

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

“ ” _____ 2025 р.

Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем»
спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для моніторингу пульсу та надання рекомендацій
на основі штучного інтелекту

Виконав студент IV курсу, групи _____ ІП-15
(шифр групи)

_____ Мешков Андрій Ігорович
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник асистент, Храмченко М. С.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Консультант асистент, Шулькевич Т. В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Сперкач М. О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Засвідчую, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії
Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 121 Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

Едуард ЖАРІКОВ

(підпис)

(ім'я прізвище)

“ ” 2025 р.

ЗАВДАННЯ на дипломний проєкт студенту

Мешкову Андрію Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для моніторингу пульсу та надання
рекомендацій на основі штучного інтелекту

керівник проєкту Храмченко Микола Сергійович, асистент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Передпроєктне обстеження предметної області: постановка завдання
дипломного проєктування, аналіз предметної області, аналіз існуючих рішень,
аналіз відомих програмних продуктів, аналіз відомих алгоритмічних та
технічних рішень, аналіз та моделювання бізнес-процесів.

2) Розроблення вимог до програмного забезпечення: варіанти використання
програмного забезпечення, розроблення функціональних вимог, розроблення
нефункціональних вимог, аналіз економічних показників програмного
забезпечення, постановка завдання на розробку програмного забезпечення.

3) Конструювання та розроблення програмного забезпечення: архітектура
програмного забезпечення, обґрунтування вибору засобів розробки та
архітектурні рішення, конструювання програмного забезпечення, опис
алгоритму вимірювання пульсу, опис структури даних, опис бази даних, аналіз
безпеки даних.

4) Аналіз якості та тестування програмного забезпечення: аналіз якості
програмного забезпечення, опис процесів тестування, опис контрольного
прикладу.

5) Розгортання та супровід програмного забезпечення: розгортання програмного
забезпечення, супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна компонентів програмного забезпечення

3) Схема бази даних

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв'язання задачі	04.04.2025	
3	Постановка та формалізація задачі	11.04.2025	
4	Розробка інформаційного забезпечення	14.04.2025	
5	Алгоритмізація задачі	22.04.2025	
6	Обґрунтування вибору використаних технічних засобів	23.04.2025	
7	Розробка програмного забезпечення	10.05.2025	
8	Налагодження програми	15.05.2025	
9	Виконання графічних документів	18.05.2025	
10	Оформлення пояснювальної записки	26.05.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент

(підпис)

Андрій МЄШКОВ

(ініціали, прізвище)

Керівник

(підпис)

Микола ХРАМЧЕНКО

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п'яти розділів, містить 39 таблиць, 36 рисунків та 13 джерел – загалом 87 сторінок.

Дипломний проєкт присвячений розробці вебзастосунку для моніторингу пульсу та надання персоналізованих рекомендацій на основі штучного інтелекту, що спрямований на підвищення якості відстеження стану здоров'я та доступності інноваційних технологій для широкого кола користувачів.

Мета роботи – створення вебзастосунку, який забезпечує точний аналіз даних пульсу в реальному часі, виявлення аномалій за допомогою штучного інтелекту та надання персоналізованих рекомендацій для підтримки здоров'я.

У першому розділі проведено аналіз предметної області, досліджено сучасний стан технологій моніторингу здоров'я, проаналізовано існуючі програмні продукти та алгоритмічні рішення, а також змодельовано бізнес-процеси й сформульовано завдання проєкту.

Другий розділ присвячений розробці вимог до програмного забезпечення, включаючи аналіз варіантів використання, формулювання функціональних і нефункціональних вимог, а також оцінку економічних показників.

У третьому розділі спроектовано архітектуру вебзастосунку, обґрунтовано вибір засобів розробки, описано алгоритми вимірювання пульсу, структуру даних і базу даних, а також проаналізовано аспекти безпеки даних.

Четвертий розділ містить аналіз якості програмного забезпечення, опис процесів тестування та контрольного прикладу.

П'ятий розділ присвячений розгортанню та супроводу програмного забезпечення, включаючи процедури впровадження та підтримки.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, МОНІТОРИНГ ПУЛЬСУ, ШТУЧНИЙ ІНТЕЛЕКТ, БАЗА ДАНИХ, ФОТОПЛЕТИЗМОГРАФІЯ, ПЕРСОНАЛІЗОВАНІ РЕКОМЕНДАЦІЇ, ВЕБІНТЕРФЕЙС.

ABSTRACT

The explanatory note of the diploma project consists of five sections, containing 39 tables, 36 figures, and 13 sources – totaling 87 pages.

The diploma project is dedicated to the development of a web application for pulse monitoring and providing personalized recommendations based on artificial intelligence, aimed at improving the quality of health monitoring and the accessibility of innovative technologies for a wide range of users.

The purpose of the project is to create a web application that ensures accurate real-time pulse data analysis, anomaly detection using artificial intelligence, and personalized recommendations to support health.

In the first section, an analysis of the subject area was conducted, the current state of health monitoring technologies was studied, existing software products and algorithmic solutions were analyzed, business processes were modeled, and the project tasks were formulated.

The second section is devoted to the development of software requirements, including analysis of usage scenarios, formulation of functional and non-functional requirements, and evaluation of economic indicators.

The third section designs the architecture of the web application, justifies the choice of development tools, describes the pulse measurement algorithms, data structure, database, and analyzes data security aspects.

The fourth section includes an analysis of software quality, a description of testing processes, and a control example.

The fifth section focuses on the deployment and maintenance of the software, including implementation and support procedures.

KEYWORDS: WEB APPLICATION, PULSE MONITORING, ARTIFICIAL INTELLIGENCE, DATABASE, PHOTOPLETHYSMOGRAPHY, PERSONALIZED RECOMMENDATIONS, WEB INTERFACE

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ПУЛЬСУ ТА НАДАННЯ
РЕКОМЕНДАЦІЙ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ**

Технічне завдання

КПІ.ПІ-1522.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Микола ХРАМЧЕНКО

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Андрій МЄШКОВ

Київ – 2025

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ.....	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	6
4.1	Вимоги до функціональних характеристик	6
4.1.1	Користувацького інтерфейсу	6
4.1.2	Для користувача:	7
4.1.3	Додаткові вимоги:	7
4.2	Вимоги до надійності.....	7
4.3	Умови експлуатації	7
4.3.1	Вид обслуговування.....	7
4.3.2	Обслуговуючий персонал	7
4.4	Вимоги до складу і параметрів технічних засобів	8
4.5	Вимоги до інформаційної та програмної сумісності	8
4.5.1	Вимоги до вхідних даних	8
4.5.2	Вимоги до вихідних даних.....	8
4.5.3	Вимоги до мови розробки	8
4.5.4	Вимоги до середовища розробки	9
4.5.5	Вимоги до представленню вихідних кодів.....	9
4.6	Вимоги до маркування та пакування.....	9
4.7	Вимоги до транспортування та зберігання	9
4.8	Спеціальні вимоги	9
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	10
5.1	Попередній склад програмної документації.....	10
5.2	Спеціальні вимоги до програмної документації	10
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ.....	11
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	12

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: ВЕБЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ПУЛЬСУ ТА НАДАННЯ РЕКОМЕНДАЦІЙ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ.

Галузь застосування: Охорона здоров'я, фітнес і персоналізоване медичне консультування

Наведене технічне завдання поширюється на розробку вебзастосунку для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту My Heart Rate Monitor - Pulse, котра використовується для відстеження частоти серцевих скорочень у реальному часі, аналізу даних пульсу та надання персоналізованих рекомендацій щодо здоров'я і фізичної активності. Застосунок призначений для використання в охороні здоров'я, спортивних програмах і повсякденному житті, орієнтований на користувачів, які піклуються про своє здоров'я, зокрема осіб із серцево-судинними захворюваннями, спортсменів, фітнес-ентузіастів, а також медичних працівників для моніторингу стану пацієнтів.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебзастосунку для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизованого моніторингу частоти серцевих скорочень у реальному часі, аналізу даних пульсу за допомогою штучного інтелекту та надання персоналізованих рекомендацій щодо здоров'я і фізичної активності на основі отриманих даних. Вебзастосунок забезпечує зручний доступ до цих функцій через інтуїтивно зрозумілий вебінтерфейс, що дозволяє використовувати його на різних пристроях, включаючи мобільні.

Метою розробки є підвищення якості моніторингу стану здоров'я користувачів шляхом створення надійного, точного та доступного інструменту, який використовує штучний інтелект для аналізу даних пульсу, виявлення аномалій і надання рекомендацій. Якість досягається через забезпечення точності аналізу, створення зручного інтерфейсу та впровадження системи, що відповідає потребам широкого кола користувачів, включаючи осіб, які піклуються про здоров'я, спортсменів і медичних працівників.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

- Відображення екрану авторизації (авторизація з Google, авторизація з Apple, реєстрація за електронною поштою, вхід за електронною поштою) – рис. 4.1а;
- Відображення екрану вимірювання пульсу – рис. 4.1б;
- Відображення екрану введення персональних даних (вага, ріст, стать) – рис. 4.1в;
- Відображення екрану результатів вимірювання (пульс, варіабельність серцевого ритму та стрес) – рис. 4.1г;
- Відображення екрану інформаційної панелі (останнє вимірювання, графіки, створення нового вимірювання) – рис. 4.1г;
- Відображення екрану історії вимірювань – рис. 4.1д;

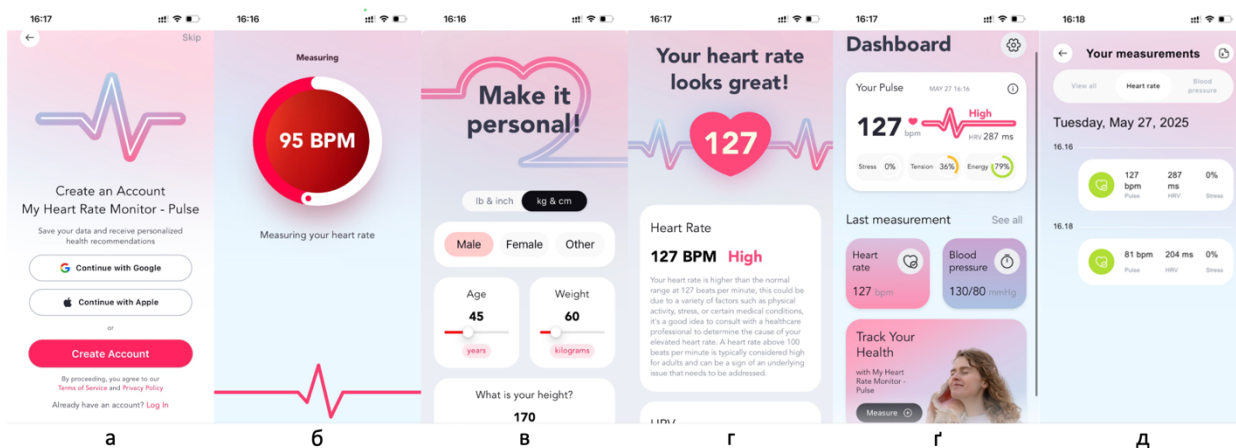


Рисунок 4.1 – UX дизайн основних екранів проекту

4.1.2 Для користувача:

- Можливість створення користувача користувача за допомогою електронної пошти;
- Можливість авторизації за допомогою Google/Apple;
- Можливість входу в систему за допомогою електронної пошти;
- Можливість вимірювання пульсу;
- Можливість введення персональних даних: ріст, вага та стать.
- Можливість отримання результатів вимірювання з аналізом та рекомендаціями;
- Можливість отримання історії вимірювань;
- Можливість отримання візуалізації історії вимірювань;
- Можливість обрати платну підписку для отримання кращих рекомендацій з використанням сильнішої ШІ моделі;

4.1.3 Додаткові вимоги:

Не висуваються.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних та системі авторизації.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення призначене для роботи на IBM-сумісних персональних комп'ютерах, а також на мобільних пристроях із підтримкою веббраузерів. Для забезпечення оптимальної роботи вебзастосунку, рекомендується використовувати пристрої з вбудованим світлодіодним спалахом (фонариком) для точнішого вимірювання пульсу за допомогою фотоплетизмографії.

Мінімальна конфігурація технічних засобів:

- веббраузер: Google Chrome (94 або новіші), Mozilla Firefox (93 або новіші), Apple Safari (15.0 або новіші) або Microsoft Edge (94 або новіші);
- підключення до мережі Інтернет зі швидкістю від 20 Мбіт/с;
- наявність вебкамери або сенсора для вимірювання пульсу (для мобільних пристроїв — камера з підтримкою фонарика);

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення вебзастосунку повинно працювати під управлінням сучасних операційних систем, включаючи сімейства Windows (Windows 10, Windows 11), macOS (10.15 Ventura або новіші), а також мобільні операційні системи Android (9.0 або новіші) та iOS (13.0 або новіші).

4.5.1 Вимоги до вхідних даних

Вимоги до вхідних даних не висуваються.

4.5.2 Вимоги до вихідних даних

Вимоги до вихідних даних не висуваються.

4.5.3 Вимоги до мови розробки

Розробку виконати на мовах програмування Javascript та Typescript.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища Visual Studio Code та IntelliJ IDEA.

4.5.5 Вимоги до представленню вихідних кодів

Вимоги до представленню вихідних кодів не висуваються.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна компонентів програмного забезпечення;
- схема бази даних;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проєкту	21.02	
2.	Розробка технічного завдання	15.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проєкту	20.05	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

**Пояснювальна записка
до дипломного проєкту**

на тему: **Вебзастосунок для моніторингу пульсу та надання рекомендацій
на основі штучного інтелекту**

КПІ.ПІ-1522.045440.02.81

Київ – 2025

ЗМІСТ

ВСТУП.....	5
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	7
1.1 Постановка завдання дипломного проєктування	7
1.2 Аналіз предметної області.....	8
1.3 Аналіз існуючих рішень	10
1.3.1 Аналіз відомих програмних продуктів	11
1.3.2 Аналіз відомих алгоритмічних та технічних рішень.....	17
1.4 Аналіз та моделювання бізнес-процесів	20
Висновки до розділу.....	23
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	25
2.1 Варіанти використання програмного забезпечення.....	25
2.2 Розроблення функціональних вимог	32
2.3 Розроблення нефункціональних вимог	36
2.4 Аналіз системних вимог	37
2.5 Аналіз економічних показників програмного забезпечення.....	37
2.6 Постановка завдання на розробку програмного забезпечення	42
Висновки до розділу.....	43
3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	46
3.1 Архітектура програмного забезпечення.....	46
3.2 Архітектурні рішення та обґрунтування вибору засобів розробки	50
3.3 Конструювання програмного забезпечення	52
3.3.1 Опис алгоритму вимірювання пульсу.....	52
3.3.2 Опис структури даних	54
3.3.3 Опис бази даних	54
3.4 Аналіз безпеки даних	57
Висновки до розділу.....	58
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	60

4.1	Аналіз якості ПЗ	60
4.2	Опис процесів тестування	62
4.3	Опис контрольного прикладу.....	69
	Висновки до розділу	76
5	РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	78
5.1	Розгортання програмного забезпечення	78
5.2	Супровід програмного забезпечення.....	80
	Висновки до розділу	81
	ВИСНОВКИ	83
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	85
	ДОДАТОК А ЗВІТ ПОДІБНОСТІ	87

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
API	– Application programming interface, прикладний програмний Інтерфейс.
SDK	– Software development kit.
IT	– Інформаційні технології.
ER	– Entity-Relation diagram.
ШІ	– Штучний інтелект.
PPG	– Photoplethysmography – фотоплетизмографія.
БД	– База даних.

ВСТУП

Актуальність теми полягає в тому, що в сучасному світі зростає потреба в доступних і зручних інструментах для моніторингу здоров'я, зокрема частоти серцевих скорочень, через поширення проблем, пов'язаних зі стресом, серцево-судинними захворюваннями та низькою фізичною активністю. Відсутність інтелектуальних рішень для раннього виявлення аномалій пульсу та надання персоналізованих рекомендацій ускладнює профілактику й підтримку здоров'я. Розробка вебзастосунку на основі штучного інтелекту для моніторингу пульсу є актуальною, оскільки забезпечує інноваційний підхід до збереження здоров'я, відповідає потребам широкої аудиторії (людей, які піклуються про здоров'я, спортсменів, медичних і фітнес-установ) та сприяє інтеграції сучасних технологій у повсякденне життя.

Сьогодні провідні тенденції у сфері моніторингу здоров'я зосереджені на інтеграції штучного інтелекту, обробці даних у реальному часі та створенні адаптивних інтерфейсів для широкого кола користувачів. Такі компанії, як Fitbit, Apple та Garmin, активно розробляють пристрої та програмне забезпечення для моніторингу фізіологічних показників, однак більшість рішень мають обмеження, пов'язані з високою вартістю, складністю інтеграції чи недостатньою персоналізацією. Наукові установи, зокрема MIT Media Lab та Stanford University, досліджують застосування машинного навчання для аналізу біометричних даних, що підтверджує перспективність цього напрямку. В Україні подібні розробки перебувають на початковому етапі, що створює можливості для створення доступних рішень із підтримкою локальних потреб, зокрема україномовного інтерфейсу.

У рамках дипломного проєкту буде створено вебзастосунок для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту. Застосунок включатиме модуль збору й обробки даних пульсу в реальному часі, алгоритм виявлення аномалій із використанням ШІ, систему персоналізованих рекомендацій та зручний вебінтерфейс.

Мета проєкту – підвищити якість і надійність моніторингу стану здоров'я шляхом створення вебзастосунку, який забезпечує точний аналіз даних пульсу та доступність для широкого кола користувачів.

Призначення застосунку – надати інструмент для повсякденного використання людьми, які піклуються про своє здоров'я, спортсменами, а також медичними установами чи фітнес-центрами.

Сфери застосування включають індивідуальний моніторинг здоров'я, підтримку спортивних тренувань, інтеграцію в медичні системи для попереднього аналізу стану пацієнтів, а також дослідження у сфері біометричних даних. Розробка цього вебзастосунку сприятиме підвищенню доступності інноваційних технологій для підтримки здоров'я та популяризації профілактичних заходів.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проектування

Дипломне проектування присвячено розробці вебзастосунку для моніторингу пульсу та надання персоналізованих рекомендацій на основі штучного інтелекту. У рамках дипломного проектування необхідно виконати наступний список завдань:

- аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки;
- аналіз існуючих рішень;
- аналіз та моделювання бізнес-процесів;
- розроблення функціональних, нефункціональних та системних вимог;
- аналіз економічних показників програмного забезпечення;
- постановка завдання на розробку програмного забезпечення;
- розроблення архітектури програмного забезпечення;
- розроблення архітектурних рішень та обґрунтування вибору засобів розробки;
- конструювання та розроблення програмного забезпечення;
- аналіз безпеки даних програмного забезпечення;
- аналіз якості та тестування програмного забезпечення;
- розгортання та супровід програмного забезпечення;
- створення супроводжувальної документації;

Виконання зазначених завдань дозволить створити функціональний, безпечний і зручний вебзастосунок, який відповідатиме потребам користувачів і сучасним технологічним стандартам.

1.2 Аналіз предметної області

Тема цього дипломного проєкту охоплює моніторинг серцевого ритму та надання персоналізованих рекомендацій на основі штучного інтелекту з використанням вебтехнологій. Серцевий ритм – це ключовий фізіологічний показник, який відображає частоту серцевих скорочень і може вказувати на стан здоров'я, рівень стресу, фізичну активність або наявність таких відхилень, як аритмія. Моніторинг серцевого ритму має широке застосування в охороні здоров'я, спорті, фітнесі та повсякденному житті для профілактики та раннього виявлення проблем зі здоров'ям.

Сучасні розробки в цій галузі зосереджені на інтеграції сенсорних технологій, обробці даних у режимі реального часу та застосуванні штучного інтелекту для біометричного аналізу даних. Зокрема, використання камери смартфона зі спалахом для зчитування серцевого ритму за допомогою фотоплетизмографії (FPG) є перспективним напрямком, оскільки забезпечує доступність та зручність для широкого кола користувачів. FIG базується на аналізі змін поглинання світла шкірою, спричинених кровотоком, і дозволяє отримувати дані про серцевий ритм без спеціалізованого обладнання [1]. Провідні дослідницькі установи, такі як Медіа-лабораторія MIT та Стенфордський університет, активно досліджують методи аналізу сигналів FIG з використанням машинного навчання для підвищення точності вимірювань [2].

Основні напрями розробок включають:

- розробку мобільних і вебзастосунків для моніторингу фізіологічних показників;
- впровадження алгоритмів штучного інтелекту для виявлення аномалій і прогнозування стану здоров'я;
- створення адаптивних інтерфейсів для забезпечення зручності використання;
- інтеграцію з хмарними платформами для обробки та зберігання даних;

Сучасні інформаційні технології відіграють дедалі більшу роль у розробці рішень для моніторингу серцевого ритму. Fitbit, Apple Health, Cardiogram та Instant Heart Rate – це деякі з програм, які використовують апаратні функції, такі як датчики та камери смартфонів, та інтегрують їх із програмними алгоритмами для збору, обробки та аналізу даних. Інтеграція цієї теми в ІТ включає кілька етапів: отримання даних через камеру смартфона зі спалахом для захоплення сигналів PPG від пальця, розміщеного на камері, фільтрація даних для видалення шуму та посилення пульсових хвиль, аналіз даних із застосуванням алгоритмів для розрахунку частоти серцевих скорочень, виявлення відхилень (тахікардії або брадикардії) та прогнозування стану, а також візуалізація результатів у вигляді графіків та надання персоналізованих рекомендацій щодо активності або відпочинку.

Предметна область має три основні бізнес-процеси: формування профілю користувача для зберігання історії вимірювань; вимірювання пульсу, де користувач прикладає палець до камери смартфона, а програма зчитує сигнали PPG, при цьому обробка даних відбувається на стороні сервера для пошуку аномалій; аналіз даних та формування рекомендацій, де відповідно до отриманих результатів користувач отримує рекомендації, такі як зменшення фізичної активності або відвідування лікаря.

Незважаючи на успіх технологій моніторингу серцевого ритму, існуючі рішення мають певні недоліки: висока вартість додатків або необхідність використання спеціального обладнання обмежують їх доступність; поради іноді не враховують конкретні характеристики користувача через стандартизацію; точність вимірювання пульсу за допомогою камери смартфона може бути знижена через перешкоди в сигналах або неправильне використання; а низький рівень захисту біометричних даних несе ризик витоку. Для вирішення цих проблем пропонується наступне: розробка безкоштовних або недорогих вебдодатків, доступних для всіх, які функціонують без використання додаткових пристроїв; удосконалення алгоритмів машинного навчання для підвищення точності обробки даних;

розробка адаптивних алгоритмів на основі віку користувача, фізичної активності та історії вимірювань; забезпечення безпеки шляхом шифрування даних та дотримання політик захисту інформації.

Цей дипломний проект передбачає створення вебдодатку, сумісного зі смартфонами, який використовує спалах камери для зчитування пульсових сигналів та застосовує штучний інтелект для аналізу даних і генерації персоналізованих рекомендацій. Основні міркування щодо дизайну додатку такі:

- Доступність: Оскільки додаток працюватиме у веббраузері, він не потребуватиме жодних додаткових чи спеціальних пристроїв.
- Точність та персоналізація: Для аналізу сигналів PPG та генерації рекомендацій будуть використовуватися алгоритми машинного навчання.
- Безпека: Дані будуть зашифровані, а також буде використовуватися безпечне сховище інформації.
- Зручність: Буде розроблено адаптивний та інтуїтивно зрозумілий інтерфейс, щоб забезпечити комфортне використання додатку на смартфоні.

Я вважаю, що створення інструменту для моніторингу здоров'я, який є доступним, функціональним та безпечним, щоб відповідати сучасним вимогам та бажанням користувачів, може бути виконано, використовуючи цей базовий дизайн та набір функцій.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації вебзастосунків для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

У предметній області моніторингу пульсу за допомогою камери смартфона існує низка програмних продуктів, які частково або повністю реалізують функціонал, передбачений технічним завданням дипломного проєкту. Для аналізу обрано два популярні застосунки: *Instant Heart Rate* та *Heart Rate Monitor - Pulse App*, які використовують фотоплетизмографію (PPG) для зчитування пульсу через камеру смартфона. Нижче наведено їхній опис та порівняльна таблиця з розробленим вебзастосунком – *My Heart Rate Monitor - Pulse App*.

Опис програмних продуктів:

– *Instant Heart Rate* – один із найпопулярніших мобільних додатків для вимірювання серцевого ритму. Щоб визначити свій серцевий ритм, користувач прикладає палець до камери смартфона зі спалахом. Потім додаток аналізує зміни кольору шкіри, що виникають через кровотік, щоб точно визначити частоту серцевих скорочень користувача. *Instant Heart Rate* надає графік серцевого ритму в режимі реального часу, зберігає історію вимірювань серцевого ритму та пропонує основні рекомендації. Ці рекомендації, щонайменше, рудиментарні. Хоча додаток пропонує інтеграцію з Apple Health та Google Fit, його інтерфейс настільки простий, що можна замислитися, чи не є він також трохи тьмяним. У будь-якому разі, більшість розширених функцій додатка доступні лише в преміум-версії. *Instant Heart Rate* не використовує штучний інтелект для аналізу даних і не пропонує персоналізованих рекомендацій [3].

На рисунках 1.1-1.3 зображено скріншоти роботи застосунку.

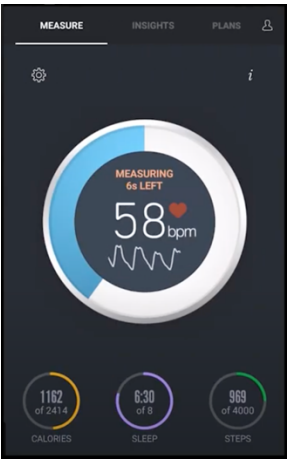


Рисунок 1.1 – Сторінка вимірювання пульсу



Рисунок 1.2 – Сторінка PPG графіку



Рисунок 1.3 – Сторінка історії вимірювань

– Heart Rate Monitor - Pulse App– це безкоштовний додаток із преміум-функціями, який також використовує камеру вашого смартфона для вимірювання частоти серцевих скорочень. Він надає детальний аналіз частоти серцевих скорочень за допомогою графіків, дозволяє експортувати дані у форматі CSV та пропонує деякі основні поради щодо здоров'я. Додаток дозволяє локально зберігати дані, а також синхронізується з Google Fit. На відміну від деяких інших додатків для вимірювання серцевого ритму, він не використовує алгоритми штучного інтелекту для виявлення аномалій або надання будь-яких рекомендацій. Інтерфейс зручний у використанні, але безпека даних може викликати занепокоєння, оскільки політика конфіденційності досить розпливчата.

На рисунках 1.4-1.7 зображено скріншоти роботи застосунку.

