Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ «COACHNOTE»

DEVELOPMENT OF THE MOBILE APPLICATION «COACHNOTE»

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи ІПЗс-31
Мельник Сергій Валерійович
_____
(підпис)

Керівник:
к.т.н., доцент
Повстяна Юлія Славомирівна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» червня 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ К. Лищина
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Мельнику Сергію Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *розробка мобільного додатку «CoachNote»*

Керівник роботи: *к.т.н., доцент, Повстяна Юлія Славомирівна*
затверджені наказом закладу вищої освіти від «*30*» *грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу: *17 рисунків; 8 лістингів; 1 таблиця.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Повстяна Ю. С.*		
Специфікації вимог до розробленої системи	*Повстяна Ю. С.*		
Розробка об'єкта проєктування	*Повстяна Ю. С.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	6,6 %		
Академічна доброчесність	*Ящук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	вик
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	вик
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	вик
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	вик
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	вик
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	вик
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	вик

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

Керівник кваліфікаційної роботи _____ (підпис) *Повстяна Ю. С.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу бакалавра

Здобувач вищої освіти:
Мельник Сергій Валерійович

Тема: Розробка мобільного додатку «CoachNote»

Коротка характеристика кваліфікаційної роботи: Робота студента демонструє глибоке розуміння теоретичних аспектів розробки мобільних додатків, використовуючи сучасні технології, такі як Expo та React Native. Студент провів ґрунтовний аналіз існуючих рішень, що дозволило чітко спланувати архітектуру та функціонал додатку. Теоретичні знання успішно застосовано для вирішення практичних задач, що свідчить про високий рівень підготовки.

Самостійні розробки і пропозиції автора: Студент проявив значну самостійність у всіх етапах розробки проекту. Від проектування інтерфейсу до впровадження функціоналу, всі завдання були виконані самостійно. Особливо варто відзначити інноваційні рішення для оптимізації авторизації користувачів та забезпечення безпеки даних. Розроблені тест кейси також є авторськими та відображають комплексний підхід до забезпечення якості продукту.

Практичне значення роботи: Мобільний додаток CoachNote має високу практичну значимість. Він може бути використаний у спортивних клубах для ефективного ведення тренувальних нотаток та управління даними спортсменів, забезпечуючи зручність та функціональність. Впровадження цього продукту здатне підвищити організаційний рівень тренувального процесу та покращити результати спортсменів.

Загальний висновок:
Роботу виконано у повному обсязі, у відповідності із завданням.Здобувач освіти Мельник Сергій Валерійович заслуговує позитивну оцінку.

Рецензент: Ліщина Валерій Олександрович к.т.н, доц
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«7» червня 2024р.

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Здобувач вищої освіти **Мельник Сергій Валейрійович**
(прізвище, ім'я, по батькові)

група ІПЗс-31

Тема кваліфікаційної роботи

Керівник к.т.н., доц. Повстяна Ю. С.

Актуальність теми. У сучасному світі технологічного прогресу мобільні додатки стали невід'ємною частиною повсякденного життя, проникаючи в різні сфери діяльності людини. Однією з таких сфер є спортивне тренування та фітнес. З огляду на зростаючу популярність здорового способу життя, тренери все частіше звертаються до цифрових рішень для оптимізації своїх професійних завдань. Мобільний додаток для тренерів «CoachNote» є саме таким рішенням, яке може значно спростити та вдосконалити процес управління тренувальним процесом.

Об'єкт дослідження є мобільні додатки для управління тренерами своїх тренувань.

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи. Робота студента демонструє глибоке розуміння теоретичних аспектів розробки мобільних додатків, використовуючи сучасні технології, такі як Expo та React Native. Студент провів ґрунтовний аналіз існуючих рішень, що дозволило чітко спланувати архітектуру та функціонал додатку. Теоретичні знання успішно застосовано для вирішення практичних задач, що свідчить про високий рівень підготовки.

Студент проявив значну самостійність у всіх етапах розробки проекту. Від проектування інтерфейсу до впровадження функціоналу, всі завдання були виконані самостійно. Мобільний додаток CoachNote має високу практичну значимість. Він може бути використаний у спортивних клубах для ефективного ведення тренувальних нотаток та управління даними спортсменів, забезпечуючи зручність та функціональність. Впровадження цього продукту здатне підвищити організаційний рівень тренувального

процесу та покращити результати спортсменів.

Загальний висновок

Роботу виконано у повному обсязі у відповідності до поставлених завдань, а здобувач Мельник С. В. заслуговує на оцінку «відмінно».

Керівник _____ **Повстяна Ю.С.**
(підпис) (прізвище, ім'я, по батькові)

«7» червня 2024 р.

АНОТАЦІЯ

Мельник С. В. Розробка мобільного додатку «CoachNote». Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

У першому розділі здійснено аналіз сучасного стану проблеми розробки комп'ютерних ігор і сформульовано завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено та протестовано мобільний додаток для тренерів «CoachNote». У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості мови програмування JavaScript та фреймворка для мультиплатформенної розробки – ReactNative.

Ключові слова: frontend, javascript, react-native, мобільна розробка, мобільний додаток, мобільний додаток для тренерів.

ABSTRACT

Melnyk S. V. Development of the mobile application "CoachNote". Manuscript.

Bachelor's qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions and proposals, a list of used sources and annexes.

In the first chapter, an analysis of the current state of the problem of developing computer games was carried out and the task was formulated. The second section defines the requirements for the developed system, as well as means, methods and development algorithms. In the third chapter, was developed and tested the mobile application for coaches "CoachNote" is developed. The conclusions summarize the information presented in the previous parts. The development results demonstrate the possibilities of features of the JavaScript programming language and framework for multi-platform development – React Native.

Keywords: frontend, javascript, react-native, mobile development, mobile app, mobile app for coaches.

ЗМІСТ

ВСТУП

Актуальність роботи. У сучасному світі технологічного прогресу мобільні додатки стали невід'ємною частиною повсякденного життя, проникаючи в різні сфери діяльності людини. Однією з таких сфер є спортивне тренування та фітнес. З огляду на зростаючу популярність здорового способу життя, тренери все частіше звертаються до цифрових рішень для оптимізації своїх професійних завдань. Мобільний додаток для тренерів «CoachNote» є саме таким рішенням, яке може значно спростити та вдосконалити процес управління тренувальним процесом.

Основною метою кваліфікаційної роботи бакалавра є розробка мобільного додатку для тренерів – зручного та функціонального інструменту, який дозволить тренерам ефективно планувати, проводити та аналізувати тренування своїх клієнтів. Додаток забезпечить можливість створення індивідуальних тренувальних програм, відстеження прогресу клієнтів, а також комунікації між тренером і клієнтом у реальному часі.

Зручність і доступність мобільних додатків роблять їх ідеальним інструментом для тренерів, які прагнуть покращити свої професійні навички та підвищити якість послуг, що надаються. Додаток може використовуватися як у великих спортивних клубах, так і незалежними тренерами, що забезпечує йому широку аудиторію користувачів.

Розробка мобільного додатку для тренерів передбачає врахування багатьох аспектів, включаючи інтуїтивно зрозумілий інтерфейс, високу продуктивність, безпеку даних, а також інтеграцію з іншими популярними сервісами та додатками. Важливим аспектом є також можливість налаштування додатку під індивідуальні потреби кожного тренера та його клієнтів.

Завданням на дану роботу є:

— продумати основний функціонал та сучасний, інтуїтивно зрозумілий інтерфейс;

- підібрати додаткові бібліотеки для роботи з запитами і формами;
- налаштувати та розгорнути додаток локально;
- спроектувати структуру проекту;
- написати код для всіх запланованих модулів;
- протестувати мануально додаток.

Об’єктом роботи є мобільний додаток для тренерів.

Предметом кваліфікаційної роботи бакалавра є процес розробки мобільного додатку.

У даній роботі буде детально розглянуто процес розробки мобільного додатку для тренерів, починаючи від аналізу вимог користувачів і закінчуючи тестуванням та впровадженням кінцевого продукту. Особлива увага буде приділена технічним аспектам розробки, методам забезпечення безпеки та конфіденційності даних, а також принципам проектування користувацького інтерфейсу.

Таким чином, мобільний додаток для тренерів є актуальним та перспективним проектом, який має потенціал значно покращити якість тренувального процесу та сприяти розвитку фітнес-індустрії в цілому.

РОЗДІЛ 1
АНАЛІЗ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ТРЕНЕРІВ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз мобільних додатків в області тренінгу

Мобільні додатки стали невід'ємною частиною нашого повсякденного життя, охоплюючи практично всі сфери діяльності. У сфері тренінгу та фітнесу ця потреба стала також актуальною.

Тренери потребують інструментів для ефективного управління своїми клієнтами та тренуваннями. Це включає в себе можливість ведення обліку клієнтів, планування та моніторинг тренувань, відстеження прогресу та здійснення комунікації з клієнтами.

Додаток повинен включати календар тренувань з можливістю додавання подій та нагадувань. Важливою функцією є збереження особистої інформації про клієнтів, їх цілей, обмін повідомленнями, створення індивідуальних тренувальних планів та відстеження прогресу. Також додаток має забезпечувати простий та зручний інтерфейс для взаємодії користувачів з тренерами, підтримувати інтеграцію з іншими фітнес-додатками та носимими пристроями для збору даних про фізичну активність. Це дозволить підвищити мотивацію клієнтів, зробити тренування більш ефективними та персоналізованими, а також полегшить моніторинг і аналіз результатів тренувань.

В процесі аналізу наткнувся на декілька додатків-аналогів.

Trainingpeaks – це фітнес-додаток для спортсменів, які тренують витривалість (як початківців, так і професіоналів). Додаток включає основний календар тренувань з можливістю додавання подій та нагадувань, функції збереження особистої інформації про клієнтів, їх цілей, обмін повідомленнями, створення індивідуальних тренувальних планів та відстеження прогресу. На рисунку 1.1 зображено головну сторінку додатку Trainingpeaks.

