Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри
комп’ютерних наук,канд. техн.
наук, доцент

_____Ірина МАШКІНА
«_____» _____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»

Спеціальність 122 Комп’ютерні науки

Освітня програма 122.00.01 Інформатика

Тема: РОЗРОБКА ЗАСТОСУНКУ ДЛЯ МОНІТОРИНГУ ВОДНОГО БАЛАНСУ КОРИСТУВАЧА

Виконав
студент групи ІНб-2-21-4.0д
(шифр академічної групи)
Поліщук Олександр Русланович
(прізвище, ім’я, по батькові)

_____
(підпис)
Науковий керівник
к.п.н., доцент кафедри комп'ютерних наук
(науковий ступінь, наукове звання)
Глушак О. М.
(прізвище, ініціали)

_____
(підпис)

Київ – 2025

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

«Затверджую»

Завідувач кафедри
комп’ютерних наук,канд. техн.
наук, доцент
)

_____Ірина МАШКІНА
(підпис)

«_____» _____2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Розробка застосунку для моніторингу водного балансу користувача»

Виконавець – студент групи ІНб-2-21-4.0д,
спеціальності 122 Комп’ютерні наук, освітньої
програми 122.00.01 Інформатика
Поліщук Олександр Русланович

1. Вихідні дані: *Публікації за напрямом розробки мобільних застосунків, кросплатформної розробки, інтеграції з хмарними сервісами, побудови інтуїтивних інтерфейсів користувача та обробки персональних даних.*

2. Основні завдання: *Проаналізувати потреби користувачів у моніторингу водного балансу та дослідити існуючі рішення для визначення ключових вимог. Обґрунтувати мету, об’єкт, предмет і наукові основи розробки кросплатформного застосунку. Обрати й обґрунтувати технології, спроектувати архітектуру застосунку. Реалізувати модулі автентифікації, обробки даних, інтерфейсу для їх відображення, прогресу та статистики. Провести тестування на різних платформах, перевірити валідацію, навігацію, відображення і збереження даних. Проаналізувати результати тестування, оцінити ефективність та зручність інтерфейсу, сформулювати висновки. Оформити пояснювальну записку за вимогами кафедри комп’ютерних наук та підготувати тези для наукової конференції.*

3. Пояснювальна записка: Обсяг – до 45 стор. формату А4 комп’ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.

4. Графічні матеріали: *не передбачено.*

5. Додатки: *не передбачено*.

6. Строк подання роботи на кафедру: *«_____» ____2025 р.*

Науковий керівник
Науковий ступінь, звання
_____ПІБ
_____Дата

Виконавець:
_____ПІБ
_____дата

РЕФЕРАТ бакалаврської роботи

Кваліфікаційна робота: 45 с., 24 рис., 0 табл., 15 посилань.

Актуальність: Недостатнє споживання води є поширеною проблемою, оскільки більшість людей не випивають рекомендовану норму води, що негативно позначається на їхньому здоров'ї та продуктивності. Люди рідко відстежують, які напої і в якій кількості вони споживають, а також не усвідомлюють, як це впливає на їхній водний баланс і як він змінюється протягом дня. Це підкреслює потребу в простому та зручному рішенні. Кросплатформний мобільний застосунок, розроблений на базі React Native, забезпечує доступність на платформах Android та iOS. Завдяки інтуїтивному інтерфейсу та інтеграції з хмарним сховищем даних Firebase, він дозволяє легко фіксувати та відстежувати споживання води, сприяючи формуванню здорових звичок.

Об'єкт дослідження: Мобільні застосунки для моніторингу водного балансу користувачів.

Предмет дослідження: Інструменти та технології для розробки кросплатформного застосунку HydrationApp.

Мета роботи: Розробити кросплатформний застосунок HydrationApp для моніторингу водного балансу з автентифікацією та хмарним збереженням даних користувача.

Завдання:

– Провести аналіз потреб користувачів та існуючих застосунків для моніторингу водного балансу з метою визначення ключових вимог.

– Дослідити сучасні технології та інструменти для розробки кросплатформного мобільного застосунку.

– Спроектувати архітектуру застосунку, включаючи модулі автентифікації, обробки даних, навігації та візуалізації.

– Реалізувати функціонал застосунку: автентифікацію, профіль користувача, фіксацію спожитих напоїв, відображення прогресу та статистики.

– Провести тестування застосунку на платформах Android та iOS для перевірки валідації, навігації, збереження даних і зручності інтерфейсу.

Основні результати: У результаті дослідження розроблено кросплатформний мобільний застосунок HydrationApp на базі React Native з інтеграцією Firebase для автентифікації та Firestore для хмарного зберігання даних. Реалізовано п’ять основних екранів: AuthScreen для безпечної автентифікації та збору персональних даних, HomeScreen з круговою діаграмою прогресу споживання води, ProfileScreen для редагування особистих даних і пароля, AddDrinkScreen для фіксації напоїв, StatisticScreen для аналізу споживання загалом та за визначений день. Застосунок забезпечує валідацію даних (email, пароль, ім’я, дату народження, вагу), реальний час оновлення даних через Firestore, а також захист даних за допомогою Firestore rules. Тестування на Android та iOS через Expo Go підтвердило стабільність, інтуїтивність інтерфейсу та коректність роботи функцій застосунку.

Практичне значення дослідження: Застосунок сприяє підтримці водного балансу, дозволяючи користувачам фіксувати спожиті напої, переглядати прогрес у реальному часі та аналізувати статистику споживання. Застосунок може бути використаний для персонального моніторингу здоров’я, а його архітектура дозволяє додавання нових функцій у майбутньому.

Ключові слова: Моніторинг водного балансу, застосунок, автентифікація, валідація даних, кругова діаграма, статистика, React Native, Firebase, Firestore, Expo Go.

ЗМІСТ

ВСТУП

У сучасних умовах зростання уваги до здоров’я та добробуту людини підтримка водного балансу набуває особливого значення. Достатнє споживання води є ключовим фактором для забезпечення фізіологічних потреб організму, підвищення продуктивності та профілактики низки захворювань. Проте багато людей не приділяють належної уваги відстеженню кількості спожитої води через брак зручних і доступних інструментів. Існуючі рішення для моніторингу водного балансу часто обмежені за функціоналом, складні у використанні або не забезпечують кросплатформної доступності, що створює потребу в розробці нового, інтуїтивного та ефективного інструменту.

Метою кваліфікаційної роботи є розробка кросплатформного мобільного застосунку HydrationApp для моніторингу водного балансу користувача з підтримкою автентифікації та хмарного збереження даних.

Об’єктом дослідження є мобільні застосунки для моніторингу водного балансу користувачів.

Предметом дослідження виступають технології та інструменти для розробки кросплатформного застосунку HydrationApp.

Завдання роботи включають:

– Проаналізувати застосунки для моніторингу водного балансу з метою визначення ключових вимог.

– Дослідити сучасні технології для розробки кросплатформного мобільного застосунку.

– Спроектувати архітектуру застосунку.

– Реалізувати функціонал застосунку: автентифікацію, профіль користувача, фіксацію спожитих напоїв, відображення прогресу та статистики.

– Протестувати застосунок для перевірки валідації, навігації, збереження даних і зручності інтерфейсу.

Практична цінність роботи полягає у створенні зручного інструменту для щоденного моніторингу водного балансу, який дозволяє користувачам фіксувати спожиті напої, переглядати прогрес у реальному часі та аналізувати статистику.

Методи дослідження включають технологічний аналіз, проєктування, програмну інженерію, тестування та валідацію.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика предметної області

Предметна область дослідження присвячена розробці мобільних застосунків для моніторингу водного балансу користувачів, що є важливим елементом підтримки здоров’я та профілактики порушень, пов’язаних із недостатнім споживанням води. Водний баланс відіграє ключову роль у забезпеченні нормального функціонування організму, зокрема в процесах терморегуляції, метаболізму, транспортування поживних речовин та виведення продуктів обміну. Недостатнє споживання води може спричинити до зниження фізичної та розумової працездатності, порушення концентрації, головний біль, а в довгостроковій перспективі — серйозні проблеми зі здоров’ям, такі як захворювання нирок або серцево-судинної системи. За даними медичних досліджень, більшість людей не дотримуються рекомендованої норми споживання води, яка становить приблизно 2-3 літри на добу для дорослих залежно від індивідуальних параметрів, що підкреслює розробку застосунків для формування здорової звички регулярного споживання води.

1. Більшість людей не ведуть облік кількості та типу спожитих напоїв (вода, чай, кава, соки тощо), що ускладнює контроль водного балансу. Мобільні застосунки дозволяють систематизувати цей процес, надаючи зручні інструменти для фіксації об’єму води, аналізу її типу та регулярних нагадувань про необхідність пиття.

2. Рекомендована добова норма води залежить від індивідуальних характеристик людини, зокрема ваги, статі, віку, рівня фізичної активності та кліматичних умов. Наприклад, стандартна формула розрахунку становить 30–35 мл води на 1 кг маси тіла, однак для осіб із високою фізичною активністю або в умовах спекотного клімату ця норма може зростати. Застосунки мають враховувати ці фактори для формування точних рекомендацій.

3. Мобільні застосунки забезпечують автоматизацію процесу відстеження водного балансу завдяки зручним інтерфейсам, інтерактивним елементам (графіки, діаграми) та інтеграції з пристроями, такими як смарт-годинники. Це сприяє підвищенню залученості користувачів і формуванню здорових звичок.

4. Для забезпечення широкого охоплення аудиторії застосунок має бути доступним на різних операційних системах, зокрема Android та iOS. Використання кросплатформних технологій дозволяє створювати універсальні рішення з єдиною кодовою базою, що знижує витрати на розробку та забезпечує однаковий користувацький досвід.

5. Хмарне зберігання даних забезпечує синхронізацію інформації між різними пристроями, захист даних і можливість їх відновлення, дозволяють реалізувати автентифікацію користувачів, зберігання профілів і записів про споживання води, а також оновлення даних у реальному часі.

6. Для підтримки інтересу до використання застосунку важливими є елементи гейміфікації (досягнення, нагороди), регулярні нагадування та візуалізація прогресу у вигляді статистики. Це сприяє формуванню сталої звички регулярного споживання води, особливо для користувачів із напруженим графіком.

1.2 Аналіз існуючих рішень

Для аналізу предметної області розглянемо популярні на даний момент рішення для моніторингу водного балансу, зокрема мобільні застосунки WaterMinder [1] і Hydro Coach [2], проведемо їхній огляд та загальну характеристику з визначеними перевагами та недоліками.

WaterMinder – це мобільний застосунок, розроблений для відстеження споживання води, який доступний на платформах iOS і Android. Він дозволяє користувачам фіксувати об’єм і тип напоїв, відображає прогрес у вигляді людського силуету або кругової діаграми та інтегрується з Apple Health для синхронізації даних про здоров’я. Застосунок пропонує персоналізовані рекомендації щодо норми води на основі ваги, статі та рівня активності, погоди. WaterMinder вирізняється простим і мінімалістичним дизайном, що робить його популярним серед користувачів, які цінують зручність (Рис.1.1).