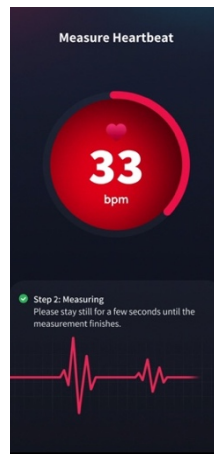


Рисунок 1.4 – Сторінка вимірювання пульсу

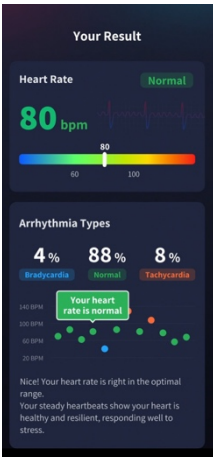


Рисунок 1.5 – Сторінка результатів вимірювання

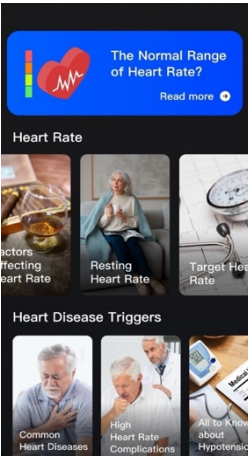


Рисунок 1.6 – Сторінка з інформацією про здоров’я від експертів

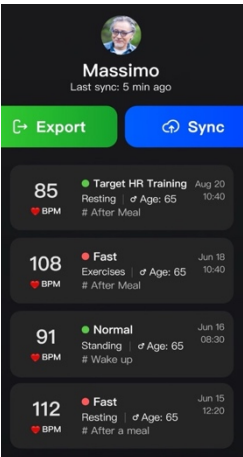


Рисунок 1.7 – Історія вимірювань

Для порівняння проєкту з аналогами було створено таблицю 1.1.

Таблиця 1.1 – Порівняння з аналогами

Функціонал	My Heart Rate Monitor - Pulse App	Instant Heart Rate	Heart Rate Monitor - Pulse App	Пояснення
Зчитування пульсу через камеру смартфона	Так	Так	Так	Усі застосунки використовують PPG для вимірювання пульсу, але розроблений застосунок оптимізовано для веббраузерів.
Аналіз даних у реальному часі	Так	Так	Так	Усі застосунки відображають графік пульсу в реальному часі.
Виявлення аномалій за допомогою ШІ	Так	Ні	Ні	Розроблений вебзастосунок використовує ChatGPT API/ Groq з промпт-інженерією для аналізу аномалій, тоді як аналоги обмежуються базовим аналізом.

Продовження таблиці 1.1

Персоналізовані рекомендації	Так	Обмежено	Обмежено	Розроблений вебзастосунок формує індивідуальні рекомендації через ChatGPT API/Groq, аналоги надають лише загальні поради.
Історія вимірювань	Так	Так	Так	Усі застосунки зберігають історію вимірювань, але розроблений проєкт та Instant Heart Rate мають зручний вебінтерфейс для перегляду.

Продовження таблиці 1.1

Безкоштовний доступ	Обмежено	Обмежено	Обмежено	Дипломний проєкт має більше безкоштовних функцій, ніж аналоги(наприклад надає результати вимірювань використовуючи слабкішу модель III)
Інтеграція з іншими платформами	Ні	Так (Apple Health, Google Fit)	Так (Google Fit)	Розроблений вебзастосунок не підтримує інтеграцію, але фокусується на автономності.
Адаптивний вебінтерфейс	Так	Ні	Ні	Дипломний проєкт працює через браузер, що забезпечує кросплатформність.

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Для реалізації функціоналу вебзастосунку My Heart Rate Monitor - Pulse App необхідно розв'язати дві основні задачі: (1) обробка PPG-сигналів для визначення частоти пульсу та (2) аналіз даних для виявлення аномалій і формування рекомендацій із використанням ChatGPT API/Groq та промпт-інженерії. Нижче наведено аналіз алгоритмів для цих задач.

Алгоритми обробки PPG-сигналів надано у таблиці 1.2.

Таблиця 1.2 – Порівняння алгоритмів обробки PPG-сигналів

Назва алгоритму	Опис	Переваги	Недоліки
Піковий детектор (Peak Detection)	Фільтрує PPG-сигнал для виявлення піків за допомогою трьох FIR-фільтрів і порогових значень [4].	Висока точність на чистих сигналах.	Висока обчислювальна складність, повільна обробка.
Швидке перетворення Фур'є (FFT)	FFT аналізує частотний спектр PPG-сигналу для визначення домінуючої частоти, яка відповідає пульсу. Цей метод ефективний для стабільних сигналів, але потребує більше обчислень і може бути менш точним при нерегулярному ритмі [5].	Висока точність оцінки HR, ефективно видалення артефактів руху, швидка обробка сигналів.	Може неправильно визначати пікові частоти при сильних артефактах руху або спектральних коливаннях.

Продовження таблиці 1.2

Автокореляційний аналіз	Використовує автокореляційну функцію для визначення затримки пульсової хвилі шляхом порівняння шаблону сигналу ЕКГ та ФПГ [6].	Висока точність визначення часових міток, оцінка якості сигналу в реальному часі, корисний для неінвазивного вимірювання тиску.	Залежність від якості сигналу, можливі обмеження при сильному шумі чи артефактах руху.
Пороговий аналіз із смуговою фільтрацією	Використовує смуговий фільтр для видалення шуму та пороговий метод для виявлення піків у RRG-сигналі, отриманому з відеопотоку камери.	Простота реалізації, швидка обробка в реальному часі, адаптивність до змін освітлення, придатність для вебзастосунків.	Чутливість до сильних артефактів руху, залежність від якості камери та спалаху, менша точність при низькій якості сигналу.

Вибір алгоритму: Для розробки проєкту вебзастосунку для зчитування пульсу з відеопотоку було обрано алгоритм порогового аналізу з фільтрацією смугового пропускання. Цей метод є похідним від загальних підходів до обробки біосигналів, таких як пороговий детектор (Peak Detection) із

додаванням смугової фільтрації, що використовується для видалення шуму та виділення корисних частот.

1.4 Аналіз та моделювання бізнес-процесів

Для розробки вебзастосунку, який відстежує частоту серцевих скорочень та надає рекомендації на основі штучного інтелекту, використовуються такі бізнес-процеси:

1. Реєстрація користувача.
2. Вимірювання частоти серцевих скорочень.
3. Аналіз фотоплетизмографії (ФПГ) та надання рекомендацій.

Оскільки ці процеси пов'язані з основною функціональністю програми, тобто збором інформації, обробкою результатів та взаємодією з користувачем, їх було обрано. Інші процеси, наприклад, зберігання даних, є допоміжними, тоді як вони зосереджені на підтримці основних процесів.

Для наочного представлення бізнес-процесів використано нотацію BPMN (Business Process Model and Notation). Нижче описано моделі двох ключових процесів: реєстрація користувача та вимірювання пульсу.

1. Модель бізнес-процесу "Реєстрація користувача"

Опис моделі (див. рисунок 1.8):

- Користувач відкриває вебзастосунок і переходить на сторінку реєстрації.
- Користувач обирає метод реєстрації.
- Якщо обрано реєстрацію з сервісом поза системою – перехід в систему реєстрації сервісу після чого створюється акаунт.
- Якщо обрано реєстрацію з поштою – перехід на форму з даними
- Користувач заповнює форму реєстрації (поля: ім'я, email, пароль з коректною інформацією і унікальною поштою).
- Система перевіряє коректність введених даних (наприклад, формат email, мінімальна довжина пароля, пошта вже є у системі).

- Якщо дані некоректні, система виділяє відповідні поля червоним і повертає користувача до заповнення форми.
- Якщо дані коректні, вони передаються на сервер.
- Користувач переходить на сторінку верифікації, де йому на пошту відправляється код.
- Якщо код прийшов створюється акаунт.
- Якщо код не прийшов, користувач може повернутись до форми, щоб змінити пошту.
- Після створення акаунту процес завершується.

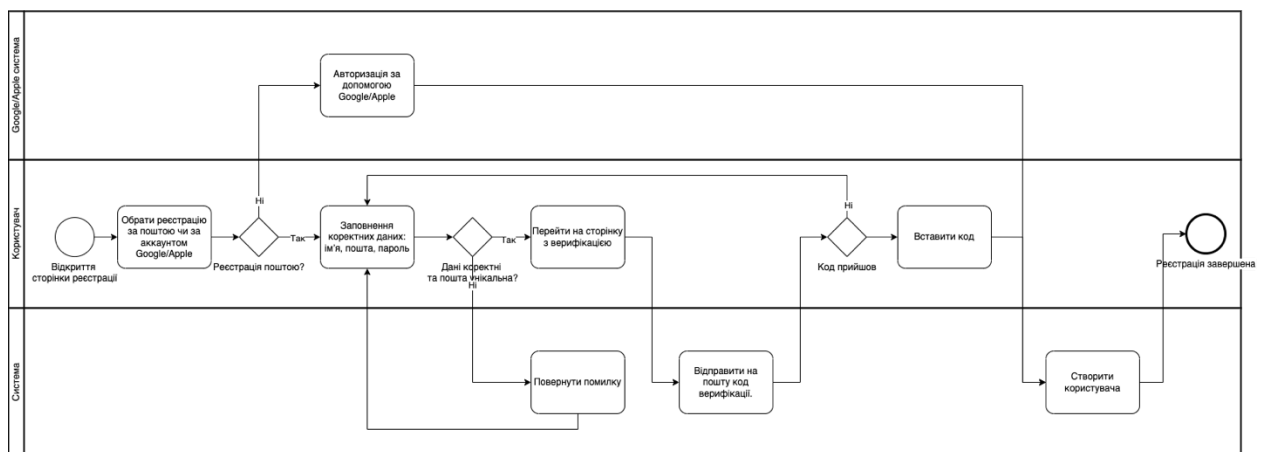


Рисунок 1.8 – Модель бізнес-процесу "Реєстрація користувача"

2. Модель бізнес-процесу "Вимірювання пульсу"

Опис моделі (див. рисунок 1.9):

- Користувач відкриває функцію вимірювання пульсу в застосунку.
- Користувач прикладає палець до камери смартфона.
- Система перевіряє якість сигналу (наприклад, чи правильно розташовано палець, чи достатнє освітлення).
- Якщо сигнал якісний, дані зчитуються протягом 20 секунд.
- Дані передаються зберігаються.
- Користувач переходить на сторінку для вводу додаткових даних.

- Користувач вводить дані: вік, стать, вага та ріст(використання слайдерів зі значеннями за замовчуванням не потребує додаткової перевірки)
- Сервер аналізує дані.
- Результати вимірювання (пульс, наявність аномалій) повертаються користувачу.
- Кінець: Процес завершується.

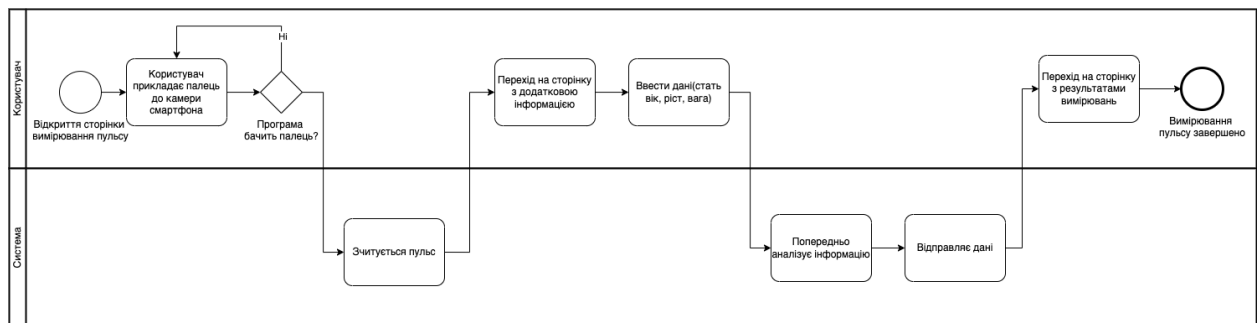


Рисунок 1.9 – Модель бізнес-процесу "Вимірювання пульсу"

3. Модель бізнес-процесу "Аналіз даних та надання рекомендацій"

Опис моделі (див. рисунок 1.10):

- Користувач відкриває вебзастосунок і переходить на сторінку інформаційної панелі.
- Сервер зберігає останнє вимірювання у історію і відправляє останній запис.
- Якщо користувач обирає створити новий запис, то натискаю на кнопку нового запису.
- Користувач переходить на сторінку вимірювання.
- Кінець: Користувач переходить у інший бізнес-процес.
- Якщо обрано продивитись останній запис, то натискає на останній запис.
- Користувач переходить на сторінку результатів.
- Сервер відправляє запит з даними запису до ШІ.
- ШІ повертає аналіз та рекомендації.
- Користувач продивляється результати вимірювань.
- Кінець: Процес завершено.

- Якщо обрано історію, то користувач натискає кнопку історії.
- Користувач переходить на сторінку історії вимірювань.
- Користувач обирає запис.
- Користувач переходить на сторінку результатів.
- Сервер відправляє запит з даними запису до ШІ.
- ШІ повертає аналіз та рекомендації.
- Користувач продивляється результати вимірювань.
- Кінець: Процес завершено.

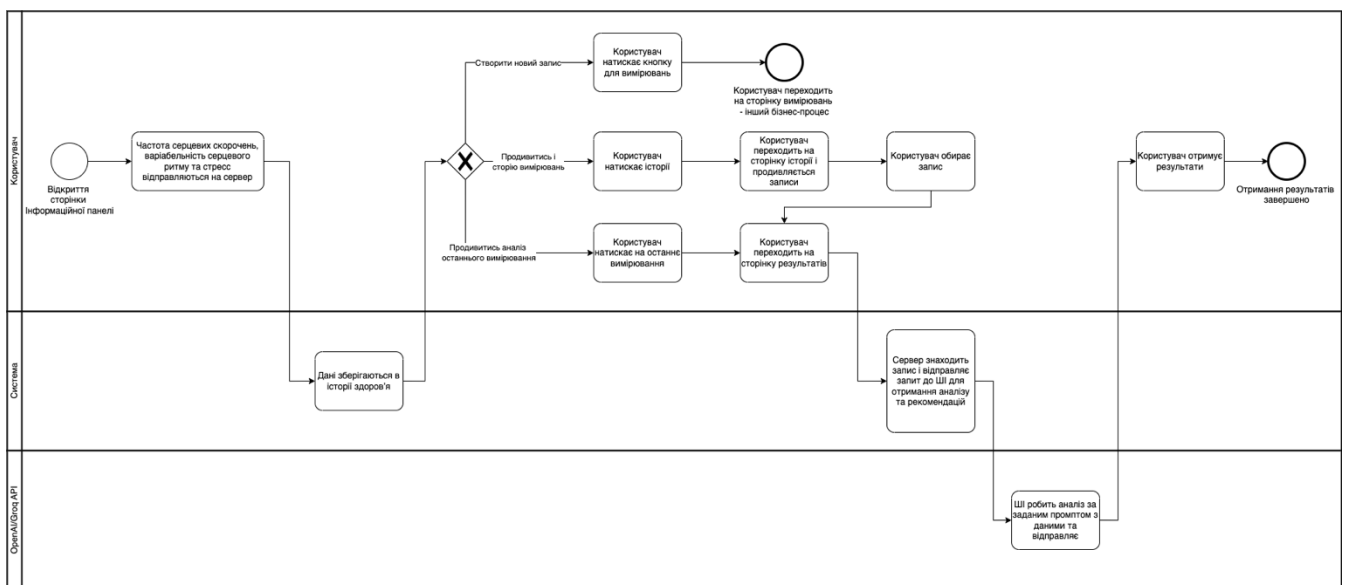


Рисунок 1.10 – Модель бізнес-процесу "Аналіз даних та надання рекомендацій "

Висновки до розділу

У цьому розділі проведено ґрунтовну оцінку розробки вебзастосунку для моніторингу серцевого ритму, який має надавати рекомендації за допомогою штучного інтелекту. Були визначені ключові аспекти дипломного проекту, а саме: проблема, вимоги, архітектура, програмне забезпечення, безпека, якість та функціональність продукту. Метою цієї діяльності є розробка доступного та безпечного вебзастосунку, заснованого на технологіях нового покоління відповідно до встановлених на даний момент стандартів.

Було проведено дослідження теми, яке показало, що моніторинг серцевого ритму є цінним інструментом у сферах здоров'я, спорту та профілактики, тоді як використання фотоплетизмографії (ФПГ) через камеру смартфона гарантує доступність разом із простотою використання. Було визначено кілька важливих бізнес-процесів: реєстрація користувачів, вимірювання серцевого ритму, а також надання індивідуальних рекомендацій. Були визначені недоліки існуючих рішень, такі як обмежена доступність, відсутність персоналізації та проблеми, пов'язані з точністю та безпекою даних, що обґрунтовує необхідність розробки інноваційного вебдодатку, який зосереджений на доступності, точності, персоналізації та безпеці даних.

Було проведено порівняльне дослідження деяких доступних програмних застосунків, таких як Instant Heart Rate та Heart Rate Monitor - Pulse App; ці застосунки використовують фотоплетизмографію (ФПГ), але не використовують штучний інтелект для виявлення аномалій або персоналізованих рекомендацій. Розроблений вебзастосунок My Heart Rate Monitor - Pulse App відрізняється адаптивним вебінтерфейсом, інтеграцією штучного інтелекту (через ChatGPT API/Groq) та розширеним спектром безкоштовних функцій. Для обробки сигналу ФПГ було обрано алгоритм аналізу порогових значень зі смуговою фільтрацією, оскільки він є простим, швидким та добре підходить для вебзастосунків.

Використання нотації BPMN у моделюванні бізнес-процесів дозволило чітко розмежувати основні процеси, тобто реєстрацію користувачів та вимірювання пульсу, що дало структурований огляд функціональності застосунку.

Отже, передпроектне опитування заклало міцну основу для подальшого розвитку вебзастосунку, визначивши його функціональні можливості, технологічні підходи та способи усунення недоліків доступних опцій.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програми є зчитування пульсу для аналізу та надання рекомендацій. Більше функцій наведено в графічному матеріалі роботи.

В таблицях 2.1-2.13 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Перегляд сторінки привітання
Use case ID	UC-01
Goals	Почати працювати з вебзастосунком
Actors	Неавторизований користувач
Trigger	Користувач вперше заходить на вебзастосунок
Pre-conditions	-
Flow of Events	Користувач відкриває застосунок у браузері. Показується сторінка привітання та знайомлення користувача з застосунком.
Extension	-
Post-Condition	Користувача переходить на сторінку вимірювання

Таблиця 2.2 – Варіант використання UC-02

Use case name	Реєстрація користувача
Use case ID	UC-02
Goals	Реєстрація нового користувача в системі
Actors	Неавторизований користувач
Trigger	Користувач бажає зареєструватися

Продовження таблиці 2.2

Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: ім'я користувача, пошта користувача, пароль в системі, та його повтор для підтвердження. Після заповнення даних користувач натискає кнопку далі, перенаправляється на сторінку підтвердження пошти. На пошту відправляється код підтвердження, користувач вводить код на сторінці коду на натискає кнопку далі. З'являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу.
Extension	В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом.
Post-Condition	Створення сторінки користувача, перехід на сторінку входу

Таблиця 2.3 – Варіант використання UC-03

Use case name	Авторизація користувача
Use case ID	UC-03
Goals	Авторизація користувача в системі
Actors	Неавторизований користувач
Trigger	Користувач бажає авторизуватися
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку авторизації. В поля для реєстрації вводяться відповідні дані: логін та пароль, натискає кнопку увійти.
Extension	В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом.
Post-Condition	Створення сторінки користувача, перехід на сторінку входу

Таблиця 2.4 – Варіант використання UC-04

Use case name	Вимірювання пульсу
Use case ID	UC-04
Goals	Виміряти пульс
Actors	Неавторизований, авторизований та авторизований з платною підпискою користувачі
Trigger	Користувач бажає виміряти пульс
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку вимірювання пульсу. Користувач прикладає палець до камери з ліхтариком смартфона. Програма бачить палець, починається вимірювання пульсу протягом 20с. Після вимірювання користувач попадає на сторінку персональних даних(у випадку відсутності), чи на сторінку з результатами.
Extension	Якщо палець прибирається з камери, вимірювання зупиняється, таймер скидується.
Post-Condition	Вимірення пульсу, перехід на сторінку результатів(гість на сторінку персональних даних).

Таблиця 2.5 – Варіант використання UC-05

Use case name	Введення персональних даних
Use case ID	UC-05
Goals	Ввести персональні дані для відстеження здоров'я
Actors	Неавторизований, авторизований та авторизований з платною підпискою користувачі
Trigger	Користувач виміряв пульс і дані не введені; користувач хоче змінити персональні дані
Pre-conditions	-

Продовження таблиці 2.5

Flow of Events	Користувач переходить на сторінку персональних даних. Користувач вводить стать, вік, вагу та ріст. Натискається кнопка зберегти.
Extension	-
Post-Condition	Збереження персональних даних, перехід далі(сторінка результатів чи інформаційна панель).

Таблиця 2.6 – Варіант використання UC-06

Use case name	Перегляд результатів
Use case ID	UC-06
Goals	Переглянути результати
Actors	Неавторизований, авторизований та авторизований з платною підпискою користувачі
Trigger	Користувач бажає отримати результати вимірювання
Pre-conditions	Користувач виміряв пульс
Flow of Events	Користувач переходить на сторінку результатів. Передивляється результати.
Extension	-
Post-Condition	Користувач успішно читає виміряні змінні та аналіз значень.

Таблиця 2.7 – Варіант використання UC-07

Use case name	Перегляд частоти серцевих скорочень
Use case ID	UC-07
Goals	Переглянути значення вимірюваного пульсу
Actors	Неавторизований, авторизований та авторизований з платною підпискою користувачі
Trigger	Користувач бажає отримати результати вимірювання – частота серцевих скорочень

Продовження таблиці 2.7

Pre-conditions	Користувач виміряв пульс
Flow of Events	Користувач переходить на сторінку результатів, або на інформаційну панель одразу після вимірювання(де показується останній запис).
Extension	-
Post-Condition	Користувач успішно читає значення частоти серцевих скорочень

Таблиця 2.8 – Варіант використання UC-08

Use case name	Перегляд варіабельності серцевого ритму
Use case ID	UC-08
Goals	Переглянути значення варіабельності серцевого ритму
Actors	Неавторизований, авторизований та авторизований з платною підпискою користувачі
Trigger	Користувач бажає отримати результати вимірювання – варіабельності серцевого ритму
Pre-conditions	Користувач виміряв пульс
Flow of Events	Користувач переходить на сторінку результатів, або на інформаційну панель одразу після вимірювання(де показується останній запис).
Extension	-
Post-Condition	Користувач успішно читає значення варіабельності серцевого ритму

Таблиця 2.9 – Варіант використання UC-09

Use case name	Перегляд стресу
Use case ID	UC-09
Goals	Переглянути значення стресу
Actors	Неавторизований, авторизований та авторизований з платною підпискою користувачі

Продовження таблиці 2.9

Trigger	Користувач бажає отримати результати вимірювання – варіабельності серцевого ритму
Pre-conditions	Користувач виміряв пульс
Flow of Events	Користувач переходить на сторінку результатів, або на інформаційну панель одразу після вимірювання(де показується останній запис).
Extension	-
Post-Condition	Користувач успішно читає значення стресу

Таблиця 2.10 – Варіант використання UC-10

Use case name	Аналіз результатів вимірювання, використовуючи слабкішої моделі III – Groq API
Use case ID	UC-10
Goals	Аналізувати значень вимірювання
Actors	Неавторизований, авторизований користувачі
Trigger	Користувач бажає отримати аналіз пульсу, варіабельності серцевого ритму та стресу.
Pre-conditions	Користувач виміряв пульс
Flow of Events	Користувач переходить на сторінку результатів.
Extension	У разі помилки з запитом API виводиться повідомлення.
Post-Condition	Користувач отримує аналіз результатів вимірювання, на можливі рекомендації для покращення значень

Таблиця 2.11 – Варіант використання UC-11

Use case name	Аналіз результатів вимірювання, використовуючи сильнішої моделі III – OpenAI API
Use case ID	UC-11
Goals	Аналізувати значень вимірювання
Actors	Авторизований користувач з платною підпискою

Продовження таблиці 2.11

Trigger	Користувач бажає отримати аналіз пульсу, варіабельності серцевого ритму та стресу.
Pre-conditions	Користувач авторизований та придбав платну підписку. Користувач виміряв пульс
Flow of Events	Користувач переходить на сторінку результатів.
Extension	У разі помилки з запитом API виводиться повідомлення.
Post-Condition	Користувач отримує аналіз результатів вимірювання, на можливі рекомендації для покращення значень

Таблиця 2.12 – Варіант використання UC-12

Use case name	Перегляд історії вимірювань
Use case ID	UC-12
Goals	Переглянути історію вимірювань
Actors	Авторизований користувач з та без підписки
Trigger	Користувач хоче продивитись історію вимірювань та переходить на сторінку історії
Pre-conditions	Користувач авторизований. Користувач виміряв пульс мінімум раз.
Flow of Events	Користувач переходить на сторінку історії. Передивляється записи по годинах та по днях. Користувач може обрати запис і продивить аналіз даних результатів.
Extension	У разі відсутності записів, сторінка пуста
Post-Condition	Користувач отримує історію вимірювань

Таблиця 2.13 – Варіант використання UC-13

Use case name	Перегляд аналізу історії вимірювань
Use case ID	UC-13
Goals	Переглянути аналіз історії вимірювань
Actors	Авторизований користувач з підпискою
Trigger	Користувач хоче продивитись аналіз усіх вимірювань

Продовження таблиці 2.13

Pre-conditions	Користувач авторизований та придбав підписку. Користувач виміряв пульс мінімум раз.
Flow of Events	Користувач переходить на інформаційну панель. Переглядаються графіки градації значень, тренди.
Extension	-
Post-Condition	Користувач отримує аналіз у вигляді графіків чи текстових рекомендацій.

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.14 наведено загальну модель вимог, а в таблиці 2.15 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.1.

Таблиця 2.14 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Перегляд сторінки привітання	FR-1	Низький	Низький
2	Система авторизації користувача	FR-2	Високий	Високий
2.1	Реєстрація користувача	FR-3	Високий	Високий
2.2	Авторизація користувача	FR-4	Високий	Високий
2.3	Вихід із акаунту	FR-5	Середній	Низький
3	Вимірювання пульсу	FR-6	Високий	Високий
4	Введення персональних даних	FR-7	Середній	Середній
5	Перегляд результатів	FR-8	Високий	Середній
5.1	Перегляд частоти серцевих скорочень	FR-9	Середній	Середній
5.2	Перегляд варіабельності серцевого ритму	FR-10	Середній	Середній

Продовження таблиці 2.14

5.3	Перегляд рівня стресу	FR-11	Середній	Середній
6	Аналіз результатів вимірювання (Groq API)	FR-12	Середній	Середній
7	Аналіз результатів вимірювання (OpenAI API)	FR-13	Середній	Середній
8	Перегляд історії вимірювань	FR-14	Середній	Низький
9	Перегляд аналізу історії вимірювань	FR-15	Середній	Середній

Таблиця 2.15 – Перелік функціональних вимог

ID вимоги	Опис
FR-1	Перегляд сторінки привітання Неавторизований користувач відкриває вебзастосунок і бачить сторінку привітання, яка знайомить із функціоналом застосунку. На сторінці доступні опції для переходу до реєстрації або авторизації.
FR-2	Система авторизації користувача Система забезпечує можливість реєстрації, авторизації та виходу з облікового запису користувача.
FR-3	Реєстрація користувача Неавторизований користувач вводить ім'я, електронну пошту, пароль і підтвердження пароля на сторінці реєстрації. Після введення даних користувач отримує код підтвердження на пошту, вводить його на сторінці підтвердження та завершує реєстрацію. У разі некоректних даних кнопка реєстрації стає неактивною, а поля з помилками підсвічуються червоним.
FR-4	Авторизація користувача Користувач вводить електронну пошту та пароль на сторінці авторизації. У разі успішної авторизації користувач перенаправляється на інформаційну панель або сторінку вимірювання. У разі помилки відображається повідомлення.