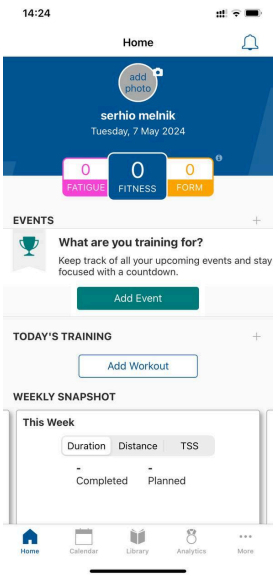

Рисунок 1.1 – Головна сторінка додатку Trainingpeaks [1]

MyFitnessPal – цей додаток спеціалізується на веденні харчового щоденника та статистики з харчування. Він також інтегрується з додатками для тренувань, такими як Fitbit, щоб забезпечити повний облік активності та калорійного балансу. На рисунку 1.2 зображено один з екранів додатку.

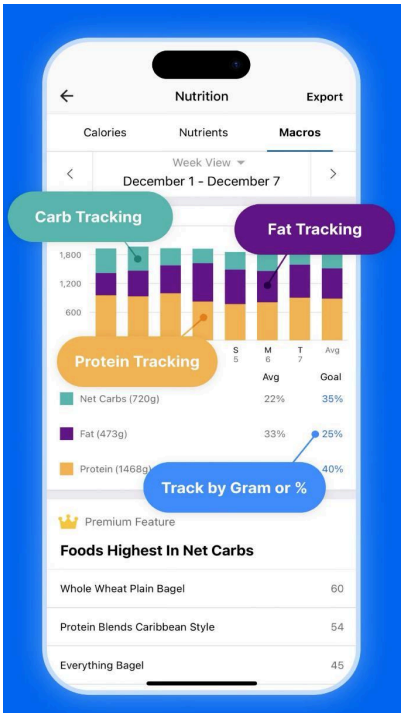

Рисунок 1.2 – Один з екранів додатку MyFitnessPal [2]

Trainerize – цей додаток призначений спеціально для тренерів, які працюють з клієнтами віддалено. Він має всі необхідні функції для планування тренувань, ведення обліку прогресу, обміну повідомленнями та збереження особистих даних клієнтів. На рисунку 1.3 зображено один з екранів додатку.

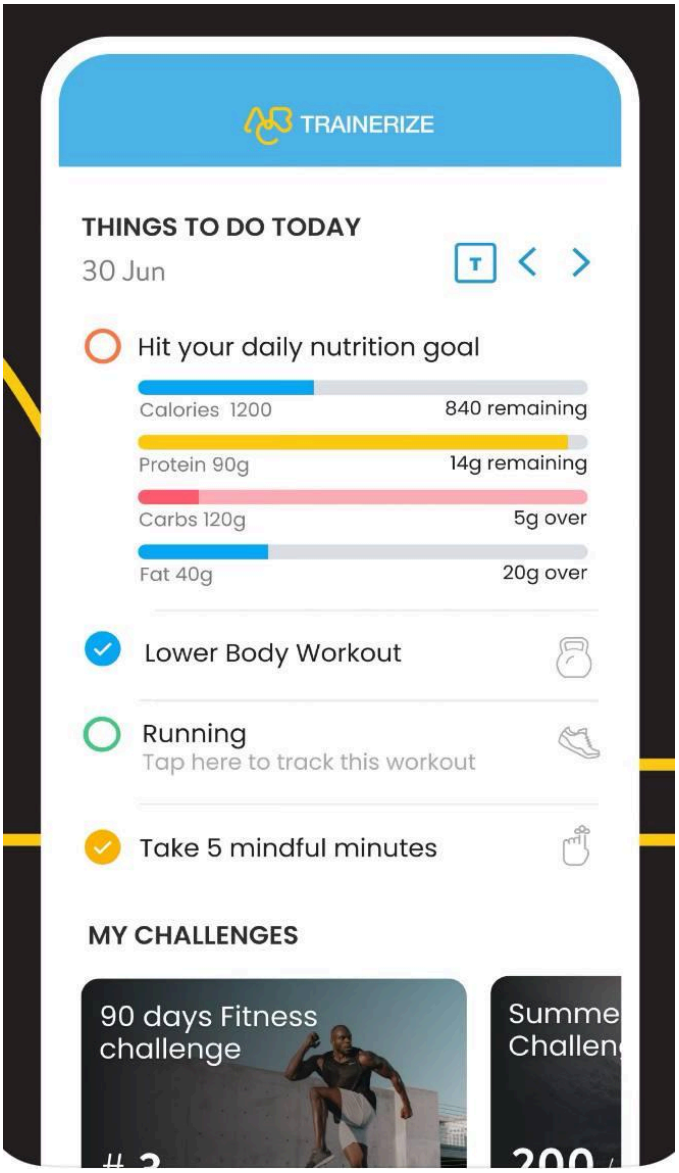

Рисунок 1.3 – Один з екранів додатку Trainerize [3]

Strong – це додаток для ведення тренувань з великим набором вправ і можливістю створення власних тренувальних планів. Він дозволяє вести облік

ваги й обсягів, надає статистику по прогресу та можливість планування тренувань.

Fitbod – цей додаток аналізує ваші попередні тренування та створює індивідуальні тренувальні плани на основі вашого рівня фізичної підготовки та цілей. Він також надає поради щодо оптимального навантаження на м’язи.

Порівнюючи ці додатки, можна зазначити, що кожен з них має свої сильні сторони. MyFitnessPal і Fitbod акцентують увагу на харчуванні та тренуваннях відповідно, що робить їх чудовими доповненнями до тренувального процесу. Strong і Trainerize спрямовані на ведення тренувань та управління клієнтами, з Trainerize більше спрямований на професійних тренерів. Trainingpeaks, у свою чергу, має великий досвід у сфері тренувань витривалості та спортивної підготовки.

Як розробник мобільного додатку CoachNote, я б хотів підкреслити кілька його ключових переваг порівняно з іншими додатками, які були згадані вище.

Комплексність функціоналу. CoachNote поєднує в собі функціонал додатків MyFitnessPal та Strong, надаючи повний спектр інструментів для тренувань та управління клієнтами в одному місці. Ви можете планувати тренування, відстежувати прогрес, спілкуватися з клієнтами та зберігати їх особисті дані в безпечному середовищі.

Персоналізація тренувальних планів. CoachNote дозволяє створювати індивідуальні тренувальні плани для кожного клієнта на основі їхніх цілей, фізичного стану та переваг.

Зручність інтерфейсу. Додаток має інтуїтивно зрозумілий і привабливий інтерфейс, що спрощує використання як для тренерів, так і для клієнтів. При розробці додатку зосередив увагу на зручність користування, щоб кожен міг легко знаходити необхідні функції та інформацію.

Ці переваги дозволяють CoachNote стати ідеальним вибором для тренерів, які шукають комплексне та ефективне рішення для управління тренуваннями та клієнтами. Мета додатку – забезпечити найвищий рівень зручності,

персоналізації та безпеки, щоб допомогти користувачам досягати найкращих результатів у їх роботі. На рисунку 1.4 зображено екран календаря тренувань у додатку CoachNote.

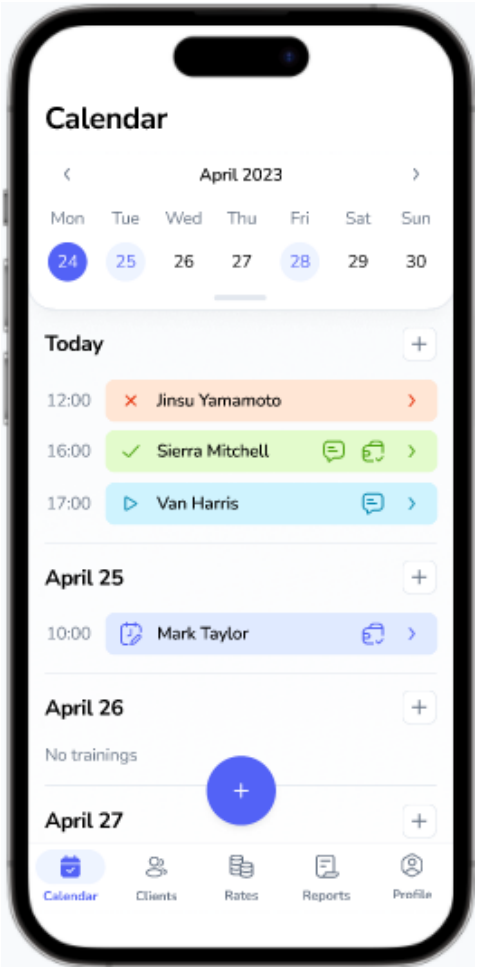

Рисунок 1.4 – Екран календаря тренувань у додатку CoachNote

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Отже було прийнято рішення створити додаток CoachNote. Постановка задачі для написання проекту має бути детальною і чіткою, для більш зрозумілого уявлення про функціонал та основні вимоги до додатку.

Реалізацію даної роботи можна описати у наступних завданнях:

— продумати основний функціонал та сучасний, інтуїтивно зрозумілий інтерфейс;

- підібрати додаткові бібліотеки для роботи з запитами і формами;
- налаштувати та розгорнути додаток локально;
- спроектувати структуру проекту;
- написати код для всіх запланованих модулів;
- протестувати мануально додаток.

Висновки до розділу 1

У першому розділі бакалаврської роботи було проведено всебічний аналіз існуючих мобільних додатків у сфері тренувань, що дозволило визначити ключові тенденції та вимоги користувачів до таких рішень. Було виявлено, що більшість додатків надають базові функції для планування тренувань, відстеження прогресу та комунікації між тренером і клієнтом. Однак, не всі з них задовольняють потреби тренерів у повному обсязі, що створює потребу в новому продукті з більш широкими можливостями та адаптивним підходом до користувача.

Аналіз ринку показав, що основні недоліки існуючих рішень включають складність інтерфейсу, недостатню функціональність для індивідуалізації тренувальних програм, обмежені можливості для аналізу результатів та відсутність інтеграції з іншими популярними сервісами. Це підтверджує актуальність розробки нового мобільного додатку для тренерів «CoachNote», який буде враховувати ці недоліки та пропонувати більш ефективне та зручне рішення.

Таким чином, проведений аналіз мобільних додатків у сфері тренінгу визначив напрямки, в яких мобільний додаток «CoachNote» має значний потенціал для вдосконалення та інновацій. Виконання поставлених завдань забезпечить створення конкурентоспроможного продукту, який задовольнить потреби тренерів та їх клієнтів, підвищуючи ефективність та якість тренувального процесу.

РОЗДІЛ 2
АНАЛІЗ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ТРЕНЕРІВ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Ідея проекту була в тому щоб замінити паперові блокноти які використовують тренери для запису вправ, відвідувань клієнтами тренувань та оплат. Додаток для тренерів повинен надавати функціонал створення і керування планами тренувань, відстежування прогресу клієнтів, ведення журналу тренувань.

