart.png';
        fs.writeFileSync(tempFilePath, chartBuffer);

        await bot.sendPhoto(chatId, tempFilePath, {
            caption: 'Ваша статистика записів за останні 7 днів',
        });

        fs.unlinkSync(tempFilePath);
    } catch (error) {
        console.error('Помилка при отриманні статистики:', error);
        bot.sendMessage(chatId, 'Сталася помилка під час отримання статистики. Спробуйте пізніше.');
    }
});

bot.on('callback_query', async (callbackQuery) => {
    const { data, message } = callbackQuery;
    const chatId = message.chat.id;

    try {
        const [user] = await findUserByChatId(chatId);

        // Розпізнавання дії на основі `callback_data`
        if (data.startsWith('view_')) {
            // Натиснуто кнопку перегляду спеціаліста
            const specialistId = data.split('_')[1];
            const [specialist] = await getSpecialists({ id: specialistId });

            if (!specialist) {
                return bot.sendMessage(chatId, 'Спеціаліста не знайдено.');
            }

            const responseText = `Ім'я спеціаліста: ${specialist.name}\nСпеціалізація: ${specialist.specialization}\nДосвід:
${specialist.experience} років\nРейтинг: ${specialist.rating}/5`;

            const options = {
                reply_markup: {
                    inline_keyboard: [
                        [{ text: 'Переглянути вільні віконечка', callback_data: `availability_${specialistId}` }]
                    ]
                }
            };

            await bot.sendMessage(chatId, responseText, options);
        } else if (data.startsWith('availability_')) {
            // Натиснуто кнопку перегляду вільних вікон спеціаліста
            const specialistId = data.split('_')[1];
            const availableSlots = await getAvailableSlotsForNextWeek(specialistId);

            if (availableSlots.length === 0) {
                return bot.sendMessage(chatId, 'На жаль, у цього спеціаліста немає вільних вікон на наступний тиждень.');
            }

            const responseText = availableSlots.map(slot => `Дата: ${slot.date}, Час: ${slot.time}`).join('\n');
            bot.sendMessage(chatId, `Вільні віконечка спеціаліста:\n${responseText}`);
        } else if (data === 'upcoming_appointments') {
            // Натиснуто кнопку перегляду майбутніх консультацій
            const appointments = await getAppointmentsByUser(user.user_id);
            const futureAppointments = appointments.filter(a => dayjs(a.appointment_datetime).isSameOrAfter(dayjs()));
```

```
        const specialists = await getSpecialists({ id: appointments.map(a => a.specialist_id) });

        if (futureAppointments.length === 0) {
          return bot.sendMessage(chatId, 'У вас немає майбутніх консультацій.');
        }

        const responseText = futureAppointments.map(app => `Дата: ${app.appointment_datetime}, Спеціаліст:
${specialists.find(({ id }) => Number(id) === app.specialist_id).name}`).join('\n');
        bot.sendMessage(chatId, `Ваші майбутні консультації:\n${responseText}`);
      } else if (data === 'all_appointments') {
        // Натиснуто кнопку перегляду всіх консультацій
        const appointments = await getAppointmentsByUser(user.user_id);
        const specialists = await getSpecialists({ id: appointments.map(a => a.specialist_id) });

        if (appointments.length === 0) {
          return bot.sendMessage(chatId, 'У вас немає записів про консультації.');
        }

        const responseText = appointments.map(app => `Дата: ${app.appointment_datetime}, Спеціаліст: ${specialists.find(({ id })
=> Number(id) === app.specialist_id).name}`).join('\n');
        bot.sendMessage(chatId, `Всі ваші консультації:\n${responseText}`);
      } else if (data.startsWith('rate_')) {
        const appointmentId = data.split('_')[1];

        // Надсилання першого питання
        bot.sendMessage(chatId, 'Оцініть ефективність консультації (1-5):', {
          reply_markup: {
            inline_keyboard: [
              [1, 2, 3, 4, 5].map((value) => ({
                text: value.toString(),
                callback_data: `effectiveness_${appointmentId}_${value}`,
              })),
            ],
          },
        });
      } else if (data.startsWith('effectiveness_')) {
        const [_, appointmentId, value] = data.split('_');
        const effectiveness = parseInt(value, 10);

        // Збереження ефективності в пам'ять або базу
        await saveRatingStep(chatId, appointmentId, { effectiveness });

        bot.sendMessage(chatId, 'Оцініть комунікацію (1-5):', {
          reply_markup: {
            inline_keyboard: [
              [1, 2, 3, 4, 5].map((value) => ({
                text: value.toString(),
                callback_data: `communication_${appointmentId}_${value}`,
              })),
            ],
          },
        });
      } else if (data.startsWith('communication_')) {
        const [_, appointmentId, value] = data.split('_');
        const communication = parseInt(value, 10);

        // Збереження комунікації
        await saveRatingStep(chatId, appointmentId, { communication });

        bot.sendMessage(chatId, 'Оцініть пунктуальність (1-5):', {
          reply_markup: {
            inline_keyboard: [
              [1, 2, 3, 4, 5].map((value) => ({
                text: value.toString(),
                callback_data: `punctuality_${appointmentId}_${value}`,
              })),
            ],
          },
        });
      } else if (data.startsWith('punctuality_')) {
```

```
            const [_, appointmentId, value] = data.split('_');
            const punctuality = parseInt(value, 10);

            // Збереження пунктуальності та завершення оцінки
            await saveRatingStep(appointmentId, { punctuality });

            bot.sendMessage(chatId, 'Дякуємо за оцінку консультації!');
        } else if (data.startsWith('skip_')) {
            bot.sendMessage(chatId, 'Оцінку пропущено. Дякуємо!');
        }
        else {
            bot.sendMessage(chatId, 'Невідома опція. Спробуйте ще раз.');
        }

        // Видалення кнопок після виконання дії
        // bot.editMessageReplyMarkup({ inline_keyboard: [] }, { chat_id: chatId, message_id: message.message_id });

    } catch (error) {
        bot.sendMessage(chatId, 'Сталася помилка під час обробки запиту.');
        console.error(error);
    }
});


const scheduleTestReminders = () => {
    console.log("Тестуємо нагадування за годину...");
    testAppointments.forEach((appointment) => {
        const reminderTime = new Date(appointment.appointment_datetime.getTime() - 60 * 60 * 1000); // За годину
        console.log(reminderTime, reminderTime >= new Date(), 'shedule.js', 96);
        if (reminderTime > new Date()) {
            schedule.scheduleJob(reminderTime, async () => {
                try {
                    const message = `
                        Нагадуємо про консультацію, яка відбудеться через годину з спеціалістом ${appointment.specialist_name}.
                        Посилання на Google Meet:googleMeet
                    `;
                    await bot.sendMessage(appointment.chat_id, message);
                    console.log(`Нагадування відправлено для консультації ID: ${appointment.id}`);
                } catch (error) {
                    console.error(`Помилка під час надсилання нагадування для ID: ${appointment.id}`, error);
                }
            });
        }
    });
};

const scheduleTestRatings = () => {
    console.log("Тестуємо запити на оцінку...");
    testAppointments.forEach((appointment) => {
        const ratingTime = new Date(appointment.appointment_datetime.getTime() + 5 * 60 * 1000); // Через 5 хвилин після
        if (ratingTime > new Date()) {
            schedule.scheduleJob(ratingTime, async () => {
                try {
                    const message = `
                        Консультація завершилась. Чи хочете оцінити консультацію?
                        Якщо так, натисніть кнопку "Оцінити".
                    `;

                    const options = {
                        reply_markup: {
                            inline_keyboard: [
                                [
                                    { text: 'Оцінити', callback_data: `rate_${appointment.id}` },
                                    { text: 'Пропустити', callback_data: `skip_${appointment.id}` },
                                ],
                            ],
                        },
                    };

                    await bot.sendMessage(appointment.chat_id, message, options);
```

```
                console.log(`Запит на оцінку відправлено для консультації ID: ${appointment.id}`);
            } catch (error) {
                console.error(`Помилка під час надсилання запиту на оцінку для ID: ${appointment.id}`, error);
            }
        });
    }
  });
};

const saveRatingStep = async (chatId, appointmentId, stepData) => {
  console.log(`Збереження оцінки для ${appointmentId}:`, stepData);
};
const startSchedulers = () => {
  schedule.scheduleJob('* * * * *', async () => {
    console.log("Оновлення нагадувань...");
    await scheduleTestReminders();
    console.log("Оновлення запитів оцінок...");
    await scheduleTestRatings();
  });

  // Початкове виконання
  scheduleTestReminders();
  scheduleTestRatings();
};

startSchedulers();

module.exports = { bot };

const dayjs = require('dayjs');
const { Chart, registerables } = require('chart.js');
Chart.register(...registerables);

const { createCanvas } = require('canvas');

const generateStatisticsChart = async (appointments) => {
  const canvasWidth = 800;
  const canvasHeight = 400;
  const padding = 50;

  const canvas = createCanvas(canvasWidth, canvasHeight);
  const ctx = canvas.getContext('2d');

  const stats = Array(7).fill(0);
  const days = ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Нд'];
  const today = dayjs();

  appointments.forEach((appointment) => {
    const diff = today.diff(dayjs(appointment.appointment_datetime), 'day');
    if (diff >= 0 && diff < 7) {
      stats[6 - diff] += 1;
    }
  });

  const maxStat = Math.max(...stats);
  const barWidth = (canvasWidth - 2 * padding) / stats.length - 10;

  ctx.fillStyle = '#ffffff';
  ctx.fillRect(0, 0, canvasWidth, canvasHeight);

  ctx.strokeStyle = '#000000';
  ctx.lineWidth = 1;
  ctx.beginPath();
  ctx.moveTo(padding, padding);
  ctx.lineTo(padding, canvasHeight - padding);
  ctx.stroke();

  ctx.beginPath();
  ctx.moveTo(padding, canvasHeight - padding);
  ctx.lineTo(canvasWidth - padding, canvasHeight - padding);
```