Продовження таблиці 2.15

FR-5	Вихід із акаунту Авторизований користувач може вийти з облікового запису, після чого перенаправляється на сторінку привітання або авторизації.
FR-6	Вимірювання пульсу Користувач (неавторизований, авторизований або з платною підпискою) прикладає палець до камери смартфона з увімкненим ліхтариком. Застосунок виконує вимірювання пульсу протягом 20 секунд і перенаправляє користувача на сторінку персональних даних (якщо вони не введені) або результатів. Якщо палець прибирається, вимірювання зупиняється, а таймер скидається.
FR-7	Введення персональних даних Користувач вводить стать, вік, вагу та ріст на сторінці персональних даних. Після натискання кнопки "Зберегти" дані зберігаються, і користувач перенаправляється на сторінку результатів або інформаційну панель.
FR-8	Перегляд результатів Користувач (неавторизований, авторизований або з платною підпискою) переглядає результати вимірювання пульсу на сторінці результатів, включаючи частоту серцевих скорочень, варіабельність серцевого ритму та рівень стресу.
FR-9	Перегляд частоти серцевих скорочень Користувач переглядає значення частоти серцевих скорочень на сторінці результатів або інформаційній панелі після вимірювання пульсу.
FR-10	Перегляд варіабельності серцевого ритму Користувач переглядає значення варіабельності серцевого ритму на сторінці результатів або інформаційній панелі після вимірювання пульсу.
FR-11	Перегляд рівня стресу Користувач переглядає значення рівня стресу на сторінці результатів або інформаційній панелі після вимірювання пульсу.

Продовження таблиці 2.15

FR-12	Аналіз результатів вимірювання (Groq API) Користувач (неавторизований або авторизований) отримує аналіз результатів вимірювання (пульс, варіабельність серцевого ритму, стрес) через Groq API. У разі помилки API відображається повідомлення.
FR-13	Аналіз результатів вимірювання (OpenAI API) Авторизований користувач із платною підпискою отримує аналіз результатів вимірювання (пульс, варіабельність серцевого ритму, стрес) через OpenAI API. У разі помилки API відображається повідомлення.
FR-14	Перегляд історії вимірювань Авторизований користувач переглядає історію вимірювань (по годинах і днях) на сторінці історії. Користувач може обрати запис для перегляду аналізу. Якщо записи відсутні, сторінка залишається порожньою.
FR-15	Перегляд аналізу історії вимірювань Авторизований користувач із платною підпискою переглядає аналіз історії вимірювань на інформаційній панелі у вигляді графіків, трендів або текстових рекомендацій.

	FR-01	FR-02	FR-03	FR-04	FR-05	FR-06	FR-07	FR-08	FR-09	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15
UC-01	+														
UC-02		+	+	+	+										
UC-03		+		+	+									+	+
UC-04						+	+	+	+	+	+			+	+
UC-05							+								
UC-06								+	+	+	+				
UC-07									+						
UC-08										+					
UC-09											+				
UC-10												+			
UC-11													+		
UC-12														+	
UC-13															+

Рисунок 2.1 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Час відгуку сторінки вебзастосунку становить менше 2 секунд при роботі зі швидкістю інтернету 10 Мбіт/с. Імпульсне зчитування, яке базується на камері смартфона, починається через 1 секунду після дотику пальця та завершується через 20 секунд тривалості. Аналіз результатів через Groq API або OpenAI API відбувається протягом 5 секунд за умови стабільного інтернет-з'єднання. Персональні дані користувачів зберігаються в зашифрованому вигляді за допомогою AES-256. Паролі хешуються за допомогою алгоритму bcrypt. Код підтвердження реєстрації дійсний протягом 10 хвилин. Система захищена від SQL-ін'єкцій, XSS та CSRF-атак. Інтерфейс адаптивний до роздільної здатності екрана від 320x480 до 2560x1600 пікселів, інтуїтивно зрозумілий, з підказками та повідомленнями про помилки. Кольори відповідають WCAG 2.1 з коефіцієнтом контрастності 4,5:1. Додаток працює

з англійською мовою. Підтримуються браузери Google Chrome 94+, Mozilla Firefox 93+, Apple Safari 15.0+ і Microsoft Edge 94+. Моніторинг серцевого ритму доступний на пристроях Android 10+ та iOS 14+.

2.4 Аналіз системних вимог

Вебзастосунок для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту працює на пристроях із операційними системами Android 10+, iOS 14+, Windows 10, Windows 11 або macOS 13.0 Ventura та новішими. Підтримуються браузери Google Chrome 94+, Mozilla Firefox 93+, Apple Safari 15.0+ і Microsoft Edge 94+. Для вимірювання пульсу потрібна камера з ліхтариком на мобільних пристроях і стабільне інтернет-з'єднання зі швидкістю від 10 Мбіт/с. Час відгуку сторінок становить до 2 секунд, вимірювання пульсу триває 20 секунд, аналіз через Groq API або OpenAI API — до 5 секунд. Дані шифруються за допомогою AES-256, паролі хешуються через bcrypt, система захищена від SQL-ін'єкцій, XSS і CSRF-атак. Інтерфейс адаптивний для роздільної здатності від 320x480 до 2560x1600 пікселів, відповідає WCAG 2.1 із контрастністю 4,5:1 і підтримує англійську мову. Базовий аналіз доступний через Groq API, розширений — через OpenAI API для користувачів із підпискою.

2.5 Аналіз економічних показників програмного забезпечення

Оцінки вартості та графіків для програмних проектів спираються на прогнозування розміру майбутньої системи, але індустрія програмного забезпечення часто має труднощі з точністю цих прогнозів через притаманні їм невизначеності. Ранні оцінки є складними, оскільки на цьому етапі бракує детальної інформації про систему, проте вони є важливими для укладання тендерів на контракт або оцінки доцільності проекту за допомогою аналізу витрат і вигод. Точні прогнози мають вирішальне значення для ефективного прийняття рішень [10].

Для розрахунку людино-годин та загальної вартості проекту розробки програмного забезпечення за методом точок варіантів використання (UCP) буде застосовано формальний процес. Він включає аналіз поданих варіантів використання, визначення їхньої складності, обчислення необхідних зусиль у людино-годинах та оцінку вартості проекту на основі зарплати молодшого розробника програмного забезпечення в Україні.

Спочатку вирахуємо нескориговані варіанти використання (Unadjusted Use-Case Points) – для цього потрібно визначити нескоригованої ваги варіантів використання, нескоригованої ваги актора та обчислити нескориговані точки варіантів використання.

Якщо варіанти використання написані з рівнями цілей користувача, транзакція еквівалентна кроку у варіанті використання. Знайдемо нескориговану вагу варіанту використання (UUCW) за допомогою таблиці 2.16.

Таблиця 2.16 – Нескоригована вага варіанту використання

Use-Case	Use-Case Complexity	Use-Case Weight
UC-01	Average	10
UC-02	Average	10
UC-03	Simple	5
UC-04	Complex	15
UC-05	Simple	5
UC-06	Simple	5
UC-07	Simple	5
UC-08	Simple	5
UC-09	Average	10
UC-10	Average	10
UC-11	Average	10

Продовження таблиці 2.16

UC-12	Average	10
UC-13	Average	10
Unadjusted Use Case Weight (UUCW)		110

Далі потрібно розрахувати складність акторів. Система має три актори: неавторизований, авторизований без підписки та авторизований з підпискою. Кожний актор має середню складність. Знайдемо нескориговану вагу актора (UAW) за допомогою таблиці 2.17.

Таблиця 2.17 – Нескоригована вага актора

Actor	Actor Complexity	Actor Weight
Неавторизований	Average	2
Авторизований	Average	2
Авторизований з платною підпискою	Average	2
Unadjusted Actor Weight (UAW)		6

Нескоригована вага варіанту використання (UUCW) та нескоригована вага актора (UAW) разом дають нескоригований розмір системи, який називають нескоригованими точками варіанту використання. (2.1)

$$UUCP = UUCW + UAW \quad (2.1)$$

$$UUCP = 110 + 6 = 116$$

Врахуймо технічну складність. Розглянемо 13 факторів, які впливають на вплив технічної складності проєкту на точки використання та відповідні їм ваги, як наведено в таблиці 2.18.

Таблиця 2.18 – Технічна складність

Factor	Description	Rating	Weight	Contribution
T1	Розподілена система	0	2.0	0
T2	Цільові показники часу відгуку або пропускну здатності	3	1.0	3
T3	Ефективність кінцевого користувача	4	1.0	4
T4	Складна внутрішня обробка	3	1.0	3
T5	Код має бути повторно використаним	3	1.0	3
T6	Легко встановлюється	2	0.5	1
T7	Легко використовується	2	0.5	1
T8	Портативна	3	2.0	6
T9	Легко змінюється	3	1.0	3
T10	Одночасна робота	2	1.0	2
T11	Включає спеціальні цілі безпеки	3	1.0	3
T12	Забезпечує прямий доступ для третіх сторін	3	1.0	3
T13	Потрібні спеціальні навчальні приміщення для користувачів	1	1.0	1
Total Technical Factor				33

Обчисліть коефіцієнт технічної складності (TCF) за формулою 2.2.

$$TCF = 0.6 + (0.01 \times \text{Total Technical Factor}) \quad (2.2)$$

$$TCF = 0.6 + (0.01 \times 33) = 0.93$$

Вирахуємо складності навколишнього середовища. Розглянемо 8 факторів навколишнього середовища, які можуть вплинути на виконання проєкту, та їх відповідні ваги, як наведено в таблиці 2.19.

Таблиця 2.19 – Технічна складність

Factor	Description	Rating	Weight	Contribution
E1	Знайомство з моделлю проекту, що використовується	4	1.5	6
E2	Досвід роботи з додатками	2	0.5	1
E3	Досвід об'єктно-орієнтованого проектування	3	1.0	3
E4	Здібності провідного аналітика	2	0.5	1
E5	Мотивація	4	1.0	4
E6	Стабільні вимоги	3	2.0	6
E7	Співробітники на умовах неповного робочого дня	0	-1.0	0
E8	Складна мова програмування	2	-1.0	-2
Total EFactor				19

Обчисліть коефіцієнт складності навколишнього середовища (EF) за формулою 2.3.

$$EF = 1.4 + (-0.03 \times EFactor) \quad (2.3)$$

$$EF = 1.4 + (-0.03 \times 19) = 0.89$$

Обчислимо скориговані точки варіантів використання (UCP) за формулою 2.4.

$$UCP = UUCP \times TCF \times EF \quad (2.4)$$

$$UCP = UUCP \times TCF \times EF = 116 \times 0.93 \times 0.89 = 96.01$$

Коефіцієнт продуктивності (PF) відображає кількість годин на одного UCP. Припускаємо, що $PF = 20$ [11].

Вирахуємо людино години за формулою 2.5.

$$\text{Person-Hours} = \text{AUCP} \times \text{PF} \quad (2.5)$$

$$\text{Person-Hours} = 96.01 \times 20 = 1920.2 \text{ годин}$$

Якщо припустити, що молодший інженер програмного забезпечення заробляє 32 тис. грн/місяць, праця 40 годин на тиждень, 4 тижні на місяць, можна вирахувати приблизну вартість проєкту.

$$\text{Total cost} = 200 \text{ грн/годину} \times 1920.2 \text{ годин} = 384040 \text{ грн}$$

2.6 Постановка завдання на розробку програмного забезпечення

Сучасне суспільство стурбоване проблемами зі здоров'ям, що виникають внаслідок аритмій, стресу та фізичної неактивності. Відсутність легкодоступних та простих технологічних пристроїв для реєстрації серцевого ритму перешкоджає ранньому виявленню проблем зі здоров'ям. Поточні варіанти, як правило, нав'язливі або не мають достатнього рівня точності для аналізу великомасштабних даних.

Мета розробки: Створити вебзастосунок для моніторингу серцевого ритму в режимі реального часу за допомогою штучного інтелекту. Забезпечити точний аналіз даних про здоров'я та персоналізовані рекомендації. Забезпечити зручність використання додатка та його доступність для широкої аудиторії спортсменів та медичних установ.

Тема розробки: Етапи створення, тестування, розгортання та забезпечення якості вебзастосунку для моніторингу серцевого ритму та надання рекомендацій.

Цілі розробки: Забезпечити точне вимірювання серцевого ритму в режимі реального часу за допомогою камери смартфона. Впровадити інтелектуальну обробку даних для виявлення порушень та можливих ризиків. Надавати персоналізовані рекомендації щодо фізичних вправ, відпочинку або запису на прийом до спеціалістів. Створити інтуїтивно зрозумілий

вебінтерфейс з адаптивним дизайном. Використовувати передові технології штучного інтелекту для обробки високоякісних даних.

Завдання, що потребують вирішення: Розробити модуль збору даних про серцевий ритм за допомогою камери смартфона з обробкою відеопотоку. Використання штучного інтелекту (на основі Groq API та OpenAI API) для аналізу серцевого ритму, виявлення аномалій та оцінки стресу. Запропонувати систему персоналізованих рекомендацій на основі даних про серцевий ритм, варіабельність серцевого ритму та персональних даних (вік, стать, вага, зріст). Розробити простий у використанні вебінтерфейс з функціональністю адаптивного дизайну. Запропонувати зберігання історії вимірювань та аналіз трендів для затверджених користувачів. Розробити механізм автоматизації та захисту даних відповідно до вимог безпеки. Протестувати додаток на стабільність, точність та виконання вимог.

Очікуваний результат: Вебзастосунок для моніторингу серцевого ритму в режимі реального часу з інтелектуальним аналізом та персоналізованими рекомендаціями. Підходить для використання індивідуальними користувачами, тренажерними залами та лікарнями.

Практичне значення: Покращення забезпечення пристроями моніторингу здоров'я. Ефективний моніторинг серцевого ритму разом із наданням рекомендацій щодо уникнення ускладнень зі здоров'ям. Можливе застосування в індивідуальних, пацієнтських та спортивних умовах для сприяння збалансованому життю.

Висновки до розділу

У даному розділі представлено комплексну оцінку та розробку вимог до програмного забезпечення для вебзастосунку для моніторингу серцевого ритму в режимі реального часу з використанням штучного інтелекту. Головною метою цієї розробки є створення простого у використанні, доступного та точного інструменту для вимірювання серцевого ритму, аналізу даних про здоров'я та надання персоналізованих рекомендацій.

Було описано варіанти використання програмного забезпечення, тобто реєстрацію та автентифікацію користувачів, вимірювання серцевого ритму, введення персональних даних, перегляд результатів, аналіз даних через Groq API та OpenAI API, а також перегляд історії вимірювань та її аналіз. Специфіка кожного варіанту використання детально зафіксована, що пропонує повне розуміння функціональності системи, адаптованої для різних категорій користувачів, включаючи неавторизованих користувачів, авторизованих користувачів та тих, хто має платну підписку.

Було визначено, які функціональні вимоги належать до різних модулів, тобто авторизація, вимірювання пульсу, аналіз результатів та перегляд історії. Модель вимог високого рівня визначає пріоритети та пов'язані з ними ризики. Крім того, матриця трасування вимог гарантує відповідність функціональних вимог сценаріям використання.

Було визначено нефункціональні вимоги, які охоплюють час відгуку сторінки, точність та час вимірювання імпульсів, безпеку даних, гнучкість інтерфейсу та сумісність із сучасними браузерами та мобільними операційними системами.

Було зроблено вичерпне дослідження економічних параметрів, включаючи розрахунок витрат на розробку з урахуванням виконавця, а також методологію оцінки.

Було сформульовано завдання розробки, визначені мета, тема, цілі та завдання, які включають створення модуля збору даних за допомогою камери смартфона, інтеграцію штучного інтелекту для аналізу, розробку адаптивного інтерфейсу та забезпечення безпеки даних. Очікуваним результатом є вебзастосунок, придатний для використання в індивідуальних, спортивних та медичних цілях, з практичним значенням для раннього виявлення проблем зі здоров'ям.

Отже, даний створює міцну основу для подальшого проектування та впровадження програмного забезпечення, чітко визначаючи як функціональні, так і нефункціональні вимоги та визначаючи завдання розробки.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Вебзастосунок, що використовується для моніторингу серцевого ритму та генерації рекомендацій за допомогою штучного інтелекту, має архітектуру, розроблену відповідно до парадигми клієнт-сервер, що базується на шаблоні Model-View-Controller (MVC). Ця парадигма забезпечує чіткий розподіл відповідальності між елементами системи, підвищуючи її масштабованість, зручність обслуговування та тестування. Фундаментальними компонентами системи є клієнтський інтерфейс (frontend), серверна архітектура (backend), база даних та зовнішні інтерфейси прикладного програмування (API), що використовуються для аналізу даних за допомогою штучного інтелекту. Нижче наведено вичерпне пояснення цих сутностей та їх залежностей.

Клієнтська частина (фронтенд) займається взаємодією з користувачем, виведенням даних та надсиланням запитів до серверної частини. Вона реалізована на основі бібліотеки Vue.js, яка забезпечує можливість реактивності, модульності та адаптивного дизайну для різних пристроїв. Фронтенд включає сторінки реєстрації, авторизації, вимірювання частоти серцевих скорочень за допомогою камери смартфона, введення персональних даних, відображення результатів, історії вимірювань та аналізу даних.

Серверна частина (бекенд) обробляє запити з боку клієнта, виконує бізнес-логіку, взаємодіє з базою даних та сторонніми API. Бекенд реалізовано за допомогою фреймворку NestJS, сумісного з Node.js та TypeScript, що пропонує модульну структуру та високу продуктивність [7]. GraphQL буде використовуватися як комунікаційний інтерфейс бекенду та фронтенду, що дозволяє отримувати лише необхідну інформацію, тим самим обмежуючи обсяг обмінюваних даних та підвищуючи загальну ефективність [8]. Бекенд також буде відповідальний за обробку сигналів PPG, їх фільтрацію та пересилання їх моделям штучного інтелекту для оцінки.

База даних використовує реляційну базу даних PostgreSQL для зберігання структурованих даних, таких як профілі користувачів, історія серцевого ритму, персональні дані (вік, стать, вага, зріст) та рекомендації. PostgreSQL забезпечує потужність, підтримку складних запитів та ефективну обробку великих обсягів даних [9]. Шифрування даних відбувається під час зберігання даних за допомогою AES-256, а доступ до даних контролюється за допомогою процедур автентифікації та авторизації.

Зовнішні API складаються з Groq API для неплатних користувачів та OpenAI API для платних користувачів. Ці API використовуються для аналізу даних серцевого ритму, виявлення аномалій (таких як тахікардія або брадикардія), оцінки рівня стресу та створення персоналізованих рекомендацій. Groq API забезпечує швидкий базовий аналіз, тоді як OpenAI API виконує більш точний та розширений аналіз, наприклад, прогнозування потенційних ризиків.

Взаємодія різних компонентів забезпечується безпечним протоколом HTTPS, який також пропонує шифрування даних під час обробки передачі. Асинхронна архітектура реалізована для обробки запитів у режимі реального часу через Apollo Server for GraphQL у серверному середовищі та Apollo Client у клієнтському середовищі. Це сприяє масштабованості завдяки характеру розгортання сервера на кількох вузлах та використанню хмарних сервісів для обробки великих обсягів даних.

Діаграма компонентів, що ілюструє зв'язки між основними компонентами системи, наведена нижче.

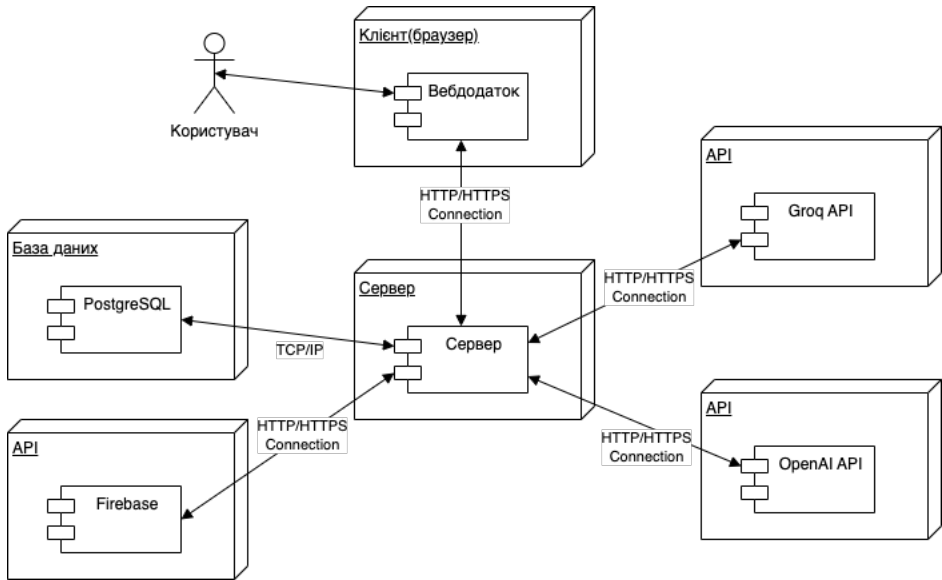


Рисунок 3.1 – Діаграма компонентів

Для деталізації поведінки системи використано діаграми послідовностей, які описують процеси реєстрації (рисунок 3.2), основні функції для запису вимірювань та отримання результатів з аналізом та рекомендаціями (рисунок 3.3).



Рисунок 3.2 – Діаграма послідовностей для авторизації



Рисунок 3.3 – Діаграма послідовностей для основних функцій

Автентифікація користувача виконується за допомогою Firebase Authentication. Вебзастосунок надсилає запит на авторизацію до Firebase Authentication (наприклад, за допомогою електронної пошти/пароля або входу в Google). Якщо інформацію успішно перевірено, Firebase Authentication повертає ID Token і Refresh Token. Для автентифікованих запитів сервер підтверджує ID Token за допомогою Firebase Admin SDK. Якщо термін дії ID Token минув, застосунок використовує Refresh Token для отримання нового ID Token від Firebase Authentication, а сервер перевіряє токен застосунку. Щоб зареєструвати нового користувача, додаток надає дані до Firebase Authentication через API. Firebase надає користувачам доступ, повертає їх UID, а сервер додає профіль користувача до бази даних та повідомляє додаток про успішне створення.

Вимірювання пульсу відбуваються повністю на клієнтській частині, після чого дані відправляються на сервер. Сервер зберігає запис в базі даних, фіксуючи час.

Для отримання результатів клієнт відправляє запит на сервер. На інформаційній панелі запрошується останній за датою запис. В іншому

випадку підіймається вся історія записів. Для отримання аналізу сервер відправляє промпт з даними на III API(Groq чи OpenAI), потім повертаються результати з аналізом та рекомендаціями.

Архітектура MVC забезпечує чітке розділення логіки: модель відповідає за дані та бізнес-логіку (NestJS, PostgreSQL), вид (View) – за відображення даних користувачу (Vue.js), а контролер – за обробку запитів і координацію між моделлю та видом (NestJS з GraphQL). Це сприяє легкості модифікації та розширення функціоналу, наприклад, додавання нових модулів аналізу чи інтеграції з іншими платформами.

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Процедури реєстрації та автентифікації користувачів здійснюються окремо за допомогою системи керування обліковими записами Firebase. Користувач ініціює реєстрацію, використовуючи своє ім'я, електронну адресу та пароль, після чого отримує код підтвердження на основі електронної пошти. Паролі хешуються за допомогою алгоритму bcrypt, а інформація безпечно зберігається в зашифрованому форматі в Firebase, токен до якої зберігається в PostgreSQL. Після успішної авторизації генерується JWT (JSON Web Token) для входу та перевіряється з кожним наступним запитом. Це забезпечує більшу безпеку та полегшує інтеграцію, ніж системи єдиного входу, що базуються на інших зовнішніх сервісах.

Інтеграція зовнішніх систем передбачає використання Groq API та OpenAI API для аналізу даних серцевого ритму. Groq API доступний безкоштовним користувачам через HTTP-запити для забезпечення швидкого базового аналізу. З іншого боку, для платних користувачів передбачено OpenAI API з розширеними аналітичними функціями, що включають прогнозування ризиків. Обидва API використовують протокол REST, доповнений HTTPS-шифруванням. GraphQL також забезпечує внутрішню

комунікацію між фронтендом та бекендом, зменшуючи дублювання передачі даних та спрощуючи процес інтеграції.

Час завантаження сторінки становить менше 2 секунд при швидкості інтернету 10 Мбіт/с завдяки оптимізації запитів GraphQL та кешуванню даних на стороні клієнта за допомогою Apollo Client.

Дані шифруються за допомогою AES-256 для захисту, а захист від атак (SQL-ін'єкції, XSS, CSRF) забезпечується внутрішньою функціональністю NestJS та бібліотекою Helmet. Код підтвердження реєстрації діє протягом 10 хвилин, що зменшує можливість несанкціонованого доступу.

Додаток підтримує сучасні браузері (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) та пристрої з Android 10+ та iOS 14+. Для зберігання структурованих даних (профілів, вимірювань, рекомендацій) використовувалася реляційна база даних PostgreSQL, оскільки вона стабільна та дозволяє виконувати складні запити. Нереляційні бази даних не застосовувалися через те, що дані мають чітку структуру і неструктурованих даних небагато.

Vue.js було обрано завдяки його легкості, реактивності та модульності. На відміну від React, Vue.js також має простіший синтаксис та менш круту криву навчання, що дозволяє швидше розробляти. Angular, представлений як варіант, зазвичай менш зручний та менш адаптований до невеликих проєктів [10].

NestJS було обрано завдяки підтримці TypeScript, модульному дизайну та інтегрованим утилітам обробки запитів. Інші варіанти, такі як Express.js, менш організовані, а Spring (Java) потребує більше ресурсів для розгортання. NestJS має високу продуктивність та простий у масштабуванні [7].

PostgreSQL було обрано завдяки його стабільності, підтримці транзакцій та здатності ефективно обробляти складні запити. Хоча MySQL пропонує ті ж функції, PostgreSQL краще справляється зі складними зв'язками та масштабуванням [9]. Нереляційні бази даних, такі як MongoDB, були виключені через організований характер даних.

