ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)
«_____»___202_ р.

КВАЛІФІКАЦІЙНА РОБОТА
на тему

СИСТЕМА ОБЛІКУ ТА НАГАДУВАНЬ ДЛЯ СПОЖИВАННЯ ВОДИ ПРОТЯГОМ ДНЯ З ПРОСТИМИ ГРАФІЧНИМИ ЗВІТАМИ

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістра

Виконавець роботи Барабаш Руслан Анатолійович
_____«____» _____202_ р.
(підпис)

Науковий керівник доцент, к.ф.-м.н. Черненко О. О.
_____«____»_____202_ р.
(підпис)

Рецензент

ПОЛТАВА 2025

РЕФЕРАТ

Записка: 57 с., 14 рис., 2 таблиці, 1 додаток, 12 джерел.

ГІДРАТАЦІЯ, МОБІЛЬНИЙ ДОДАТОК, REACT NATIVE

Об’єктом розробки є процес формування та підтримання здорової поведінки щодо споживання води.

Предметом розробки є методи обліку щоденного споживання рідини та алгоритми персоналізації рекомендацій у мобільних застосунках.

Метою роботи є розробка програмного забезпечення, яке дозволяє автоматизувати процес контролю гідратації, забезпечуючи користувача персоналізованими рекомендаціями та наочними графічними звітами.

Результатом роботи став мобільний застосунок HydrationApp, реалізований технологією React Native, який включає модулі: онбординг користувача (вибір статі, віку, ваги, зросту, рівня активності та клімату), розрахунок добової норми, облік спожитих напоїв, система push-нагадувань, графічні звіти та профіль користувача. Застосунок підтримує різні типи напоїв, зміну розміру чашки, історію записів та автоматичне оновлення прогресу протягом дня.

Однією з ключових функцій є динамічний розрахунок добової норми споживання води, що враховує фізіологічні параметри користувача та зовнішні умови. Зібрані протягом дня дані формуються у структури, які використовуються для побудови щоденних та тижневих графіків, відображення відсотку виконання норми, аналізу звичок та формування рекомендацій.

У застосунку реалізовано візуалізацію аналітичних даних: гістограми, стовпчикові діаграми, відсоткові індикатори прогресу. Для нагадувань використовується механізм локальних push-сповіщень, які налаштовуються індивідуально залежно від графіка сну та активності користувача.

Для перевірки коректності роботи алгоритмів було проведено тестування серверної логіки, реалізованої на TypeScript. Тести охоплювали модулі: HydrationService, DailyIntakeService та ReportsService. Перевірено правильність розрахунку добової норми, оновлення фактичного об’єму споживання води,

визначення статусу дня та формування узгоджених тижневих даних для графічних звітів. Тестування засвідчило стабільність роботи алгоритмів та правильну взаємодію між модулями.

Розроблений застосунок був протестований на реальних сценаріях використання. Застосунок показав стабільну роботу, правильну фіксацію даних, коректне формування графіків та зручність інтерфейсу для кінцевого користувача.

Застосунок може бути використаний як інструмент для формування здорових звичок, підвищення добробуту користувачів та моніторингу рівня гідратації у режимі реального часу.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕННЬ, СИМВОЛІВ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
мл (ml)	Мілілітр — одиниця вимірювання об’єму рідини.
L (літр)	Одиниця вимірювання об’єму для агрегованих звітів.
Добова норма споживання води	Розрахована кількість води, яку користувач має випити протягом дня.
Гідратація	Процес забезпечення організму необхідною кількістю води.
Щоденний прогрес	Відсоток виконання добової норми споживання води.
Activity level (рівень активності)	Показник фізичного навантаження (low, medium, high), що впливає на норму води.
Climate factor (кліматичний коефіцієнт)	Коригувальний параметр норми води залежно від кліматичних умов (cold, moderate, hot).
DailyIntake	Структура даних, що містить щоденні записи споживання води.
Report / Weekly report	Звіт, побудований на основі зібраних даних (щоденний, тижневий, місячний).
Push-сповіщення (notifications)	Оповіщення мобільного застосунку про необхідність випити воду.

