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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
В. о. завідувача кафедри
_____ Михайло НОВОТАРСЬКИЙ
(підпис)

“__” _____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

на тему: __Веб-платформа для психо-фізичної реабілітації_____

Виконала : студентка __4__ курсу, групи _ІМ-13
(шифр групи)

_____Мартинюк Марія Павлівна_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____професор, д.т.н., проф. Сергієнко А. М._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант___(нормоконтроль) ас. Пономаренко А. М._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент __ас. каф. СП і СКС, д.ф. Молчанов О. А._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2025 р.

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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри
Михайло НОВОТАРСЬКИЙ
(підпис)

“__” _____2025 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студентки

Мартинюк Марії Павлівни

1. Тема проєкту *Веб-платформа для психо-фізичної реабілітації*
керівник проєкту Сергієнко Анатолій Михайлович, проф. д.т.н
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 23 травня 2025 року № 1705-с

2. Термін здачі студенткою закінченого проєкту 03.06.2025

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1 Сучасні вебплатформи для психофізичної реабілітації

Розділ 2 Технології для розробки вебплатформи для психофізичної реабілітації

Розділ 3 Реалізація вебплатформи для психофізичної реабілітації

Розділ 4 Тестування вебплатформи для психофізичної реабілітації

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема, алгоритм взаємодії користувача з системою (принципова схема), функціональна схема.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Пономаренко А.М.		

7. Дата видачі завдання «01» грудня 2024 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.10.2024-30.11.2024*	
2.	*Вивчення та аналіз завдання*	*01.12.2024-01.02.2025*	
3.	*Розробка архітектури та загальної структури системи*	*02.02.2025-13.04.2025*	
4.	*Програмна реалізація системи*	*14.04.2025-18.05.2025*	
5.	*Оформлення пояснювальної записки*	*14.04.2025-08.06.2025*	
6.	*Захист програмного продукту*	*10.06.2025*	
7.	*Передзахист*	*10.06.2025*	
8.	*Захист*	*17.06.2025*	

Студентка-дипломниця _____ Марія МАРТИНЮК
(підпис)

Керівник проєкту _____ Анатолій СЕРГІЄНКО
(підпис)

АНОТАЦІЯ

У дипломній роботі глибоко досліджено проблему наявності дистанційних засобів для реабілітації населення. Проведено аналіз вимог до сучасних систем, потенційних можливостей їх покращення, а також наявних пропозицій на світовому та українському ринках. На основі цього аналізу сформовано вимоги до розроблювальної системи.

На базі мови програмування JavaScript, платформи Node.js та інструментів React і Express.js розроблено вебплатформу, що надає пропозиції відео методик психологічної та фізичної реабілітації відповідно до аналізу стану користувача, забезпечує відслідковування прогресу та надає статистичний аналіз отриманих даних. Система підтримує можливості збереження інформації за допомогою голосового вводу та інтегрує гейміфіковані мотиваційні елементи для забезпечення кращого користувацього досвіду. У розробці використовувались такі сторонні сервіси, як Auth0 та Google Cloud Platform, що надають надійний набір інструментів для реалізації високого рівня безпеки та вільного доступу до певних типів даних відповідно.

Ключові слова: вебплатформа, реабілітація, трекінгова система, голосовий ввід, React, Node.js, GCP, API інтеграція, Auth0.

ANNOTATION

The thesis researched and analyzed available options for remote rehabilitation on the global and Ukrainian markets, as well as the requirements for modern web systems of such subject. In addition, potential opportunities for improvement were identified. Based on the analysis, requirements for the development system were formed.

Using the JavaScript language, the Node.js platform, and the React and Express.js instruments, a web platform has been developed. It provides video suggestions for psychological and physical rehabilitation techniques based on the user's condition analysis, provides progress tracking, and statistical analysis of the obtained data. The system supports the ability to save information using voice input and integrates gamified motivational elements to ensure a better user experience. The development used third-party services such as Auth0 and Google Cloud Platform that provide reliable implementation of high level security and free access to certain types of data, respectively.

Keywords: web platform, rehabilitation, tracking system, voice input, React, Node.js, GCP, API integration, Auth0.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Веб-платформа для	4		
			психо-фізичної реабілітації			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Веб-платформа для	83		
			психо-фізичної реабілітації			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.007 Д1***	Веб-платформа для	1		
			психо-фізичної реабілітації			
			Структурна схема			
	А4	***ІАЛЦ.467200.007 Д2***	Веб-платформа для	1		
			психо-фізичної реабілітації			
			Алгоритм взаємодії			
			користувача з системою			
			(принципова схема)			
	А4	***ІАЛЦ.467200.007 Д3***	Веб-платформа для	1		
			психо-фізичної реабілітації			
			Функціональна схема			
	А4	***ІАЛЦ.467200.007 Д4***	Веб-платформа для	23		
			психо-фізичної реабілітації			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА	
Зм	***Лист***	***№ докум.***	***Підп***	***Дата***		
Розроб		Мартинюк М.П.			*Веб-платформа для психо-фізичної реабілітації* Опис альбому	Літ. / Аркуш: 1 / Аркушів: 1
Перев		Сергієнко А.М.				
						КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-13

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-платформа для психо-фізичної реабілітації*»

Київ – 2025

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Це технічне завдання поширюється на розробку вебплатформи для психофізичної реабілітації. Область застосування: медицина, корпоративний сектор, соціологія, навчальні заклади, повсякденне життя.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням дипломного проєкту є розробка вебплатформи для психофізичної реабілітації, що дозволить задовільнити потребу у отриманні дистанційної реабілітаційної допомоги, відслідковуванні та коригуванні фізичного та психологічного стану здоров’я.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс системи.
- Надати користувачам можливість реєстрації та авторизації.
- Надати користувачам можливість відновлення паролю у разі його втрати.
- Надати користувачам можливість отримувати щоденні реабілітаційні завдання відповідно до актуального стану здоров’я.
- Надати користувачам можливість пошуку відео інструкцій та методик реабілітації.
- Надати користувачам можливість зберігати корисні відео інструкції та методики реабілітації.
- Надати користувачам можливість відслідковувати стан здоров’я за певний період.
- Надати користувачам можливість отримувати статистичний аналіз показників за певний період.
- Надати користувачам можливість вводу щоденних показників за допомогою голосу.
- Надати користувачам можливість вручну коригувати введені дані.

5.2. Вимоги до програмного забезпечення

- ОС Mac, Linux чи Windows
- Веббраузер.

5.3. Вимоги до апаратної частини

- 32-розрядний (x86) або 64-розрядний (x64) процесор із тактовою частотою 1 ГГц або швидший.
- RAM не менше, ніж 2 ГБ.
- Підключення до мережі Інтернет.
- Мікрофон.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.10.2024-30.11.2024
Вивчення та аналіз завдання	01.12.2024-01.02.2025
Розробка архітектури та загальної структури системи	02.02.2025-13.04.2025
Програмна реалізація системи	14.04.2025-18.05.2025
Виправлення помилок	05.05.2025-18.05.2025
Оформлення пояснювальної записки	14.04.2025-08.06.2025

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-платформа для психо-фізичної реабілітації*»

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

WCAG	(Web Content Accessibility Guidelines) Настанови з доступності вмісту
DRY	(Don’t Repeat Yourself) Не повторюй себе
API	(Application Programming Interface) Прикладний програмний інтерфейс
UI	(User Interface) Користувацький інтерфейс
FSD	(Feature Sliced Design) Дизайн зрізу особливостей
ООП	Об’єктно-орієнтоване програмування
SOA	(Service-oriented architecture) Сервісно-орієнтована архітектура
HTTP	(HyperText Transfer Protocol) Протокол передачі гіпертексту
SPA	(Single Page Application) Застосунок єдиної сторінки
DOM	(Document Object Model) Об'єктна модель документа
ORM	(Object–Relational Mapping) Об’єктно-реляційне відображення
SQL	(Structured Query Language) Мова структурованих запитів
NPM	(Node Package Manager)
СУБД	Система Управління Базами Даних
JSON	(JavaScript Object Notation) Запис об'єктів JavaScript
NLP	(Natural language processing) Обробка живої мови
XSS-атаки	(Cross-Site Scripting) Міжсайтовий скриптинг
PWA	(Progressive Web App) Прогресивний вебзастосунок

ВСТУП

У сучасних умовах нестабільного соціального становища, воєнних конфліктів, високого рівня стресових навантажень питання психофізичної реабілітації набуває особливої актуальності. Зі збільшенням кількості потерпілих, зростає проблема в достатньому забезпеченні професійною допомогою усіх нужденних. Традиційні методи реабілітації – амбулаторні чи стаціонарні програми – часто виявляються ресурсозатратними, недостатньо гнучкими чи позиційно обмеженими. У такому контексті використання цифрових технологій, зокрема вебплатформ, відкриває нові можливості для забезпечення доступної, ефективної та персоналізованої реабілітаційної програми.

Аналіз сучасного стану в галузі психофізичної реабілітації свідчить про зростання інтересу до цифрових рішень цієї тематики як в Україні, так і закордоном. Однак, наявні на ринку застосунки зазвичай мають вузьку спеціалізацію, обмежені наданням допомоги лише одного вектору реабілітації – фізичного чи психологічного – та потребують дорогого обладнання чи сторонньої кваліфікації. Окрім цього, більшість систем не інклюзивними, надаючи лише один механізм взаємодії без інтеграції потенційних альтернатив, як-от голосовий ввід чи зчитування даних з периферійних пристроїв. Таким чином потреба в створенні гнучкої автоматизованої системи дистанційної реабілітації та відстеження власного стану здоров’я для пересічного користувача залишається незадоволеною.

Для її вирішення в рамках дипломного проєкту запропоновано вебплатформу із надання відео інструкцій для покращення психофізичного стану, відслідковування та статистичного аналізу самопочуття впродовж певного періоду, інтеграцією голосового вводу та доступу до збереженої інформації через механізм власного кабінету. Дана реалізація може знайти

широке застосування у медичних інституціях як допоміжний інструмент супроводу пацієнтів, у корпоративному секторі для запобігання професійного вигорання, у навчальних закладах для реалізації програм психоемоційної стійкості студентів та викладачів тощо.

Таким чином розробка вебплатформи для психофізичної реабілітації не лише відповідає сучасним запитам ринку, але й має вагоме соціальне та економічне значення, обумовлене її науковою новизною та практичною цінністю.

РОЗДІЛ 1
СУЧАСНІ ВЕБПЛАТФОРМИ ДЛЯ ПСИХОФІЗИЧНОЇ РЕАБІЛІТАЦІЇ

1.1 Предметна область

Психофізична реабілітація є комплексним процесом, спрямованим на відновлення фізичних, психологічних та соціальних функцій особи, що були порушені внаслідок різного роду травм, захворювань або інших несприятливих факторів. Даний процес поєднує у собі медичні, фізичні, психологічні, педадогічні та соціальні заходи для попередження розвитку патологічних процесів, які призводять до тимчасової або стійкої втрати працездатності, та націлені на ефективне і раннє повернення хворих та інвалідів у суспільство і до суспільнокорисної праці.[1]

Актуальність застосування такого роду відновлювального комплексу вправ і методик є беззаперечною в умовах нестабільного життя суспільства. Світова пандемія, спровокована розповсюдженістю COVID-19 стала тригером формування широкого кола проблем в медичній, соціально-психологічній, реабілітаційній сферах. Частота неврологічного або психіатричного діагнозу у пацієнтів після перенесеної COVID-19 в наступні 6 місяців склала 33,62%; частота постановки діагнозу вперше – 25,79%. Самі порушення психічного здоров’я є різноманітними й торкаються практично всіх сторін психічного функціонування людини, значно погіршуючи якість її життя.[2]

В Україні з початком повномасштабного військового вторгнення Російської Федерації проблема погіршення стану здоров’я серед військових та цивільного населення набула більшої розповсюдженності. Внаслідок бойових дій та терористичних атак на цивільну інфраструктуру кількість травмованих

суттєво зросла. Згідно з даними Міністерства охорони здоров’я з 2022 року кількість реабілітаційних відділень зросла практично у 2.5 рази.[3] Утім доцільно буде відмітити, що можливість повноцінного обліку травм серед населення сьогодні ускладнена. Через часті терористичні акти та напружену ситуацію на фронті кількість людей, що потребують реабілітаційних заходів зростає. У зв’язку з цим, відновлення здоров’я постраждалих військових та цивільного населення є одним із стратегічних пріоритетів країни.

У цьому контексті, сучасний ринок представляє низку інформаційних ресурсів різного типу – від фізичних текстових джерел до онлайн платформ. Серед них особливу увагу заслуговують вебплатформи, які на відміну від інших типів комп'ютерних сервісів мають низку суттєвих переваг.

По-перше, вебплатформи надають користувачеві можливість доступу до реабілітаційних програм з будь-якого пристрою, будь-коли та будь-де. Таким чином користувач не стикається з проблемою недостатньої кількості вільної пам’яті чи обмежень операційної системи або можливостей власного пристрою. Уніфікований сервіс для усіх типів користувачів дозволяє отримати однаковий досвід використання незалежно від апаратного забезпечення клієнта.

По-друге, використання браузера як середовища розгортання платформи забезпечує додаткову гнучкість в інтеграції нового функціоналу системи, дозволяючи оновлювати контент сервісу автоматично для усіх користувачів.

По-третє, в реаліях обмеженості ресурсів вебплатформи не потребують значного збільшення витрат на підтримку великої кількості користувачів. Таким чином ймовірна потреба масштабування сервісу в майбутньому є легшою в реалізації.

По-четверте, варто згадати про можливості інтеграції застосунку зі сторонніми онлайн сервісами, наприклад електронними медичним записами (ЕМЗ). Як результат, можливе удосконалення системи для кращої співпраці пацієнта та терапевта.

1.2 Вимоги до сучасних реабілітаційних вебплатформ та їх приклади

Сучасні комп’ютерні реабілітаційні системи надають широкий набір функціональних можливостей для забезпечення ефективного відновлення пацієнтів. Особливу популярність займають формати вебзастосунків, що дозволяють значно розширити доступність надання реабілітаційних послуг. Такого роду платформи допомагають реалізовувати практично безперевний процес лікування, надаючи зручні шляхи для використання пацієнтами в умовах післягоспоспітального відновлення чи віддаленого від будь-яких професійних медичних установ проживання.

Основними вимогами до сучасних систем відновлення такого формату є:

- Цілодобова доступність з будь-якого пристрою: застосунок має надавати користувачеві постійний стабільний доступ до персоналізованих програм тренувань, їх інструкцій, зворотнього зв’язку із лікарем чи медичною установою, забезпечуючи таким чином безперервність в терапевтичному процесі;
- Інтуїтивний інтерфейс та користувацький дизайн: інтерфейс застосунку повинен бути адаптованим до цільової аудиторії. Значною перевагою є наявність інклюзивних методів взаємодії – контрастні кольори, великі кнопки, простота використання чи голосовий супровід. Такі механізми забезпечують зручне використання платформи людьми з порушенням зору, моторики чи когнітивними обмеженнями;
- Інтеграція з перифирійними пристроями: сучасні застосунки повинні бути здатними збирати та обробляти дані з окремих сенсорних пристроїв, наприклад смарт-браслетів, датчиків руху або гіроскопів. Таким чином відстежування активності пацієнта стає повністю чи

напівавтоматизованим процесом та дозволяє динамічно адаптувати програму реабілітації до потреб клієнта;

- Можливість асинхронної підтримки з боку фахівців: мобільні та вебплатформи дозволяють підтримувати дистанційний зв’язок із обраним терапевтом. Це забезпечує швидке втручання професіоналів у разі виникнення нагальної потреби.

З підвищенням актуальності проблеми забезпечення лікуванням все більшої кількості людей ринок реабілітаційних сервісів за останні кілька років стрімко розширився. Кожен сервіс окрім спільної ідеї та основного функціоналу для реалізації вимог згаданих вище, має власні особливості, що вирізняють його серед конкурентів. Для кращого розуміння наявних рішень та підходів до забезпечення зручного дистанційного процесу реабілітації, нижче розглянуто декілька прикладів існуючих сервісів у межах українського та світового ринків.

1.2.1 RehabGuru

Британська платформа, що дозволяє оптимізувати низку функціональних аспектів клініки – від бронювань, записів про лікування та виставлення рахунків до цифрових форм, сповіщень і призначення комплексу вправ (рис. 1.1).[4] Має два варіанти реалізації – вебплатформу та мобільний застосунок. Вбудовані функції включають велику кількість вправ та шаблонів лікування, нотатки про лікування, вимірювання результатів, а також ведення щоденника пацієнта.

Вагомою особливістю цієї платформи, яка може бути розцінена як недолік для певних груп осіб, є безпосередня співпраця з клініками, що унеможливлює використання додатку для самостійного виконання вправ без попереднього огляду фахівця.

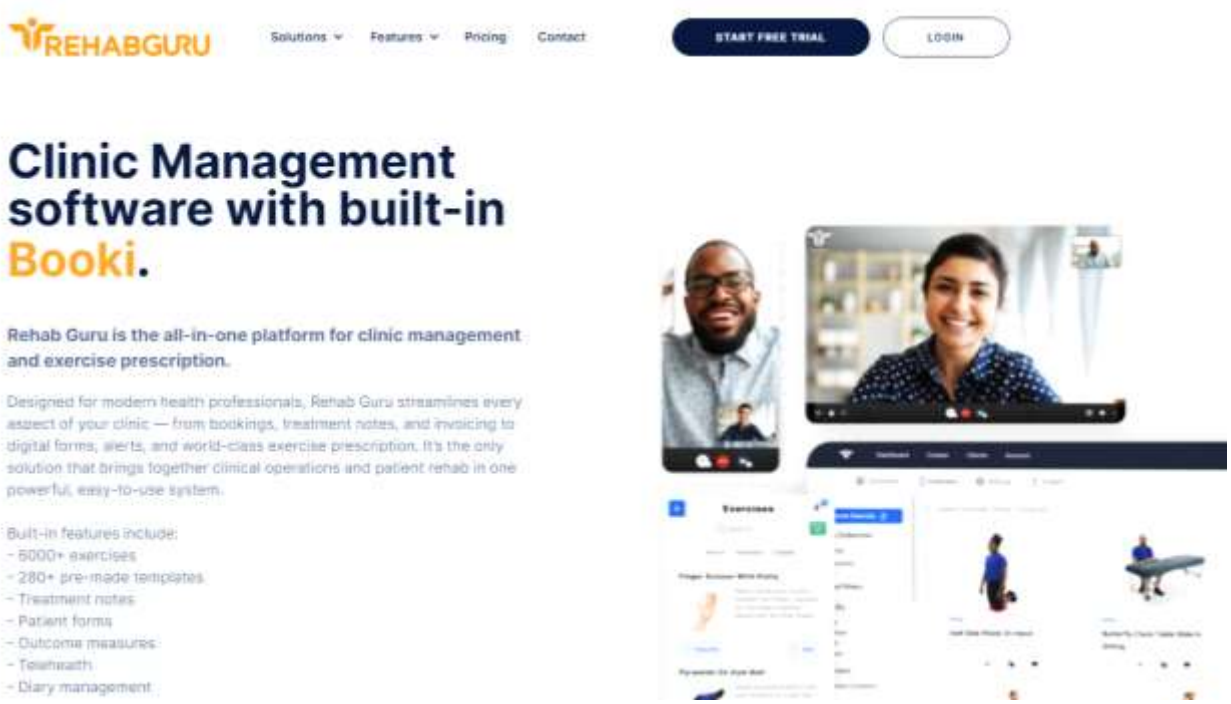


Рисунок 1.1 – Вебплатформа RehabGuru

1.2.2 SelfBack

Данська мобільна платформа, яка фокусується на відновленні спини та попереку (рис. 1.2).[5] Головною особливістю та перевагою даного застосунку, через що доцільним буде згадати його в контексті огляду наявних платформ для реабілітації, є використання штучного інтелекту для генерування рекомендацій вправ для виконання. Платформа збирає дані через смартфон за допомогою ввідного опитування і на основі отриманих результатів про стан пацієнта підлаштовує програму реабілітації. Окрім цього застосунок надає низку статей про причини виникнення болю та методів його зменшення, а також підтримує мотивацію пацієнта різного роду підбадьоруючими повідомленнями.

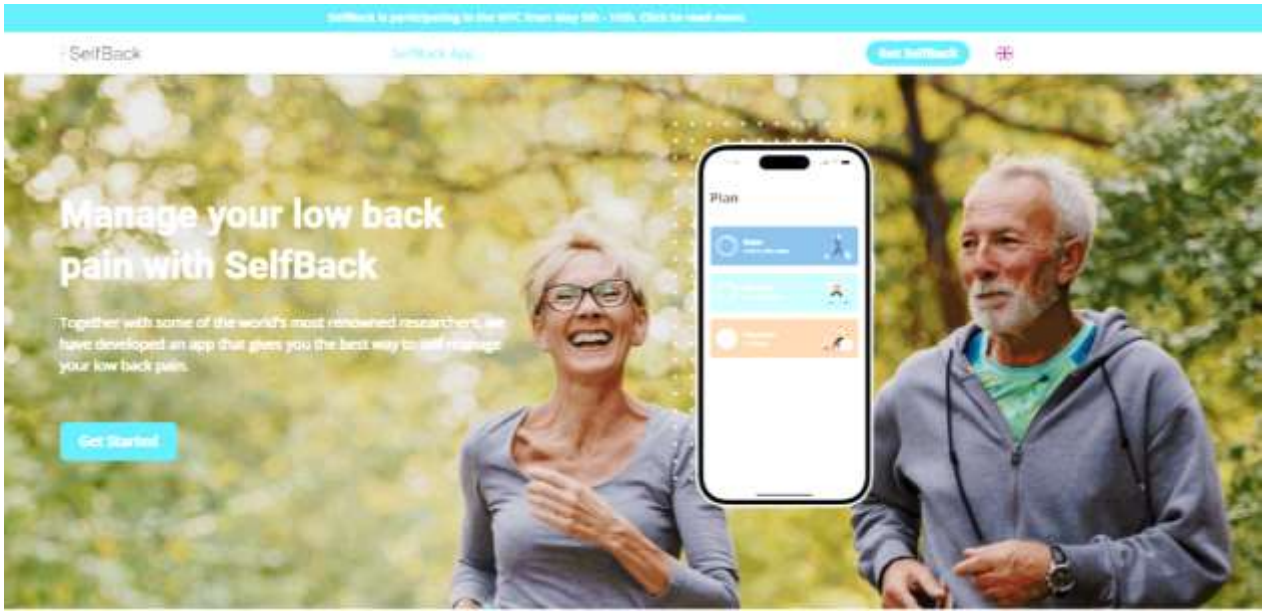


Рисунок 1.2 – Вебсайт мобільного застосунку SelfBack

Іншою особливістю SelfBack застосунку є його вузька спеціалізація – болі у спині та попереку. Така концентрація лише на одному виді фізичної реабілітації значно обмежує потенційну цільову аудиторію, адже застосунок не підходить для користувачів з іншими видами фізичних чи психологічних реабілітаційних проблем. Також для отримання необмеженого доступу до повного набору основних функцій, потрібно оформити відповідну підписку на сервіс.

