МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМУВАННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
(бакалавр, магістр)

спеціальність 121 „Інженерія програмного забезпечення“
(шифр і назва спеціальності)

спеціалізація „ Інженерія програмного забезпечення “
(назва спеціалізації)

на тему „Мобільний додаток для відстеження стану здоров’я“

Виконала: студентка групи ІПЗ-19д _____ Ярова С.В.
(підпис) (ініціали і прізвище)

Керівник _____ Лифар В.О., д.т.н., доцент,
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ Лифар В.О., д.т.н., доцент,
(підпис) (ініціали і прізвище)

Рецензент д.т.н., професор Захожай О.І

Київ – 2023

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студентки Ярової С.В.

Науковий керівник

Доцент, д.т.н., _____ Лифар В.О.

Оцінка наукового керівника: _____

Рецензент д.т.н., професор Захожай О.І
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК _____ Лифар В.О.
підпис

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) факультет інформаційних
(повне найменування інституту, факультету)
технологій та електроніки
Кафедра інформаційних технологій та програмування
(повна назва кафедри)
Освітній ступінь бакалавр
(бакалавр, магістр)
спеціальність 121 „ Інженерія програмного забезпечення“
(шифр і назва спеціальності)
спеціалізація „ Інженерія програмного забезпечення“
(назва спеціалізації)

ЗАТВЕРДЖУЮ:

завідувач кафедри ІТП Лифар В.О.

«____» _____2023 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Ярової Світлани Володимирівни
(прізвище, ім’я, по батькові)

1. Тема роботи: Мобільний додаток для відстеження стану здоров’я

Керівник роботи Лифар Володимир Олексійович, д.т.н „доцент ,
затверджений наказом університету від “ 26 ” 04 2023року № 240/15.15-ОД

2. Строк подання студенткою роботи 17 червня 2023
3. Вихідні дані до роботи матеріали переддипломної практики

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

- аналіз предметної області;

- вибір та обґрунтування програмних засобів
- розробки системи;
- розробка застосунку;
- висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)

_____
_____
_____
_____

6. Консультанти розділів проекту (роботи)

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

7. Дата видачі завдання 24.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	24.03.2023-04.04.2023	
2	Пошук та аналіз наукових та технічних джерел, що стосуються теми.	04.04.2023-08.04.2023	
3	Визначення основних вимог і функціональності застосунку.	08.04.2023-23.04.2023	

4	Опис функціональності та інтерфейсу, реалізація можливості зміни даних.	23.04.2023-28.04.2023	
5	Розробка архітектури застосунку, включаючи визначення компонентів, модулів та їх взаємодії.	28.04.2023-13.05.2023	
6	Розробка функціоналу додавання нових даних, включаючи перевірку коректності даних.	13.05.2023-30.05.2023	
7	Оптимізація роботи застосунку для забезпечення швидкості та продуктивності.	28.05.2023-30.05.2023	
8	Підготовка документації, оформлення дипломного проекту та написання пояснювальної записки.	31.05.2023-15.06.2023	

Здобувач вищої освіти _____ Ярова С.В.
(підпис) (ініціали, прізвище)

Керівник _____ Лифар В.О.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 55 с., 13 мал., 19 бібліографічних джерел, 2 додатки.

Об'єкт розробки: Об'єктом розробки є застосунок для відстеження стану здоров'я, який буде працювати на платформі React. Застосунок буде мати зручний та інтуїтивно зрозумілий інтерфейс, що дозволить користувачам легко вводити та відстежувати свої показники здоров'я.

Мета роботи: Основною метою даної дипломної роботи є розробка застосунку, який надасть можливість користувачам відстежувати свій стан здоров'я та збирати дані про різні показники, пов'язаними з їх фізичним та психічним станом.

В процесі виконання дипломного проекту були досягнуті наступні цілі:

1. Був проведений аналіз існуючих застосунків, які надають можливість відстежувати стан здоров'я користувачів. В процесі аналізу були виявлені їхні переваги та недоліки, а також інноваційні рішення, які можна впровадити у розробку власного застосунку.

2. Був розроблений стартовий екран, де користувачі можуть зареєструватись при першому вході. Також було створено вкладки з відображенням графіків і налаштуваннями застосунку.

3. Реалізована можливість введення та збереження різних показників здоров'я. Додано перевірки, які забороняють додавання неповних даних.

4. Графічне відображення даних: розроблено графік, який відображає, в які дні користувач додавав дані, а в які пропускав. Це дозволяє користувачеві відстежувати свою активність та стежити за регулярністю внесення даних.

5. Використання технології React: для реалізації застосунку була обрана технологія React, що дозволило створити ефективну та зручну для розробки та використання систему відстеження стану здоров'я.

ЗМІСТ

ВСТУП

Останні роки зростає значущість питань, пов'язаних зі здоров'ям та самопочуттям людей. Спостереження за станом здоров'я може допомогти вчасно виявити можливі проблеми та підвищити якість життя. Однак, багатозадачність та швидкий ритм сучасного життя часто ускладнюють відстеження цих показників. Тому розробка застосунку, який би дозволяв зручно та ефективно відстежувати стан здоров'я, є актуальним завданням.

Метою даного дипломного проекту є розробка застосунку для відстеження стану здоров'я, який надасть користувачам зручні та інтуїтивно зрозумілі інструменти для введення та аналізу даних про їх фізичне та емоційне самопочуття. Основні завдання проекту включають створення інтерфейсу, де користувачі матимуть змогу ввести та зберігати інформацію про своє ім'я, вагу, зріст, вік, кількість кроків, час початку та закінчення сну, настрій, бадьорість та шкалу навантаження за день.

Для розробки застосунку обрано технологію React, яка є однією з найпопулярніших та потужних бібліотек для розробки веб-інтерфейсів. React дозволяє побудувати ефективний та модульний інтерфейс, що полегшує розробку, тестування та підтримку застосунку. Крім того, будуть використані інші сучасні інструменти та бібліотеки, такі як Redux для керування станом додатку, React Router для навігації між сторінками та Chart.js для побудови графіків.

У рамках проекту будуть розроблені перевірки та обмеження для введених даних, щоб забезпечити коректність та консистентність інформації. Буде запроваджено заборону на додавання даних, якщо не всі поля заповнені, щоб забезпечити повноту та достовірність інформації.

Після завершення розробки застосунку буде проведено тестування та валідацію, щоб перевірити його відповідність заданим вимогам та функціональності. Будуть виконані регресійні тести, які перевірять, чи

працює програма без помилок та збоїв після внесення змін або додавання нового функціоналу. Аналіз результатів допоможе виявити та виправити можливі помилки та недоліки перед фінальною версією застосунку.

Застосунок надасть користувачам зручні та інтуїтивні інструменти для самостійного відстеження та контролю за їх здоров'ям, що сприятиме покращенню якості життя та свідомому підходу до здорового способу життя.

Виконання цього дипломного проекту сприятиме розвитку інженерії програмного забезпечення, поглибленню знань та навичок у розробці веб-додатків, а також допоможе покращити якість життя користувачів, сприяючи свідомому відстеженню та підтримці їх здоров'я.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Аналізуючи сучасну предметну область, відстеження стану здоров'я, ми спостерігаємо значний розквіт інтересу до застосунків, які допомагають людям зберігати та поліпшувати своє фізичне і психічне самопочуття. Це свідчить про важливість здорового способу життя та активної самопідтримки.

Цікавою тенденцією в цій сфері є широке поширення мобільних додатків, які роблять відстеження стану здоров'я більш доступним та зручним для користувачів. Завдяки цим додаткам, власники смартфонів та планшетів можуть легко контролювати свої фізичні показники у будь-який час і в будь-якому місці.

Окрім загального відстеження фізичної активності, у цій сфері акцентується увага на важливості якісного сну. Багато застосунків пропонують функціонал, який допомагає користувачам вимірювати тривалість та якість свого сну. Вони надають цінну інформацію про глибину сну, кількість пробуджень та інші параметри, що сприяють поліпшенню режиму сну.

Зараз відстеження стану здоров'я стає все більш інтелектуальним завдяки використанню штучного інтелекту та машинного навчання. Деякі додатки можуть навчитися розпізнавати зміни в здоров'ї користувача та робити прогнози стосовно можливих захворювань або проблем. Це відкриває нові перспективи для вчасного виявлення та попередження хвороб.

Усе більша популярність цих застосунків підтверджує їх значущість та вплив на суспільство. Вони допомагають людям стежити за своїм здоров'ям, мотивують до активності та покращення життєвого стилю. Інноваційні функції та технології, які використовуються в цих додатках, постійно розвиваються та розширюються, що дає нам змогу досягати ще більших досягнень у відстеженні та покращенні стану здоров'я.

1.1. Аналіз розробки застосунків

Розробка схожих застосунків для відстеження стану здоров'я є складним процесом, який вимагає ретельного планування та використання різноманітних технологій.

Першим кроком у розробці таких застосунків є аналіз потреб користувачів. Необхідно з'ясувати, які саме показники здоров'я користувачі хочуть відстежувати і які функції вони хотіли б мати у додатку. Цей аналіз включає вивчення існуючих ринкових рішень, опитування потенційних користувачів та врахування їхніх пропозицій та вимог.

Далі важливим етапом є проектування інтерфейсу користувача. Необхідно розробити зручний та привабливий дизайн, який дозволить легко взаємодіяти з додатком. Доцільно враховувати принципи юзабіліті, розташування елементів та інтуїтивне керування. Також варто пам'ятати про можливості адаптації до різних пристроїв та розмірів екранів.

Після цього настає етап розробки функціональної частини додатку. Для цього використовуються різноманітні технології та мови програмування, зокрема такі як JavaScript, React, Angular або інші. Важливо врахувати потреби користувачів та реалізувати необхідні функції, такі як вимірювання показників здоров'я, відображення графіків, сповіщення та інші.

Особлива увага повинна бути приділена аспектам безпеки та конфіденційності. Дані про стан здоров'я є особистими та приватними, тому необхідно забезпечити надійний захист від несанкціонованого доступу та зловживання.

Крім того, розробникам таких застосунків важливо враховувати різні особливості та нюанси, які пов'язані з певними групами користувачів. Наприклад, додаток може мати функції для відстеження стану здоров'я дітей, жінок під час вагітності або людей з певними хронічними захворюваннями. У

таких випадках потрібно враховувати специфічні вимоги та надати користувачам необхідну інформацію та функціональність.

Оглядаючи цю сферу, варто відзначити, що розробка застосунків для відстеження стану здоров'я постійно розвивається та збагачується новими можливостями. З'являються інтеграції з різними пристроями та сенсорами, що дозволяють автоматично отримувати дані про фізичну активність, серцевий ритм, рівень кисню в крові та інші параметри. Також з'являються розумні алгоритми та штучний інтелект, які допомагають аналізувати отримані дані та надавати корисні рекомендації.