Слід описати функціональні та не функціональні вимоги до продукту для глибокого та чіткого розуміння бажаного результату.

Спочатку визначимо функціональні вимоги:

1) реєстрація та автентифікація користувачів:
    a) користувачі повинні мати можливість реєструватися за допомогою email та телефону;
    b) підтримка входу через соціальні мережі (Facebook, Google);
2) створення та редагування профілю:
    a) тренери повинні мати можливість додавати інформацію про себе, включаючи біографію, досвід роботи, спеціалізацію, сертифікації та контактну інформацію;
    b) можливість завантаження фотографії профілю;
3) створення та редагування планів тренувань:
    a) тренери можуть створювати індивідуальні та групові плани тренувань;
    b) можливість додавання вправ, встановлення кількості повторень, підходів, ваги, часу та інших параметрів;
4) відстеження:

a) відображення статистики прогресу клієнтів (наприклад, вага, обсяги тіла, результати тестів);
b) графічне відображення прогресу у вигляді графіків та діаграм;

5) ведення календарів:
   a) клієнти та тренери можуть вести журнал тренувань, записуючи результати кожного тренування;
   b) можливість додавання нотаток та коментарів до кожного запису;
6) встановлення нагадувань для клієнтів про заплановані тренування;
7) сповіщення про нові повідомлення в чаті, оновлення плану тренувань, досягнення нових цілей тощо.

Не функціональні вимоги:

1) швидкість завантаження:
   a) додаток повинен завантажуватися не більше ніж за 3 секунди;
   b) відповідь на запити;
   c) відповідь на дії користувача (наприклад, відкриття профілю, перегляд плану тренувань) повинна відбуватися не більше ніж за 2 секунди;
2) захист даних:
   a) всі особисті дані користувачів повинні зберігатися у зашифрованому вигляді;
   b) використання HTTPS для захисту даних під час передачі;
3) використання двофакторної автентифікації для підвищення безпеки облікових записів;
4) сумісність:
   a) додаток повинен підтримувати операційні системи iOS та Android;
   b) підтримка різних розмірів екранів та роздільної здатності.ю

Виходячи з цих вимог ми можемо побудувати модель поведінки користувача в ролі тренера з усіма потрібними прикладами з дизайнами екранів (рис. 2.1). Модель створено за допомогою програмного забезпечення Figma.

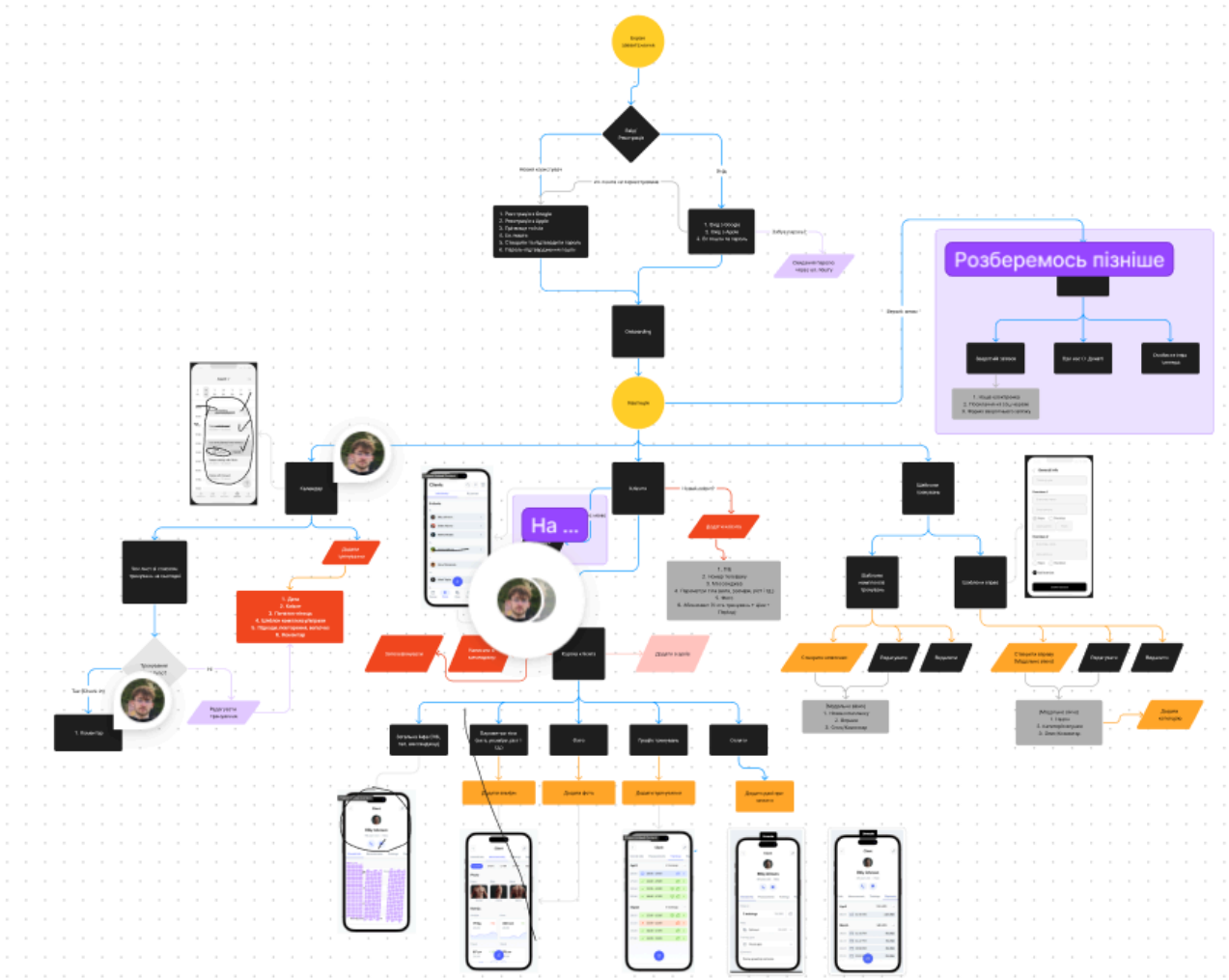

Рисунок 2.1 – Модель поведінки користувача

На основі цієї моделі була побудована UML діаграма функціональної роботи мобільного додатку у ролі тренера (рис. 2.2).

Рисунок 2.2 – UML діаграма роботи мобільного додатку у ролі тренера

Також слід визначити функціональний порядок роботи мобільного додатку у ролі клієнта тренера для можливого майбутнього додавання такого функціоналу у додаток (рис. 2.3).

Рисунок 2.3 – UML діаграма роботи мобільного додатку у ролі клієнта

За допомогою UML діаграм там моделей поведінки користувача ми можемо краще зрозуміти порядок виконання дій та взаємодію між різними компонентами системи.

Для повноцінного забезпечення зручності використання додатку, важливо також розробити UI/UX дизайн. Це допомагає візуалізувати інтерфейси, забезпечити логічну навігацію та оптимізувати користувацький досвід. Ось визначення специфікації вимог до UI/UX дизайну для проекту:

1) принципи дизайну:
    a) простота і зручність: інтерфейс має бути інтуїтивно зрозумілим, з мінімумом кроків для виконання основних завдань;
    b) консистентність: використання єдиних стилів, кольорів та шрифтів на всіх екранах додатку;
    c) доступність: додаток повинен бути зручним для користувачів з обмеженими можливостями, відповідати стандартам доступності;
2) прототип реєстрації та входу:
    a) екран реєстрації з полями для введення email, пароля та підтвердження пароля;
    b) екран входу з полями для введення email та пароля;
3) прототип створення плану тренувань:
    a) форми для введення деталей тренувань (вправи, повторення, підходи, вага);
    b) кнопки для збереження та скасування плану;
4) стильовий гід (Style Guide, UI Kit):
    a) кольорова палітра: основні та додаткові кольори, використання для різних елементів інтерфейсу;
    b) шрифти: основні шрифти, розміри, стилі (наприклад, заголовки, основний текст, підписи);
    c) іконки та графіка: використання іконок, їх стиль, розмір та розташування;

d) кнопки та поля введення: стилі кнопок (заповнені, контурні), поля введення (з рамками, без рамок), стани (нормальний, активний, з помилкою).

На основі цих вимог було побудовано UI/UX дизайн систему у програмі Figma. Декілька екранів дизайну проекту можна переглянути на рисунку 2.4.