```
    ctx.stroke();

    ctx.fillStyle = '#000000';
    ctx.font = '16px Arial';
    days.forEach((day, index) => {
        const x = padding + index * (barWidth + 10) + barWidth / 2;
        ctx.fillText(day, x - 10, canvasHeight - padding + 20);
    });

    stats.forEach((value, index) => {
        const barHeight = (value / maxStat) * (canvasHeight - 2 * padding);
        const x = padding + index * (barWidth + 10);
        const y = canvasHeight - padding - barHeight;

        ctx.fillStyle = 'rgba(54, 162, 235, 0.6)';
        ctx.fillRect(x, y, barWidth, barHeight);


        ctx.fillStyle = '#000000';
        ctx.fillText(value, x + barWidth / 4, y - 10);
    });

    return canvas.toBuffer('image/png');
};


module.exports = {
    generateStatisticsChart
}
```

Frontend

```
// SpecialistCatalogPage.js
import React, {useEffect, useState} from "react";
import SpecialistCard from "./SpecialistCardComponent";
import { CatalogWrapper, Header, Filters, SpecialistList } from "../styled/SpecialistCatalogStyled";
import {useSearchParams} from "react-router-dom";
const SpecialistCatalogPage = () => {
    const [filterSpecialization, setFilterSpecialization] = useState("");
    const [filterGender, setFilterGender] = useState("");
    const [filterExperience, setFilterExperience] = useState(0); // Мінімальний досвід роботи
    const [sort, setSort] = useState("rating");
    const [specialists, setSpecialists] = useState([]);
    const [error, setError] = useState(null);
    const [searchParams] = useSearchParams();

    // eslint-disable-next-line react-hooks/exhaustive-deps
    const buildQueryFromSearchParams = () => {
        const params = new URLSearchParams();
        for (const [key, value] of searchParams.entries()) {
            params.append(key, value);
        }
        return params.toString();
    };

    useEffect(() => {
        // Функція для отримання даних з API
        const fetchSpecialists = async () => {
            try {
                const query = buildQueryFromSearchParams();
                const response = await fetch(`http://localhost:3001/specialists/?${query}`);

                if (!response.ok) {
                    throw new Error('Помилка завантаження спеціалістів');
                }

                const data = await response.json();
                setSpecialists(data); // Зберігаємо дані у локальному стані
            } catch (error) {
                setError(error.message);
```

```
        }
      };

      fetchSpecialists(); // Виклик функції при завантаженні компонента
    }, [buildQueryFromSearchParams]);

    if (error) return <p>{error}</p>;
    const filteredPsychologists = specialists
      .filter((specialist) =>
        (filterSpecialization === "" || specialist.specialization.toLowerCase().includes(filterSpecialization.toLowerCase())) &&
        (filterGender === "" || specialist.gender === filterGender) &&
        specialist.experience >= filterExperience
      )
      .sort((a, b) => (sort === "rating" ? b.rating - a.rating : a.name.localeCompare(b.name)));

    return (
      <CatalogWrapper>
        <Header>Обрати спеціаліста</Header>

        <Filters>
          <select value={filterSpecialization} onChange={(e) => setFilterSpecialization(e.target.value)}>
            <option value="">Всі спеціалізації</option>
            <option value="Управління енергією">Управління енергією</option>
            <option value="Допомога при тривозі та стресі">Допомога при тривозі та стресі</option>
            <option value="Сімейна терапія">Сімейна терапія</option>
            <option value="Вирішення міжособистісних конфліктів">Вирішення міжособистісних конфліктів</option>
            <option value="Управління емоціями">Управління емоціями</option>
            <option value="Терапія залежностей">Терапія залежностей</option>
          </select>

          <select value={filterGender} onChange={(e) => setFilterGender(e.target.value)}>
            <option value="">Обидві статі</option>
            <option value="female">Жінка</option>
            <option value="male">Чоловік</option>
          </select>

          <select value={filterExperience} onChange={(e) => setFilterExperience(Number(e.target.value))}>
            <option value="0">Всі рівні досвіду</option>
            <option value="1">1+ років</option>
            <option value="3">3+ років</option>
            <option value="5">5+ років</option>
            <option value="10">10+ років</option>
          </select>

          <select value={sort} onChange={(e) => setSort(e.target.value)}>
            <option value="rating">Сортувати за рейтингом</option>
            <option value="name">Сортувати за іменем</option>
          </select>
        </Filters>

        <SpecialistList>
          {filteredPsychologists.map((specialist) => (
            <SpecialistCard key={specialist.name} specialist={specialist}/>
          ))}
        </SpecialistList>
      </CatalogWrapper>
    );
};

export default SpecialistCatalogPage;

// SpecialistCard.js
import React, { useState } from "react";
import {
  SpecialistCardWrapper,
  ImageWrapper,
  SpecialistInfo,
  Name,
  Specialty,
  Rating,
```

```
    ReviewsWrapper,
    ReviewsContainer,
    Review,
    ReviewHeader,
    ReviewAuthor,
    ReviewRating,
    Button,
    ScrollLeftButton,
    ScrollRightButton,
} from "../styled/SpecialistCardStyled";
import {Link} from "react-router-dom";

const SpecialistCard = ({ specialist }) => {
    const [currentReviewIndex, setCurrentReviewIndex] = useState(0);

    const handleScrollLeft = () => {
        setCurrentReviewIndex((prevIndex) =>
            prevIndex > 0 ? prevIndex - 1 : specialist.reviews.length - 1
        );
    };

    const handleScrollRight = () => {
        setCurrentReviewIndex((prevIndex) =>
            prevIndex < specialist.reviews.length - 1 ? prevIndex + 1 : 0
        );
    };

    return (
        <SpecialistCardWrapper>
            <ImageWrapper>
                <img src={specialist.profile_image_url} alt={specialist.name} />
            </ImageWrapper>
            <SpecialistInfo>
                <Name>{specialist.name}</Name>
                <Specialty>{specialist.specialization}</Specialty>
                <Rating>Рейтинг: {specialist.rating} ★</Rating>
                <Link to={`/specialist/${specialist.id}`}>
                    <Button>Перейти на сторінку спеціаліста</Button>
                </Link>
            </SpecialistInfo>
            <ReviewsWrapper>
                <ScrollLeftButton onClick={handleScrollLeft}>{"<"}</ScrollLeftButton>
                <ReviewsContainer style={{ transform: `translateX(-${currentReviewIndex * 260}px)` }}>
                    {specialist.reviews?.map((review, index) => (
                        <Review key={index}>
                            <ReviewHeader>
                                <ReviewAuthor>{review.author}</ReviewAuthor>
                            </ReviewHeader>
                            <p>{review.text}</p>
                            <ReviewRating>Оцінка: {review.rating} ★</ReviewRating>
                        </Review>
                    ))}
                </ReviewsContainer>
                <ScrollRightButton onClick={handleScrollRight}>{">"}</ScrollRightButton>
            </ReviewsWrapper>
        </SpecialistCardWrapper>
    );
};

export default SpecialistCard;

export const SET_SPECIALISTS = 'SET_SPECIALISTS';

export function setSpecialists(specialists) {
    return {
        type: SET_SPECIALISTS,
        payload: specialists
    };
}
```

```
export function fetchSpecialists() {
   return async (dispatch) => {
      try {
         // Виконуємо запит до API
         const response = await fetch('http://localhost:3001/specialists/');

         // Перевірка на успішний статус відповіді
         if (!response.ok) {
            throw new Error('Помилка завантаження спеціалістів');
         }

         // Розбираємо відповідь як JSON
         const data = await response.json();

         // Викликаємо екшен для збереження даних у Store
         dispatch(setSpecialists(data));
      } catch (error) {
         console.error("Помилка завантаження спеціалістів:", error);
         // Тут можна реалізувати екшен для обробки помилок, якщо потрібно
      }
   };
}

import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import {
   PageWrapper, Header, SpecialistInfoWrapper, ImageWrapper, SpecialistDetails,
   Specialty, ScheduleSection, ReviewsWrapper, Review, Author, Button,
} from "../styled/SpecialistPageStyled";
import SpecialistRatingComponent from "./SpecialistRating/SpecialistRatingComponent";
import SpecialistSchedule from "./CalendarComponent";
import ConsultationPopup from "../ConsultationPopupComponent";

const SpecialistPage = () => {
   const { id } = useParams();
   const [specialists, setSpecialists] = useState([]);
   const [error, setError] = useState(null);
   const [showPopup, setShowPopup] = useState(false);
   const [unavailableDates, setUnavailableDates] = useState([]);

   useEffect(() => {
      let isMounted = true;

      const fetchData = async () => {
         try {
            const response = await fetch('http://localhost:3001/specialists/');
            if (!response.ok) throw new Error('Помилка завантаження спеціалістів');

            const data = await response.json();
            if (isMounted) {
               setSpecialists(data);
            }

            const appointmentsResponse = await fetch(`http://localhost:3001/calendar/?specialistId=${id}`);
            const appointmentsData = await appointmentsResponse.json();
            if (isMounted) {
               setUnavailableDates(appointmentsData.map(app => new Date(app.appointment_datetime).toDateString()));
            }
         } catch (error) {
            if (isMounted) {
               setError(error.message);
            }
         }
      };

      fetchData();

      return () => {
         isMounted = false;
      };
```

```
  }, [id]);

  if (error) return <p>{error}</p>;

  const specialist = specialists.find((specialist) => specialist.id === parseInt(id));
  if (!specialist) {
    return <p>Спеціаліст не знайдений</p>;
  }

  const handleOpenPopup = () => {
    setShowPopup(true);
  };

  const handleClosePopup = () => {
    setShowPopup(false);
  };

  const handleAppointmentSubmit = (date) => {
    console.log("Обрана дата:", date);
    // Додайте логіку для обробки запису на консультацію
  };

  return (
    <PageWrapper>
      <Header>{specialist.name}</Header>

      <SpecialistInfoWrapper>
        <ImageWrapper>
          <img src={specialist.profile_image_url} alt={specialist.name} />
        </ImageWrapper>

        <SpecialistDetails>
          <Specialty>{specialist.specialization}</Specialty>
          <p>Досвід роботи: {specialist.experience} років</p>
          <SpecialistRatingComponent id={specialist.id} />
        </SpecialistDetails>
      </SpecialistInfoWrapper>

      <Button onClick={handleOpenPopup}>Записатися на консультацію</Button>

      <ScheduleSection>
        <h3>Графік роботи: {specialist.schedule}</h3>
        <SpecialistSchedule specialist={specialist}/>
      </ScheduleSection>

      {showPopup && (
        <ConsultationPopup
          onClose={handleClosePopup}
          onSubmit={handleAppointmentSubmit}
          specialist={specialist}
          unavailableDates={unavailableDates}
        />
      )}