ВСТУП

У сучасному динамічному ритмі життя питання підтримання здорових звичок набуває особливої актуальності. Однією з таких звичок є достатнє споживання води протягом дня, що безпосередньо впливає на роботу всіх систем організму людини. Наукові дослідження підтверджують, що навіть легка дегідратація знижує концентрацію, продуктивність, погіршує самопочуття та фізичні показники. Попри це, більшість людей не відстежує власне споживання води, не маючи зручних інструментів контролю та нагадувань.

Разом із розвитком мобільних технологій з'явилася можливість створювати інтелектуальні засоби підтримки здорової поведінки. Мобільні wellness-застосунки посідають важливе місце у формуванні корисних звичок, оскільки поєднують доступність, персоналізацію, інтерактивність та здатність до аналізу індивідуальних даних. Проте більшість доступних рішень або перевантажені функціоналом, або не мають інструментів персоналізованого розрахунку добової норми та зручних графічних звітів, що ускладнює їх використання широким колом користувачів.

У зв'язку з цим актуальною є розробка мобільного застосунку, що забезпечує простий, інтуїтивний та науково обґрунтований облік щоденного споживання води, формування індивідуальних рекомендацій та графічне представлення динаміки гідратації.

Дана робота присвячена створенню системи обліку та нагадувань для споживання води протягом дня з простими графічними звітами, що реалізована у вигляді мобільного застосунку. Програма містить модуль первинного налаштування профілю, алгоритм розрахунку щоденної норми на основі віку, статі, маси тіла, фізичної активності та кліматичних умов, систему нагадувань, інтерфейс для обліку напоїв, а також модулі візуалізації статистики.

Об'єкт дослідження — процес формування та підтримання здорової поведінки щодо споживання води.

Предмет дослідження — методи обліку щоденного споживання рідини та алгоритми персоналізації рекомендацій у мобільних застосунках.

Метою роботи є розробка програмного забезпечення, яке дозволяє автоматизувати процес контролю гідратації, забезпечуючи користувача персоналізованими рекомендаціями та наочними графічними звітами.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати наукові та практичні аспекти гідратації людини;
- дослідити підходи до розробки wellness-застосунків та візуалізації даних;
- розробити математичну модель розрахунку добової норми споживання води;
- реалізувати алгоритми врахування фізичної активності, клімату та індивідуальних параметрів користувача;
- створити архітектуру програмної системи та модель даних;
- реалізувати мобільний застосунок із модулями обліку, нагадувань та звітності;
- провести тестування роботи системи та оцінити її ефективність.

Практичне значення роботи полягає у створенні інструмента, який може бути використаний широким колом користувачів для формування корисної звички регулярного споживання води, що сприяє покращенню здоров’я та якості життя.

1. ПОСТАНОВКА ЗАДАЧІ

Проблема недостатнього щоденного споживання води є поширеною серед різних вікових груп населення. Більшість людей не контролює власну гідратацію та не має інструментів, які допомагали б відстежувати споживання рідини протягом дня. Наявні мобільні застосунки для моніторингу водного балансу часто є перевантаженими функціональністю, не забезпечують персоналізований підхід або не мають достатньо наочних засобів візуалізації даних. Це обмежує їх ефективність та не сприяє формуванню стійкої корисної звички.

Удосконалення процесу контролю гідратації можливе шляхом розробки мобільного застосунку, що поєднує простоту використання, персоналізацію рекомендацій і зручні інструменти аналізу — графічні звіти, інтерактивні елементи та нагадування. Такий підхід дозволяє користувачеві усвідомлювати власні звички, відстежувати прогрес і формувати індивідуальний режим споживання води.

Метою дипломної роботи є створення програмної системи, яка забезпечує облік споживання води, автоматичне формування нагадувань та наочне представлення статистики у вигляді графічних звітів. Для реалізації цієї мети необхідно вирішити низку задач, що визначають функціональні та технічні вимоги до розробленої системи.

До основних завдань роботи належать:

1. Аналіз наукових джерел щодо фізіологічної ролі води та сучасних норм споживання рідини.
2. Аналіз існуючих мобільних wellness-рішень та їх функціональних можливостей.
3. Розробка математичної моделі визначення щоденної індивідуальної норми споживання води з урахуванням статі, віку, маси тіла, рівня активності та клімату.
4. Проєктування архітектури мобільного застосунку та моделі зберігання даних.
5. Реалізація модуля первинного налаштування профілю користувача.
6. Реалізація модуля обліку спожитих напоїв з можливістю вибору типу напою та об’єму.