1.2.3 Вільний Step

Українська вебплатформа, що об’єднує контактну інформацію різного роду установ з надання медичної, психологічної, юридичної чи соціальної допомоги для людей та членів їх родин, що пережили полон, постраждали від тортур або сексуального насильства (рис. 1.3).[6]

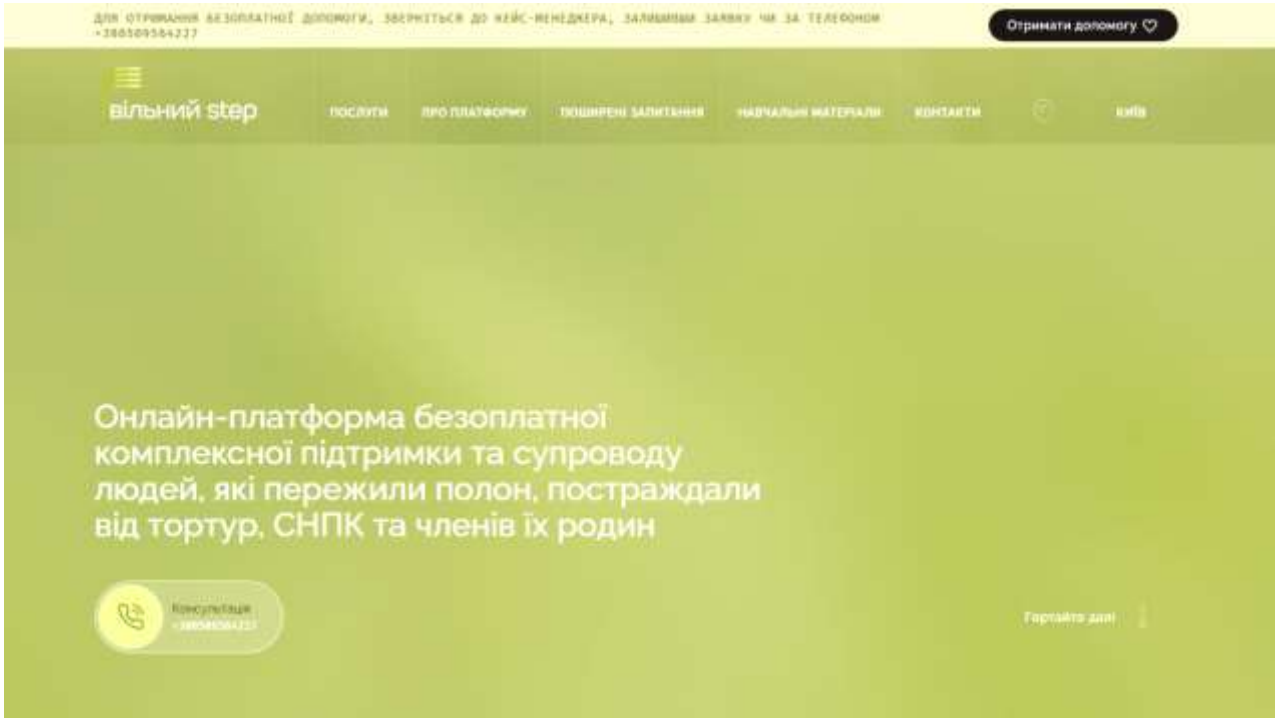


Рисунок 1.3 – Вебплатформа Вільний Step

Перевагою даної платформи є зручна, інтуїтивно-зрозуміла організація інтерфейсу, групурування та фільтрація усіх потрібних ресурсів для консультацій та повноцінного професійного супроводу відповідно до індивідуального запиту. Також вагомою перевагою є наявність на порталі опції безкоштовної допомоги.

Недоліком програми є відсутність відгуків користувачів про ту чи іншу установу, через що від потенційних клієнтів вимагається проведення додаткового аналізу опцій за допомогою сторонніх ресурсів. Окрім цього даний онлайн сервіс не передбачає можливостей проходження відео чи текстових тренінгів для надання самодопомоги і фактично являє собою лише інформаційний портал.

1.2.4 Ти як?

Всеукраїнська програма ментального здоров’я, яка ініційована першою леді Оленою Зеленською. Дана платформа містить курси технік самодопомоги, рекомендації від професіоналів та можливіть організації безкоштовних онлайн та офлайн зустріч із волонтерами з надання психологічної підтримки (рис. 1.4).[7]

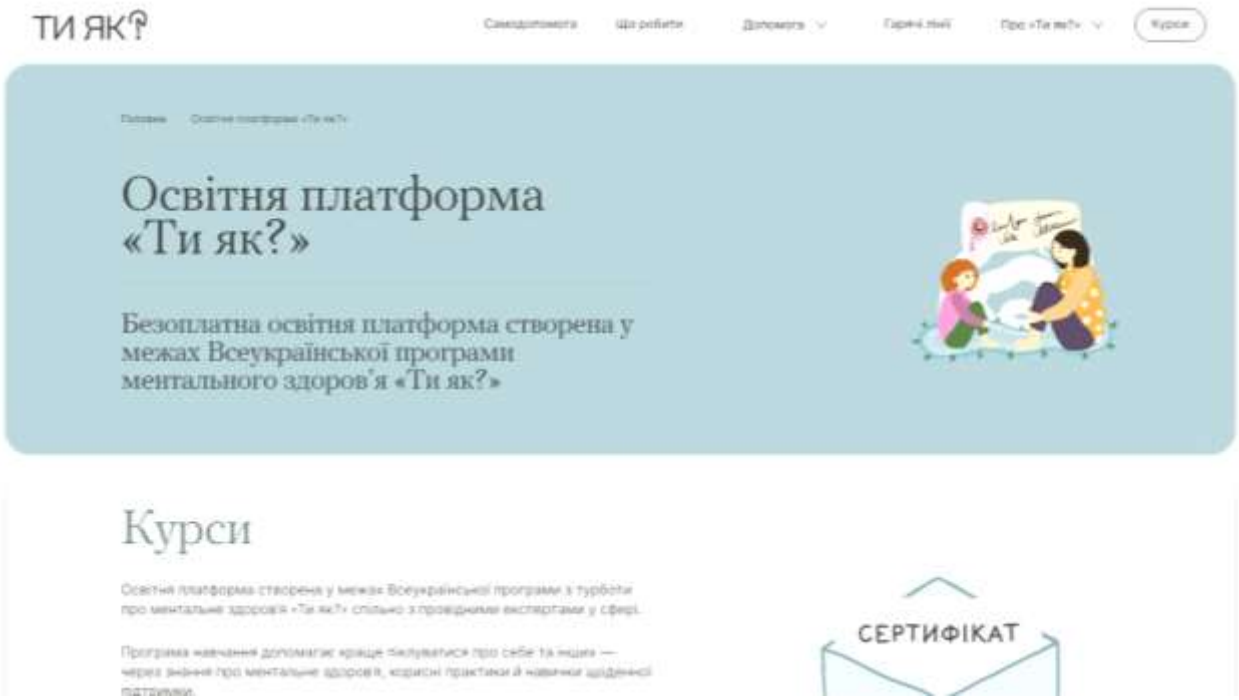


Рисунок 1.4 – Вебплатформа Ти як?

Окрім цього у застосунку вказані телефони гарячих ліній з широкого спектру питань, а також години роботи та ціна вхідних дзвінків до відповідних установ. Це безумовно є корисним аспектом, адже дозволяє користувачеві одразу знайти усю потрібну інформацію в одному місці.

Ще однією перевагою цієї платформи є доступність як з фінансової точки зору, так і з точки зору автономності користувача. Таким чином відновлювати власне ментальне здоров’я можливо самостійно у зручний час та в комфортних умовах, навіть без спеціального обладнання. Це робить

платформу особливо цінною для людей з обмеженим доступом до офлайн-центрів реабілітації чи фінансових ресурсів.

1.3 Порівняння наявних рішень

Для наочнішого співставлення особливостей та відокремлення переваг і недоліків кожного варіанту реалізації платформ для реабілітації, згаданих раніше, нижче наведено порівняльну таблицю.

Таблиця 1.1 – Порівняльна таблиця платформ для реабілітації

	RehabGuru	**SelfBack**	**Вільний Step**	**Ти як?**
Вебплатформа, як основний формат застосунку	+		+	+
Наявність власного кабінету	+	+		
Підтримка фізичної реабілітації	+	+	+	
Підтримка психологічної реабілітації			+	+
Наявність індивідуалізації програми реабілітації	+	+		
Наявність методів автоматизації процесу		+	+	
Автономність проведення процесу реабілітації		+		+
Наявність безкоштовного виду послуг			+	+

1.4 Можливості для покращення функціональності системи дистанційної реабілітації

1.4.1 Поєднання методів фізичної та психологічної реабілітації

Відповідно до досліджень інтеграція психологічних вправ в програму фізичної реабілітації сприяє покращенню загального стану пацієнта. Хоча психоемоційна підтримка не здатна повноцінно замінити фізичні відновлювальні вправи, згідно з дослідженням, опублікованим у Journal of Medical Internet Research [8], вона безпосередньо впливає на сприйняття та толерантність пацієнта до болю під час виконання тих чи інших тренувань. Окрім цього психологічна складова сприяє зниженню рівня тривожності та депресивних симптомів, які часто супроводжують хронічні болі чи тривалий відновлювальний процес.

Відповідно до інших відкритих наукових досліджень [9, 10], інтеграція психологічної підтримки та вправ у комплекс фізичної реабілітації ефективно покращує психологічне здоров’я та управління болем у пацієнтів, що потерпають від різного роду фізичних страждань. При порівнянні з фізіотерапією наодинці, психологічні інтервенції сприяють покращенню емоційної регуляції, що в свою чергу зменшує інтесивність болю у пацієнтів та загальний їх стан під час реабілітаційного процесу.

Таким чином для досягнення найбільш ефективного результу доцільним є фокусування на обидвох векторах реабілітації – фізичному та психологічному. Утім більшість сучасних вебплатформ концентруються лише на одному напрямку з двох, змушуючи пацієнтів таким чином використовувати декілька систем паралельно, що може виявитись незручним при потребі відслідковування загального стану здоров’я для можливого коригування обраного першочергово плану реабілітаційного процесу. Окрім

цього так як відповідно до згаданих ресурсів зв’язок між ментальним та фізичним аспектом є досить тісним, виникає потреба обміну даними між декількома окремими за напрямком відновлення платформами, що може виявитись незручним в використанні на постійній основі.

Таким чином поєднання в одній відновлювальній програмі вправ обидвох векторів реабілітації може принести більшу користь пацієнтові та створити цілісну й ефективну програму відновлення. Застосунок, який надає такий функціонал, не лише краще відповідатиме потребам користувачів та охоплюватиме більшу цільову аудиторію, а й зможе вирізнитись серед конкурентів на ринку цифрових рішень для охорони здоров’я.

1.4.2 Інтеграція мотиваційних елементів та гейміфікація

Окрім першочергового залучення користувачів до використання власної продукції, сучасні компанії на ринку часто зтикаються із проблемою утримання вже наявних клієнтів. Одним з варіантів вирішення такого завдання є впровадження елементів гри, таких як бали, досягнення, серії успіхів (страйки) та щоденні заохочення для підтримання внутрішньої мотивації користувача. Ці механізми не лише забезпечують позитивний досвід використання платформи, формуючи відчуття досягнення та контролю, що в свою чергу позитивно впливає на психоемоційний стан, а й допомагають у формуванні звички, що є неймовірно важливим аспектом у питаннях такого довготривалого процесу як реабілітація.

За даними дослідження Effectiveness of Gamification in Knee Replacement Rehabilitation: Protocol for a Randomized Controlled Trial With a Qualitative Approach [11] визначено, що використання ігрових механік у реабілітації позитивно впливає на рівень фізичних функцій та активності серед пацієнтів. Окрім того виявлено, що впровадження підбадьоруючих елементів покращує дотримання режиму лікування та підвищує мотивацію пацієнтів.

У контексті війни, коли багато пацієнтів мають обмежений доступ до офлайн-центрів реабілітації, вебплатформи з мотиваційними елементами гейміфікації можуть стати ефективною альтернативою для самостійного відновлення, забезпечуючи при цьому послідовне виконання визначеної програми завдяки постійній залученості користувачів та зворотньому зв’язку з боку платформи.

З точки зору ринку цифрових рішень в сфері охорони здоров’я, застосунки з підтримкою таких елементів мають потенціал виділитись серед конкурентів.

1.4.3 Інтеграція відстеження стану та прогресу

Моніторинг прогресу є ключовим елементом будь-якої реабілітаційної програми, оскільки надає медичним фахівцям і не тільки можливість своєчасно визначити ефективність застосованих стратегій та виявити ті, що не приносять очікуваних результатів. Таким чином даний механізм дозволяє оперативно коригувати план лікування. Відповідно до відкритих досліджень, індивідуальне адаптування програми лікування завдяки регулярному моніторингу сприяє підвищенню залученості та рівня задоволення пацієнта. Окрім цього клієнти, що отримують чіткий заворотній зв’язок та мають змогу до того ж самостійно відстежувати власний прогрес, більш мотивовані до активної участі в курсі реабілітації.[12]

Відстеження прогресу має ще більший вплив у випадках, коли інформація чітко фіксується або стає доступною іншій визначеній особі. Такий соціальний аспект контролю підсилює відповідальність, а також формує відчуття підтримки та спонукає до більшої послідовності в діях. Публічне або зафіксоване зобов’язання часто виступає додатковим стимулом для дотримання плану відновлення чи змін у поведінці, що підтверджено багатьма поведінковими дослідженнями.

Таким чином інтерграція трекінгових систем для відстеження стану та прогресу допомагає вирішити ще одну проблему, згадану у підпункті вище, – утримання клієнтів у інфополі застосунку та забезпечення послідовного виконання призначеної програми.

1.4.4 Інтеграція альтернативних методів збору даних

У сучасних реаліях, де різноманітність користувачів стає не винятком, а нормою, інтеграція інклюзивних методів взаємодії системи та користувача набуває все більшого значення та популярності. Люди мають різні рівні доступу до технологій та різні фізичні або когнітивні можливості, ігнорування яких призводить до втрати важливих сегментів аудиторії та посилення цифрової ізоляції. Вирішенням такої проблеми може стати інтеграція у за стосунок голосового вводу чи взаємодії з перефирійними пристроями (смарт годинниками, спортивними браслетами тощо) для збору даних, підтримка багатомовності, адаптивних шрифтів чи максимально простого за структурою та виглядом інтерфейсу.

Варто відмітити, що у глобальному контексті, де цифрова етика, відповідальність та нормативні вимоги, як-от WCAG, стають частиною юридичних і соціальних зобов’язань, інклюзивність переходить з розряду «бажаних» до «обов’язкових» характеристик програмного забезпечення. Таким чином, проєкти, що інтегрують альтернативні методи взаємодії на ранніх етапах розробки, не лише демонструють соціальну відповідальність, але й забезпечують свою готовність до масштабування, довготривалого успіху та позитивного сприйняття на ринку.

ВИСНОВОК ДО РОЗДІЛУ 1

В першому розділі розглянуто актуальність проблеми реабілітації в сучасному світі. Доцільність розроблення застосунків, зокрема вебплатформ, які спрямовані на організацію та забезпечення відновлювального процесу людям різного фінансового та соціального становища аргументована багатьма дослідженнями останнього дисятиліття.

Наявність широкого кола різноманітніх платформ та конкуренція серед компаній на ринку підтверджують потребу у впровадженні нових підходів до організації застосунками реабілітаційного процесу, зокрема для самостійного відслідковування зміни власного стану та можливостей. Для кращого розуміння актуальної ситуації та існуючих рішень на світовій арені, проведено аналіз ринку на прикладі низки платформ, серед яких іноземні компанії RehabGuru та SelfBack, а також вітчизняні сервіси Вільний Step та Ти як?. За результатами аналізу особливостей виокремлено переваги та недоліки кожної платформи.

Як наслідок, досліджено та запропоновано можливі покращення до систем, серед яких поєднання двох векторів реабілітаційного процесу, інтеграція трекінгових систем, впровадження альтернативних методів взаємодії системи та користувача, а також додавання мотиваційних постів та елементів гейміфікації.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ВЕБПЛАТФОРМИ ДЛЯ ПСИХОФІЗИЧНОЇ РЕАБІЛІТАЦІЇ

2.1 Архітектура та підходи до організації вебплатформи

Аріхітектура застосунку являє собою набір певних модулів та компонентів, а також зв’язків між ними. При проєктуванні вебплатформи для психофізичної реабілітації важливо зважено підійти до вибору архітектурного підходу, адже в залежності від кінцевих цілей продукту обраний тип може відрізнятись.

Сучасні вебплатформи являють собою складні структуровані проєкти, як на рівні організації бекенду та фронтенду (серверної та клієнтської частин відповідно), так і на рівні взаємодії компонентів всередині цих двох сфер. Через це є доцільним окремо розглянути підходи до організації кожної зони відповідальності.

2.1.1 Архітектура клієнтської частини застосунку

Загалом розрізняють 5 основних підходів до структурування фронтенд (клієнтської) частини проєкту – класична архітектура, модульна проста архітектура, Atomic Aesign, Feature Sliced Design та мікрофронтед у поєднанні з мікросервісною архітектурою та модульністю.

Класичний (або безархітектурний) підхід передбачає відсутність явних зв’язків між компонентами/модулями, які визначаються як набір UI компонентів, що характеризують певну бізнес-логіку всередині себе. Це можуть бути як великі компоненти, як-от сторінки, чи менші, як-от окремі кнопки або поля вводу.

Головною проблемою даного підходу є наявність великої кількості неявних зв’язків або навіть суцільної кільцевої структури, коли один компонент використовує всередині себе низку інших. Як наслідок, компоненти користувацького інтерфейсу змішуються з логічними компонентами, утворюючи цілі логічні модулі, наприклад редагування профілю, вибір місця призначення тощо. Утім чіткого розподілення зони відповідальності того чи іншого функціоналу в цій архітектурній моделі не споглядається. Варто відмітити, що на ранніх етапах розробки це не є великою проблемою. Утім із розвитком та збільшенням масштабу застосунку існує висока ймовірність плутанини між функціоналом та призначенням компонентів, а також недодерження Don’t Repeat Yourself (DRY) принципу програмування через появу компонентів з невеликими відмінностями, але спільною основною логікою. На розуміння зв’язків між структурними блоками потрібно все більше і більше часу, що є значним негативним ефектом. Така ж проблема може виникнути і в області взаємодії з даними та станом. Таким чином на рівні застосунку зв’язки всередині модулів та між ними є слабкими, що не є ідеальним варіантом для фінального вигляду системи.

Утім такий підхід має і свої переваги. Перш за все, його доцільно використовувати у невеликих командах розробників та відносно простих проєктах з невеликою кодовою базою. Також якщо ціллю проєкта є розробка саме прототипу для майбутньої системи, а не повноцінного її варіанту, та довгострокова підтримка не є однією з цілей фінального продукту, такий підхід може стати прийнятним вибором.

Модульна проста архітектура передбачає наявність базових шарів організації процесу, таких як сторінки, самостійні модулі з власною бізнес-логікою та зоною відповідальності, компоненти з можливістю перевикористання на багатьох непов’язаних сторінках та UI елементи, наприклад кнопки, модульні вікна чи поля вводу. Основною особливостю є те,

що нижчі шари не можуть використовувати елементи шарів вище. Наприклад, UI елементи не містять у собі жодної логіки. Натомість компоненти вже можуть використовувати у собі елементи для відображення користувацького інтерфейсу, але не можуть містити модулі. Модулі є незалежними структурами зі своїм окремим функціоналом, залежностями та взаємодіями з серверною частиною додатка. Таким чином забезпечується висока зв'язність між елементами функціонального блока системи. Утім варто відмітити важливість ізолювання кожного модуля для неможливості спроби витягу якогось компоненту ззовні. Для цього використовується деякий публічний інтерфейс, завдяки якому існуватиме можливість дістати потрібний елемент. Іншими словами, принцип даної архітектури є забезпечення інкапсуляції модулів з можливістю використання в структурах вище лише чітко визначених публічним API елементів.

Як результат, перевагою такого підходу до організації фронтеду є можливість легкого видалення та зміни глобальної структури застосунку без потреби ручного розплутування неявних зв'язків між великою кількістю окремих блоків. З недоліків можна відмітити ймовірну появу проблеми використання одного модуля іншим чи загальної побудови структури застосунку. Також даний підхід не виключає появу глобального стану, тим самим не викорінюючи повністю проблему неявних зв'язків. Через це використання такої моделі для великих та складних застосунків все ще не є ідеальним варіантом.

Архітекутра Atomic Design набула своєї популярності у кінці 2010-х років. По своїй суті вона дуже схожа на згадану раніше модульну архітектуру. Весь застосунок розбивається на 5 головних шарів: атом, молекула, організм, шаблон та сторінка. Аналогічно до модульної архітектури вищі шари будуються з елементів нижчих. Основну популярність дана модель структуризації отримала серед дизайнерів, адже така організація компонентів є дуже зручною при побудові макетів тих чи інших додатків. Утім на рівні

розробки даний підхід не має значних відмінностей від згаданої раніше простої модульної архітектури, окрім як появи ще одного шару абстракції – шаблонів. Тому усі недоліки та переваги попередньої моделі присутні і у цій методології.

На відміну від інших, Feature Sliced Design (FSD) будується на основі трьох основних понять: шар, модуль-зріз (slice) та сегмент. Шари є чітко визначеними та бізнес-орієнтованими. Серед елементів їх ієрархії виділяють спільні дані (shared), сутності (entities), особливості (features), віджети (widgets), сторінки (pages), процеси (processes) та застосунок (app). Ієрархія шарів є лінійною – вищі шари використовують у собі нижчі і ніяк не навпаки. Кожний шар містить в собі певні чітко визначені модулі-зрізи.

Shared – це найбільш часті у перевикористанні компоненти, що напряму не прив’язані до конкретної бізнес-логіки – кнопки, модальні вікна, допоміжні функції тощо. Щодо рівня entities варто відмітити, що прив’язка функціоналу в аспекті модуля відбувається не до абстрактних глобальних елементів, а до конкретних бізнес-сутностей – запитів до користувача, запитів до товару, допоміжних функцій до користувача тощо. Як наслідок, у компонентах цього рівня зберігається весь функціонал, який пізніше використовується для реалізації функціоналу конкретного направлення на шарах вище. Features – модулі-сценарії, що несуть в собі якусь бізнес-цінність та можуть бути самостійними елементами. Widgets являють собою незалежні елементи. До них можна віднести різні хедери, сайдбари, футери тощо. Pages – сторінки з об’єднаних елементів з шарів нижче. Processes – досить опціональний рівень, який використовується при наявності у застосунку певного функціонлу, що потребує використання декількох сторінок, наприклад поетапна реєстрація з декількох форм. App містить логіку застосунку при ініціалізації. Іншими словами, на цьому рівні визначаються роутери, провайдери, глобальна конфігурація застосунку тощо.

Кожний сегмент в середині себе містить певні конструкції:

- UI – елементи користувацького інтерфейсу;
- Model – бізнес-логіка, тобто взаємодія зі станом, різні селектори, обробники події тощо;
- Lib – допоміжні функції, які можуть використовуватись всередині модуля;
- Config – відносно рідкий у використанні модуль, представляє собою конфігурацію модуля;
- Api – містить запити до серверу, які потрібні в даному модулі;
- Consts – загальні константні зміні, що використовуються в даному модулі.