      <ReviewsWrapper>
        <h3>Відгуки про спеціаліста</h3>
        {specialist.reviews.map((review, index) => (
          <Review key={index}>
            <Author>{review.author}</Author>
            <p>{review.text}</p>
            <p>Оцінка: {review.rating} ⭐</p>
          </Review>
        ))}
      </ReviewsWrapper>
    </PageWrapper>
  );
};

export default SpecialistPage;
```

```
import React, { useEffect, useState } from 'react';
import Calendar from 'react-calendar';
import 'react-calendar/dist/Calendar.css';
import styled from 'styled-components';
import dayjs from 'dayjs';
import SubscribeButton from "./SubscribeButton";

const CalendarWrapper = styled.div`
    .legend {
        margin: 10px 0;
        display: flex;
        align-items: center;
    }
    .legend span {
        display: inline-flex;
        align-items: center;
        gap: 5px;
    }
    .legend .color-box {
        width: 15px;
        height: 15px;
        border-radius: 3px;
        display: inline-block;
    }
    .occupied-date .color-box {
        background-color: #ffcccc;
    }
    .partially-occupied-date .color-box {
        background-color: #ffe0b3;
    }
    .working-date .color-box {
        background-color: #ccffcc;
    }
    .subscribe-button {
        background-color: #0088cc;
        color: white;
        padding: 10px 20px;
        border: none;
        border-radius: 4px;
        font-size: 16px;
        cursor: pointer;
        text-align: center;
        margin-top: 20px;
        &:hover {
            background-color: #005f99;
        }
    }
`;

const HourTab = styled.div`
    display: inline-block;
    margin: 5px;
    padding: 10px;
    border-radius: 5px;
    background-color: ${props => (props.isOccupied ? '#d3d3d3' : '#ccffcc')};
    color: ${props => (props.isOccupied ? 'grey' : 'black')};
    border: 1px solid #ccc;
    cursor: ${props => (props.isOccupied ? 'not-allowed' : 'pointer')};
    text-align: center;
    width: 50px;
`;

const SpecialistSchedule = ({ specialist, handleDateChange }) => {
    const [appointments, setAppointments] = useState([]);
    const [workingDays, setWorkingDays] = useState([]);
    const [date, setDate] = useState(dayjs());
    const [userId, setUserId] = useState(null);
    const [selectedDayHours, setSelectedDayHours] = useState([]);
    const specialistId = specialist.id;
```

```
useEffect(() => {
    const storedUserId = localStorage.getItem('userId');
    if (storedUserId) {
        setUserId(storedUserId);
    }
}, []);

useEffect(() => {
    const fetchAppointments = async () => {
        try {
            const response = await fetch(`http://localhost:3001/calendar/?specialistId=${specialistId}`);
            const data = await response.json();
            setAppointments(data);
            if (specialist.schedule) {

                const workingDaysArray = parseSchedule(specialist.schedule);
                setWorkingDays(workingDaysArray);
            }
        } catch (error) {
            console.error('Помилка при завантаженні графіку:', error);
        }
    };
    if (specialistId) fetchAppointments();
}, [specialistId, specialist.schedule]);

const parseSchedule = (schedule) => {
    const daysMap = { 'Пн': 1, 'Вт': 2, 'Ср': 3, 'Чт': 4, 'Пт': 5, 'Сб': 6, 'Нд': 7 };
    const [days, time] = schedule.split(': ');
    const [startTime, endTime] = time.split('-').map(t => parseInt(t.split(':')[0], 10));
    const dayRanges = days.split('-').map(day => daysMap[day]);

    const workingHours = [];
    for (let i = dayRanges[0]; i <= dayRanges[1]; i++) {
        workingHours.push({
            day: i,
            hours: Array.from({ length: endTime - startTime + 1 }, (_, j) => startTime + j),
        });
    }
    return workingHours;
};

const handleDateClick = (selectedDate) => {
    const dayOfWeek = dayjs(selectedDate).day() || 7;
    handleDateChange(selectedDate);
    const workingHoursForDay = workingDays.find(day => day.day === dayOfWeek);

    if (workingHoursForDay) {
        const appointmentsForDay = appointments
            .filter(appointment => dayjs(appointment.appointment_datetime).isSame(dayjs(selectedDate), 'day'))
            .map(appointment => dayjs(appointment.appointment_datetime).hour());

        const hoursWithStatus = workingHoursForDay.hours.map(hour => ({
            hour,
            isOccupied: appointmentsForDay.includes(hour),
        }));
        setSelectedDayHours(hoursWithStatus);
    } else {
        setSelectedDayHours([]);
    }
};

return (
    <CalendarWrapper>
        <div className="legend">
            <span><span className="color-box occupied-date"></span> - Повністю зайняті дати</span>
            <span><span className="color-box partially-occupied-date"></span> - Частково зайняті дати</span>
            <span><span className="color-box working-date"></span>☐ - Робочі дати</span>
        </div>
```

```
        <Calendar
          onChange={(date) => {
            setDate(dayjs(date));
            handleDateClick(date);
          }}
          value={date.toDate()}
          tileClassName={({ date, view }) => {
            if (view === 'month') {
              const currentDay = dayjs(date);
              if (appointments.some(appt => dayjs(appt.appointment_datetime).isSame(currentDay, 'day'))) {
                return appointments.filter(appt => dayjs(appt.appointment_datetime).isSame(currentDay, 'day')).length > 8
                  ? 'occupied-date'
                  : 'partially-occupied-date';
              }
              if (workingDays.some(day => day.day === currentDay.day())) {
                return 'working-date';
              }
            }
            return null;
          }}
          tileDisabled={({ date }) => dayjs(date).isBefore(dayjs(), 'day')}
        />
        {selectedDayHours.length > 0 && (
          <div>
            <h4>Робочі години у цей день</h4>
            <div style={{ display: 'flex', flexWrap: 'wrap' }}>
              {selectedDayHours.map(({ hour, isOccupied }, index) => (
                <HourTab key={index} isOccupied={isOccupied}>
                  {hour}:00
                </HourTab>
              ))}
            </div>
          </div>
        )}
        {userId && <SubscribeButton specialistId={specialistId} userId={userId} />}
      </CalendarWrapper>
    );
};

export default SpecialistSchedule;

import React, { useState, useEffect } from 'react';
import PropTypes from 'prop-types';
import styled from 'styled-components';
import RatingMessage from './RatingMessageComponent';
import RatingItemsComponent from './RatingItemsComponent';

const RatingWrapper = styled.div`
  >div:first-child {
    display: flex;
    align-items: center;
    gap: 16px;
  }
  @media (max-width: 1024px) {
    margin: ${(props) => (props.hasRating ? '38px 0' : 0)};
    border: 1px solid #E1EAF1;
    border-radius: 12px;
    padding: 24px 12px;
    box-shadow: 0 1px 2px 0 #1018280F, 0 1px 3px 0 #1018281A;
  }
`;

const SpecialistRatingComponent = ({ id }) => {
  const [rating, setRating] = useState({});
  useEffect(() => {
    const fetchRating = async () => {
      try {
        const response = await fetch(`http://localhost:3001/rating/?id=${Number(id)}`);
        const data = await response.json();
        setRating(data);
```

```
      } catch (error) {
        console.error('Помилка при завантаженні рейтингу:', error);
      }
    };

    if (id) fetchRating();
  }, [id]);
  if (!Object.keys(rating).length) return null;
  const { rating: ratingScore, ratingDetails } = rating;

  return (
      <RatingWrapper>
        <div role="button" data-tm="show-rating">
          <RatingMessage ratingScore={ratingScore} />
        </div>
        <RatingItemsComponent ratingDetails={ratingDetails}/>
      </RatingWrapper>
  );
};

SpecialistRatingComponent.propTypes = {
  id: PropTypes.number.isRequired,
};

export default SpecialistRatingComponent;

import React from 'react';
import styled from 'styled-components';
import RatingSvg from './svg/RatingSvgComponent';

const Wrapper = styled.div`
    display: flex;
    align-items: center;
    gap: 12px;
`;

const Message = styled.div`
  font-size: 16px;
  line-height: 20px;
  font-weight: 500;
  color: ${(props) => (props.ratingScore >= 3 ? '#1A2638' : '#62718A')};
  @media (max-width: 1024px) {
    font-size: 14px;
    line-height: 17px;
  }
`;

const Label = styled.div`
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 5px;
  padding: 6px 12px;
  width: 61px;
  background-color: ${(props) => (props.ratingScore >= 3 ? '#7FC700' : '#98A2B3')};
  color: #FFFFFF;
  border-radius: 0 0 0 12px;
  font-size: 18px;
  font-weight: 700;
  line-height: 22px;
  text-align: center;
`;

const RatingMessageComponent = ({ ratingScore }) => {
  const showRatingCount = ratingScore ? ratingScore : 0;
  return (
    <Wrapper>
      <Label ratingScore={showRatingCount}>
        {ratingScore < 3 ? null :  <RatingSvg />}
        <span>{`${showRatingCount}`}</span>
```

```
      </Label>
      <Message ratingScore={showRatingCount}>
        Рейтинг спеціаліста {showRatingCount}
      </Message>
    </Wrapper>
  );
};

export default RatingMessageComponent;

import React from 'react';
import PropTypes from 'prop-types';
import styled from 'styled-components';
import InfoSvg from './svg/InfoSvgComponent';
import ProgressBarSvg from './svg/ProgressBarSvgComponent';
import TooltipComponent from './TooltipComponent';

const Wrapper = styled.div`
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    gap: 32px;
    margin-top: 32px;
    @media (max-width: 1024px) {
        display: flex;
        flex-direction: column;
        gap: 24px;
    }
`;

const RatingItem = styled.div`
    display: flex;
    flex-direction: column;
    gap: 8px;
    cursor: pointer;
`;

const ItemInfo = styled.div`
    display: flex;
    align-items: center;
    justify-content: space-between;
    color: #1A2638;
    font-weight: 500;
    line-height: 20px;
    font-size: 16px;
    >div:first-child {
        font-size: 14px;
        display: flex;
        gap: 4px;
        align-items: center;
    }
`;

const RatingItemsComponent = ({ ratingDetails }) => {
  return (
    <Wrapper>
      {Object.entries(ratingDetails).map(([type, value]) => (
        <TooltipComponent tooltip={itemsInfo[type].description}>
          <RatingItem role="button">
            <ItemInfo>
              <div>
                <span>{itemsInfo[type].name}</span>
                <InfoSvg />
              </div>
              <span>{value}/5</span>
            </ItemInfo>
            <ProgressBarSvg percentage={value} />
          </RatingItem>
        </TooltipComponent>
      ))}
    </Wrapper>
```