GraphQL було обрано замість REST через його покращену гнучкість запитів та зменшення передачі даних [8]. Apollo Client та Apollo Server забезпечують ефективну інтеграцію та можливості кешування. Хоча REST може бути резервним варіантом, він відносно менш ефективний для динамічних запитів.

Groq API було обрано для безкоштовних користувачів, оскільки він має швидку продуктивність та ефективність у своїй роботі, тоді як OpenAI API було призначено для платних користувачів, оскільки він точніший та здатний до складного аналізу. Інші рішення, такі як Google Cloud AI, мають більш просунуті процедури інтеграції та є дорожчими.

Вибір технологій обґрунтований їхньою ефективністю, сумісністю та відповідністю вимогам проєкту, що забезпечує швидку розробку, безпеку та зручність використання.

3.3 Конструювання програмного забезпечення

Представлено детальне дослідження алгоритмів обробки даних, структур даних, бази даних, а також використаних утиліт та бібліотек. Розширення базується на клієнт-серверній архітектурі, де фронтенд (Vue.js) взаємодіє з серверною частиною (NestJS) через GraphQL API, а серверна частина інтегрується з PostgreSQL та сторонніми API (Groq, OpenAI).

3.3.1 Опис алгоритму вимірювання пульсу

Алгоритм визначення частоти серцевих скорочень базується на обробці фотоплетизмографічних (PPG) сигналів з відеопотоку камери телефону. Аналіз зміни кольору (червоний канал) використовується для виявлення пульсацій, з подальшою смуговою фільтрацією та пороговим значенням для визначення частоти серцевих скорочень.

WebRTC використовується для ініціалізації задньої камери смартфона з роздільною здатністю 300x300 пікселів та частотою кадрів 30 кадрів на

секунду. Спалах увімкнено для забезпечення світла для пальця користувача для забезпечення стабільного освітлення.

Кожен кадр відеопотоку обробляється контекстом полотна. Розраховується середнє значення RGB-компонентів (червоний, зелений, синій) усіх пікселів. RGB перетворюється в простір HSV, де перевіряються насиченість та яскравість, щоб визначити, чи палець належним чином закриває камеру (зображення має бути "червонуватим").

Якщо кадр дійсний (червоний колір), лічильник дійсних кадрів збільшується на один. Після 60 дійсних кадрів (приблизно 2 секунди з 30 FPS) ініціюється вимірювання. Якщо кадр недійсний, лічильник скидається, і користувачеві пропонується закрити камеру.

Значення відтінку HSV фільтрується через смуговий фільтр для видалення постійного струму та шуму. Розділення в діапазоні частоти імпульсів здійснюється через фільтр Баттерворта з коефіцієнтами альфа та бета.

Оброблений сигнал передається до об'єкта PulseDetector, який підтримує позитивні та негативні відхилення. Додатні значення (зростаючі піки) та негативні значення (спадаючі піки) усереднюються в буферах. Пік вважається виявленим, якщо його значення перевищує 50% від середнього піку в буфері, за умови, що попередній стан був спадом. Піковий інтервал (тривалість між зростаючими піками) зберігається, якщо він потрапляє в діапазон від 0,4 до 1,5 секунди.

Середній період обчислюється щосекунди з 5 останніх дійсних періодів. Частота серцевих скорочень (BPM) = $60 / \text{середній період (у секундах)}$. Повертає 0, якщо періодів недостатньо або вони недійсні.

Після завершення вимірювання інтервали ЧСС між піками перетворюються на мілісекунди. Якщо є щонайменше 2 інтервали, варіабельність серцевого ритму (HRV) обчислюється як квадратний корінь із середнього значення квадратів різниць між послідовними інтервалами. Стрес приблизно визначається як $100 - \text{HRV} * 1,5$, обмежений інтервалом 0–100.

Дані імпульсів надсилаються до мутації pulseHistoryCreate через GraphQL для включення до бази даних.

3.3.2 Опис структури даних

Дані користувача: ім'я, email та пароль зберігається через Firebase. Інші дані зберігаються у PostgreSQL базі даних. Ключові структури:

- Profile: Містить ідентифікатор, firebase token – ідентифікатор користувача у Firebase, стать, зріст та час створення.
- PulseHistory містить ідентифікатори запису та користувача, який зробив вимірювання, значення пульсу та дату вимірювання.
- WeightHistory містить ідентифікатори запису та користувача значення ваги та дату вимірювання.

3.3.3 Опис бази даних

В якості системи управління базами даних використовується PostgreSQL. Опис таблиць бази даних наведено у таблицях 3.1-3.3. Модель бази даних наведена на рисунку 3.4.

Таблиця 3.1 – Опис таблиці Profile

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор профілю
firebase_token	varchar(128)	Ідентифікатор користувача у Firebase
gender	int	Стать (female, male, preferNotToSay)
height	float	Зріст користувача (см)
created_at	timestamp	Дата створення профілю

Таблиця 3.2 – Опис таблиці PulseHistory

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор вимірювання
user_id	integer	Ідентифікатор користувача (FK)
pulse	float	Частота пульсу (bpm)
created_at	timestamp	Дата вимірювання

Таблиця 3.3 – Опис таблиці WeightHistory

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор вимірювання
user_id	integer	Ідентифікатор користувача (FK)
weight	float	Вага користувача (кг)
created_at	timestamp	Дата вимірювання

Таблиці PulseHistory і WeightHistory пов’язані з Profile через зовнішній ключ user_id, що забезпечує асоціацію вимірювань із профілем користувача. Поле created_at автоматично заповнюється під час створення запису.

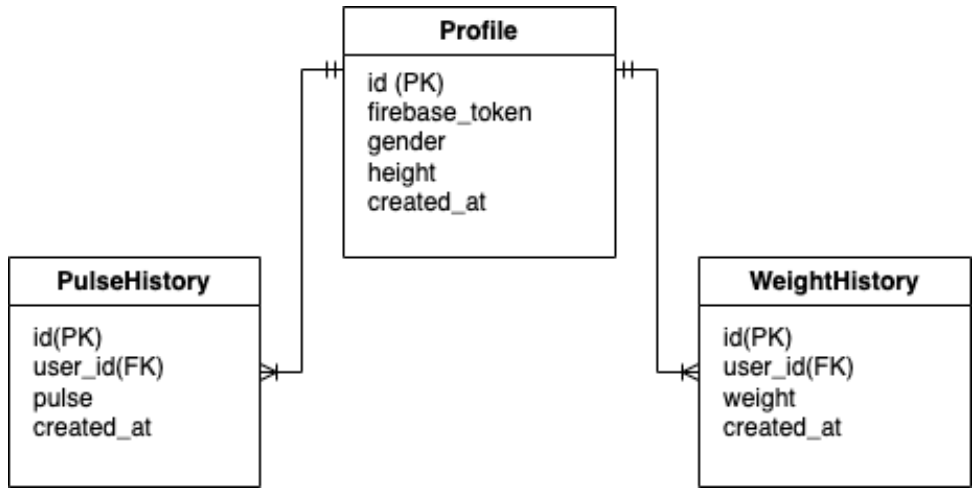


Рисунок 3.4 – ER діаграма сутностей

ER-діаграма на рисунку ілюструє зв’язки між таблицями Profile, PulseHistory і WeightHistory, де Profile є батьківською таблицею, а PulseHistory і WeightHistory містять зовнішні ключі до Profile.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.4.

Таблиця 3.4 – Опис утиліт

№	Назва	Опис
1	Apollo Client/Server	Інструменти для роботи з GraphQL. Relay, як альтернатива, складніший у налаштуванні.
2	Vite	Інструмент для збирання фронтенду, швидший за Webpack.
3	IntelliJ IDEA	IDE для розробки бекенду через підтримку TypeScript і NestJS.
4	Visual Studio Code	IDE для фронтенду через легкість і розширення для Vue.js.
5	Postman	Для тестування GraphQL-запитів.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Додаток обробляє конфіденційні персональні дані, які мають бути добре захищені від потенційних витоків, несанкціонованого використання та порушень безпеки. Firebase, що використовується як для авторизації, так і для зберігання даних, пропонує покращену систему безпеки завдяки інтегрованим функціям автентифікації та правил безпеки Firebase. Автентифікація Firebase захищає особисті дані користувачів за допомогою безпечних процедур автентифікації, тоді як правила безпеки обмежують доступ до авторизованих осіб на основі їхніх UID. Це запобігає доступу неавторизованих користувачів до конфіденційних даних, таких як профілі користувачів.

Управління вразливостями стосується пошуку та виправлення потенційних вразливостей у коді застосунку, залежностях та конфігурації. Критичні вразливості включають SQL-ін'єкції через неправильну обробку запитів PostgreSQL, XSS-атаки через неправильну перевірку вхідних даних клієнтом, CSRF-атаки через незахищені кінцеві точки GraphQL та розкриття ключів API для Groq та OpenAI через неправильне зберігання. Запобігання цим проблемам здійснюється шляхом регулярного сканування коду за допомогою OWASP ZAP, яке виявляє проблеми безпеки, такі як недостатній захист HTTP-заголовків або неадекватні налаштування CORS.

Такі залежності, як Vue.js, NestJS, оновлюються за допомогою npm-аудиту для усунення відомих вразливостей. Потенційно підозріла активність, така як велика кількість невдалих запитів на автентифікацію, реєструється за допомогою бібліотеки Winston у NestJS для моніторингу можливих атак методом повного перебору. Тестування безпеки включає статичний та динамічний аналіз коду та тестування стійкості до атак. Статичне тестування виконується за допомогою ESLint та TSLint для пошуку проблем у кодовій базі фронтенду (Vue.js) та бекенду (NestJS), наприклад, небезпечних методів

обробки інформації або пошкодженої валідації. Динамічне тестування проводиться за допомогою Postman та автоматизованих Jest-тестів, які перевіряють кінцеві точки GraphQL на захист від шкідливих запитів, тобто отримання профілю без токена JWT. Тестування на проникнення включає імітацію атак, таких як SQL-ін'єкція, XSS та CSRF. Результати тестування підтверджують, що перевірка вхідних даних для мутацій GraphQL та шифрування даних успішно запобігають серйозним загрозам.

Доступ до бази даних здійснюється за принципом керування доступом на основі ролей, тому дані, тобто історію серцевого ритму, можуть переглядати або змінювати лише авторизовані користувачі. Дані в PostgreSQL шифруються за допомогою AES-256, а обмін даними між клієнтом, сервером та API відбувається через HTTPS з використанням TLS 1.3. Firebase автоматично використовує TLS для всіх передач даних, забезпечуючи захист від перехоплення. Відстеження запитів, зроблених до зовнішніх API (наприклад, Groq та OpenAI), дозволяє виявляти підозрілу поведінку, тобто надзвичайно велику кількість запитів. У хмарному середовищі впроваджено мережеві політики, які обмежують доступ до сервера лише відомими IP-адресами. Резервні копії баз даних у Firestore та PostgreSQL слугують засобом для аварійного відновлення.

Ключові ризики включають перехоплення даних під час передачі, витік біометричної інформації через порушення безпеки бази даних та несанкціонований доступ до API через викрадені ключі. Ці ризики пом'якшуються за допомогою шифрування, регулярної ротації ключів, нагляду та ізоляції компонентів.

Висновки до розділу

У цьому розділі пояснюється дослідницький підхід, використаний у проектуванні та розробці вебзастосунку на основі штучного інтелекту для моніторингу частоти серцевих скорочень та ваги. Архітектура застосунку є клієнт-серверною з фреймворком Model-View-Controller (MVC), який

підтримує належне розділення логіки, масштабованість та простоту тестування. Клієнтська частина, розроблена за допомогою Vue.js, відповідає за спрощення взаємодії з користувачем, керування відеопотоком та представлення даних у графічному вигляді. Серверна частина NestJS керує запитами через GraphQL, взаємодіє з Firebase та PostgreSQL та іншими зовнішніми API (Groq для безкоштовних користувачів, OpenAI для платних користувачів). Структуровані дані зберігаються в зашифрованому вигляді в Firebase та базі даних PostgreSQL. Діаграми компонентів та послідовностей ілюструють, як модулі взаємодіють один з одним, тобто процеси авторизації, вимірювання частоти серцевих скорочень та аналізу даних.

Архітектурні рішення передбачають використання JWT для автентифікації, GraphQL для оптимального зв'язку, шифрування даних та захисту від атак через NestJS. Технологічний стек був обраний, оскільки він є ефективним, сумісним та відповідає потребам проекту, що підтверджено порівнянням з альтернативами. Додаток пропонує гнучкість інтерфейсу, сумісність із сучасними браузерами та швидку реакцію.

Алгоритм обробки, що застосовується до сигналу PPG, включає смугову фільтрацію та пороговий аналіз для визначення пульсу, варіабельності серцевого ритму та рівня стресу, а також зберігання оброблених даних. База даних PostgreSQL розроблена на основі третьої нормальної форми, де зв'язки встановлюються за допомогою зовнішніх ключів. Використані утиліти та бібліотеки підвищують ефективність розробки разом із процедурами тестування.

Оцінка безпеки включає управління вразливостями, тестування та безпеку в контейнерах та хмарі.

Результати, представлені в цьому розділі, складають основу для розробки безпечного, ефективного та простого у використанні вебдодатку відповідно до сучасних стандартів та очікувань користувачів.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Для оцінки якості програмного забезпечення використано метод статичної оцінки коду, який дозволяє виявляти потенційні проблеми в коді без його виконання. У якості інструменту статичного аналізу застосовується SonarQube [12] — платформа, що забезпечує комплексний аналіз коду за різними метриками, включаючи безпеку, надійність, підтримуваність, покриття тестами та дублювання коду.

Метриками для оцінки якості ПЗ обрано наступні:

- Цикломатична складність — для оцінки складності структури коду;
- Безпека — кількість відкритих проблем, пов'язаних із вразливостями безпеки;
- Надійність — кількість помилок, що можуть вплинути на стабільність роботи програми;
- Підтримуваність — кількість проблем, що ускладнюють подальший розвиток і супровід коду;
- Покриття тестами — відсоток коду, охопленого автоматичними тестами;
- Дублювання коду — відсоток повторюваного коду;
- Гарячі точки безпеки — потенційні місця в коді, що потребують додаткової перевірки на вразливості.

Результати аналізу фронтенду, отримані за допомогою SonarQube, наведено на рисунку 4.1.

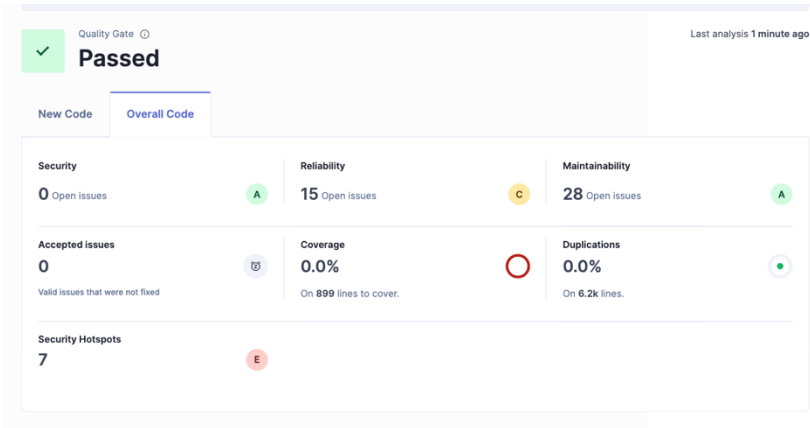


Рисунок 4.1 – Статичний аналіз фронтенду

Загалом перевірка має статус Passed, що свідчить про відсутність критичних помилок.

- Цикломатична складність: 443.
- Безпека: 0 відкритих проблем, 0 прийнятих проблем, 0 валідних проблем не виправлено, 7 гарячих точок безпеки.
- Надійність: 15 відкритих проблем (0 для backend, 15 для frontend).
- Підтримуваність: 28 відкритих проблем.
- Покриття тестами: 0.0% (на 889 рядках коду для покриття).
- Дублювання: 0.0% (на 6.2 тис. рядків).

Результати аналізу бекенду, отримані за допомогою SonarQube, наведено на рисунку 4.2.

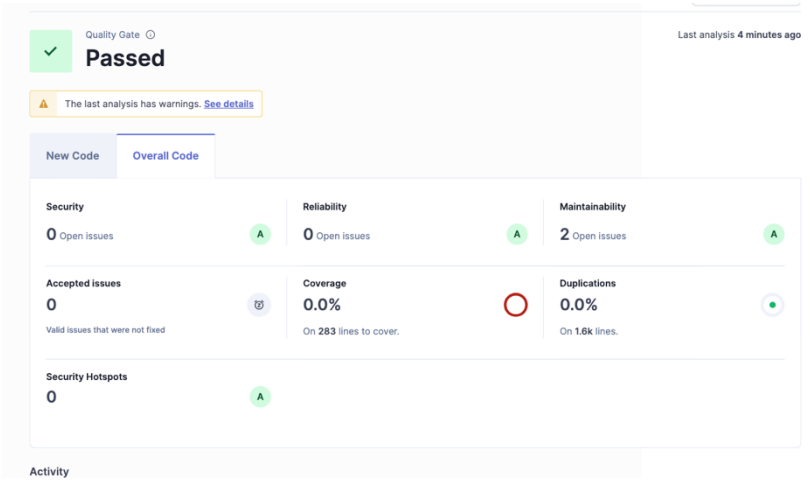


Рисунок 4.2 – Статичний аналіз бекенду

Перевірка має статус Passed, що свідчить про відсутність критичних помилок.

- Цикломатична складність: 188.
- Безпека: 0 відкритих проблем, 0 прийнятих проблем, 0 валідних проблем не виправлено, 0 гарячих точок безпеки.
- Надійність: 0 відкритих проблем.
- Підтримуваність: 2 відкриті проблеми.
- Покриття тестами: 0.0% (на 283 рядках коду для покриття).
- Дублювання: 0.0% (на 1.6 тис. рядків).

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування». Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1-4.15

Таблиця 4.1 – Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Ім'я, email, пароль
Опис проведення тесту	Користувач вводить ім'я, email, пароль, повтор пароля і натискає "Зареєструватися"
Очікуваний результат	Новий користувач створений, перенаправлення на сторінку головну сторінку
Фактичний результат	Збігається з очікуванням.

Таблиця 4.2 – Реєстрація з некоректним email

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Ім'я, email, пароль
Опис проведення тесту	Користувач вводить ім'я, email, пароль, повтор пароля і натискає "Зареєструватися"
Очікуваний результат	Поле email підсвічується червоним, реєстрація не пройдена
Фактичний результат	Збігається з очікуванням.

Таблиця 4.3 – Авторизація користувача

Початковий стан системи	Користувач на екрані авторизації
Вхідні дані	email, пароль
Опис проведення тесту	Користувач вводить email, пароль і натискає "Увійти"
Очікуваний результат	Перенаправлення на інформаційну панель
Фактичний результат	Збігається з очікуванням.

Таблиця 4.4 – Авторизація з некоректним паролем

Початковий стан системи	Користувач на екрані авторизації
Вхідні дані	email, пароль
Опис проведення тесту	Користувач вводить email, пароль і натискає "Увійти"
Очікуваний результат	Повідомлення про некоректний пароль, кнопка "Увійти" неактивна
Фактичний результат	Збігається з очікуванням.

Таблиця 4.5 – Перегляд сторінки привітання

Початковий стан системи	Користувач на екрані авторизації
Вхідні дані	-
Опис проведення тесту	Користувач відкриває застосунок у браузері
Очікуваний результат	Відображається сторінка привітання, перенаправлення на вимірювання
Фактичний результат	Збігається з очікуванням.

Таблиця 4.6 – Вимірювання пульсу

Початковий стан системи	Користувач на сторінці вимірювання пульсу
Вхідні дані	Палець користувача на камері смартфона
Опис проведення тесту	Користувач прикладає палець, програма вимірює пульс протягом 20 секунд
Очікуваний результат	Перенаправлення на сторінку результатів із даними пульсу
Фактичний результат	Збігається з очікуванням.

Таблиця 4.7 – Переривання вимірювання пульсу

Початковий стан системи	Користувач на сторінці вимірювання пульсу
Вхідні дані	Палець користувача на камері
Опис проведення тесту	Користувач прикладає палець, потім забирає його
Очікуваний результат	Вимірювання зупиняється, таймер скидається
Фактичний результат	Збігається з очікуванням.

Таблиця 4.8 – Введення персональних даних

Початковий стан системи	Користувач на сторінці персональних даних
Вхідні дані	Стать, вага, вік, ріст
Опис проведення тесту	Користувач вводить дані, натискає "Зберегти"
Очікуваний результат	Дані збережено, перенаправлення на сторінку результатів
Фактичний результат	Збігається з очікуванням.

Таблиця 4.9 – Перегляд результатів результатів вимірювання

Початковий стан системи	Користувач на сторінці результатів після вимірювання
Вхідні дані	-
Опис проведення тесту	Користувач переглядає результати вимірювання
Очікуваний результат	Відображаються значення частоти серцевих скорочень, варіабельність та стрес
Фактичний результат	Збігається з очікуванням.

Таблиця 4.10 – Аналіз результатів через Grok API

Початковий стан системи	Користувач на сторінці результатів після вимірювання(неавторизований)
Вхідні дані	Дані пульсу, варіабельності, стресу
Опис проведення тесту	Система відправляє результати вимірювань у ІІІ та отримує аналіз з рекомендаціями.
Очікуваний результат	Відображається текстовий аналіз і рекомендації
Фактичний результат	Збігається з очікуванням.

Таблиця 4.11 – Отримання підписки

Початковий стан системи	Користувач на сторінці результатів після вимірювання
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку “AI Research” для платної підписки
Очікуваний результат	Користувач на переходить на сайт оплати, після оплати повертається на сторінку
Фактичний результат	Використовується імітація оплати – повідомлення, що користувач отримав підписку.

Таблиця 4.12 – Аналіз результатів через OpenAI API

Початковий стан системи	Авторизований користувач із підпискою на сторінці результатів
Вхідні дані	Дані пульсу, варіабельності, стресу
Опис проведення тесту	Система запитує аналіз через OpenAI API
Очікуваний результат	Відображається текстовий аналіз і рекомендації
Фактичний результат	Збігається з очікуванням.

Таблиця 4.13 – Перегляд історії вимірювань

Початковий стан системи	Авторизований користувач на сторінці історії
Вхідні дані	-
Опис проведення тесту	Користувач переглядає записи вимірювань
Очікуваний результат	Відображаються записи за годинами та днями
Фактичний результат	Збігається з очікуванням.

Таблиця 4.14 – Перегляд аналізу історії вимірювань

Початковий стан системи	Авторизований користувач із підпискою на інформаційній панелі
Вхідні дані	-
Опис проведення тесту	Користувач переглядає графік історії вимірювань.
Очікуваний результат	Відображається графік градації значень
Фактичний результат	Збігається з очікуванням.

4.3 Опис контрольного прикладу

Заходячи на сайт користувач бачить сторінку привітання (рис. 4.3).



Рисунок 4.3 – Сторінка привітання

Після цього користувач попадає на сторінку вимірювання пульсу и починає виміряти дотикаючи палець до камери (рис. 4.4).

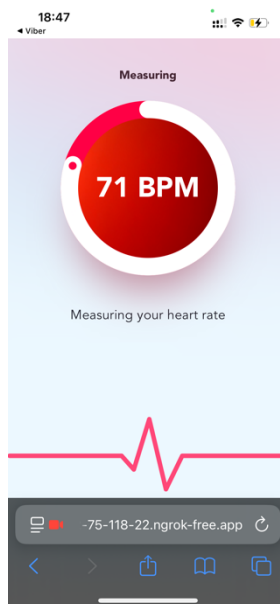


Рисунок 4.4 – Сторінка вимірювання

Через 20 секунд користувач переходить до результатів (рис. 4.5), але якщо користувач неавторизований спочатку потрібно ввести персональні дані (рис. 4.6).

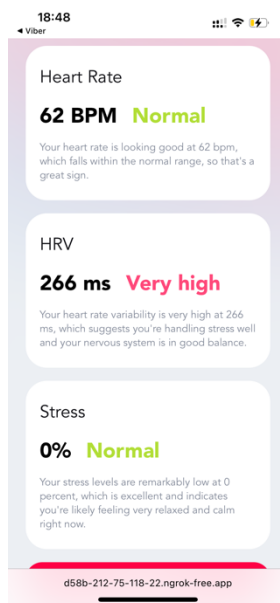


Рисунок 4.5 – Сторінка результатів

Рисунок 4.6 – Сторінка персональних даних

Якщо користувач ще неавторизований, переходить на сторінку авторизації, цей крок користувач може пропустити.

Рисунок 4.7 – Сторінка авторизації

Користувач може як створити новий акаунт (рис. 4.8) та увійти у існуючий (рис. 4.9).

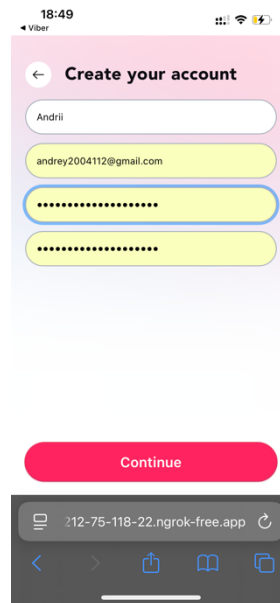


Рисунок 4.9 – Сторінка реєстрації

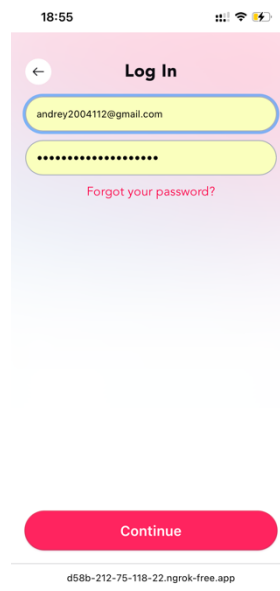


Рисунок 4.9 – Сторінка входу

Після кожного вимірювання чи авторизації користувач попадає на головну сторінку – інформаційну панель (рис. 4.10).

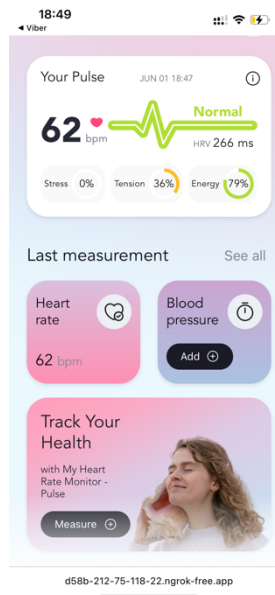


Рисунок 4.10 – Сторінка інформаційної панелі

Користувач може легко зробити нове вимірювання, натиснувши кнопку, переходить на сторінку вимірювань (рис. 4.11).