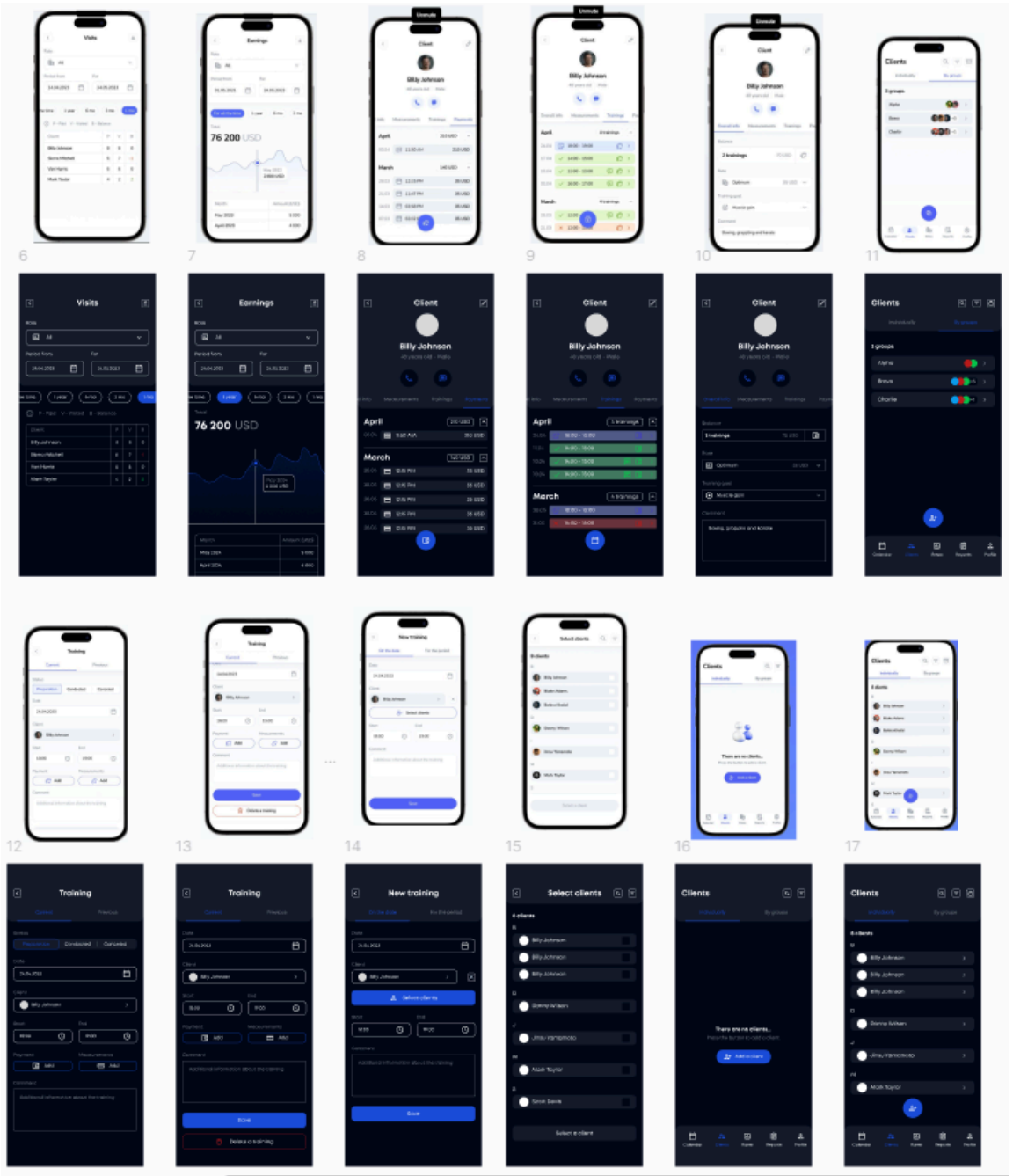

Рисунок 2.4 – Дизайн екранів у Figma

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Вибір технологій для мобільної розробки додатків залежить від багатьох факторів, таких як вимоги до продуктивності, підтримка платформ, час розробки, ресурси команди та інше. Розглянемо кілька основних підходів і технологій для мобільної розробки:

1) нативна розробка (Native Development):
    a) інструменти: Swift/Objective-C для iOS, Kotlin/Java для Android;
    b) переваги:
        — максимальна продуктивність та доступ до всіх можливостей платформи;
        — краща інтеграція з платформою та її сервісами;
    c) недоліки:
        — потрібні окремі команди для кожної платформи (iOS та Android);
        — вищі витрати на розробку та обслуговування;
2) кроссплатформенна розробка (Cross-Platform Development):
    a) інструменти: React Native, Flutter, Xamarin;
    b) переваги:
        — спільний код для обох платформ (iOS та Android);
        — менші витрати на розробку та обслуговування;
    c) недоліки:
        — можуть виникати обмеження у використанні специфічних платформних функцій;
        — можлива деяка втрата продуктивності порівняно з нативними додатками;
3) гібридна розробка (Hybrid Development):
    a) інструменти: Ionic, Cordova;

b) переваги:

- використання веб-технологій (HTML, CSS, JavaScript);
- швидкий час розробки;

c) недоліки:

- обмежена продуктивність;
- обмежений доступ до нативних функцій;

Для розробки такого мобільного додатку вибрано технологію React Native як основу. React Native дозволяє розробникам використовувати ту ж архітектуру, що і для веб-додатків React, але при цьому надає можливість створювати справжні нативні додатки для iOS та Android. Він використовує ті ж фундаментальні блоки інтерфейсу користувача, що і звичайні додатки для iOS і Android, але надає розробникам змогу створювати їх за допомогою React та JavaScript. Це значно полегшує процес розробки, оскільки розробники можуть писати код лише один раз і використовувати його для створення додатків під різні платформи. React Native забезпечує високу продуктивність завдяки використанню нативних компонентів. Технологія має активну спільноту розробників, велику кількість бібліотек та інструментів. Це полегшує вирішення проблем, знаходження відповідей на питання та прискорює процес розробки.

Слід навести декілька прикладів всесвітньовідомих мобільних додатків, які реалізовані за допомогою React Native, для кращого розуміння потужності цієї технології.

Instagram – одна з найпопулярніших платформ у світі. За один місяць програму відвідує 2 мільярди активних користувачів, понад 500 мільйонів використовують його щодня. У своєму офіційному блозі команда Instagram повідомила, що завдяки React Native їм вдалося за короткий час реалізувати функціонал одночасно для iOS та Android-версій (рис. 2.5).

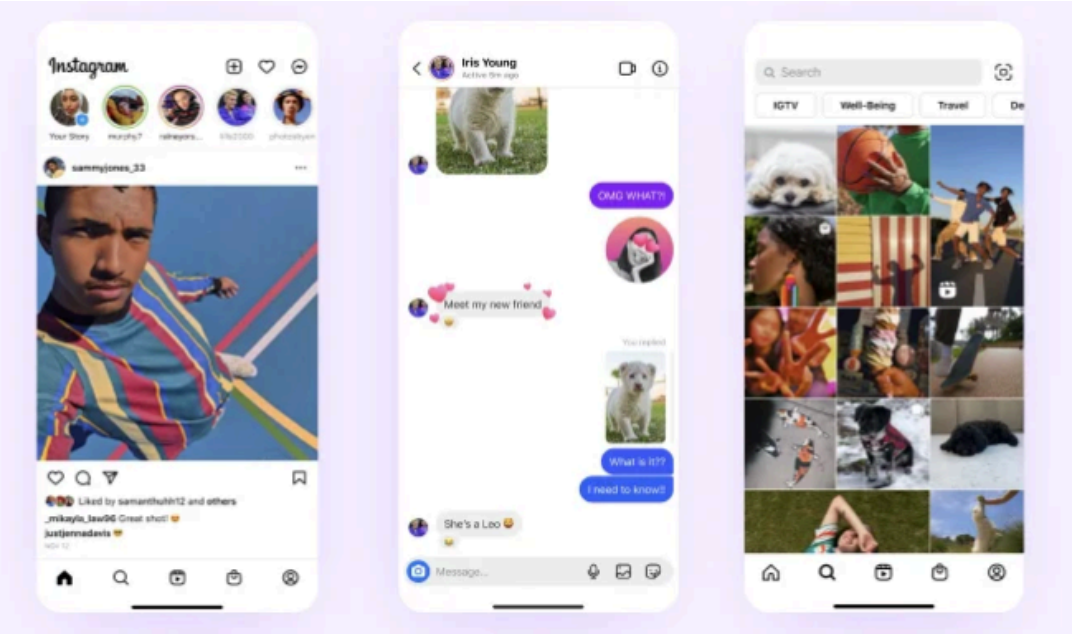

Рисунок 2.5 – Приклад екранів додатку Instagram [4]

На початку 2017 року команда Skype повідомила про створення мобільного додатку на React Native. Android-версія програми являла собою повністю оновлену версію Skype. Змін зазнала істотна частина продукту: від дизайну до функціоналу (рис. 2.6).

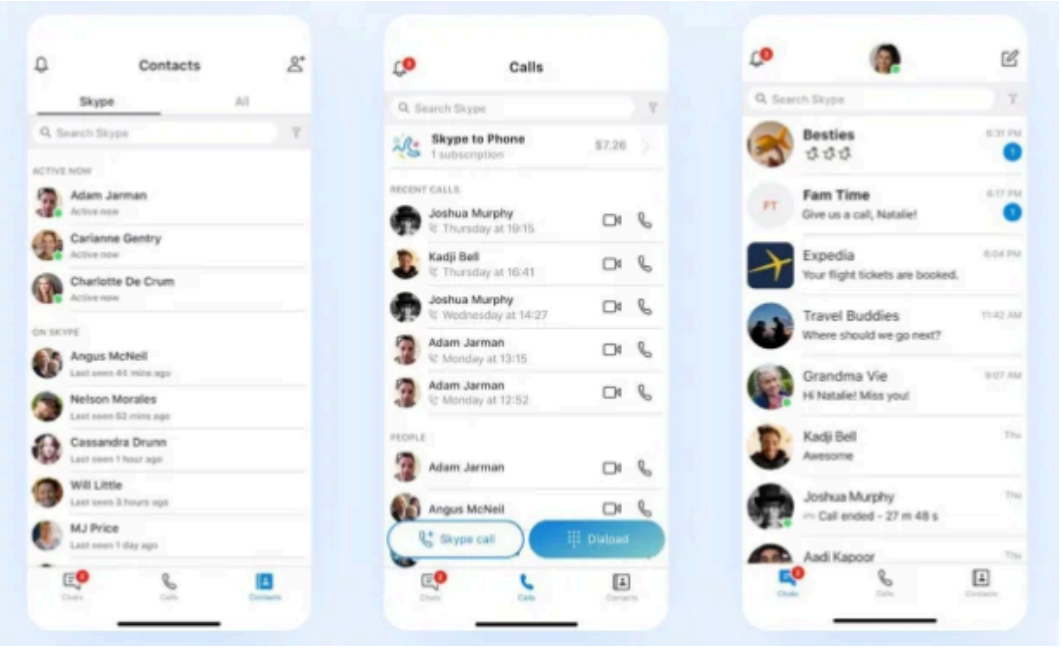

Рисунок 2.6 – Приклад екранів додатку Skype [4]

Tesla, всесвітньо відомий виробник електрокарів, також використовували React Native для розробки мобільного додатку. Зовнішній вигляд та поведінка iOS та Android-версій нічим не відрізнялися для кінцевого користувача. Весь функціон програми, включаючи зарядку, фари, клаксон, панорамний дах працює однаково на обох платформах (рис. 2.7).