7. Створення системи нагадувань із використанням механізмів мобільних сповіщень.
8. Розробка модулів відображення статистичних даних у вигляді щоденних, тижневих і місячних графічних звітів.
9. Проведення тестування роботи застосунку та оцінка його функціональної коректності.

У результаті виконання роботи має бути створено зручний, інтуїтивний та ефективний інструмент для моніторингу гідратації, який сприяє покращенню здоров’я користувача та формуванню позитивних щоденних звичок.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Роль води в організмі людини та норми щоденного споживання

Вода є ключовим компонентом людського організму та бере участь у всіх життєво важливих процесах. У середньому вода становить від 55 до 70 % маси тіла людини залежно від віку, статі та фізіологічного стану. Вона виконує транспортну, терморегуляційну, структурну та метаболічну функції, забезпечуючи стабільність внутрішнього середовища організму. Вода бере участь у процесах травлення, виведення токсинів, перенесення поживних речовин і кисню, регулюванні температури тіла, а також підтримує здоровий стан шкіри, суглобів і серцево-судинної системи.

Регулярне оновлення водного балансу є необхідною умовою для нормального функціонування організму. Добова втрата рідини через дихання, потовиділення, виділення та метаболічні процеси становить у середньому від 1,5 до 2,5 літра. Ці втрати повинні компенсуватися шляхом споживання води та інших рідин. Навіть помірна дегідратація — втрата 1–2 % маси тіла у вигляді рідини — може призвести до відчуття втоми, зниження концентрації, головного болю та зменшення фізичної витривалості.

Водночас надмірне вживання води також може мати негативні наслідки, зокрема порушення електролітного балансу, тому важливо дотримуватися індивідуально рекомендованих норм. Норми добового споживання води залежать від багатьох факторів, таких як вік, маса тіла, стать, рівень фізичної активності, кліматичні умови та стан здоров’я.

У загальних рекомендаціях Всесвітньої організації охорони здоров’я зазначається, що середня добова потреба у воді для дорослих становить від 2 до 3 літрів рідини на добу, включно з водою, напоями та їжею, що містить воду. У більшості країн застосовується спрощений підхід: приблизно 30 мілілітрів води на 1 кілограм маси тіла, що дозволяє швидко оцінити індивідуальну норму споживання. Для людей із високим рівнем фізичної активності або тих, хто проживає у спекотних кліматичних умовах, потреба у воді може бути на 0,5–1 літр вищою.

Додатковим важливим чинником є різниця у водному балансі між чоловіками та жінками. Через більшу частку м’язової тканини чоловіки зазвичай мають більшу потребу у воді, тоді як у жінок рекомендований обсяг у середньому на 10–15 % нижчий. Вікові зміни також впливають на потребу у рідині: у молодому віці метаболічні процеси проходять активніше, що підвищує потребу у воді, тоді як у людей старшого віку відчуття спраги зменшується, що вимагає додаткового контролю споживання.

Таким чином, достатнє та регулярне споживання рідини є необхідним для підтримання фізичного та когнітивного здоров’я. В умовах сучасного способу життя, коли люди часто ігнорують сигнали спраги або не мають змоги відстежувати власний водний баланс, особливого значення набувають технологічні засоби автоматизованого контролю споживання води. Ці засоби, зокрема мобільні застосунки, допомагають формувати здорові щоденні звички, забезпечують персоналізовані рекомендації та сприяють підвищенню рівня обізнаності користувачів щодо власного стану гідратації.

2.2. Підходи до проєктування мобільних wellness-застосунків

Мобільні wellness-застосунки посідають важливе місце в сучасній цифровій екосистемі, оскільки вони підтримують формування корисних звичок, сприяють покращенню фізичного та психічного здоров’я користувачів і забезпечують персоналізований підхід до щоденних рутин. Додатки, що допомагають контролювати рівень активності, сон, харчування чи споживання води, мають відповідати одночасно як технічним, так і психологічним вимогам, оскільки їхня ефективність безпосередньо залежить від здатності впливати на поведінку користувача та підтримувати його мотивацію в довгостроковій перспективі. [1-3]

Проєктування wellness-застосунків ґрунтується на поєднанні принципів UX-дизайну, поведінкової психології, доступності та технічної оптимізації. Одним із ключових аспектів є простота інтерфейсу. Користувач повинен отримати можливість швидко виконувати основні дії: додавати записи, переглядати прогрес,

отримувати нагадування. Для цього застосовуються мінімалістичні інтерфейсні рішення, структурована навігація та зрозуміла система візуальних акцентів. Дослідження ринку wellness-додатків показують, що надмірна кількість кроків або перевантажений дизайн знижують регулярність використання застосунку.