```
  )
};

RatingItemsComponent.propTypes = {
  ratingDetails: PropTypes.shape({}).isRequired,
};

import React, { useState } from 'react';
import styled from 'styled-components';

// Стилі для блоку
const SubscribeContainer = styled.div`
    display: flex;
    align-items: center;
    justify-content: space-between;
    background-color: #f5faff;
    border-radius: 8px;
    padding: 16px;
    border: 1px solid #e0e0e0;
    max-width: 600px;
    margin: 16px auto;
    box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.1);
`;

const ContentContainer = styled.div`
    display: flex;
    flex-direction: column;
    justify-content: center;
`;

const Title = styled.h3`
    font-size: 18px;
    font-weight: bold;
    color: #333;
    margin: 0 0 8px 0;
`;

const Description = styled.p`
    font-size: 14px;
    color: #666;
    margin: 0 0 16px 0;
`;

const Button = styled.button`
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 4px;
    padding: 10px 16px;
    font-size: 14px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    align-self: flex-start;
    &:hover {
        background-color: #0056b3;
    }
    &:disabled {
        background-color: #ccc;
        cursor: not-allowed;
    }
`;
const Image = styled.img`
    width: 80px;
    height: 80px;
    object-fit: contain;
`;
const SubscribeButton = ({ userId, specialistId }) => {
    const [isSubscribed, setIsSubscribed] = useState(false);
    const handleSubscribe = async () => {
        try {
```

```
            const response = await fetch('http://localhost:3001/subscribe', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ userId, specialistId }),
            });
            if (response.ok) {
                setIsSubscribed(true);
                window.open(`https://t.me/ozerovaSoulBot/?start=${userId}_${specialistId}`, '_blank');
            } else {
                console.error('Не вдалося підписатися на Telegram-бота');
            }
        } catch (error) {
            console.error('Помилка при підписці на Telegram-бота:', error);
        }
    };

    return (
        <SubscribeContainer>
            <ContentContainer>
                <Title>Підписатися на Telegram-бота</Title>
                <Description>Отримуйте актуальні новини та повідомлення від спеціаліста в Telegram.</Description>
                <Button onClick={handleSubscribe} disabled={isSubscribed}>
                    {isSubscribed ? 'Ви підписані' : 'Підписатися'}
                </Button>
            </ContentContainer>
            <Image src="telegram-logo.png" alt="Telegram Logo" />
        </SubscribeContainer>
    );
};
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { createStore } from 'redux';
import { Provider } from 'react-redux';
import rootReducer from './reducer/rootReducers';

const store = createStore(rootReducer);