У підсумку, розробка застосунків для відстеження стану здоров'я є складним процесом, який вимагає аналізу потреб користувачів, проектування зручного інтерфейсу, розробки функціональної частини та забезпечення безпеки та конфіденційності. Ця сфера постійно розвивається, пропонуючи нові можливості та інноваційні рішення для покращення здоров'я та життя людей.

1.2. Аналіз застосунків представлених на ринку

Аналізуючи існуючі застосунки в сфері відстеження стану здоров'я, варто звернути увагу на такі популярні рішення, як Mi Fit, Apple Health, Google Fit і Samsung Health. Кожен з цих застосунків пропонує широкий спектр функціональних можливостей, спрямованих на відстеження фізичної активності, серцевого ритму, сну та інших показників здоров'я.

Mi Fit є популярним застосунком для відстеження стану здоров'я, розробленим компанією Xiaomi. Він пропонує користувачам простий та зручний інтерфейс, який дозволяє легко встановлювати цілі щодо фізичної активності, серцевого ритму та інших показників здоров'я. Застосунок також

надає можливість спілкуватися та конкурувати з іншими користувачами, що стимулює досягнення цілей та активну життєвий стиль.

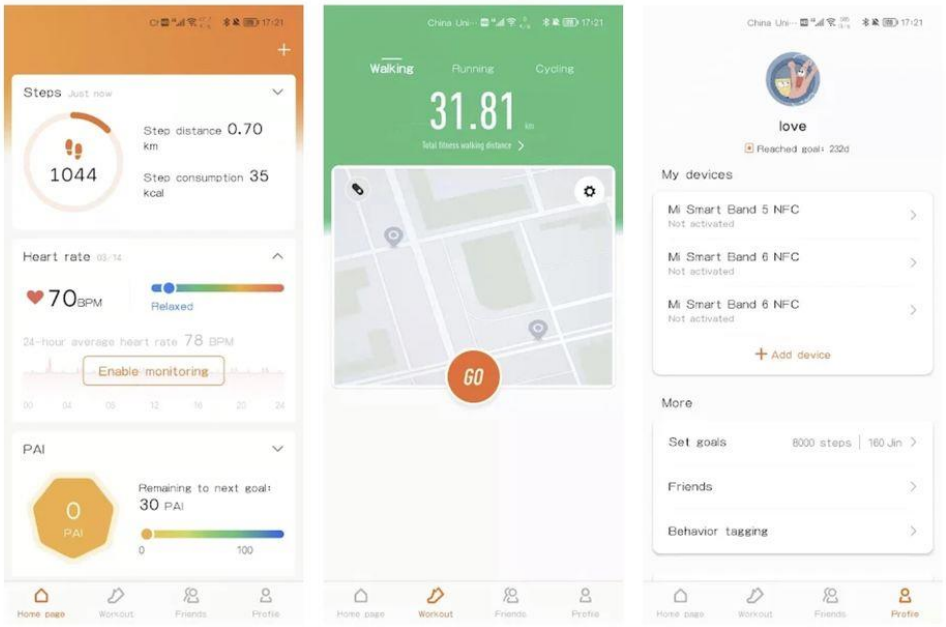

Рис. 1.1 – Огляд застосунку Mi Fit

Apple Health є вбудованим застосунком для iOS, який пропонує інтегровану платформу для відстеження різних аспектів здоров'я. Він автоматично збирає дані з різних джерел, таких як фітнес-трекери, сенсори та медичні пристрої, і зберігає їх в одному місці. Застосунок дозволяє користувачам відстежувати фізичну активність, засипання та якість сну, пульс, а також інші показники здоров'я. Крім того, він може бути інтегрований з іншими додатками та пристроями, що робить його універсальним інструментом для відстеження здоров'я на пристроях Apple.

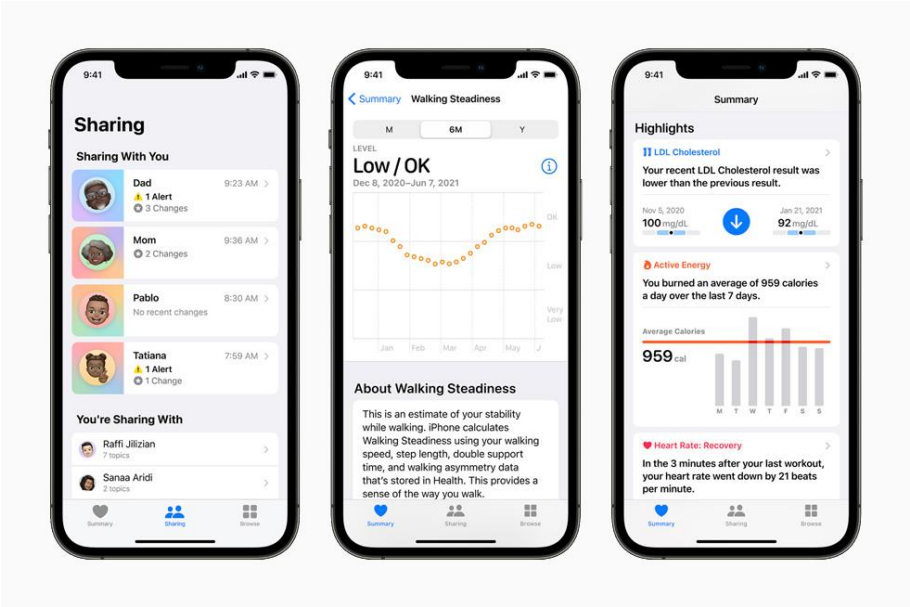

Рис. 1.2 – Огляд застосунку Apple Health

Google Fit є розширеним застосунком для відстеження фізичної активності та стану здоров'я, розробленим компанією Google. Цей застосунок надає можливість відстежувати різні показники здоров'я, включаючи кроки, пройдену відстань, активність на основі серцевого ритму, а також засипання та якість сну. Застосунок також пропонує інтеграцію з іншими додатками та пристроями, що дозволяє збирати інформацію з різних джерел і отримувати повну картину про стан здоров'я користувача.

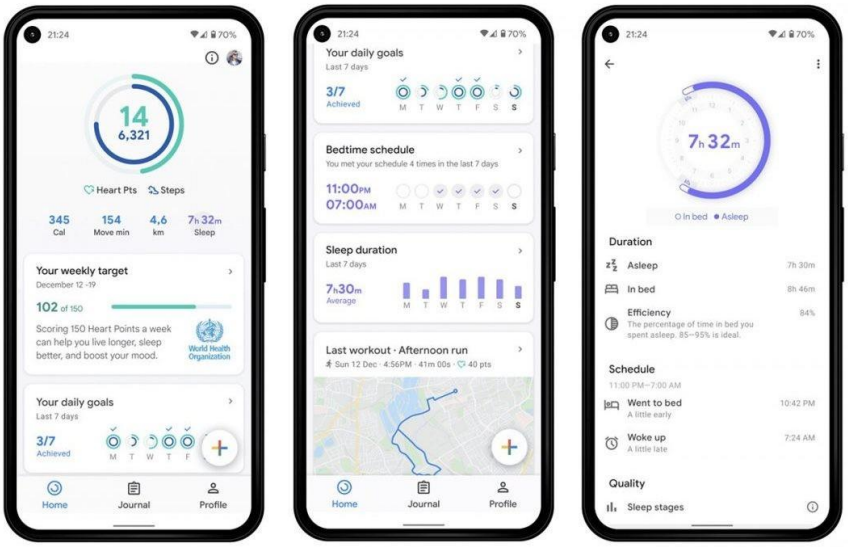

Рис. 1.3 – Огляд застосунку Google Fit

Samsung Health є застосунком, розробленим компанією Samsung, який пропонує широкий спектр можливостей для відстеження здоров'я. Він дозволяє користувачам відстежувати фізичну активність, серцевий ритм, якість сну та інші показники. Застосунок також має інтуїтивно зрозумілий інтерфейс та вбудовані функції, такі як нагадування про прийом лікарських препаратів, інтерактивні тренування та спільноту користувачів. Він може бути синхронізований з різними пристроями Samsung, що забезпечує зручну інтеграцію і співпрацю між різними пристроями.

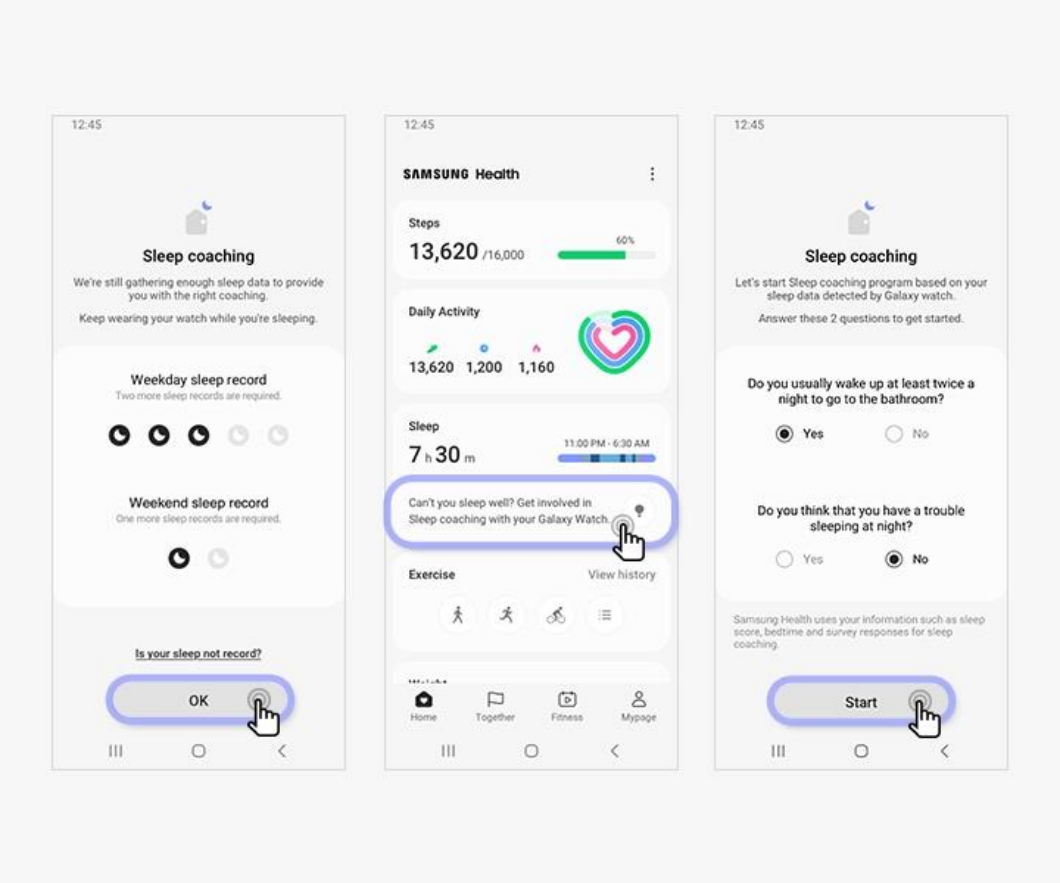

Рис. 1.4 – Огляд застосунку Samsung Health

Кожен з цих застосунків має свої особливості і переваги, проте сам застосунок для відстеження стану здоров'я може виділятися своїм унікальним функціоналом та інтерфейсом, який задовольнятиме потреби користувачів у зручному та ефективному відстеженні їхнього здоров'я і також ідентифікувати незадоволені потреби користувачів. Враховуючи особливості існуючих застосунків, можливо створити продукт, який буде відрізнятися

своїми унікальними рішеннями, зручним інтерфейсом та інтеграцією з різними пристроями, що забезпечить високу конкурентоспроможність продукту на ринку та пропонуватиме користувачам нові можливості для підтримки і поліпшення їхнього здоров'я.