Даний вид архітектури фронтенду застосунку через наявність чітко визначених компонентів та їх зон відповідальності задовільняє можливість реалізувати основні приципи Об’єктно-орієнтованого програмування (ООП)– абстракцію, поліморфізм, інкапсуляцію та наслідування. Аналогічно до принципу в простій модульній архітектурі доступ до тих чи інших компонентів відбувається через публічний API, інкапсулюючи таким чином модулі один від одного.

Як результат структура додатку наближається до ідеальної – наявні сильні зв’язки всередині окремих модулів, тобто кожен модуль відповідає лише за один чітко визначений функціонал, та наявні слабкі зв’язки між самими модулями, тобто компоненти легко модифікувати чи видаляти в залежності від оновлених потреб до кінцевого продукту. Іншою перевагою даного підходу є легкість в розумінні призначення кожного окремого блоку розробником ззовні та ведення загальної документації системи.

Серед недоліків доцільно буде виділити складність в першочерговому визначенні особливостей декомпозиції модуля на ранніх етапах розробки. У разі відсутності чіткого бачення фінального продукту, розбити проєкт на логічні незалежні блоки є важкою задачею. Окрім цього FSD методологія не релевантна для малих проектів, у яких немає на меті

підтримка довгострокового продакшену. Також проблеми, що виникають під час розробки вимагають знаходження рішень одразу і не передбачають можливості відкласти на потім, що унеможливлює швидку розробку для звичайних прототипів чи низьким за бюджетом часу та ресурсів проєктів.

Останній вид архітектури фронтенд частини – це мікрофронтед у поєднанні з мікросервісною архітектурою та модульністю. На відміну від попередніх чотирьох, де головна суть будувалась на базі монолітного типу архітектури, дана методологія передбачає розбиття загальної кодової бази на незалежні мікросервіси. Окрім цього усі спільні компоненти та налаштування виносяться в окремі пакети.

Перевагою такого підходу є зменшення часу на деплой, тестування та сборку проєкта при значному його обсязі. Також варто відмітити, що у великі монолітні застосунки додавати новий функціонал чи виконувати суцільний перепис системи для використання інших технологій або фреймоворків є набагато важче, ніж у менші. Через це підхід із подрібнення одного великого проєкту на окремі менші одиниці є доцільним варіантом розвитку. Однак з цього випливає відповідний недолік – потреба у фундаментальних знаннях розробника через високу складність загальної інфраструктури проєкту. [13, 14]

2.1.2 Архітектура серверної частини застосунку

Серед методологій організації бекенду можна виділити 6 основних типів архітектури – монолітна, сервісно-орієнтована (SOA), мікросервісна, безсерверна (serverless). [15]

Монолітна архітектура є найпростішою, традиційною формою організації та передбачає розміщення всієї функціональності застосунку – UI, доступу до даних, обробки запитів, бізнес-логіки тощо – в одному програмному базисі. Таким чином, система являє собою єдину неподільну

одиницю. Дана методологія розробки найкраще підходить для малих та середніх за розміром додатків, проєктів-доказів концепції/прототипів (proof-of-concept) чи мінімально життєздатних продуктів (MVP).

Значною перевагою є зниження складності та спрощення процесу розробки, тестування та підтримки, що дозволяє швидко отримати бажаний результат. Окрім цього, налаштування автоматизованого тестування є простішим через потребу у конфігурації лише одного CI/CD пайплайну та керування лише одного екземпляру системи. Однак даний підхід має свої недоліки. Із ростом складності та об’єму застосунку будь-яка зміна в окремому модулі потребує повторної збірки та ретестування всього додатку, що на пізніх етапах розробки потребує значних часових та функціональних ресурсів. Таким чином дана методологія не забезпечує високу гнучкість у модифікаціях застосунку, що може стати недоліком при зміні фінального бачення продукту.

На противагу такому підходу існує сервісно-орієнтована архітектура (SOA). Це архітектурний підхід, який організовує функції застосунку у вигляді окремих сервісів, взаємодія між якими відбувається завдяки чітко визначеним інтерфейсам (API). Кожен такий сервіс інкапсулює конкретну бізнес-функцію, наприклад автентифікацію користувача, обробку платежів або керування даними, та може викликатися незалежно. Основна мета SOA – забезпечення повторного використання компонентів великої системи та її гнучкості загалом, дозволяючи використання одних і тих же сервісів у різних застосунках. Хоча SOA забезпечує більш модульну структуру, аніж монолітна методологія, відокремлені сервіси все ще належать до спільної кодової бази, а тому часто залежать від центральної бази даних.

Перевагою такої методології є повторне використання сервісних компонентів, що сприяє заощадженню часу розробки та забезпечує послідовність функціональності. Також виникає слабкий зв’язок між окремими сервісами-блоками, що забезпечує можливість незалежної

модифікації та масштабування без ризику негативних наслідків на інші компоненти. Утім через ріст кількості функціональних модулів, зростає і складність в їх управлінні, забезпеченні коректних залежностей та узгодженності даних. Також зростає навантаження через необхідність підтримки постійної комунікації між елементами системи.

Мікросервісна архітектура бере за основу приципи SOA та розбиває функціонал на ще менші незалежні блоки, кожен з яких відповідальний за окремий аспект бізнес-логіки. На відміну від сервісно-орієнтованої архітектури, де сервіси часто є міцно зв’язанимим та мають одну ресурсну базу, мікросервіси оперують повність автономно. Комунікація між ними забезпечується HTTP протоколом або чергами повідомлень, що забезпечує мінімальні залежності та значну гнучкість. Окрім цього дана методологія дозволяє використання низки різних технологій для реалізації модулю, відповідно до задачі кожного. Утім варто відмітити, що даний підхід значно ускладнює інфраструктуру платформи, так як кожен незалежний модуль потребує окремого CI/CD пайплайну для тестування, процесу моніторингу роботи та механізму обробки запитів. Також зростає складність в управлінні даними через необхідність забезпечення чіткої комунікації сервісів, що мають окремі індивідуальні бази даних.

Порівнюючи мікросервісну та сервісно-орієнтовану архітектуру, може виникнути питання доцільності обрання того чи іншого варіанту. SOA доцільна для систем, що потребують забезпечення модульності, наприклад через власні розміри, але не потребують повної ізольованості сервісів. Вона підходить для великих організацій, яким потрібно інтегрувати різні застосунки, наприклад легасі-системи, через спільні сервіси. SOA часто використовується, коли важливим є забезпечення повторного використання сервісів і централізованого управління транзакціями. Ця архітектура дозволяє відокремити бекенд модулі, які обслуговують різні платформи (веб, мобільні

додатки), але сервіси все ще можуть залишатись частиною єдиної кодової бази та спільної інфраструктури.

Мікросервісну архітектуру слід обирати, коли важливими є масштабованість, гнучкість у розгортанні та швидкість змін. Цей метод організації підходить для компаній, що активно використовують методологію Agile, DevOps та CI/CD, а також для систем, які повинні витримувати великі навантаження, часто оновлювати окремі компоненти та мати високу відмовостійкість. На відміну від SOA, мікросервіси розділяють усе – від коду до інфраструктури, що відіграє значну роль при потребі забезпечення стабільної роботи платформи загалом.

Існує також варіант гібридної моделі, коли SOA-сервіси розгортаються окремо, подібно мікросервісам, але залишаються частиною спільної кодової бази чи інфраструктури. Такий підхід зменшує складність повністю ізольованих мікросервісів, зберігаючи при цьому певний рівень модульності, масштабованості та ізоляції збоїв. Варто відмітити, що на практиці багато компаній починають із SOA, поступово переходячи до мікросервісів у процесі масштабування або за нових вимог системи.

Serverless-архітектура, або архітектура без сервера, передбачає розміщення бізнес-логіки у вигляді окремих функцій у якості сервісу (Function-as-a-Service), які викликаються на вимогу у відповідь на певні події. Такі функції запускаються у деякому хмарному середовищі, наприклад AWS Lambda, Azure Functions чи Google Cloud Functions, що дозволяє розробникам не турбуватись про серверну інфраструктуру, балансування навантаження чи масштабування системи. Така делегація функціоналу стороннім сервісам чудово підходить при обробці подій, як-от авторизація користувача чи обробка черг повідомлень. Утім варто відмітити, що дані сервіси зазвичай не є повністю безкоштовними та мають обмежений час виконання функцій. Також тісний зв’язок власної системи із конкретним стороннім сервісом ускладнює процес переходу на іншу платформу без значних змін у кодовії реалізації.

2.2 Інструменти для розробки фронтенду вебплатформи

Існує безліч різноманітніх технологій для розробки фронтенд частини вебзастосунків. Зазвичай, усі вони базуються на єдиній мові JavaScript, що на сьогоднішній день є провідною у розробці додатків у сфері Web. Для розуміння особливостей та доцільності використання, нижче наведено аналіз найпопулярніших варіантів.

2.2.1 Angular

Angular представляє собою повноцінний фреймворк для розробки вебдодатків та мобільних застосунків. Найкраще дана технологія підходить для реалізації великих за обсягом та складністю проєктів, адже забезпечує чітке структурування коду та надає широкий вибір вбудованих інструментів для маршрутизації, управління станом, формами, HTTP-запитами, тестуванням тощо. Ще однією з його особливостей є використання базису TypeScript, що забезпечує статичну типізацію і полегшує майбутнє масштабування проєктів. Окрім цього Angular підтримує двостороннє зв’язування даних, що в свою чергу спрощує роботу з динамічними даними, які потребують реагування та оновлення стану системи в режимі реального часу.

Утім завдяки широкому функціоналу, даний фреймворк є досить великим за розміром, через що час завантаження додатка значно зростає. Особливо ця проблема помітна при запуску системи на мобільних пристроях. Із великого обсягу технології випливає і проблема “легкого входу”. Іншими словами, новим розробникам, не знайомим із даним інструментом, складніше освоїти технологію за одиницю часу, в порівнянні із іншими засобами, як-от React чи Vue. Хоча як згадувалось раніше, структурний підхід Angular має свої

переваги для великих проєктів, ця особливість одночасно й обмежує кордони гнучкості застосунку, унеможливлюючи швидку адаптацію функціоналу до оновлених вимог технічного завдання. [16]

2.2.2 React

Хоча React не являється фреймворком у звичному розумінні, з кожним роком ця JavaScript бібліотека для реалізації UI функціоналу набуває все більшої популярності. Особливістю даного інструменту є поєднання інтуїтивно зрозумілого синтаксису мови JavaScript із використанням принципу модульної архітектури, що дозволяє розробникам створювати універсальні багаторазові компоненти користувацького інтерфейсу, які в свою чергу слугують основними блоками для побудови складніших структур. Таким чином забезпечується зрозумілість кодової реалізації, а також масштабованість та гнучкість системи загалом.

Окрім забезпечення компонентного підходу, дана бібліотека підтримує легку інтеграцію з низкою інших інструментів розробки, такими як Node.js, Gatsby або Remix. Це забезпечує універсальність використання даного засобу для реалізації як невеликих Single Page Application (SPA) із простим функціоналом, так і складних рендерингових програм. Завдяки віртуальному Document Object Model (DOM), React мінімізує кількість операцій з реальним DOM, тим самим значно підвищуючи продуктивність додатків. Також варто відмітити, що завдяки значній популярності, спільнота React розробників є великою та активною. Цей аспект є неменш важливим під час вибору інструментарію, адже дозволяє уникнути непередбачуваних проблем із реалізацією через банальну відсутність документації чи відкритих форумів.

Утім популярність даної технології спонукає до активної появи нових підходів та принципів реалізації тих чи інших аспектів. Через що використання React потребує постійного знаходження в контексті технології

та адаптування розробленої системи під нові стандарти, задля підтримки стабільності функціонування. Окрім цього, так як бібліотека не являє собою повноцінний фреймворк, а тому не покриває усі можливі аспекти розробки, виникає потреба у використанні додаткових інструментів.[17]

2.2.3 Vue.js

Це прогресивний фреймворк з відкритою кодовою базою для створення елементів інтерфейсу користувача і SPA застосунків, який заснований на моделі Model-View-ViewModel (MVVM). Даний фреймворк має поступово адаптовану архітектуру, яка зосереджена на декларативному рендерінгу та об'єднанні компонентів, що полегшує повторне використання, тестування та рефакторинг коду. Кожен компонент може бути незалежним, з власним шаблоном, логікою та стилем, що відповідає сучасним підходам до розробки інтерфейсів.

Хоча Vue першочергово був орієнтований на JavaScript, з версії 3.0 фреймворк отримав повноцінну інтеграцію з мовою TypeScript, що як наслідок полегшило розробку великих проєктів, де важлива статична типізація та масштабованість. Подібно до React, розширені можливості, що необхідні для написання більш складних застосунків, які, до прикладу, мають реалізацію маршрутизації чи керування станом, надаються через офіційно підтримувані допоміжні бібліотеки та пакети. Схоже до Angular, фреймворк підтримує двостороннє зв’язування даних, яке забезпечує синхронізацію між моделлю (даними) та UI представленням в режимі реального часу.

Перевагою використання даного фреймворку є його простота вивчення. Із цього випливає зручність використання Vue.js у якості технології для швидкого написання простих застосунків чи створення прототипів до складніших систем. Іншим важливим аспектом є гнучкість фреймворку, що дозволяє впроваджувати дану технологію у існуючі проєкти плавно та легко.

Окрім цього, схоже до React, використання віртуального DOM забезпечує високі показники продуктивності та швидкодії.

Серед недоліків даного фреймворку варто відмітити обмежену підтримку з боку великих компаній у порівнянні з React або Angular, що може впливати на впровадження сторонніх сервісів в довгострокових проєктах. Хоча Vue.js активно підтримується спільнотою, його екосистема менш усталена, ніж у конкурентів, особливо у сфері великих інструментів для розробки, DevOps або enterprise-рішень. Крім того, документація, хоча й добре написана, іноді відстає від нових релізів або нових офіційних бібліотек, що створює певні труднощі при оновленні проєктів.[18]

2.3 Інструменти для розробки бекенду вебплатформи

Набір інструментів для розробки бекенду (серверної частини) вебплатформи охоплює різні аспекти створення серверної частини системи: від обробки HTTP-запитів до управління базами даних, маршрутизації, автентифікації, реалізації бізнес-логіки, логування, масштабування та тестування. Серед найпопулярніших технологій, які активно використовуються у реалізації платформ різного масштабу, варто відмітити Django, Node.js, та Spring Boot.

2.3.1 Django

Django – це високорівневий фреймворк для мови програмування Python, що дотримується таких принципів програмування, як DRY та Convention over Configuration. Він призначений для швидкої розробки серверної частини вебзастосунків і має широкий набір функціоналу: об’єктно-релеляційне відображення (ORM), маршрутизацію, систему шаблонів, систему авторизації,

адміністративний інтерфейс, механізми кешування та тестування. Даний фреймворк чудово підходить для проєктів типу Model-View-Controller (MVC), завдяки своїй простоті, масштабованості та безпеці.

Однією з найсильніших сторін Django є ORM, що дозволяє працювати з базою даних по аналогії з об’єктами Python, тобто без використання Structured Query Language (SQL). Такий механізм значно спрощує розробку, скорочує кількість повторюваного коду, а також підвищує безпеку. Окрім того, Django має потужний вбудований функціонал для автоматичної генерації інтерфейсу керування базою даних, що є надзвичайно корисним для організації бізнес-логіки та внутрішніх інструментів.

Ще однією перевагою даного фреймворку є легке інтегрування з екосистемою мови Python, що робить його популярним вибором у проєктах, пов’язаних із аналізом даних, машинним навчанням чи штучним інтелектом. Однак, гнучкість Django обмежується його монолітною природою – він потребує конкретної структури проєкту і дотримання визначенного архітектурного стилю. У деяких випадках це стає обмеженням, особливо при роботі з мікросервісами або дуже нестандартними API.[19]

2.3.2 Node.js

Node.js – це середовище виконання мови JavaScript, побудоване на рушії V8 від Google Chrome, яке дозволяє запускати JavaScript код на сервері. Воно базується на неблокуючій, подієорієнтованій моделі вводу-виводу, що робить його ідеальним для застосунків, які потребують високої продуктивності при обробці великої кількості одночасних з'єднань, наприклад у чатах, стримінгу чи RESTful API. Серед фрейморків на базі Node.js найбільшою популярністю користуються Express.js та NestJS.

Express.js – це легковаговий фреймворк, що працює поверх Node.js та дозволяє швидко реалізовувати маршрутизацію, обробку HTTP-запитів та

будувництво API. На відміну від Django, він не нав’язує суворої структури, що задовільняє потребу у гнучкості платформи, але водночас вимагає ручної організації проєкту, що може викликати додаткові проблеми у великих системах. Ще однією позитивною особливістю Express.js є підтримка роботи проміжних функцій (middleware). Це дозволяє оперативно та структуровано реалізовувати різноманітний функціонал, як-от автентифікацію, обробку помилок чи перевірку значень.

NestJS – ще один приклад сучасного фреймворку для Node.js, який написаний на TypeScript. Дана технологія пропонує архітектуру на базі модулів і контролерів, подібну до Angular, та має вбудовану підтримку ін’єкції залежностей (Dependency Injection), інтерфейсів, GraphQL, вебсокетів, мікросервісів та OpenAPI (Swagger). Окрім цього, NestJS підтримує використання мікросервісної архітектури, завдяки чому він ідеально підходить для складних корпоративних застосунків, де важлива чітка структура і масштабованість. Однак, через такі механізми абстракції, даний фреймворк має вищу складність, аніж його аналог Express.js, завдяки чомувиникає потреба в наявності додаткових знань та глибшому розумінні організації системи та її залежностей.

Головна перевага Node.js платформи полягає в можливості використовувати єдиною технологією – JavaScript або TypeScript – як для фронтенду, використовуючи один з популярних фреймворків цього напряму – React, Vue чи Angular – так і для бекенду, що спрощує розробку та налаштування проєктного середовища. Крім того, нині доступна величезна кількість бібліотек для будь-яких потреб, які можуть бути легко інтегровані у систему завдяки Node Package Manager (npm). Проте варто відмітити, що при використанні даної технології встановлення залежностей, обробка помилок та керування асинхронним кодом потребує особливої уваги і дотримання стандартизованих документацією практик.[20]

2.3.3 Spring Boot

Spring Boot – це надбудова над фреймворком Spring, що в свою чергу базується на мові програмування Java, яка дозволяє створювати повноцінні вебзастосунки без необхідності додаткового опису великої кількості конфігурацій. Як і Django, дана технологія використовує принцип Convention over Configuration, що забезпечує простий процес запуску, тестування та деплойменту Java-застосунків.

Утім на відміну від згаданих вище альтернативних фреймоворків для реалізації бекенду, Spring Boot надає набагато більше можливостей для тонкого, детального налаштування. Він глибоко інтегрований із інфраструктурними рішеннями типу Kafka, RabbitMQ, Redis, PostgreSQL, MySQL, ElasticSearch, а також має вбудовану підтримку REST, GraphQL, WebSockets і повноцінного контролю безпеки через Spring Security. Завдяки цим особливостям Spring Boot є досить ефективним у створенні великих корпоративних систем, банківських платформ, страхових сервісів, завдяки підтримці складних бізнес-процесів, високій надійності та стабільності. Даний фреймворк легко масштабується і добре працює у мікросервісній архітектурі разом зі Spring Cloud.

У поєднанні з потужним функціоналом, головним недоліком Spring Boot є складність використання для новачків. Повноцінне володіння цією технологією вимагає фундаментальних знань мови Java, принципів ООП, ін’єкції залежностей та інших аспектів, властивих Java-екосистемі. Також варто відмітити, що написані цією мовою застосунки споживають більше ресурсів комп’ютерної системи, ніж, наприклад, Node.js або Python-застосунки, що може стати одним з основних факторів при виборі тієї чи іншої технології для реалізації бекенду системи.[21]

2.4 Іструменти для інтеграції голосовогу вводу

2.4.1 Актуальні технології для обробки аудіоконтенту

Серед інструментів та підходів для реалізації підтримки голосового вводу можна відмітити декілька варіантів, серед яких використання безкоштовних API для транскрибування тексту, кастомної моделі обробки живої мови (NLP), open-source систем чи хмарних сервісів. Кожен підхід має свої особливості реалізації, які варто брати до уваги під час проектування та розробки системи.

Перший варіант із використанням загальнодоступних API для перетворення аудіодоріжки в текст є найпростішим та найшвидшим варіантом реалізації серед усіх запропонованих. Як приклад такого інструменту можна навести Web Speech API, що являє собою стандартну повністю безкоштовну вебтехнологію для розпізнавання мовлення безпосередньо у веббраузері. Таким чином її використання не потребує встановлення жодних додаткових пакетів чи підключення сторонніх сервісів. Згідно із відкритою документацією, Web Speech API підтримується усіма найпопулярнішими сучасними веббраузерами, що задовільняє використання цього інструменту широкою аудиторією користувачів. Окрім цього даний API дозволяє обробляти аудіопотік в режимі реального часу, роблячи його потужним інструментом для вебплатформ.[22]

Серед недоліків доцільно відмітити обмежену мовну підтримку, а також залежність від стабільного підключення до мережі. Окрім того, Web Speech API не призначений для обробки комплексних довгих запитів, де потрібна надвисока точність аналізу. Ці особливості роблять його зручним для використання лише у невеликих додатках чи прототипах до майбутніх систем.

Налаштована під конкретну мету NLP модель – це спеціалізована мовна модель (нейронна мережа), натренована на домен-специфічних даних. Така модель може бути побудована повністю самостійно, так і мати в своїй основі вже розроблені та частково натреновані екземпляри. Сьогодні існує багато популярних фреймворків для розробки такого роду моделей: spaCy, Hugging Face Transformers, Rasa NLU тощо. Основною перевагою цього методу розробки є гнучкість у налаштуванні контексту роботи, адже мережу можливо підкорегувати під вузькі спеціалізації, задля розпізнавання більш специфічних чи навпаки завуальованих запитів. Так загальні фрази, як-от “мені важко спати останній тиждень”, розпізнаються кастомною моделлю як важливі тривожні симптоми, а не просто невалідний набір слів. Звідси випливає ще одна перевага – можливість ідентифікувати ознаки через лінгвістичні патерни, а не чітковизначене формулювання. Окрім цього варто відмітити гнучкість адаптації та тренування моделі для конкретної мови, що є вагомим аспектом при розробці платформи для різного роду цільової аудиторії.