const container = document.getElementById('root');
const root = ReactDOM.createRoot(container);
root.render(
    <Provider store={store}>
        <React.StrictMode>
            <App />
        </React.StrictMode>
    </Provider>
);
```

ДОДАТОК Г

Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ВЕБПОРТАЛУ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ

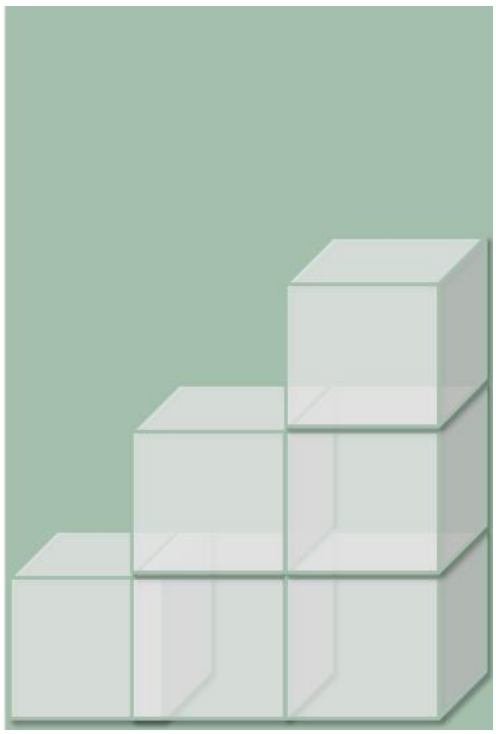

Рисунок Г.1 – Титульний слайд

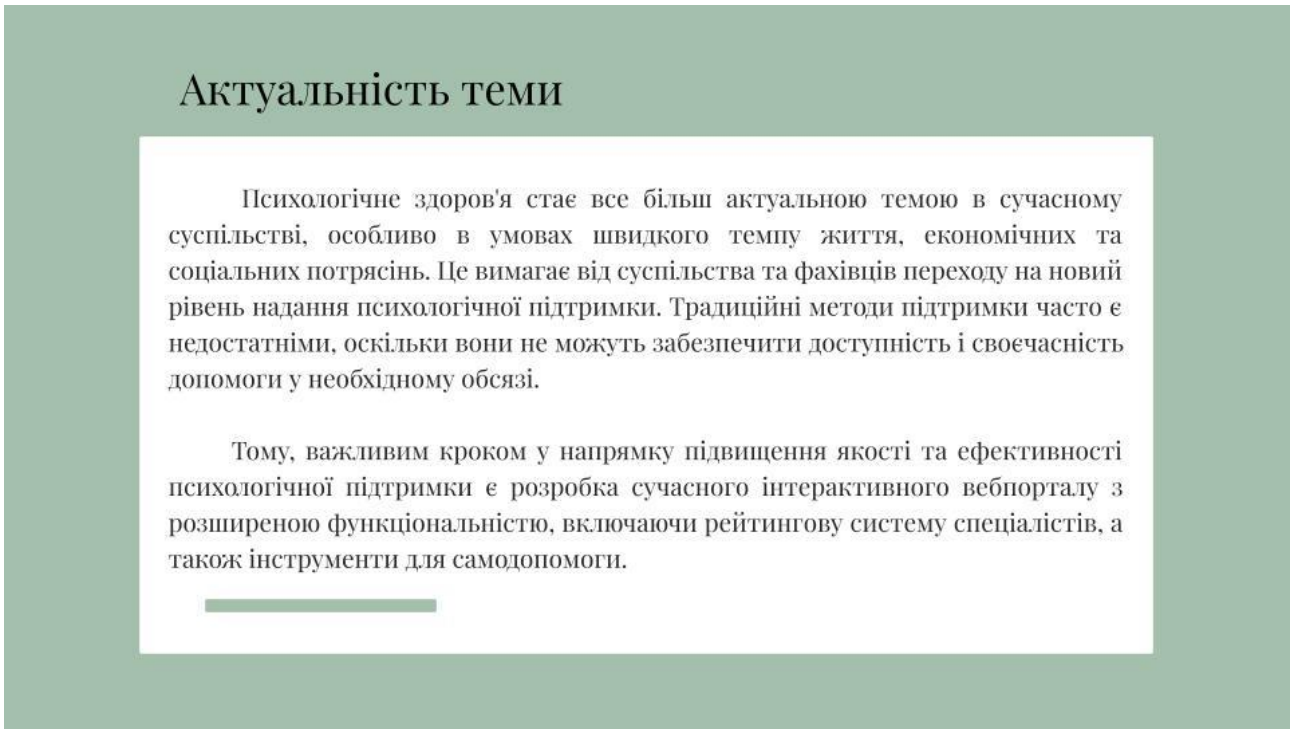

Рисунок Г.2 – Актуальність теми

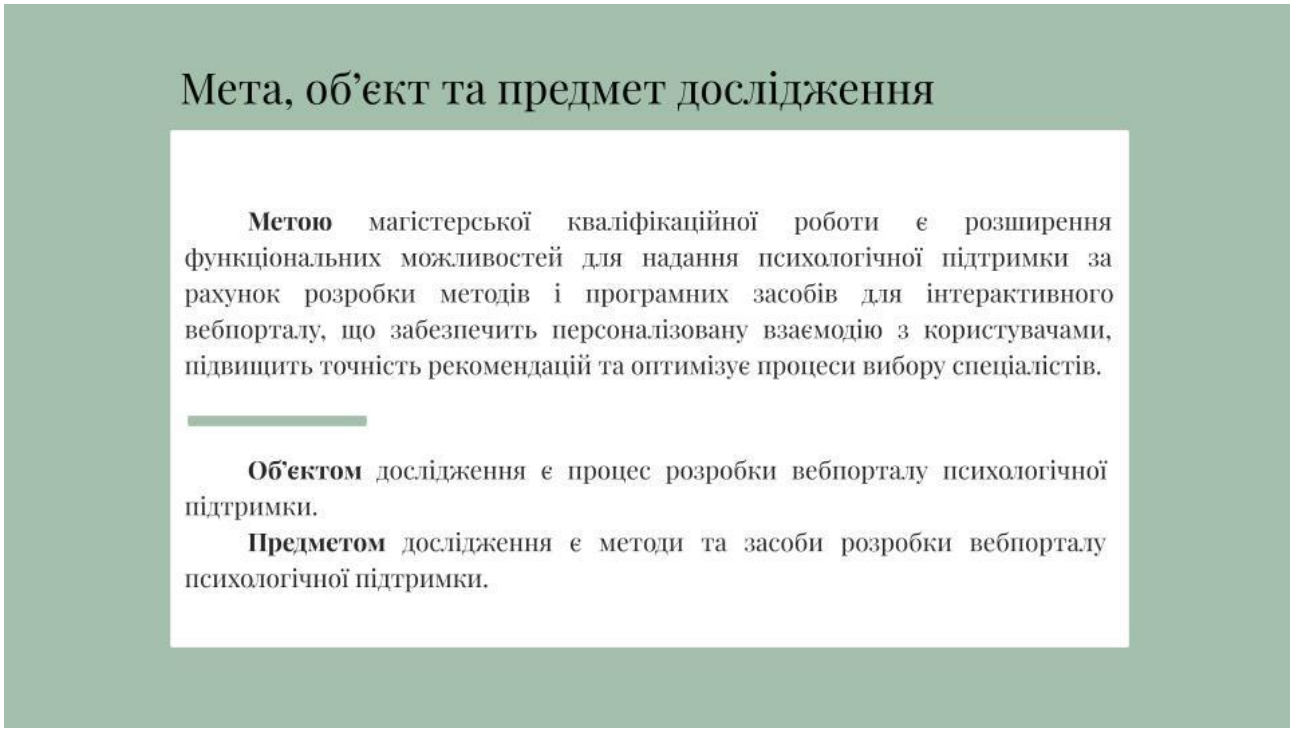

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

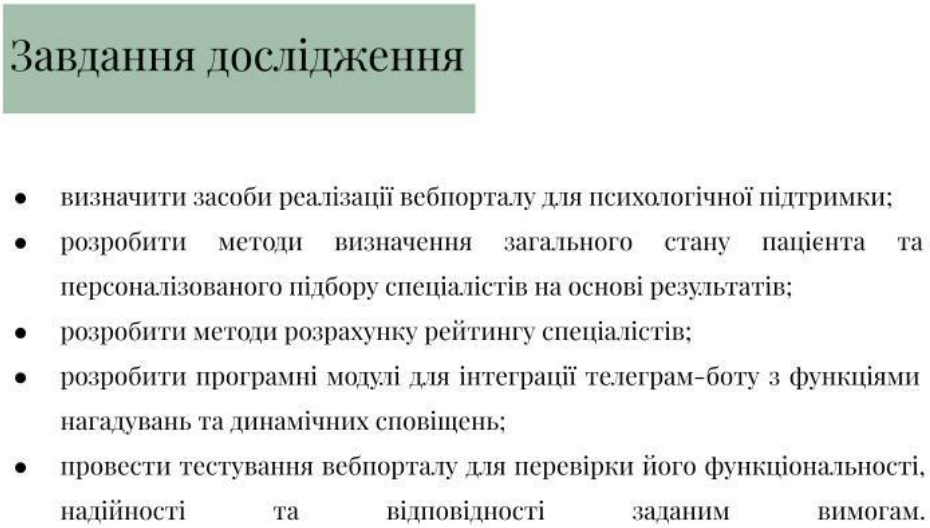

Рисунок Г.4 – Завдання дослідження

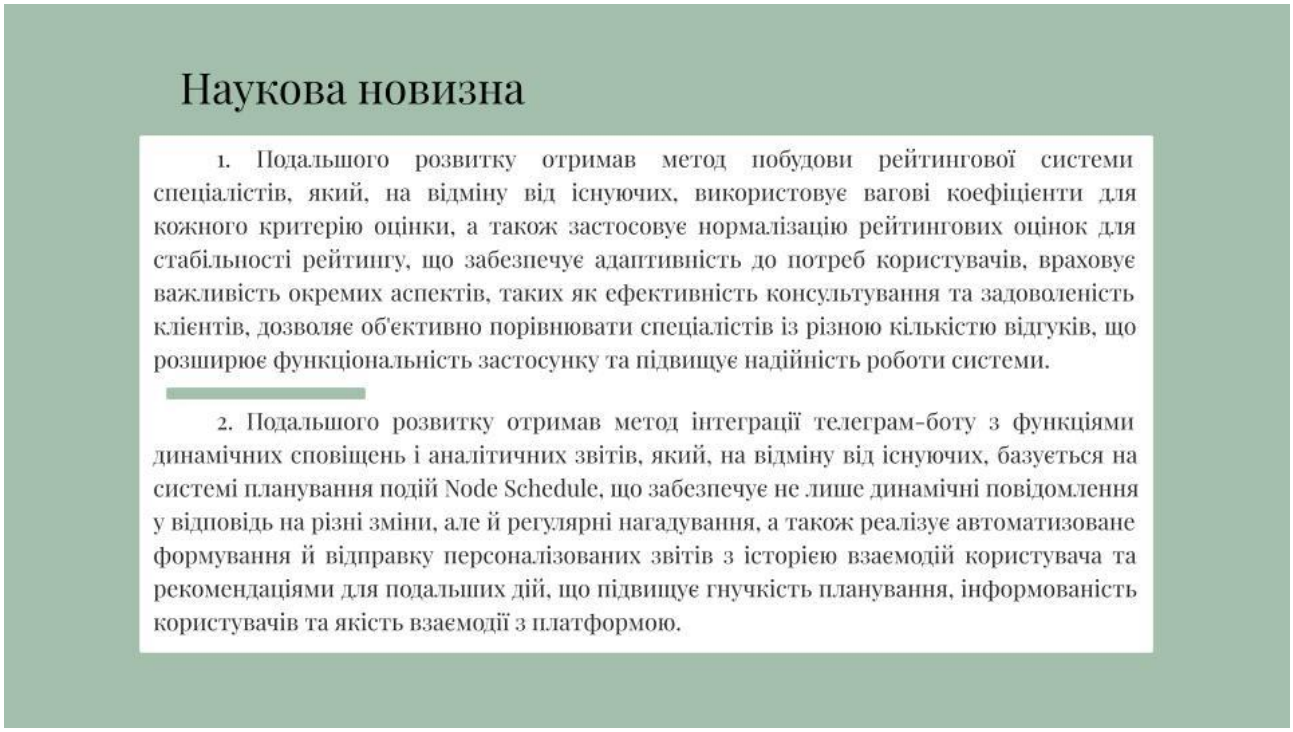

Рисунок Г.5 – Наукова новизна

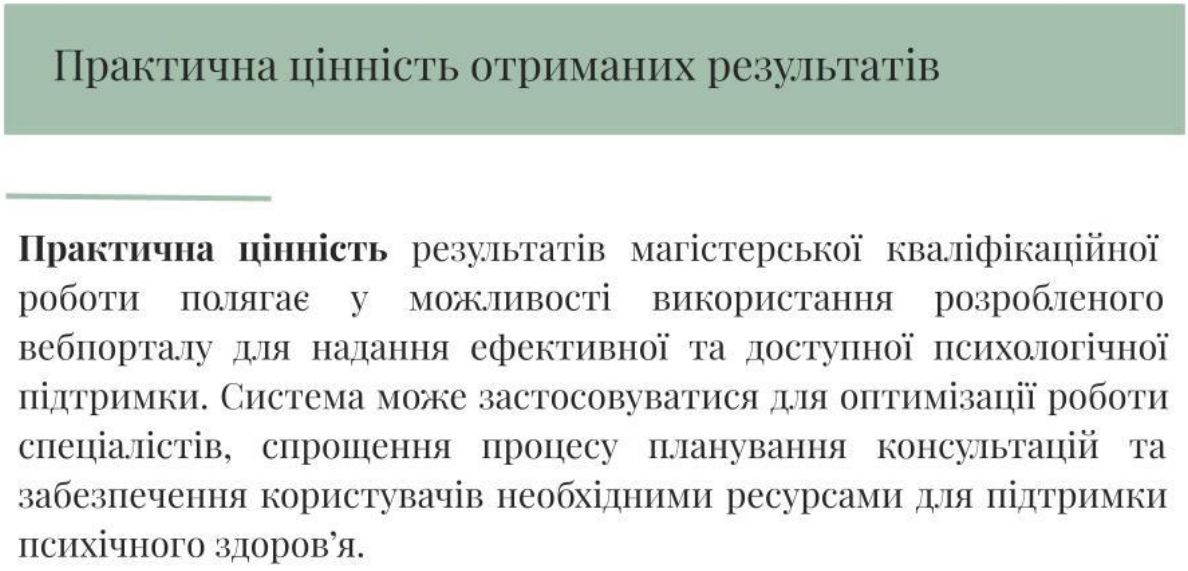

Рисунок Г.6 – Практична цінність

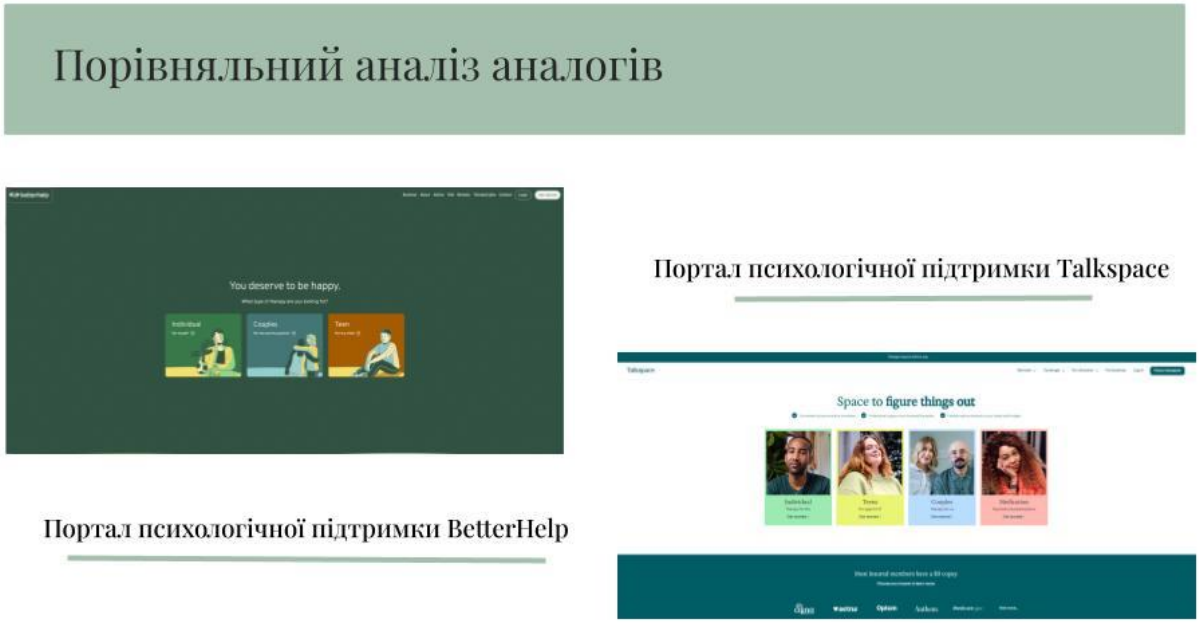

Рисунок Г.7 – Аналоги (частина 1)

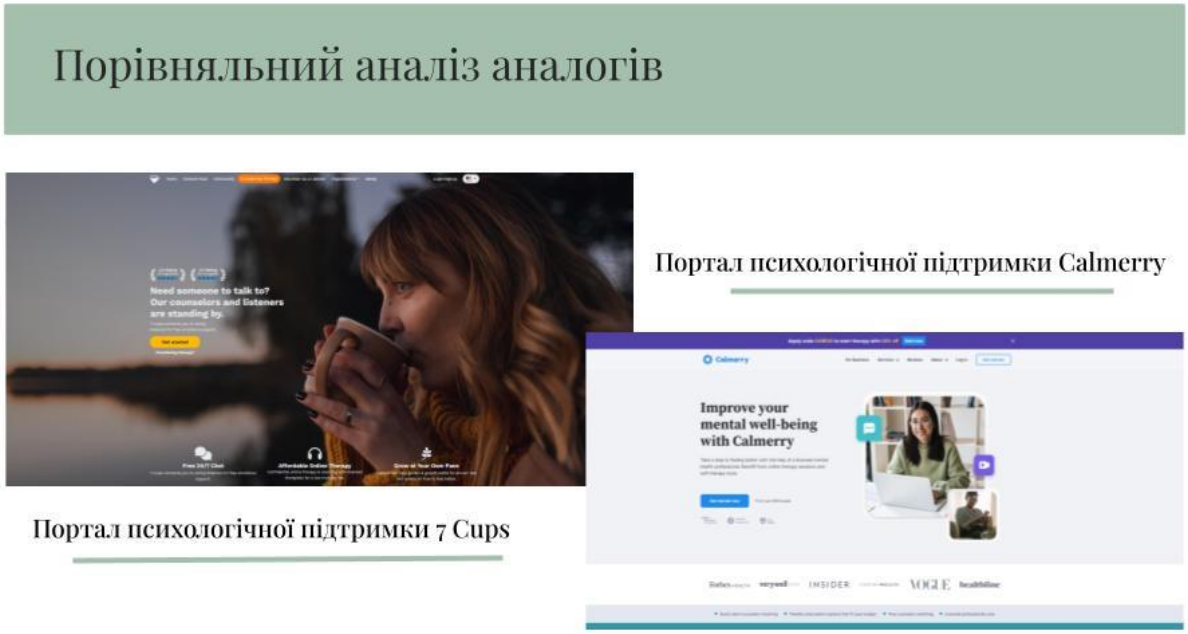

Рисунок Г.8 – Аналоги (частина 2)

Порівняльний аналіз аналогів

Характеристика	BetterHelp	Talkspace	7 Cups	Calmerry	Soul Talk
Наявність безкоштовної підтримки	-	-	+	-	+
Можливість вибору психолога	+	+	+	+	+
Доступність 24/7	-	-	+	-	+
Швидкість зміни психолога	+	+	-	+	+
Результат	2	2	3	2	4

Рисунок Г.9 – Порівняння аналогів

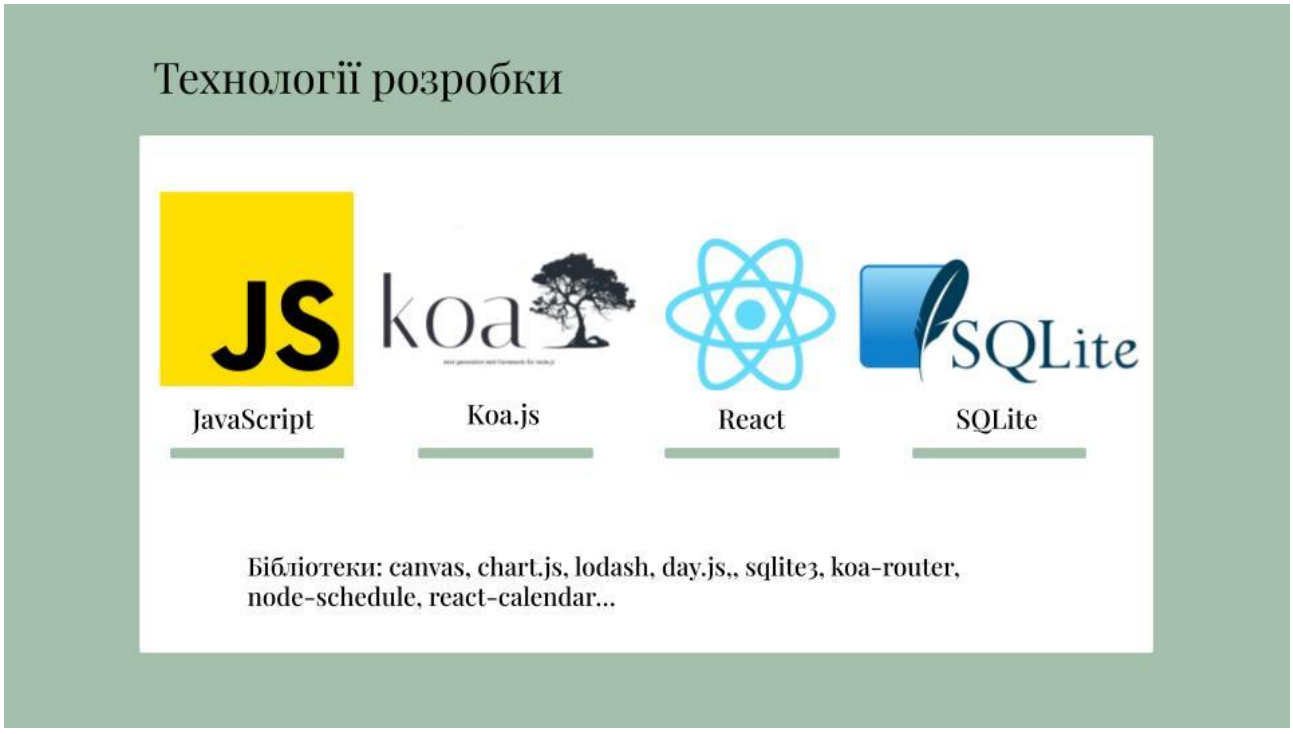

Рисунок Г.10 – Технології розробки

Рисунок Г.11 – Загальна модель порталу

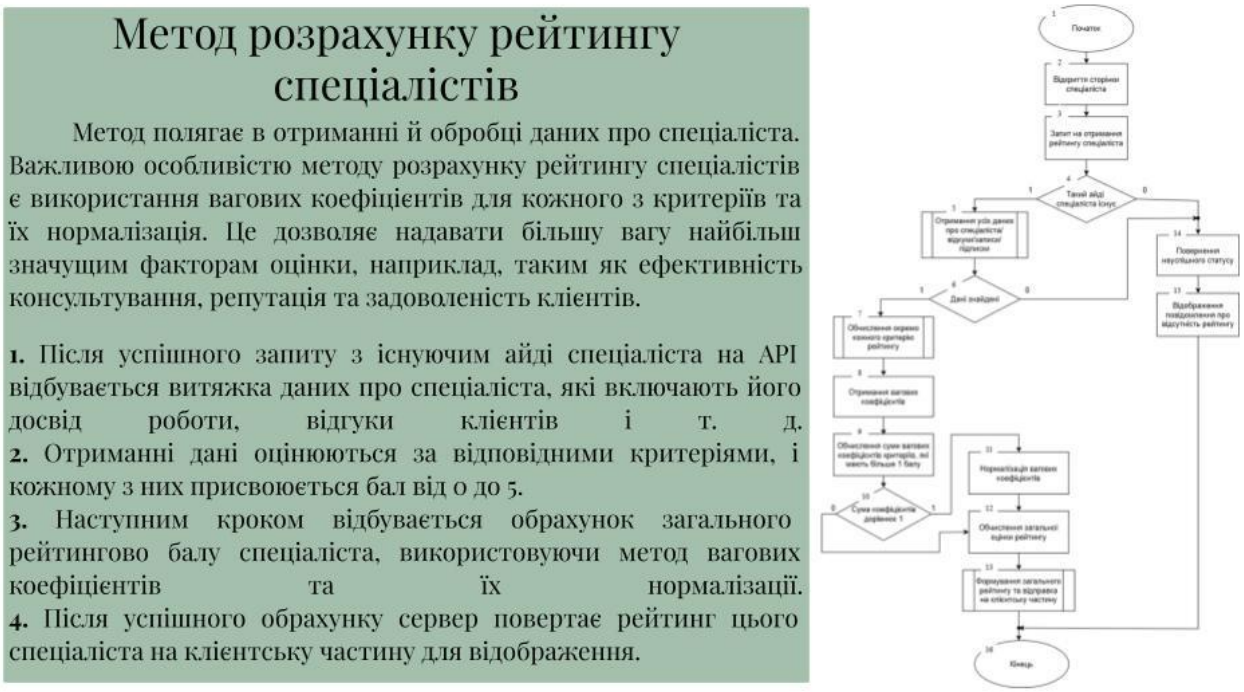