2. ВИБІР ТЕХНОЛОГІЙ ТА АРХІТЕКТУРИ РОЗРОБКИ

2.1. Мова JavaScript

Дипломна робота присвячена розробці застосунку для відстеження стану здоров'я, з використанням мови програмування JavaScript. JavaScript [3] є однією з найпопулярніших мов для розробки веб-додатків, вона була створена в 1995 році і відтоді знаходить широке застосування у сфері веб-розробки.

Дослідження предметної області включало аналіз існуючих застосунків. Вони вже зарекомендували себе як корисні інструменти для відстеження фізичної активності, аналізу даних про здоров'я та сприяння покращенню загального стану організму. Проте, у багатьох з них виявлені певні обмеження та відсутність налаштувань під потреби конкретного користувача.

Використання JavaScript у данному застосунку дозволить забезпечити динамічність та інтерактивність веб-додатка. Ви зможете створювати графіки, діаграми та інші візуальні елементи, що допоможуть відображати інформацію про стан здоров'я у зручному та зрозумілому форматі. JavaScript також надає можливість проводити валідацію та перевірку введених даних, що гарантує правильність та достовірність інформації.

Детальне вивчення JavaScript дозволить реалізувати функціональність застосунку згідно зі специфікаціями та вимогами. Можливо створити інтерфейс, який забезпечить зручну навігацію та налаштування згідно з вподобаннями користувача. Застосунок буде мати гнучку архітектуру, що дозволить легко розширювати його функціональність та додавати нові можливості у майбутньому.

Застосування JavaScript відкриває безліч можливостей для розробки інноваційного та корисного застосунку для відстеження стану здоров'я. З

урахуванням історії та потенціалу JavaScript, дипломна робота сприятиме розвитку та вдосконаленню області e-Health, допомагаючи людям берегти та покращувати своє здоров'я.

JavaScript Object Notation (JSON) є популярним форматом обміну даними, який використовується в багатьох сферах програмування, включаючи веб-розробку. JSON використовується для структурування та передачі даних між сервером та клієнтом.

Один із ключових аспектів JavaScript у контексті дипломної роботи є його здатність легко працювати з JSON. JavaScript має вбудовані функції для перетворення об'єктів JavaScript у JSON-строки та навпаки. Це дає змогу зберігати, обробляти та передавати дані у структурованому форматі.

Застосунок може використовувати JSON для зберігання та обробки даних, які вводяться користувачем, такі як ім'я, вага, зріст, кількість кроків, час початку і закінчення сну та інші параметри, що стосуються стану здоров'я. Використання JSON дозволяє зберігати ці дані в структурованому форматі, що спрощує їх обробку та передачу між різними компонентами застосунку.

Крім того, JavaScript також має вбудовані методи для роботи з JSON-даними. Можливе використання цих методів для здійснення розбору JSON-строки та доступу до конкретних елементів об'єкта JSON. Це дозволяє легко отримувати та обробляти інформацію, яка міститься в JSON-структурах даних.

Також варто зазначити, що JSON має простий та зрозумілий синтаксис, який є легким для вивчення та роботи. Він є популярним стандартом обміну даними в багатьох середовищах програмування та має велику підтримку спільноти розробників. Використання JSON у застосунку сприятиме його сумісності з іншими системами та сервісами, що працюють з даними у цьому форматі.

Загалом, використання JSON у дипломній роботі забезпечить ефективну та зручну обробку даних стану здоров'я, спрощуючи зберігання, обмін та обробку інформації. Його зрозумілий синтаксис та широка підтримка роблять його ідеальним інструментом для розробки застосунку.

2.2. Мова TypeScript

TypeScript [13] є мовою програмування, розробленою компанією Microsoft, яка використовується для розробки веб-додатків. Вона є розширенням мови JavaScript і надає розробникам багато переваг і можливостей. Однією з найважливіших особливостей TypeScript є статична типізація, що дозволяє визначати типи для змінних, параметрів функцій та інших елементів програми. Це полегшує розробку, допомагає виявляти помилки на ранніх етапах і забезпечує більшу надійність коду.

Завдяки статичній типізації, TypeScript дозволяє розробникам встановлювати типи даних і забезпечувати їх правильність. Є можливість визначати типи для змінних, параметрів функцій, повернутих значень та інших об'єктів. Це полегшує розуміння коду і покращує співпрацю між розробниками. Крім того, TypeScript допомагає виявляти помилки на етапі розробки, що дозволяє їх виправляти до запуску програми.

Однією з корисних можливостей TypeScript є використання типу єнам (enum). Тип єнам дозволяє визначати набір іменованих констант, що можуть використовуватись у коді. Використання єнами для представлення фіксованих значень, таких як стани, категорії або варіанти є важливою складовою даної мови. Це спрощує роботу з дискретними наборами значень і полегшує читання та розуміння коду.

Ще однією важливою перевагою TypeScript є його широка підтримка і активна спільнота розробників. Багато популярних фреймворків та бібліотек, таких як Angular, React та Vue, мають вбудовану підтримку TypeScript. Це

дозволяє розробникам використовувати TypeScript у своїх проектах і отримувати підтримку інструментів, документації та спільноти.

У своїй історії TypeScript пройшов шлях від початкової версії до стабільного і потужного інструменту для розробки веб-додатків. Вперше TypeScript був представлений у 2012 році як експериментальний проект компанії Microsoft. За час свого існування він отримав значний розвиток, ставши популярним серед розробників.

Узагальнюючи, TypeScript є потужним інструментом для розробки веб-додатків, який надає статичну типізацію, підтримку типу єнам та розширені можливості мови JavaScript. Він сприяє полегшенню розробки, забезпечує більшу надійність коду і покращує співпрацю між розробниками. Завдяки широкій підтримці і активній спільноті, TypeScript став популярним і незамінним інструментом для багатьох проектів розробки веб-додатків.

2.3. Огляд популярних бібліотек

2.3.1 Xamarin

Xamarin - кросплатформовий фреймворк для розробки мобільних додатків на основі мови програмування C# та з потужним потенціалом платформи .NET. Він дозволяє використовувати спільний код для створення додатків для iOS, Android та Windows.

Однією з основних переваг Xamarin є його наближеність до нативних додатків. Він дозволяє розробникам користуватися усіма перевагами та можливостями кожної платформи, використовуючи спільний код. Це означає, що розробники можуть ефективно використовувати готові бібліотеки, інструменти та ресурси, що робить розробку більш зручною і

швидкою. Доступ до специфічних функцій платформи та великий вибір інструментів та ресурсів вважається за неабиякий плюс.

Проте є і кілька недоліків, з якими можна зіткнутися при використанні Xamarin. Один з них - це вимога до додаткового часу та зусиль для вивчення фреймворку. Він має свою унікальну структуру та концепції, що потребує деякого часу для освоєння. Також, порівняно з нативними рішеннями, розробка на Xamarin може бути трохи повільнішою. Однак, ці невеликі виклики не зменшують популярності Xamarin серед розробників. Його здатність створювати потужні, кросплатформові додатки, що використовують спільний код, зберігаючи доступ до функціональності платформи, робить його знаковим інструментом для сучасних розробників мобільних додатків. Все це робить Xamarin незамінним союзником у світі мобільного програмування.

2.3.2. Flutter

Flutter - це сучасний відкритий фреймворк для розробки кросплатформових мобільних та веб-додатків. Його основна перевага полягає в тому, що він дозволяє розробникам створювати красиві та високопродуктивні додатки, які працюють на різних платформах, використовуючи один і той же код.

Ключовими перевагами Flutter є його швидкість та продуктивність. Він використовує власний движок рендерингу, що дозволяє створювати додатки з високою швидкістю відгуку та плавною анімацією. Крім того, Flutter має гарний набір вбудованих віджетів і багатий набір інструментів для стилізації та налаштування вигляду додатків.

Ще одна перевага Flutter полягає в його гарній підтримці спільного коду. Розробники можуть використовувати один і той же код для створення додатків для iOS та Android, що дозволяє значно скоротити час розробки та

зменшити зусилля. Його швидкість, продуктивність та спільний код роблять його привабливим вибором для розробників, які прагнуть створювати сучасні та ефективні мобільні додатки.

Проте, як і у будь-якого фреймворка, є і кілька недоліків у використанні Flutter. Один з них - це його відносно новий статус на ринку, що означає, що деякі розробники можуть бути менш знайомі з ним і мати обмежений доступ до підтримки та ресурсів. Крім того, Flutter, будучи фреймворком з власним двигуном рендерингу, може мати деякі обмеження в екосистемі плагінів та бібліотек, порівняно з встановленою базою інструментів для розробки.

2.3.3. Ionic

Ionic - це відкритий фреймворк для розробки кросплатформових мобільних додатків з використанням веб-технологій, таких як HTML, CSS та JavaScript. Основна перевага Ionic полягає в його простоті використання та швидкості розробки. Завдяки використанню веб-стандартів, розробники можуть використовувати свої вміння в HTML, CSS та JavaScript для створення мобільних додатків для різних платформ.

Ще однією перевагою Ionic є його широкий набір UI компонентів та готових стилів, що дозволяють швидко створювати естетично привабливі додатки. Крім того, Ionic має велику спільноту розробників, що забезпечує доступ до різноманітних розширень, плагінів та інструментів для розробки.

Проте, у фреймворка Ionic є деякі недоліки. Один з них - це обмежені можливості доступу до функціональності пристрою, порівняно з нативною розробкою. Деякі специфічні функції можуть бути складнішими для реалізації у фреймворку Ionic, порівняно з іншими платформами розробки. Крім того, швидкість виконання додатків у фреймворку Ionic може бути трохи нижчою, порівняно з нативними додатками.

Загалом, Ionic є зручним і простим у використанні фреймворком для швидкої розробки кросплатформових мобільних додатків. Його привабливість полягає в використанні веб-технологій, багатому наборі компонентів та готових стилів, а також активній спільноті розробників, яка забезпечує доступ до різноманітних розширень та підтримку.