Утім такий підхід має і свої недоліки. Перш за все, при використанні мовної моделі з'являється проблема наявності відповідного датасету. Для коректного навчання мережі в рамках реабілітаційної платформи потрібно забезпечити достатній набір даних з медичної галузі. Це може стати значним викликом, адже публічні набори такого роду даних, а особливо українською мовою, є рідкістю. Формування набору, побудова та тренування моделі потребують значних часових та апаратних ресурсів навіть при використанні претренованої мережі. Це, відповідно, є значним недоліком для невеликих за складністю та об'ємом застосунків. Також результати аналізу нейронної мережі не є детермінованими. У деяких випадках важко пояснити чому модель зробила той чи інший висновок. У сфері реабілітації ціна похибки є знадто високою, а тому без належного тестування та навчання моделі її використання не є повністю безпечним.[23]

Хмарні сервіси розпізнавання мовлення, такі як Google Cloud Speech-to-Text, Azure Speech Services та Amazon Transcribe, надають досить потужний та точний функціонал розпізнавання аудіоконтенту. Через наявність в їх основі великих нейронних мереж, вони підтримують транскрибування десятками мов з урахуванням різних акцентів чи фонових шумів. Окрім цього дані сервіси дозволяють додавати власні словники до налаштувань для розпізнавання спеціалізованих вузьконаправлених термінів. Серед інших переваг є швидка інтеграція в будь-яку платформу та підтрика автоматизованого розпізнавання мови запиту, що виключає необхідність чітко визначати даний параметр. Серед основних недоліків вирізняється необхідність передавати аудіо на зовнішній сервер, що може створювати проблеми з конфіденційністю чутливих даних. Також потужний набір можливостей мовних моделей не є вільним у використанні та потребує фінансових витрат, які зростають зі збільшенням кількості запитів.

Останнім підходом для інтеграції обробки голосового вводу є використання open-source рішень, наприклад Whisper, Vosk чи DeepSpeech. Дані інструменти надають можливість повністю автономного розпізнавання мовлення без залежності від зовнішніх API. Зокрема Whisper демонструє виняткові показники точності, працює з більш ніж 90 мовами та підтримує трансляцію мовлення з аудіофайлів або потоків. Перевагою open-source систем є повний контроль над даними, що є критично важливим для застосунків специфічної тематики. Також на відміну від використання послуг сторонніх сервісів чи написання власної моделі, у цьому варіанті рішення відсутні витрати на API та наявна можливість кастомізації моделей. Також присутня підтримка офлайн-режиму, що робить open-source моделі придатними для віддалених або захищених середовищ. Однак використання цих рішень вимагає значних технічних зусиль: розгортання та налаштування середовища, управління ресурсами. Особливо негативний аспект має при використанні

моделей Whisper, які потребують значних обчислювальних потужностей та грунтовних технічних знань.

2.4.2 Порівняння наявних рішень інтеграції голосового вводу

Для кращого співставлення особливостей та відокремлення переваг і недоліків кожного варіанту інтеграції обробки голосового вводу в вебплатформах, нижче наведено порівняльну таблицю.

Таблиця 2.1 – Порівняльна таблиця наявних рішень інтеграції голосового вводу

	Web Speech API	**Кастомна NLP-модель**	**Open-source системи (на прикладі Whisper)**	**Хмарні сервіси (на прикладі Google STT)**
Основна задача	Транскрипція голосу в текст	Розпізнавання сутностей, намірів, команд	Транскрипція (можлива інтеграція з NLP)	Транскрипція (можлива NLP-обробка)
Підтримка мов	Обмежена	Залежить від навчання моделі	~100	100+
Якість розпізнавання	Середня	Висока	Висока	Висока

Продовження таблиці 2.1

	Web Speech API	**Кастомна NLP-модель**	**Open-source системи (на прикладі Whisper)**	**Хмарні сервіси (на прикладі Google STT)**
Швидкість роботи	Висока	Середня	Висока	Низька
Складність інтеграції	Низька	Висока	Середня	Середня
Масштабованість	Обмежена	Висока	Обмежена	Висока
Гнучкість	Низька	Висока	Висока	Середня
Можливість офлайн роботи		+	+	

Продовження таблиці 2.1

	Web Speech API	**Кастомна NLP-модель**	**Open-source системи (на прикладі Whisper)**	**Хмарні сервіси (на прикладі Google STT)**
Потреба в додаткових ресурсах (CPU/GPU)		+	+	
Наявність безкоштовного доступу	+	+	+	
Підходить для невеликих проєктів/MVP	+		+	+

2.5 Інші іструменти для розробки вебплатформи

2.5.1 MySQL

MySQL – це реляційна система керування базами даних (RDBMS), яка використовує мову SQL для організації, зберігання та доступу до даних. Є однією з найпопулярніших систем управління базами даних (СУБД) та має безкоштовне програмне забезпечення, яке широко застосовується для зберігання та обробки структурованих даних. Дана технологія здобула популярність завдяки своїй стабільності, простоті використання та потужному наборі можливостей для розробки вебзастосунків і корпоративних систем. Вона підтримує стандартні функції реляційних баз даних, такі як таблиці, індекси, зв'язки між таблицями, транзакції та нормалізація даних.

Така технологія роботи із базами даних, навідміну від нереляційних, має особливу користь у разі потреби в подальшій обробці збереженої інформації, наприклад при проведенні статистичного аналізу чи пошуку полів відповідно до заданих ідентифікаторів. Окрім цього реляційна модель дозволяє створювати складні зв’язки між різними наборами даних, забезпечуючи реалізацію взаємозалежного функціоналу застосунку. Ще однією особливістю є високий рівень безпеки, адже MySQL надає механізми шифрування для захисту даних, підтримує аутентифікацію за допомогою паролів, а також має можливість налаштування прав доступу для різних користувачів. Попри потужний набір процедур, дана реляційна система не втрачає у показниках швидкодії та продуктивності, так як є оптимізованою для роботи з великими обсягами даних. Також доцільним буде зазначити, що простота у використанні підкріплюється наявністю широкої документаційної бази та активного ком’юніті, тому використання цього механізму організації бази даних для

платформи зменшуватиме ймовірність появи неочікуваної проблеми без можливості вирішення.

Серед недоліків варто відмітити, обмежену підтримку великих за обсягом транзакцій та складних запитів. Окрім цього в MySQL не забезпечена підтримка деяких типів даних, що без проблем зберігаються при використанні інших технологій, як-от PostgreSQL, Oracle або SQL Server. Ця система найкраще себе рекомендує при використанні у невеликих проєктах, а для складних застосунків із потребами у надвисокій доступності або специфічних типах даних не є доцільною у використанні.[24]

2.5.2 Auth0 та JWT

Auth0 – це платформа для управління аутентифікацією та авторизацією, яка дозволяє розробникам швидко інтегрувати механізми входу в систему, реєстрації користувачів, керування сесіями та захисту доступу в веб і мобільних застосунків. Auth0 забезпечує технологію Authentication-as-a-Service (AaaS), тобто сервіс, який бере на себе всю логіку перевірки користувачів, реалізовуючи високий рівень безпеки та дозволяючи такими чином сконцентруватись лише на розробці головного функціоналу застосунку. Платформа підтримує кілька стандартів безпеки, зокрема OAuth 2.0, OpenID MFA, SAML, а також інтеграцію з JSON Web Token (JWT). Це дозволяє розробникам створювати гнучкі й безпечні системи доступу для користувачів, партнерів чи внутрішніх команд.[25]

JSON Web Token (JWT) – це відкритий стандарт, який визначає компактний і самодостатній спосіб безпечної передачі інформації між сервером та клієнтом у форматі JSON. Найчастіше дану технологію використовують для реалізації механізмів аутентифікації та авторизації в вебзастосунках та мобільних додатках, особливо в архітектурах без стану (stateless), таких як REST API.

JWT складається з трьох основних частин:

- Header – заголовок, що описує тип токена та алгоритм його підпису (шифрування);
- Payload – загальні дані, що передаються. Зазвичай містить інформацію про користувача та метадані щодо видавця токена, ідентифікатора користувача, часу створення та терміну дії;
- Signature – підпис, що створюється шляхом підпису першої частини токена – header та payload – з використанням секретного ключа або приватного ключа, в залежності від симетричного чи асиметричного механізму шифрування відповідно. Завдяки цьому пізніше відбувається перевірка валідності токену загалом.

Серед переваг використання механізму JWT є його самодостатність, адже токен містить усю інформацію, яка необхідна для перевірки користувача, через що серверу не потрібно зберігати сесійний стан чи виконувати інший функціонал для забезпечення коректного захисту даних. Це робить JWT ідеальним для масштабованих REST API, мікросервісів і застосунків загалом. Іншою перевагою є підтримка асиметричної криптографії, наприклад RSA, що дозволяє підписувати токен на одному сервері й перевіряти на іншому без потреби в окремій передачі секретного ключа. Також JWT є сумісним із більшістю мов програмування та легко передається через HTTP-заголовки, cookies або WebSockets. Оскільки токени можуть мати час життя, через ідентифікатор exp в блоці payload, доступ до тих чи інших даних не є вічним, що дозволяє запобігти виникненню фатальних несанкціонованих крахів системи через витік інформації.

Серед недоліків варто відмітити те, що токен неможливо відкликати без додаткового створення “чорного списку”, через що ускладнюється реалізація термінового блокування користувачів чи повноцінного логіну/логауту. Також при надто довгому терміні дії токену, зловмисник може скористатися ним

навіть після зміни пароля або ролі користувача. Ці аспекти потребують особливої уваги при використанні даної технології.[26]

Тим не менш дані способи реалізації безпеки є досить розповсюдженими рішеннями для платформ, що реалізовують функціонал особистих кабінетів чи інші механізми доступу користувачів різних ролей до тих чи інших даних.

2.5.3 Google Cloud Platform

Google Cloud Platform (GCP) - це набір хмарних сервісів та інструментів, які надаються компанією Google для розробки, розгортання та масштабування вебзастосунків, сервісів та інфраструктури. Вона входить до трійки лідерів світового ринку хмарних платформ поряд із Amazon Web Services (AWS) та Microsoft Azure.

Основною особливістю GCP є широкий діапазон пропозицій, висока продуктивність, надійність та безпека, яка забезпечується глобальною інфраструктурою Google. Дана платформа включає понад 100 сервісів, які охоплюють обчислення, зберігання, бази даних, штучний інтелект, машинне навчання, DevOps, аналітику даних та безпеку. Серед інших переваг варто відмітити тісну інтеграцію з іншими сервісами компанії, такими як Firebase, Chrome, Youtube чи Android Studio. Попри те, що користування сервісами є платним, надається трьохмісячний доступ у розмірі 300 доларів США для нових користувачів, що забезпечує легкий старт у використанні тих чи інших API.[27]

2.6 Обрані технології для розробки вебплатформи для психофізичної реабілітації

Відповідно до аналізу наявних інструментів та методологій для реалізації архітектури, бекенд та фронтед частини сучасних вебплатформ сформовано відповідний набір технологічних рішень.

З огляду на складність та розмір майбутньої системи для реалізації архітектурної моделі обрано відштовхуватись від монолітного типу. Зокрема для реалізації фронтенд частини вибір припав на просту модульну архітектуру, задля забезпечення приципів інкапсуляції та логічного розмежування функціоналу.

Фреймворком для реалізації користувацького вебінтерфейсу обрано React завдяки його компонентній структурі, особливості роботи із DOM та легкій інтеграції із багатьма іншими інструментами технологічного стеку, у тому числі із Node.js та фреймворком Express.js, який був обраний для реалізації бекенд частини через свою універсальність, швидкодію та гнучкість.

Окрім цього, для розширення функціоналу майбутньої платформи інтегровані такі інструменти та сервіси, як MySQL для роботи з реляційною базою даних, Auth0 для реалізації високого рівня безпеки, GCP для доступу до сторонніх даних через API та Web Speech API для інтеграції обробки голосового вводу.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі проведено дослідження наявних підходів до організації сучасних вебплатформ, а також інструментів реалізації обидвох частин системи - фронтенду та бекенду. Кожен варіант має низку переваг та недоліків, що в тій чи іншій мірі впливають на вибір залежно від розміру, складності та можливостей реалізації бажаної вебсистеми.

Серед ряду наявних варіантів сформовано набір інструментів для реалізації технічних вимог до майбутньої системи та забезпечення високих показників гнучкості, швидкодії, безпеки та можливості до масштабування у майбутньому. Зокрема в якості архітектури для скелету розробки обрано монолітну та просту модульну моделі, для реалізації яких використовуватиметься React та Express.js через особливості в зручній співпраці, підтримці широкого вибору додаткових бібліотек, та високих показниках швидкої. Окрім цього для реалізації певного функціоналу, обов’язкового для сучасних вебсистем, як-от запобігання несанкціонованого доступу, чи функціоналу, що вирізнятиме власний продукт серед конкурентів на ринку, як-от надання певної інформації завдяки додатковому доступу через сторонні API, інтегровано додаткові сервіси – Auth0 та GCP. Для збереження даних обрано реляційну СУБД MySQL. Як інструмент для підтримки функціоналу голосового вводу обрано Web Speech API завдяки його високим показникам продуктивності, мінімальній затримці обробки мовлення в реальному часі та легкості в інтеграції з середовищем розгортання платформи – веббраузером.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ДЛЯ ПСИХОФІЗИЧНОЇ РЕАБІЛІТАЦІЇ

Відповідно до парадигм обраної архітектури кодова реалізація застосунку була поділена на низку модулів-директорій. Зокрема клієнтська частина складається з елементів (elements), що являють собою найменші одиниці користувацького інтерфейсу, наприклад кнопки чи поля вводу, компоненти (components), які мають в своїй основі низку елементів та реалізують певну бізнес логіку, та сторінки (pages), що є найбільшими UI елементами. Загальні функції обробки даних, як-от приведення дати до єдиного стандарту чи реалізація приватного React маршруту, особливості та мету використання яких наведено у настпуних підпунктах розділу, також винесено в окрему директорію сервісів (services).

Серверна частина хоч і має в своїй основі монолітну структуру поділена на окремі логічні директорії зі своїми зонами відповідальності. Так окремо визначені контролери (controllers) – функції обробки запитів до сторонніх сервісів чи бази даних, маршрути (routes), де відбувається виклик функції контролерів, проміжні функції (middleware) для декомпозиції обробки даних, наприклад фільтрація обраних щоденних завдань чи перевірка справжності токену доступу, та головний файл серверу, де відбувається його ініціалізація та запуск на визначеному змінними середовища порті. Графічне представлення структури вебплатформи наведено у додатку 1.

Відповідно до технічних вимог система повинна реалізовувати низку завдань. Основними серед них є надання користувачам можливості реєстрації та авторизації, вести трекінг власного стану та отримувати статистичний аналіз показників впродовж певного періоду, а також отримувати щоденні реабілітаційні завдання та відеоінструкції до них відповідно до актуального

стану здоров’я. Детальний алгоритм взаємодії користувача з системою наведений у додатку 2. Функціональна схема вебплатформи наведена у додатку 3.

3.1 Реалізація реєстрації та авторизації

Реєстрація користувача відбувається шляхом звернення до відповідної точки доступу (endpoint) стороннього сервісу Auth0 та інтегрованої з ним бази даних. У тілі запиту окрім введених користувачем значень поштової скриньки та паролю присутні такі параметри, як ідентифікатор користувача сервісу та тип з’єднання «Username-Password-Authentication», що в свою чергу закріплює за новоствореним користувачем пару облікових даних логін-пароль та можливість майбутнього входу в систему за допомогою неї. Окрім створення користувача на стороні сервісу Auth0, відбувається його додаткова ініціалізація у локальній базі даних, що значно спрощує навантаження на систему при майбутній взаємодії з платформою.

Авторизація виконується аналогічним чином. Відбувається звертання до відповідного ендпоїнта, передаючи у тілі запиту значення аудиторії застосунку на сервісі Auth0, ідентифікатора клієнта та секретного ключа клієнта. Окрім цього вказуються специфічні набори прав доступу, серед яких openid, який вказує на відповідність протоколу OpenID Connect, що забезпечує отримання токену доступу (access token) з інформацією про користувача, profile, що дозволяє отримати основні поля профілю користувача, як-от його унікальний ідентифікатор, email – для підтвердження справжності користувача завдяки надсиланню додаткового листа з посиланням для верифікації акаунту та offline_access, що забезпечує отримання токену оновлення (refresh token) для оновлення токену доступу без повторної процедури авторизації. Після успішної аутентифікації сервер вебплатформи

отримує два види JWT – токен доступу, що є основним способом перевірки справжності користувача та має відносно невеликий час життя (зазвичай не більше п’ятнадцяти хвилин) та оновлення, що є допоміжним для продовження тривалості користувацької сесії та має більший час життя (близько 30 днів) – та зберігає обидва в файлах cookies – невеликих текстових файлах, які вебсайт записує у браузері користувача для збереження інформації, такої як сесійні дані, налаштування або токени аутентифікації. Вони автоматично надсилаються назад на сервер під час наступних запитів до того ж домену. Такий механізм є доцільним рішенням з точки зору безпеки та зручності інтеграції. Зокрема, cookies забезпечують захист токенів від доступу зі сторони JavaScript, що мінімізує ризик викрадення токенів у випадку XSS-атаки.

3.2 Інші механізми безпеки

Окрім інтеграції стороннього сервісу для реалізації високого рівня безпеки застосунку, використано й додаткові механізми для покращення її стану.

Перший – проміжна функція jwtCheck, що відповідно до своєї назви перевіряє справжність отриманого токену доступу. Для вирішення цієї задачі використано метод verify бібліотеки «jsonwebtoken», який забезпечує перевірку підпису токена, його цілісності та відповідності вказаним параметрам, зокрема, секретному ключу. Якщо токен не проходить перевірку, система припиняє подальше виконання запиту, захищаючи ресурс від несанкціонованого доступу.

Окрім цього, функція jwtCheck реалізує оновлення токена в тому випадку, якщо ключ відсутній або його час життя завершився. Така логіка

дозволяє підтримувати безперервність сесії користувача, не змушуючи його повторно вводити облікові дані, за умови наявності дійсного refresh-токена.

Таким чином унеможливлюється підробка унікального ключа, а також забезпечується динамічне оновлення доступу, що відповідає сучасним підходам до захисту RESTful-інтерфейсів. У результаті jwtCheck виступає не лише засобом верифікації, а й захисним бар'єром між клієнтською частиною та захищеними ресурсами бекенду, реалізуючи централізований контроль авторизації та автоматичне управління життєвим циклом сесій. Такий підхід значно знижує ризики, пов'язані з крадіжкою або використанням підробних токенів.

Другий додатковий механізм – використання приватного React маршруту, що виступає у якості обгортки компонент для обмеження доступу до них для неавторизованих користувачів. Такий маршрут, реалізований через спеціальний компонент PrivateRoute, інтегрується в структуру клієнтської маршрутизації та працює як логічний фільтр: під час спроби переходу за внутрішнім шляхом React Router завантажує зміст захищеного компонента не одразу, а лише після проходження перевірки відповідності користувача умовам доступу. У контексті реалізованої платформи перевірка здійснюється на основі наявності та дійсності токена доступу. У випадку його відсутності або недійсності, замість завантаження очікуваного контенту користувача перенаправляють на сторінку входу, тим самим запобігаючи витоку або несанкціонованій взаємодії із чутливою інформацією.

Таким чином, приватні маршрути не лише створюють додатковий рівень захисту на клієнтському боці, але й сприяють підвищенню продуктивності застосунку, що особливо важливо для SPA. Також такий підхід спрощує розробку й обслуговування клієнтської частини застосунку: логіка доступу централізовано управляється в React, а маршрути не дублюються на сервері.

3.3 Реалізація трекінгу стану

Іншою вагомою особливістю системи є забезпечення можливості відслідковування стану здоров’я користувача впродовж певного періоду часу. Для реалізації такого завдання були використані як невеликі UI елементи, як-от кнопки, поля вводу та вибору, так і більші компоненти зі своїми зонами відповідальності, наприклад календар чи графік. Для інтеграції останніх були використані додаткові React бібліотеки «react-calendar» та «recharts» відповідно, що надають широкий та зручний інструментарій.

Для відслідковування користувачеві пропонуються такі параметри, як тривалість сну, тренувань та відпочинку, а також рівень стресу та енергії. Задля забезпечення отримання правдивих даних та коректної їх обробки у майбутньому відбувається додаткова перевірка на введені значення. Так показники тривалості передбачають числові значення від 0 до 24, а рівні стресу та енергії – один з п'яти варіантів відповіді: дуже низький (very low), низький (low), середній (middle), високий (high) та дуже високий (very high). Окрім цього користувач має змогу вести додаткові довільні нотатки, заповнюючи відповідне поле.

Запуск режиму голосового вводу реалізований за допомогою натиску на відповідну кнопку «Use Voice Input». В цьому режимі з’являється модальне вікно з текстовими інструкціями, полем відображення проміжного результату розпізнавання та двома кнопками «Start» та «Stop». Після старту та надання дозволу на використання браузером мікрофона відбувається перетворення отриманої аудіодоріжки в текст за допомогою Web Speech API. Варто зазначити, що система підтримує англійську мову, а також автоматично ототожнює розпізнаний текст чисел із словесними їх аналогами завдяки додатково визначеному словнику відповідностей. Іншими словами, результат розпізнавання числа «10» цифрами рівен його прописному варіанту –

«десять». Після отримання перетвореного з аудіо тексту, система викорінює ключові слова та заповнює відповідні поля бази даних потрібними значеннями. Варто відмітити, що у разі помилки користувач має змогу вручну відредагувати результат вводу, замінивши помилкові значення на потрібні. Також задля коректної організації трекінгового процесу система блокує спроби заповнити дані «завтрашнього дня», в той час як редагування днів, що минули, можливе без будь-яких проблем.

В окремому блоці сторінки трекінгу під назваю «Statistical Analysis» відображаються середні значення введених даних впродовж обраного періоду – останнього тижня, місяця чи трьох місяців. Для наочнішого представлення поруч оголошений компонент графіка, який демонструє зміну кожного числового поля, що відслідковується.

3.4 Реалізація призначення щоденних завдань

При першому вході у систему користувачеві пропонується пройти опитування для визначення його контактної інформації, актуального стану здоров’я, бажаних цілей, комфортного темпу вправ тощо. Отримані дані додаються до окремої таблиці бази даних «users», повністю заповнюючи профіль споживача та слугуючи при цьому основою для призначення низки вправ. Список можливих завдань є сталим та реалізований у вигляді таблиці «rehab_tasks» (рис. 3.1). Варто відмітити, що кожне поле, окрім основного тексту задачі, містить графу ключових слів та своєрідний «період очікування» cooldown_days, що становить ціле значення від одного до трьох. Цей параметр днів потрібен для забезпечення постійної зміни завдань та мінімізування ймовірності призначення одних і тих же задач щодня, так як виконання складних вправ, що потребують значних фізичних чи моральних зусиль, потенційно може принести пацієнтові більше шкоди, аніж користі. Іншими

словами, якщо задача має показник cooldown_days рівний одиниці, таке завдання може з’явитись у списку активностей користувача раз на два дні; якщо рівний двом – раз на три дні тощо.

id int	* task_text text	keywords json	cooldown_days int
1	5-minute deep breathing session	["Minor limitation", "Menta	2
2	Gentle knee stretches	["Knee", "Moderate limitati	3
3	Gratitude journaling	["Mental", "Moderate limit	1
4	Short walk outdoors	["Physical", "Moderate limi	1
5	Chair yoga for shoulder relief	["Shoulder", "Severe limita	3
6	Do a wall-assisted shoulder mobility drill	["Shoulder", "Physical", "M	3
7	Try a 10-minute guided meditation for stress relief	["Mental", "Manage stress"	2
8	Practice seated leg raises for knee stability	["Knee", "Severe limitation'	3
9	Go on an evening wind-down yoga for better sleep	["Sleep", "Physical", "Build	2

Рисунок 3.1 – Приклади завдань з таблиці «rehab_tasks»

Завдання призначаються на основі ключових слів із пройденого опитування, а також середніх показників тривалості сну, рівня стресу та енергії. Так, якщо середня тривалість сну за тиждень менша 6 годин на добу, система може запропонувати завдання на проведення вечірньої медитації з метою зниження збудження нервової системи. У випадку виявлення підвищеного рівня стресу, користувачеві може бути призначено додаткова прогулянка, що сприяє відновленню психоемоційного балансу. Таким чином, система адаптує завдання до актуального стану користувача, орієнтуючись на дані трекінгу й персональні особливості.