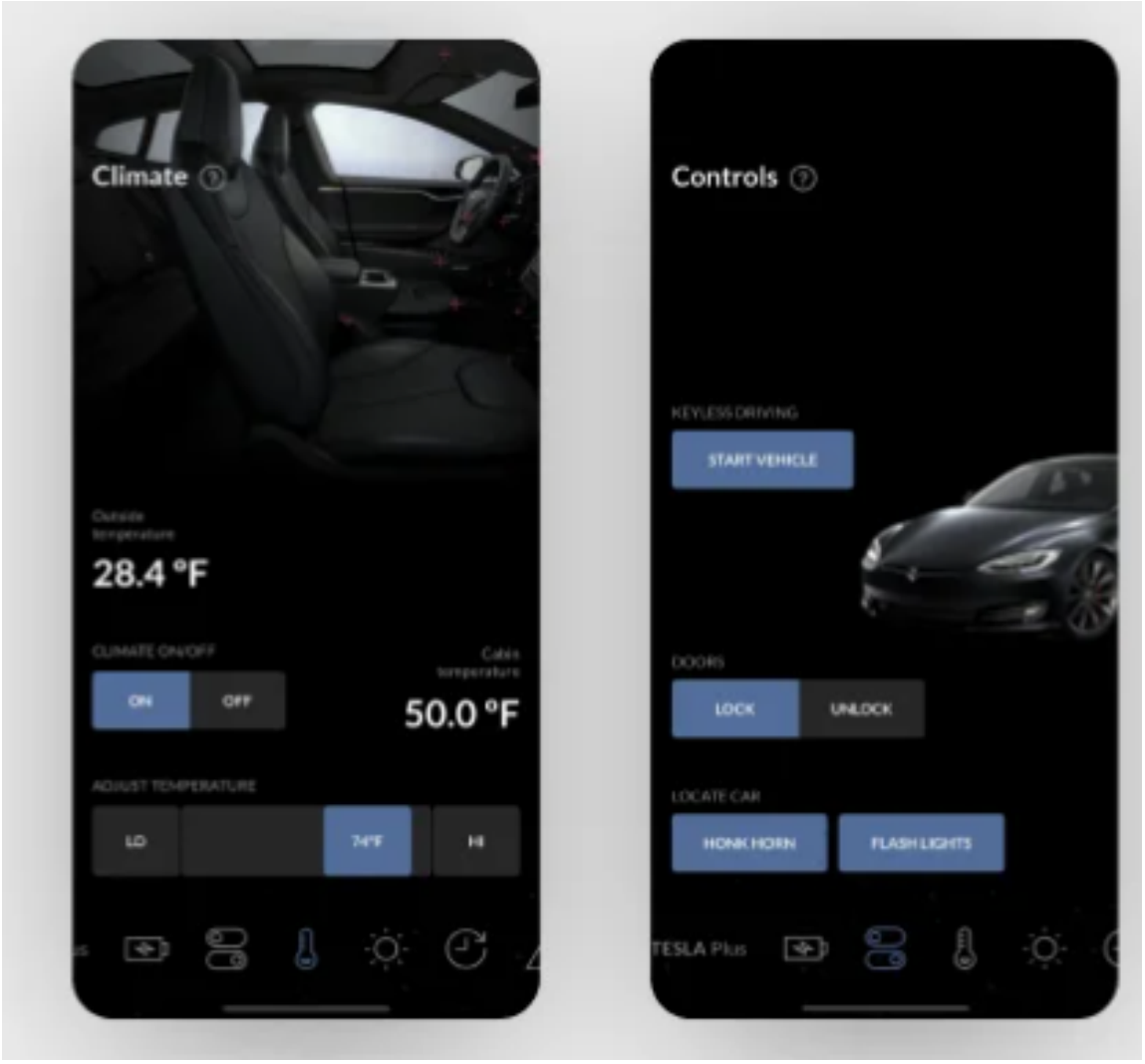

Рисунок 2.7 – Приклад екранів додатку Tesla [4]

Discord – це програма для обміну текстовими та голосовими повідомленнями, створена спеціально для геймерів. Для створення програми на iOS вони використовували React Native та були задоволені результатом. Зараз у

Діскорді мільйони активних користувачів, додаток на 99,9 % crash-free, а його рейтинг у App Store – 4,8 зірки (рис. 2.8).

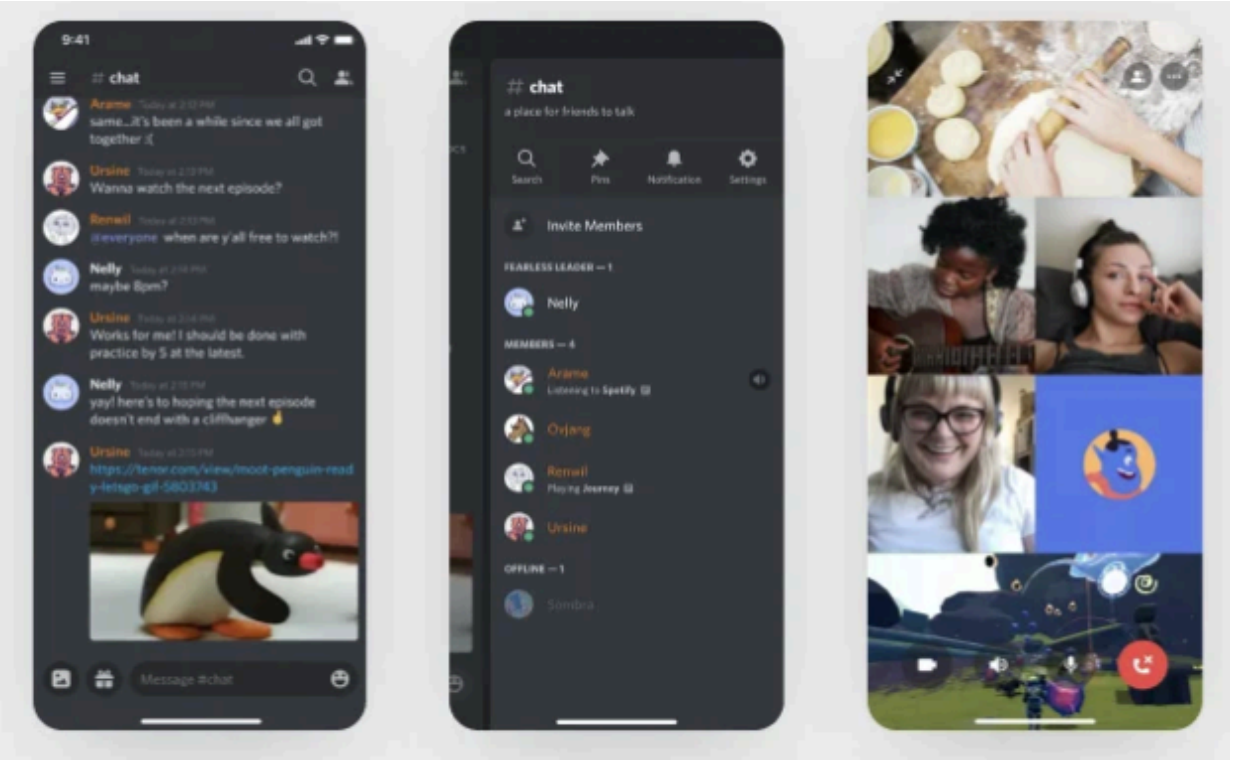

Рисунок 2.8 – Приклади екранів додатку Discord [4]

У процесі розробки додатку я також використовував наступні технології, фреймворки та бібліотеки:

— TypeScript – це типізована версія JavaScript, яка додає статичну типізацію. TypeScript допомагає виявляти та усувати помилки на етапі написання коду, покращуючи читабельність та підтримку коду;

— Expo Router – це потужна технологія для керування навігацією та роутингом у мобільних додатках, розроблених за допомогою React Native. Ця бібліотека надає низку переваг, які спрощують процес розробки та підвищують продуктивність. Однією з ключових переваг Expo Router є його файлове представлення маршрутів (file-based routing). Замість того, щоб визначати всі маршрути в одному центральному місці, Expo Router дозволяє розподілити їх

по окремих файлах, що відповідають різним екранам або розділам додатку. Це значно покращує структурованість та читабельність коду, а також полегшує його підтримку та масштабування. File-based роутинг у Expo Router працює наступним чином: кожен файл у спеціальній папці «app» вважається окремим маршрутом. Назва файлу визначає шлях для цього маршруту, а його вміст містить компоненти React, які будуть відображатися для цього екрану. Ще однією перевагою file-based роутингу є можливість використання динамічних маршрутів. Наприклад, якщо ви маєте екран, який відображає деталі певного елемента, ви можете використати динамічні параметри у назві файлу, наприклад, «app/screens/trainings/[id].tsx». Це дозволить передавати ідентифікатор елемента як частину маршруту і отримувати його у компоненті для подальшої обробки. Загалом, file-based роутинг Expo Router забезпечує більш структурований та інтуїтивно зрозумілий підхід до керування навігацією у мобільних додатках React Native. Він покращує читабельність коду, полегшує його підтримку та масштабування, а також надає зручні функції для створення вкладених та динамічних маршрутів. Ця функціональність допомагає розробникам зосередитися на логіці додатку, не витрачаючи багато часу на налаштування складної навігації;

— NativeBase – це UI бібліотека, яка надає готові стилізовані компоненти React Native для швидкого створення інтерфейсів користувача. NativeBase дозволяє прискорити процес розробки та забезпечити послідовний дизайн у додатку;

— Zustand – бібліотека для керування станом (state management) у React додатках. Вона забезпечує централізоване управління даними та спрощує процес передачі стану між компонентами;

— React Query – це бібліотека для управління сервісами даних у React додатках, що полегшує роботу з асинхронними операціями, такими як запити до API. Вона забезпечує ефективне кешування, оновлення даних, синхронізацію з сервером і зручний спосіб обробки стану завантаження та помилок.

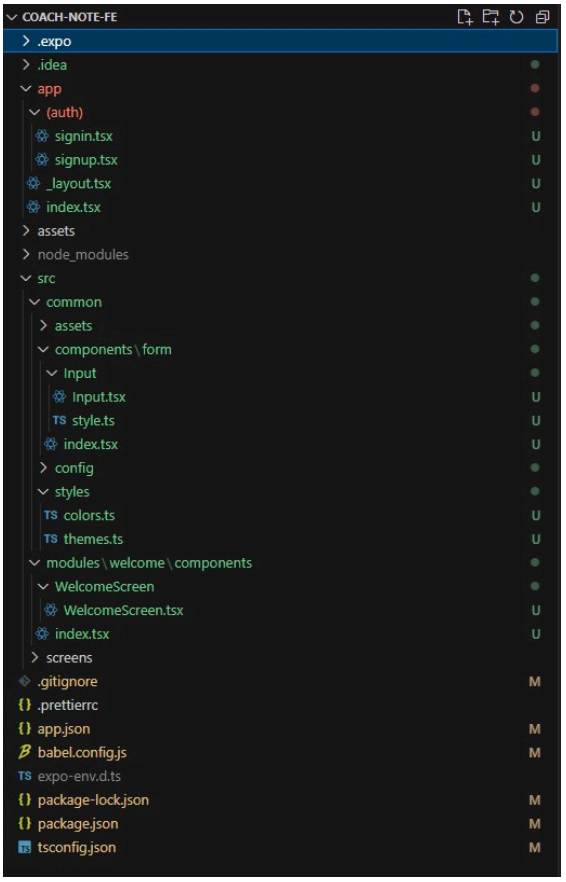

Рисунок 2.9 – Структура проекту

Структура проекту була побудована з використанням модульного підходу, де кожен компонент додатку був розділений на окремі файли та папки для кращої організації коду та полегшення його підтримки. Додаток складався з декількох основних розділів, таких як екран входу, головний екран із списком тренерів, екран деталей тренера та екран редагування профілю тренера.