Другим важливим підходом є персоналізація. Оскільки потреби користувачів суттєво різняться залежно від віку, активності, стану здоров'я чи способу життя, сучасні застосунки впроваджують алгоритми адаптивних рекомендацій. Зокрема, у додатках для контролю споживання води персоналізація може включати: розрахунок щоденної норми відповідно до маси тіла, кліматичних умов і рівня фізичної активності, коригування частоти нагадувань, різні типи візуальних звітів. Застосування персональних параметрів підвищує точність рекомендацій і рівень довіри користувача до системи. [4]

Наступним ключовим аспектом є використання елементів поведінкової аналітики. Добре спроєктований wellness-застосунок не лише фіксує дії користувача, а й допомагає формувати та закріплювати корисні звички. Для цього застосовуються механізми м'якої мотивації: позитивні підказки, візуальний прогрес, щоденні цілі, досягнення та ігрові елементи. Наприклад, поступове заповнення графічного індикатора водного балансу допомагає користувачеві інтуїтивно відстежувати прогрес і підсилює бажання завершити денну норму.

Додаткове значення має забезпечення доступності та інклюзивності. Wellness-застосунки повинні враховувати потреби користувачів різного віку, у тому числі тих, хто має обмеження зору, слуху чи моторики. Тому рекомендації WAI-ARIA, відповідний контраст кольорів, коректний масштаб тексту та підтримка адаптивної верстки є обов'язковими елементами якісного дизайну.

Технічний аспект проєктування включає оптимізацію продуктивності, стійкість до помилок та забезпечення офлайн-режиму. Оскільки користувачі можуть взаємодіяти з додатком у будь-яких умовах, важливо гарантувати швидке завантаження інтерфейсу, коректне збереження історії навіть без доступу до мережі, а також мінімальне споживання ресурсів пристрою. У wellness-застосунках, де взаємодія відбувається часто та короткими сесіями, ці фактори є критичними для

підтримки регулярності використання.

Не менш важливим є аспект конфіденційності. Оскільки wellness-додатки обробляють персональні й іноді медичні дані, необхідно забезпечити дотримання принципів приватності: локальне зберігання історії, шифрування чутливої інформації, прозора політика доступу до даних. Наявність таких механізмів підвищує довіру та зменшує ризики витоку персональної інформації.

Таким чином, проєктування мобільних wellness-застосунків потребує комплексного підходу, який поєднує технічні рішення з психологічними та дизайнерськими принципами. Ефективний застосунок має бути простим, адаптивним, інформативним та орієнтованим на довгострокову взаємодію. Саме такі підходи стали основою для розробки системи обліку та нагадувань щодо щоденного споживання води, описаної у цій дипломній роботі.

2.3. Методи візуалізації даних для простих графічних звітів

Ефективна візуалізація даних є ключовим елементом wellness-застосунків, оскільки саме графічні індикатори дозволяють користувачеві швидко оцінювати свій прогрес, аналізувати тенденції та приймати рішення щодо зміни поведінки. У системах контролю споживання води візуальні методи мають бути простими, наочними та зрозумілими навіть під час коротких сесій взаємодії зі смартфоном. Основне завдання таких інтерфейсів — перетворити числові показники на інформативні та легко сприйнятні графічні форми.

На рисунку 2.1 подано приклади найпоширеніших типів графіків, що застосовуються в мобільних wellness-додатках. Ці методи охоплюють широкий спектр завдань: від відображення короткострокового прогресу до аналізу довготривалих змін у поведінці користувачів (див. рис 2.1).

Рисунок 2.1 — Приклади основних методів візуалізації даних у мобільних застосунках

Серед найважливіших видів візуалізації, що використовуються у системах контролю водного балансу, можна виділити такі:

1. Індикатори заповнення (progress indicators).

Кільцеві, вертикальні чи горизонтальні індикатори дозволяють миттєво оцінити відсоток виконання щоденної норми. Подібні елементи забезпечують високий рівень наочності та виступають мотиваційним фактором, адже стимулюють користувача заповнити «порожню частину» індикатора. У wellness-додатках цей тип графіки є одним з найефективніших для щоденних цілей. [5]

2. Стовпчикові діаграми (bar charts).