Кожне завдання зі списку можна позначити як виконане, натиснувши на відповідний елемент поруч. Доцільно відмітити, що невирішені задачі додаються до нового списку на наступний день. Утім, щоденна їх кількість не перебільшує трьох, задля запобігання перевантаженню користувача,

зниженню залученості та втраті мотивації продовжувати виконання завдань у довготривалій перспективі.

3.5 Реалізація надання відеоінструкцій

Для дієвого покращення стану користувача, окрім загальних текстових пропозицій вправ надається набір відео інструкцій до кожної з них. Цей список формується завдяки відповідному запиту із текстової складовою кожної вправи до стороннього API – YouTube Data API v3, доступ до якого забезпечено через сторонній хмарний сервіс GCP. У спеціальному блоці інтерфейсу відображаються отримані відео, зокрема їхні назви, автори та короткий опис, що дозволяє швидко зорієнтуватися в їх змісті. Таким чином користувач має змогу одразу переглянути якісний приклад виконання вправи, що знижує імовірність помилок під час виконання, забезпечує краще розуміння рухів або технік та підвищує загальну ефективність взаємодії з програмою. Інтеграція відеоматеріалів також сприяє зменшенню бар'єру входження для новачків та надає додаткову підтримку візуального типу сприйняття інформації.

Також на окремій сторінці реалізовано механізм пошуку відео через спеціальне текстове поле. Це дозволяє користувачеві самостійно знаходити контент за власним запитом, незалежно від автоматично призначених на день завдань, що суттєво розширює надані платформою можливості та сприяє індивідуалізації досвіду.

Окрім цього, система підтримує механізм закладок, що дає змогу зберігати будь-яке відео для подальшого перегляду. Для цього достатньо натиснути на відповідну позначку поруч із відео в списку. Збережені відео відображаються в окремому розділі, доступному через особистий кабінет, що дозволяє легко повертатися до вибраного матеріалу без необхідності

повторного пошуку. Такий підхід не лише покращує зручність користування платформою, але й формує персоналізовану відеобазу, з якою користувач може працювати у власному темпі та відповідно до власних потреб.

3.6 Реалізація мотиваційних елементів

З огляду на підтримку залученості користувачів, у системі реалізовано мотиваційний елемент серій – лічильник днів безперервного внесення даних у трекінгову систему. Цей підхід активно використовується у сучасних цифрових платформах, адже він створює ефект досягнення, формує корисну звичку та сприяє регулярності взаємодії з додатком.

У разі пропуску дня лічильник повертається до початкового значення – нуль, що виконує функцію «м’якого покарання» і стимулює користувача не порушувати безперервність. Такий психологічний тригер формує відповідальність і зменшує вірогідність довготривалого ігнорування застосунку.

Збільшення кількості днів серії супроводжується відображенням випадково обраної підбадьорюючої фрази з попередньо підготовленого масиву. Ці фрази створюють позитивний емоційний фон, викликають відчуття підтримки та можуть слугувати короткими повідомленнями-підсилювачами внутрішньої мотивації.

Таким чином забезпечується стійкий елемент гейміфікації, який, з одного боку, не перевантажує інтерфейс, а з іншого – виконує важливу роль у формуванні емоційної прив’язаності до платформи, підтримці регулярності дій та відчутті прогресу. У сукупності це дозволяє користувачеві не просто виконувати щоденні завдання, а відчувати цінність та успішність у власній діяльності.

ВИСНОВОК ДО РОЗДІЛУ 3

У третьому розділі проведено детальний огляд особливостей та алгоритмів реалізації основних завдань системи, серед яких реєстрація та авторизація користувача, відслідковування стану з можливістю введення актуальних даних вручну та голосом, пошук та збереження відеоінструкцій, отримання та виконання щоденних тренувальних завдань. Також наведений опис додаткових елементів покращення користувацького досвіду використання платформи, серед яких інтеграція елементу гейміфікації – лічильника серії – та щоденних мотиваційних фраз.

Окрім цього у розділі представлені особливості організації роботи системи відповідно до парадигм обраної простої модульної архітектури – поділ функціоналу між окремими модулями відповідно до їх зон відповідальності.

Для наочнішого представлення структури, алгоритму взаємодії користувача із системою та узагальненої функціональності платформи наведено відповідні графічні матеріали у додатках 1-3.

РОЗДІЛ 4
ТЕСТУВАННЯ ВЕБПЛАТФОРМИ ДЛЯ ПСИХОФІЗИЧНОЇ РЕАБІЛІТАЦІЇ

4.1 Тестування системи

Для перевірки справності роботи, зокрема коректності взаємодії користувача із різними модулями системи, проведено юніт-тестування низки сценаріїв, що симулюють усі можливі випадки під час реальної роботи застосунку. Зокрема протестовано роботу механізму реєстрації та авторизації, виходу користувача з системи, отримання та опрацювання токенів доступу та оновлення. Також симульовано різні випадки при роботі із базами даних користувачів, завдань, відслідкованої інформації та закладок. Варто відмітити, що незначні допоміжні функції, такі як отримання різних мотиваційних фраз чи приведення дати до єдиного формату також покриті тестами, задля унеможливлення виникення непередбачуваних крахів системи.

Для реалізації тестів використано бібліотеку Jest, що надає широкий набір інструментів для реалізації тестових сценаріїв. Зокрема активно використовувався механізм імітацій (mock) для роботи з базами даних не напряму, а через визначені екземпляри-дублери. Такий підхід дозволяє ізолювати логіку бізнес-процесів від фактичного з'єднання з базою даних, що забезпечує стабільність, вищий рівень контролю, а також передбачуваність результатів тестування. Завдяки цьому забезпечується швидше виконання тестових сценаріїв та покриття граничних випадків, як-от помилки з'єднання чи повернення порожніх результатів.

За результатами тестування можна стверджувати, що система поводиться передбачувано та коректно, жодних помилок не виявлено. Результати наведені на рис. 4.1 та 4.2.

```
Test Suites: 3 passed, 3 total
Tests:       45 passed, 45 total
Snapshots:   0 total
Time:        1.985 s, estimated 2 s
```

Рисунок 4.1 – Результати тестустування 1

```
Test Suites: 2 passed, 2 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        2.199 s
Ran all test suites.
```

Рисунок 4.2 – Результати тестустування 2

4.2 Використання вебплатформи

При першому відвідуванні вебсайту користувач бачить форму авторизації (рис. 4.3), де пропонується ввести поштову скриньку та пароль.

Рисунок 4.3 – Форма авторизації

У випадку введення помилкових даних, відображаєтсья відповідне повідомлення (рис. 4.4).

Рисунок 4.4 – Повідомлення про введення помилкових даних

Користувачеві надається можливість відновити пароль у випадку його втрати. Для цього потрібно перейти за відповідним посиланням рядка «Forgot password». Як результат, з’являється оновлена форма для отримання листа відновлення на вказану у полі поштову адресу (рис. 4.5).

Рисунок 4.5 – Форма отримання листа для відновлення паролю

Тепер користувач має змогу перейти за відповідним посиланням у отриманому листі та відновити втрачені дані (рис. 4.6).

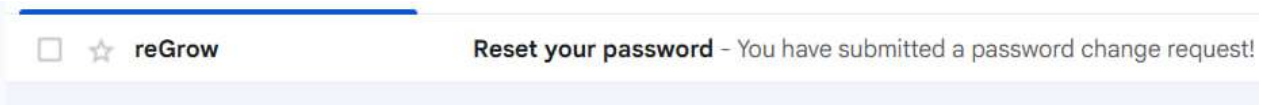


Рисунок 4.6 – Отриманий лист для відновлення паролю

У випадку відсутності акаунту користувача у системі, пропонується пройти процес реєстрації. Для цього реалізована відповідна форма (рис. 4.7).

Рисунок 4.7 – Форма реєстрації нового користувача

Аналогічно до форми авторизації (див. рис. 4.3) у разі неспівпадіння значень паролів, відображається попереджувальне повідомлення «Passwords do not match» (рис. 4.8).

Рисунок 4.8 – Повідомлення про неспівпадіння введених паролей

Після створення нового користувача і відповідно при першому вході в систему, надається можливість пройти опитування для визначення контактної інформації, актуального стану здоров'я, бажаного темпу та типу вправ тощо. Питання та варіанти відповідей наведені на рис. 4.9 – 4.17.

Рисунок 4.8 – Ввідне опитування. Сторінка 1

Рисунок 4.9 – Ввідне опитування. Сторінка 2

Рисунок 4.10 – Ввідне опитування. Сторінка 3

Рисунок 4.11 – Ввідне опитування. Сторінка 4.1

Рисунок 4.12 – Ввідне опитування. Сторінка 4.2

Рисунок 4.13 – Ввідне опитування. Сторінка 5

Рисунок 4.14 – Ввідне опитування. Сторінка 6

Рисунок 4.15 – Ввідне опитування. Сторінка 7

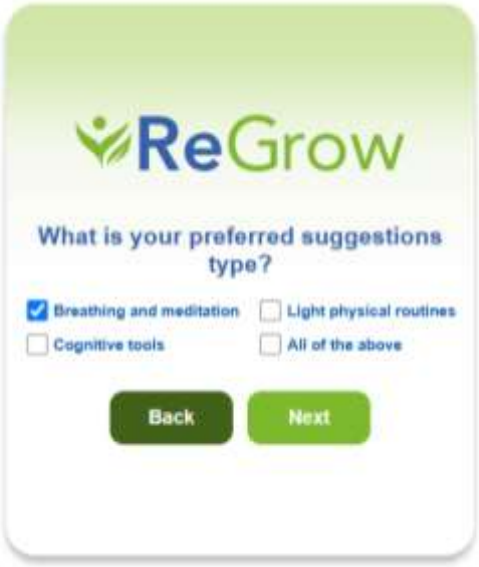


Рисунок 4.16 – Ввідне опитування. Сторінка 8

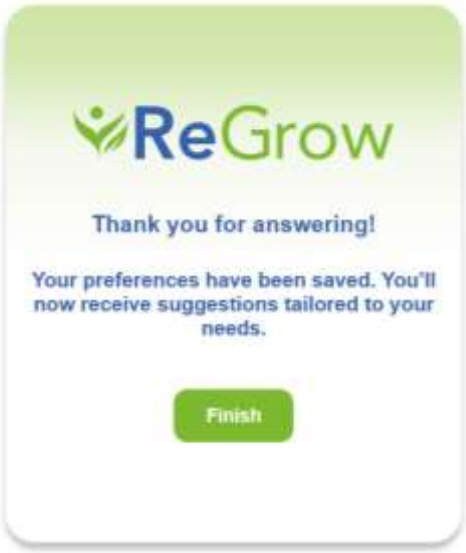


Рисунок 4.17 – Ввідне опитування. Сторінка 9

Після проходження форми користувача перенаправляють на головну сторінку вебплатофрми (рис. 4.18), де він може переглянути та відредагувати у разі потреби власні дані. Також сторінка містить блок із закладками, де у майбутньому відображатимуться збережені відео.

Рисунок 4.18 – Головна сторінка

Для заповнення полів стану на актуальний день реалізована відповідна сторінка «Tracking» (рис. 4.19).

Рисунок 4.19 – Сторінка відслідковування стану

Для заповнення даних потрібно обрати бажану дату на календарі та натиснути на відповідну кнопку «Edit» для активації режиму редагування (рис. 4.20). Варто відмітити, що редагування можливе лише днів до актуального дня включно. У іншому випадку, система блокує можливість вводу значень у поля.

Рисунок 4.20 – Активований режим редагування

Також при введені некоретних значень відображається модальне вікно із зазначенням помилки (рис. 4.21).

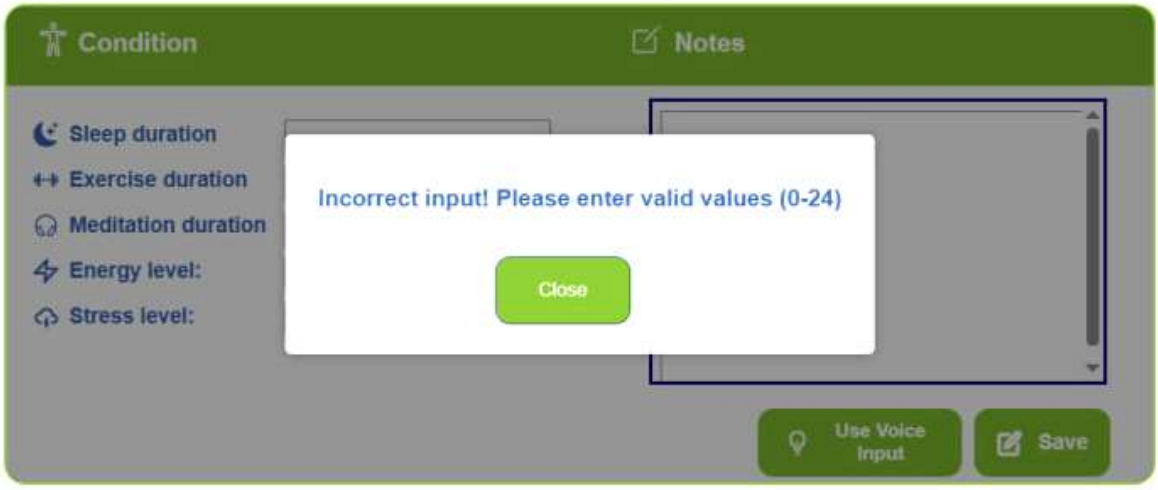


Рисунок 4.21 – Модальне вікно помилки введених значень

Режим голосового вводу активується натиском відповідної кнопки «Use Voice Input». У вікні, що з'явилось, відображаються текстові інструкції, кнопки старту та зупинки, а також поле розпізнаного тексту (рис. 4.22).

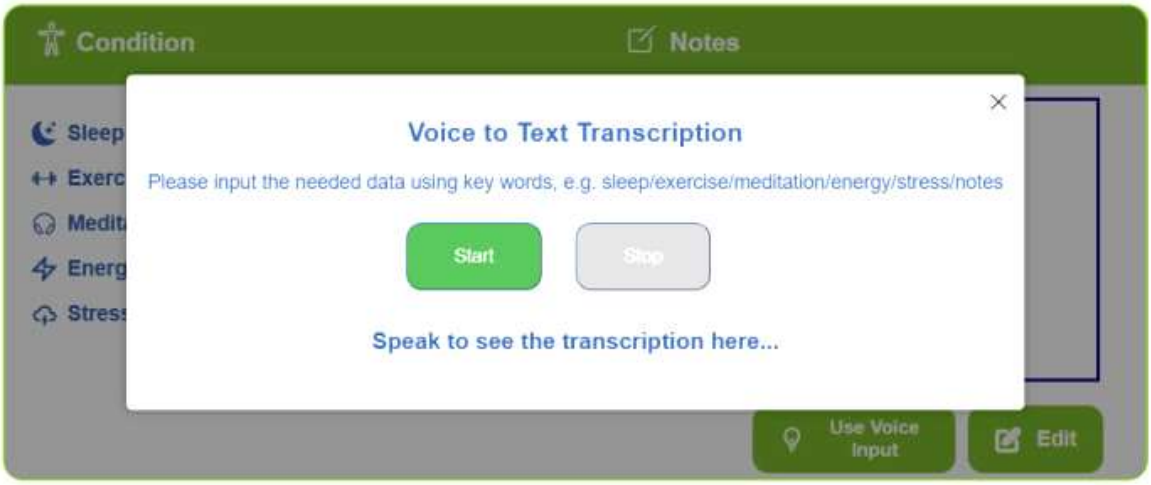


Рисунок 4.22 – Модальне вікно режиму голосового вводу

Після успішного аналізу аудіо, дані автоматично відображаються у відповідних полях. Приклад використання голосового вводу зображений на рис. 4.23 – 4.24.

Рисунок 4.23 – Модальне вікно режиму голосового вводу із розпізнаним текстом

Рисунок 4.23 – Блок відслідковування стану із заповненими полями відповідно до розпізнаного тексту

Після введення інформації впродовж певного періоду (тиждень, місяць, три місяці), користувач має змогу отримати статистичний аналіз стану, що відображається у відповідному блоці сторінки. Поруч знаходиться графік для начного представлення зміни показників тривалості сну, тренувань та відпочинку (рис. 4.24 – 4.26).

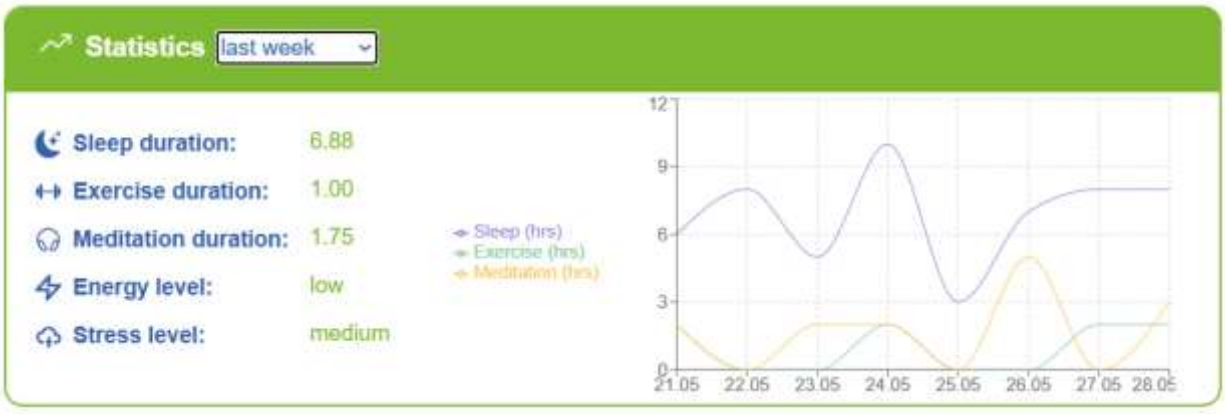


Рисунок 4.24 – Блок статистики за останній тиждень

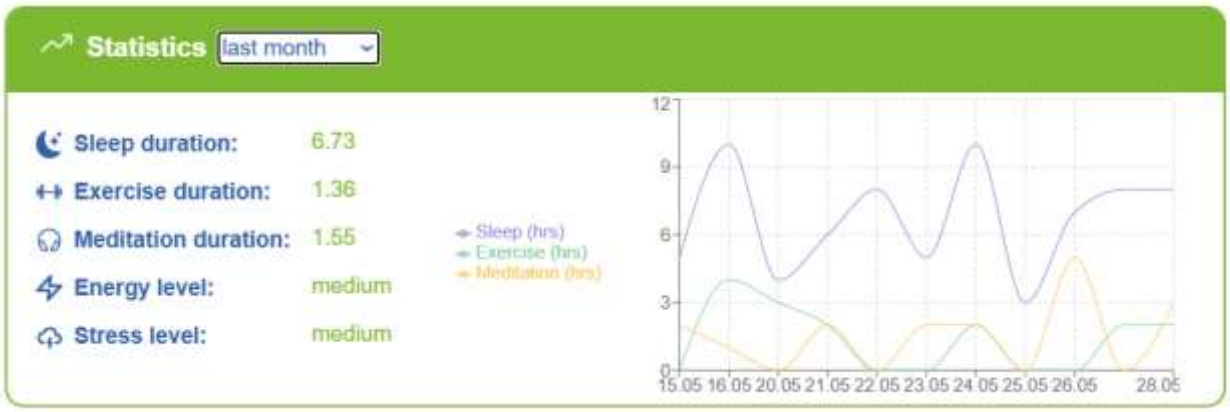


Рисунок 4.25 – Блок статистики за останній місяць

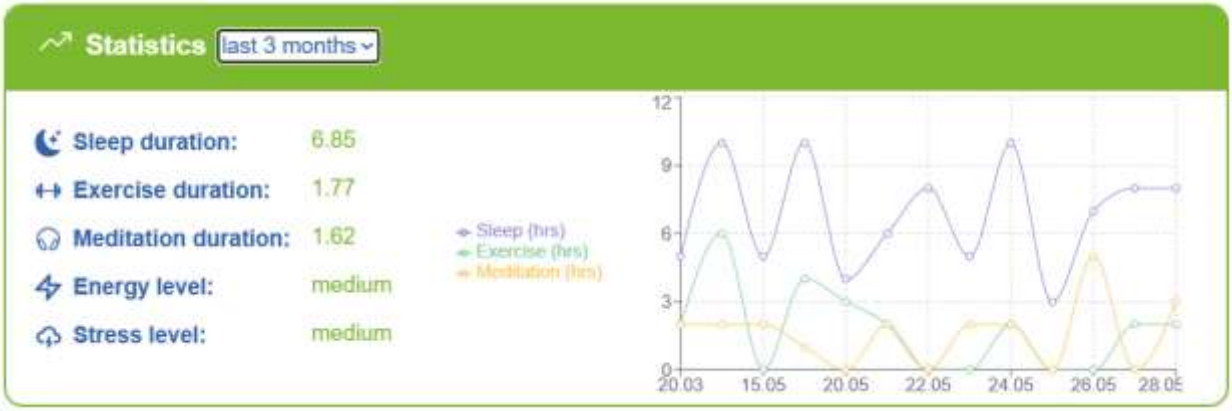


Рисунок 4.26 – Блок статистики за останні три місяці

Сторінка пошуку відео «Search Tutorials» містить у собі текстове поле пошуку та блок із динамічно відображеними відео (рис. 4.27). Користувач має змогу відшукати матеріал будь-якої потрібної тематики та зберегти його у випадку вподоби, натиснувши на відповідну іконку в правому верхньому куті.