Висновки до розділу 2

У цьому розділі було проведено аналіз різних технологій, які могли б бути використані для розробки клієнтської частини мобільного додатку для тренерів. Після ретельного аналізу було обрано наступні технології: React Native, Typescript, Expo, Native Base, Zustand, React Query. Вибір цих технологій

ґрунтувався на їх простоті використання, продуктивності, масштабованості та гнучкості.

Вибрані технології дозволили створити мобільний додаток, який є високопродуктивним, легко масштабованим та простим у використанні. Додаток відповідає всім вимогам і може бути легко розміщеним у магазин додатків таких як App Store для IOS та Play Market для Android. В даному випадку було обрано технології, які, на мою думку, найкраще відповідають вимогам проекту.

Однак інші технології також могли б бути використані для розробки такого продукту. У майбутньому може бути переглянуто вибір технологій, якщо вимоги до проекту зміняться.

Загалом, робота над цим проектом використовуючи ці технології була дуже корисною та надала мені цінний досвід у розробці мобільних додатків. Я отримав практичні навички роботи з різними бібліотеками, керуванням станом, роутингом та створенням зручних користувацьких інтерфейсів для мобільних пристроїв.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ТРЕНЕРІВ «COACHNOTE»

3.1 Практична реалізація об’єкта проєктування

Середовище розробки програмного забезпечення (Integrated Development Environment, IDE) є важливим інструментом для програмістів. Його значення полягає у підвищенні продуктивності, зручності, можливостях автозавершення та підказок коду, інтеграції з системами контролю версій, налагоджуванні та тестуванні. IDE надає широкий набір інструментів для написання, налагодження та тестування коду, що значно прискорює процес розробки. Всі необхідні функції (редактор коду, компілятор/інтерпретатор, налагоджувач, інструменти для контролю версій та інші) зібрані в одному місці. Це дозволяє розробникам зосередитись на кодуванні, а не на налаштуванні різних інструментів. Функції автозавершення, інтелектуального аналізу та підказок коду допомагають уникати помилок та швидше писати код. Зручна робота з системами контролю версій (наприклад, Git) дозволяє легко відстежувати зміни в коді, співпрацювати з іншими розробниками та керувати різними версіями проекту. Вбудовані інструменти для налагоджування та тестування дозволяють виявляти та виправляти помилки на ранніх етапах розробки.

Visual Studio Code (VS Code) є одним з найпопулярніших і найбільш використовуваних текстових редакторів серед розробників. Це безкоштовний і відкритий редактор, що робить його доступним для широкого кола розробників. На відміну від багатьох IDE, VS Code є легким та швидким, що забезпечує швидке завантаження і роботу навіть на менш потужних комп’ютерах. Він підтримує широкий спектр розширень та плагінів, що дозволяє адаптувати середовище під будь-які потреби розробника. Можна додати підтримку будь-якої мови програмування, інструменти для налагодження, інтеграцію з системами контролю версій і багато іншого. VS Code має вбудовану підтримку Git, що дозволяє легко виконувати операції з контролю версій без необхідності

залишати редактор. Інтуїтивно зрозумілий інтерфейс та можливість налаштування під власні потреби роблять роботу більш комфортною. IDE включає вбудований термінал, інструменти для налагоджування, підтримку для різних мов програмування прямо «з коробки». Завдяки цим перевагам VS Code став вибором багатьох розробників по всьому світу, незалежно від типу проектів, над якими вони працюють.

Перед початком роботи над проектом потрібно розгорнути проект за допомогою команди: npx create-expo-app@latest. Усі команди для інсталяції пакетів запуску та тестування проекту запускаються з командного рядка Powershell, який також інтегрованив в IDE.

Далі нам потрібно інсталювати всі пакети та бібліотеки. Усі ці залежності можна побачити у файлі конфігурацій та котролю версій package.json (додаток А).

Після інсталяції всіх необхідних пакетів та бібліотек можна починати розробку основних компонентів та функцій. Почати вирішено було з функцій надсилання запиту та авторизації.

Основна та найбільш використовувана функція: apiRequest, код зображено на лістингу 3.1. В тілі цієї функції використовується ще одна функція: getAuthToken (ліст. 3.2), призначення якої отримання отримання токену авторизації для виконання запиту на приватні роути які доступні лише авторизованим користувачам.

Лістинг 3.1 – Код функції apiRequest

```
export const API_URL = 'http://185.235.219.33:5000/'
const api = axios.create({ baseURL: API_URL })
export type DefaultResponse = { success: boolean }
export const apiRequest = async <
  ResponseDataType = DefaultResponse,
  PayloadData = any,
```

```
>(
  config: AxiosRequestConfig<PayloadData>,
) => {
    const isAuthRoute = Object.values(AUTH_ROUTES).some(r => r ===
config.url)
  let config_ = config
  if (!isAuthRoute) {
    const authToken = await getAuthToken()
    if (authToken) {
      config_ = {
        ...config,
        headers: { ...config.headers, Authorization: authToken },
      }
    }
  }
  return api.request<
    ResponseDataType,
    AxiosResponse<ResponseDataType, PayloadData>,
    PayloadData
  >(config_)
}
```

Кінець лістингу 3.1

Лістинг 3.2 – Код функції getAuthToken

```
export const getAuthToken = () => {
  if (Platform.OS === 'web') {
    return localStorage.getItem(AUTH_TOKEN_STORAGE_KEY)
  } else return SecureStore.getItemAsync(AUTH_TOKEN_STORAGE_KEY)
}
```

Кінець лістингу 3.2

Після цього потрібно було написати основні UI компоненти які будуть використовуватись у різних місцях додатку.

Одна з найбільш використовуваних компонент AppButton (ліст. 3.3). Компонента побудована використовуючи компоненту Button з бібліотеки native-base, із додатковими переданими параметрами та стилями відповідно до дизайну (рис. 3.2).

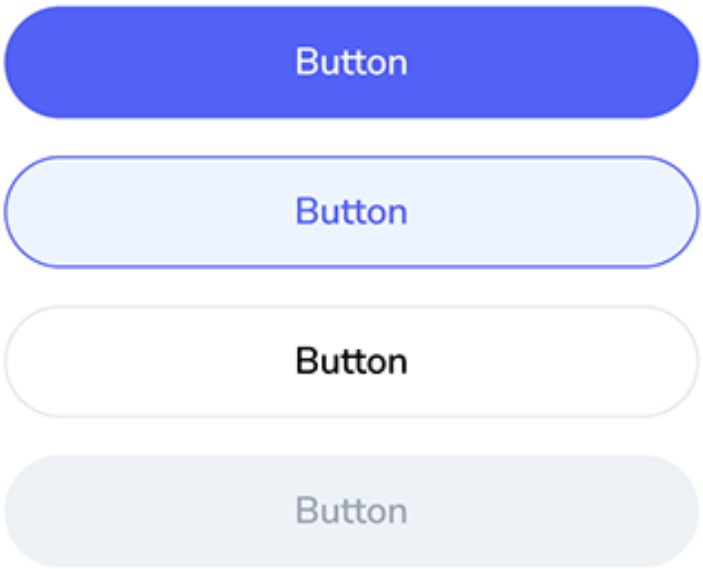

Рисунок 3.2 – Дизайн кнопки додатку в різних станах

Лістинг 3.3 – Код компоненти AppButton

```
export const AppButton: FC<Props> = ({
  type = 'filled',
  buttonColor = 'darkBlue.500',
  ...props
}) => {

  const stylesByType: Record<ButtonType, IButtonProps> = {

    [BUTTON_TYPE.outline]: {
      bg: 'transparent',
      _disabled: { bg: 'muted.300' },
```

```
      _pressed: { bg: 'transparent', shadow: '1' },
      width: 'max-content',
      _icon: { color: buttonColor },
      _text: { color: 'black' },

      borderColor: buttonColor,
      borderStyle: 'solid',
      borderWidth: '1',
    },

    [BUTTON_TYPE.filled]: {
      bg: buttonColor,
      _disabled: { bg: 'muted.300' },
        _pressed: { bg: getColorNativeBase({ color: buttonColor, shade:
600 }) },
      width: 'max-content',
      _icon: { color: 'white' },
      _text: { color: 'white' },
    },
  }

  return (
       <Button  size={'lg'}  borderRadius={'3xl'}  {...stylesByType[type]}
{...props}>

      {props.children}
    </Button>
  )
}
```

Кінець лістингу 3.3

Не менш важлива компонента проекту – InputField.

Відповідно до дизайну має бути назва поля безпосередньо саме поле та підказка (рис. 3.3).

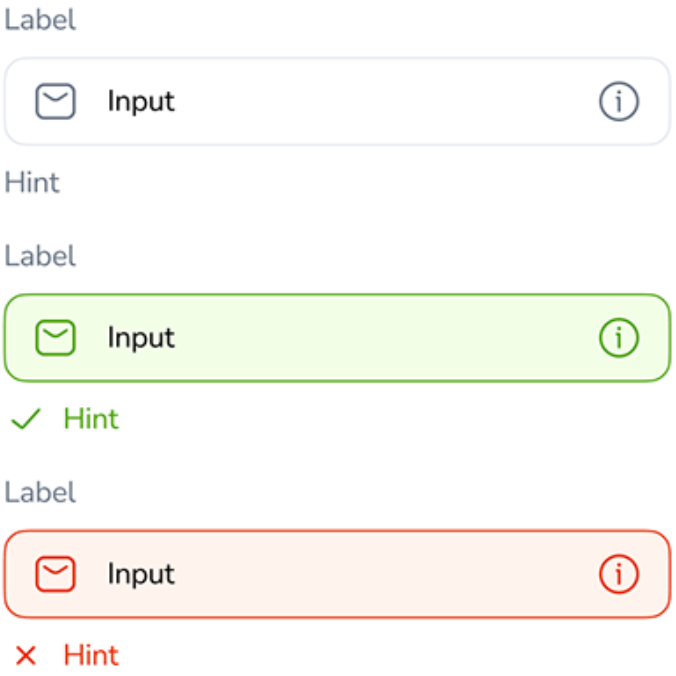

Рисунок 3.3 – Дизайн компоненти InputField

Логічного навантаження на компоненту немає адже вся логіка валідації та визначення доступності введення, проводиться на рівень вище у формі або інших компонентах (ліст. 3.4).