Цей тип графіків відображає споживання води за днями тижня або за місяць. На рисунку 2.1 подано кілька варіантів стовпчикових діаграм — вертикальні, горизонтальні та комбіновані. Вони добре підходять для аналізу звичок користувача у середньо- та довготривалій перспективі, а також для порівняння обсягів споживання в різні дні.

3. Лінійні графіки (line charts).

Використовуються для відстеження тенденцій. Лінійна діаграма дозволяє виявити, чи підвищується або знижується рівень гідратації протягом місяця, чи є стабільність у досягненні щоденної норми. Це особливо важливо для користувачів, які намагаються побудувати довгострокову корисну звичку.

4. Площинні (area) графіки.

Вони дозволяють відобразити зміну обсягів у часі та водночас підкреслити різницю між днями або періодами. У wellness-додатках area charts застосовуються рідше, проте можуть бути корисними для місячної та сезонної статистики.

5. Кругові діаграми (pie та donut charts).

У контексті контролю водного балансу такі діаграми можуть використовуватися для відображення структури споживаних напоїв: вода, чай, кава, соки тощо. Формат donut-chart дозволяє поєднати відсоткове співвідношення та абсолютні значення. [6]

6. Radar- та pyramid-діаграми.

Такі методи більш характерні для аналітичних wellness-платформ, проте можуть застосовуватися для комплексного аналізу: наприклад, для оцінки впливу гідратації на активність, сон чи фізичне навантаження. На рисунку ці типи подано як приклади нетипових, але потенційно корисних методів візуалізації.

Вибір методу візуалізації залежить від цілей користувача та типу даних. У додатках для контролю щоденного споживання води перевага надається простим та інтуїтивним форматам — індикаторам заповнення та стовпчиковим діаграмам, тоді як для аналізу тенденцій застосовуються лінійні графіки. Використання різних методів візуалізації дозволяє зробити інформацію не лише зрозумілою, але й мотиваційною, що сприяє формуванню здорових звичок.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Математична модель розрахунку щоденної норми споживання води

Розрахунок щоденної норми споживання води є ключовим елементом роботи мобільних wellness-застосунків, оскільки саме цей показник визначає персоналізовані цілі користувача та впливає на рекомендації щодо гідратації протягом дня. Математична модель повинна враховувати індивідуальні фізіологічні параметри, поведінкові чинники та умови навколишнього середовища, щоб забезпечити точність і практичну корисність результатів. [7]

У загальному вигляді добова норма води формується з урахуванням базового метаболічного запиту організму та додаткових навантажень, пов’язаних із віком, статтю, активністю та кліматичними умовами. У розробленій системі використано комбінований підхід, який узагальнює положення сучасних рекомендацій дієтології та спортивної медицини.

Базове рівняння

Основою моделі є розрахунок добової норми залежно від маси тіла користувача:

$$\mathrm{Wbase} = \mathrm{m} \times 30$$

де

Wbase — базова кількість води в мілілітрах,

m — маса тіла користувача у кілограмах,

30 — середня норма споживання води (мл) на кожен кілограм маси тіла, запропонована більшістю медичних джерел.

Корекція за статтю

Фізіологічні відмінності чоловіків та жінок впливають на кількість рідини, необхідної для підтримання водного балансу. У моделі застосовано такі коефіцієнти:

- для чоловіків:

$$\mathrm{Wgender} = \mathrm{Wbase} \times 1.10,$$

- для жінок:

$$\text{Wgender} = \text{Wbase}.$$

Підвищувальний коефіцієнт для чоловіків (+10%) враховує більший відсоток м’язової тканини та вищу інтенсивність метаболічних процесів.