2.3.4. Apache Cordova

Apache Cordova - це відкритий фреймворк для розробки кросплатформових мобільних додатків, який дозволяє використовувати веб-технології, такі як HTML, CSS та JavaScript, для створення додатків для різних платформ. Основна перевага полягає в його широкій підтримці платформ, включаючи iOS, Android, Windows Phone та багато інших.

Однією з ключових переваг Apache Cordova є його можливість використовувати один і той же код для розробки додатків для різних платформ. Це дозволяє розробникам економити час і зусилля, оскільки вони можуть уникнути потреби у створенні окремих додатків для кожної платформи. Крім того, Apache Cordova надає доступ до різноманітних плагінів, які дозволяють використовувати функціональність пристрою, таку як камера, геолокація та інші.

Однак, Apache Cordova також має деякі недоліки. Оскільки додатки розробляються з використанням веб-технологій, їх швидкість виконання може бути трохи повільнішою, порівняно з нативними додатками. Тому, бажано, враховувати можливі обмеження швидкості виконання та доступу до функціональності пристрою. Крім того, доступ до певної функціональності пристрою може бути обмеженим або потребувати додаткових налаштувань.

2.3.5. Фреймворк React

React [10] - це відкрита JavaScript бібліотека для створення користувацького інтерфейсу. Вона була розроблена компанією Facebook і спочатку випущена в 2013 році. Однак, досягнувши великої популярності, вона швидко стала одним з найуспішніших інструментів для розробки веб-додатків.

Основними перевагами використання React є:

- Висока продуктивність: React використовує віртуальний DOM, що дозволяє ефективно оновлювати лише ті елементи, які потребують змін. Це забезпечує швидку роботу додатків навіть з великою кількістю даних.

- Компонентна архітектура: React побудований на основі компонентів, що дозволяє розбити складний інтерфейс на невеликі, самодостатні блоки. Це спрощує розробку, тестування та підтримку додатків.

- Інструменти тестування: React надає потужні інструменти для тестування компонентів, що дозволяє швидко та надійно перевірити їх працездатність.

- Широке співтовариство: React має велике співтовариство розробників, що приносить багато готових рішень, бібліотек та інструментів. Це дозволяє зекономити час та зусилля при розробці додатків.

Однак, на ринку також є деякі мінуси використання React:

- Важкість навчання: Розробка в React вимагає від розробника знань JavaScript та JSX, а також розуміння компонентної моделі. Це може бути складним для початківців.

- Відсутність вбудованої підтримки для маршрутизації та керування станом: Для цих завдань потрібно використовувати додаткові бібліотеки, такі як React Router або Redux. Це може призвести до додаткового часу та складнощів у розробці.

- React залишається одним з найактуальніших інструментів на ринку розробки веб-додатків. Він має широке застосування в різних галузях, від створення односторінкових додатків до масштабних веб-платформ. Завдяки активному розвитку спільноти та постійному оновленню функціональності, Реакт надає розробникам потужний інструментарій для творення сучасних інтерактивних інтерфейсів.

Його екосистема також постійно розширюється, включаючи багато додаткових бібліотек, фреймворків та інструментів розробки, що спрощують і прискорюють процес створення додатків. React є перспективним вибором для розробників, які прагнуть створювати сучасні та ефективні веб-додатки з високою продуктивністю та швидкими часами розробки.

2.3.6. Фреймворк React Native

React Native [9] - це відкритий фреймворк для розробки мобільних додатків, розроблений компанією Facebook. Він дозволяє розробникам створювати нативні мобільні додатки за допомогою JavaScript та React. React Native був створений з метою вирішити проблему кросплатформленої розробки. Це означає, що можна використовувати спільний код для різних платформ, замість того, щоб писати окремий код для кожної платформи. Це зберігає час та зусилля розробника, а також сприяє швидкому впровадженню продукту на різних мобільних платформах.

Крім того, React Native надає доступ до багатьох пристрійних функцій, таких як камера, геолокація, гіроскоп та інші. Це дозволяє розробникам створювати додатки з багатим функціоналом, що можуть використовувати пристрійні можливості для забезпечення кращого користувацького досвіду.

Однією з головних переваг React Native є його здатність до нативної продуктивності та користувацького досвіду. Завдяки використанню нативних компонентів, додатки, розроблені з використанням React Native, можуть

досягати високої продуктивності, плавних анімацій та реагувати на взаємодію користувача. Це можливо завдяки тому, що React Native перекладає JavaScript-код у нативні компоненти користувацького інтерфейсу, роблячи додаток нерозрізним від додатків, розроблених з використанням специфічних для платформи мов, таких як Java або Swift.

Звичайно, надзвичайним плюсом React Native є широке співтовариство розробників, що внесло великий вклад у розробку плагінів, бібліотек та інструментів, що роблять процес розробки більш простим та продуктивним. Розробники можуть легко знайти рішення для своїх конкретних потреб або навіть співпрацювати з іншими розробниками для обміну знаннями та досвідом.

Ще однією значною перевагою React Native є можливість повторного використання коду. За допомогою фреймворка розробники можуть писати єдину базу коду, яку можна використовувати для розробки додатків для обох платформ - iOS та Android. Це значно зменшує час та зусилля, потрібні для розробки, оскільки розробники більше не повинні підтримувати окремі бази коду для кожної платформи. Крім того, React Native надає функцію гарячої перезавантаження, яка дозволяє розробникам бачити зміни в реальному часі без повного перекомпілювання додатку.

Також важливою у React Native є підтримка від компанії Facebook, яка забезпечує активний розвиток фреймворку та постійне оновлення його можливостей. Це гарантує, що React Native залишається актуальним і відповідає сучасним вимогам розробки мобільних додатків.

Незважаючи на багато переваг, у React Native також є свої обмеження. Однією з головних складнощів є необхідність використовувати нативні модулі. Хоча React Native надає широкий набір готових компонентів, можуть бути випадки, коли розробникам потрібно отримати доступ до специфічних для платформи функцій або сторонніх бібліотек, які не входять до комплекту

поставки. У таких випадках розробники повинні писати власні нативні модулі, що потребує додаткових зусиль та експертизи.

Бібліотека також має більш крутий поріг вивчення порівняно з традиційною веб-розробкою з використанням React. Розробникам потрібно ознайомитися з екосистемою мобільної розробки, розуміти нативні API та навчитися оптимізувати продуктивність для мобільних пристроїв. Крім того, у React Native можуть бути складнощі з налагодженням та пошуком помилок через містку між JavaScript та нативним кодом.

Основною метою React Native є забезпечення безперешкодного досвіду розробки для створення мобільних додатків, що мають зовнішній вигляд та функціонал нативних додатків. Він прагне перенести переваги компонентної архітектури та декларативного програмування React у світ мобільної розробки. Завдяки використанню JavaScript, мови, яку широко володіють веб-розробники, React Native знижує бар'єри для входу в мобільну розробку та дозволяє веб-розробникам створювати потужні та продуктивні мобільні додатки.

Підсумовуючи, React Native пропонує потужний фреймворк для розробки кросплатформових мобільних додатків. Він поєднує переваги компонентної архітектури та декларативного програмування з можливістю використовувати єдину базу коду для розробки на різних платформах. Це дозволяє економити час і зусилля розробників, прискорювати процес розробки та забезпечувати нативний вигляд та функціонал мобільних додатків. Незважаючи на деякі обмеження, React Native залишається актуальним і популярним фреймворком в галузі мобільної розробки.

2.3.7. Фреймворк Expo

Expo [16] - це потужна платформа для розробки мобільних додатків, яка виникла в 2014 році з метою спростити та прискорити процес розробки

додатків для платформ Android та iOS. За цей час Expo зарекомендував себе як один з найпопулярніших інструментів для розробки мобільних додатків серед розробників усього світу.

Основна ідея Expo полягає в тому, щоб забезпечити розробникам зручний і простий спосіб створення мобільних додатків, не вимагаючи глибоких знань мов програмування, складностей налаштування та підтримки різних платформ. За допомогою Expo, розробники можуть швидко створювати додатки, використовуючи знайомі інструменти, такі як JavaScript та React Native, що дозволяє легко переносити код між платформами.

Однією з найважливіших компонентів Expo є Expo Go - власний додаток, що дозволяє розробникам переглядати та тестувати свої додатки безпосередньо на мобільних пристроях. Це унікальна можливість, оскільки розробники можуть відразу ж бачити результати своєї роботи без необхідності компіляції та встановлення додатків. Крім того, Expo Go дозволяє швидко та легко надсилати запрошення іншим людям для тестування створених додатків.

Окрім того, Expo надає Expo SDK - набір інструментів, бібліотек та API, які допомагають в створенні різноманітних функціональних можливостей для додатків. За допомогою Expo SDK, розробники можуть легко використовувати функції камери, геолокації, веб-сервісів, пуш-сповіщень та багато іншого, що значно розширює можливості додатків.

Значний інтерес викликає також Expo Client SDK, який дозволяє вбудовувати функціональність Expo в існуючі додатки, надаючи можливість використовувати Expo SDK у вже існуючому коді. Це дозволяє розробникам поетапно впроваджувати Expo у свої проекти, зберігаючи при цьому зручність та швидкість розробки.

Expo також підтримує різні платформи розгортання, включаючи Apple App Store та Google Play Store. Завдяки цьому, розробники можуть легко

опублікувати свої додатки та забезпечити їх доступність для мільйонів користувачів по всьому світу.

Наприкінці слід зазначити, що Expo продовжує активно розвиватись, додавати нові функції та поліпшення, щоб забезпечити розробникам ще більше можливостей. Завдяки своїй простоті, широкій функціональності та підтримці спільноти, Expo залишається одним з найпопулярніших інструментів для розробки мобільних додатків, що надає розробникам можливість швидко та ефективно реалізувати свої ідеї на мобільних платформах.

3. СТВОРЕННЯ ЗАСТОСУНКУ «GESUNDHEIT»

3.1. Середовище розробки Visual Studio Code

У сучасному світі програмування, де технологічні вимоги постійно зростають, розробникам стикаються з викликами і завданнями, пов'язаними з ефективністю, швидкістю та якістю розробки програмного забезпечення. Для впевненості у досягненні цих цілей важливим є використання відповідного розробницького середовища. Середовище розробки є невід'ємним помічником, надаючи розробникам необхідні інструменти, зручність та можливості для написання, тестування та налагодження коду. Воно створює простір, де ідеї перетворюються на дійсність, а розробники можуть концентруватися на досягненні своїх цілей.

VS Code [14], розроблений компанією Microsoft, є безкоштовним і відкритим середовищем розробки, яке володіє широким набором функцій та можливостей. Це інструмент, створений з метою забезпечити розробникам потужне середовище для зручного написання коду. Завдяки своїй швидкості та простоті використання, VS Code стає незамінним помічником у творчому процесі розробки програмного забезпечення. Забезпечуючи зручний інтерфейс та можливість розширення функціональності, це середовище робить процес розробки більш продуктивним і задовольняє потреби розробників у швидкості та масштабованості.