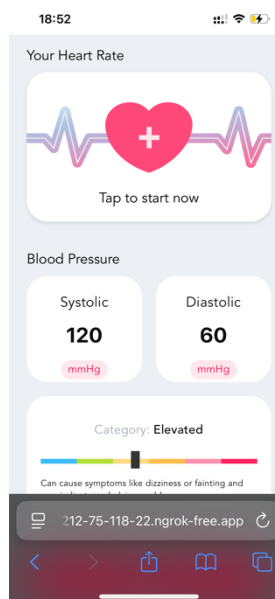


Рисунок 4.11 – Сторінка нових вимірювань

Авторизований користувач може купити платну підписку, натиснувши кнопку "AI Research" на сторінці результатів вимірювання чи історії (рис. 4.12).

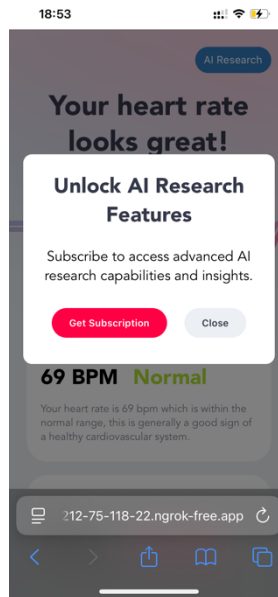


Рисунок 4.12 – Сторінка результатів вимірювання (імітація купівлі підписки)

З підпискою користувач має доступ до кращого аналізу та аналізу історії (рис. 4.13).

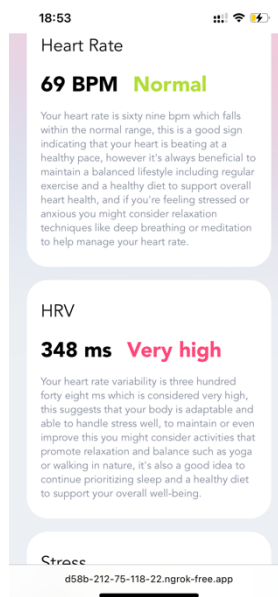


Рисунок 4.13 – Сторінка результатів вимірювання (сильніша можедь III)

Авторизований користувач має доступ до історії вимірювань з можливістю видалення та перегляду аналізу кожного запису (рис. 4.14).

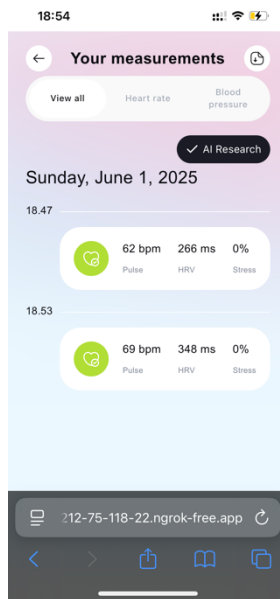


Рисунок 4.14 – Сторінка історії вимірювань

Авторизований користувач з підпискою має доступ до графіку історії (рис. 4.15).

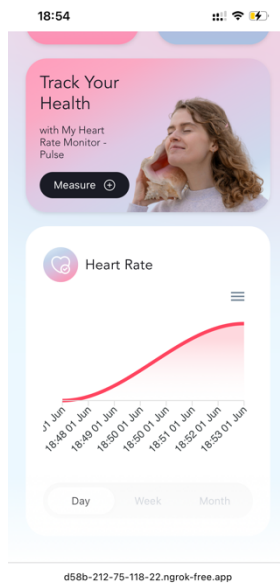


Рисунок 4.15 – Інформаційна панель користувача з підпискою

Натиснувши кнопку налаштувань, користувач може перейти на сторінку налаштувань, де є можливість виходу з акаунту (рис. 4.16).

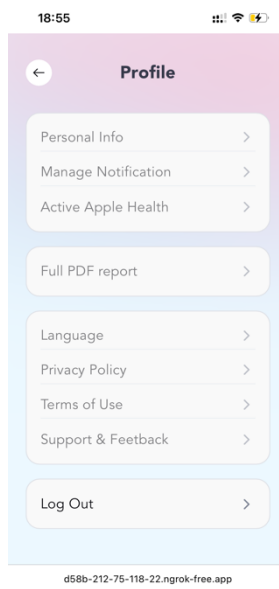


Рисунок 4.16 – Сторінка налаштувань

Висновки до розділу

У цьому розділі було розпочато детальне дослідження тестування та якості програмного забезпечення з використанням сучасних підходів та інструментів. Статичний аналіз коду було проведено на платформі SonarQube для оцінки якості програмного забезпечення на основі фундаментальних показників: цикломатичної складності, безпеки, надійності, ремонтпридатності, покриття тестами та дублювання коду. Фронтенд-аналіз виявив 443 цикломатичні проблеми складності, 15 відкритих проблем надійності, 28 проблем ремонтпридатності, 7 гарячих точок безпеки, але жодних помилок безпеки чи дублювання коду. Покриття тестами становить 0,0% з 889 рядків коду. Бекенд-аналіз виявив меншу цикломатичну складність (188), 2 проблеми ремонтпридатності, жодних проблем безпеки, надійності чи дублювання коду, але однакове 0,0% покриття тестами для 283 рядків. Обидва програмні модулі отримали статус "Зараховано", що означає відсутність критичних помилок; проте, через низьке покриття тестами існує потреба в покращенні автоматизованої діяльності з тестування.

Ручне тестування програмного забезпечення проводилося на основі "Програми та методології тестування". Перевірені критичні функції включали

реєстрацію та автентифікацію користувачів, обробку дефектних даних, відображення сторінки привітання, вимірювання частоти серцевих скорочень, введення персональних даних, аналіз результатів через Grok API та OpenAI API, перегляд історії вимірювань та придбання підписки. Усі тести пройшли успішно, фактичні результати відповідали очікуваням, що підтвердило належне функціонування функцій програмного забезпечення.

У контрольному прикладі продемонстровано процес взаємодії користувача з системою: від відображення сторінки привітання до вимірювання частоти серцевих скорочень, введення персональних даних, авторизації, перегляду результатів та історії вимірювань. Також є додаткові функції для авторизованих користувачів, які мають підписку, такі як аналіз історії та графіків вимірювань, які додають більше функціональності. Інтерфейс програмного забезпечення, включаючи сторінки авторизації, реєстрації, панелі інструментів, налаштувань та результатів, працює стабільно та відповідає вимогам користувачів.

Таким чином, проведене тестування та аналіз гарантували достатній рівень якості програмного забезпечення, але виявлені фактори, наприклад, низьке покриття тестами та велика кількість проблем з підтримкою, потребують додаткової уваги для підвищення стабільності та масштабованості системи.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання програмного забезпечення було обрано Digital Ocean [13]. Розгортання ПЗ включало налаштування фронтенду, бекенду та бази даних за допомогою App Platform. Процес розгортання описано нижче з покроковими діями, які були виконані.

Налаштування фронтенду почалося з входу до панелі керування Digital Ocean, де був обраний пункт "Create" та "App" (рис. 5.1).

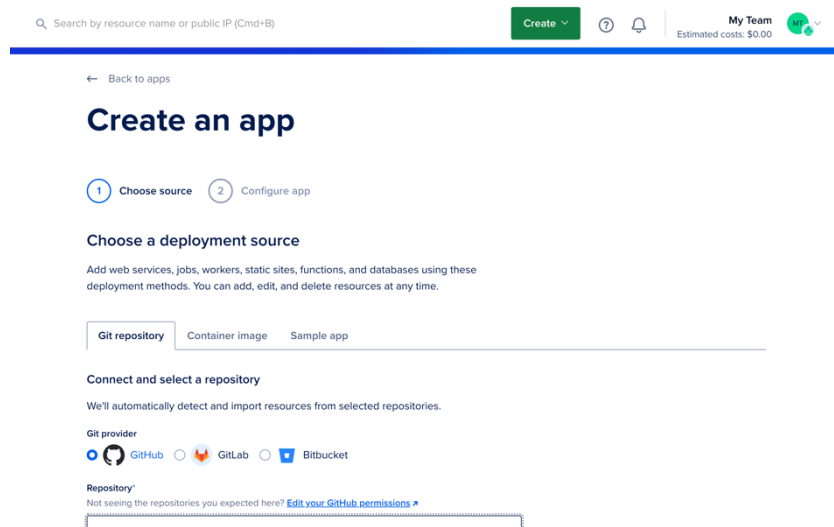


Рисунок 5.1– Створення програми на хостінгу

Був обраний GitHub як провайдер, підключено репозиторій. Була вказана гілка main, а також активовано автодеплой для автоматичного оновлення при змінах у коді (рис. 5.2).

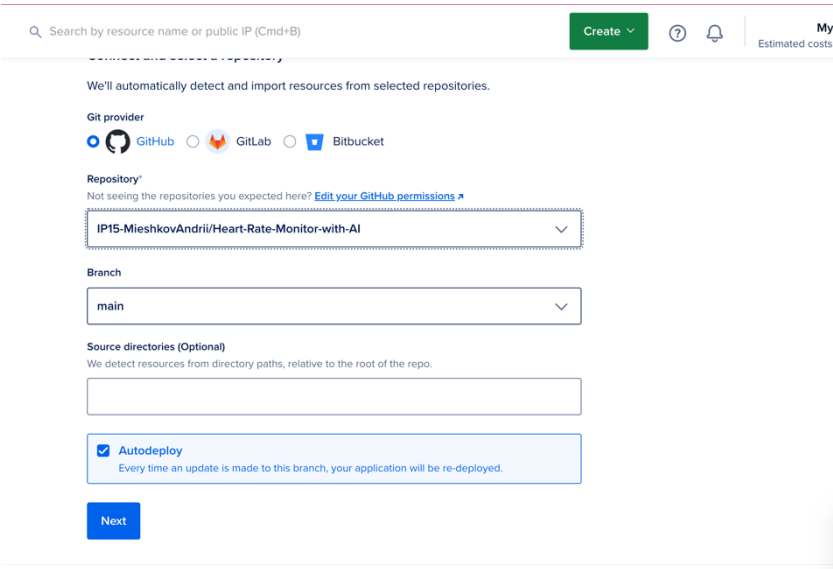


Рисунок 5.2 – Підключення репозиторію з кодом

Далі перейшли до налаштувань розгортання, де був обраний тип ресурсу "Static Site", залишивши параметри за замовчування. Було встановлено необхідні змінні оточення. Додано домен та створено додаток (рис. 5.3).

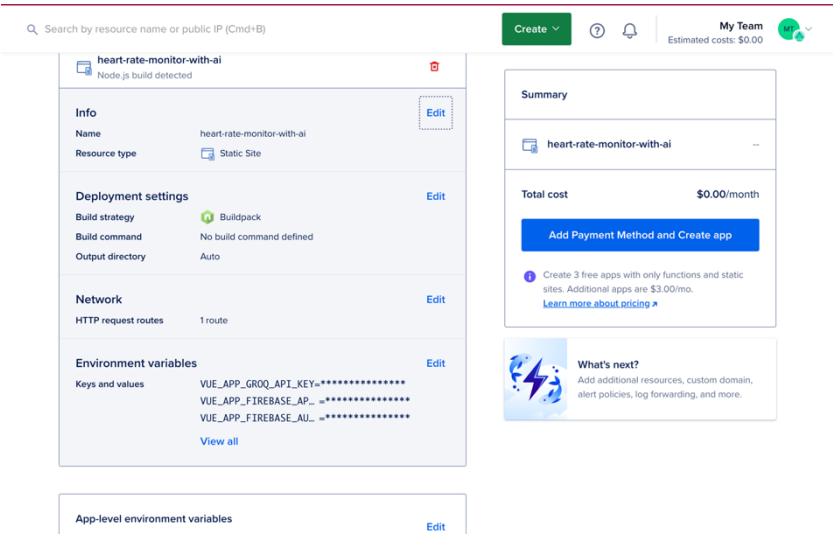


Рисунок 5.3 – Налаштування проєкту

Налаштування бекенду проводилося шляхом додавання нового компонента у тому ж проєкті з вибором типу ресурсу "Web Service". Був підключений інший репозиторій. Був обраний регіон та активовано автодеплой. Конфігурація була збережена, і чекали завершення розгортання.

Налаштування бази даних включало додавання ресурсу "Database" у проєкті з вибором PostgreSQL. Була вказана конфігурація за замовчуванням.

База даних була підключена до бекенду через змінні оточення: URL та крендінали. Статус доступності перевірявся через панель керування (рис. 5.4-5.6).

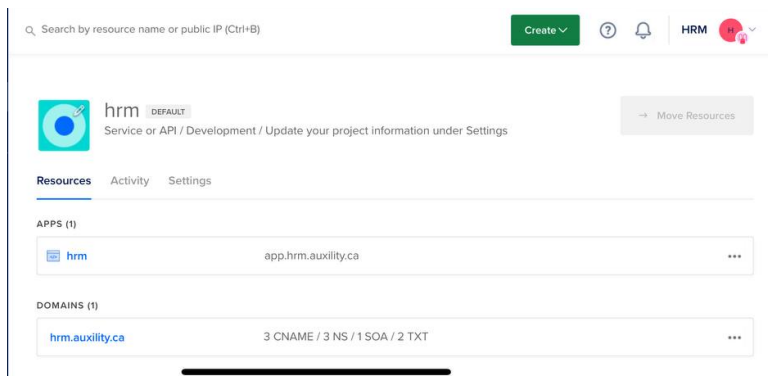


Рисунок 5.4 – Задеплований бекенд

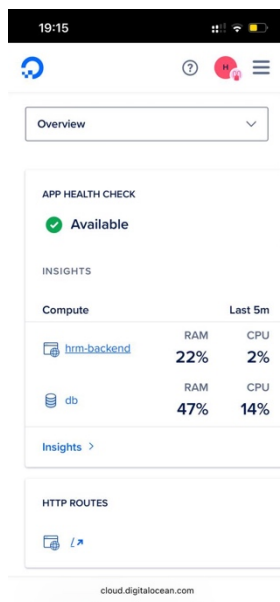


Рисунок 5.5 – Статус задеплованих компонентів

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі.

Підтримка програмного забезпечення включає регулярне оновлення коду на хостингу. Зміни у коді вносяться до репозиторіїв GitHub. Завдяки активованому автодеплою Digital Ocean автоматично виявляє оновлення та

перерозгортає додаток. Статус розгортання та продуктивність перевіряються через панель "Insights" у Digital Ocean. У разі необхідності вносилися корективи у змінні оточення або конфігурацію через розділ "App Settings".

Висновки до розділу

У цьому розділі було розглянуто процеси розгортання та супроводу програмного забезпечення, що забезпечують його стабільну роботу та доступність для користувачів. Для розгортання ПЗ було використано хмарну платформу Digital Ocean, зокрема її App Platform, яка дозволила ефективно налаштувати фронтенд, бекенд та базу даних. Процес розгортання включав покрокові дії: створення програми через панель керування Digital Ocean, підключення репозиторіїв із кодом через GitHub, активацію автодеплою для автоматичного оновлення при змінах у коді, налаштування фронтенду як статичного сайту, бекенду як вебсервісу та бази даних PostgreSQL. Для забезпечення коректної роботи компонентів були встановлені необхідні змінні оточення, підключено домен та перевірено статус доступності через панель керування. Усі етапи розгортання виконано успішно, що забезпечило стабільне функціонування системи.

Супровід програмного забезпечення організовано з урахуванням регулярного оновлення коду через репозиторії GitHub та автоматичне розгортання змін завдяки налаштованому автодеплою на Digital Ocean. Для моніторингу продуктивності та стану системи використовується панель "Insights", яка дозволяє відстежувати статус розгортання та вносити корективи у змінні оточення або конфігурацію через розділ "App Settings". Інструкція користувача, що описує взаємодію з системою, підготовлена як окремий документ, що сприяє зручності використання ПЗ.

Таким чином, розгортання та супровід програмного забезпечення виконано з використанням сучасних хмарних технологій, що забезпечують автоматизацію, надійність та легкість оновлення системи. Проведені дії гарантують стабільну роботу ПЗ, однак для підвищення ефективності

супроводу рекомендується регулярний моніторинг продуктивності та своєчасне внесення змін у конфігурацію за потреби.

ВИСНОВКИ

У результаті виконання дипломного проєкту розроблено вебзастосунок для моніторингу пульсу та надання персоналізованих рекомендацій на основі штучного інтелекту. Застосунок інтегрує технології фотоплетизмографії для збору даних через камеру смартфона, обробки сигналів та аналізу за допомогою Groq API і OpenAI API. Реалізовано адаптивний вебінтерфейс на базі Vue.js, серверну частину на NestJS з GraphQL та базу даних PostgreSQL для зберігання структурованих даних.

Мета створення доступного, точного та безпечного інструменту для моніторингу здоров'я досягнута. Вебзастосунок забезпечує вимірювання пульсу в реальному часі, виявлення аномалій, оцінку варіабельності серцевого ритму та рівня стресу, а також формування рекомендацій на основі персональних даних.

Проведено аналіз предметної області, бізнес-процесів та існуючих рішень, що дозволило усунути недоліки аналогів, зокрема обмежену персоналізацію та високу вартість. Розроблено функціональні та нефункціональні вимоги, включаючи шифрування даних, захист від атак і адаптивність інтерфейсу. Алгоритм обробки PPG-сигналів на основі смугової фільтрації та порогового аналізу забезпечує точне визначення фізіологічних показників. Статичний аналіз коду за допомогою SonarQube підтвердив відсутність критичних помилок, хоча низьке покриття тестами вказує на потребу вдосконалення. Ручне тестування всіх ключових функцій показало відповідність очікуваним результатам. Розгортання виконано на платформі Digital Ocean з автодеплоєм та моніторингом продуктивності. Інструкція користувача підготовлена для зручної взаємодії з системою.

Доцільність продовження розробки обґрунтована зростанням попиту на інтелектуальні інструменти моніторингу здоров'я. Подальший розвиток передбачає інтеграцію з носимими пристроями, підтримку неструктурованих даних та вдосконалення алгоритмів ШІ.

Стан вирішення всіх поставлених завдань є задовільним, що забезпечує створення робочого прототипу, готового до подальшого вдосконалення. Вебзастосунок має практичне значення для індивідуального моніторингу здоров'я, спортивних тренувань та медичних систем, сприяючи ранньому виявленню проблем і популяризації профілактичних заходів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Allen J. Photoplethysmography and its application in clinical physiological measurement. – Physiological Measurement, 2007. – Vol. 28, No. 3. – P. R1–R39.
- 2) Poh M.-Z., McDuff D. J., Picard R. W. Advancements in Noncontact, Multiparameter Physiological Measurements Using a Webcam. – IEEE transactions on bio-medical engineering, 2011. – Vol. 58, No. 1. – P. 7–11.
- 3) Instant Heart Rate. – URL:<https://www.instantheartate.com/>(date of access: 09.05.2025).
- 4) Goda M. A., Charlton P. H., Behar J. A. Robust peak detection for photoplethysmography signal analysis. – Physiological Measurement, 2023. – Vol. 43, No. 8. – P. 085007.
- 5) Jayadevappa B. M., Holi M. S. An Estimation Technique using FFT for Heart Rate Derived from PPG Signa. – Global Journal of Research in Engineering: F Electrical and Electronics Engineering, 2015. – Vol. 15, No. 7. – P. 43–51.
- 6) Sieczkowski K., Sondej T., Dobrowolski A. P., Olszewski R. Autocorrelation algorithm for determining pulse wave delay. – Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2016. – P. 321–326.
- 7) NestJS: Official Documentation. – URL:<https://docs.nestjs.com/> (date of access: 11.05.2025).
- 8) GraphQL: Official Documentation. – URL:<https://graphql.org/learn/>(date of access: 11.05.2025).
- 9) PostgreSQL: Official Documentation. – URL:<https://www.postgresql.org/docs/>(date of access: 11.05.2025).
- 10) Banerjee G. Use Case Points - An Estimation Approach. – [Б.м.]: [Б.В.], 2001.
- 11) Karner G. Resource Estimation for Objectory Projects. – [Б.м.]: Objective Systems SF AB, 1993.

12) SonarQube: Official Documentation. —

URL:<https://docs.sonarsource.com/>(date of access: 30.05.2025).

13) Digital Ocean: Official Documentation. —

URL:<https://docs.digitalocean.com/>(date of access: 30.05.2025).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



Дата звіту 6/3/2025
Дата редагування 6/3/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-15_Мешков_ПЗ
Автор Науковий керівник / Експерт
ІП-15_МешковХрамченко М.С.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



10
Довжина фрази для коефіцієнта подібності 2

11390
Кількість слів

88937
Кількість символів

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ПУЛЬСУ ТА НАДАННЯ
РЕКОМЕНДАЦІЙ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ**

Текст програми

КПІ.ПІ-1522.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Микола ХРАМЧЕНКО

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Андрій МЄШКОВ

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду фронтенду
<https://github.com/IP15-MieshkovAndrii/hrm-frontend>

Посилання на репозиторій з повним текстом програмного коду бекенду
<https://github.com/IP15-MieshkovAndrii/hrm-backend>

Файл PulseDiagram.vue

Реалізація функціональної задачі “Перегляд аналізу історії вимірювань”

```
<template>
  <div class="pulse-diagram">
    <div class="d-header">
      <svg width="40" height="40" viewBox="0 0 40 40" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
        <h2>Heart Rate</h2>
      </div>
      <div id="chart">
        <apexchart type="area" height="200" :options="chartOptions"
:series="series"></apexchart>
      </div>
      <div class="range-buttons">
        <button
          :class="{ active: selectedRange === 'day' }"
          @click="setRange('day')"
        >
          Day
        </button>
        <button
          :class="{ active: selectedRange === 'week' }"
          @click="setRange('week')"
        >
          Week
        </button>
        <button
          :class="{ active: selectedRange === 'month' }"
          @click="setRange('month')"
        >
          Month
        </button>
      </div>
    </div>
  </template>

<script>
```

```

import moment from 'moment';

export default {
  name: 'PulseDiagram',
  data() {
    return {
      selectedRange: 'week',
      allData: [],
      series: [{
        name: 'Heart Rate (bpm)',
        data: [],
        color: '#FF4560'
      }],
      chartOptions: {
        chart: {
          type: 'area',
          stacked: false,
          height: 200,
          zoom: {
            enabled: false
          },
        },
        dataLabels: {
          enabled: false
        },
        markers: {
          size: 0,
        },
        fill: {
          type: 'gradient',
          gradient: {
            shadeIntensity: 1,
            inverseColors: false,
            opacityFrom: 0.45,
            opacityTo: 0.05,
            stops: [20, 100, 100, 100]
          },
        },
        yaxis: {
          labels: {
            show: false,
            style: {
              colors: '#8e8da4',
            },
          },
          offsetX: 0,

```

```

        formatter: function(val) {
          return val;
        },
      },
      axisBorder: {
        show: false,
      },
      axisTicks: {
        show: false
      }
    },
    xaxis: {
      type: 'datetime',
      tickAmount: 8,
      labels: {
        formatter: function(val, timestamp) {
          return moment(new Date(timestamp)).format('HH:mm\n DD MMM');
        }
      },
      grid: {
        show: false
      },
      title: {
        align: 'left',
        offsetX: 14
      },
      tooltip: {
        shared: true
      },
      legend: {
        position: 'top',
        horizontalAlign: 'right',
        offsetX: -10
      }
    }
  }
};
},
mounted() {
  this.loadPulseData();
},
methods: {
  loadPulseData() {
    const healthHistory = JSON.parse(localStorage.getItem('healthHistory')) || [];
    const pulseData = healthHistory.map(record => ({

```

```

        x: new Date(record.date).getTime(),
        y: record.data.bpm
    }));

    this.allData = pulseData;
    this.setRange(this.selectedRange);
  },
  setRange(range) {
    this.selectedRange = range;

    const currentDate = moment();
    let startDate;

    if (range === 'day') {
      startDate = currentDate.clone().subtract(1, 'day');
    } else if (range === 'week') {
      startDate = currentDate.clone().subtract(1, 'week');
    } else {
      startDate = currentDate.clone().subtract(1, 'month');
    }

    const filteredData = this.allData.filter(data => {
      const dataDate = moment(data.x);
      return
        dataDate.isSameOrAfter(startDate)           &&
        dataDate.isSameOrBefore(currentDate);
    });

    this.series[0].data = filteredData;

    this.chartOptions.xaxis.tickAmount = range === 'day' ? 4 : range === 'week' ? 7
: 8;
  }
};
</script>

```

Файл BpmCircle.vue

Реалізація функціональної задачі “Вимірювання пульсу”

```

<template>
  <div class="bpm-circle big-shadow" @click="toggleMonitoring">
    <video ref="video" class="camera-feed" muted playsinline></video>
    <div v-if="!showCamera" class="bpm-result ">

```

```

    <span>{{ bpm || 0 }} BPM</span>
  </div>
  <div v-if="!showCamera" class="progress-container">
    <svg class="progress-circle " viewBox="0 0 100 100">
      </svg>
    </div>
  </div>
  <!-- <span v-if="!showCamera" class="timer">{{ time }} s</span> -->
</template>
<script>
import { heartRateMonitor } from '../utils/heartRateLogic';
import emitter from '@utils/eventBus';

export default {
  name: 'BpmCircle',
  data() {
    return {
      bpm: null,
      finalBpm: null,
      showCamera: true,
      progress: 0,
      time: 20,
      monitoring: false,
      canvas: null,
      measurementStartTime: null,
      measurementTimer: null,
    };
  },
  computed: {
    progressOffset() {
      const circumference = 2 * Math.PI * 45;
      return circumference * (1 - this.progress / 100);
    },
    indicatorX() {
      const radius = 45;
      const angle = (this.progress / 100) * 2 * Math.PI;
      return 50 + radius * Math.cos(angle);
    },
    indicatorY() {
      const radius = 45;
      const angle = (this.progress / 100) * 2 * Math.PI;
      return 50 + radius * Math.sin(angle);
    },
  },
  watch: {

```

```

showCamera(newValue) {
  this.$emit('update:show-camera', newValue);
},
},
mounted() {
  this.canvas = document.createElement('canvas');
  heartRateMonitor.initialize({
    videoElement: this.$refs.video,
    samplingCanvas: this.canvas,
    onBpmChange: (bpm) => {
      if (!this.finalBpm) {
        this.bpm = bpm;
        emitter.emit('bpm-update', bpm);
      }
    },
    onStatusChange: (status) => {
      if (status.includes('Hold')) {
        if (this.showCamera) {
          this.showCamera = false;
          this.measurementStartTime = performance.now();
          this.startMeasurementTimer();
        }
      } else {
        this.showCamera = true;
        this.progress = 0;
        this.time = 20;
        this.measurementStartTime = null;
        this.bpm = null;
        this.clearMeasurementTimer();
      }
    },
  });
  this.toggleMonitoring();

  const updateProgress = () => {
    if (this.monitoring && !this.showCamera && this.measurementStartTime) {
      const elapsed = (performance.now() - this.measurementStartTime) / 1000;
      this.progress = Math.min((elapsed / 20) * 100, 100);
      this.time = Math.ceil(20 - elapsed);
      if (this.time < 0) this.time = 0;
    }
    if (this.monitoring) requestAnimationFrame(updateProgress);
  };
  this.$refs.video.addEventListener('play', () => {