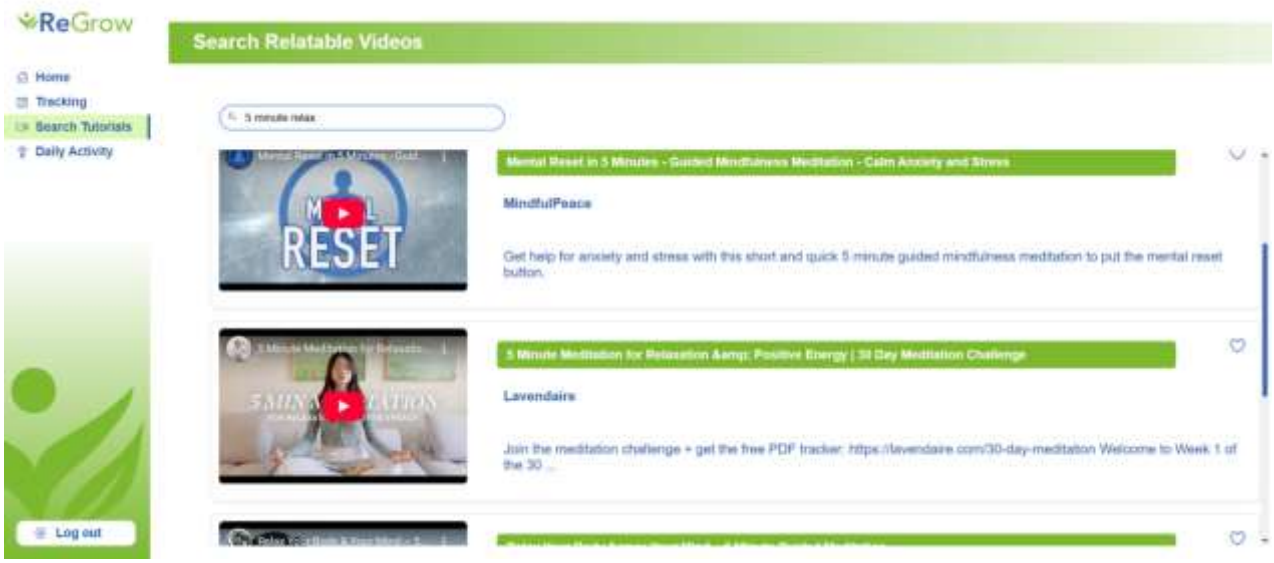


Рисунок 4.27 – Сторінка пошуку відео

Сторінка щоденної активності «Daily Activity» містить блоки актуальних персоналізованих завдань, блок потрібних для їх виконання відео та індикатор серії (рис. 4.28).

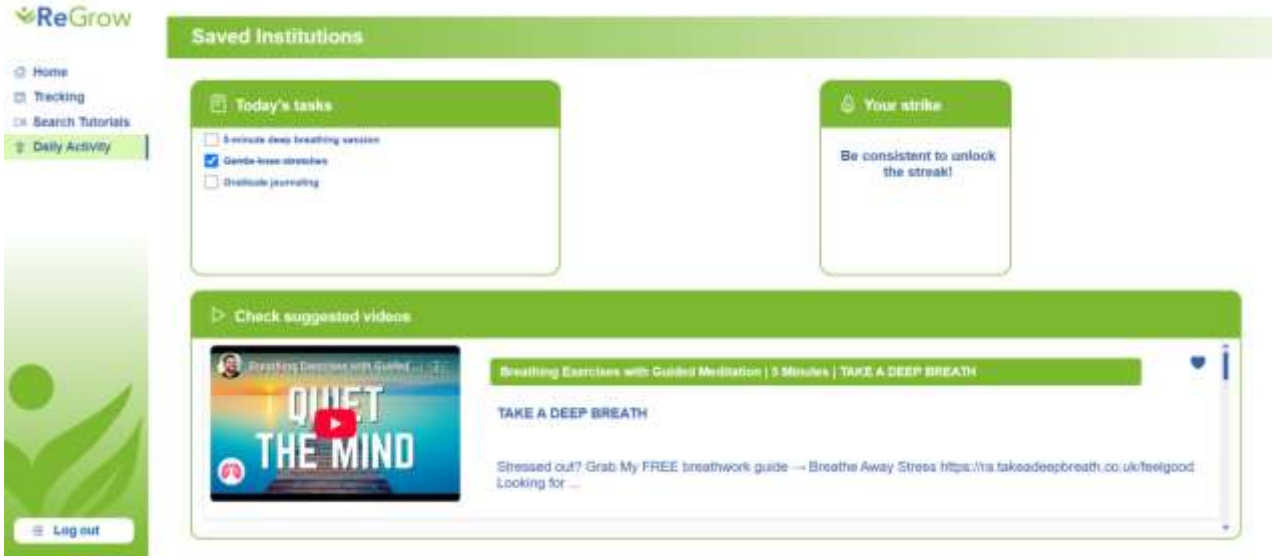


Рисунок 4.28 – Сторінка щоденної активності

Варто відмітити, що до кожного з трьох завдань надається по два відеоприклади, таким чином забезпечується баланс між варіантивністю контенту та помірним використанням API-запитів. Такий підхід дозволяє

уникнути перевищення ліміту на звернення до стороннього сервісу, водночас залишаючи користувачеві можливість обрати найбільш зручне чи зрозуміле відео для виконання завдання. Приклади інших підібраних системою відео відповідно до актуального списку вправ наведені на рис. 4.29 – 4.30.

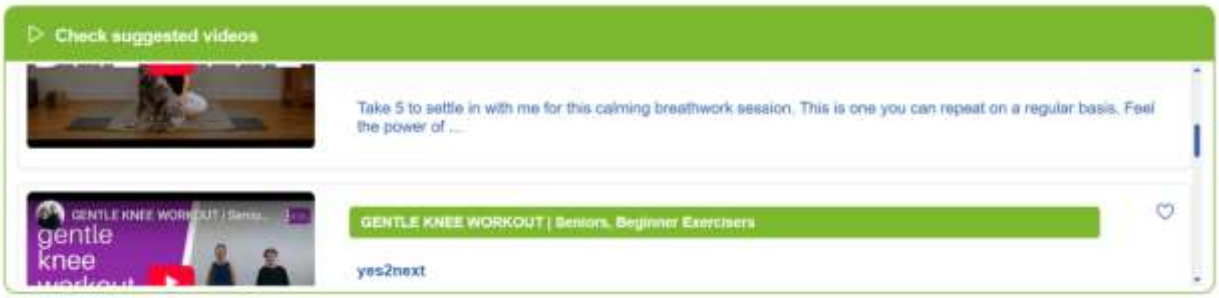


Рисунок 4.29 – Приклади підібраних відео 1

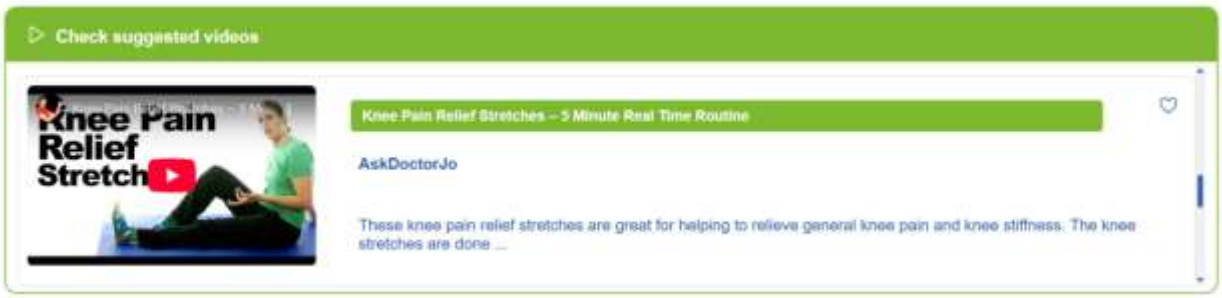


Рисунок 4.30 – Приклади підібраних відео 2

Показник серій при відмінному від нуля значенні змінює відображення. Так в окремому блоці користувачеві представлено значення лічильника та обрана випадковим чином підбадьорююча фраза (рис. 4.31) для підтримки мотивації клієнта.

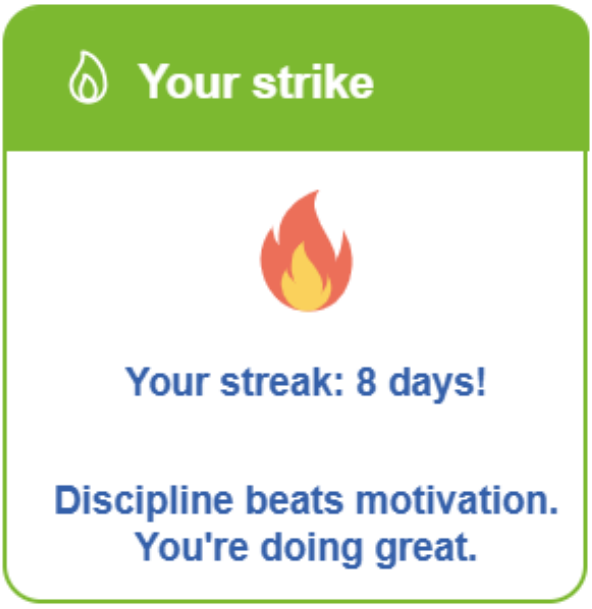


Рисунок 4.31 – Блок серії при значенні лічильника 8

4.3 Можливості для майбутнього розвитку вебплатформи

Хоч реалізована вебплатформа повністю відповідає вимогам сучасних систем та реалізує усі першочергово поставлені завдання, існує низка можливостей для майбутнього удосконалення.

Одним з варіантів розвитку є адаптація інтерфейсу під мобільні пристрої або трансформація платформи у прогресивний вебдодаток (PWA). Такий підхід дозволяє поєднати зручність вебтехнологій із функціональністю нативних застосунків. Зокрема з'являється можливість працювати з вебплатформою у офлайн режимі, підтримки механізму push-сповіщень, що особливо актуально при потребі щоденного відслідковування стану здоров’я, та реалізації кешування для швидшого завантаження змісту платформи та зменшення споживання трафіку.

Іншим можливим удосконаленням є інтеграція з медичними фахівцями та структурами, що дозволить удосконалити процес відновлення користувачів, забезпечивши швидший та оперативніший план реабілітації шляхом

персоналізованого втручання на основі медичних висновків, постійного моніторингу динаміки стану пацієнта, а також прямого коригування програми вправ відповідно до клінічних показань і професійних рекомендацій. Це надасть можливість тіснішої взаємодії між цифровою платформою та реальним медичним середовищем, що значно підвищить ефективність і достовірність запропонованих рішень.

ВИСНОВОК ДО РОЗДІЛУ 4

У четвертому розділі наведено опис особливостей та результатів тестування ймовірних сценаріїв роботи з реалізованою вебплатформою. Тестування охоплює типові та граничні ситуації, що дозволило комплексно оцінити стабільність та надійність системи. Для ілюстрації функціоналу наведено демонстраційні скріншоти роботи, які відображають етапи виконання ключових сценаріїв згідно з алгоритмом взаємодії користувача та системи.

Висновок аналізу отриманих результатів засвідчує коректність роботи вебплатформи та відповідність реалізованого функціоналу всім попередньо визначеним технічним завданням. Усі елементи системи працюють злагоджено, без критичних помилок чи збоїв у роботі.

Додатково розглянуто можливі шляхи для подальшого розвитку системи, потенційні напрями масштабування, зокрема трансформація платформи у прогресивний вебдодаток, удосконалення інтерфейсу під мобільні платформи та інтеграція медичних установ і працівників у реабілітаційний процес користувача.

ВИСНОВКИ

Дипломний проєкт присвячений розробці вебплатформи для психофізичної реабілітації населення та реалізує низку технічних завдань, серед яких основними є надання користувачам можливості реєстрації та авторизації, вести трекінг власного стану вручну та голосом, отримувати статистичний аналіз показників впродовж певного періоду, а також отримувати щоденні реабілітаційні завдання та відеоінструкції до них відповідно до актуального стану здоров'я.

Перший розділ присвячений огляду актуальності тематики реабілітації в нинішніх реаліях суспільства, потребі в розробці програмного забезпечення у цій сфері, переваги розробки вебплатформ у порівнянні із іншими видами інформаційних продуктів та аналізу наявних рішень на світовому та вітчизняному ринках.

У другому розділі проведено огляд та аналіз переваг і недоліків популярних способів організації вебплатформ, а також використовуваних інструментів для розробки клієнтської та серверної частин застосунку. На основі дослідження та кінцевої мети проєкту обрано тип архітектури платформи та набір інструментів для його реалізації. Також обґрунтовано вибір застосування додаткових сервісів для забезпечення коректної реалізації поставлених технічних завдань.

Третій розділ містить детальний опис реалізації можливостей вебплатформи за допомогою обраних засобів та сервісів.

Четвертий розділ присвячений демонстрації роботи реалізованої системи. Також наведені результати автоматизованого тестування, що підтверджують коректність роботи платформи як у типових, так і у граничних сценаріях взаємодії користувача із застосунком. Додатково наведено

рекомендації до майбутнього розвитку платформи, враховуючи її потенціал до масштабування.

Як результат мета дипломного проєкту досягнута: розроблена система повністю задовольняє вимогам сучаних вебплатформ, надаючи безпечний та широкий набір можливостей згідно своєї тематики та визначеного напередодні технічного завдання. Окрім цього розроблений проєкт реалізує у собі новітні можливості голосового вводу даних, що вирізняє його серед аналогів та створює додаткову цінність для користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основні визначення фізично-реабілітаційної освіти / Бєлікова Н. // Збірник «Актуальні проблеми навчання та виховання людей з особливими потребами» – 6(8) 2009 – https://ap.uu.edu.ua/article/177
2. Особливості психічного здоров'я громадян України в умовах епідемічної небезпеки та реформування системи охорони здоров`я на сучасному етапі. / Табачніков С, Осуховська О, Хаустова О, Марценковська І, Марков А, Салдень В, Товалович Т. // PMGP [інтернет]. – 2021 – https://uk.e-medjournal.com/index.php/psp/article/view/345
3. Актуальні питання розвитку фізичної реабілітації в Україні. / Гордійчук С., Олефір Л., Поплавська С., Самунь Н., & Шатило В. // (2024). Україна. Здоров'я нації, (2) – Р.132–140. – https://doi.org/10.32782/2077-6594/2024.2/22
4. Rehab Guru Офіційний сайт компанії Rehab Guru – 2024 – https://www.rehabguru.com/
5. Manage your low back pain with Selfback Офіційний сайт компанії Selfback – 2024 – https://www.selfback.dk/en/home
6. Вільний step – онлайн платформа комплексної підтримки. Офіційний сайт компанії Вільний step – 2025 – https://vilnyystep.com.ua/
7. Ти як?: Програма ментального здоров'я Офіційний сайт компанії Ти як?: – 2025 – https://howareu.com/
8. Effect of Internet-Based Rehabilitation Programs on Improvement of Pain and Physical Function in Patients with Knee Osteoarthritis: Systematic Review and Meta-analysis of Randomized Controlled Trials. / Xie S. H., Wang Q., Wang L. Q., Wang L., Song K. P., & He C. Q. // Journal of medical Internet research, 23(1) – 2021 – https://doi.org/10.2196/21542

9. Effectiveness of psychological interventions delivered by physiotherapists in the management of neck pain: a systematic review with meta-analysis. / Farrell Scott F.a,*; Edmunds Devonb; Fletcher Johnb; Martine Harryb; Mohamed Hashemb; Liimatainen Jennaa; Sterling Michelea. // PAIN Reports 8(3) – May/June 2023 – https://journals.lww.com/painrpts/fulltext/2023/06000/effectiveness_of_psychological_interventions.8.aspx
10. Chronic pain sent Jabez into a spiral of despair. Behaviour therapy brought her back / Natasha May // The Guardian News & Media – 2025 – https://www.theguardian.com/australia-news/2025/may/07/chronic-pain-sent-jabez-into-a-spiral-of-despair-behaviour-therapy-brought-her-back
11. Effectiveness of Gamification in Knee Replacement Rehabilitation: Protocol for a Randomized Controlled Trial With a Qualitative Approach / Aartolahti E, Janhunen M, Katajapuu N, Paloneva J, Pamilo K, Oksanen A, Keemu H, Karvonen M, Luimula M, Korpelainen R, Jämsä T, Mäkelä K, Heinonen A // JMIR Res Protoc – 2022 – https://www.researchprotocols.org/2022/11/e38434/
12. Frequently Monitoring Progress Toward Goals Increases Chance of Success / Jim Sliwa // American Psychological Association – 2015 – https://www.apa.org/news/press/releases/2015/10/progress-goals
13. A Comprehensive Guide to Modern Frontend Architecture Patterns / Jean claude adjanohoun – 2023 – https://medium.com/@johnadjanohoun/a-comprehensive-guide-to-modern-frontend-architecture-patterns-eb39debbd503
14. Feature-Sliced Design: The Ideal Frontend Architecture / Davit Gasparyan // Medium – 2024 – https://medium.com/@dtgasparyan/feature-sliced-design-the-ideal-frontend-architecture-84d701ad44ba

15. Roadmap to Backend Programming Master: Architectural Patterns / Lagu // Medium – 2024 – https://medium.com/@hanxuyang0826/roadmap-to-backend-programming-master-architectural-patterns-c763c9194414
16. Home • Angular Офіційний сайт – 2025 – https://angular.dev/
17. React.dev: website. Офіційний сайт – 2025 – https://react.dev/
18. Vue.js - The Progressive JavaScript Framework Офіційний сайт – 2025 – https://vuejs.org/
19. Django: The web framework for perfectionists with deadlines Офіційний сайт – 2025 – https://www.djangoproject.com/
20. Node.js – Run JavaScript Everywhere Офіційний сайт – 2025 – https://nodejs.org/
21. Spring Boot Офіційний сайт – 2025 – https://spring.io/projects/spring-boot
22. Web Speech API / Mozilla // MDN Web Docs – 2025 – https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
23. What is NLP? - Natural Language Processing Explained Офіційний сайт AWS – 2025 – https://aws.amazon.com/what-is/nlp/#:~:text=Natural%20language%20processing%20(NLP)%20combines,with%20computers%20and%20software%20tools.
24. MySQL Офіційний сайт – 2025 – https://www.mysql.com/
25. Auth0: Secure access for everyone. But not just anyone. Офіційний сайт – 2025 – https://auth0.com/
26. JSON Web Tokens Офіційний сайт Okta Inc. – 2025 – https://auth0.com/docs/secure/tokens/json-web-tokens
27. Google Cloud: Cloud Computing Services Офіційний сайт – 2025 – https://cloud.google.com

Додаток 1

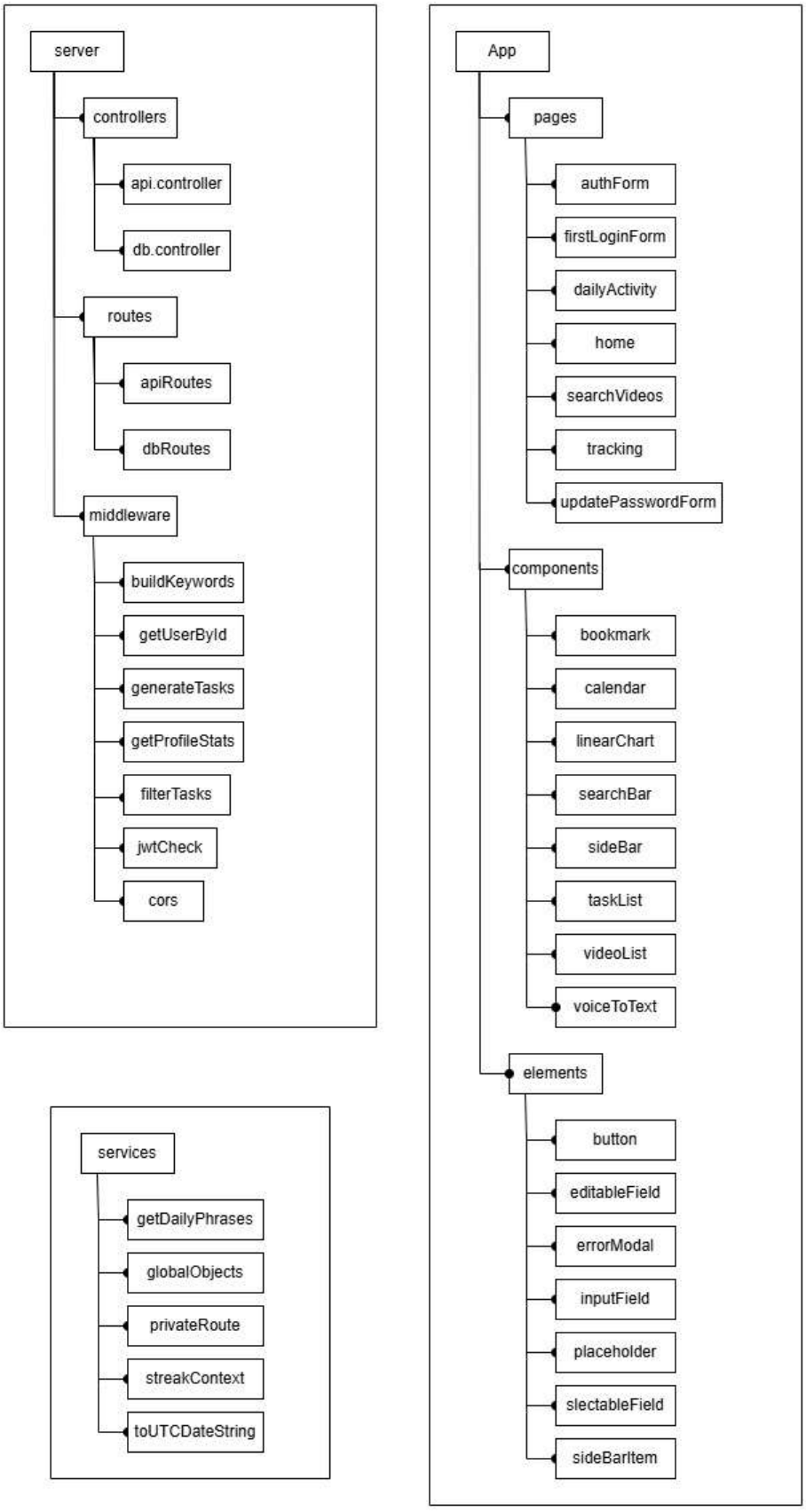
Веб-платформа для психо-фізичної реабілітації

Структурна схема

ІАЛЦ.467200.001 Д1

Аркушів 1

Київ 2025



					ІАЛЦ.467200.001 Д1	
		№ докум.	Підпис	Дата		
Розробив		Мартинюк М. П.			**Веб-платформа для психо-фізичної реабілітації** Структурна схема	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Сергієнко А. М.				КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-13
Н. Контр.		Пономаренко А. М.				
Затвердив						