Корекція за віком

Із віком потреба у воді зменшується через зниження інтенсивності обміну речовин. Запроваджено такі коефіцієнти:

- до 30 років:

$$\text{Wage} = \text{Wgender} \times 1.05,$$

- 30–50 років:

$$\text{Wage} = \text{Wgender},$$

- після 50 років:

$$\text{Wage} = \text{Wgender} \times 0.95.$$

Корекція за фізичною активністю

Рівень активності суттєво впливає на втрати рідини, тому до моделі включено фіксовані корекції:

$$\text{W}activity = \begin{cases} Wage, & \text{сидячий режим}, \\ Wage + 200, & \text{легка активність} \\ Wage + 400, & \text{помірна активність} \\ Wage + 600, & \text{висока активність} \end{cases}$$

Ці значення узгоджуються з рекомендаціями спортивної медицини щодо додаткового споживання води під час тренувань. [8]

Корекція за кліматом

Температура довкілля також впливає на рівень втрати рідини через потовиділення. Для цього передбачено:

$$\text{W}climate = \begin{cases} Wactivity, & \text{холодний клімат}, \\ Wactivity + 200, & \text{помірний клімат} \\ Wactivity + 400, & \text{жаркий клімат} \end{cases}$$

Підсумкове рівняння

Фінальна добова норма визначається як:

$$\text{Wfinal} = \text{Wclimate},$$

де всі попередні етапи застосовуються послідовно відповідно до введених користувачем параметрів.

Таким чином, математична модель поєднує зважений набір коригувальних факторів, які впливають на водний баланс людини. Це дозволяє отримати персоналізоване значення добової норми, що відповідає реальним потребам організму. Модель є достатньо простою для обчислення на мобільному пристрої, але водночас охоплює основні чинники, які визначають потребу в гідратації.

3.2. Алгоритми врахування фізичної активності, клімату та маси тіла

Персоналізація добової норми споживання води у мобільному застосунку ґрунтується на поетапній обробці даних, які користувач вводить під час первинного налаштування профілю. Серед цих параметрів ключову роль відіграють маса тіла, рівень фізичної активності та кліматичні умови проживання. Для кожної з цих величин розроблено окремі алгоритми коригування, які забезпечують точність розрахунку індивідуальної норми гідратації. [9]

Алгоритми реалізовано у вигляді послідовних функціональних блоків, де кожний блок обробляє конкретний тип даних, після чого передає результат наступному етапу. Така структура дозволяє розширювати або змінювати модель без порушення логіки застосунку.

Алгоритм врахування маси тіла

Маса тіла є базовим параметром, від якого залежить початкове значення добової норми води. Розрахунок виконується за формулою:

$$W = m \times 30,$$

де

m — маса користувача (кг),

30 — середня кількість мілілітрів води на кілограм маси тіла, рекомендована дієтологами.

Алгоритм:

1. Користувач вводить свою масу тіла у відповідному полі профілю.
2. Система перевіряє коректність значення.

3. На основі маси тіла обчислюється базовий рівень гідратації.
4. Результат передається до модуля обробки статі та віку, а потім — до модуля активності та клімату.

Таким чином, маса тіла визначає вихідну точку у формуванні персоналізованої норми.

Алгоритм врахування рівня фізичної активності

Рівень фізичної активності впливає на інтенсивність обміну речовин і втрати води з організму. У застосунку передбачено чотири категорії активності, що відповідають елементам інтерфейсу, які користувач вибирає під час реєстрації:

- Сидячий режим
- Легка активність
- Помірна активність
- Висока активність

Кожній з категорій відповідає фіксоване значення корекції добової норми:

$$W activity = \begin{cases} 0, & \text{сидячий режим,} \\ 200, & \text{легка активність} \\ 400, & \text{помірна активність} \\ 600, & \text{висока активність} \end{cases}$$

Алгоритм:

1. Користувач обирає свій рівень активності (екран «Який ваш рівень активності?»).
2. Після вибору система встановлює відповідний коригувальний коефіцієнт.
3. Корекція додається до базового значення, сформованого на етапі розрахунку маси, статі та віку.
4. Результат передається до наступного модуля — кліматичних умов.

Такий механізм дозволяє врахувати підвищені втрати рідини під час рухової активності. [10]

Алгоритм врахування кліматичних умов

Клімат та температура навколишнього середовища також істотно впливають на рівень споживання води. У застосунку передбачено три варіанти:

- Холодний клімат

- Помірний клімат
- Спекотний клімат

Залежно від вибору користувача використовується одна з таких корекцій:

$$\mathrm{W}climate = \begin{cases} 0, & \text{холодний клімат,} \\ 200, & \text{помірний клімат} \\ 400, & \text{жаркий клімат} \end{cases}$$

Алгоритм:

1. Користувач обирає клімат своєї місцевості (екран «Який клімат/погода у вашій місцевості?»).
2. Система встановлює відповідне значення корекції.
3. Корекція додається до значення, отриманого після врахування активності.
4. На основі отриманого результату формується остаточна «щоденна мета» користувача (екран із результатом, наприклад 3505 ml).