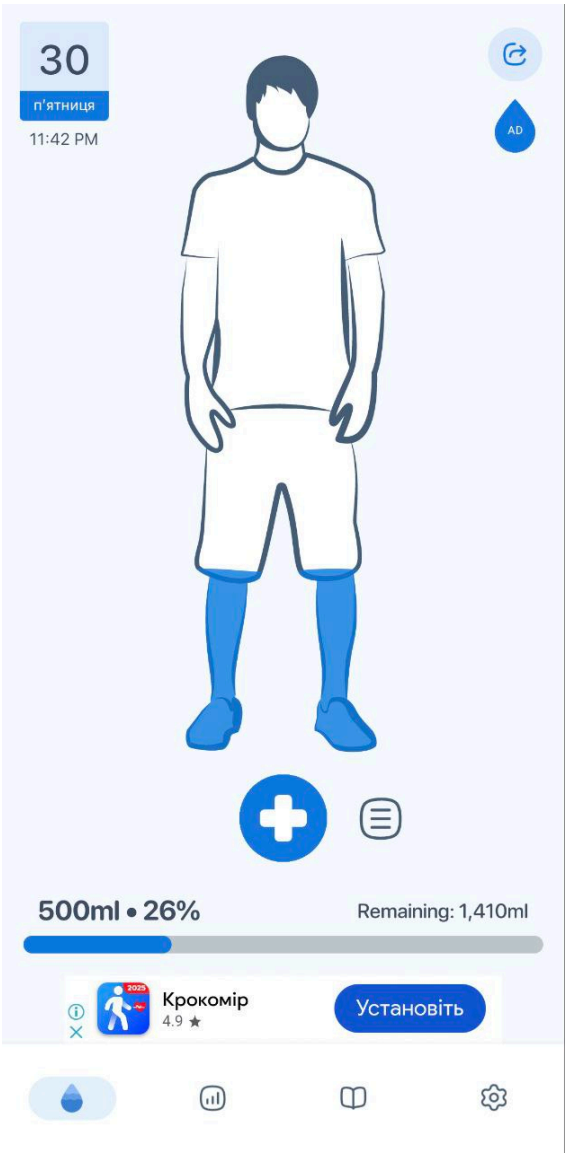

Рисунок 1.1. Екран застосунку WaterMinder

Hydro Coach – це застосунок для моніторингу водного балансу, доступний на Android, але також сумісний із iOS. Він пропонує розширені функції, такі як детальна категоризація напоїв із урахуванням їхнього гідратаційного внеску, інтеграція з Google Fit, Fitbit і Samsung Health, а також експорт статистики у CSV-файли. Застосунок розраховує норму води на основі особистих параметрів і рівня активності, надаючи детальні звіти та нагадування. Hydro Coach орієнтований на користувачів, які прагнуть глибокого аналізу своїх звичок споживання води (Рис.1.2).

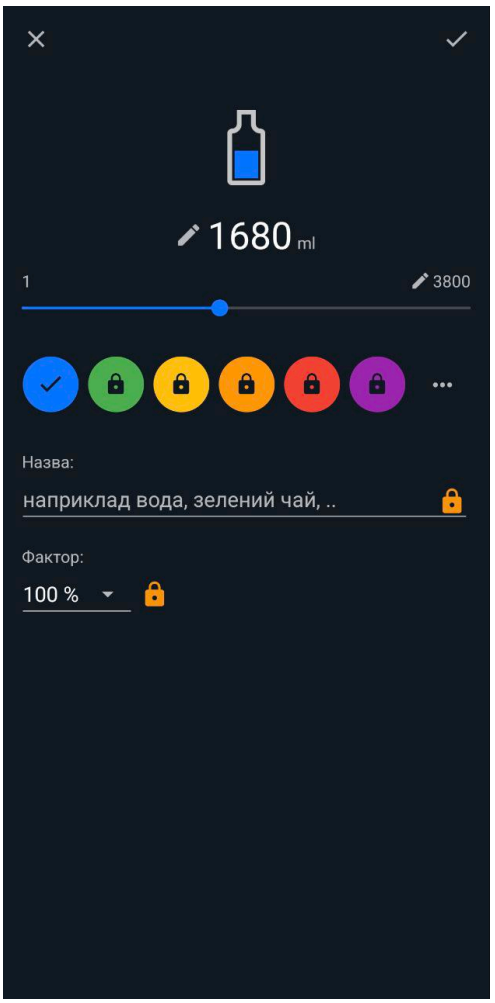

Рисунок 1.2. Екран застосунку Hydro Coach

Для оцінки функціональності WaterMinder і Hydro Coach було проаналізовано їхні ключові можливості і наведено огляд основних функцій з їх перевагами та недоліками.

Візуалізація прогресу за допомогою кругової діаграми:

Опис функціональності: Обидва застосунки використовують кругові діаграми для відображення щоденного прогресу споживання води. У WaterMinder діаграма показує відсоток виконаної норми та об’єм у мілілітрах, оновлюючись у реальному часі після додавання запису про напій. У Hydro Coach діаграма відображає прогрес також в мілілітрах з літрів загальної потреби споживання. Обидва застосунки підтримують кольорове кодування, де колір діаграми відповідає типу доданого напою (наприклад, синій для води, зелений для чаю), що відображається як у діаграмі, так і в статистиці. Однак ця функція кастомізації доступна лише в платних версіях обох застосунків (Рис.1.3).

Переваги: Кругова діаграма є інтуїтивним і візуально привабливим інструментом, зрозумілим навіть для початківців.

Недоліки: У безкоштовній версії Hydro Coach діаграма обмежена відображенням загального об’єму за типами напоїв, а кастомізація кольорів недоступна. У WaterMinder безкоштовна версія також не дозволяє налаштувати стиль діаграми чи додати деталізацію, наприклад, розподіл за типами напоїв.

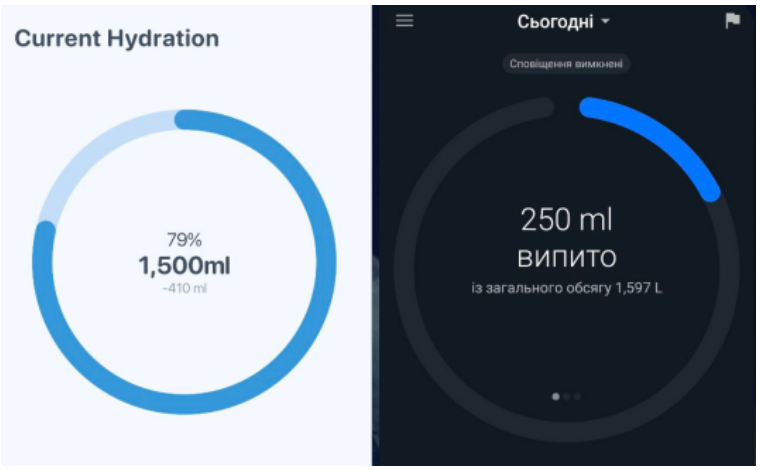

Рисунок 1.3. Кругові діаграми застосунків WaterMinder та Hydro Coach

Налаштування профілю користувача:

- **Опис функціональності:** Обидва застосунки дозволяють створювати профіль із зазначенням імені, статі, ваги, віку та рівня активності. Підтримується також автентифікація через Google для швидкого створення профілю. На основі введених даних розраховується індивідуальна норма споживання води. Обидва застосунки включають елементи гейміфікації, такі як досягнення (наприклад, за виконання норми протягом тижня).
- **Переваги**: Персоналізація профілю забезпечує загалом точні рекомендації щодо норми води. Інтеграція з Google для автентифікації спрощує реєстрацію, а гейміфікація мотивує користувачів регулярно споживати воду.
- **Недоліки**: У Hydro Coach зміна параметрів (наприклад, ваги) є ускладненою, замість введення відповідного значення доводиться проводити пальцем повзунок, що може призвести до випадкового закриття меню зі зміни ваги і прогрес скидається, а у WaterMinder валідація введених даних недостатньо сувора (наприклад, допускаються нереалістичні значення ваги (Рис.1.4)). Експорт даних профілю чи статистики споживання у CSV доступний лише в платних версіях. Крім того, обидва застосунки перенасичені рекламою, яка з’являється після кожного дії, що погіршує користувацький досвід.

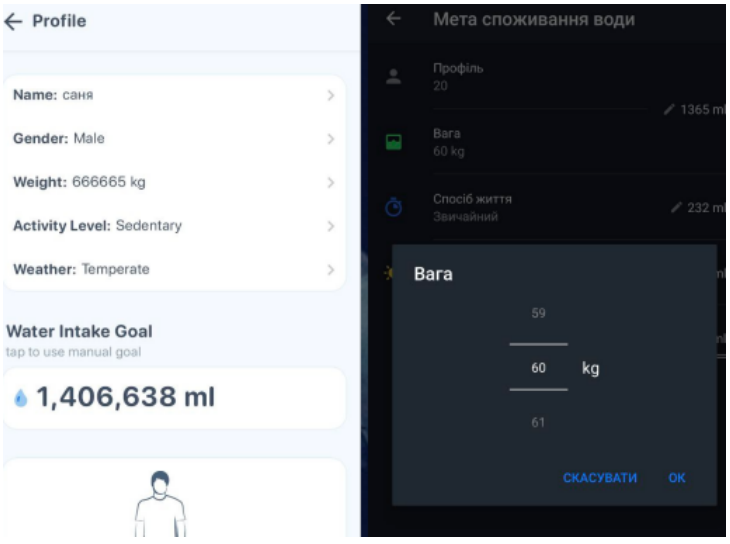

Рисунок 1.4. Зміна особистих даних (ваги) в застосунках WaterMinder та Hydro Coach

Фіксація спожитих напоїв:

Опис функціональності: Користувачі можуть додавати напої, обираючи тип і об’єм. У WaterMinder доступний обмежений набір напоїв у безкоштовній версії (вода, чай, кава) із фіксованими об’ємами (250 мл, 500 мл), а такі категорії, як смузі чи сік, доступні лише в платній версії. Hydro Coach дозволяє підписувати напої власними назвами, але кастомізація гідратаційного фактора (наприклад, вплив кави на норму) також вимагає платної підписки. Обидва застосунки підтримують редагування та видалення записів, але інтерфейс Hydro Coach є більш громіздким через велику кількість опцій.

Переваги: Можливість фіксації напоїв є базовою і зручною функцією. У платній версії Hydro Coach детальна категоризація напоїв і гідратаційні фактори підвищують точність відстеження.

Недоліки: У безкоштовній версії WaterMinder арсенал напоїв дуже обмежений, а параметри (наприклад, об’єм) не редагуються. У Hydro Coach кастомізація напоїв у безкоштовній версії зводиться до назви, а всі розширені функції платні. Обидва застосунки активно пропонують купити PRO-версію (Рис.1.5).