Історія створення VS Code почалася у 2011 році, коли команда розробників Microsoft почала працювати над проектом з відкритим кодом під назвою "Monaco". Їхнім бажанням було створити середовище розробки, яке було б не тільки потужним, але й легким для використання на різних платформах. У 2015 році Microsoft офіційно представила VS Code як самостійний продукт.

Програма заслужила популярність серед розробників завдяки його вражаючим функціям та зручному інтерфейсу. Однією з ключових особливостей є великі можливості налаштування та розширення. Завдяки масиву яких, розробники можуть налаштувати робоче середовище за своїми потребами, додавши необхідні функції і розширення.

VS Code підтримує широкий спектр мов програмування, включаючи JavaScript, TypeScript, Python, Java, C# та багато інших. Це робить його універсальним інструментом для розробки різноманітних проектів.

Інтерфейс VS Code простий і зрозумілий, з панелями і вкладками, що дозволяють зручно організовувати робоче середовище. Крім того, він має вбудовану підтримку систем керування версіями, редактор Git та інші інструменти для спільної роботи над проектами.

Visual Studio Code є середовищем розробки, яке налічує декілька значних переваг. Його потужний редактор коду забезпечує розробникам зручність та продуктивність під час роботи з кодом. Завдяки розширеному функціоналу, такому як підказки, автодоповнення та перевірка синтаксису, розробники можуть швидше та точніше писати код, уникати помилок та підвищувати ефективність своєї роботи.

Однією з ключових переваг Visual Studio Code є його велика екосистема розширень. Відкритий характер VS Code призвів до того, що велика спільнота розробників активно співпрацює, розвиває розширення та допомагає покращувати функціональність середовища. У магазині додатків Visual Studio Code розробники можуть знайти різноманітні розширення, які розширюють функціональність редактора і дозволяють налаштувати його під свої потреби. Це дозволяє розробникам вибрати саме ті інструменти, які необхідні для їх конкретного проекту або мови програмування.

У форумах, блогах та соціальних мережах розробників зі всього світу активно обговорюють нові функції, техніки розробки та надають один

одному підтримку. Цей взаємодія дозволяє розробникам вдосконалювати свої навички, вивчати нові підходи до програмування та вирішувати труднощі шляхом обміну знаннями.

Ще однією перевагою Visual Studio Code є його кросплатформеність. Він підтримує роботу на різних операційних системах, таких як Windows, macOS та Linux, що робить його доступним для широкого кола розробників. Розробники можуть використовувати одне й те ж середовище розробки на різних платформах без необхідності перекомпіляції або модифікації свого коду.

Visual Studio Code, як і будь-яке інше середовище розробки, має деякі недоліки. Одним з них є нестабільність під час роботи з великими проектами або при використанні багатьох розширень одночасно. Це може призводити до зниження продуктивності та затримок під час роботи з кодом. Хоча розробники постійно вдосконалюють цю проблему, вона може виникати в окремих випадках.

Інший недолік полягає у складнощах налаштування та конфігурування Visual Studio Code. Завдяки широкому спектру налаштувань та розширень, середовище розробки може вимагати деякого часу та досвіду для досягнення оптимальних налаштувань. Некоректна конфігурація може призводити до некоректної роботи редактора коду та втрати продуктивності.

Крім того, Visual Studio Code може виявлятися вимогливим до ресурсів комп'ютера, особливо при роботі з великими проектами або при використанні деяких розширень. Це може вплинути на продуктивність системи та призводити до зниження швидкості роботи з програмою. Рекомендується мати достатньо потужний комп'ютер для забезпечення оптимальної роботи Visual Studio Code.

Необхідність встановлення різних розширень для отримання повного функціоналу також може бути недоліком. Розробники можуть витрачати час

на пошук, вибір та налаштування розширень, що може забирати частину часу, яке можна було би витратити на сам процес розробки.

Незважаючи на ці недоліки, Visual Studio Code є потужним інструментом для розробки програмного забезпечення, який забезпечує зручність та продуктивність розробників. Розуміння цих недоліків допомагає розробникам планувати свою роботу, шукати оптимальні рішення та уникати можливих проблем.

На сьогоднішній день, Visual Studio Code займає визначну позицію на ринку середовищ розробки. Його широкий спектр функціональності, висока швидкодія та активна спільнота розробників забезпечують йому значну популярність. Visual Studio Code отримав велику кількість позитивних відгуків від розробників, які відзначають його зручний інтерфейс, розширену підтримку мов програмування та багатофункціональність.

У порівнянні з іншими середовищами розробки, Visual Studio Code володіє своїми унікальними перевагами. Він відмінно підходить для веб-розробки, забезпечуючи багатий набір інструментів для роботи з HTML, CSS та JavaScript. Його широко розповсюджена підтримка розширень дозволяє розробникам налаштувати середовище розробки під свої потреби та використовувати розширення сторонніх розробників для покращення продуктивності.

У порівнянні зі старшою версією - Visual Studio, Visual Studio Code є легшим, швидшим та більш простим у використанні. Він підтримує різні платформи, включаючи Windows, macOS та Linux, що дає можливість розробникам працювати на своєму улюбленому операційній системі.

Що стосується порівняння з іншими популярними середовищами розробки, Visual Studio Code конкурує з такими інструментами, як Atom, Sublime Text та IntelliJ IDEA. Він приваблює розробників своєю швидкодією,

розширеною функціональністю та активною спільнотою, що постійно вносить внески в його розвиток.

Visual Studio Code успішно протистоїть конкуренції на ринку середовищ розробки завдяки своїм унікальним можливостям, зручному інтерфейсу та активній спільноті розробників. Він набуває все більшої популярності і продовжує розвиватись, надаючи розробникам потужний інструмент для ефективної роботи.

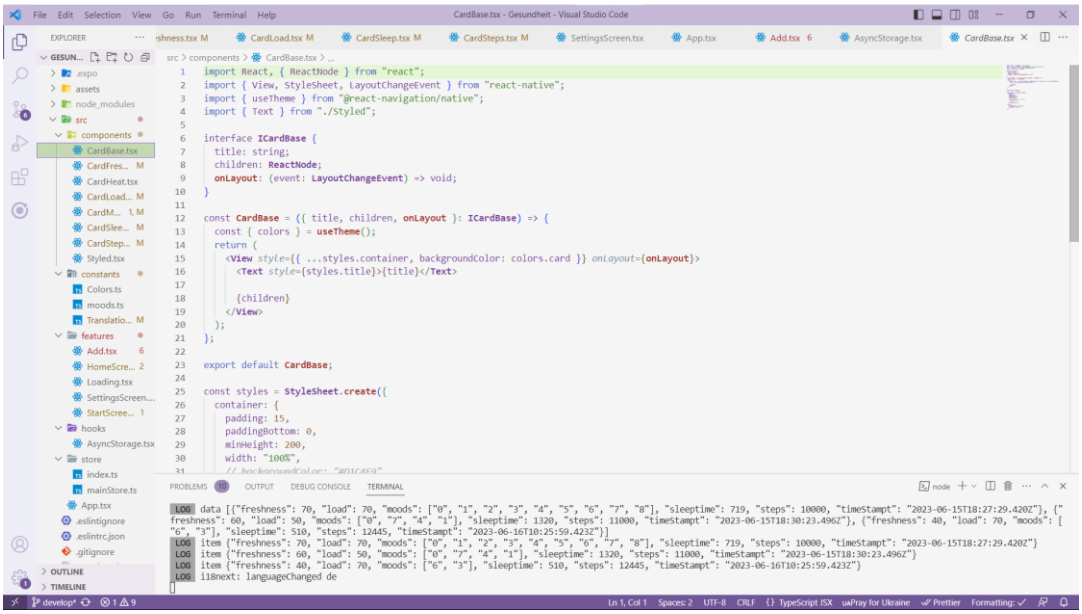

Рис. 3.1 Огляд програми Visual Studio Code

3.2. Опис асинхронного сховища

В сучасному програмуванні виникає необхідність ефективно та гнучко зберігати та обробляти дані. Існує підхід, який дозволяє забезпечити цю функціональність, даючи можливість виконувати операції з даними в асинхронному режимі. Цей підхід допомагає покращити продуктивність та забезпечує більш гнучку обробку даних у програмах.

Асинхронне сховище, яке застосовується у програмуванні, відіграє важливу роль у зберіганні даних та їх обробці. Цей механізм надає

можливість доступу до даних в асинхронному режимі, не блокуючи виконання інших операцій.

Однією з особливостей асинхронного сховища є можливість одночасно виконувати багато запитів до даних без затримок. Це забезпечує швидкодію та ефективність взаємодії з сховищем навіть при великому обсязі даних чи високій навантаженості.

Недоліками асинхронного сховища є складність в розробці та управлінні. Оскільки операції з даними відбуваються асинхронно, необхідно добре організувати асинхронний код і обробку помилок. Крім того, при зберіганні даних на диску можуть виникати проблеми зі швидкодією через обмеження вводу-виводу.

Проте, асинхронне сховище має свої переваги. Воно дозволяє ефективно працювати з даними, які потребують частого оновлення або синхронізації з іншими джерелами. Крім того, використання асинхронного сховища дозволяє покращити відповідність часу відгуку системи, що особливо важливо в ситуаціях, коли потрібно швидко обробляти та відправляти дані.

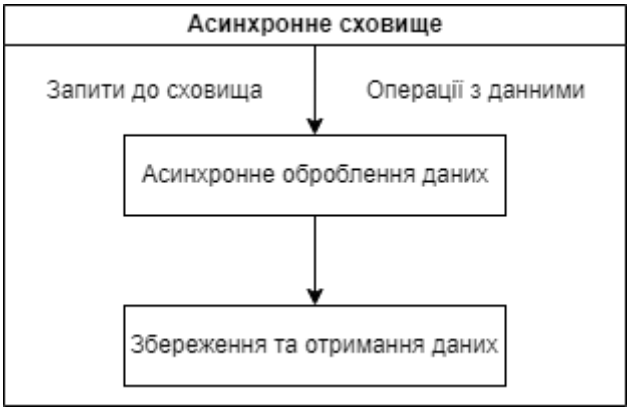

Рис. 3.2 Схема роботи асинхронного сховища

На схемі видно, що асинхронне сховище приймає запити та виконує операції з даними в асинхронному режимі. Це дозволяє зберігати та отримувати дані без блокування виконання інших операцій. Запити до

сховища обробляються в асинхронному обробленні даних, що дозволяє ефективно працювати з великими обсягами даних та забезпечує швидкодію взаємодії з сховищем.

Більш детально код можна побачити в додатку А.

3.3. Опис сесійного сховища

Сесійне сховище - це механізм у програмуванні, що використовується для зберігання даних протягом сесії роботи. Воно дозволяє зберігати інформацію, яка потрібна протягом перебування користувача на веб-сторінці або в додатку, і забезпечує доступ до неї між різними запитами або сторінками.