Додаток 2

Веб-платформа для психо-фізичної реабілітації

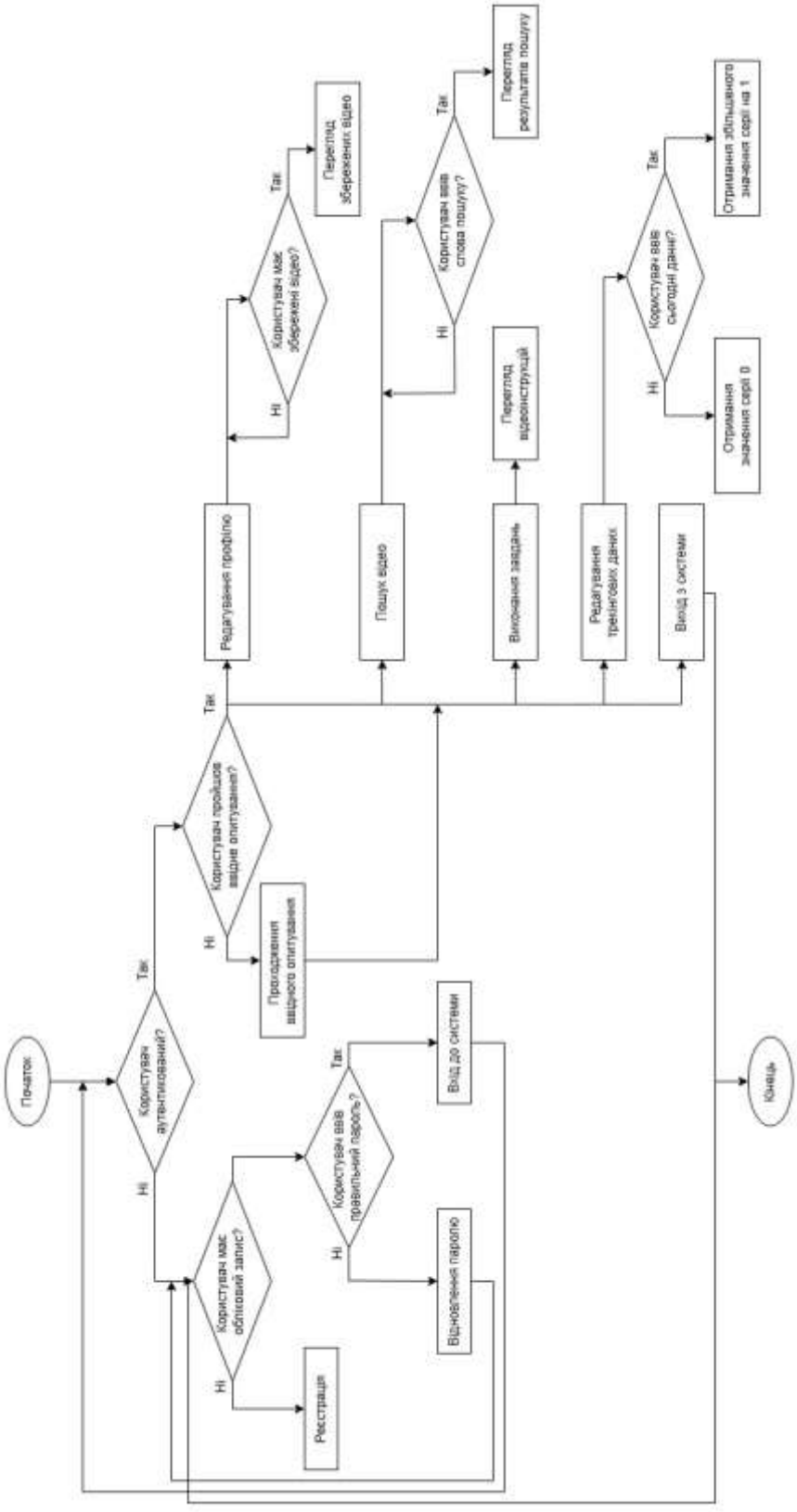
Алгоритм взаємодії користувача з системою

(принципова схема)

ІАЛЦ.467200.002 Д2

Аркушів 1

Київ 2025



					ІАЛЦ.467200.002 Д2	
		№ докум.	Підпис	Дата		
Розробив		Мартинюк М. П.			**Веб-платформа для психо-фізичної реабілітації** Алгоритм взаємодії користувача з системою (принципова схема)	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Сергієнко А. М.				КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-13
Н. Контр.		Пономаренко А. М.				
Затвердив						

Додаток 3

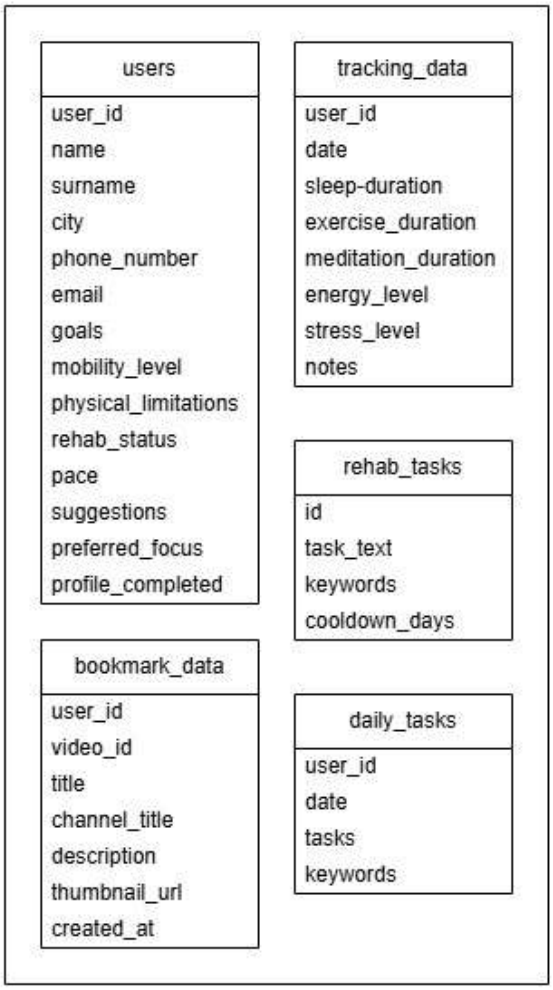
Веб-платформа для психо-фізичної реабілітації

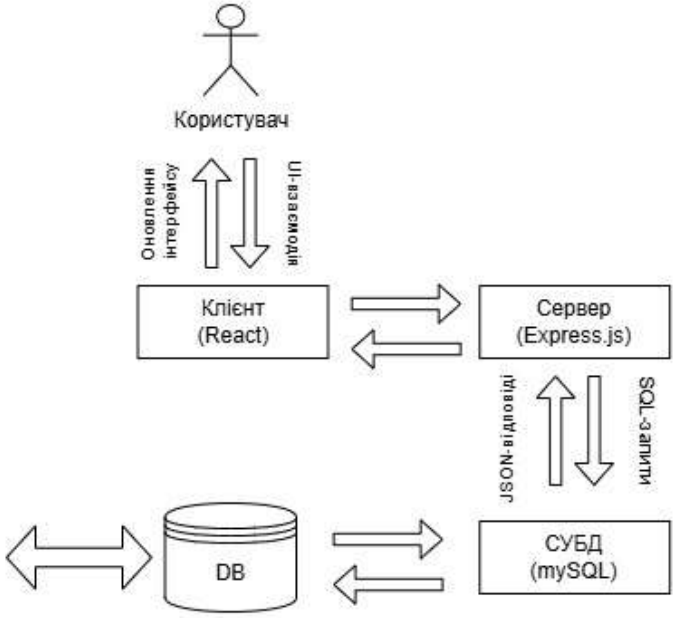
Функціональна схема

ІАЛЦ.467200.003 Д3

Аркушів 1

Київ 2025





					ІАЛЦ.467200.003 Д3	
		№ докум.	Підпис	Дата		
Розробив		Мартинюк М. П.			**Веб-платформа для психо-фізичної реабілітації** Функціональна схема	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Сергієнко А. М.				КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-13
Н. Контр.		Пономаренко А. М.				
Затвердив						

Додаток 4

Веб-платформа для психо-фізичної реабілітації

Текст програмного коду

ІАЛЦ.467200.004 Д4

Аркушів 23

Київ 2025

App.js

```
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { AuthForm, UpdatePasswordForm, Home, Tracking, SearchVideos, DailyActivity,
FirstLoginForm } from './pages/index.js';
import PrivateRoute from '../services/privateRoute.js';

const App = () => {
  return (
    <div className='app-container'>
      <Router>
        <Routes>
          <Route path='/' element={<AuthForm />} />
          <Route path='/password/new' element={<UpdatePasswordForm />} />
          <Route path='/firstLoginForm' element={<FirstLoginForm />} />
          <Route
            path='/home'
            element={
              <PrivateRoute>
                <Home />{' '}
              </PrivateRoute>
            }
          />
          <Route
            path='/tracking'
            element={
              <PrivateRoute>
                <Tracking />{' '}
              </PrivateRoute>
            }
          />
          <Route
            path='/search-tutorials'
            element={
              <PrivateRoute>
                <SearchVideos />{' '}
              </PrivateRoute>
            }
          />
          <Route
            path='/daily-activity'
            element={
              <PrivateRoute>
                <DailyActivity />{' '}
              </PrivateRoute>
            }
          />
        </Routes>
      </Router>
    </div>
  );
};

export default App;
```

authForm.js

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { InputFieldLogin, InputFieldPassword } from '../elements/inputField.js';
import { Button } from '../elements/index.js';
import { Logo } from '../../external/index.js';
import '../styles/styleForms.css';

const AuthForm = () => {
  const [email, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [repeatPassword, setRepeatPassword] = useState('');
  const [passwordMatch, setPasswordMatch] = useState(true);
  const [isLogin, setIsLogin] = useState(true);
  const [isInfoCorrect, setisInfoCorrect] = useState(true);
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    console.log({ email, password, isLogin });
    try {
      const response = await fetch(`${process.env.REACT_APP_API_URL}/api/login`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email, password }),
        credentials: 'include',
      });

      const data = await response.json();

      if (!response.ok) {
        setisInfoCorrect(!isInfoCorrect);
        throw new Error(data.error || 'Authentication failed');
      }

      console.log('Auth Successful:', data);

      const protectedRes = await fetch(`${process.env.REACT_APP_API_URL}/api/protected`, {
        credentials: 'include',
      });

      const protectedData = await protectedRes.json();
      console.log(protectedData);
      if (!protectedRes.ok) {
        throw new Error('Failed to verify login');
      }
      // console.log(protectedData.profile_completed);
      if (!protectedData.profile_completed) {
        navigate('/firstLoginForm');
      } else {
        navigate('/home');
      }
    } catch (error) {
      console.error('Authentication failed:', error);
    }
  };

  const handleAccCreation = async (e) => {
    e.preventDefault();
    if (password !== repeatPassword) {
      setPasswordMatch(false);
      return;
    }
```

```
    setPasswordMatch(true);
    if (setPasswordMatch) {
      try {
        const endpoint = `${process.env.REACT_APP_API_URL}/api/signup`;

        const response = await fetch(endpoint, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ email, password }),
        });

        const responseJSON = await response.json();

        if (!response.ok) {
          throw new Error(responseJSON.error || 'User Creation failed failed');
        }

        console.log('USER:', response.status);
        console.log(responseJSON);
        setIsLogin(!isLogin);
      } catch (error) {
        console.error('User Creation failed:', error);
      }
    }
  };

  return (
    <div className='auth-container'>
      <div>
        <img src={Logo} alt='Logo' className='vector-image-forms' />
      </div>
      <div className='text-welcome-login'>{isLogin ? 'Welcome' : 'Create an account'}</div>
      <div className='text-login-to-your-acc'>{isLogin ? 'Log in to your account to
continue' : ''}</div>
      {isLogin ? (
        <form onSubmit={handleSubmit}>
          <InputFieldLogin type='text' placeholder='Email address' value={email}
onChange={setUsername} />
          <InputFieldPassword type='password' placeholder='Password' value={password}
onChange={setPassword} />
          {isInfoCorrect ? <div></div> : <div id='text-wrong'>Wrong email or password</div>}
          <Button className='forms' btnID='forms' type='submit' text='Login'
logo={undefined} id='text-button' />
        </form>
      ) : (
        <form onSubmit={handleAccCreation}>
          <InputFieldLogin type='text' placeholder='Email' value={email}
onChange={setUsername} />
          <InputFieldPassword type='password' placeholder='Password' value={password}
onChange={setPassword} />
          <InputFieldPassword type='repeatPassword' placeholder='Repeat password'
value={repeatPassword} onChange={setRepeatPassword} />
          <div id='text-wrong'>{passwordMatch ? '' : 'Passwords do not match!'}</div>
          <Button className='forms' btnID='forms' type='submit' text='Sign Up'
logo={undefined} id='text-button' />
        </form>
      )}
      {isLogin ? (
        <>
          <div className='text-login-to-your-acc'>
            <button className='link-button' onClick={() => navigate('/password/new')}>
              Forgot password
            </button>
          </div>
          <div className='text-login-to-your-acc'>
```

```
            Don't have an account?{' '}
            <button className='link-button' onClick={() => setIsLogin(!isLogin)}>
              Create one
            </button>
          </div>
        </>
      ) : (
        <>
          {' '}
          <div className='text-login-to-your-acc'>
            Already have an account?{' '}
            <button className='link-button' onClick={() => setIsLogin(!isLogin)}>
              Log in
            </button>
          </div>
        </>
      )}
    </div>
  );
};

export default AuthForm;
```

firstLoginForm.js

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { Button, SelectableField, EditableField } from '../elements/index.js';
import { Logo } from '../../external/index.js';
import '../styles/styleQuestionnaire.css';
import {
  editableFieldsConfigForm,
  mobilityOptions,
  rehablitationOptions,
  goalsOptions,
  limitationsOptions,
  focusOptions,
  paceOptions,
  suggestionsOptions,
} from '../../services/globalObjects.js';

const sections = new Array(9);

const FirstLoginForm = () => {
  const [page, setPage] = useState(0);
  const [formData, setFormData] = useState({
    name: '',
    surname: '',
    city: '',
    phone_number: '',
    goals: [],
    mobility_level: '',
    rehab_status: '',
    physical_limitations: '',
    preferred_focus: '',
    pace: '',
    suggestions: [],
    profile_completed: false,
  });

  const navigate = useNavigate();

  const handleNext = () => {
    if (page < sections.length - 1) setPage(page + 1);
  };
```

```
  const handleBack = () => {
    if (page > 0) setPage(page - 1);
  };

  const handleChange = (field, value) => {
    setFormData((prev) => ({
      ...prev,
      [field]: value,
    }));
  };

  const handleCheckboxChange = (field, value) => {
    setFormData((prev) => {
      const values = new Set(prev[field]);
      values.has(value) ? values.delete(value) : values.add(value);
      return { ...prev, [field]: Array.from(values) };
    });
  };

  const handleFinish = async () => {
    const response = await fetch(`${process.env.REACT_APP_API_URL}/db/update-user-profile`,
{
      method: 'POST',
      credentials: 'include',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        fields: { ...formData, profile_completed: true },
      }),
    });

    if (response.ok) {
      navigate('/home');
    }
  };

  return (
    <div className='auth-container'>
      <div>
        <img src={Logo} alt='Logo' className='vector-image-forms' />
      </div>
      {page === 0 && (
        <div className='text-welcome'>
          Welcome!
          <br />
          Please fill out the form to personalize the experience
        </div>
      )}
      {page === 1 && (
        <div>
          <h2>Enter your contact information</h2>
          <div className='contact-container'>
            {editableFieldsConfigForm.map(({ key, label, fieldStyle, labelStyle }) => (
              <EditableField
                key={key}
                plhID='1'
                label={label}
                value={formData[key]}
                onChange={(value) => handleChange(key, value)}
                isEditing={true}
                style={fieldStyle}
                labelStyle={labelStyle}
              />
```

```
          ))}
        </div>
      </div>
    )}

    {page === 2 && (
      <div>
        <h2>What are your top goals?</h2>
        <div className='checkbox-grid'>
          {goalsOptions.map((goal) => (
            <label key={goal} className='flex items-center space-x-2'>
              <input type='checkbox' checked={formData.goals.includes(goal)} onChange={()
=> handleCheckboxChange('goals', goal)} className='form-checkbox' />
              <span>{goal}</span>
            </label>
          ))}
        </div>
      </div>
    )}

    {page === 3 && (
      <div>
        <h2 className='text-xl font-bold mb-4'>What is your current physical state?</h2>
        <div className='checkbox-grid'>
          <SelectableField
            restrictEditing={false}
            label='Mobility level'
            value={formData.mobility_level}
            onChange={(value) => handleChange('mobility_level', value)}
            isEditing={true}
            style={{ position: 'relative', left: '15%' }}
            labelStyle={{ position: 'relative', left: '15%' }}
            options={mobilityOptions}
          />
          <SelectableField
            restrictEditing={false}
            label='Rehab status'
            value={formData.rehab_status}
            onChange={(value) => handleChange('rehab_status', value)}
            isEditing={true}
            style={{ position: 'relative', left: '15%' }}
            labelStyle={{ position: 'relative', left: '15%' }}
            options={rehablitationOptions}
          />
        </div>
      </div>
    )}

    {page === 4 && (
      <div>
        <h2 className='text-xl font-bold mb-4'>Do you have any injuried body parts?</h2>
        <div className='checkbox-grid'>
          {limitationsOptions.map((bodyPart) => (
            <label key={bodyPart} className='flex items-center space-x-2'>
              <input
                type='checkbox'
                checked={formData.physical_limitations.includes(bodyPart)}
                onChange={() => handleCheckboxChange('physical_limitations', bodyPart)}
                className='form-checkbox'
              />
              <span>{bodyPart}</span>
            </label>
          ))}
        </div>
      </div>
```

```
      )}

      {page === 5 && (
        <div>
          <h2 className='text-xl font-bold mb-4'>What is your preferred reabilitation
focus?</h2>
          <div className='option-grid'>
            {focusOptions.map((p) => (
              <label key={p} className='label'>
                <input
                  type='radio'
                  name='preferred_focus'
                  value={p}
                  checked={formData.preferred_focus === p}
                  onChange={(e) => handleChange('preferred_focus', e.target.value)}
                />
                {p}
              </label>
            ))}
          </div>
        </div>
      )}

      {page === 6 && (
        <div>
          <h2 className='text-xl font-bold mb-4'>What is your preferred pace?</h2>
          <div className='option-grid'>
            {paceOptions.map((p) => (
              <label key={p} className='label'>
                <input type='radio' name='pace' value={p} checked={formData.pace === p}
onChange={(e) => handleChange('pace', e.target.value)} className='mr-2' />
                {p}
              </label>
            ))}
          </div>
        </div>
      )}

      {page === 7 && (
        <div>
          <h2 className='text-xl font-bold mb-4'>What is your preferred suggestions
type?</h2>
          <div className='option-grid'>
            {suggestionsOptions.map((opt) => (
              <label key={opt} className='label'>
                <input type='checkbox' checked={formData.suggestions.includes(opt)}
onChange={() => handleCheckboxChange('suggestions', opt)} className='mr-2' />
                {opt}
              </label>
            ))}
          </div>
        </div>
      )}

      {page === 8 && (
        <div className='text-center'>
          <h2 className='text-2xl font-bold mb-4'>Thank you for answering!</h2>
          <p>Your preferences have been saved. You'll now receive suggestions tailored to
your needs.</p>
        </div>
      )}

      <div className='button-container'>
        {page > 1 && page < sections.length - 1 && <Button className='forms' btnID='forms-2-
back' text={'Back'} onClick={handleBack} />}
```

```
        {page > 0 && page < sections.length - 2 && <Button className='forms' btnID='forms-2-
skip' text={'Skip'} onClick={handleNext} />}
        {page < sections.length - 1 && <Button className='forms' btnID='forms-2'
text={'Next'} onClick={handleNext} />}
        {page === sections.length - 1 && <Button className='forms' btnID='forms-2'
text={'Finish'} onClick={handleFinish} />}
      </div>
    </div>
  );
};

export default FirstLoginForm;
```

home.js

```
import { useState, useEffect } from 'react';
import axios from 'axios';
import { Edit, DefaultImage } from '../../external/index.js';
import { Button, EditableField, Placeholder } from '../elements/index.js';
import { Sidebar, BookmarkedVideos } from '../components/index.js';
import { editableFieldsConfig, menuItems } from '../../services/globalObjects.js';
import '../styles/styleProfile.css';

const Home = () => {
  const [userData, setUserData] = useState(null);
  const [isEditing, setIsEditing] = useState(false);
  const [profileImage, setProfileImage] = useState(null);

  const [fieldValues, setFieldValues] = useState({
    name: '',
    surname: '',
    city: '',
    phone_number: '',
  });

  const handleFieldChange = (fieldName, value) => {
    setFieldValues((prev) => ({
      ...prev,
      [fieldName]: value,
    }));
  };

  const handleSave = async () => {
    try {
      const response = await fetch(`${process.env.REACT_APP_API_URL}/db/update-user-
profile`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include',
        body: JSON.stringify({
          fields: fieldValues,
        }),
      });

      if (!response.ok) {
        throw new Error('Failed to update user fields');
      }

      const data = await response.json();
      console.log('Update successful:', data);

      setIsEditing(false);
    } catch (error) {
      console.error('Save failed', error);
    }
```

```
  };

  const handleImageUpload = (e) => {
    const file = e.target.files[0];
    if (file) {
      const imageUrl = URL.createObjectURL(file);
      setProfileImage(imageUrl);
    }
  };

  useEffect(() => {
    const fetchUserData = async () => {
      try {
        const response = await axios.get(`${process.env.REACT_APP_API_URL}/db/user-
data/profile`, {
          withCredentials: true,
        });

        const data = response.data;
        setUserData(data);
        setFieldValues({
          name: data.name || '',
          surname: data.surname || '',
          city: data.city || '',
          phone_number: data.phone_number || '',
        });
      } catch (error) {
        console.error('Error fetching user data:', error);
      }
    };

    fetchUserData();
  }, []);

  if (!userData) return <div>Loading user data...</div>;

  return (
    <div className='profile-container'>
      <div className='sidebar'>
        <Sidebar activeId={menuItems[0].id}></Sidebar>
      </div>
      <div className='header'>
        <div className='text-header'>Home</div>
      </div>
      <div className='personal-info'>
        <div className='profile-picture-container'>
          <label htmlFor='profile-upload'>
            <img src={profileImage || DefaultImage} alt='Profile' className='profile-
picture' />
          </label>
          <input type='file' id='profile-upload' accept='image/*'
onChange={handleImageUpload} style={{ display: 'none' }} />
        </div>
        <Button className='edit' text='Edit' btnID='1' logo={Edit} onClick={() =>
setIsEditing(true)}></Button>
        {isEditing && <Button className='edit' btnID='1' text='Save' logo={Edit}
onClick={handleSave}></Button>}
        {editableFieldsConfig.map(({ key, label, fieldStyle, labelStyle }) => (
          <EditableField
            key={key}
            plhID='1'
            label={label}
            value={fieldValues[key]}
            onChange={(value) => handleFieldChange(key, value)}
            isEditing={isEditing}
```

```
            style={fieldStyle}
            labelStyle={labelStyle}
          />
        ))}
        <Placeholder
          label='Email'
          value={userData.email}
          isEditing={isEditing}
          style={{ position: 'absolute', top: '8.9vh', left: '1218px' }}
          labelStyle={{ position: 'absolute', top: '8.7vh', left: '1150px' }}
        />
      </div>
      <div className='recently-block'>
        <h2>BOOKMARKS:</h2>
        <BookmarkedVideos />
      </div>
    </div>
  );
};

export default Home;
```