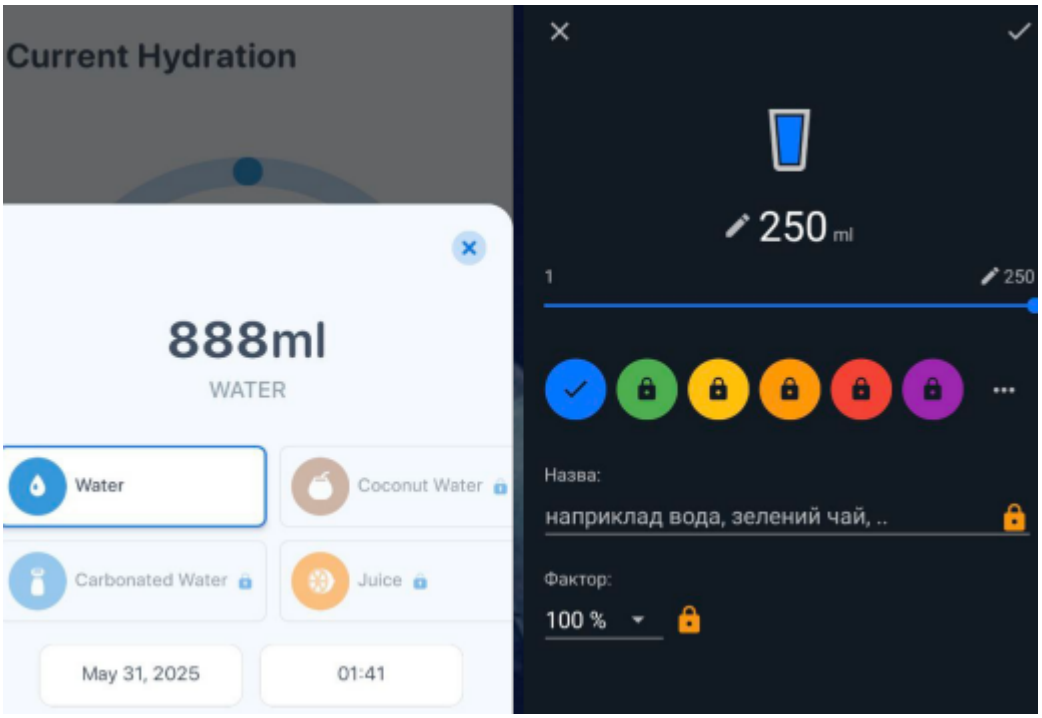

Рисунок 1.5. Вибір напоїв в застосунках WaterMinder та Hydro Coach

Аналіз статистики споживання:

- **Опис функціональності**: Обидва застосунки надають статистику споживання води у вигляді графіків і списків. WaterMinder пропонує деталізовану статистику по днях, тижнях, місяцях і роках, але лише в платній версії. Hydro Coach також підтримує розширені звіти з можливістю експорту в CSV, але ця функція доступна тільки за підпискою. Обидва застосунки використовують кольорове кодування для типів напоїв у статистиці, якщо активована платна версія.
- **Переваги**: Детальна статистика мотивує користувачів і дозволяє аналізувати тенденції. Кольорове кодування у платній версії робить звіти більш наочними.
- **Недоліки**: У безкоштовних версіях застосунків статистика обмежена базовими графіками без деталізації за типами напоїв. У Hydro Coach статистика за місяць та рік доступна лише в платній версії, також не інтуїтивне зображення графіку статистики за тиждень (Рис.1.6). Постійні рекламні банери та пропозиції купити PRO-версію погіршують досвід використання.

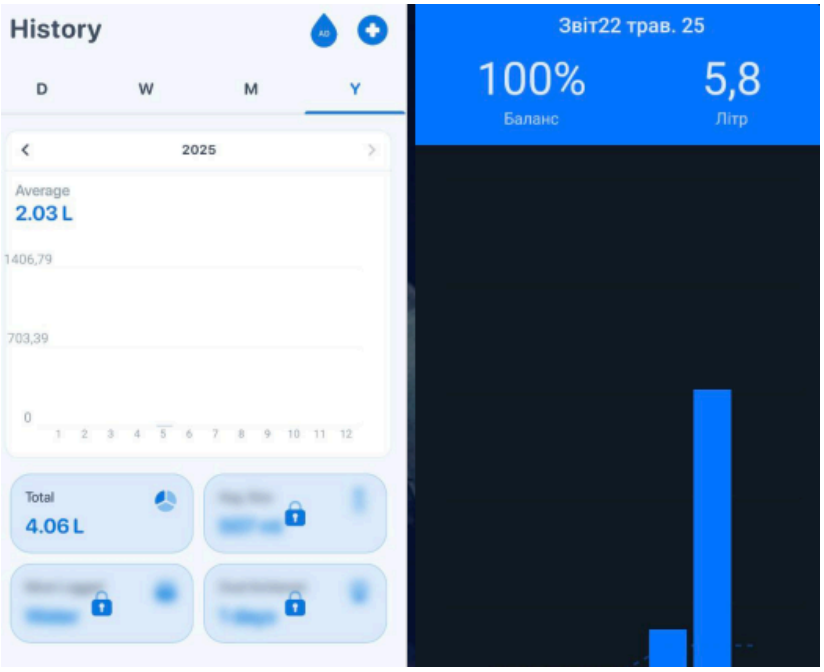

Рисунок 1.6. Перегляд статистики в застосунках WaterMinder за рік та Hydro Coach за тижнем

1.3 Вимоги до застосунку

На основі аналізу предметної області, потреб користувачів та оцінки існуючих рішень (WaterMinder і Hydro Coach) сформульовано вимоги до кросплатформного мобільного застосунку HydrationApp для моніторингу водного балансу. Вимоги враховують виявлені переваги та недоліки конкурентів, сучасні технологічні стандарти та орієнтацію на зручність і доступність для користувачів. Вимоги поділено на функціональні, нефункціональні, з урахуванням того, що деякі функції, такі як розширена кастомізація напоїв, можуть бути реалізовані в майбутніх версіях застосунку.

Функціональні вимоги:

1. **Автентифікація користувачів**: Застосунок має забезпечувати безпечну автентифікацію через email і пароль із можливістю реєстрації, входу та скидання пароля. Реєстрація повинна включати введення особистих даних (ім’я, стать, вага, вік, дата народження) для персоналізації норми споживання води. Валідація введених даних: email має відповідати формату (наявність @ і домену), пароль — містити щонайменше 8 символів, вага та вік — бути реалістичними (наприклад, вага > 0 кг, вік > 0 років).

2. **Профіль користувача**: Користувач має мати можливість створювати та редагувати профіль із зазначенням особистих параметрів (ім’я, стать, вага, вік тощо). Норма споживання води розраховується автоматично на основі формули: вага × 35 мл для чоловіків, вага × 30 мл для жінок, із коригуванням на 10% зменшення для користувачів старше 50 років. Інтерфейс профілю має бути інтуїтивним, дозволяючи легко оновлювати дані без обмежень, на відміну від Hydro Coach, де редагування ускладнене.

3. **Фіксація спожитих напоїв**: Користувач може додавати записи про спожиті напої, обираючи тип (вода, чай, кава, сік тощо) і об’єм (у мілілітрах). На відміну від WaterMinder, де безкоштовна версія обмежує вибір напоїв, HydrationApp має надавати широкий набір типів напоїв без платних обмежень. Підтримка редагування та видалення записів про напої для виправлення помилок.

4. **Візуалізація прогресу**: Прогрес споживання води відображається через кругову діаграму, яка показує виконану норму та об’єм у мілілітрах, подібно до WaterMinder. Діаграма оновлюється в реальному часі після додавання/видалення запису про напій.

5. **Аналіз статистики**: Застосунок надає статистику споживання води у вигляді обраного дня за календарем, з повним прогресом та історією споживання. Статистика включає загальний об’єм спожитої води та розподіл за типами напоїв, усуваючи обмеження Hydro Coach, де деталізація доступна лише за підпискою. Інтерфейс статистики має бути інтуїтивним, уникаючи перевантаженості, як у платній версії WaterMinder.

6. **Навігація та інтерфейс**: Навігація між екранами має бути простою та швидкою, з чіткою структурою, що усуває громіздкість інтерфейсу Hydro Coach.

Нефункціональні вимоги:

1. **Кросплатформність**: Застосунок має бути сумісним із платформами Android та iOS для забезпечення єдиного коду та однакового користувацького досвіду.

2. **Інтеграція з хмарними сервісами**: Використання безпечної автентифікації та хмарного зберігання даних (профіль, записів про спожиті напої тощо). Синхронізація даних в реальному часі. Захист даних, який дає доступ лише для авторизованих користувачів.

3. **Валідація та стабільність**: Усі введені дані (email, пароль, ім’я, вага, вік) проходять сувору валідацію для запобігання помилкам, на відміну від WaterMinder, де валідація недостатньо чітка. Застосунок має бути стабільним, без збоїв під час навігації, збереження даних чи оновлення інтерфейсу.

4. **Зручність інтерфейсу**: Інтерфейс має бути мінімалістичним, інтуїтивним і вільним від реклами, на відміну від WaterMinder і Hydro Coach, де реклама погіршує досвід. Усі функції (додавання напоїв, перегляд прогресу, статистика) доступні без платних підписок, усуваючи обмеження конкурентів.

5. **Продуктивність**: Час відгуку інтерфейсу не має перевищувати 1 секунди при додаванні напоїв чи оновленні діаграми. Застосунок має ефективно працювати на пристроях із різними технічними характеристиками.

6. **Безпека даних**: Персональні дані користувачів (email, пароль, профіль) мають захищатися через шифрування. Доступ до даних обмежується авторизованими користувачами, що відповідає стандартам безпеки.

2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ

2.1 Обґрунтування вибору інструментальних засобів

Для розробки кросплатформного мобільного застосунку HydrationApp обрано набір інструментальних засобів, які забезпечують ефективність, зручність розробки, стабільність і відповідність вимогам щодо кросплатформності, безпеки, реального часу оновлення даних та інтуїтивності інтерфейсу. Обрані технології включають: React Native, Firebase (Authentication і Firestore), Expo Go, Visual Studio Code, AsyncStorage, а також додаткові бібліотеки: react-navigation, react-native-keyboard-aware-scroll-view і @expo/vector-icons тощо. Вибір кожної технології обґрунтовано їхніми перевагами, порівнянням з альтернативами та відповідністю функціональним і нефункціональним вимогам застосунку.

React Native [3] обрано як основну платформу для розробки HydrationApp через її здатність створювати кросплатформні застосунки для Android та iOS з єдиною кодовою базою. Це значно скорочує час і ресурси на розробку порівняно з нативними технологіями, такими як Swift (для iOS) або Kotlin (для Android). React Native використовує JavaScript і React, що дозволяє розробникам із досвідом веб-розробки швидко адаптуватися до створення мобільних застосунків. За даними офіційної документації React Native, платформа забезпечує високу продуктивність завдяки нативним компонентам, які компілюються в машинний код, що усуває значні відмінності в швидкодії порівняно з нативними застосунками.

Порівняно з альтернативами, такими як Flutter або Xamarin, React Native має більшу спільноту розробників і ширший набір бібліотек, що спрощує інтеграцію додаткових функцій, наприклад, навігації чи роботи з хмарними сервісами. Flutter, хоча й пропонує високу продуктивність і власний рендеринг інтерфейсу, вимагає вивчення мови Dart, що може бути бар’єром для команд, орієнтованих на JavaScript. Xamarin, своєю чергою, менш популярний і має обмежену підтримку бібліотек.

React Native також підтримує гаряче перезавантаження (hot reloading), що прискорює процес розробки та тестування, що є критично важливим для дотримання строків проєкту (Рис.2.1).