Однією з особливостей сесійного сховища є те, що дані, збережені в ньому, залишаються доступними протягом усієї сесії, поки користувач не закриє її або не вийде з додатку. Це дозволяє зберегти стан або прогрес користувача і використовувати його на різних сторінках або в різних запитах.

Однак, варто враховувати недоліки сесійного сховища. Наприклад, при великій кількості користувачів або високому навантаженні на сервер, сесійне сховище може викликати проблеми з продуктивністю, оскільки дані тримаються в оперативній пам'яті сервера. Крім того, якщо сервер перезавантажується або сесія закривається, дані в сесійному сховищі втрачаються.

Звичайно, що сесійне сховище має свої переваги. Воно дозволяє зручно зберігати дані, які потрібні на протязі всієї сесії, не повторюючи їх передачу між сторінками або запитами. Це забезпечує економію ресурсів і поліпшує швидкість дії програми. Крім того, сховище може бути використане для зберігання конфіденційної інформації, оскільки дані знаходяться на сервері і недоступні для користувачів.

Рис. 3.3 Схема роботи сесійного сховища

Дана схема показує, що сесійне сховище виконує запити до бази даних через сесійний менеджер. Сесійний менеджер відповідає за керування сесіями, включаючи створення, управління та знищення сесій. Запити до сховища передаються через сесійний менеджер, який взаємодіє зі збереженням та отриманням даних, що знаходяться в базі даних.

Можна зауважити, що сесійне сховище служить посередником між додатком та базою даних, забезпечуючи управління сесіями та зручний доступ до даних. Сесійний менеджер забезпечує ізоляцію сесій та взаємодію з фізичним сховищем даних.

Більш детально код можна побачити в додатку Б.

3.4. Огляд інтерфейсу користувача

Завжди, коли ми вперше зустрічаємося з новим додатком чи сервісом, перший екран має велике значення. Він допомагає нам створити персоналізований досвід, налаштувати параметри та встановити особисті налаштування. Початковий екран є місцем, де ми можемо ввести важливі особисті дані і налаштувати додаток під свої потреби. Він створює основу

для подальшої взаємодії та допомагає нам отримати максимальну користь від додатку.

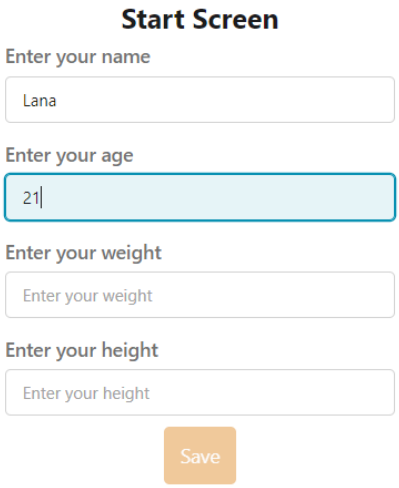

Рис. 3.4 Стартовий екран

При першому вході в додаток користувач зустрічає стартовий екран, який представляє собою вхідну точку користувача і містить надпис "Стартовий екран" для ідентифікації. На екрані розташовані чотири поля вводу, де користувач може ввести свої персональні дані, такі як: ім'я, вік, вагу та зріст. Для забезпечення точності та коректності даних, проводиться перевірка на валідність, зокрема перевірка використання лише цифр для віку, ваги та зросту. Додатково, на стартовому екрані присутня кнопка "Зберегти", проте за замовчуванням вона заблокована допоки всі обов'язкові поля не будуть заповнені. Всі ці функціональні елементи стартового екрана спроектовані з метою забезпечення зручного та надійного введення персональних даних користувача, зокрема для подальшої обробки та використання у функціональних можливостях додатку.

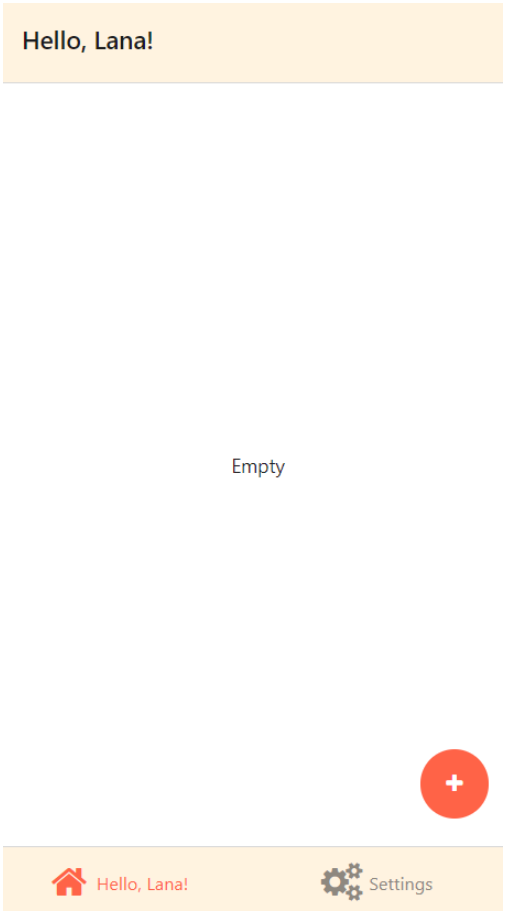

Рис. 3.5 Домашня сторінка

Після введення персональних даних, ми потрапляємо на екран з привітанням, який відображає наше ім'я і вітання обраною мовою. На цьому екрані також видно повідомлення "Даних поки немає", а нижче розташована невелика кругла кнопка, яка дозволяє нам додавати дані про здоров’я до застосунку. Крім того, ми можемо переключатись між двома вкладками - "Домашній екран" і "Налаштування". Цей екран є початковим пунктом для збору та управління нашими персональними даними. Він надає нам можливість збирати, оновлювати і контролювати нашу інформацію в зручний спосіб. За допомогою цього екрану ми можемо активно залучатись до додатку, налаштовувати його під свої потреби та отримувати користь зі збору персональних даних.

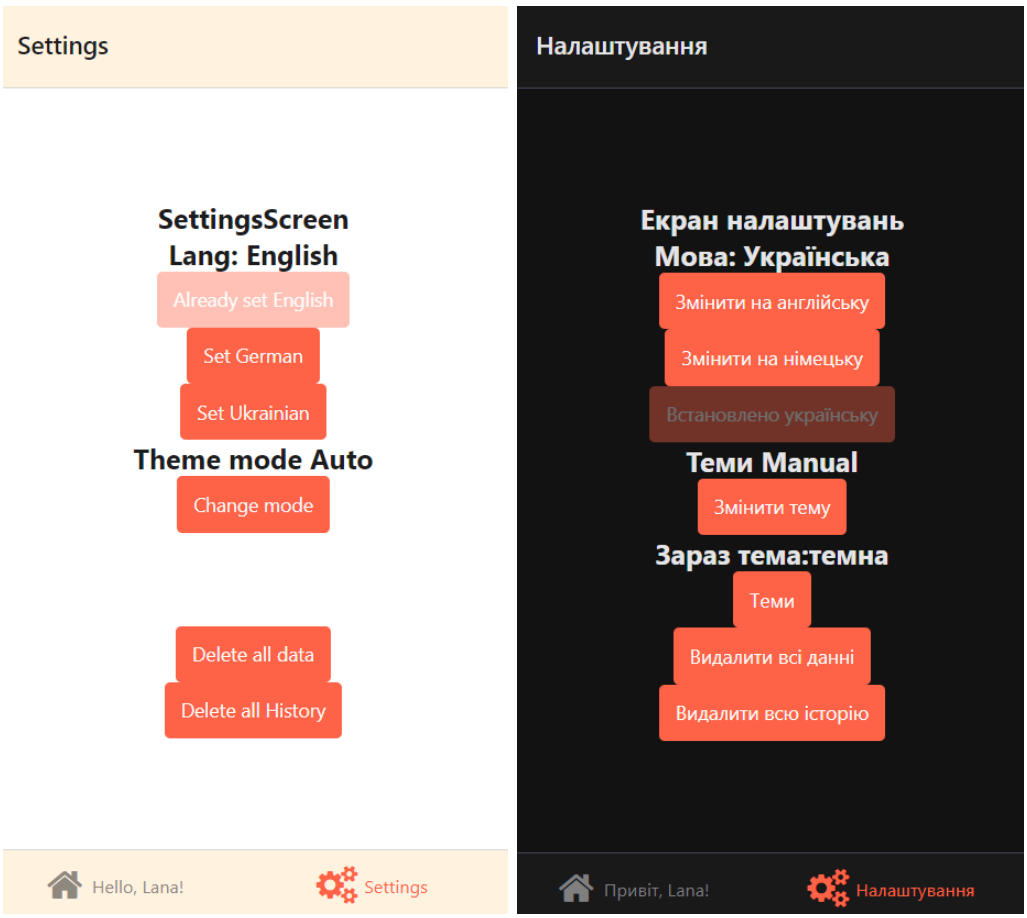

Рис. 3.6 Екран налаштувань

Вкладка налаштувань надає гнучкість і персоналізацію для кожного користувача, дозволяючи їм налаштувати додаток так, як вони бажають і відповідно до їхніх індивідуальних вимог. Також дозволяючи користувачам створювати комфортне та персоналізоване робоче середовище.

Зміна мови (англійська, німецька, українська) та вибір теми (темна або світла) дозволяють адаптувати інтерфейс до власних вподобань і забезпечити комфортну взаємодію з програмою.

Більш того, налаштування також дозволяють користувачам управляти своїми даними. Вони можуть видаляти щоденні записи або повністю очищати всі дані, забезпечуючи контроль та приватність їхньої інформації. Ця функціональність дає можливість адаптувати програму до змінюваних потреб користувачів і забезпечити гнучкість в управлінні даними.

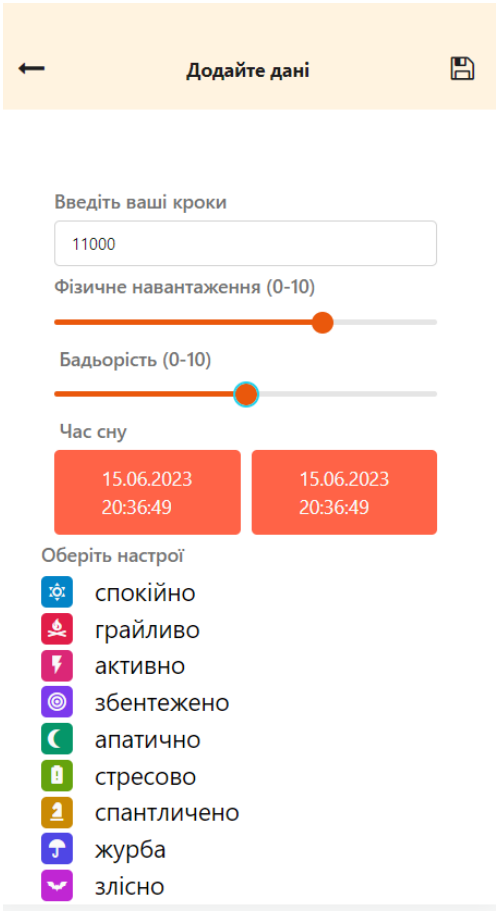

Рис. 3.7 Додавання щоденних даних

При переході на екран додавання щоденних даних, користувач отримує зручну можливість введення важливої інформації щодо свого фізичного стану та самопочуття. Цей екран включає ряд важливих полів, які дозволяють точно відображати щоденну активність та розподіл часу.