Лістинг 3.4 – Код функції InputField

```
export const InputField: FC<InputFieldProps> = props => {
   const { label, isRequired, onChangeText, message, wrapperProps } =
props

  return (
    <Box alignItems="center" {...wrapperProps}>
      <FormControl isInvalid={!!message?.error} w="full">
        {label && <FormControl.Label>{label}</FormControl.Label>}

        <Input
          {...props}
```

```
            borderRadius={10}
            size={'xl'}
            backgroundColor={'white'}
            isRequired={isRequired}
            onChangeText={t => {
              onChangeText && onChangeText(t)
            }}
            style={{ borderRightWidth: 0 }}
          />

                  <FormControl.ErrorMessage  leftIcon={<WarningOutlineIcon
size="xs" />}>
            {message?.error}
          </FormControl.ErrorMessage>
        </FormControl>
      </Box>
    )
}
```

Кінець лістингу 3.4

Компонента ScreenHeader використовується майже на кожному екрані додатку. Дизайн може відрізнятись наявністю кнопок зправа для виконання різних дій та кнопки Back (рис. 3.4). Код компоненти зображено на лістингу 3.5.

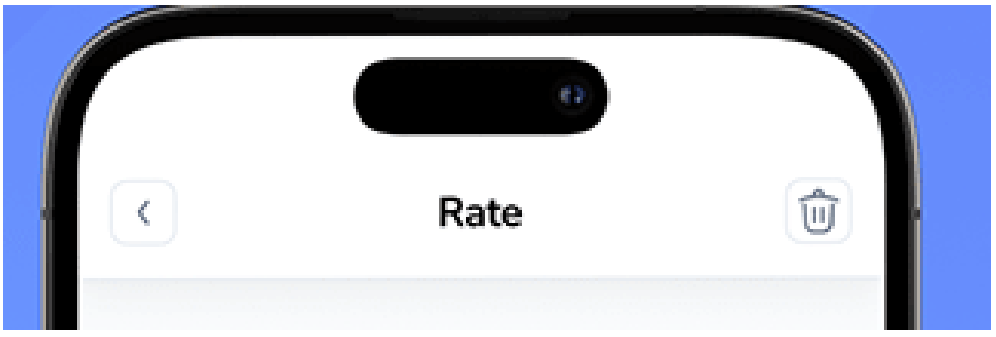

Рисунок 3.4 – Дизайн компоненти ScreenHeader

Лістинг 3.5 – Код компоненти ScreenHeader

```
const ScreenHeader: FC<ScreenHeaderProps> = ({
  headerTitle,
  backBtn,
  headerRightContent,
}) => {
  const router = useRouter()
  const insets = useSafeAreaInsets()
  return (
    <Box
      flexDirection={'row'}
      alignItems={'center'}
      justifyContent={'space-between'}
      borderBottomRadius={'16px'}
      padding={'20px'}
      paddingX={'4'}
      background={'white'}
      paddingTop={`${insets.top}px`}
      shadow={'1'}
      marginBottom={'30px'}
    >
      {backBtn && backBtn.show && (
        <BackButton
          onPress={() => {
            router.back()
            backBtn.onBackPress && backBtn.onBackPress()
          }}
        />
      )}
      <Heading fontSize={'2xl'}>{headerTitle}</Heading>
      {headerRightContent && headerRightContent()}
    </Box>)
}
```

Кінець лістингу 3.5

Окрім повторно використовуваних компонент є також функції які дуже часто можуть знадобитися для форматування та обробки даних.

Одна з таких функцій – groupByKey (ліст. 3.6). Призначення функції – групувати дані по введеному ключу. Перший аргумент – data це масив об’єктів, другий це ключ який є в об’єктах.

Логіка виконання функції:

— ітеруємось по масиву за допомогою методу reduce;

— вибираємо елемент об’єкту по його ключу як ключ для нового об’єкта;

— якщо в новому об’єкті в якому групуємо дані немає ще цього ключа то створюємо його та присвоюємо по цьому ключу пустий масив;

— інакше додаємо в існуючий масив елемент ітерації;

— в кінці повертаємо з функції масив з елементів які являються статичними масивами з двома елементами, перший – ключ, другий – масив даних групи.

Лістинг 3.7 – Код функції groupByKey

```
export const groupByKey = <T>(data: T[], key: keyof T): [string, T[]][]
=> {
  const grouped = data.reduce(
    (acc, item) => {
      const groupKey = item[key] as unknown as string
      if (!acc[groupKey]) {
        acc[groupKey] = []
      }
      acc[groupKey].push(item)
      return acc}
    {} as {   [key: string]: T[] })
  return Object.entries(grouped)}
```

Кінець лістингу 3.7

З основними функціями та компонентами ми вже ознайомились, також варто показати реалізацію та кінцевий результат окремих цілих екранів, для прикладу візьмемо екран списку клієнтів. Код компоненти екрану викладено у лістингу 3.8. Кінцевий вигляд екрану зображено на рисунку 3.5.

Лістинг 3.8 – Код компоненти Clients

```
const Clients: FC<ClientsProps> = ({ data }) => {
  const router = useRouter()
  return (
    <AppScreen
      screenHeaderProps={{
        headerTitle: 'Clients',
        backBtn: { show: true },
        headerRightContent: () => {
          return (<Box flexDirection={'row'}>
              <AppIconButton icon={<MaterialIcons name={'search'} />} />
              <AppIconButton
                icon={<Ionicons name={'filter'} />}
                marginLeft={'1'} />
              <AppIconButton
                icon={<Ionicons name={'add'} />}
                marginLeft={'1'}
                onPress={() => {
                  router.push('clients/new')
                } />
            </Box>)}
      }}>
      <ClientsList
         data={[...data, ...data].map((el, i) => ({ ...el, id: el.id + i
}))} />
    </AppScreen>)}
export default Clients
```

Кінець лістингу 3.8

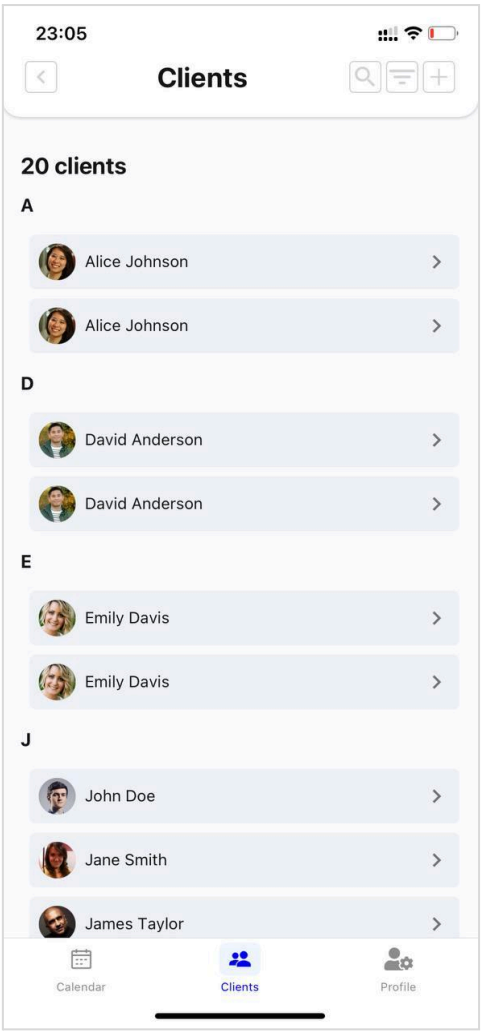

Рисунок 3.5 – Кінцевий вигляд екрану Clients

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування програмного забезпечення є критично важливим етапом у розробці, який забезпечує високу якість кінцевого продукту та його відповідність заданим вимогам. Процес тестування дозволяє виявити та виправити помилки ще до того, як продукт або новий функціонал буде випущено на ринок. Крім того, це забезпечує задоволеність користувачів, підвищує довіру до продукту та сприяє збереженню репутації компанії-розробника.

Описання тест кейсів, у свою чергу, є невід’ємною частиною процесу тестування. Тест кейси детально описують умови, при яких тестування повинно

бути виконано, кроки для виконання, очікувані результати та фактичні результати. Вони є своєрідним сценарієм для тестування, який допомагає забезпечити повне покриття всіх можливих варіантів використання продукту. Це дозволяє тестувальникам послідовно і систематично перевіряти функціональність та стабільність програмного забезпечення.

Добре продумані та детально описані тест кейси допомагають виявити навіть ті дефекти, які можуть залишитися непоміченими при поверхневому тестуванні. Вони також забезпечують повторюваність тестування, що є важливим у випадках, коли необхідно провести регресійне тестування після внесення змін у код. Крім того, тест кейси слугують документацією, яка може бути використана новими членами команди для швидкого ознайомлення з тестовими сценаріями та вимогами до продукту.

Таким чином, тестування та описання тест кейсів забезпечують високу якість програмного забезпечення, підвищують ефективність процесу розробки, знижують ризики та витрати на виправлення помилок, а також сприяють задоволеності користувачів кінцевим продуктом.

Опишемо декілька тест кейсів, оскільки розроблюваний додаток вміщує в собі багато функціоналу, виберемо один модуль для прикладу – авторзацію (табл. 3.1). Авторизація є критично важливою частиною додатка, оскільки від неї залежить безпека користувацьких даних та доступ до персоналізованих функцій. Тест кейси для модуля авторизації включатимуть перевірку коректного введення логіна та пароля, обробку помилкових даних, відновлення пароля, та багатофакторну аутентифікацію. Ці тест кейси дозволять гарантувати, що процес авторизації є надійним і безпечним, що сприятиме загальній безпеці та функціональності додатку. Описані тест кейси також слугуватимуть основою для майбутніх тестувань при внесенні змін до функціоналу авторизації, що забезпечить стабільність і безперебійність роботи додатка.