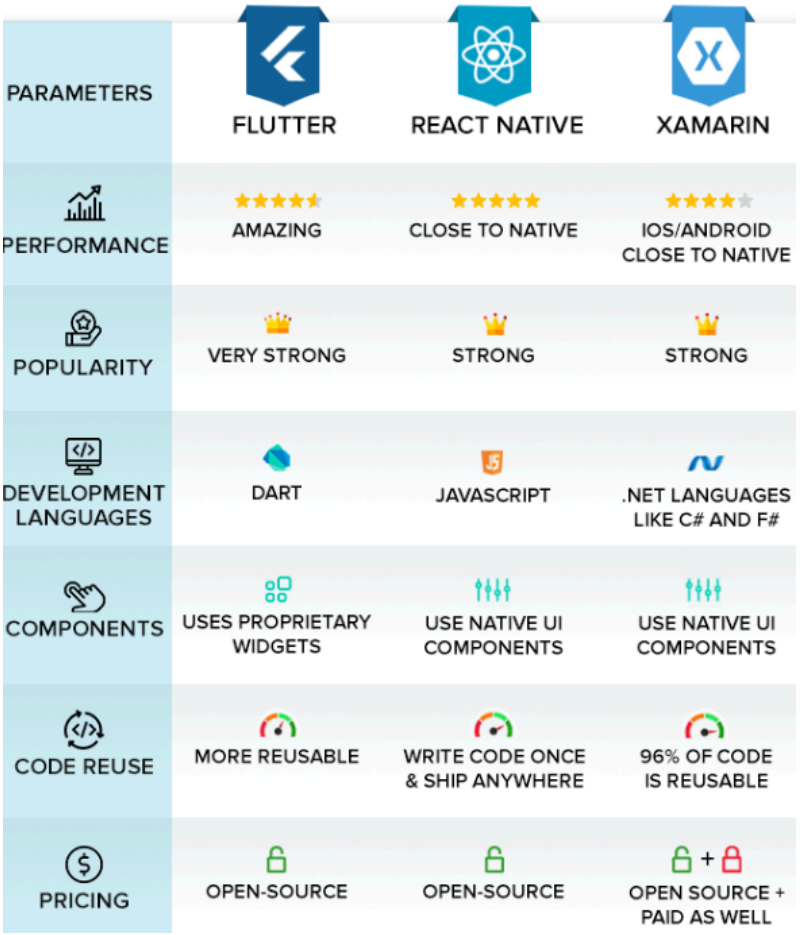

Рисунок 2.1. Порівняння Flutter, React Native та Xamarin між собою

Firebase [4] обрано як основну платформу для автентифікації та зберігання даних завдяки її інтеграції з React Native, простоті налаштування та підтримці функцій, необхідних для HydrationApp. Firebase Authentication забезпечує безпечну автентифікацію через email і пароль із вбудованими функціями реєстрації, входу та скидання пароля (sendPasswordResetEmail). Це усуває потребу в розробці власної системи автентифікації, що заощаджує час і знижує ризик помилок безпеки. Порівняно з альтернативами, такими як Auth0 або власний сервер автентифікації на Node.js, Firebase Authentication є більш економічним і простішим у налаштуванні, оскільки не вимагає окремого серверного адміністрування.

Firestore, хмарна NoSQL база даних Firebase, обрана для зберігання даних користувачів (профілі, записи про спожиті напої) завдяки її підтримці реального часу оновлення даних через слухачів onSnapshot. Це дозволяє оновлювати прогрес і статистику в реальному часі без додаткових запитів. Firestore також забезпечує масштабованість і автоматичну синхронізацію даних між пристроями, що відповідає вимогам кросплатформності. Firestore rules дозволяють обмежити доступ до даних лише автентифікованим користувачам із відповідним UID, забезпечуючи безпеку персональних даних. Порівняно з альтернативами, такими як MongoDB або Realtime Database (також від Firebase), Firestore пропонує кращу гнучкість для структури даних NoSQL і простішу інтеграцію з Firebase Authentication. Realtime Database, хоча й підтримує реальний час, менш ефективна для складних запитів, а MongoDB вимагає окремого серверного налаштування, що ускладнює розробку.

Expo Go [5] обрано як інструмент для спрощення розробки та тестування HydrationApp. Ця платформа дозволяє швидко запускати застосунок на реальних пристроях Android та iOS без необхідності налаштування нативного середовища (Xcode для iOS або Android Studio). Expo Go підтримує гаряче перезавантаження та забезпечує доступ до API для роботи з камерою, файлами та push-повідомленнями, що використано, наприклад, для функції вибору аватара (через ImagePicker). Порівняно з чистим React Native CLI, Expo Go зменшує складність конфігурації проєкту, що є перевагою для студентського проєкту з обмеженим часом. Однак Expo має обмеження, такі як неможливість використання деяких нативних модулів, але для HydrationApp ці обмеження не є критичними, оскільки всі необхідні функції (автентифікація, зберігання даних, навігація) підтримуються.

Visual Studio Code [6] (VS Code) обрано як основне середовище розробки завдяки його універсальності, підтримці JavaScript і React Native, а також широкому набору розширень.

VS Code забезпечує автодоповнення коду, налагодження (debugging) і інтеграцію з Git, що полегшує управління кодовою базою. Розширення, такі як ESLint і Prettier, допомагають підтримувати чистий і консистентний код, що є важливим для командної розробки або подальшого масштабування проєкту (Рис.2.2). Порівняно з альтернативами, такими як WebStorm або Android Studio, VS Code є безкоштовним, легким і більш універсальним, оскільки підтримує не лише React Native, а й роботу з Firebase та іншими інструментами.

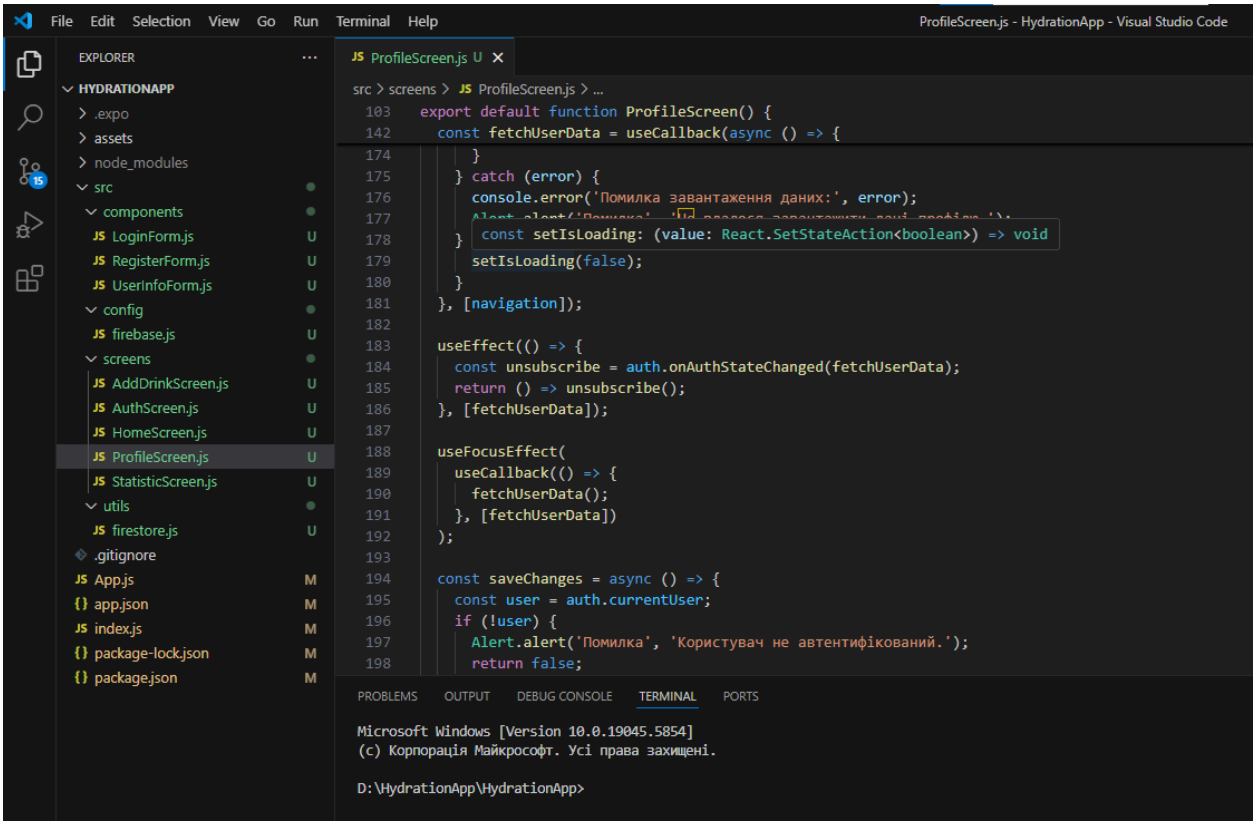

Рисунок 2.2. Фрагмент коду та загальний огляд середовища розробки Visual Studio Code

AsyncStorage [7] обрано для локального зберігання даних, зокрема часу останнього входу користувача, що дозволяє контролювати тривалість сесії (24 години). Ця бібліотека є стандартним рішенням для React Native, забезпечуючи просте асинхронне зберігання даних у форматі ключ-значення. AsyncStorage обрано замість SecureStore або SQLite, оскільки для збереження часу входу не потрібні складні структури чи додаткові бібліотеки.

SecureStore більше підходить для чутливих даних, таких як паролі, але в HydrationApp паролі обробляються Firebase Authentication, що робить AsyncStorage достатнім і ефективним рішенням.

Додаткові бібліотеки:

1. **react-navigation** [8]: Обрано для реалізації навігації між екранами через createStackNavigator. Бібліотека підтримує плавні переходи, управління стеком екранів і є стандартом для React Native. Порівняно з react-router-native, react-navigation краще оптимізована для мобільних застосунків і має ширшу підтримку спільноти.

2. **react-native-keyboard-aware-scroll-view** [9]: Використано для коректної обробки клавіатури на екранах із формами. Бібліотека автоматично прокручує екран, щоб поля введення залишалися видимими, покращуючи користувацький досвід. Альтернативи, такі як вбудована KeyboardAvoidingView, менш гнучкі та можуть викликати проблеми з позиціонуванням.

3. **@expo/vector-icons** [10]: Обрано для додавання іконок (наприклад, для кнопок навігації чи типів напоїв) завдяки широкому набору готових іконок і простій інтеграції з Expo. Порівняно з react-native-vector-icons, @expo/vector-icons не вимагає додаткового налаштування нативних модулів, що спрощує використання в проєкті на базі Expo.

2.2 Архітектура застосунку

Архітектура застосунку HydrationApp розроблена з урахуванням вимог до кросплатформності, безпеки, зручності інтерфейсу та реального часу оновлення даних, визначених у розділі 1.3. Основна мета архітектури – забезпечити модульність, легкість інтеграції з хмарними сервісами та стабільність роботи на платформах Android та iOS. Для цього застосунок побудовано на основі React Native з використанням Firebase (Authentication і

Firestore) для автентифікації та зберігання даних, а також Expo Go для спрощення розробки й тестування.

Архітектура включає чіткий розподіл на клієнтську частину (інтерфейс користувача та логіку взаємодії) і серверну частину (хмарне сховище та автентифікацію), що забезпечує гнучкість і можливість масштабування в майбутньому.

Клієнтська частина HydrationApp складається з п’яти основних екранів, які реалізують функціонал, описаний у функціональних вимогах (розділ 1.3). Кожен екран є окремим React-компонентом, що взаємодіє з Firebase через спеціально розроблені функції доступу до даних, визначені в модулі firestore.js. Навігація між екранами реалізована за допомогою бібліотеки react-navigation, яка забезпечує плавні переходи та управління стеком екранів:

AuthScreen: Відповідає за автентифікацію користувачів. Екран включає компоненти LoginForm, RegisterForm і UserInfoForm, які забезпечують вхід, реєстрацію та введення персональних даних (стать, вага, дата народження). AuthScreen взаємодіє з Firebase Authentication для обробки облікових даних і з Firestore для ініціалізації профілю через функцію initUserProfile. Валідація введених даних (email, пароль, вага) проводиться на клієнтській стороні для підвищення зручності та зменшення навантаження на сервер.

HomeScreen: Головний екран, який відображає прогрес споживання води через кругову діаграму. Цей екран підписується на зміни в Firestore через функцію listenToWaterIntake, що дозволяє оновлювати дані в реальному часі. HomeScreen також забезпечує навігацію до інших екранів (ProfileScreen, AddDrinkScreen, StatisticScreen).