Рисунок Г.12 – Метод розрахунку рейтингу спеціалістів

Рисунок Г.13 – Метод інтеграції Телеграм-боту

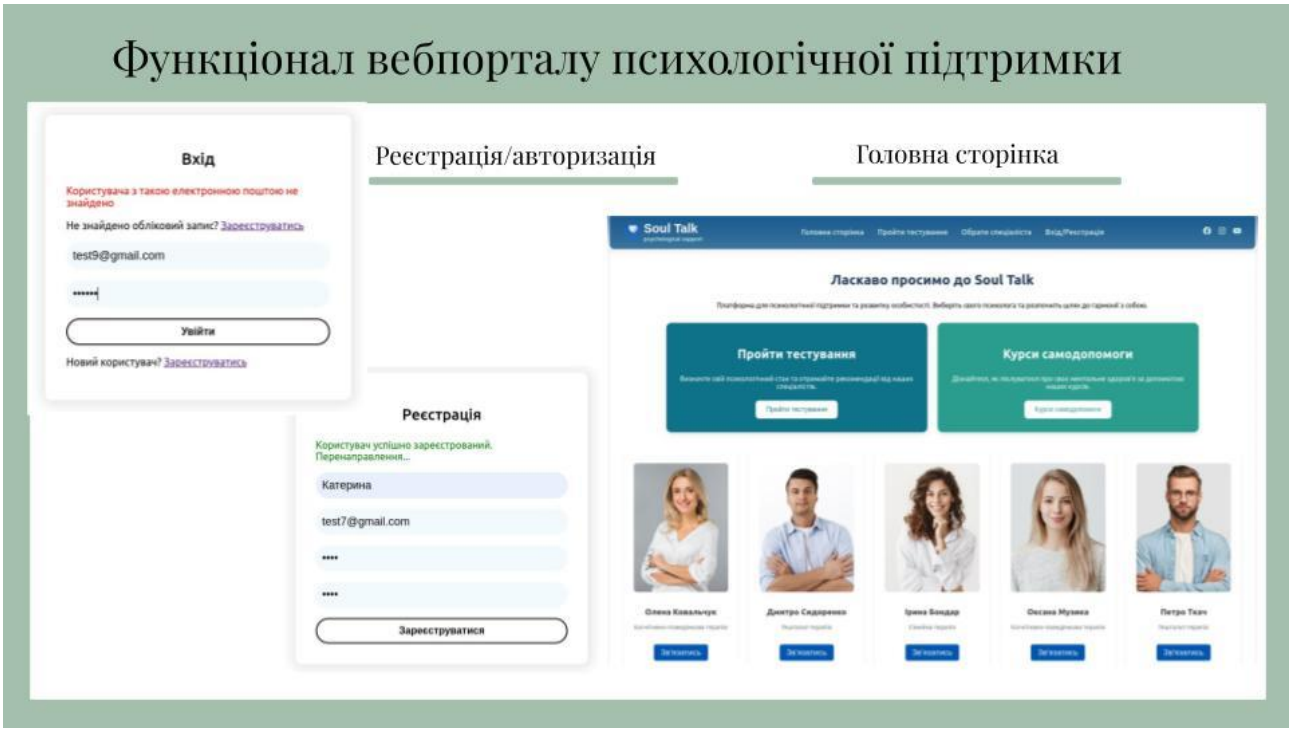

Рисунок Г.14 – Тестування програми (частина 1)

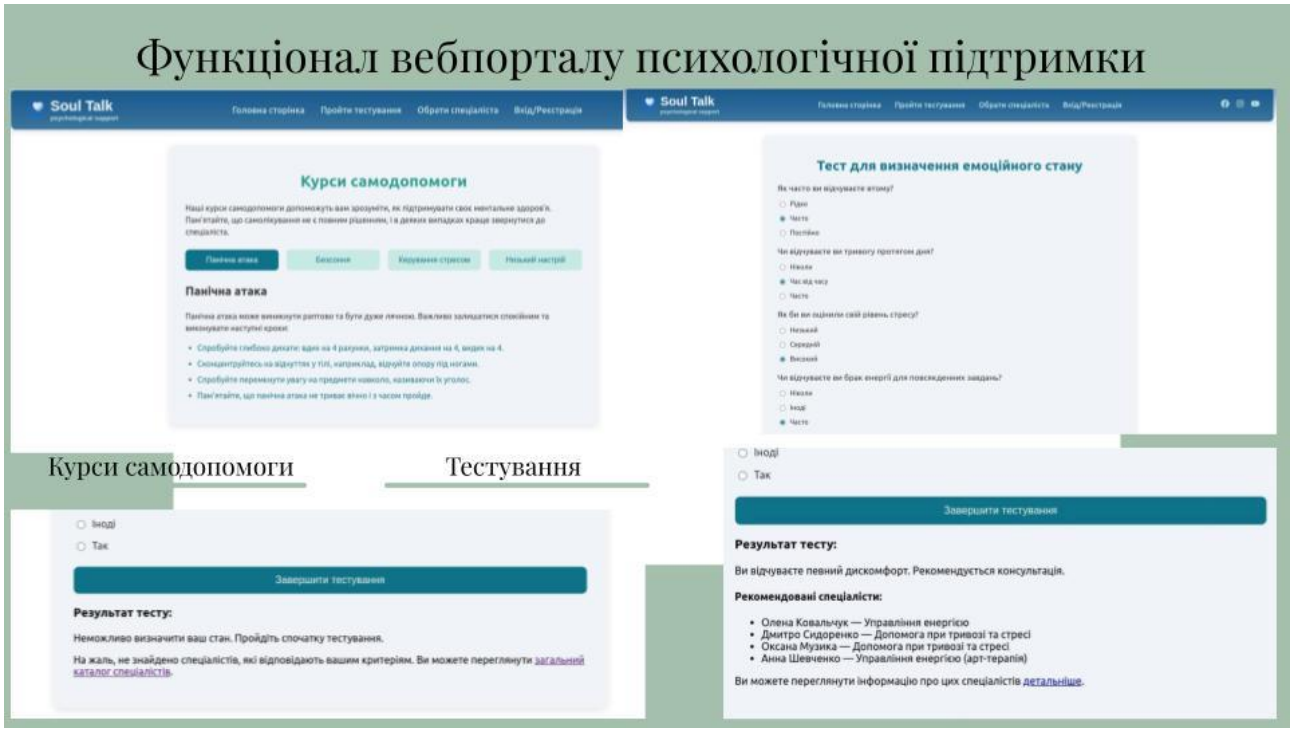

Рисунок Г.15 – Тестування програми (частина 2)

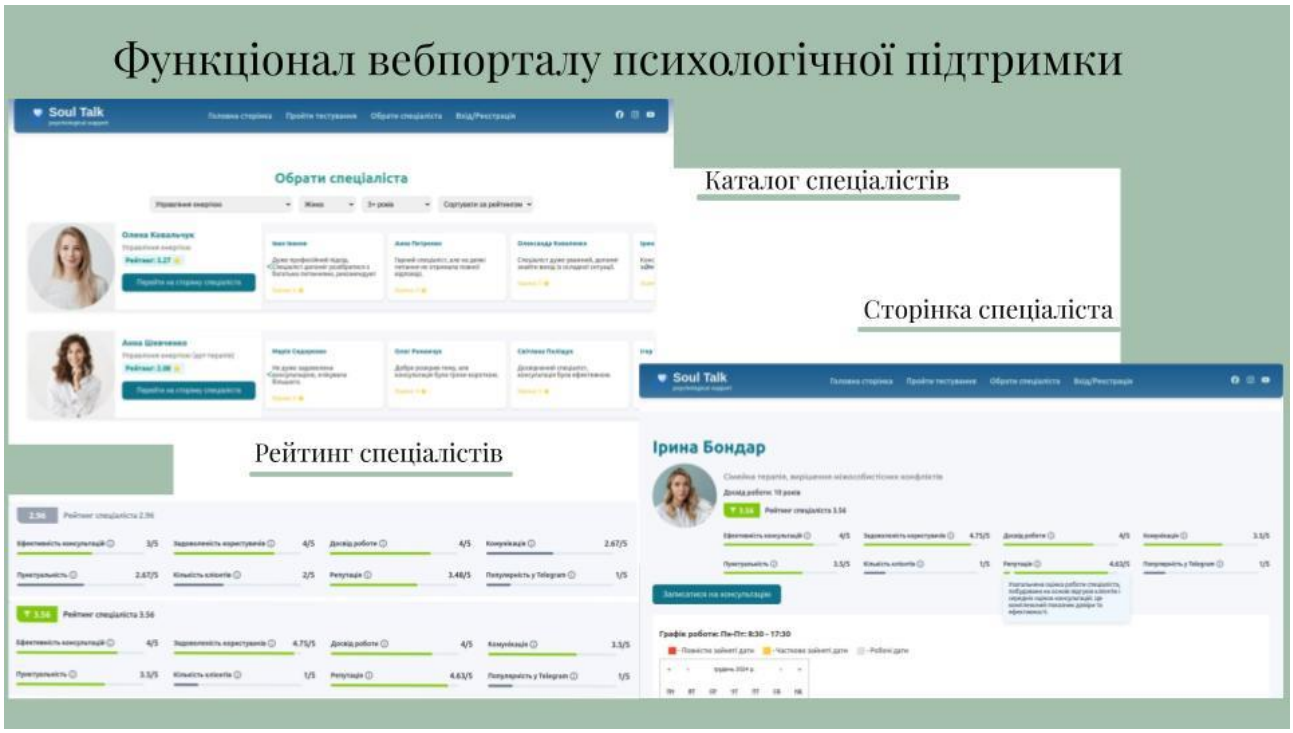

Рисунок Г.16 – Тестування програми (частина 3)

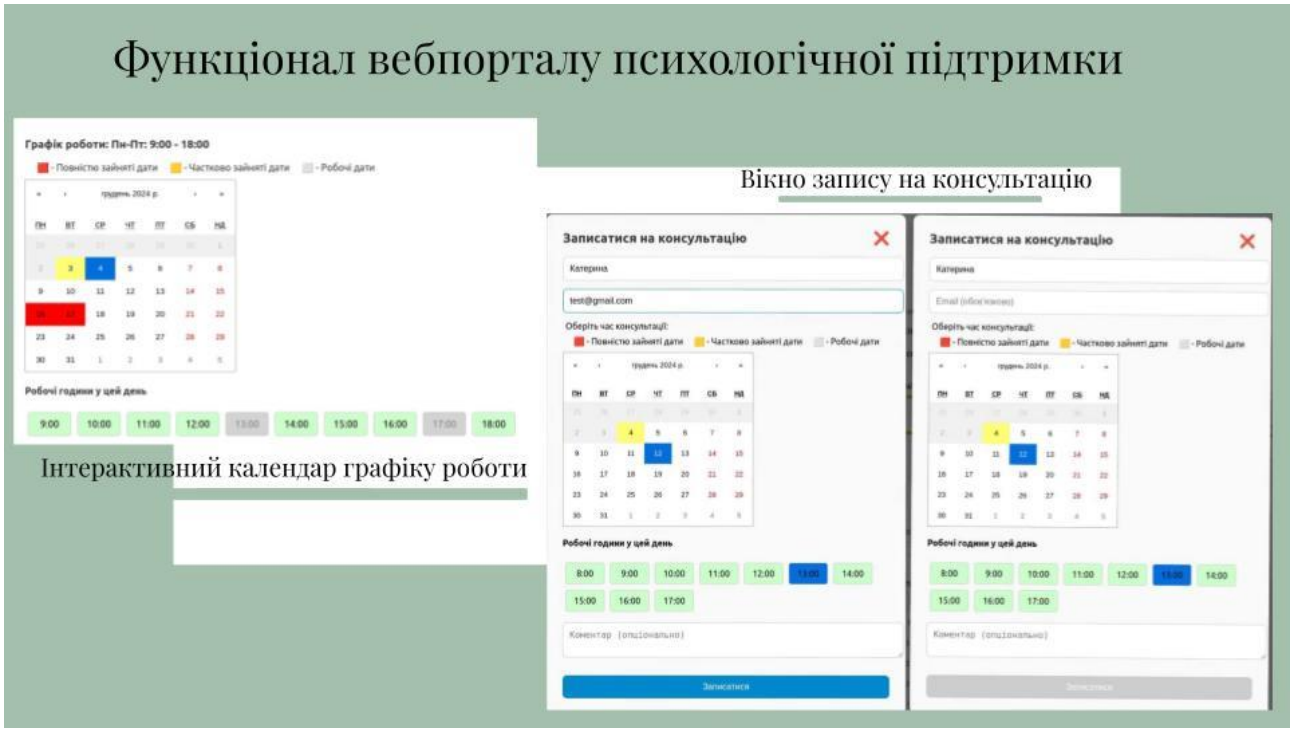

Рисунок Г.17 – Тестування програми (частина 4)

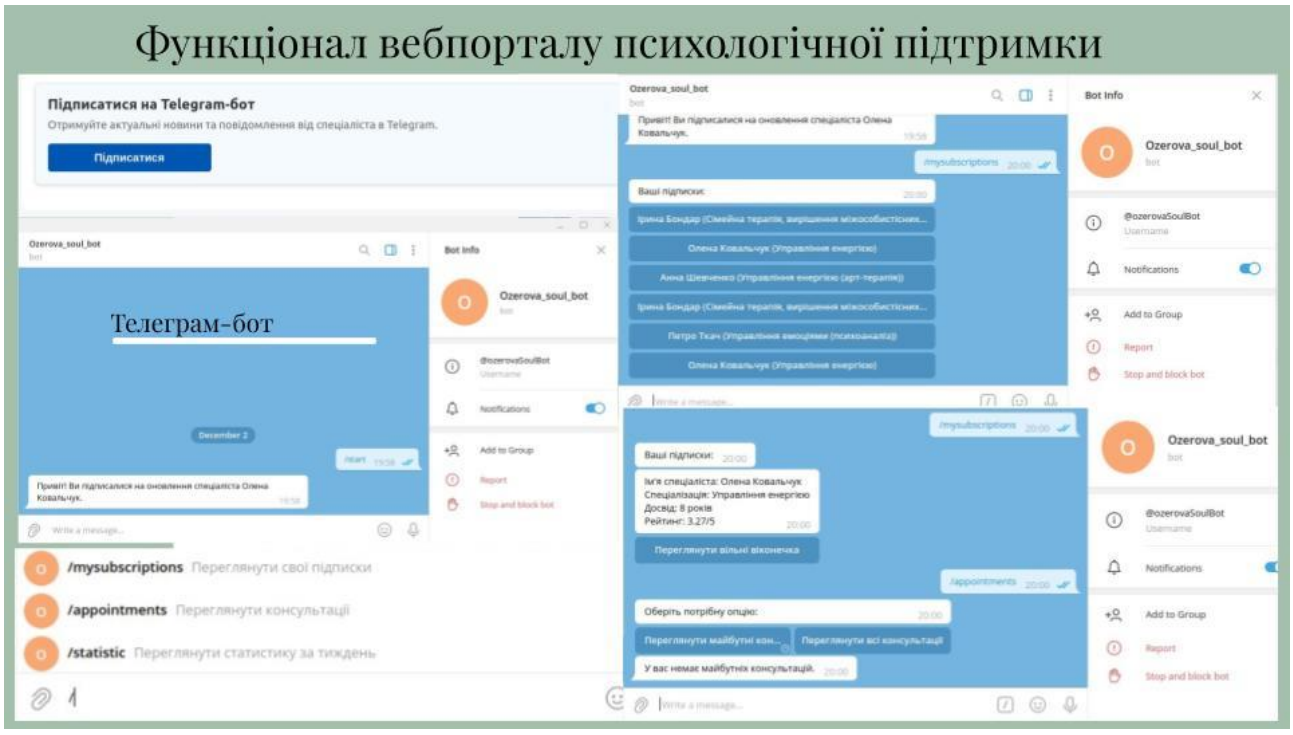

Рисунок Г.18 – Тестування програми (частина 5)

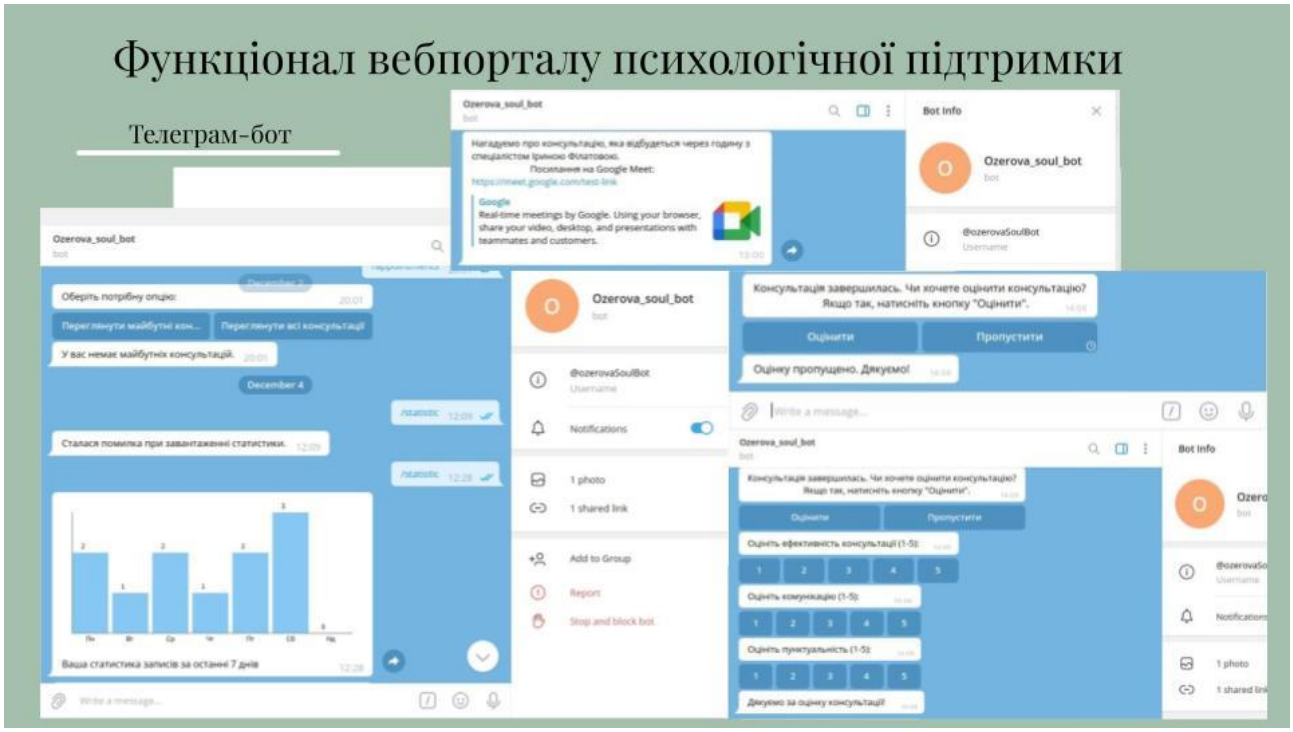

Рисунок Г.19 – Тестування програми (частина 6)

Висновки

Під час виконання магістерської кваліфікаційної роботи було розроблено інтерактивний вебпортал психологічної підтримки з розширеним функціоналом, що включає рейтингову систему спеціалістів, інструменти тестування психічного стану, персоналізований підбір спеціалістів, а також інтеграцію Telegram-боту для нагадувань і сповіщень.

Для реалізації роботи було використано сучасні технології, зокрема Node.js для серверної частини, React для клієнтської частини, а також SQLite для управління базою даних. Робота оформлена згідно методичних вказівок.

Було проведено аналіз сучасного стану проблеми, описано основні аналоги існуючих систем психологічної підтримки, виявлено їхні недоліки та обмеження. На основі цього аналізу обґрунтовано переваги розробленого вебпорталу порівняно з існуючими рішеннями. Розкрито, яким чином розроблена система вирішує проблеми доступності, персоналізації та інтерактивності, забезпечуючи зручність використання та ефективність взаємодії користувачів із платформою.