```

```

    if (this.monitoring) updateProgress();
  });
},
methods: {
  async toggleMonitoring() {
    if (this.monitoring) {
      this.finalBpm = this.bpm || 0;
      await heartRateMonitor.stopMonitoring();
      this.monitoring = false;
      this.showCamera = false;
      this.progress = 100;
      this.time = 0;
      this.bpm = this.finalBpm;
      this.clearMeasurementTimer();
    } else {
      this.finalBpm = null;
      this.bpm = null;
      this.progress = 0;
      this.time = 20;
      try {
        await heartRateMonitor.startMonitoring();
        this.monitoring = true;
      } catch (error) {
        console.error('Failed to start monitoring:', error);
      }
    }
  },
  startMeasurementTimer() {
    this.clearMeasurementTimer();
    this.measurementTimer = setTimeout(async () => {
      this.finalBpm = this.bpm || 0;
      await heartRateMonitor.stopMonitoring();
      this.monitoring = false;
      this.showCamera = false;
      this.progress = 100;
      this.time = 0;
      this.bpm = this.finalBpm;
      this.measurementTimer = null;
      const firebaseToken = localStorage.getItem('firebaseToken');
      if (!firebaseToken) {
        this.$router.push('/personal-data');
      } else {
        this.$router.push('/results');
      }
    });
  }
}

```

```

    }, 20000);
  },
  clearMeasurementTimer() {
    if (this.measurementTimer) {
      clearTimeout(this.measurementTimer);
      this.measurementTimer = null;
    }
  },
};
</script>

```

Файл Results.vue

Реалізація функціональних задач “Перегляд результатів”: пульсу, варіабельності та стресу; Аналіз результатів вимірювання (Groq API), Аналіз результатів вимірювання (OpenAI API).

```

<template>
  <div class="results">
    <div class="ellipse ellipse-top"></div>
    <AIResearch/>
    <div class="header">
      <h1>Your heart rate looks great!</h1>
      <div class="heart-rate-display">
        <svg class="heart-icon" width="206" height="180" viewBox="0 0 206
180" fill="none" xmlns="http://www.w3.org/2000/svg">
          </svg>
        <div class="bpm">{{ bpm }}</div>
        <svg class="wave-icon" viewBox="0 0 375 96" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
      </div>
    </div>
    <Metrics/>
    <button class="continue-btn" @click="continueButton">Continue</button>
  </div>
</template>

<script>
import { useQuery, useMutation } from '@vue/apollo-composable';
import Metrics from '@components/Metrics.vue';
import AIResearch from '@components/AIResearch.vue';
import { CREATE_PROFILE, CREATE_PULSE_HISTORY, GET_PROFILE }
from '@utils/database';

```

```

export default {
  data() {
    return {
      bpm: 0,
      profileId: null,
    };
  },
  components: {
    Metrics,
    AIResearch,
  },
  methods: {
    async continueButton() {
      try {
        const firebaseToken = localStorage.getItem('firebaseToken');
        if (!firebaseToken) {
          console.log('No firebaseToken found in localStorage');
          this.$router.push('/sign-in');
          return;
        }
        const { data: profileData, error: profileError } = await this.getProfile();
        let profileId = profileData?.profile?.id;

        if (profileError) {
          console.log('Error fetching profile:', profileError);
        }

        if (!profileId) {

          const healthData = JSON.parse(localStorage.getItem('health') || '{}');
          const profileInput = {
            gender: healthData?.gender?.toLowerCase() || 'male',
            height: parseFloat(healthData.height) || 180,
            weight: parseFloat(healthData.weight) || 68
          };

          const { data: createData, error: createError } = await this.createProfile({
            input: profileInput,
          });

          if (createError) {
            console.log('Error creating profile:' + createError);
            this.$router.push('/sign-in');
          }
        }
      }
    }
  }
}

```

```

        return;
    }

    if (createData?.profileCreate?.id) {
        console.log('Profile created with ID:', createData.profileCreate.id);
        this.profileId = createData.profileCreate.id;
        console.log('Profile created successfully!');
    } else {
        console.log('Profile creation failed: No ID returned');
        return;
    }

    if(createData && createData.profileCreate.id) {
        console.log('YAS')
    }
}

    if (this.bpm) {
        const { data: pulseData, error: pulseError } = await
this.createPulseHistory({
        input: {

            pulse: parseInt(this.bpm) },
        });

        if (pulseError) {
            console.log('Error creating pulse history:'+pulseError);
        }

        console.log('Pulse history created:', pulseData);
    }

    localStorage.removeItem('heartRateData');
    this.$router.push('/sign-in');
} catch (error) {
    console.log('Error in continueButton:'+error);
    this.$router.push('/sign-in');
}
},
computed: {

},
mounted() {

```

```

const heartRateData = localStorage.getItem('heartRateData');

if (!heartRateData) {
  this.$router.push('/dashboard');
  return;
}
try {
  const parsedData = JSON.parse(heartRateData);
  this.bpm = parsedData.bpm;
  this.date = parsedData.date;
} catch (error) {
  console.error('Error parsing heart rate data:', error);
  this.$router.push('/dashboard');
}
},
setup() {
  const { result: profileResult, error: profileError } = useQuery(GET_PROFILE);
  const { mutate: createProfileMutate, error: createProfileError } =
useMutation(CREATE_PROFILE);
  const { mutate: createPulseHistoryMutate, error: createPulseHistoryError } =
useMutation(CREATE_PULSE_HISTORY);

  return {
    getProfile: async () => ({ data: profileResult.value, error: profileError.value
  )),
    createProfile: async (variables) => {
      const { data, error } = await createProfileMutate(variables);
      return { data, error: error || createProfileError.value };
    },
    createPulseHistory: async (variables) => {
      const { data, error } = await createPulseHistoryMutate(variables);
      return { data, error: error || createPulseHistoryError.value };
    },
  };
},
};
</script>

```

Файл Dashboard.vue

Реалізація функціональних задач “Перегляд результатів”, “Вимірювання пульсу”, “Перегляд історії вимірювань”, “Перегляд аналізу історії вимірювань”

```

<template>
  <div class="dashboard">

```

```

<div class="ellipse ellipse-top"></div>

<header class="header">
  <h1>Dashboard</h1>
  <button @click="goToProfile">
    <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
      </svg>
    </button>
</header>

<section class="card pulse-section">
  <div v-if="flag">
    <div class="section-header">
      <h2>Your Pulse</h2>
      <span>{{ formattedDate }}</span>
      <button @click="pulseInfo">
        <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        </button>

      </div>
      <div class="pulse-data">
        <div class="bpm">
          <div class="pulse-value">{{ bpm }}</div>
          <div class="bpm-units">
            <svg class="heart-icon" width="14" height="11" viewBox="0 0
14 11" fill="none" xmlns="http://www.w3.org/2000/svg"></svg>
            <span class="unit">bpm</span>
          </div>
        </div>

        </div>
        <div class="graph-row">
          <svg width="186" height="61" viewBox="0 0 186 61" fill="none"
xmlns="http://www.w3.org/2000/svg">
            </svg>

          <div class="status">
            <p :class="`status-text ${bpmStatus.replace(/\s+/g, '-')}">{{ bpmStatus }}</p>
            <p class="hrv">
              HRV <span>{{ hrv }} ms</span>
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>

```

```

    </div>
  </div>
  <div class="metrics">
    <div class="metric">
      <p>Stress</p>
      <div class="progress-circle">
        <svg viewBox="0 0 36 36">
          </svg>
          <div class="progress-value">{{stress}}%</div>
        </div>
      </div>
      <div class="metric">
        <p>Tension</p>
        <div class="progress-circle">
          <svg viewBox="0 0 36 36">
            </svg>
            <div class="progress-value">36%</div>
          </div>
        </div>
      <div class="metric">
        <p>Energy</p>
        <div class="progress-circle">
          <svg viewBox="0 0 36 36">
            </svg>
            <div class="progress-value">79%</div>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div v-else class="new-day">
    <h3>How are you feeling?</h3>
    <div class="icons">
      <button @click="$router.push('/progress')">
        <svg class="heart-icon2" width="68" height="55" viewBox="0 0 68
55" fill="none" xmlns="http://www.w3.org/2000/svg">
          </svg>
        </button>
        <svg class="plus" viewBox="0 0 26 27" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        <svg class="wave-icon" viewBox="0 0 375 96" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
      </div>
    </div>
  </div>

```

```

    </div>
    <p>Tap to start now</p>
  </div>
</section>

<section class="last-measurement">
  <div class="section-header">
    <h2>Last measurement</h2>
    <button class="see-all" @click="this.$router.push('/history')">See
all</button>
  </div>
  <div class="measurement-cards">
    <div class="card heart-rate" @click="pulseInfo">
      <div class="card-header">
        <span>Heart<br>rate</span>
        <svg width="40" height="40" viewBox="0 0 40 40" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
      </div>
      <p>{{ bpm }} <span>bpm</span></p>
    </div>
    <div class="card blood-pressure">
      <div class="card-header">
        <span>Blood<br>pressure</span>
        <svg width="40" height="40" viewBox="0 0 40 40" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
      </div>
      <div v-if="lastMeasurement" class="measurement">
        <p>{{ lastMeasurement.systolic }}/{{ lastMeasurement.diastolic }}
<span>mmHg</span></p>
      </div>
      <div v-else >
        <button class="measure-btn" @click="goToMeasurement">
          <span>Add</span>
          <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
            </svg>
        </button>
      </div>
    </div>
  </div>
</section>

<div class="card track-health">

```

```

    <div class="new-health">
      <h2>Track Your Health</h2>
      <p>with My Heart Rate Monitor - Pulse</p>
      <button @click="goToMeasurement" class="measure-btn">
        <span>Measure</span>
        <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>

      </button>

    </div>
    
  </div>

  <PulseDiagram v-if="isSubscribed"/>

  <div class="overlay"></div>
</div>
</template>

<script>

import PulseDiagram from '@components/PulseDiagram.vue';
import Cookies from 'js-cookie';

export default {
  name: 'Dashboard',
  data() {
    return {
      bpm: 67,
      hrv: 68,
      stress: 0,
      tension: 36,
      energy: 79,
      bloodPressure: '130/80',
      flag: false,
      id: 0,
      summaries: undefined,
      formattedDate: "",
      date: "",
      pressureHistory: [],
    };
  },
  components: {

```

PulseDiagram,

```

},
methods: {
  pulseInfo() {
    localStorage.setItem('heartRateData', JSON.stringify({
      bpm: this.bpm,
      hrv: this.hrv,
      stress: this.stress,
      id: this.id,
      summaries: this.summaries || undefined,
      date: this.date
    }));
    this.$router.push(`/results/${this.id}`)
  },
  goToMeasurement() {
    this.$router.push('/measure/new');
  },
  goToProfile() {
    this.$router.push('/profile');
  },
},
computed: {
  imagePath() {
    const publicPath = process.env.BASE_URL || '/';
    return `${publicPath}images/woman.png`;
  },
  bpmStatus() {
    if (this.bpm < 60) return 'Low';
    if (this.bpm <= 100) return 'Normal';
    return 'High';
  },
  statusColor() {
    if (this.bpm < 60) return 'rgba(61, 187, 255, 1)';
    if (this.bpm <= 100) return 'rgba(177, 222, 53, 1)';
    return 'rgba(255, 72, 122, 1)';
  },
  lastMeasurement() {
    return this.pressureHistory.length > 0
      ? this.pressureHistory[this.pressureHistory.length - 1]
      : null;
  },
  isSubscribed() {
    return Cookies.get('ai-research') === 'subscribed';
  },
},

```

```

    },
    mounted() {
      const healthHistory = localStorage.getItem('healthHistory');
      this.pressureHistory = JSON.parse(localStorage.getItem('pressureHistory')) ||
    [];

    if (healthHistory) {
      try {
        const parsedData = JSON.parse(healthHistory);

        const latestEntry = parsedData.reduce((latest, current) => {
          return new Date(current.date).getTime() > new
Date(latest.date).getTime() ? current : latest;
        }, parsedData[0]);

        if (latestEntry && latestEntry.data) {
          this.bpm = Math.round(latestEntry.data.bpm) || this.bpm;
          this.hrv = Math.round(latestEntry.data.hrv) || this.hrv;
          this.stress = Math.round(latestEntry.data.stress) || this.stress;
          this.id = latestEntry.id || this.id;
          this.summaries = latestEntry.summaries || this.summaries;
          this.date = latestEntry.date;

          const today = new Date();
          const entryDate = new Date(latestEntry.date);
          this.flag = entryDate.getFullYear() === today.getFullYear() &&
            entryDate.getMonth() === today.getMonth() &&
            entryDate.getDate() === today.getDate();
        }

        const entryDate = new Date(latestEntry.date);
        const month = entryDate.toLocaleString('en-GB', { month: 'short'
}).toUpperCase();
        const day = String(entryDate.getDate()).padStart(2, '0');
        const hours = String(entryDate.getHours()).padStart(2, '0');
        const minutes = String(entryDate.getMinutes()).padStart(2, '0');
        this.formattedDate = `${month} ${day} ${hours}:${minutes}`;

      } catch (error) {
        console.error('Error parsing heart rate data:', error);
        this.$router.push('/');
      }
    }
  }

```

```

    }
  }
};
</script>

```

Файл Login.vue

Реалізація функціональних задач “Система авторизації користувач”, “Авторизація користувача”

```

<template>
  <div class="login">
    <div class="ellipse ellipse-top"></div>
    <div class="header">
      <button class="back-btn" @click="$router.go(-1)">
        <svg width="36" height="36" viewBox="0 0 36 36" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        </button>
      <h2>Log In</h2>
      <button class="back-btn">
        </button>
    </div>
    <div class="content">
      <div class="input-container">
        <input
          v-model="email"
          placeholder="Email"
          :class="{ 'invalid-input': errors.email }"
        />
        <span class="error-message" v-if="errors.email">
          <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
            </svg>
            {{ errors.email }}
          </span>
      </div>
      <div class="input-container">
        <input
          v-model="password"
          type="password"
          placeholder="Password (8+ characters)"
          :class="{ 'invalid-input': errors.password }"
        />
        <span class="error-message" v-if="errors.password">

```

```

    <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
    {{ errors.password }}</span>
</div>
<a>Forgot your password?</a>

```

```

</div>
<button
  class="continue-btn"
  :class="{ 'disabled-btn': isButtonDisabled }"
  :disabled="isButtonDisabled"
  @click="login"
>
  Continue
</button>

```

```

</div>
</template>

```

```

<script>
import { useAuth } from '@composables/useAuth';
export default {
  data() {
    return {
      password: "",
      email: "",
      errors: {
        password: "",
        email: "",
      },
    };
  },
  computed: {
    isButtonDisabled() {
      return !this.password.trim() || !this.email.trim();
    },
  },
  methods: {
    validateEmail(email) {
      const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      return re.test(email);
    },
  },
};

```

```

async login() {
  this.errors = {
    password: "",
    email: "",
  };

  let hasError = false;

  if (!this.email.trim()) {
    this.errors.email = 'Email is required';
    hasError = true;
  } else if (!this.validateEmail(this.email.trim())) {
    this.errors.email = 'Your email is not in the correct format';
    hasError = true;
  }

  if (!this.password.trim()) {
    this.errors.password = 'Password is required';
    hasError = true;
  }

  if (!hasError) {
    const { login } = useAuth();
    try {
      const result = await login(this.email.trim(), this.password);

      if (result.success) {
        localStorage.setItem('firebaseToken', result.token);
        this.$router.push('/dashboard');
      } else {
        switch (result.error) {
          case 'auth/user-not-found':
            this.errors.email = 'No account found with this email';
            break;
          case 'auth/wrong-password':
            this.errors.password = 'Incorrect password';
            break;
          case 'auth/invalid-email':
            this.errors.email = 'Invalid email format';
            break;
          default:
            this.errors.email = result.message || 'Login failed';
        }
      }
    }
  }
}

```

```

    } catch (error) {
      console.error('Login error:', error);
      this.errors.email = 'An unexpected error occurred';
    }
  },
},
};
};
</script>

```

Файл Progress.vue

Реалізація функціональних задач “Вимірювання пульсу”

```

<template>
  <div class="progress">
    <h5>Measuring</h5>
    <BpmCircle @update:show-camera="updateShowCamera" />

    <p v-if="showCamera">
      Try to cover the entire lens and hold the phone comfortably
    </p>
    <p v-else class="p2">
      {{ instructionText }}
    </p>

    
    <div v-else class="pulse-animation-wrapper">
      <svg
        class="pulse-svg"
        :class="{ 'pulsating': bpm > 0 }"
        width="375"
        viewBox="0 0 375 100"
        fill="none"
        xmlns="http://www.w3.org/2000/svg"
      >
        <path
          d="M2    52.1609H152.106L168.476    70.5287L189.213    3L211.586
97L233.414 52.1609H383"
          stroke-width="6"
          stroke-linecap="round"

```

```

        stroke-linejoin="round"
        class="pulse-path"
      />
    </svg>
  </div>

  <div v-if="showCamera" class="ellipse ellipse-bottom"></div>
  <div v-else class="ellipse ellipse-top"></div>
</div>
</template>

<script>
import BpmCircle from '../components/BpmCircle.vue';
import emitter from '@/utils/eventBus';

export default {
  name: 'Progress',
  data() {
    return {
      publicPath: process.env.BASE_URL,
      showCamera: true,
      elapsedTime: 0,
      timer: null,
      bpm: 0,
    };
  },
  components: {
    BpmCircle,
  },
  computed: {
    instructionText() {
      if (this.elapsedTime < 9) {
        return 'Hold the phone in a comfortable position and avoid trying to control your breathing';
      } else if (this.elapsedTime < 18) {
        return 'Measuring your heart rate';
      } else {
        return 'Studies indicate that HRV offers valuable insights into your body's health';
      }
    },
    pulseDuration() {
      return this.bpm > 0 ? 60 / this.bpm : 1;
    },
  },
},

```

```

methods: {
  updateShowCamera(value) {
    this.showCamera = value;
    if (!value) {
      this.startTextTimer();
    } else {
      this.stopTextTimer();
      this.elapsedTime = 0;
    }
  },
  startTextTimer() {
    this.stopTextTimer();
    this.elapsedTime = 0;
    this.timer = setInterval(() => {
      this.elapsedTime += 1;
      if (this.elapsedTime >= 20) {
        this.elapsedTime = 0;
      }
    }, 1000);
  },
  stopTextTimer() {
    if (this.timer) {
      clearInterval(this.timer);
      this.timer = null;
    }
  },
},
mounted() {
  emitter.on('bpm-update', (bpm) => {
    this.bpm = bpm;
  });
},
beforeUnmount() {
  this.stopTextTimer();
  emitter.off('bpm-update');
},
};
</script>

```

Файл **PersonalData.vue**

Реалізація функціональних задач “Введення персональних даних”

```

<template>
  <div class="personal-data">

```

```

<h1>Make it personal!</h1>
<div class="heart-container">
  <svg      viewBox="0      0      375      156"      fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
</div>

```

```

<div class="form-container">
  <div class="unit-toggle">
    <button
      :class="{ active: unit === 'imperial' }"
      @click="unit = 'imperial'"
    >
      lb & inch
    </button>
    <button
      :class="{ active: unit === 'metric' }"
      @click="unit = 'metric'"
    >
      kg & cm
    </button>
  </div>

```

```

<div class="gender-selection">
  <button
    v-for="gender in genders"
    :key="gender"
    :class="{ selected: selectedGender === gender }"
    @click="selectedGender = gender"
  >
    {{ gender }}
  </button>
</div>

```

```

<div class="two-sliders">
  <div class="slider-container">
    <label>Age</label>
    <div class="slider-value">{{ age }}</div>
    <input
      type="range"
      v-model="age"
      min="16"
      max="100"
      step="1"
    >

```

```

    <div class="unit">years</div>
  </div>

```

```

<div class="slider-container">
  <label>Weight</label>
  <div class="slider-value"> {{ weight }}</div>
  <input
    type="range"
    v-model="weight"
    :min="unit === 'metric' ? 0 : 0"
    :max="unit === 'metric' ? 300 : 600"
    step="1"
  >
  <div class="unit">{{ unit === 'metric' ? 'kilograms' : 'pounds' }}</div>
</div>
</div>

```

```

<div class="slider-container one-slider">
  <label>What is your height?</label>
  <div class="slider-value">{{ height }}</div>
  <input
    type="range"
    v-model="height"
    :min="unit === 'metric' ? 0 : 0"
    :max="unit === 'metric' ? 300 : 100"
    step="1"
  >
  <div class="unit">{{ unit === 'metric' ? 'centimeters' : 'inches' }}</div>
</div>

```

```

    <button @click="submitForm" >Save</button>
  </div>

```

```

<div class="ellipse ellipse-top"></div>

```

```

</div>
</template>

```

```

<script>

```

```

export default {
  name: 'PersonalData',
  data() {

```

```

return {
  publicPath: process.env.BASE_URL,
  unit: 'metric',
  genders: ['Male', 'Female', 'Other'],
  selectedGender: 'Female',
  age: 45,
  weight: 60,
  height: 170,
  assessment: null,
  heartRateData: null,
};
},
mounted() {
  const data = localStorage.getItem('heartRateData');
  if (data) {
    this.heartRateData = JSON.parse(data);
  }
},
methods: {
  async submitForm() {
    this.assessment = null;

    const personData = {gender: this.selectedGender, age: this.age, weight:
this.weight, height: this.height};
    if(this.personData) {
      localStorage.setItem('health', JSON.stringify(personData));
    }

    this.$router.push('/results');
  },
},
};
</script>

```

Файл Home.vue

Реалізація функціональних задач “Перегляд сторінки привітання”

```

<template>
  <div class="home">
    <div v-if="showLoading" class="loading-screen">
      <h1>Welcome to <br>My Heart Rate Monitor - Pulse</h1>
      <div class="heartbeat-line" :style="{ backgroundImage:
'url(${publicPath}images/Layer_1.png)'" loading="lazy"></div>
      <div class="ellipse"></div>
    </div>
  </div>

```

```

    <div class="ellipse2"></div>

</div>

<div v-else class="message-screen" >
  <div class="splash" :style="{ backgroundImage: `url(${currentBackground})`
}"></div>
  <div class="message-box">
    <div class="bg-figure">
      <svg      viewBox="0      0      375      375"      fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
      </div>
      <div class="pulse-line">
        <svg      viewBox="0      0      375      100"      fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>

      </div>
      <p>{{ currentMessage }}</p>
      <button @click="nextMessage" class="continue-btn">Continue</button>
    </div>
  </div>
</div>

</template>

```

```

<script>
export default {
  name: 'Home',
  data() {
    return {
      publicPath: process.env.BASE_URL,
      showLoading: true,
      currentStep: 0,
      messages: [
        'Measure your pulse using only the camera',
        'Heart health diary in your phone',
        'Statistics, insights and much more!'
      ],
      backgrounds: [
        `/images/splash1.png`,

```

```

        `/images/splash2.png`,
        `/images/splash3.png`,
    ]
    };
},
computed: {
    currentMessage() {
        return this.messages[this.currentStep];
    },
    currentBackground() {
        return this.backgrounds[this.currentStep];
    }
},
methods: {
    nextMessage() {
        this.currentStep++;
        if (this.currentStep >= this.messages.length) {
            this.$router.push('/progress');
        }
    }
},
mounted() {
    setTimeout(() => {
        this.showLoading = false;
    }, 3000);
}
};
</script>

```

Файл Register.vue

Реалізація функціональних задач “Система авторизації користувача”, “Реєстрація користувача”

```

<template>
  <div class="register">
    <div class="ellipse ellipse-top"></div>
    <div class="header">
      <button class="back-btn" @click="$router.go(-1)">
        <svg width="36" height="36" viewBox="0 0 36 36" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        </button>
      <h2>Create your account</h2>
      <button class="back-btn">
        </button>
    </div>
  </div>
</template>

```

```

</div>
<div class="content">
  <div class="input-container">
    <input
      v-model="name"
      placeholder="Name"
      :class="{ 'invalid-input': errors.name }"
    />
    <span class="error-message" v-if="errors.name">
      <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
        {{ errors.name }}
      </span>
    </div>
    <div class="input-container">
      <input
        v-model="email"
        placeholder="Email"
        :class="{ 'invalid-input': errors.email }"
      />
      <span class="error-message" v-if="errors.email">
        <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
          {{ errors.email }}
        </span>
      </div>
      <div class="input-container">
        <input
          v-model="password"
          type="password"
          placeholder="Password (8+ characters)"
          :class="{ 'invalid-input': errors.password }"
        />
        <span class="error-message" v-if="errors.password">
          <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
            </svg>
            {{ errors.password }}</span>
        </div>
        <div class="input-container">
          <input
            v-model="repeatPassword"
            type="password"

```

```

        placeholder="Repeat password"
        :class="{ 'invalid-input': errors.repeatPassword }"
      />
      <span class="error-message" v-if="errors.repeatPassword">
        <svg width="14" height="14" viewBox="0 0 14 14" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
          {{ errors.repeatPassword }}</span>
    </div>

  </div>
  <button
    class="continue-btn"
    :class="{ 'disabled-btn': isButtonDisabled }"
    :disabled="isButtonDisabled"
    @click="register"
  >
    Continue
  </button>

</div>
</template>

<script>
import { useAuth } from '@composables/useAuth';

export default {
  data() {
    return {
      name: "",
      email: "",
      password: "",
      repeatPassword: "",
      errors: {
        name: "",
        email: "",
        password: "",
        repeatPassword: "",
      },
    };
  },
  computed: {
    isButtonDisabled() {

```

```

        return !this.name.trim() || !this.email.trim() || !this.password.trim() ||
!this.repeatPassword.trim();
    },
    },
    methods: {
        validateEmail(email) {
            const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            return re.test(email);
        },
        async register() {
            this.errors = {
                name: "",
                email: "",
                password: "",
                repeatPassword: "",
            };

            let hasError = false;

            if (!this.name.trim()) {
                this.errors.name = 'Name is required';
                hasError = true;
            }

            if (!this.validateEmail(this.email.trim())) {
                this.errors.email = 'Your email is not in the correct format';
                hasError = true;
            }

            if (this.password.length < 8) {
                this.errors.password = 'Please enter 8+ characters password';
                hasError = true;
            }

            if (this.password !== this.repeatPassword) {
                this.errors.password = 'Passwords are not the same';
                this.errors.repeatPassword = 'Passwords are not the same';
                hasError = true;
            }

            if (!hasError) {
                const { register } = useAuth();
                try {
                    const result = await register(this.email.trim(), this.password,
this.name.trim());

```

```

    if (result.success) {
      localStorage.setItem('firebaseToken', result.token);
      this.$router.push('/dashboard');
    } else {
      switch (result.error) {
        case 'auth/email-already-in-use':
          this.errors.email = 'This email is already registered';
          break;
        case 'auth/invalid-email':
          this.errors.email = 'Invalid email format';
          break;
        case 'auth/weak-password':
          this.errors.password = 'Password is too weak';
          break;
        default:
          this.errors.email = result.message || 'Registration failed';
      }
    }
  } catch (error) {
    console.error('Registration error:', error);
    this.errors.email = 'An unexpected error occurred';
  }
},
},
};
</script>