AddDrinkScreen: Призначений для фіксації спожитих напоїв. Користувач може обрати тип напою та вказати об’єм, після чого дані записуються в Firestore через функцію updateDoc. Екран інтегрується з HomeScreen для оновлення прогресу після додавання нового запису.

ProfileScreen: Дозволяє користувачу переглядати та редагувати особисті дані (ім’я, стать, вага, дата народження, аватар, пароль).

Дані оновлюються в Firestore через запит updateUserData, а також автоматично перераховується норма споживання води на основі змінених параметрів.

StatisticScreen: Відображає статистику споживання води за обраний день. Дані отримуються через запит getDocs до Firestore і представлені у вигляді списку або діаграми, що дозволяє користувачу аналізувати тенденції споживання.

Клієнтська частина також включає допоміжні компоненти та бібліотеки, такі як react-native-keyboard-aware-scroll-view для коректного відображення форм при появі клавіатури, та @expo/vector-icons для використання іконок у інтерфейсі. Локальне зберігання часу останнього входу реалізовано через AsyncStorage, що дозволяє завершувати сесію після 24 годин.

Серверна частина HydrationApp базується на хмарних сервісах Firebase:

- **Firebase Authentication**: Відповідає за безпечну автентифікацію через email і пароль. Використовуються функції signInWithEmailAndPassword, createUserWithEmailAndPassword і sendPasswordResetEmail для входу, реєстрації та скидання пароля. Кожен користувач отримує унікальний ідентифікатор (UID), який використовується для зв’язку з даними в Firestore.
- **Firestore**: Використовується як NoSQL база даних для зберігання профілів користувачів і записів про спожиті напої:
  - Колекція users/{userId}: Зберігає профіль користувача (ім’я, стать, вага, дата народження).

- Підколекція users/{userId}/waterIntake/{date}: Зберігає записи про спожиті напої за конкретний день. Firestore підтримує реальний час оновлення через слухачів onSnapshot, що використовуються в HomeScreen для відображення актуального прогресу. Безпека даних забезпечується через Firestore rules [11], які дозволяють доступ до даних лише автентифікованим користувачам із відповідним UID.

Для наочного представлення взаємодії між компонентами розроблено діаграму компонентів (Рис.2.3). Діаграма показує, як користувач (User) взаємодіє з HydrationApp через Expo Go, яке запускає застосунок шляхом сканування QR-коду. HydrationApp виступає головним контейнером, що об'єднує екрани (AuthScreen, HomeScreen, AddDrinkScreen, ProfileScreen, StatisticScreen). Кожен екран взаємодіє з Firebase: AuthScreen звертається до Firebase Authentication для автентифікації, а інші екрани – до Firestore для читання та запису даних. Наприклад, HomeScreen використовує слухачів для оновлення прогресу, а AddDrinkScreen надсилає запити updateDoc для збереження нових записів.

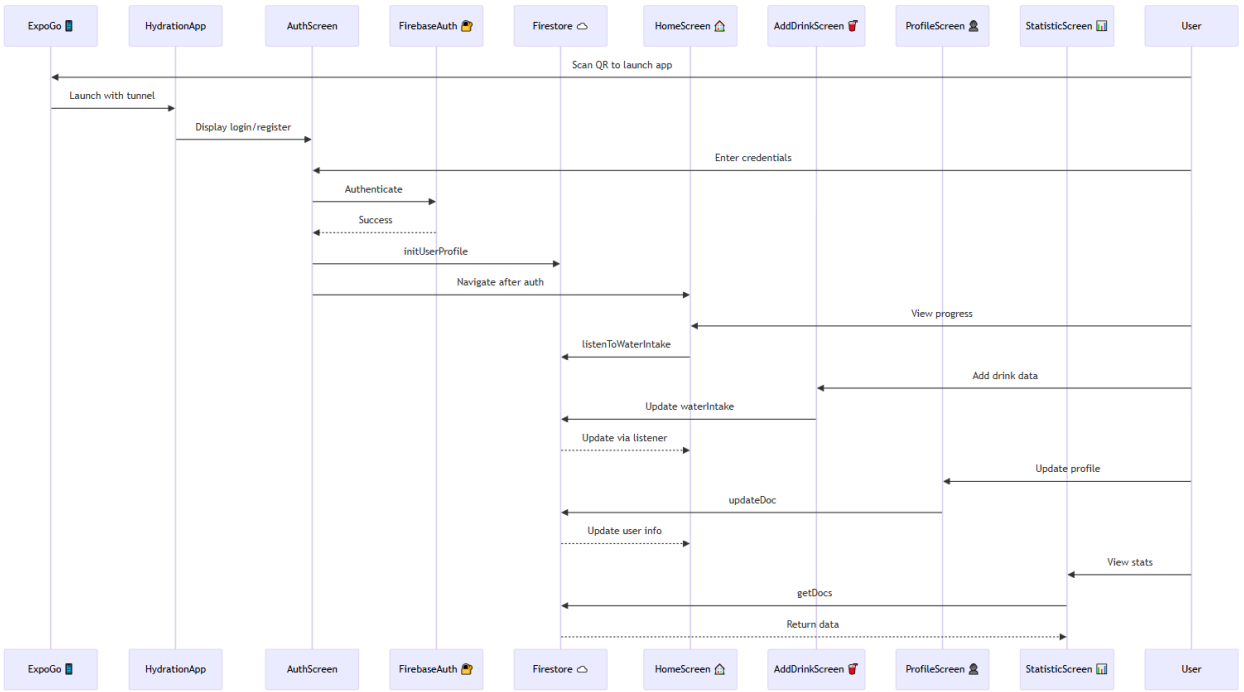

Рисунок 2.3. Діаграма компонентів HydrationApp

Для опису сценаріїв взаємодії користувача із системою розроблено діаграму прецедентів (Use Case), яка представлена на рисунку 2.4. Діаграма включає основні сценарії використання: автентифікацію (вхід, реєстрація, скидання пароля), перегляд і оновлення профілю, додавання напоїв, перегляд прогресу та статистики, а також автоматичне завершення сесії після 24 годин. Акторами виступають користувач (User), Firebase (для автентифікації та зберігання даних) і Time (для контролю сесії). Відносини типу <<include>> і <<extend>> показують залежності між прецедентами, наприклад, «Реєстрація» включає «Введення додаткової інформації», а «Скидання пароля» розширює «Вхід у систему».

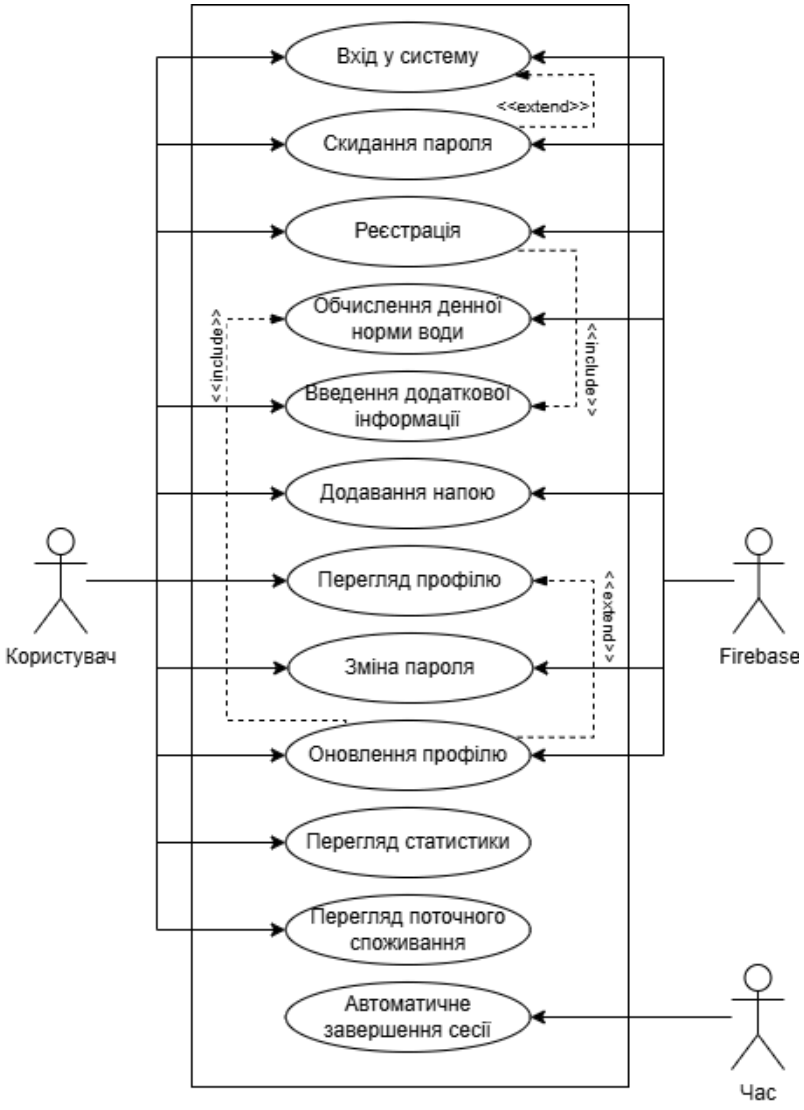

Рисунок 2.4. Діаграма прецендентів HydrationApp

Архітектура забезпечує виконання нефункціональних вимог:

- **Кросплатформність**: React Native і Expo Go дозволяють запускати застосунок на Android та iOS з єдиною кодовою базою, забезпечуючи однаковий користувацький досвід.
- **Безпека**: Firebase Authentication і Firestore rules гарантують захист даних, дозволяючи доступ лише автентифікованим користувачам.
- **Продуктивність**: Використання слухачів Firestore і локального кешування через AsyncStorage зменшує кількість запитів до сервера, забезпечуючи швидкий відгук інтерфейсу.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

3.1 Опис та реалізація компонентів застосунку

Розроблений кросплатформний мобільний застосунок HydrationApp побудовано на основі React Native з інтеграцією Firebase для автентифікації та хмарного зберігання даних. Застосунок складається з п’яти основних екранів (компонентів), кожен із яких відповідає за певний функціонал, визначений у функціональних вимогах. Компоненти взаємодіють між собою через бібліотеку react-navigation для навігації та з Firebase через модуль firestore.js для доступу до даних. Наведемо детальний опис кожного компонента, його призначення, взаємодію з іншими частинами системи та ключові особливості реалізації.

AuthScreen:

Призначення: AuthScreen (Рис.3.1) є початковим екраном застосунку, який відповідає за автентифікацію користувачів (вхід, реєстрація, введення персональних даних) та скидання пароля. Цей екран забезпечує безпечний доступ до застосунку, використовуючи Firebase Authentication, і дозволяє ініціалізувати профіль користувача в Firestore.

Функціонал: Відображення трьох підкомпонентів: LoginForm (форма входу), RegisterForm (форма реєстрації) та UserInfoForm (форма введення персональних даних). Валідація введених даних: перевірка формату email (наявність символу «@» та домену), пароля (не менше 8 символів, включаючи літери та цифри), імені (лише літери), ваги (реалістичні значення, наприклад, від 30 до 200 кг), віку (від 1 до 120 років) та дати народження (коректний формат). Виклик функцій Firebase Authentication (signInWithEmailAndPassword, createUserWithEmailAndPassword, sendPasswordResetEmail) для обробки входу, реєстрації та скидання пароля. Ініціалізація профілю користувача в Firestore через функцію initUserProfile з модуля firestore.js після успішної реєстрації.