Було розроблено модулі для реєстрації та авторизації користувачів із застосуванням безпечного хешування. Метод розрахунку рейтингу спеціалістів із використанням вагових коефіцієнтів і нормалізації, що дозволяє забезпечити об'єктивність та стабільність рейтингу.

Інтегровано Telegram-бот, який виконує функції динамічних сповіщень, нагадувань та автоматизованого інформування користувачів.

Рисунок Г.20 – Висновки (частина 1)

Висновки

А також розроблено модуль проведення тестування психічного стану та підбору спеціалістів, які найкраще відповідають потребам користувача.

Подальшого розвитку отримав метод розрахунку рейтингу спеціалістів, який, на відміну від існуючих, базується на використанні вагових коефіцієнтів для кожного критерію оцінки та нормалізації результатів, що забезпечує стабільність і об'єктивність порівняння. Це дозволяє адаптувати рейтинг до потреб користувачів, враховувати важливість окремих аспектів роботи спеціалістів і коригувати оцінки залежно від стабільності отриманих відгуків.

Також удосконалено метод інтеграції Telegram-боту, який, на відміну від існуючих, реалізує автоматичне формування персоналізованих сповіщень, аналітичних звітів і нагадувань через систему планування подій Node Schedule. Це підвищує ефективність взаємодії користувачів із платформою, забезпечуючи своєчасну комунікацію та інформування.