```

Файл Measurement.vue

Реалізація функціональних задач “Перегляд результатів”: пульсу, варіабельності та стресу; Аналіз результатів вимірювання (Groq API), Аналіз результатів вимірювання (OpenAI API).

```

<template>
  <div class="results">
    <div class="ellipse ellipse-top"></div>
    <div class="header">
      <button class="back-btn" @click="$router.go(-1)">
        <svg width="36" height="36" viewBox="0 0 36 36" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        </button>
      <h1>{{ date }}</h1>
    </div>
  </Metrics/>

```

```

</div>
</template>

<script>
import Metrics from '@components/Metrics.vue';

export default {
  data() {
    return {
      bpm: 0,
      date: ''
    };
  },
  components: {
    Metrics,
  },
  methods: {
    continueButton() {
      localStorage.removeItem('heartRateData');
      this.$router.push('/dashboard');
    }
  },
  computed: {
  },
  mounted() {
    const heartRateData = localStorage.getItem('heartRateData');

    if (!heartRateData) {
      this.$router.push('/dashboard');
      return;
    }
    try {
      const parsedData = JSON.parse(heartRateData);
      this.bpm = parsedData.bpm;
      const timestamp = parsedData.date;
      const dateObj = new Date(parseInt(timestamp));
      this.date = new Intl.DateTimeFormat('en-GB', {
        day: 'numeric',
        month: 'long',
        year: 'numeric',
        hour: '2-digit',
        minute: '2-digit',
        hour12: false,
      })
    }
  }
}

```

```

        .format(dateObj)
        .replace(',', ' at');
    } catch (error) {
        console.error('Error parsing heart rate data:', error);
        this.$router.push('/dashboard');
    }
}
};
</script>

```

Файл Profile.vue

Реалізація функціональних задач “ Вихід із акаунту”

```

<template>
<div class="profile">
  <div class="ellipse ellipse-top"></div>
  <div class="header">
    <button class="back-btn" @click="$router.go(-1)">
      <svg width="36" height="36" viewBox="0 0 36 36" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
      </button>
    <h1>Profile</h1>
  </div>
  <div class="p-content">
    <div class="p-container">
      <div class="p-element gray">
        <span>Personal Info</span>
        <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
          <path d="M7.5 15L12.5 10L7.5 5" stroke="#818898" stroke-
width="1.66667" stroke-linecap="round" stroke-linejoin="round"/>
        </svg>
      </div>
      <div class="p-element gray">
        <span>Manage Notifcation</span>
        <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
          <path d="M7.5 15L12.5 10L7.5 5" stroke="#818898" stroke-
width="1.66667" stroke-linecap="round" stroke-linejoin="round"/>
        </svg>
      </div>
      <div class="p-element gray">
        <span>Active Apple Health</span>

```

```

    <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
    <path d="M7.5 15L12.5 10L7.5 5" stroke="#818898" stroke-
width="1.66667" stroke-linecap="round" stroke-linejoin="round"/>
    </svg>
  </div>
</div>
<div class="p-container">
  <div class="p-element gray">
    <span>Full PDF report</span>
    <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
  </div>
</div>
<div class="p-container">
  <div class="p-element gray">
    <span>Language</span>
    <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
  </div>
  <div class="p-element gray">
    <span>Privacy Policy</span>
    <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
  </div>
  <div class="p-element gray">
    <span>Terms of Use</span>
    <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
  </div>
  <div class="p-element gray">
    <span>Support & Feedback</span>
    <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
  </div>
</div>
<div class="p-container">
  <div class="p-element" @click="handleLogout">
    <span>Log Out</span>

```

```

    <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
    </svg>
  </div>
</div>
</div>

```

```

</div>
</template>

```

```

<script>
import { useAuth } from '@composables/useAuth';

export default {
  data() {
    return {
    };
  },
  methods: {
    async handleLogout() {
      const { logout } = useAuth();
      await logout();
    }
  },
};
</script>

```

Файл History.vue

Реалізація функціональних задач “Перегляд історії вимірювань”

```

<template>
  <div class="history-page">
    <div class="ellipse ellipse-top"></div>
    <!-- Header -->

    <div class="header">
      <button class="back-btn" @click="$router.go(-1)">
        <svg width="36" height="36" viewBox="0 0 36 36" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
      </button>
      <h1>Your measurements</h1>
      <button class="add-btn">

```

```

<svg width="36" height="36" viewBox="0 0 36 36" fill="none"
xmlns="http://www.w3.org/2000/svg">
  </svg>
</button>
</div>

<!-- Tabs -->
<div class="tabs">
  <button
    v-for="(tab, index) in tabs"
    :key="index"
    class="tab"
    :class="{ active: activeTab === index }"
    @click="setActiveTab(index)"
  >
    {{ tab }}
  </button>
</div>
<AIResearch/>
<!-- Health Records -->
<div v-for="(group, date) in groupedRecords" :key="date" class="record-
section">

  <div class="selected-date">
    {{ formatDate(date) }}
  </div>
  <div v-for="(record, index) in group" :key="record.id" class="record">
    <div class="record-header">
      <span>{{ formatTime(record.date) }}</span>
    </div>
    <div class="record-data">
      <svg class="line" viewBox="0 0 289 1" fill="none"
xmlns="http://www.w3.org/2000/svg">
        <line y1="0.5" x2="289" y2="0.5" stroke="white"/>
      </svg>
      <div class="record-content" :ref="'recordContent' + record.id"
@touchstart="onTouchStart($event, record.id)"
@touchmove="onTouchMove($event, record.id)"
@touchend="onTouchEnd($event, record.date + index)" >

        <button
          v-if="record.type === 'pressure'"
          @click="pressureInfo(record)"
          class="icon"
        >

```

```

        <svg width="100%" height="100%" viewBox="0 0 48 48" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
    </button>

    <button
        v-if="record.type === 'pulse'"
        @click="pulseInfo(record)"
        class="icon"
    >
        <svg width="100%" height="100%" viewBox="0 0 48 48" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
    </button>

    <div class="data">
        <div class="data-element" v-if="record.type === 'pulse'">
            {{ record.data.bpm }} bpm <br /><span>Pulse</span>
        </div>
        <div class="data-element" v-if="record.type === 'pulse'">
            {{ record.data.hrv }} ms <br /><span>HRV</span>
        </div>
        <div class="data-element" v-if="record.type === 'pulse'">
            {{ record.data.stress }} % <br /><span>Stress</span>
        </div>
        <div class="data-element" v-if="record.type === 'pressure'">
            {{ record.systolic }} / {{ record.diastolic }} <br /><span>Blood
Pressure</span>
        </div>
        <div class="data-element" v-if="record.type === 'pressure'">
            {{ record.category }}
        </div>
    </div>
    <button @click="deleteRecord(record.date)" class="delete-btn"
:ref="'deleteBtn' + record.id">
        <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>

    </button>
</div>

</div>
</div>
</div>

```

```
</div>
</template>
```

```
<script>
import { PULSE_HISTORY_QUERY } from '@/utils/database';
import { useQuery } from '@vue/apollo-composable';
import { ref } from 'vue';
import AIResearch from '@components/AIResearch.vue';

export default {
  name: 'History',
  data() {
    return {
      pressureHistory: [],
      healthHistory: [],
      tabs: ['View all', 'Heart rate', 'Blood pressure'],
      activeTab: 0,
      touchState: {}
    };
  },
  components: {
    AIResearch,
  },
  computed: {
    groupedRecords() {
      let filteredRecords = [...this.healthHistory, ...this.pressureHistory];

      if (this.activeTab === 1) {
        filteredRecords = filteredRecords.filter(record => record.type === 'pulse');
      } else if (this.activeTab === 2) {
        filteredRecords = filteredRecords.filter(record => record.type === 'pressure');
      }
      const sortedRecords = filteredRecords.sort((a, b) => b.date - a.date);

      const grouped = {};
      sortedRecords.forEach(record => {
        const recordDate = new Date(record.date);
        const dateKey = recordDate.toLocaleDateString('en-US', {
          weekday: 'long',
          day: 'numeric',
          month: 'long',
          year: 'numeric'
        });
      });
    }
  }
};
</script>
```

```

    if (!grouped[dateKey]) {
      grouped[dateKey] = [];
    }
    grouped[dateKey].push(record);
  });

  Object.keys(grouped).forEach(dateKey => {
    grouped[dateKey].sort((a, b) => a.date - b.date);
  });

  return grouped;
},
},
methods: {
  formatDate(dateString) {
    return dateString;
  },
  formatTime(timestamp) {
    const date = new Date(timestamp);
    return date.toLocaleString('en-GB', {
      hour: '2-digit',
      minute: '2-digit',
    })
    .toLowerCase()
    .replace(':', '.');
  },
  async loadHealthHistory() {
    try {
      const { data, error } = await this.getHistory({ limit: 100, offset: 0 });
      if (data && data.pulseHistory && data.pulseHistory.items) {
        this.healthHistory = data.pulseHistory.items.map((item) => ({
          id: item.id,
          date: item.createdAt,
          type: 'pulse',
          data: {
            bpm: item.pulse,
          },
        }));
        console.log('Fetched history:', this.healthHistory);
      }
      if (error) {
        console.error('Error with history:', error);
      }
    } catch (error) {
      console.error("Error with request" + error);
    }
  }
}

```

```

    }

    let healthHistory = localStorage.getItem('healthHistory');
    healthHistory = healthHistory ? JSON.parse(healthHistory) : [];

    if (!Array.isArray(healthHistory)) {
        healthHistory = [];
    }
    this.healthHistory = healthHistory;
    this.healthHistory.forEach(record => {
        record.type = 'pulse';
    });
},
loadPressureHistory() {
    this.pressureHistory = JSON.parse(localStorage.getItem('pressureHistory')) ||
[];
    this.pressureHistory.forEach(record => {
        if (record.systolic < 90 || record.diastolic < 60) record.category =
'Hypotension';
        else if (record.systolic < 120 && record.diastolic < 80) record.category =
'Normal';
        else if (record.systolic < 130 && record.diastolic < 80) record.category =
'Elevated';
        else if (record.systolic < 140 || (record.systolic < 180 && record.diastolic <
90)) record.category = 'Stage 1 Hypertension';
        else if (record.systolic < 180 || (record.systolic < 180 && record.diastolic <
120)) record.category = 'Stage 2 Hypertension';
        else record.category = 'Hypertensive Crisis';
        record.type = 'pressure';
    });
},
setActiveTab(index) {
    this.activeTab = index;
},
pulseInfo(record) {
    localStorage.setItem('heartRateData', JSON.stringify({
        id: record.id,
        bpm: record.data.bpm,
        hrv: record.data.hrv,
        stress: record.data.stress,
        date: record.date,
        summaries: record.summaries || undefined
    }));
    this.$router.push(`/results/${record.id}`)
},

```

```

pressureInfo(record) {
  localStorage.setItem('pressureData', JSON.stringify({
    id: record.id,
    systolic: record.systolic,
    diastolic: record.diastolic,
    category: record.category,
    date: record.date,
  }));
},
onTouchStart(event, index) {
  const touch = event.touches[0];
  this.touchState[index] = {
    startX: touch.clientX,
    currentX: touch.clientX,
    translateX: 0,
    isSwiping: false
  };
},
onTouchMove(event, index) {
  if (!this.touchState[index]) return;
  const touch = event.touches[0];
  const deltaX = touch.clientX - this.touchState[index].startX;
  this.touchState[index].currentX = touch.clientX;
  this.touchState[index].isSwiping = true;

  const translateX = Math.max(-80, Math.min(0, deltaX));
  this.touchState[index].translateX = translateX;

  const recordContent = this.$refs['recordContent' + index][0];
  recordContent.style.transform = `translateX(${translateX}px)`;

  const deleteBtn = this.$refs['deleteBtn' + index][0];
  const btnWidth = Math.min(80, Math.abs(translateX));
  deleteBtn.style.width = `${btnWidth}px`;
},
onTouchEnd(event, index) {
  if (!this.touchState[index] || !this.touchState[index].isSwiping) return;

  const translateX = this.touchState[index].translateX;
  const recordContent = this.$refs.recordContent[index];
  const deleteBtn = this.$refs.deleteBtn[index];

  if (translateX < -50) {
    recordContent.style.transform = 'translateX(-80px)';
  }
}

```

```

        deleteBtn.style.width = '80px';
      } else {
        recordContent.style.transform = 'translateX(0px)';
        deleteBtn.style.width = '0px';
      }

      this.touchState[index].isSwiping = false;
    },
    deleteRecord(recordDate) {
      let healthHistory = JSON.parse(localStorage.getItem('healthHistory')) || [];
      healthHistory = healthHistory.filter(record => record.date !== recordDate);
      localStorage.setItem('healthHistory', JSON.stringify(healthHistory));
      this.loadHealthHistory();
    }
  },
  mounted() {
    this.loadHealthHistory();
    this.loadPressureHistory();
  },
  setup() {
    const variables = ref({ limit: 100, offset: 0 });
    const { result: pulseResult, error: pulseError, refetch } =
useQuery(PULSE_HISTORY_QUERY);
    return {
      getHistory: async (vars) => {
        try {
          variables.value = { ...variables.value, ...vars };
          await refetch(variables.value);
          return {
            data: pulseResult.value,
            error: pulseError.value,
          };
        } catch (error) {
          return { data: null, error };
        }
      },
    };
  }
};
</script>

```

Файл **SignIn.vue**

Реалізація функціональних задач “Система авторизації користувача”, “
 Реєстрація користувача”, “ Авторизація користувача ”

```

<template>
  <div class="sign-in">
    <div class="ellipse ellipse-top"></div>
    <div class="header">
      <button class="back-btn" @click="$router.go(-1)">
        <svg width="36" height="36" viewBox="0 0 36 36" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        </button>

      <button class="skip-btn" @click="goDashboard">
        Skip
      </button>
    </div>
    <div class="pulse">
      <svg width="233" height="139" viewBox="0 0 233 139" fill="none"
xmlns="http://www.w3.org/2000/svg">
        </svg>
      </div>
    <div class="content">
      <h1>Create an Account <br> My Heart Rate Monitor - Pulse</h1>
      <p>Save your data and receive personalized health recommendations</p>
      <button class="service-login">
        <svg viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        <span>Continue with Google</span>
      </button>
      <button class="service-login">
        <svg viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
          </svg>
        <span>Continue with Apple</span>
      </button>
      <span>or</span>
      <button class="register-btn" @click="goToRegister">Create Account</button>
      <span>By proceeding, you agree to our <a>Terms of Service</a> and
<a>Privacy Policy</a></span>
      <p>Already have an account? <a @click="goToLogin">Log In</a></p>
    </div>

  </div>
</template>

<script>

```

```

export default {
  data() {
    return {
      name: "",
      email: "",
      password: "",
    };
  },
  methods: {
    goToRegister() {
      this.$router.push('/register');
    },
    goToLogin() {
      this.$router.push('/login');
    },
    goDashboard() {
      this.$router.push('/dashboard');
    }
  },
};
</script>

```

Файл heartRateLogic.js

Реалізація функціональних задач “Вимірювання пульсу”

```

import { v4 as uuidv4 } from 'uuid';

export const heartRateMonitor = (function () {
  const IMAGE_WIDTH = 300;
  const IMAGE_HEIGHT = 300;
  const FPS = 30;
  const SAMPLE_BUFFER = [];
  const VALID_FRAME_THRESHOLD = 60;
  const PULSE_THRESHOLD = -60;
  const MAX_PERIODS = 20;
  const AVERAGE_SIZE = 20;
  const MIN_PERIOD = 0.4;
  const MAX_PERIOD = 1.5;
  const INVALID_ENTRY = -100;

  let VIDEO_ELEMENT;
  let SAMPLING_CANVAS;
  let SAMPLING_CONTEXT;
  let ON_BPM_CHANGE;
  let ON_STATUS_CHANGE;

```

```

let VIDEO_STREAM;
let MONITORING = false;
let VALID_FRAME_COUNTER = 0;
let MEASUREMENT_STARTED = false;
let PULSE_DETECTOR;
let HUE_FILTER;
let TIMER;
let LAST_VALID_BPM = 0;

const publicMethods = {};

class PulseDetector {
  constructor() {
    this.upVals = new Array(AVERAGE_SIZE).fill(INVALID_ENTRY);
    this.downVals = new Array(AVERAGE_SIZE).fill(INVALID_ENTRY);
    this.upValIndex = 0;
    this.downValIndex = 0;
    this.lastVal = 0;
    this.periods = new Array(MAX_PERIODS).fill(INVALID_ENTRY);
    this.periodTimes = new Array(MAX_PERIODS).fill(INVALID_ENTRY);
    this.periodIndex = 0;
    this.wasDown = false;
    this.periodStart = 0;
  }

  addNewValue(newVal, time) {
    if (newVal > 0) {
      this.upVals[this.upValIndex] = newVal;
      this.upValIndex = (this.upValIndex + 1) % AVERAGE_SIZE;
    }
    if (newVal < 0) {
      this.downVals[this.downValIndex] = -newVal;
      this.downValIndex = (this.downValIndex + 1) % AVERAGE_SIZE;
    }

    const averageUp =
      this.upVals.filter((v) => v !== INVALID_ENTRY).reduce((sum, v) => sum +
v, 0) /
      this.upVals.filter((v) => v !== INVALID_ENTRY).length || 1;
    const averageDown =
      this.downVals.filter((v) => v !== INVALID_ENTRY).reduce((sum, v) => sum
+ v, 0) /
      this.downVals.filter((v) => v !== INVALID_ENTRY).length || 1;

    if (newVal < -0.5 * averageDown) {

```

```

    this.wasDown = true;
  }
  if (newVal >= 0.5 * averageUp && this.wasDown) {
    this.wasDown = false;
    const period = time - this.periodStart;
    if (period < MAX_PERIOD && period > MIN_PERIOD) {
      this.periods[this.periodIndex] = period;
      this.periodTimes[this.periodIndex] = time;
      this.periodIndex = (this.periodIndex + 1) % MAX_PERIODS;
    }
    this.periodStart = time;
  }

  return newVal < -0.5 * averageDown ? -1 : newVal > 0.5 * averageUp ? 1 : 0;
}

getAverage() {
  const now = performance.now() / 1000;
  const validPeriods = this.periods
    .map((p, i) => ({ period: p, time: this.periodTimes[i] }))
    .filter((p) => p.period !== INVALID_ENTRY && now - p.time < 10);
  if (validPeriods.length > 1) {
    const recentPeriods = validPeriods.slice(-5);
    const avgPeriod =
      recentPeriods.reduce((sum, p) => sum + p.period, 0) / recentPeriods.length;
    return avgPeriod;
  }
  return PULSE_THRESHOLD;
}

reset() {
  this.upVals.fill(INVALID_ENTRY);
  this.downVals.fill(INVALID_ENTRY);
  this.periods.fill(INVALID_ENTRY);
  this.periodTimes.fill(INVALID_ENTRY);
  this.upValIndex = 0;
  this.downValIndex = 0;
  this.periodIndex = 0;
  this.wasDown = false;
  this.periodStart = 0;
}

getPeriods() {
  return this.periods.filter((p) => p !== INVALID_ENTRY);
}

```

```
}
```

```
class BandPassFilter {
  constructor() {
    this.lastValue = 0;
    this.lastFiltered = 0;
    this.alpha = 0.1;
    this.beta = 0.1;
  }

  processValue(value) {
    const highPass = value - this.lastValue;
    this.lastValue = value;
    const lowPass = this.alpha * highPass + (1 - this.alpha) * this.lastFiltered;
    this.lastFiltered = lowPass;
    return lowPass;
  }
}
```

```
function rgb2hsv({ red, green, blue }) {
  const r = red / 255;
  const g = green / 255;
  const b = blue / 255;
  const max = Math.max(r, g, b);
  const min = Math.min(r, g, b);
  const delta = max - min;
  let h = 0;
  if (delta !== 0) {
    if (max === r) h = ((g - b) / delta) % 6;
    else if (max === g) h = (b - r) / delta + 2;
    else h = (r - g) / delta + 4;
    h *= 60;
    if (h < 0) h += 360;
  }
  const s = max === 0 ? 0 : delta / max;
  const v = max;
  return { h, s, v };
}
```

```
publicMethods.initialize = (config) => {
  VIDEO_ELEMENT = config.videoElement;
  SAMPLING_CANVAS = config.samplingCanvas;
  ON_BPM_CHANGE = config.onBpmChange;
  ON_STATUS_CHANGE = config.onStatusChange;
```

```

    SAMPLING_CONTEXT    =    SAMPLING_CANVAS.getContext('2d',    {
willReadFrequently: true });
    PULSE_DETECTOR = new PulseDetector();
    HUE_FILTER = new BandPassFilter();

    if (!navigator.mediaDevices) {
        console.error('Camera access is required but not supported. ');
        return false;
    }
};

publicMethods.startMonitoring = async () => {
    VALID_FRAME_COUNTER = 0;
    MEASUREMENT_STARTED = false;
    PULSE_DETECTOR.reset();
    SAMPLE_BUFFER.length = 0;
    LAST_VALID_BPM = 0;

    VIDEO_STREAM = await startCameraStream();

    if (!VIDEO_STREAM) {
        throw new Error('Unable to start video stream');
    }

    try {
        await setTorchStatus(VIDEO_STREAM, true);
    } catch (e) {
        console.warn('Torch setup failed:', e);
    }

    SAMPLING_CANVAS.width = IMAGE_WIDTH;
    SAMPLING_CANVAS.height = IMAGE_HEIGHT;
    VIDEO_ELEMENT.srcObject = VIDEO_STREAM;
    VIDEO_ELEMENT.play();
    MONITORING = true;

    ON_BPM_CHANGE(0);
    monitorLoop();
};

publicMethods.stopMonitoring = async () => {
    try {
        if (VIDEO_STREAM) {
            VIDEO_STREAM.getVideoTracks().forEach((track) => track.stop());
            VIDEO_STREAM = null;
        }
    }
};

```

```

    }
    await setTorchStatus(VIDEO_STREAM, false);
    if (VIDEO_ELEMENT) {
      VIDEO_ELEMENT.pause();
      VIDEO_ELEMENT.srcObject = null;
    }
    MONITORING = false;
    if (TIMER) {
      clearInterval(TIMER);
      TIMER = null;
    }
    const periods = PULSE_DETECTOR.getPeriods();
    let intervalsMs = periods.map(seconds => seconds * 1000);

    let hrv = 0;
    if (intervalsMs.length > 4) {
      intervalsMs = intervalsMs.slice(3);
    }

    if (intervalsMs.length >= 2) {
      const successiveDifferences = intervalsMs.slice(1).map((val, i) =>
Math.abs(val - intervalsMs[i]));
      const squaredDifferences = successiveDifferences.map(val => val * val);

      if (squaredDifferences.length === 0) {
        hrv = 0;
      } else {
        const meanOfSquares = squaredDifferences.reduce((sum, val) => sum +
val, 0) / squaredDifferences.length;
        hrv = Math.sqrt(meanOfSquares);
      }
    }

    let stress = 0;
    if (hrv > 0) {
      stress = Math.max(0, Math.min(100, 100 - hrv * 1.5));
    }

    const id = uuidv4();
    localStorage.setItem('heartRateData', JSON.stringify({
      id,
      bpm: LAST_VALID_BPM,
      hrv: parseInt(hrv),
      stress: parseInt(stress),
    }));
  });

```

```

let healthHistory = localStorage.getItem('healthHistory');
healthHistory = healthHistory ? JSON.parse(healthHistory) : [];

if (!Array.isArray(healthHistory)) {
  healthHistory = [];
}
healthHistory.push({
  id,
  date: Date.now(),
  data: {
    bpm: LAST_VALID_BPM,
    hrv: parseInt(hrv),
    stress: parseInt(stress)
  }
});

localStorage.setItem('healthHistory', JSON.stringify(healthHistory));
ON_BPM_CHANGE(LAST_VALID_BPM);
ON_STATUS_CHANGE('Cover the back camera until the image turns red ');
} catch (error) {
  console.error('Error stopping monitoring:', error);
}
};

const getCamera = async () => {
  try {
    const initialStream = await navigator.mediaDevices.getUserMedia({
      video: {
        facingMode: 'environment'
      }
    });
  });

  const devices = await navigator.mediaDevices.enumerateDevices();
  const cameras = devices.filter((device) => device.kind === 'videoinput' &&
device.deviceId !== "");
  initialStream.getTracks().forEach(track => track.stop());
  return cameras[cameras.length - 2] || cameras[0] || null;
} catch (error) {
  console.error('Failed to access camera:', error.message);
  return null;
}
};

const startCameraStream = async () => {

```

```

try {
  const camera = await getCamera();
  if (!camera || !camera.deviceId) {
    throw new Error('No valid camera found');
  }

  const stream = await navigator.mediaDevices.getUserMedia({
    video: {
      deviceId: { exact: camera.deviceId },
      facingMode: 'environment',
      width: { ideal: IMAGE_WIDTH },
      height: { ideal: IMAGE_HEIGHT },
      frameRate: { ideal: FPS },
    },
  });
  return stream;
} catch (error) {
  console.error('Failed to access camera:', error.message);
  return null;
}
};

const setTorchStatus = async (stream, status) => {
  try {
    if (!stream) return;
    const track = stream.getVideoTracks()[0];
    const capabilities = track.getCapabilities();
    if ('torch' in capabilities) {
      await track.applyConstraints({
        advanced: [{ torch: status }],
      });
    } else {
      console.warn('Torch is not supported on this device.');
```

```

const processFrame = () => {
  SAMPLING_CONTEXT.drawImage(VIDEO_ELEMENT, 0, 0,
    IMAGE_WIDTH, IMAGE_HEIGHT);
  const pixelData = SAMPLING_CONTEXT.getImageData(0, 0,
    IMAGE_WIDTH, IMAGE_HEIGHT).data;
  let redSum = 0,
    greenSum = 0,
    blueSum = 0;
  for (let i = 0; i < pixelData.length; i += 4) {
    redSum += pixelData[i];
    greenSum += pixelData[i + 1];
    blueSum += pixelData[i + 2];
  }
  const pixelCount = pixelData.length / 4;
  const rgb = {
    red: redSum / pixelCount,
    green: greenSum / pixelCount,
    blue: blueSum / pixelCount,
  };
  const hsv = rgb2hsv(rgb);

  if (hsv.s > 0.4 && hsv.v > 0.4) {
    VALID_FRAME_COUNTER++;
    ON_STATUS_CHANGE('Hold your index finger 🙌 still.');
```

```

    if (!MEASUREMENT_STARTED) {
      startMeasurement();
      MEASUREMENT_STARTED = true;
    }
    if (VALID_FRAME_COUNTER > VALID_FRAME_THRESHOLD) {
      const filtered = HUE_FILTER.processValue(hsv.h);
      PULSE_DETECTOR.addNewValue(filtered, performance.now() / 1000);
    }
  } else {
    VALID_FRAME_COUNTER = 0;
    MEASUREMENT_STARTED = false;
    PULSE_DETECTOR.reset();
    ON_STATUS_CHANGE('Cover the back camera until the image turns red ■');
```

```

    ON_BPM_CHANGE(0);
  }
};

const startMeasurement = () => {
  TIMER = setInterval(() => {
    const averagePeriod = PULSE_DETECTOR.getAverage();