Збереження часу останнього входу в AsyncStorage для контролю сесії (24 години).

Взаємодія: Звертається до Firebase Authentication для автентифікації та до Firestore для створення профілю. Після успішного входу або реєстрації перенаправляє користувача на HomeScreen через react-navigation. Використовує react-native-keyboard-aware-scroll-view для коректного відображення форм при появі клавіатури.

Особливості реалізації: Валідація на клієнтській стороні зменшує кількість некоректних запитів до сервера. Інтерфейс мінімалістичний, з використанням іконок з @expo/vector-icons для кнопок входу та скидання пароля. Обробка помилок автентифікації (наприклад, «користувач не знайдений» або «невірний пароль») з відображенням зрозумілих повідомлень користувачу.

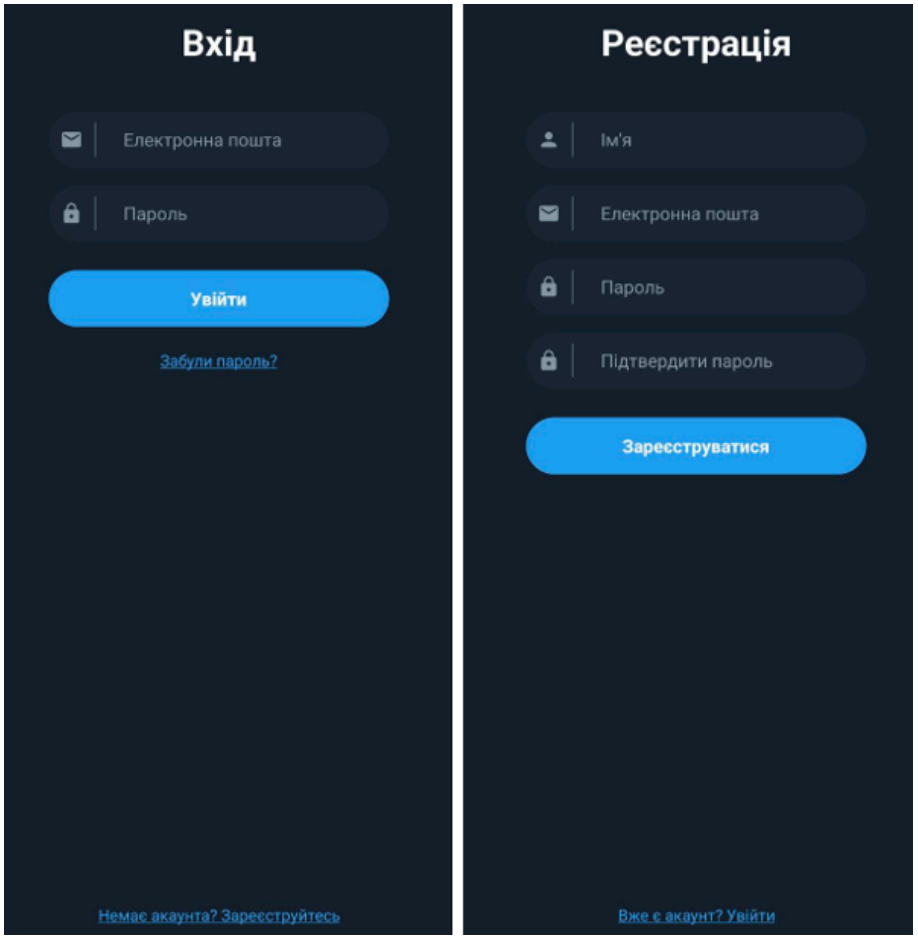

Рисунок 3.1. Інтерфейс AuthScreen із формами входу та реєстрації

HomeScreen:

Призначення: HomeScreen (Рис.3.2) є основним екраном застосунку, який відображає прогрес споживання води за поточний день у вигляді кругової діаграми та забезпечує навігацію до інших екранів.

Функціонал: Відображення кругової діаграми, яка показує відсоток виконаної норми споживання води (розраховується на основі параметрів користувача: вага × 35 мл для чоловіків, вага × 30 мл для жінок, з коригуванням для віку > 50 років). Слухач на оновлення даних у реальному часі через функцію listenToWaterIntake з модуля firestore.js, яка отримує записи про спожиті напої з підколекції users/{userId}/waterIntake/{date}. Навігаційні кнопки для переходу до AddDrinkScreen, ProfileScreen та StatisticScreen. Відображення поточного об’єму спожитої води в мілілітрах та порівняння з добовою нормою.

Взаємодія: Отримує дані користувача та записи про напої з Firestore через слухачів onSnapshot. Використовує react-navigation для переходу до інших екранів. Оновлює діаграму в реальному часі після додавання або видалення напоїв через AddDrinkScreen.

Особливості реалізації: Кругова діаграма реалізована за допомогою бібліотеки react-native-progress [12], що забезпечує плавну анімацію. Інтерфейс оптимізований для швидкого відгуку (менше 1 секунди) завдяки локальному кешуванню даних через AsyncStorage. Використання іконок з @expo/vector-icons для навігаційних кнопок.

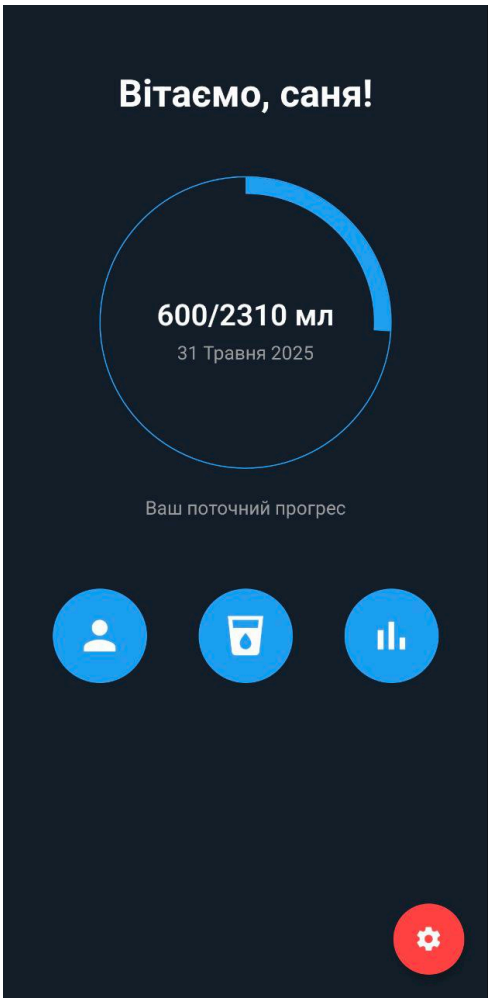

Рисунок 3.2. Інтерфейс HomeScreen із круговою діаграмою прогресу

AddDrinkScreen:

Призначення: AddDrinkScreen (Рис.3.3) дозволяє користувачу фіксувати спожиті напої, вказуючи їх тип і об’єм, із подальшим збереженням даних у Firestore.

Функціонал: Вибір типу напою з попередньо визначеного списку (вода, чай, кава, сік тощо) через інтерактивний список. Введення об’єму напою в мілілітрах (з валідацією: значення > 0). Збереження запису в Firestore через функцію updateDoc у підколекцію users/{userId}/waterIntake/{date}. Можливість редагування або видалення раніше доданих записів. Кнопка підтвердження для збереження даних і повернення на HomeScreen.

Взаємодія: Звертається до Firestore для запису даних про напої. Оновлює HomeScreen через слухачів onSnapshot після додавання нового запису. Використовує react-native-keyboard-aware-scroll-view для коректного відображення форми введення.

Особливості реалізації: Валідація об’єму напою на клієнтській стороні запобігає некоректним записам. Використання іконок з @expo/vector-icons для позначення типів напоїв.

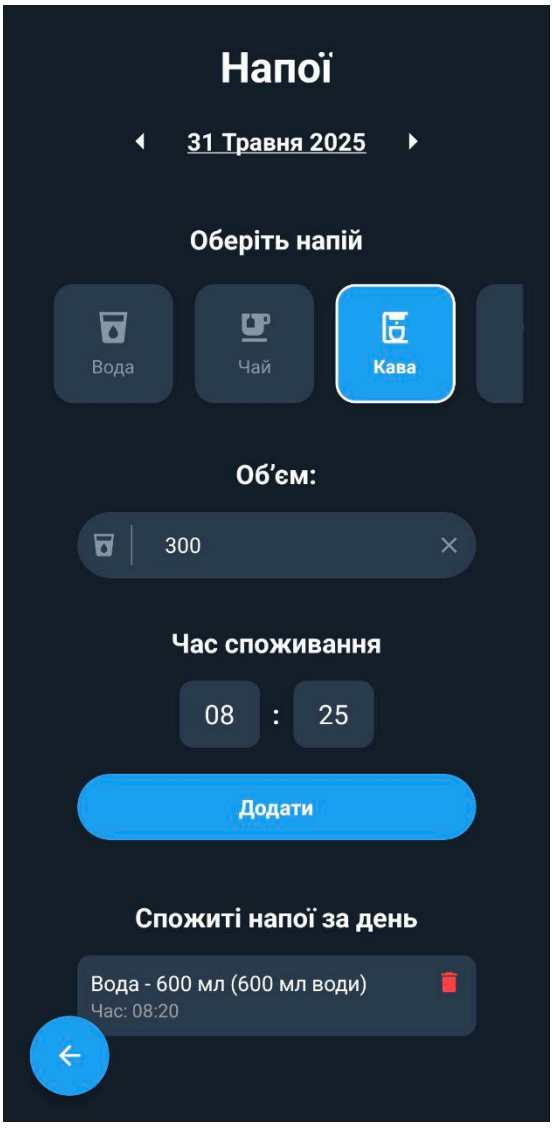

Рисунок 3.3. Інтерфейс AddDrinkScreen для фіксації напоїв

ProfileScreen:

Призначення: ProfileScreen (Рис.3.4) дозволяє користувачу переглядати та редагувати особисті дані, а також змінювати пароль і аватар.

Функціонал: Відображення поточних даних профілю (ім’я, стать, вага, вік, дата народження). Форма редагування даних із валідацією (аналогічно до UserInfoForm, RegisterForm). Зміна пароля через функцію sendPasswordResetEmail Firebase Authentication. Завантаження аватара через ImagePicker [13] з бібліотеки expo-image-picker. Оновлення даних у Firestore через запит updateDoc до колекції users/{userId}. Перерахунок норми споживання води після зміни параметрів (вага, стать, вік).

Взаємодія: Отримує дані профілю з Firestore через getDoc. Оновлює профіль у Firestore та локально через AsyncStorage для кешування. Використовує react-navigation для повернення на HomeScreen.

Особливості реалізації: Інтерфейс включає попередній перегляд аватара та кнопку для його зміни. Валідація введених даних забезпечує коректність профілю. Використання react-native-keyboard-aware-scroll-view для зручного редагування форм.