Користувач може ввести кількість кроків, що відображає його фізичну активність протягом дня. Шкала фізичного навантаження та бадьорості дозволяє оцінити загальний рівень енергії та самопочуття. Крім того, користувач може вказати час початку і закінчення сну, що допомагає відстежувати режим сну та аналізувати якість відпочинку.

Окрім цього, на екрані додавання щоденних даних є можливість вибору настрою зі списку. Це дозволяє користувачеві відображати своє емоційне становище або настрій, що створює повнішу картину щоденного самопочуття.

Можливість вводити та відстежувати важливі показники, що відображають їх фізичну активність, енергію, режим сну та емоційний стан. Це допомагає зрозуміти вплив щоденних факторів на загальне самопочуття та покращити якість життя.

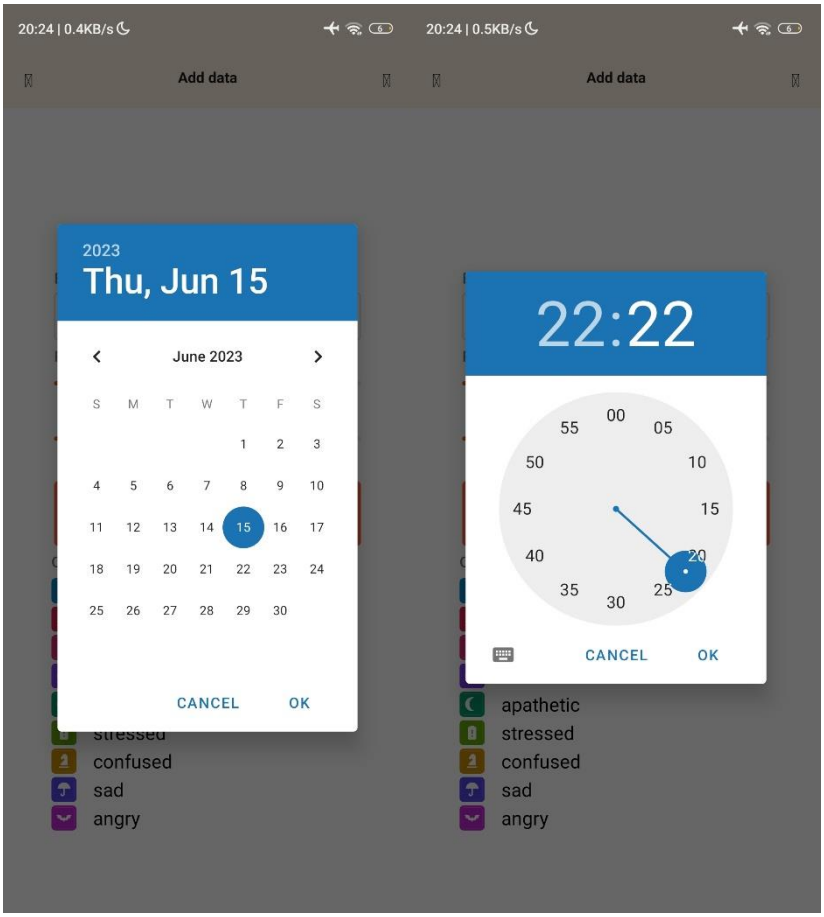

Рис. 3.8 Ввід початку/закінчення сну

На екрані додавання щоденних даних, коли користувач обирає час початку і закінчення сну, при натисканні яких відкриваються додаткові елементи для точного вводу цих значень.

При натисканні спочатку відкривається календарний інтерфейс, де користувач може обрати дату, коли почався або закінчився сон. Він може прокручувати календар, переглядати дні та місяці, щоб точно обрати відповідну дату.

Після вибору дати відкривається годинниковий інтерфейс. Тут користувач може точно вказати години і хвилини початку і закінчення сну. Він може обертати стрілки годинника або використовувати цифрову клавіатуру для введення потрібного часу. Для запобігання неточних даних через неуважність або помилкові введення, система встановлює перевірку, що час закінчення сну не може бути раніше за час початку. Це допомагає уникнути некоректних показників і забезпечує точність введених даних.

Такий метод вводу початку і закінчення сну надає користувачеві зручну можливість встановлювати точний час, коли він лягає спати та прокидається. Це допомагає створити точний щоденний графік сну, який сприяє покращенню якості відпочинку та загальному самопочуттю користувача.

З отриманих даних застосунок вимальовує різноманітні дані, які позначені на рисунку 3.9:

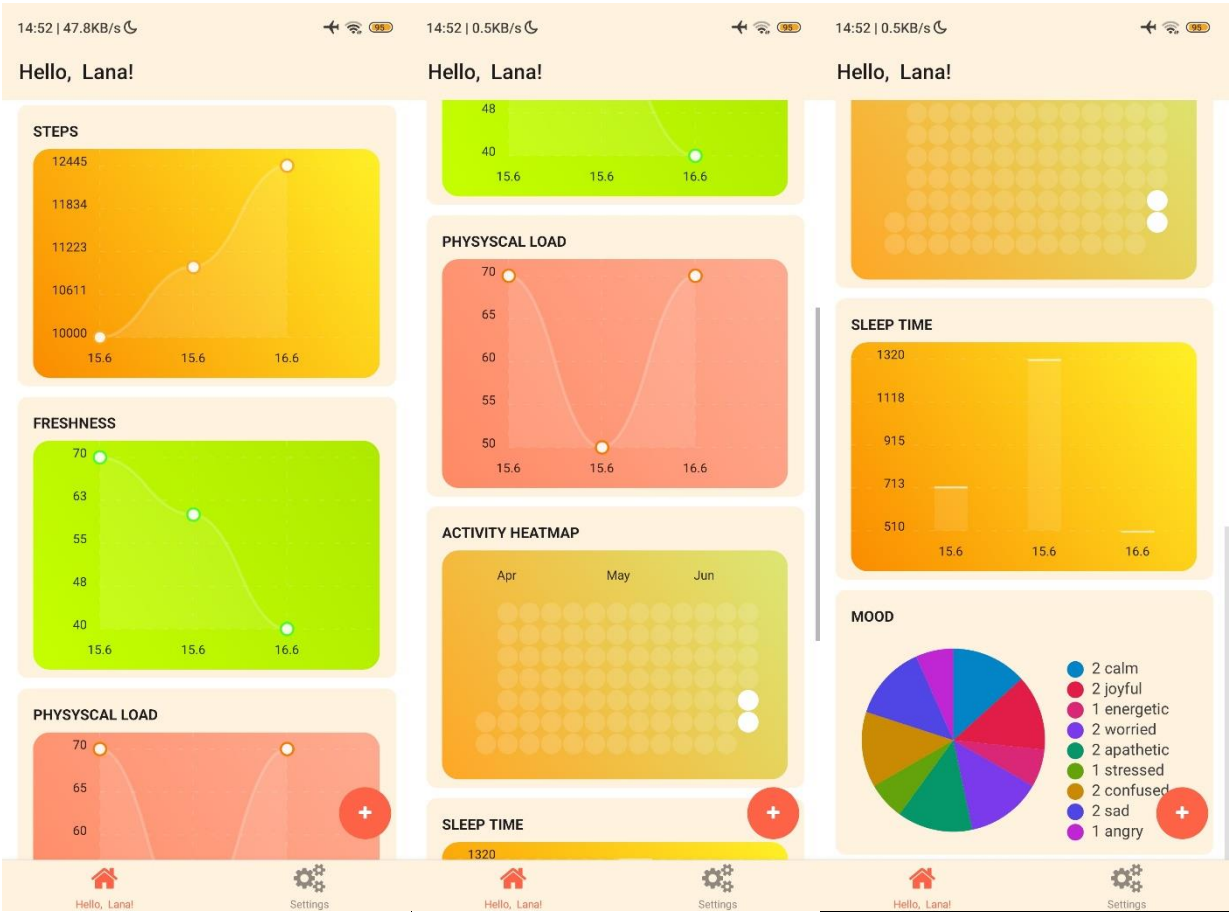

Рис. 3.9 Графіки

З накопичених даних вибудовуються графіки, які відображають інформацію про кроки, бадьорість, фізичне навантаження, активність у застосунку та настрої. Графіки кроків, бадьорості та фізичного навантаження представлені у вигляді плавних округлих ліній, що дозволє порівнювати поточні значення з найбільшими та найменшими показниками. Це допомагає користувачу оцінити свою активність та покращити свої досягнення в цих областях. Свіжіші дані розташовуються праворуч, забезпечуючи актуальну оцінку стану та змін.

Присутній графік, що відображає дні, коли були додані або пропущені дані. Він має форму календаря з круглими порожніми кулями, які поступово заповнюються. Цей графік дозволяє користувачеві переглядати свою послідовність внесення даних та визначати, які дні можуть потребувати більшої уваги або перевірки. Змінення кольору порожніх місць на білий, коли дані були додані в цей день, надає візуального підтвердження заповнення і допомагає відслідковувати консистентність введення даних.

Графік сну, представлений у вигляді прямокутних стовпчиків, не з'єднаних між собою, надає користувачеві детальну інформацію про тривалість сну в певні дні. Кожен стовпчик відповідає одному дню, а його висота відображає тривалість сну у цей день. Така візуальна представленість дозволяє легко спостерігати зміни в тривалості сну протягом періоду часу і встановлювати зв'язок між якістю сну та загальним самопочуттям.

Аналізуючи графік сну, користувач може побачити, які дні відзначаються тривалим або недостатнім сном. Високі стовпчики свідчать про достатню тривалість сну, тоді як низькі стовпчики вказують на недостатню кількість годин сну. Такі візуальні зміни дозволяють користувачеві оцінити ефективність свого сну і зробити відповідні зміни в режимі сну, якщо це необхідно.

Більш того, графік сну також допомагає користувачеві встановити зв'язок між якістю сну і загальним самопочуттям. Наприклад, якщо

користувач помічає, що дні з недостатнім сном співпадають зі значним погіршенням самопочуття, це може вказувати на важливість покращення якості та тривалості сну для поліпшення загального стану здоров'я.

Графік настрою, представлений у круглій формі з відсотковим співвідношенням, є важливим інструментом для оцінки емоційного стану користувача та відстеження змін у його настроях. Цей графік дозволяє візуально представити розподіл різних настроїв і встановити їх взаємний вплив.