Таблиця 3.1 – Тест кейси для модуля авторизації

Передумови	Кроки	Очікуваний результат
Успішна авторизація з правильним номером телефону та кодом підтвердження		
Номер телефону зареєстровано в системі	— ввести номер телефону та продовжити; — ввести код підтвердження, отриманий на телефон; — натиснути кнопку «Увійти».	Користувач успішно авторизується в системі
Невдала авторизація з неправильним кодом підтвердження		
Номер телефону зареєстровано в системі	— ввести номер телефону; — натиснути кнопку «Продовжити»; — ввести неправильний код підтвердження; — натиснути кнопку «Увійти».	Відображається повідомлення про помилку: «Неправильний код підтвердження»
Невдала авторизація з незареєстрованим номером телефону		
Номер телефону не зареєстровано в системі	— ввести номер телефону; — Натиснути кнопку «Продовжити».	Відображається повідомлення про помилку: «Номер телефону не зареєстровано»
Введення неправильного формату номера телефону		
Немає	— ввести номер телефону у неправильному форматі (наприклад, з літерами).	Відображається помилка: «Неправильний формат номера телефону»
Невдала авторизація після декількох неправильних спроб введення коду		
Номер телефону зареєстровано в системі	— ввести номер телефону; — натиснути кнопку «Продовжити»; — ввести неправильний код підтвердження; — повторити введення неправильного коду кілька разів.	Відображається повідомлення про помилку: «Забагато неправильних спроб. Будь ласка, спробуйте пізніше»

3.3 Розробка документації для програмного продукту

Ця документація допоможе вам інсталювати та розгорнути проект на Expo та React Native. Ми розглянемо детальні кроки для встановлення необхідних інструментів, налаштування середовища розробки, завантаження проекту з GitHub та запуску проекту.

Вимоги до системи. Перед тим як розпочати, переконайтеся, що у вас встановлені наступні компоненти:

- Node.js (рекомендована версія: LTS);
- npm або yarn;
- Expo CLI.

Крок 1: встановлення Node.js та npm.

Якщо у вас ще не встановлений Node.js, завантажте та встановіть останню версію з офіційного сайту Node.js. Після встановлення перевірте, чи все працює, відкривши термінал і виконавши наступні команди:

- node -v;
- npm -v.

Крок 2: встановлення Expo CLI.

Expo CLI – це інструмент командного рядка, який спрощує створення та управління проектами на базі Expo. Встановіть Expo CLI за допомогою npm або yarn: npm install -g expo-cli або yarn global add expo-cli.

Крок 3: клонування проекту з GitHub.

- склонуйте проект з GitHub за допомогою наступної команди: git clone https://github.com/SerjeyMelnik/coach-note-fe.git;
- перейдіть у директорію проекту: cd coach-note-fe.

Крок 4: встановлення залежностей.

Встановіть всі необхідні залежності для проекту за допомогою npm або yarn: npm install або yarn install.

Крок 5: запуск проекту.

Запустіть проект за допомогою наступної команди: expo start. Ця команда запустить Expo DevTools у вашому браузері та надасть QR-код для запуску програми на фізичному пристрої через додаток Expo Go (доступний для iOS та Android).

Крок 6: запуск на емуляторі або фізичному пристрої.

Для запуску на фізичному пристрої:

— встановіть додаток Expo Go з App Store або Google Play;

— відскануйте QR-код, що з’явиться у Expo DevTools, за допомогою Expo Go.

Для запуску на емуляторі:

— встановіть Android Studio або Xcode;

— налаштуйте віртуальний пристрій (AVD) або iOS симулятор;

— відкрийте віртуальний пристрій і натисніть "Run on Android device/emulator" або "Run on iOS simulator" в Expo DevTools.

Крок 7: налаштування середовища.

Ви можете змінити налаштування вашого проекту, редагуючи файли app.json або expo-config.js. Ці файли містять конфігурації для вашого додатку, такі як ім’я, версія, іконки та інші параметри.

Крок 8: розгортання проекту.

Коли ваш додаток готовий для розгортання, скористайтеся командою expo build для створення готових пакетів для iOS та Android: expo build:android або expo build:ios. Ці команди створять файли .apk (для Android) та .ipa (для iOS), які можна завантажити на відповідні магазини додатків.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи бакалавра було детально розглянуто практичну реалізацію мобільного додатку CoachNote, проведено його тестування та складено документацію. Розробка додатку здійснювалася з

використанням технологій Expo та React Native, що дозволило створити кросплатформенний продукт, здатний працювати як на iOS, так і на Android пристроях.

На етапі практичної реалізації було проведено комплексне тестування для забезпечення високої якості додатку, яке включало функціональне тестування, перевірку коректності роботи всіх функцій додатку та оцінку зручності використання інтерфейсу користувачами. Було також проведено кросплатформенне тестування, яке включало тестування додатку на різних платформах (iOS та Android) для виявлення можливих проблем сумісності.

Також було складено детальну документацію до проекту, яка включала інструкції щодо його інсталяції та використання. Інструкція з інсталяції та розгортання надавала покрокові вказівки для встановлення та налаштування середовища розробки, завантаження та запуску проекту з GitHub.

ВИСНОВКИ

Результати, досягнуті у даній кваліфікаційній роботі бакалавра, свідчать про успішну реалізацію поставлених завдань. Розроблений мобільний додаток CoachNote повністю відповідає вимогам проекту, демонструє стабільну роботу та забезпечує зручність використання.

Аналіз ринку показав, що основні недоліки існуючих рішень включають складність інтерфейсу, недостатню функціональність для індивідуалізації тренувальних програм, обмежені можливості для аналізу результатів та відсутність інтеграції з іншими популярними сервісами. Це підтверджує актуальність розробки нового мобільного додатку для тренерів «CoachNote», який враховує ці недоліки та пропонує більш ефективне та зручне рішення.

Вибрані технології та підібрані додаткові бібліотеки дозволили створити мобільний додаток, який є високопродуктивним, легко масштабованим та простим у використанні. Підібрані додаткові бібліотеки дозволили легко та зручно працювати із запитами та формами. Було розроблено декілька основних модулів проекту: модуль авторизації, тренування, та модуль клієнти.

Додаток відповідає всім вимогам і може бути легко розміщеним у магазин додатків таких як App Store для IOS та Play Market для Android. В даному випадку було обрано технології, які, на мою думку, найкраще відповідають вимогам проекту.

Проведене мануальне тестування підтвердило високу якість продукту, а складена документація дозволить легко інсталювати та підтримувати розробку проекту.

Таким чином, проведення аналізу ринку, виконання поставлених завдань, тестування продукту дозволило створити якісний та конкурентоспроможний мобільний додаток «CoachNote».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт додатку «trainingpeaks». URL: https://www.trainingpeaks.com/ (дата звернення: 29.04.2024).

2. Офіційний сайт додатку «myfitnesspal». URL: https://www.myfitnesspal.com/ (дата звернення: 29.04.2024).

3. Офіційний сайт додатку «trainerize». URL: https://www.trainerize.com/ (дата звернення: 29.04.2024).

4. Сайт «purrweb». URL: https://www.purrweb.com/blog/best-react-native-apps/ (дата звернення: 29.04.2024).

5. «React Native Documentation» – офіційна документація React Native. URL: https://reactnative.dev/docs/getting-started (дата звернення: 15.04.2024).

6. «Expo Documentation» – офіційна документація Expo. URL: https://docs.expo.dev/ (дата звернення: 16.04.2024).

7. «Expo Router» – офіційна документація Expo Router. URL: https://docs.expo.dev/router/introduction/ (дата звернення: 16.04.2024).

8. «NativeBase Documentation» – документація UI бібліотеки для React Native. URL: https://docs.nativebase.io/ (дата звернення: 17.04.2024).

9. «TypeScript Documentation» – офіційна документація TypeScript. URL: https://www.typescriptlang.org/docs/ (дата звернення: 25.04.2024).

10. «Zustand» – офіційна документація Zustand. URL: https://docs.pmnd.rs/zustand/getting-started/introduction (дата звернення: 25.04.2024).

11. «React Query» – офіційна документація React Query. URL: https://reactdev.ru/libs/react-query/ (дата звернення: 28.04.2024).

12. «VS code» – офіційна документація редактору коду Visual Studio Code. URL: https://code.visualstudio.com/docs (дата звернення: 29.04.2024).

ДОДАТКИ

Додаток А

Лістинг файлу package.json

```
{
  "name": "coach-note-fe",
  "version": "1.0.0",
  "main": "expo-router/entry",
  "scripts": {
    "start": "expo start",
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web",
    "format": "prettier --write \"**/*.{js,jsx,ts,tsx,json,md}\""
  },
  "dependencies": {
    "@emotion/react": "^11.11.4",
    "@emotion/styled": "^11.11.5",
    "@expo/vector-icons": "^14.0.2",
    "@mui/icons-material": "^5.15.16",
    "@mui/material": "^5.15.16",
    "@react-native-community/datetimepicker": "^8.0.1",
    "@react-native-vector-icons/material-icons": "^0.0.1-alpha.9",
    "@react-navigation/stack": "^6.3.20",
    "axios": "^1.6.8",
    "expo": "^51.0.2",
    "expo-constants": "~16.0.1",
    "expo-font": "~12.0.5",
    "expo-linking": "~6.3.1",
    "expo-router": "~3.5.11",
    "expo-secure-store": "^13.0.1",
    "expo-status-bar": "~1.12.1",
    "formik": "^2.4.6",
    "moment": "^2.30.1",
    "native-base": "^3.4.28",
    "normalize-css-color": "^1.0.2",
    "react": "18.2.0",
```

```
    "react-dom": "18.2.0",
    "react-native": "0.74.1",
    "react-native-safe-area-context": "4.10.1",
    "react-native-screens": "3.31.1",
    "react-native-svg": "15.2.0",
    "react-native-web": "~0.19.6",
    "react-query": "^3.39.3",
    "yup": "^1.4.0",
    "zustand": "^4.5.0"
  },
  "devDependencies": {
    "@babel/core": "^7.24.0",
    "@types/node": "^20.11.17",
    "@types/react": "~18.2.79",
    "prettier": "3.1.1",
    "typescript": "^5.1.3"
  },
  "private": true
}
```