tracking.js

```
import { useState, useEffect } from 'react';
import { useStreak } from '../../services/streakContext.js';
import axios from 'axios';
import { Sidebar, CustomCalendar, CustomStatsChart, VoiceToText } from
'../components/index.js';
import { Button, EditableField, SelectableField, Placeholder, ErrorModal } from
'../elements/index.js';
import { Edit, NotesIcon, StatisticsIcon, ConditionIcon, BulbIcon } from
'../../external/index.js';
import {
  menuItems,
  fieldOptions,
  periodOptions,
  periodMap,
  editableFieldsConfigTracking,
  selectableFieldsConfigTracking,
  placeholderAvgConfigTracking,
} from '../../services/globalObjects.js';
import { toUTCDateString, getYesterdayUTCString } from '../../services/toUTCDateString.js';
import '../styles/styleTracking.css';

const Tracking = () => {
  const [selectedDate, setSelectedDate] = useState(new Date());
  const [selectedPeriod, setSelectedPeriod] = useState('week');
  const [userData, setUserData] = useState(null);
  const [showModalError, setShowModalError] = useState(false);
  const [showModalVoiceInput, setShowModalVoiceInput] = useState(false);
  const [restrictEditing, setRestrictEditing] = useState(false);
  const [errorMessage, setErrorMessage] = useState('');

  const [fieldValues, setFieldValues] = useState({
    sleep_duration: '',
    exercise_duration: '',
    meditation_duration: '',
    energy_level: undefined,
    stress_level: undefined,
    notes: '',
  });

  const [averages, setAverages] = useState({
    avg_sleep: 'loading...',
```

```
    avg_exercise: 'loading...',
    avg_meditation: 'loading...',
    most_common_energy: 'loading...',
    most_common_stress: 'loading...',
  });

  const [isEditing, setIsEditing] = useState(false);
  const { setStreak } = useStreak();

  const now = toUTCDateString(new Date());
  const yesterday = getYesterdayUTCString(new Date());
  console.log(now, yesterday);
  const formattedDate = toUTCDateString(selectedDate);

  const handleFieldChange = (fieldName, value) => {
    setFieldValues((prev) => ({
      ...prev,
      [fieldName]: value,
    }));
  };

  const calculateStreak = async () => {
    try {
      const doesEntryExist = await fetch(`${process.env.REACT_APP_API_URL}/db/user-
data/tracking/${yesterday}`, {
        method: 'GET',
        credentials: 'include',
      });

      const result = await doesEntryExist.json();
      console.log(result.found);
      let streak = 0;

      if (result.found) {
        streak += 1;
        setStreak(streak);
      } else {
        setStreak(0);
      }
      console.log('streak', streak);
    } catch (e) {}
  };

  const handleSave = async () => {
    try {
      const response = await fetch(`${process.env.REACT_APP_API_URL}/db/update-user-
tracking-data`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include',
        body: JSON.stringify({
          fields: {
            sleep_duration: fieldValues.sleep_duration || 0,
            exercise_duration: fieldValues.exercise_duration || 0,
            meditation_duration: fieldValues.meditation_duration || 0,
            energy_level: fieldValues.energy_level || 'medium',
            stress_level: fieldValues.stress_level || 'medium',
            notes: fieldValues.notes || '',
            date: formattedDate,
          },
        }),
      });

      if (!response.ok) {
        setErrorMessage('Incorrect input! Please enter valid values (0-24)');
```

```
      setShowModalError(true);
      throw new Error('Failed to update user fields');
    }

    const data = await response.json();
    console.log('Update successful:', data);

    setIsEditing(false);
  } catch (error) {
    console.error('Save failed', error);
  }
};

useEffect(() => {
  // console.log(formattedDate);
  // console.log(now);
  // console.log(yesterday);

  if (formattedDate > now) setRestrictEditing(true);
  else setRestrictEditing(false);

  const fetchUserData = async () => {
    try {
      const dataResponse = await fetch(`${process.env.REACT_APP_API_URL}/db/user-
data/tracking/${formattedDate}`, {
        method: 'GET',
        credentials: 'include',
      });

      const result = await dataResponse.json();
      console.log('data:', result);

      if (!dataResponse.ok || !result.data) {
        console.log('No data found for this date');
        setFieldValues({
          sleep_duration: '',
          exercise_duration: '',
          meditation_duration: '',
          energy_level: undefined,
          stress_level: undefined,
          notes: '',
        });
        return;
      }

      setUserData(result.data);
      setFieldValues({
        sleep_duration: result.data.sleep_duration,
        exercise_duration: result.data.exercise_duration,
        meditation_duration: result.data.meditation_duration,
        energy_level: result.data.energy_level,
        stress_level: result.data.stress_level,
        notes: result.data.notes || '',
      });
    } catch (error) {
      console.error('Error fetching user data:', error);
    }
  };

  const fetchAverages = async () => {
    try {
      console.log('fetching avg');
      const period = periodMap[selectedPeriod] || 7;
```

```
        const response = await axios.get(`${process.env.REACT_APP_API_URL}/db/user-
data/statistic_analyse/avg`, {
          params: {
            endDate: now,
            period,
          },
          withCredentials: true,
        });

        let data = await response.data;
        console.log(data);
        setAverages(data);
      } catch (error) {
        console.error('Error fetching averages:', error);
      }
    };

    const runBoth = async () => {
      try {
        await fetchUserData();
        await fetchAverages();
        await calculateStreak();
      } catch (err) {
        console.error('Failed to run fetch operations:', err);
      }
    };

    runBoth();
  }, [formattedDate, selectedPeriod]);

  return (
    <div className='tracking-container'>
      <div className='sidebar'>
        <Sidebar activeId={menuItems[1].id}></Sidebar>
      </div>
      <div className='header'>
        <div className='text-header'>Tracking</div>
      </div>
      <div className='info-block'>
        <div className='daily-info'>
          <div id='info-header'>
            <div className='condition-header'>
              <span id='text-naming'>
                <img src={ConditionIcon} alt='Logo' className='vector-image-tracking' />
                <span>Condition</span>
              </span>
            </div>
            <div className='notes-header'>
              <span id='text-naming'>
                <img src={NotesIcon} alt='Logo' className='vector-image-tracking' />
                <span>Notes</span>
              </span>
            </div>
          </div>
          <div className='condition-block-tracking'>
            {editableFieldsConfigTracking.map(({ key, label, logo, fieldStyle, labelStyle })
=> (
              <EditableField
                key={key}
                restrictEditing={restrictEditing}
                plhID='1'
                label={label}
                logo={logo}
                value={fieldValues[key]}
                onChange={(value) => handleFieldChange(key, value)}
```

```
              isEditing={isEditing}
              style={fieldStyle}
              labelStyle={labelStyle}
            />
          ))}
          {selectableFieldsConfigTracking.map(({ key, label, logo, fieldStyle, labelStyle
}) => (
            <SelectableField
              key={key}
              restrictEditing={restrictEditing}
              label={label}
              logo={logo}
              value={fieldValues[key]}
              onChange={(value) => handleFieldChange(key, value)}
              isEditing={isEditing}
              style={fieldStyle}
              labelStyle={labelStyle}
              options={fieldOptions}
            />
          ))}
        </div>
        <div className='notes-block'>
          <div className='notes-text'>
            <EditableField
              restrictEditing={restrictEditing}
              plhID='2'
              value={fieldValues.notes}
              onChange={(value) => handleFieldChange('notes', value)}
              isEditing={isEditing}
            />
          </div>
          <Button className='edit' btnID='2' text='Edit' logo={Edit} onClick={() =>
setIsEditing(true)}></Button>
          {isEditing && !restrictEditing && <Button className='edit' btnID='2' text='Save'
logo={Edit} onClick={handleSave}></Button>}
          <Button className='edit' btnID='4' text='Use Voice Input' logo={BulbIcon}
onClick={() => setShowModalVoiceInput(true)}></Button>
        </div>
        {showModalVoiceInput && <VoiceToText
setShowModalVoiceInput={setShowModalVoiceInput} date={formattedDate}
setFieldValues={setFieldValues} />}
        {showModalError && <ErrorModal message={errorMessage} onClose={() =>
setShowModalError(false)} />}
      </div>
      <div className='statistic-info'>
        <div id='info-header'>
          <span id='text-naming'>
            <img src={StatisticsIcon} alt='Logo' className='vector-image-tracking' />
            <span>Statistics</span>
            <select className='placeholder-1' id='statistic-period-select'
value={selectedPeriod} onChange={(e) => setSelectedPeriod(e.target.value)}>
              {periodOptions.map((opt) => (
                <option key={opt} value={opt}>
                  {opt}
                </option>
              ))}
            </select>
          </span>
        </div>
        <div className='condition-block-tracking'>
          {placeholderAvgConfigTracking.map(({ key, label, logo, fieldStyle, labelStyle })
=> (
            <Placeholder
              key={key}
              label={label}
```

```
                logo={logo}
                value={averages[key]}
                onChange={(value) => handleFieldChange(key, value)}
                style={fieldStyle}
                labelStyle={labelStyle}
              />
            ))}
          </div>
          <CustomStatsChart selectedPeriod={selectedPeriod} />
        </div>
      </div>
      <div className='calendar-block'>
        <CustomCalendar setIsEditing={setIsEditing} setSelectedDate={setSelectedDate} />
      </div>
    </div>
  );
};

export default Tracking;
```

updatePassword.js

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { Button, InputFieldLogin } from '../elements/index.js';
import { Logo } from '../../external/index.js';
import '../styles/styleForms.css';

const UpdatePasswordForm = () => {
  const [email, setUsername] = useState('');
  const navigate = useNavigate();

  const updatePassword = async (e) => {
    const endpoint = `${process.env.REACT_APP_API_URL}/api/updatePassword`;

    try {
      const response = await fetch(endpoint, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email }),
      });

      const data = await response.json();

      if (!response.ok) {
        throw new Error(data.error || 'Authentication failed');
      }

      console.log('Password Change Successful:', data);
    } catch (error) {
      console.error('Authentication failed:', error);
    }
  };

  return (
    <div className='auth-container'>
      <div>
        <img src={Logo} alt='Logo' className='vector-image-forms' />
      </div>
      <div className='text-welcome'>Password Recover</div>
      <div className='text-login-to-your-acc'>Enter the email to send the password recover
letter</div>
      <form onSubmit={updatePassword}>
```

```
        <InputFieldLogin id='input-text' type='text' placeholder='Email address'
value={email} onChange={setUsername} />
        <Button className='forms' btnID='forms' type='submit' text='Send Email'
logo={undefined} id='text-button' />
      </form>
      <div className='text-login-to-your-acc'>
        <button className='link-button' onClick={() => navigate('/')}>
          Return to Log in page
        </button>
      </div>
    </div>
  );
};

export default UpdatePasswordForm;
```

searchVideos.js

```
import { useState } from 'react';
import { Sidebar, VideoList, SearchBar } from '../components/index.js';
import { menuItems } from '../../services/globalObjects.js';
import '../styles/styleVideos.css';

const SearchVideos = () => {
  const [keywords, setKeywords] = useState('');

  return (
    <div className='profile-container'>
      <div className='sidebar'>
        <Sidebar activeId={menuItems[2].id}></Sidebar>
      </div>
      <div className='header'>
        <div className='text-header'>Search Relatable Videos</div>
      </div>
      <div className='page-container'>
        <SearchBar value={keywords} onChange={setKeywords} placeholder='Enter search
query...' />
        {keywords.trim().length > 0 && <VideoList keywords={[keywords.trim()]} id={'3'}
maxResults={undefined} />}
      </div>
    </div>
  );
};

export default SearchVideos;
```

server.js

```
require('dotenv').config();
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const cookieParser = require('cookie-parser');
const corsMiddleware = require('./middleware/cors.js');

const apiRoutes = require('./routes/apiRoutes.js');
const dbRoutes = require('./routes/dbRoutes.js');

const PORT = process.env.PORT_SERVER || 3001;

const app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

```
app.use(cookieParser());
app.use(corsMiddleware);
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.use(express.static(path.join(__dirname, '../frontend/')));

app.use('/api', apiRoutes);
app.use('/db', dbRoutes);

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, '../frontend/'));
});

app.get('*', (req, res) => {
  res.sendFile(path.join(__dirname, '../frontend/'));
});

app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

apiRoutes.js

```
import express from 'express';
import jwtCheck from '../middleware/jwtCheck.js';
import { userLogin, userSignUp, userLogOut, protectedRoot, updatePassword } from
'../controller/api.controller.js';

const router = express.Router();

router
  .get('/protected', jwtCheck, async (req, res) => protectedRoot(req, res))
  .post('/login', async (req, res) => userLogin(req, res))
  .post('/signup', async (req, res) => userSignUp(req, res))
  .post('/updatePassword', async (req, res) => updatePassword(req, res))
  .post('/logout', (req, res) => userLogOut(req, res));

export default router;
```

api.controller.js

```
import axios from 'axios';
import db from '../db.js';
import { getUserById } from '../middleware/index.js';

const domain = process.env.DOMAIN;
const clientID = process.env.CLIENT_ID;
const clientSecret = process.env.CLIENT_SECRET;
const audience = process.env.AUDIENCE;

async function userLogin(req, res) {
  const { email, password } = req.body;
  try {
    const response = await axios.post(`https://${domain}/oauth/token`, {
      grant_type: 'password',
      username: email,
      password,
      audience: audience,
      client_id: clientID,
      client_secret: clientSecret,
      scope: 'openid profile email offline_access',
    });

    console.log('Auth0 Response:', response.data);
    const { access_token, refresh_token } = response.data;
```

```
    res.cookie('access_token', access_token, {
      httpOnly: true,
      sameSite: 'lax',
      secure: false,
      maxAge: 15 * 60 * 1000, // 15 minutes
    });

    res.cookie('refresh_token', refresh_token, {
      httpOnly: true,
      sameSite: 'lax',
      secure: false,
      maxAge: 30 * 24 * 60 * 60 * 1000, // 30 days
    });

    res.json({ access_token });
  } catch (error) {
    console.error('Auth0 Error:', error.response ? error.response.data : error.message);
    res.status(400).json({ error: 'Invalid credentials or Auth0 issue' });
  }
}

async function userSignUp(req, res) {
  const { email, password } = req.body;
  console.log('Received signup request:', { email, password });

  const data = {
    client_id: clientID,
    email,
    password,
    connection: 'Username-Password-Authentication',
  };

  try {
    const response = await axios.post(`https://${domain}/dbconnections/signup`, data, {
      headers: { 'Content-Type': 'application/json' },
    });

    const userEmail = email;
    const userID = response.data._id;

    const sql_profile = `
        INSERT IGNORE INTO users (user_id, email)
        VALUES (?, ?)
      `;
    const values = [userID, userEmail];
    await db.query(sql_profile, values);
    res.json({ user: response.data });
  } catch (error) {
    console.error('Signup Error:', error.response?.data || error.message);
    res.status(400).json({
      error: 'Signup failed',
      details: error.response?.data || error.message,
    });
  }
}

async function userLogOut(req, res) {
  res.clearCookie('token', {
    path: '/',
    secure: true,
    sameSite: 'lax',
  });

  res.status(200).json({ message: 'Logged out and cookie cleared' })}
```

```
async function tokenRefresh(req, res) {
  const refreshToken = req.cookies.refresh_token;
  console.log('refresh token: ', refreshToken);

  if (!refreshToken) {
    return res.status(401).json({ error: 'Missing refresh token' });
  }

  try {
    const response = await axios.post(`https://${domain}/oauth/token`, {
      grant_type: 'refresh_token',
      client_id: clientID,
      client_secret: clientSecret,
      refresh_token: refreshToken,
    });

    const { access_token } = response.data;
    res.json({ access_token });
  } catch (error) {
    console.error('Refresh token error:', error.response?.data || error.message);
    res.status(403).json({ error: 'Invalid refresh token' });
  }
}

async function protectedRoot(req, res) {
  let { profile_completed } = await getUserById(req.userId);
  console.log('from protected', profile_completed);
  res.json({
    message: 'This is a protected route',
    userId: req.userId,
    profile_completed,
  });
}

async function updatePassword(req, res) {
  const { email } = req.body;

  try {
    const response = await axios.post(`https://${domain}/dbconnections/change_password`, {
      client_id: clientID,
      email,
      connection: 'Username-Password-Authentication',
    });

    res.json({ message: 'Password reset email sent' });
  } catch (error) {
    console.error('Password reset error:', error.response?.data || error.message);
    res.status(400).json({ error: 'Failed to send password reset email' });
  }
}

export { userLogin, userSignUp, userLogOut, tokenRefresh, protectedRoot, updatePassword };
```

dbRoutes.js

```
import express from 'express';
import { jwtCheck } from '../middleware/index.js';
import {
  getUserProfile,
  updateUserProfile,
  getTrackingDatabyDate,
  getStatisticForChart,
  getStatisticAvg,
  updateTrackingData,
  postBookmarks,
```

```
  deleteBookmarks,
  getBookmarks,
  getTasks,
  updateTasks,
} from '../controller/db.controller.js';

const router = express.Router();

router
  .get('/user-data/profile', jwtCheck, async (req, res) => getUserProfile(req, res))
  .get('/user-data/tracking/:date', jwtCheck, async (req, res) => getTrackingDatabyDate(req,
res))
  .get('/user-data/statistic_analyse/chart', jwtCheck, async (req, res) =>
getStatisticForChart(req, res))
  .get('/user-data/statistic_analyse/avg', jwtCheck, async (req, res) =>
getStatisticAvg(req, res))
  .get('/user-data/bookmarks', jwtCheck, async (req, res) => getBookmarks(req, res))
  .get('/user-tasks', jwtCheck, async (req, res) => getTasks(req, res))
  .post('/update-user-tracking-data', jwtCheck, async (req, res) => updateTrackingData(req,
res))
  .post('/update-user-profile', jwtCheck, async (req, res) => updateUserProfile(req, res))
  .post('/user-data/bookmarks', jwtCheck, async (req, res) => postBookmarks(req, res))
  .post('/user-tasks/update', jwtCheck, async (req, res) => updateTasks(req, res))
  .delete('/user-data/bookmarks', jwtCheck, async (req, res) => deleteBookmarks(req, res));
export default router;
```

voiceToText.js

```
import { useState, useEffect, useRef } from 'react';
import axios from 'axios';
import { wordToNumber, fieldPatterns } from '../../services/globalObjects.js';
import '../styles/styleVoiceToTextModal.css';

const VoiceToText = ({ setShowModalVoiceInput, date, setFieldValues }) => {
  const [transcript, setTranscript] = useState('');
  const [isListening, setIsListening] = useState(false);
  const recognitionRef = useRef(null);

  useEffect(() => {
    const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;

    if (!SpeechRecognition) {
      alert('Web Speech API is not supported in this browser.');
      return;
    }
    if (recognitionRef.current) {
      recognitionRef.current.stop();
    }

    const recognition = new SpeechRecognition();
    recognition.lang = 'en-US';
    recognition.interimResults = true;
    recognition.continuous = true;

    recognition.onresult = (event) => {
      let finalTranscript = '';
      for (let i = event.resultIndex; i < event.results.length; i++) {
        finalTranscript += event.results[i][0].transcript;
      }
      setTranscript(finalTranscript);
    };

    recognition.onend = () => {
      if (isListening) recognition.start();
    };
```

```
  recognitionRef.current = recognition;
}, [isListening]);

const handleStart = () => {
  if (recognitionRef.current) {
    recognitionRef.current.start();
    setIsListening(true);
  }};
const handleStop = () => {
  if (recognitionRef.current) {
    recognitionRef.current.stop();
    setIsListening(false);
    const extractedData = handleFieldsExtraction(transcript);
    setFieldValues((prev) => ({ ...prev, ...extractedData }));
    axios.post(
      `${process.env.REACT_APP_API_URL}/db/update-user-tracking-data`,
      {
        fields: {
          sleep_duration: extractedData.sleep_duration || 0,
          exercise_duration: extractedData.exercise_duration || 0,
          meditation_duration: extractedData.meditation_duration || 0,
          energy_level: extractedData.energy_level || 'medium',
          stress_level: extractedData.stress_level || 'medium',
          notes: extractedData.notes || '',
          date,
     },
      },
      {
        headers: {
          'Content-Type': 'application/json',
        },
        withCredentials: true,
      }
    );

    setShowModalVoiceInput(false);
  }
};

const handleClose = () => {
  handleStop();
  setShowModalVoiceInput(false);
};

const handleFieldsExtraction = (transcript) => {
  const result = {};

  for (const [field, pattern] of Object.entries(fieldPatterns)) {
    const matches = [...transcript.matchAll(pattern)];

    matches.forEach((match) => {
      let value = match[1]?.toLowerCase();
      if (wordToNumber[value]) {
        value = wordToNumber[value];
      } else if (!isNaN(value)) {
        value = parseInt(value, 10);
      }

      result[field] = value;
      console.log(`${field}:`, value);
    });
  }

  return result;
```

```
  };

  return (
    <div className='modal-backdrop'>
      <div className='modal '>
        <button id='btn-close' onClick={handleClose} aria-label='Close'>
          ×
        </button>
        <h2>Voice to Text Transcription</h2>
        <h3>Please input the needed data using key words, e.g.
sleep/exercise/meditation/energy/stress/notes</h3>
        <button className={`button-modal ${isListening ? 'button-disabled' : 'button-
active'}`} onClick={handleStart} disabled={isListening}>
          <span id='button-modal-text'>Start</span>
        </button>
        <button className={`button-modal ${!isListening ? 'button-disabled' : 'button-
active'}`} onClick={handleStop} disabled={!isListening}>
          <span id='button-modal-text'>Stop</span>
        </button>
        <p>{transcript || 'Speak to see the transcription here...'}</p>
      </div>
    </div>
  );
};

export default VoiceToText;
```

generateTasks.js

```
import db from '../db.js';
import { getUserProfileWithStats } from './getProfileStats.js';
import { buildKeywords } from './buildKeywords.js';
import { getRecentTasks, filterEligibleTasks } from './filterTasks.js';

async function generateTasks(userId, now, yesterday) {
  const [existing] = await db.query('SELECT tasks, keywords FROM daily_tasks WHERE user_id =
? AND date = ?', [userId, now]);
  console.log(existing[0]);
  if (existing.length) return existing[0];

  const nowDate = new Date(now);

  const [pastTasks] = await db.query(
    `SELECT * FROM daily_tasks
     WHERE user_id = ? AND STR_TO_DATE(date, '%Y-%m-%d') >= DATE_SUB(STR_TO_DATE(?, '%Y-%m-
%d'), INTERVAL 4 DAY)`,
    [userId, now]
  );
  const userProfile = await getUserProfileWithStats(userId, now);
  const keywords = buildKeywords(userProfile);
  const [rawTasks] = await db.query(
    `SELECT task_text, cooldown_days, keywords FROM rehab_tasks
     WHERE JSON_OVERLAPS(keywords, CAST(? AS JSON))`,
    [JSON.stringify(keywords)]
  );
  const recentTaskList = getRecentTasks(pastTasks);
  const { eligibleTasks, keywords: taskKeywords } = filterEligibleTasks(rawTasks,
recentTaskList, nowDate);

  const uncompletedFromYesterday = pastTasks
    .filter((d) => d.date === yesterday)
    .flatMap((d) => {
      const tasks = typeof d.tasks === 'string' ? JSON.parse(d.tasks) : d.tasks;
      return tasks.filter((t) => t.completed === false);
    })
```

```
    .map((t) => t.text);

  const uncompletedTasks = eligibleTasks.filter((t) =>
uncompletedFromYesterday.includes(t.task_text));
  const remainingTasks = eligibleTasks.filter((t) =>
!uncompletedFromYesterday.includes(t.task_text));
  const additionalTasks = remainingTasks.slice(0, 3 - uncompletedTasks.length);
  const finalTasks = [...uncompletedTasks, ...additionalTasks].slice(0, 3);
  const generatedTasks = finalTasks.map((t) => ({ text: t.task_text, completed: false }));

  const allKeywords = Array.from(new Set([...keywords, ...taskKeywords]));

  await db.query('INSERT INTO daily_tasks (user_id, date, tasks, keywords) VALUES (?, ?, ?,
?)', [
    userId,
    now,
    JSON.stringify(generatedTasks),
    JSON.stringify(allKeywords),
  ]);
  return { tasks: generatedTasks, keywords: allKeywords };
}
export default generateTasks;
```