```

```
    const pulse = averagePeriod !== PULSE_THRESHOLD ? Math.round(60 /  
averagePeriod) : 0;  
    if (pulse > 0) LAST_VALID_BPM = pulse; // Store valid BPM  
    ON_BPM_CHANGE(pulse);  
  }, 1000);  
};  
  
return publicMethods;  
})();
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ПУЛЬСУ ТА НАДАННЯ
РЕКОМЕНДАЦІЙ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ**

Програма та методика тестування

КПІ.ПІ-1522.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Микола ХРАМЧЕНКО

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Андрій МЄШКОВ

Київ – 2025

ЗМІСТ

1	ОБ’ЄКТ ВИПРОБУВАНЬ	3
2	МЕТА ТЕСТУВАННЯ.....	4
3	МЕТОДИ ТЕСТУВАННЯ	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є вебзастосунок для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту. Цей додаток створений для браузерів: Google Chrome 94+, Mozilla Firefox 93+, Apple Safari 15.0+ і Microsoft Edge 94+, переважно мобільних девайсів, як Android 10+ і iOS 14+.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності вебзастосунка з останніми версіями сучасних браузерів;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

Вибрані методи тестування забезпечують комплексний підхід до перевірки якості та надійності вебзастосунку для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту. Статичне тестування дозволяє виявити потенційні проблеми на ранніх етапах розробки шляхом аналізу коду та документації, забезпечуючи відповідність стандартам програмування та попереджаючи помилки до запуску програми. Динамічне тестування дає змогу оцінити коректність роботи вебзастосунку в реальних умовах, перевіряючи його функціональність на різних сценаріях використання. Це дозволяє виявити та проаналізувати проблеми, які можуть виникнути під час виконання програми, зокрема в процесі обробки даних пульсу та генерації рекомендацій. Мануальне тестування забезпечує детальну перевірку функціональності з точки зору кінцевого користувача, дозволяючи гнучко оцінити зручність графічного інтерфейсу та коректність роботи специфічних функцій. Тест-кейси, створені вручну, сприяють виявленню помилок, які можуть бути пропущені під час автоматизованих перевірок, та забезпечують відповідність застосунку потребам користувачів.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Порядок проведення тестування буде наступним:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на мобільних пристроях з різною роздільною здатністю екрана;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування зручності використання;
- тестування інтерфейсу користувача;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ПУЛЬСУ ТА НАДАННЯ
РЕКОМЕНДАЦІЙ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТ**

Керівництво користувача

КПІ.ПІ-1522.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Микола ХРАМЧЕНКО

Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

Виконавець:

_____ Андрій МЄШКОВ

Київ – 2025

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ.....	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи	4
2.2	Завантаження застосунку	4
2.3	Перевірка коректної роботи	4
3	ВИКОНАННЯ ПРОГРАМИ.....	5

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Вебзастосунок для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту призначений для автоматизованого відстеження частоти серцевих скорочень у реальному часі, аналізу даних пульсу за допомогою алгоритмів штучного інтелекту та надання персоналізованих рекомендацій щодо здоров'я та фізичної активності. Програма забезпечує зручний доступ до функціоналу через інтуїтивно зрозумілий вебінтерфейс, оптимізований для використання на мобільних пристроях (Android та iOS) і настільних браузерях.

Кінцева збірка програмного забезпечення включає вебзастосунок із модулями для збору даних пульсу, їх аналізу, генерації рекомендацій та відображення результатів у зрозумілій формі. Інсталяційна версія не потребує встановлення додаткового програмного забезпечення, оскільки доступ до застосунку здійснюється через сучасні веббраузери.

Метою розробки є створення надійного, точного та доступного інструменту для моніторингу стану здоров'я, який підтримує користувачів, зокрема осіб, що піклуються про своє здоров'я, спортсменів і медичних працівників, у відстеженні пульсу, виявленні аномалій і отриманні персоналізованих рекомендацій для покращення фізичного стану.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи вебзастосунку для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту необхідне виконання наступних системних вимог:

- веббраузер: Google Chrome 94+, Mozilla Firefox 93+, Apple Safari 15.0+ і Microsoft Edge 94+.
- підключення до мережі Інтернет зі швидкістю від 20 Мбіт/с;
- наявність вебкамери або сенсора для вимірювання пульсу (для мобільних пристроїв — камера з підтримкою фонарика);

2.2 Завантаження застосунку

На даний момент застосунок не потребує завантаження. Для користування проєктом користувач повинен зайти у веббраузер за посиланням вебзастосунку.

2.3 Перевірка коректної роботи

Вебзастосунок повинен коректно завантажуватись у веббраузері без помилок. Має відображатись екран привітання. Після цього користувач повинен мати змогу перейти до екрану вимірювання пульсу. При доступі до камери браузер повинен запитати дозвіл — у разі відмови застосунок не зможе виконувати основні функції. Якщо інтерфейс завантажився, камера активується, а функція вимірювання пульсу працює стабільно — застосунок встановлено і працює коректно.

3 ВИКОНАННЯ ПРОГРАМИ

Заходячи на сайт користувач бачить сторінку привітання (рис. 3.1).

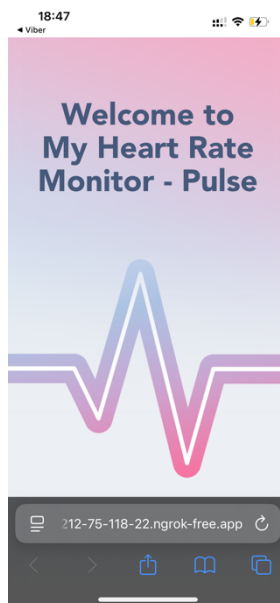


Рисунок 3.1 – Сторінка привітання

Після цього користувач попадає на сторінку вимірювання пульсу и починає виміряти дотикаючи палець до камери (рис. 3.2).

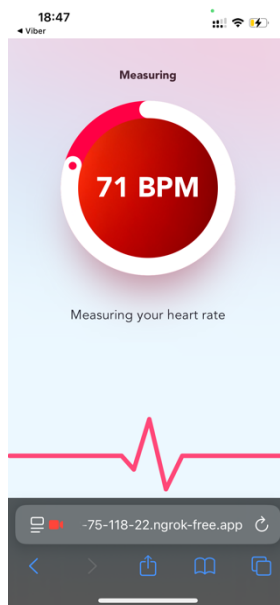


Рисунок 3.2 – Сторінка вимірювання

Через 20 секунд користувач переходить до результатів (рис. 3.3), але якщо користувач неавторизований спочатку потрібно ввести персональні дані (рис. 3.4). Користувач може змінювати значення статі, віку, ваги та рости.

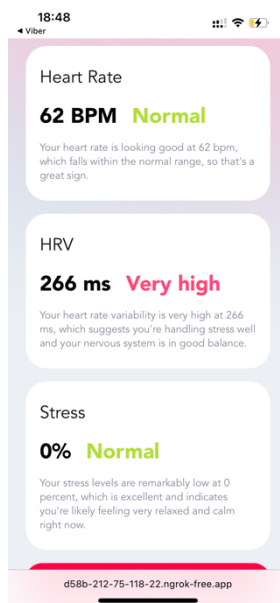


Рисунок 3.3 – Сторінка результатів

18:48
Viber

personal

lb & inch kg & cm

Male Female Other

Age 21 years

Weight 70 kilograms

What is your height?
181 centimeters

Save

d58b-212-75-118-22.ngrok-free.app

Рисунок 3.4 – Сторінка персональних даних

Якщо користувач ще неавторизований, переходить на сторінку авторизації, цей крок користувач може пропустити (рис. 3.5).

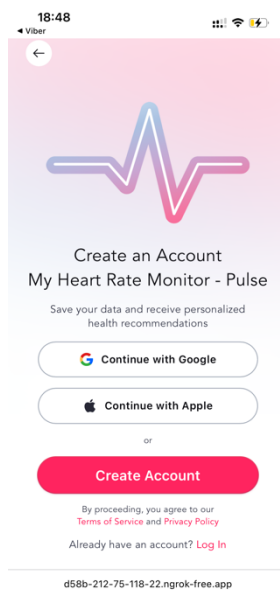


Рисунок 3.5 – Сторінка авторизації

Користувач може як створити новий акаунт (рис. 3.6) так увійти у існуючий (рис. 3.7). Користувач має змогу вводити ім'я, електронну пошту та пароль.



Рисунок 3.6 – Сторінка реєстрації

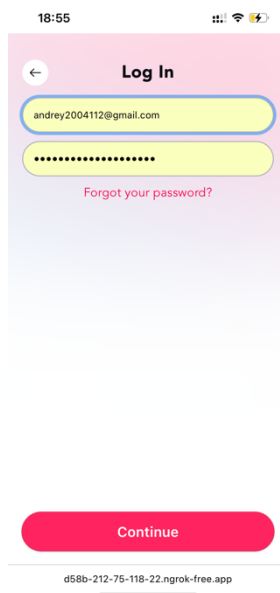


Рисунок 3.7 – Сторінка входу

Після кожного вимірювання чи авторизації користувач попадає на головну сторінку – інформаційну панель (рис. 3.8).

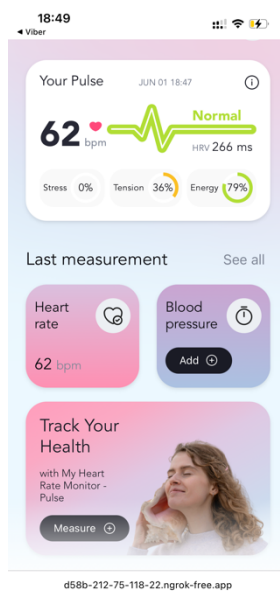


Рисунок 3.8 – Сторінка інформаційної панелі

Користувач може легко зробити нове вимірювання, натиснувши кнопку, переходить на сторінку вимірювань (рис. 3.9).

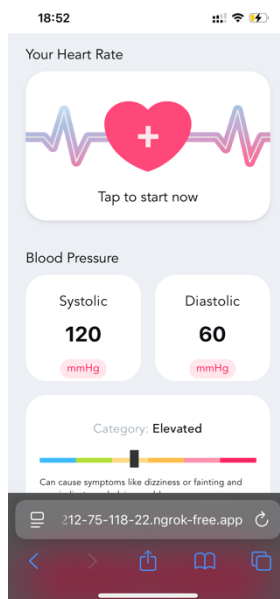


Рисунок 3.9 – Сторінка нових вимірювань

Авторизований користувач може купити платну підписку, натиснувши кнопку “AI Research” на сторінці результатів вимірювання чи історії (рис. 3.10).

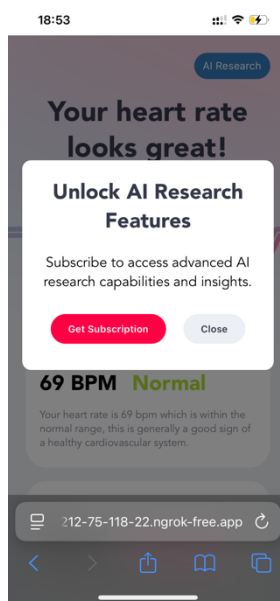


Рисунок 3.10 – Сторінка результатів вимірювання (імітація купівлі підписки)

З підпискою користувач має доступ до кращого аналізу та аналізу історії (рис. 3.11).

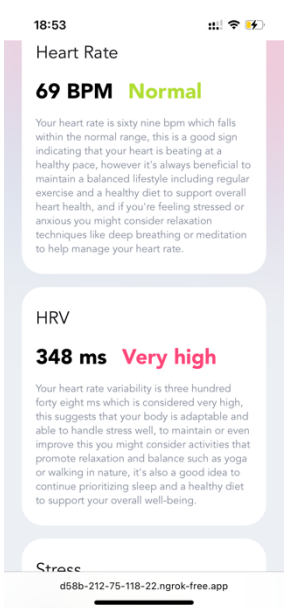


Рисунок 3.11 – Сторінка результатів вимірювання (сильніша можедь III)

Авторизований користувач має доступ до історії вимірювань з можливістю видалення та перегляду аналізу кожного запису (рис. 3.12).

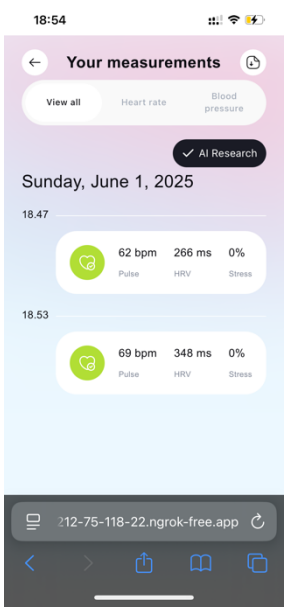


Рисунок 3.12 – Сторінка історії вимірювань

Авторизований користувач з підпискою має доступ до графіку історії (рис. 3.13).

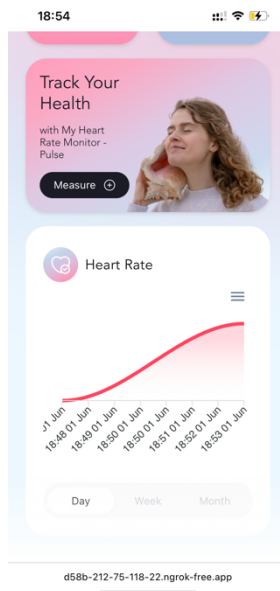


Рисунок 3.13 – Інформаційна панель користувача з підпискою

Натиснувши кнопку налаштувань, користувач може перейти на сторінку налаштувань, де є можливість виходу з акаунту (рис. 3.14).

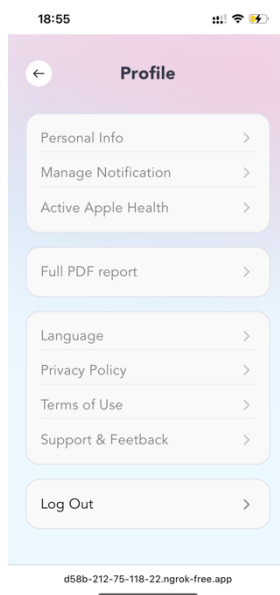


Рисунок 3.14 – Сторінка налаштувань

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ВЕБЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ПУЛЬСУ ТА НАДАННЯ
РЕКОМЕНДАЦІЙ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ**

Графічний матеріал

КПІ.ПІ-1522.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Микола ХРАМЧЕНКО

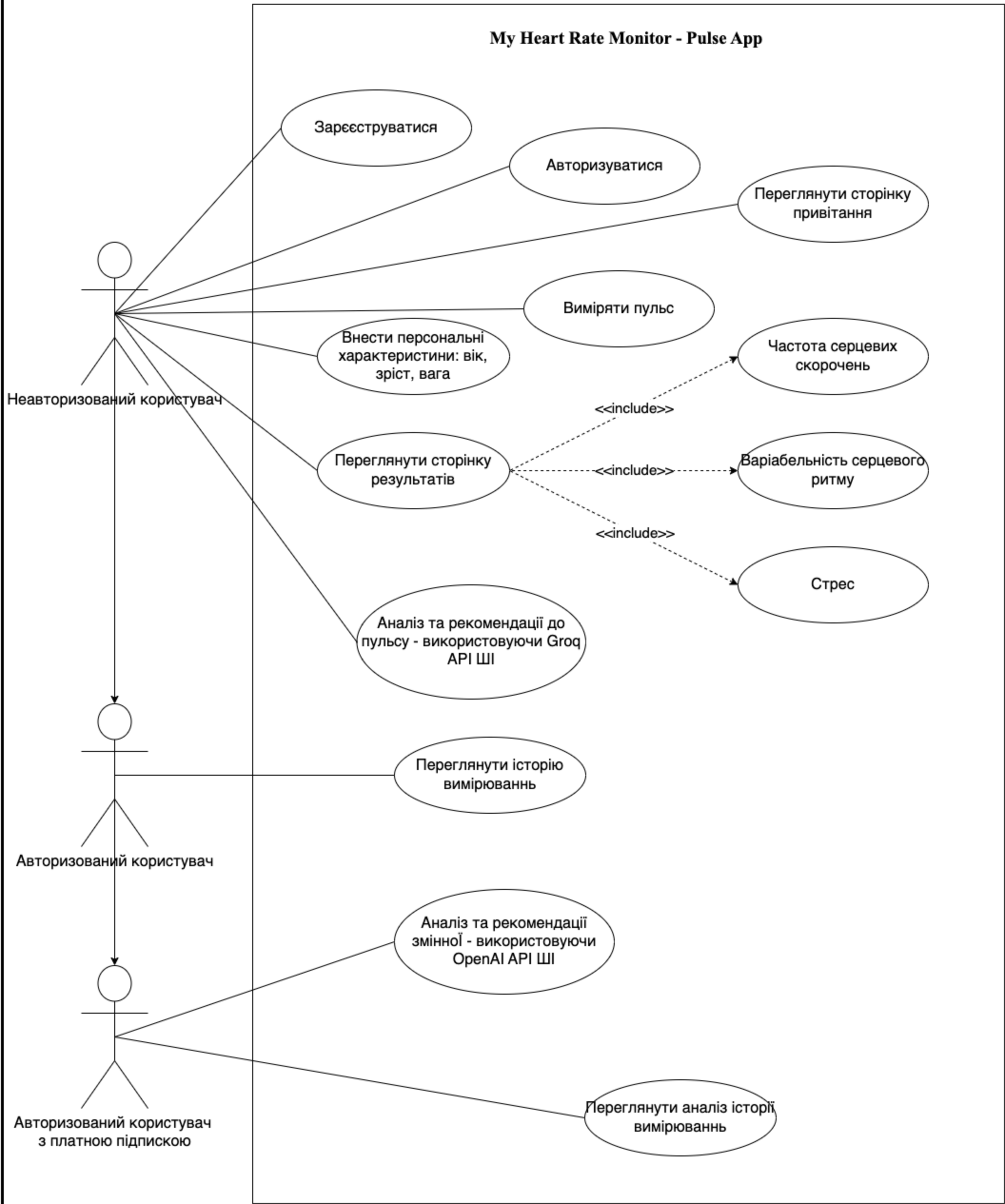
Нормоконтроль:

_____ Тетяна ШУЛЬКЕВИЧ

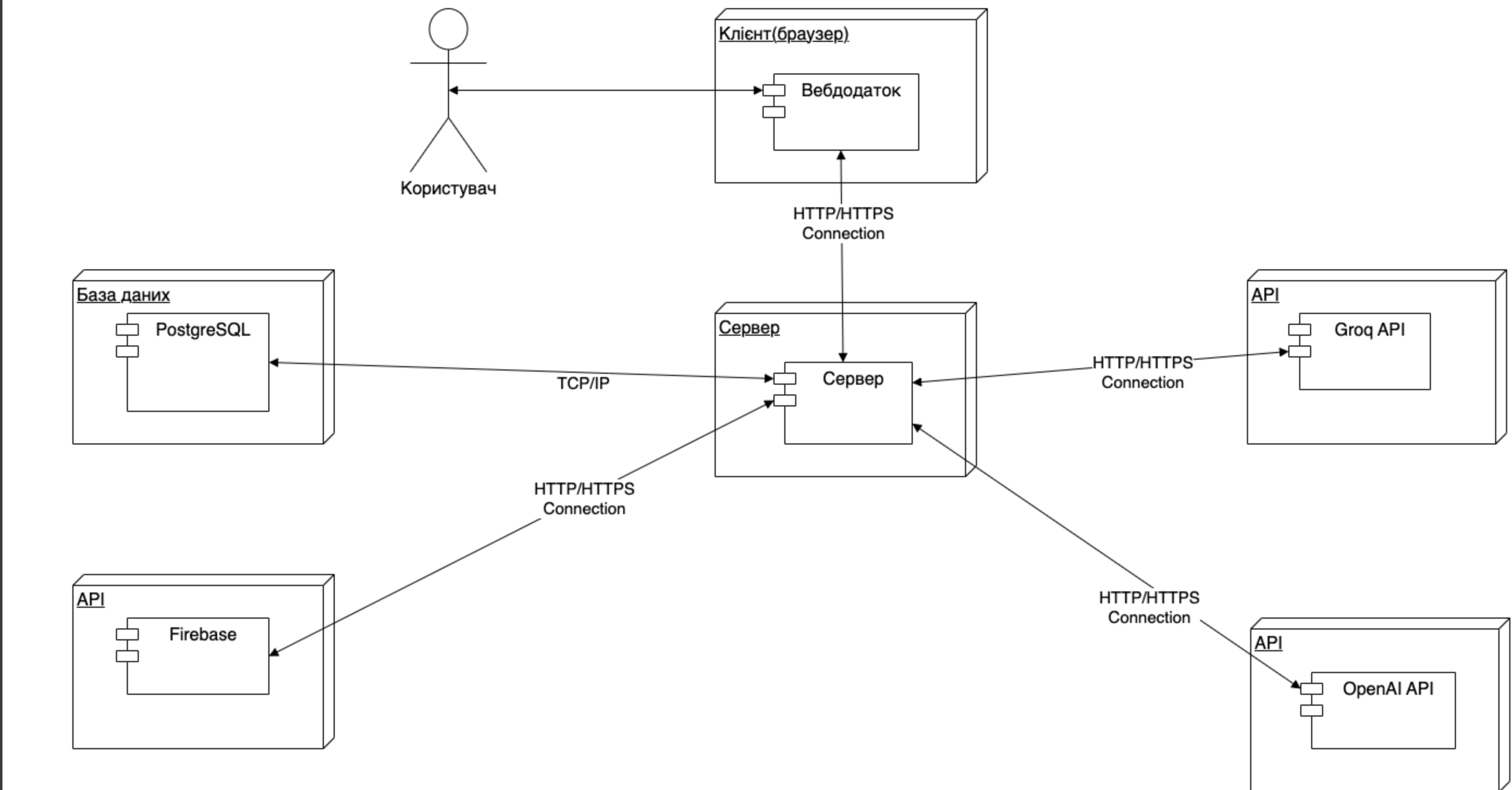
Виконавець:

_____ Андрій МЄШКОВ

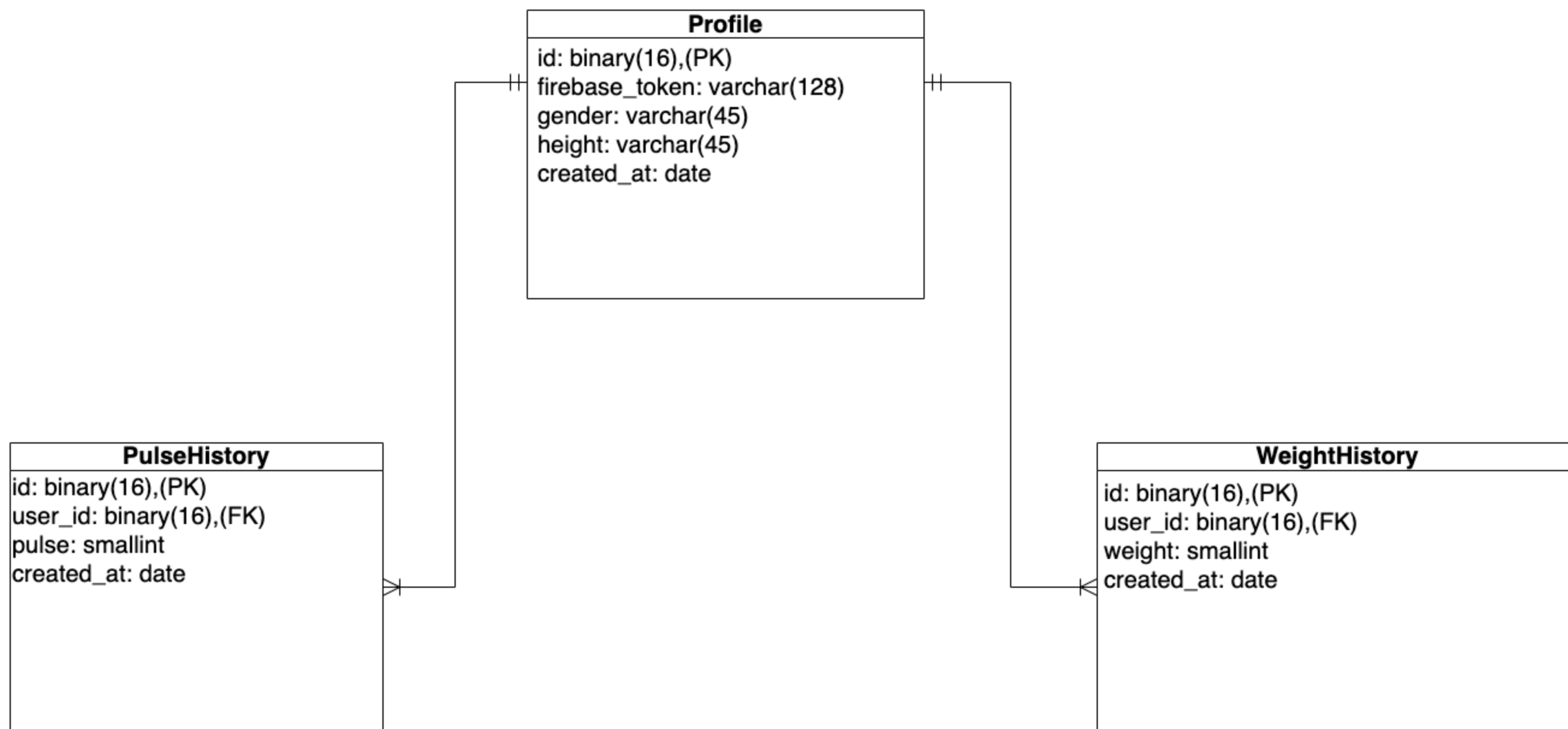
Київ – 2025



					КПІ.ІП-1522.045440.06.99.CBB								
					Схема структурна варіантів використань				Лит.		Маса	Масштаб	
Зм.	Арк.	№ докум.	Підп.	Дата	Вебзастосунок для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту				Аркуш 1		Аркушів 1		
Розробив		Мешков А.І.											
Перевірів		Храмченко М.С.											
Т. контр.													
					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15								
Н. контр.		Шулькевич Т.В.											
Затвердив		Жаріков Е.В.											



					КПІ.ІП-1522.045440.06.99.CCM							
					Схема структурна компонентів програмного забезпечення	Літера			Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата								
Розробив		Мешков А.І.										
Перевірив		Храмченко М.С.										
Т. контр.												
					Вебзастосунок для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту	Аркуш 1			Аркушів 1			
Н. контр.		Шулькевич Т.В.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15						
Затвердив		Жаріков Е.В.										



					КПІ.ІП-1522.045440.06.99.СБД						
					Схема бази даних	Літера		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата							
Розробив	Мешков А.І.										
Перевірів	Храмченко М.С.										
Т. контр.											
					Вебзастосунок для моніторингу пульсу та надання рекомендацій на основі штучного інтелекту	Аркуш 1		Аркушів 1			
Н. контр.	Шулькевич Т.В.					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-15					
Затвердив	Жаріков Е.В.										