Було проведено комплексне тестування системи, яке підтвердило її працездатність і відповідність технічним вимогам. Результати тестування продемонстрували стабільну роботу платформи в умовах реального використання та підтвердили ефективність розробленого функціоналу.

Розроблений вебпортал має значний практичний потенціал і може бути використаний для підвищення доступності психологічної підтримки, оптимізації роботи спеціалістів та надання користувачам ефективних інструментів для підтримки психічного здоров'я.

Рисунок Г.21 – Висновки (частина 2)

Апробації та публікації матеріалів

Апробація матеріалів магістерської кваліфікаційної роботи. Результати магістерської кваліфікаційної роботи обговорювалися на міжнародній науково-практичній конференції «Інформаційні технології та автоматизація – 2024».

Публікації. Результати дослідження опубліковані в тезах доповіді міжнародної науково-практичної конференції «Інформаційні технології та автоматизація – 2024».

Войтко В.В. Особливості розробки програмних модулів веб-системи психологічної підтримки / В.В. Войтко, Озерова К. О., Барчук Н.Є., Гаврилюк О.В. // Інформаційні технології і автоматизація – 2024 / Матеріали XVII міжнародної науково-практичної конференції. Одеса, 31 жовтня - 1 листопада 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. – С.482-484.

Рисунок Г.22 – Апробації та публікації