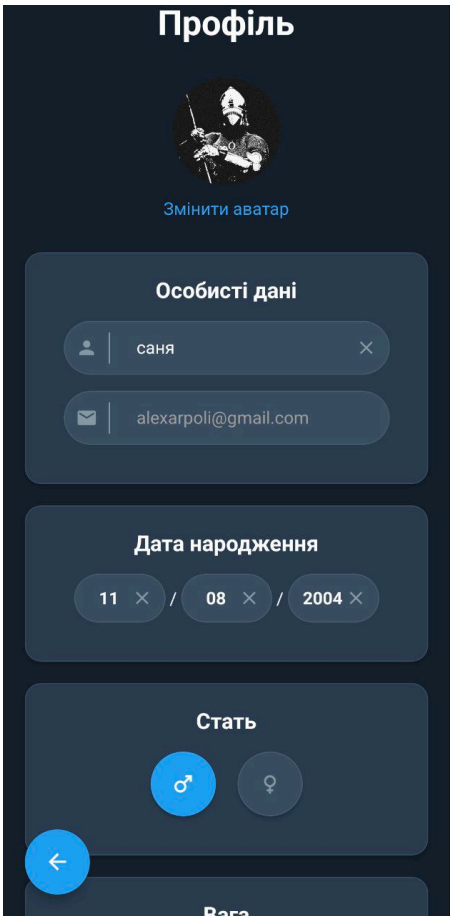

Рисунок 3.4. Інтерфейс ProfileScreen для редагування профілю

StatisticScreen:

Призначення: StatisticScreen (Рис.3.5) відображає детальну статистику споживання води за обраний день, включаючи загальний об’єм і розподіл за типами напоїв.

Функціонал: Вибір дати через календар (реалізований через react-native-modal-datetime-picker [14]). Відображення списку записів про спожиті напої за обраний день із вказівкою типу, об’єму та часу. Візуалізація статистики у вигляді діаграми за допомогою react-native-progress. Отримання даних із Firestore через запит getDocs до підколекції users/{userId}/waterIntake/{date}.

Взаємодія: Отримує дані з Firestore для відображення статистики. Використовує react-navigation для повернення на HomeScreen. Оновлює інтерфейс після вибору нової дати.

Особливості реалізації: Інтерфейс включає календар для швидкого вибору дати та діаграму для наочної візуалізації прогресу за обраний день. Дані кешуються локально через AsyncStorage для зменшення запитів до Firestore. Використання іконок з @expo/vector-icons для позначення типів напоїв у списку та кнопок.

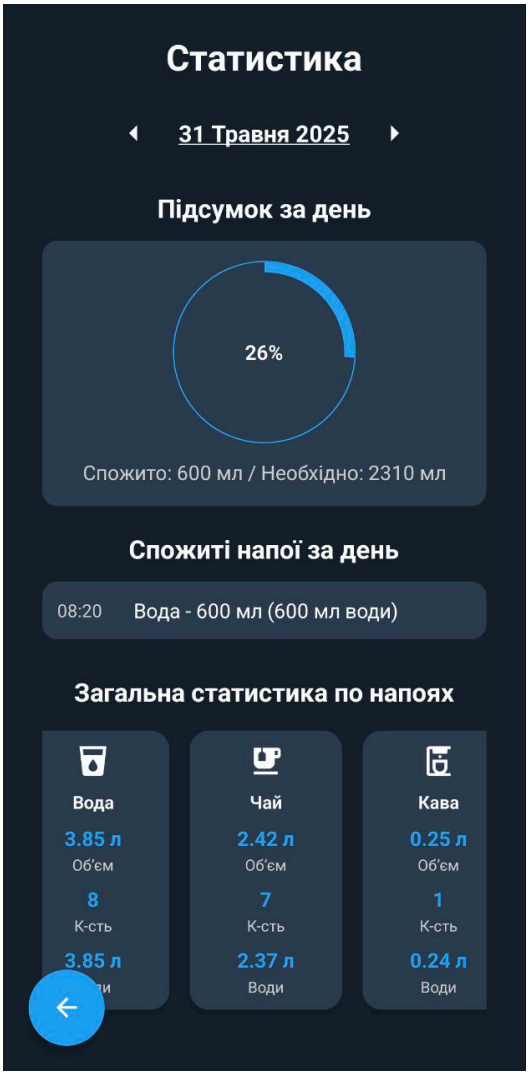

Рисунок 3.5. Інтерфейс StatisticScreen зі статистикою споживання

Допоміжні компоненти та модулі:

- **firestore.js**: модуль містить функції для взаємодії з Firestore, такі як initUserProfile (ініціалізація профілю), listenToWaterIntake (слухач на оновлення даних), updateDoc (збереження напоїв), getDocs (отримання статистики). Це забезпечує централізований доступ до бази даних і зменшує дублювання коду.

- **firebase.js**: модуль ініціалізації Firebase, що налаштовує підключення до Firebase Authentication і Firestore.
- **AsyncStorage**: використовується для локального зберігання часу останнього входу та кешування даних профілю.
- **react-navigation**: забезпечує навігацію між екранами через createStackNavigator, з плавними переходами та управлінням стеком.
- **react-native-keyboard-aware-scroll-view**: використовується для коректного відображення форм.
- **@expo/vector-icons**: надає іконки для кнопок і типів напоїв, покращуючи візуальну привабливість інтерфейсу.

3.2 Тестування застосунку

Тестування мобільного застосунку HydrationApp проведено для перевірки відповідності функціональним і нефункціональним вимогам, із забезпеченням стабільності та зручності на платформах Android та iOS. Тестування виконувалося через Expo Go з тунельним підключенням, що дозволяло розгортати застосунок на реальних пристроях (iPhone 11 Pro, iOS 17; Samsung Galaxy A35, Android 14) шляхом сканування QR-коду після запуску команди npx expo start --tunnel. Перевірено ключові функції, інтерфейс, навігацію, валідацію даних та інтеграцію з Firebase, а результати підтвердили коректну роботу застосунку.

Логи підтвердили коректне відключення слухачів Firestore, що забезпечило належне управління ресурсами (Рис.3.6).

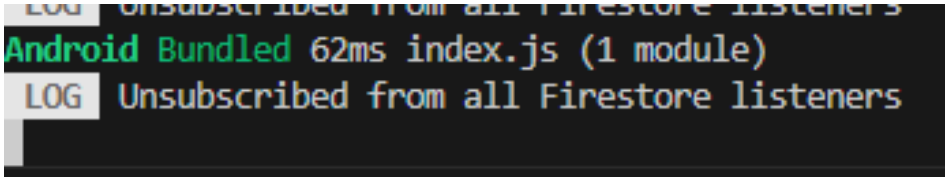

Рисунок 3.6. Логування підключеного Android пристрою

Автентифікація та валідація: Перевірено вхід, реєстрацію та скидання пароля через Firebase Authentication. Валідація форм (LoginForm, RegisterForm) виявляла помилки, наприклад, «Невірний формат email» або «Пароль має містити щонайменше 8 символів». Алерти відображалися коректно, попереджаючи про некоректні дані (Рис.3.7–3.8).

Профіль користувача: На ProfileScreen тестувалося редагування імені, статі, дати народження, ваги та аватара. Валідація блокувала некоректні значення (вага <20 кг, некоректна дата). Завантаження аватара через ImagePicker працювало стабільно, а норма води правильно перераховувалася (наприклад, 70 кг × 35 мл = 2450 мл для 30 річного чоловіка). Зміна пароля через updatePassword викликала алерт про успіх (Рис.3.9–3.10).

Фіксація напоїв: На AddDrinkScreen перевірено додавання, редагування та видалення напоїв. Валідація об’єму (0 мл відхиляється) та слухач listenToWaterIntake забезпечували оновлення даних у Firestore (Рис.3.11).

Візуалізація та статистика: Кругова діаграма відображали прогрес і статистику в реальному часі через react-native-progress. Календар (react-native-modal-datetime-picker) коректно фільтрував дані (Рис.3.12).

Навігація та інтерфейс: Переходи між екранами через react-navigation були швидкими. Ширина полів, розміри кнопок і списки компонентів відповідали дизайну, вміщалися на екранах обох пристроїв без перекриттів. Використання react-native-keyboard-aware-scroll-view забезпечувало коректне відображення форм.

Налаштування та вихід: Кнопка налаштувань (в HomeScreen) дозволяла вихід (signOut) та видалення акаунта (deleteUserData). Вихід очищав AsyncStorage та повертав на AuthScreen. Видалення рекурсивно очищувало документи в Firestore та перенаправляло на AuthScreen (Рис.3.13–3.14).

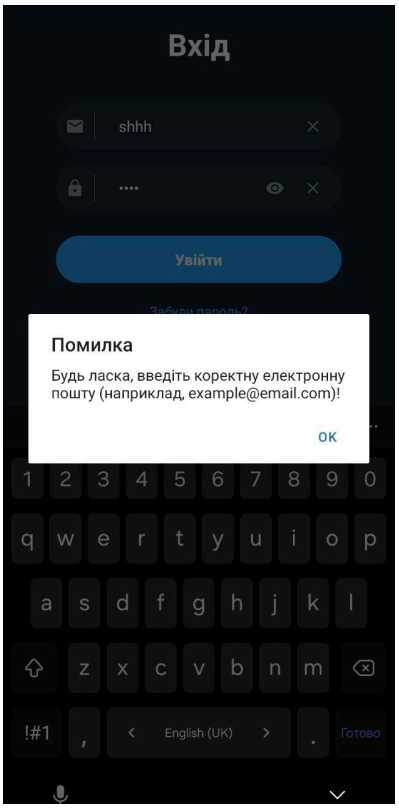

Рисунок 3.7. Перевірка поля електронної пошти

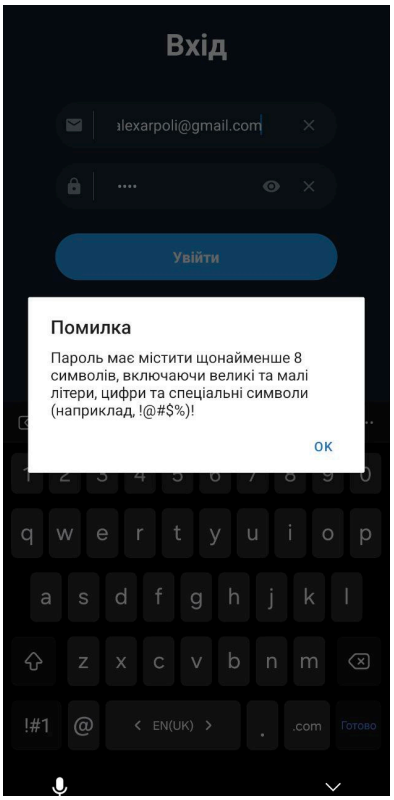

Рисунок 3.8. Перевірка поля паролю

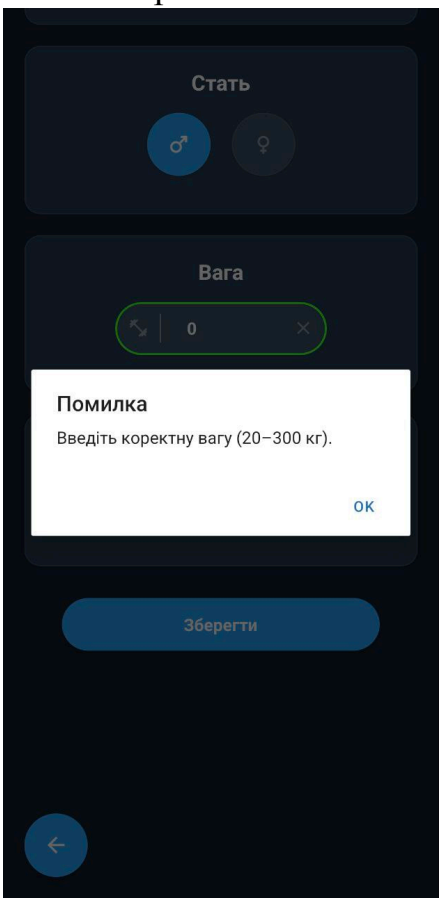

Рисунок 3.9. Введення некоректної ваги в поле

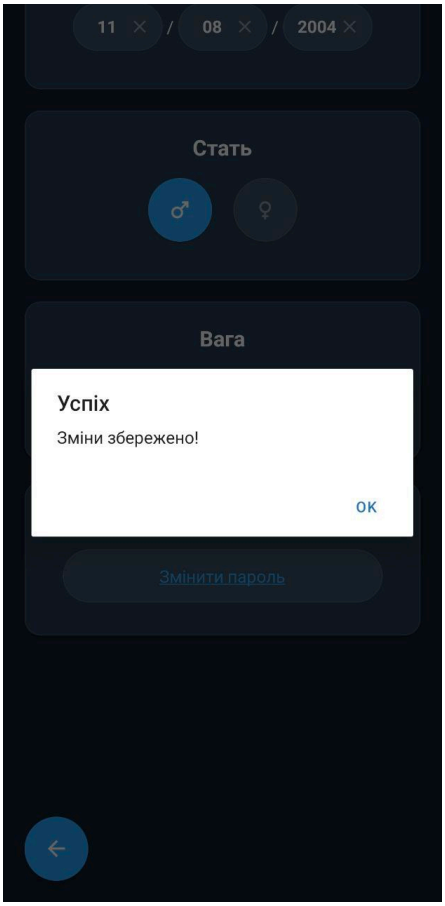

Рисунок 3.10. Повідомлення про успішне збереження даних (паролю)