Бічна панель графіку містить перелік можливих настроїв, кожен з яких має свій унікальний колір. Користувач може побачити цей перелік і візуально зрозуміти, який колір відповідає кожному настрою. Крім того, поряд з кольорами на панелі вказана кількість разів, коли було обрано певний настрій. Ця інформація дозволяє користувачеві аналізувати, які настрої переважають у певний період часу.

Завдяки графіку настрою користувач може легко бачити динаміку своїх емоційних станів. Він може відстежувати, які настрої виникають найчастіше і які переважають. Це дозволяє зробити зв'язок між настроями та зовнішніми факторами або подіями, що може сприяти більшому розумінню себе і своїх емоційних реакцій.

Графік настрою є цінним інструментом для самоспостереження та самоаналізу. Він допомагає користувачеві бачити індивідуальні тенденції в своїх емоціях, встановлювати зв'язки між настроями та іншими аспектами життя, такими як фізичне здоров'я, соціальні взаємини або рівень стресу. Цей графік дозволяє зробити свідомий вплив на свої емоції та покращити загальний емоційний стан, встановлювати зв'язки між настроями та іншими аспектами життя, а також спостерігати за змінами у своєму емоційному стані з часом. Він стимулює саморефлексію, самоаналіз та саморегуляцію, що може сприяти покращенню загального благополуччя та якості життя користувача.

Графіки у застосунку пропонують користувачам цінний інструмент для самоспостереження та самозростання. Шляхом візуалізації своїх даних у вигляді графіків, користувачі можуть виявляти тенденції, знаходити зв'язки та розуміти, як їхні дії та звички впливають на їхнє майбутнє. Це дає можливість приймати усвідомлені рішення, коригувати негативні звички, підтримувати позитивний настрій та стимулювати активний спосіб життя. Візуалізація надає мотивацію для досягнення поставлених цілей та вдосконалення фізичного та емоційного благополуччя, сприяючи тим самим позитивному впливу на майбутнє.

ВИСНОВКИ

В процесі розробки дипломної роботи було реалізовано задуманий застосунок для відстеження стану здоров'я, з використанням потужного та популярного інструменту - середовища розробки Visual Studio Code.

На підставі ретельного аналізу та дослідження вимог та потреб користувачів була розроблена оптимальна архітектура мобільного додатку. Ця архітектура була ретельно спроектована з метою забезпечення ефективної та зручної роботи додатку, забезпечуючи високу продуктивність та задоволення користувачів.

Також, згідно з проектними рішеннями, було розроблено сам додаток на основі обраного фреймворку React. Використання React дозволило забезпечити гнучкість, швидкість розробки та переносимість додатку на різні платформи. Крім того, використання TypeScript як мови програмування сприяло забезпеченню високої якості коду, його читабельності та підтримці типізації, що сприяє зменшенню помилок та полегшує роботу команди розробників.

У результаті виконання дипломної роботи було досягнуто успішного створення функціонального та ефективного додатку для відстеження стану здоров'я, який може знайти своє місце у сфері здоров'я та покращення рівня життя. Розроблений застосунок має інтуїтивний і зручний інтерфейс, що дозволяє користувачеві з легкістю вводити свої дані та отримувати зручний перегляд статистики та графіків. Застосунок був реалізований з врахуванням сучасних технологій та найкращих практик розробки програмного забезпечення, що гарантує його стабільну та надійну роботу.

Застосунок має стартовий екран з можливістю реєстрації, а також розширений набір функцій, які дозволяють відстежувати стан здоров'я, включаючи кількість кроків, час сну, настрій та інші параметри. Також

пропонуються зручні налаштування, які дозволяють користувачу змінювати тему та мову інтерфейсу.

Виконання дипломної роботи підтверджує успішність реалізації проекту з використанням відповідних технологій та розробки оптимальної архітектури. Результати отриманого додатку свідчать про його потенціал та перспективи для використання в реальних сценаріях відстеження і покращення стану здоров'я.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to client-side Frameworks - learn web development: MDN - URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction (дата звернення 17.04.2023)

2. Best JavaScript Frameworks For 2023. URL: https://www.lambdatest.com/blog/best-javascript-frameworks/ (дата звернення 11.04.2023)

3. "JavaScript: The Good Parts" - Дуглас Крокфорд.

4. Документація JavaScript - URL: https://developer.mozilla.org/en-US/docs/Web/ JavaScript/Guide (дата звернення 13.04.2023)

5. Пірсон, Р. (2017). "Оволодіння патернами проектування на JavaScript". Packt Publishing - URL: https://www.packtpub.com/product/learningjavascript-design-patterns-third-edition/978178588262

6. Хавербеке, М. (2012). "Елегантний JavaScript: Сучасне введення в програмування" - URL: https://eloquentjavascript.net/ (дата звернення 19.04.2023)

7. React Navigation - Routing and navigation for Expo and React Native apps - URL: reactnavigation.org/ (дата звернення: 12.04.2023)

8. Документація React Native - URL: https://reactnative.dev/docs (дата звернення 09.04.2023)

9. "React Native in Action" - Кристофер Некке, Влад Бец.

10. Mardan A. React quickly : painless web apps with React, JSX, Redux, and GraphQL. Shelter Island, NY: Manning Publications Co, 2017,

11. React Native - Learn once, write anywhere - URL: https://reactnative.dev/docs/getting-started (дата звернення: 04.04.2023)

12. Документація TypeScript - URL: https://www.typescriptlang.org/docs/ (дата звернення 13.04.2023)

13. "TypeScript Deep Dive" - Бас Беккерт.

14. vscode.dev - Visual Studio Code for the Web - URL: https://code.visualstudio.com/blogs/2021/10/20/vscode-dev (дата звернення: 02.04.2023)

15. Офіційний сайт Visual Studio - URL: https://code.visualstudio.com/ (дата звернення 03.04.2023)

16. Документація Expo - URL: https://docs.expo.io/ (дата звернення 09.04.2023)

17. Expo Application Services (EAS) - URL: https://expo.dev/eas (дата звернення: 22.04.2023)

18. Expo Go – Expo documentation - URL: https://docs.expo.dev/getstarted/expo-go/ (дата звернення: 30.04.2023)

19. Expo - Create amazing apps that run everywhere - URL: https://docs.expo.dev/get-started/expo-go/ (дата звернення: 29.04.2023)

ДОДАТКИ

ДОДАТОК А. КЛАС АСИНХРОННОГО СХОВИЩА

```
import AsyncStorage from "@react-native-async-storage/async-storage";

export enum UserPrefs {
  language = "language",
  isLightTheme = "isLightTheme",
  isThemeModeManual = "isThemeModeManual",
  isStarted = "isStarted",
  dataHistory = "dataHistory",
  user = "user",
}

export default class Storage {
  setData = async (key: UserPrefs, value: string) => {
    try {
      await AsyncStorage.setItem(key, value);
    } catch (e) {
      console.warn("Data saving error");
    }
  };
  setObject = async (key: UserPrefs, value: object) => {
    try {
      const jsonValue = JSON.stringify(value);
      await AsyncStorage.setItem(key, jsonValue);
    } catch (e) {
      console.warn("Data saving error");
    }
  };

  getData = async (key: UserPrefs) => {
    try {
      const value = await AsyncStorage.getItem(key);
      return value;
    } catch (e) {
      console.warn("Data reading error");
    }
  };
  getObject = async (key: UserPrefs) => {
    try {
      const jsonValue = await AsyncStorage.getItem(key);
      return jsonValue !== null ? JSON.parse(jsonValue) : null;
    } catch (e) {
      console.warn("Data reading error");
    }
  };
}
```

ДОДАТОК Б. СЕСІЙНЕ СХОВИЩЕ

```
import { createSlice } from "@reduxjs/toolkit";
import Storage, { UserPrefs } from "../hooks/AsyncStorage";

interface IUser {
  name: string | undefined;
  age: number | undefined;
  weight: number | undefined;
  height: number | undefined;
}

export interface IHistoryItem {
  load: number;
  moods: string[];
  freshness: number;
  steps: number;
  timeStampt: string;
  sleeptime: number;
}

interface IStore {
  isLightTheme: boolean | undefined;
  isThemeModeManual: boolean | undefined;
  isStarted: boolean | undefined;
  user: IUser | undefined;
  dataHistory:
    | {
        data: IHistoryItem[];
      }
    | undefined;
  isWritedToday: boolean | undefined;
}

const getInitialState = (): IStore => {
  return {
    isLightTheme: undefined,
    isThemeModeManual: undefined,
    isStarted: undefined,
    dataHistory: undefined,
    isWritedToday: undefined,
    user: undefined,
  };
};

const storage = new Storage();

export const dataSlice = createSlice({
  name: "userSlice",
```

```
  initialState: getInitialState(),
  reducers: {
    setIsLightTheme: (state, action) => {
      state.isLightTheme = action.payload;
      storage.setData(UserPrefs.isLightTheme, action.payload ? "true" : "");
    },
    setIsThemeModeManual: (state, action) => {
      state.isThemeModeManual = action.payload;
      storage.setData(UserPrefs.isThemeModeManual, action.payload ? "true" : "");
    },
    setLanguage: (state, action) => {
      storage.setData(UserPrefs.language, action.payload);
    },
    setIsStarted: (state, action) => {
      state.isStarted = action.payload;
      storage.setData(UserPrefs.isStarted, action.payload ? "true" : "");
    },

    setHistoryFromStore: (state, action) => {
      state.dataHistory = action.payload.dataHistory;
      state.isWritedToday = action.payload.isWritedToday;
    },

    setDataHistoryNew: (state, action) => {
      let data;

      const date = new Date().toISOString();
      if (state.dataHistory)
        data = {
          data: [...state.dataHistory.data, { ...action.payload, timeStampt: date
}],
        };
      else
        data = {
          data: [{ ...action.payload, timeStampt: date }],
        };

      state.dataHistory = data;

      storage.setObject(UserPrefs.dataHistory, data);
    },
    deleteData: (state) => {
      const init = getInitialState();

      state.isLightTheme = init.isLightTheme;
      state.isThemeModeManual = init.isThemeModeManual;
      state.isStarted = false;
      state.dataHistory = init.dataHistory;

      storage.setData(UserPrefs.isStarted, "");
      storage.setData(UserPrefs.dataHistory, "");
```

```
    },

    deleteHistory: (state) => {
      const init = getInitialState();

      state.dataHistory = init.dataHistory;

      storage.setData(UserPrefs.dataHistory, "");
    },

    setUser: (state, action) => {
      state.user = typeof action.payload === "string" ?
JSON.parse(action.payload) : action.payload;

      storage.setObject(UserPrefs.user, action.payload);
    },
  },
});

export const {
  setIsLightTheme,
  setIsThemeModeManual,
  setLanguage,
  setIsStarted,
  deleteData,
  deleteHistory,
  setHistoryFromStore,
  setDataHistoryNew,
  setUser,
} = dataSlice.actions;

export default dataSlice.reducer;
```