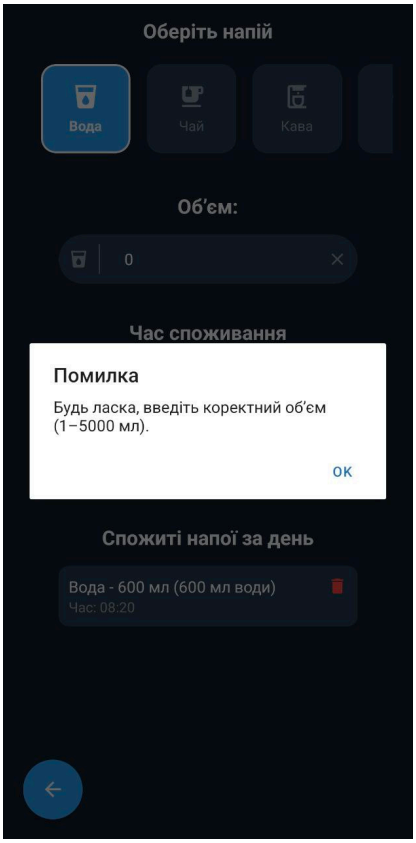

Рисунок 3.11. Введення некоректного об’єму напою

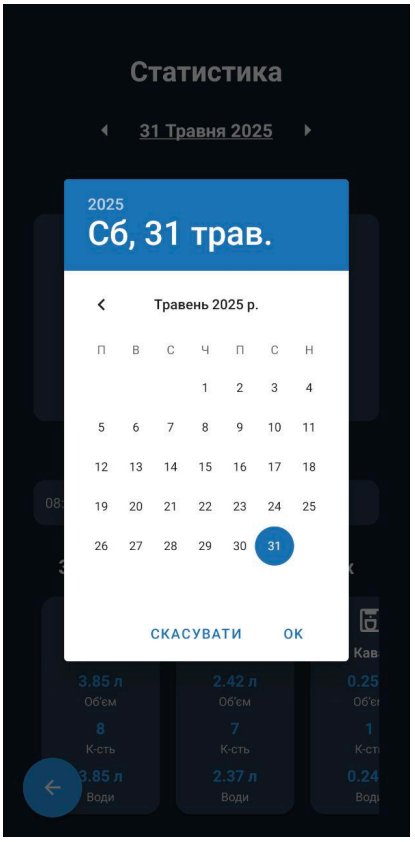

Рисунок 3.12. Модальне вікно календаря

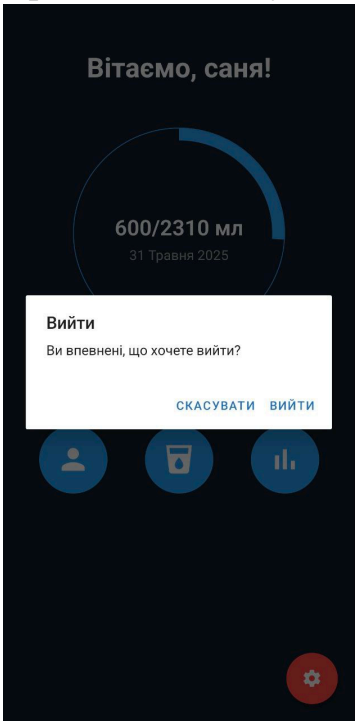

Рисунок 3.13. Попередження про вихід з акаунту

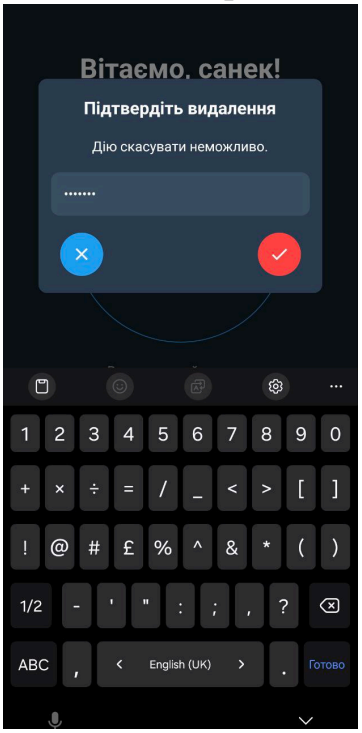

Рисунок 3.14. Модальне вікно видалення акаунта

Загалом протестовано всі поля введення, функціонал кнопок та компонентів застосунку, і кожен елемент демонстрував відповідну коректну реакцію. Усі функції, від автентифікації до відображення статистики, працювали чітко та стабільно. Інтуїтивний інтерфейс із зрозумілими сповіщеннями (наприклад, алерти про помилки чи успішні дії) допомагав користувачу легко орієнтуватися в застосунку, забезпечуючи комфортне використання навіть у разі некоректних дій.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено кросплатформний мобільний застосунок HydrationApp для моніторингу водного балансу користувача, який відповідає поставленій меті та завданням. Проведений аналіз предметної області підтвердив актуальність проблеми недостатнього споживання води та необхідність створення зручного інструменту для її вирішення. Дослідження потреб користувачів та існуючих рішень (WaterMinder, Hydro Coach) дозволило сформулювати функціональні та нефункціональні вимоги для розроблюваного застосунку.

Для реалізації застосунку обрано сучасні технології: React Native для кросплатформної розробки, Firebase Authentication і Firestore для безпечної автентифікації та хмарного зберігання даних, Expo Go для спрощення тестування, а також різні бібліотеки для забезпечення зручної навігації, коректного відображення форм і візуальної привабливості. Вибір інструментів обґрунтовано їхньою ефективністю, простотою інтеграції та відповідністю вимогам щодо продуктивності, безпеки та кросплатформності.

Архітектура застосунку спроєктована з урахуванням модульності та масштабованості, включаючи клієнтську частину з п’ятьма основними екранами (AuthScreen, HomeScreen, AddDrinkScreen, ProfileScreen, StatisticScreen) та серверну частину на базі Firebase. Реалізовано функціонал автентифікації, створення та редагування профілю, фіксації спожитих напоїв, відображення прогресу через кругову діаграму та статистики за обраний день. Використання слухачів onSnapshot забезпечує оновлення даних у реальному часі, а Firestore rules гарантують захист персональних даних. Локальне кешування через AsyncStorage оптимізує продуктивність, дозволяючи завершувати сесії після 24 годин.

Тестування застосунку на платформах Android (Samsung Galaxy A35, Android 14) та iOS (iPhone 11 Pro, iOS 17) через Expo Go підтвердило його стабільність і відповідність вимогам. Перевірено всі функції: автентифікацію, валідацію даних (email, пароль, вага, об’єм напоїв), навігацію, збереження та

відображення даних. Усі поля введення, кнопки та компоненти реагували коректно, а інтуїтивний інтерфейс із чіткими сповіщеннями забезпечував зручність навіть при некоректних діях користувача.

Практична цінність HydrationApp полягає у створенні доступного інструменту для моніторингу водного балансу, який дозволяє фіксувати спожиті напої, відстежувати прогрес у реальному часі та аналізувати статистику без платних обмежень чи реклами, на відміну від аналогів. Архітектура застосунку підтримує можливість розширення, наприклад, додавання функцій гейміфікації чи кастомізації напоїв. Робота підтверджує ефективність обраних технологій і методів, включаючи аналіз, проєктування, програмну інженерію та тестування, для створення зручного та надійного мобільного рішення для підтримки здоров’я користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. WaterMinder – Track Your Water Intake [Електронний ресурс]. – 2025. – Режим доступу: https://waterminder.com/.
2. Hydro Coach – Drink Water [Електронний ресурс]. – 2025. – Режим доступу: https://hydrocoach.com/.
3. React Native | A framework for building native apps using React [Електронний ресурс]. – 2025. – Режим доступу: https://reactnative.dev/.
4. Firebase | Google's Mobile and Web App Development Platform [Електронний ресурс]. – 2025. – Режим доступу: https://firebase.google.com/.
5. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.
6. Expo | Build, deploy, and iterate on apps faster [Електронний ресурс]. – 2025. – Режим доступу: https://expo.dev/.
7. Visual Studio Code – Code Editing. [Електронний ресурс]. – 2025. – Режим доступу: https://code.visualstudio.com/.
8. React Native AsyncStorage [Електронний ресурс]. – 2025. – Режим доступу: https://react-native-async-storage.github.io/async-storage/.
9. React Navigation | Routing and navigation for your React Native apps [Електронний ресурс]. – 2025. – Режим доступу: https://reactnavigation.org/.
10. React Native Keyboard Aware Scroll View [Електронний ресурс]. – 2025. – Режим доступу: https://github.com/APSL/react-native-keyboard-aware-scroll-view.
11. Expo Vector Icons [Електронний ресурс]. – 2025. – Режим доступу: https://icons.expo.fyi/.
12. Firebase. Firestore Security Rules [Електронний ресурс]. – 2025. – Режим доступу: https://firebase.google.com/docs/firestore/security/get-started.
13. React Native Progress – Progress indicators and spinners for React Native [Електронний ресурс]. – 2025. – Режим доступу: https://github.com/oblador/react-native-progress.
14. Expo Image Picker – A library to pick images from the device [Електронний ресурс]. – 2025. – Режим доступу: https://docs.expo.dev/versions/latest/sdk/imagepicker/.
15. ІВ Машкіна, ВЙ Машкін, НІ Аралова, ОМ Ключко Mathematical models of respiratory and blood circulatory system … - Biotechnologia Acta, 2022
16. Яскевич, Владислав Олександрович (2023) *JavaScriptбібліотека React Навчальний посібник для студентів спеціальності Комп'ютерні науки* Київський університет імені Бориса Грінченка, Україна, Київ. https://elibrary.kubg.edu.ua/id/eprint/48587/

17. React Native Modal Datetime Picker [Електронний ресурс]. – 2025. – Режим доступу: https://github.com/mmazzarolo/react-native-modal-datetime-picker.
18. І.В. Машкіна, В.О. Абрамов, Т.І. Носенко, Є.В. Іваніченко. Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.
19. Draw.io – Free Online Diagram Editor [Електронний ресурс]. – 2025. – Режим доступу: https://www.draw.io/.