Таким чином, кліматичний блок завершує процес персоналізації.

Інтегрований алгоритм обчислення норми

У загальному вигляді всі три алгоритми працюють у єдиному ланцюжку:

1. Обчислити базову норму з урахуванням маси тіла.
2. Застосувати корекцію за статтю та віком.
3. Застосувати корекцію за фізичною активністю.
4. Застосувати кліматичну корекцію.
5. Показати користувачу остаточний результат та надати можливість його відредагувати.

Фінальна формула:

$$\mathrm{Wfinal} = \mathrm{Wbase} + \mathrm{Kgender} + \mathrm{Kage} + \mathrm{Kactivity} + \mathrm{Kclimate}$$

Усі коефіцієнти застосовуються послідовно, що забезпечує гнучкість та можливість подальшого розширення моделі (наприклад, додавання параметра «вагітність», «індивідуальні медичні рекомендації» тощо).

3.3. Алгоритми формування нагадувань протягом дня

Нагадування про необхідність вживання води є ключовим інструментом підтримання стабільного рівня гідратації протягом дня. Мобільні wellness-застосунки використовують різні механізми формування таких сповіщень, але найбільш ефективними вважаються алгоритми, що враховують персональні параметри користувача та динаміку його денного розкладу.

У розробленій системі реалізовано інтелектуальний модуль нагадувань, який забезпечує оптимальний інтервал між сповіщеннями, адаптується до темпу споживання води та не створює надмірного навантаження на користувача.

Загальні принципи роботи модуля нагадувань

Алгоритм ґрунтується на трьох ключових засадах:

1. Регулярність: користувач має отримувати нагадування з рівномірними інтервалами, що запобігає тривалим перервам між прийомами води.
2. Персоналізація: частота сповіщень повинна відповідати добовій нормі води, яку система розрахувала з урахуванням фізіологічних і середовищних чинників.
3. Адаптивність: якщо користувач п’є воду частіше або рідше за середній темп, інтервали між нагадуваннями коригуються автоматично.

Алгоритм визначення базового інтервалу нагадувань

Після встановлення щоденної норми споживання води система формує рекомендовану кількість прийомів рідини протягом дня. Для цього використовується наступний підхід:

1. Задається тривалість активної частини дня T (типово — 14 годин).
2. Система розраховує кількість порцій води, орієнтуючись на середній обсяг однієї порції P=250 мл.
3. Кількість прийомів води визначається як

$$n = \left[\frac{Wfinal}{P}\right]$$

4. Базовий інтервал між нагадуваннями обчислюється за формулою:

$$\mathrm{Ibase} = \frac{\mathrm{T}}{\mathrm{n}}$$

Таким чином, чим вища добова норма води, тим частіше надходитимуть нагадування.

Алгоритм планування сповіщень

Після визначення базового інтервалу час сповіщень формується за схемою:

$$\mathrm{ti} = \mathrm{tstart} + \mathrm{i} * \mathrm{Ibase}$$

де

tstart — час початку активного дня (за замовчуванням 08:00),

i — порядковий номер нагадування.

Система генерує масив сповіщень і передає його локальному планувальнику смартфона (у React Native використовується механізм push-нотифікацій).

Адаптивний алгоритм коригування нагадувань

Щоб уникнути ситуацій, коли користувач ігнорує сповіщення або навпаки — випереджає рекомендований графік, використовується адаптивна модель:

1. Якщо користувач випив воду раніше за план
   - Наступне нагадування зсувається на більший інтервал.
   - Обчислення:

$$\mathrm{Inew} = \mathrm{Ibase} + \Delta$$

де Δ — коефіцієнт зміщення (від 10 до 20 хвилин).

2. Якщо користувач пропустив нагадування
   - Система зменшує інтервал до наступного сповіщення:

$$\mathrm{Inew} = \mathrm{Ibase} - \Delta$$

3. Якщо мета майже досягнута
   - Нагадування надсилаються рідше або повністю припиняються.

Цей механізм дозволяє зробити процес природним та ненав’язливим.

Алгоритм урахування часу останнього прийому води

Система зберігає часові мітки усіх прийомів води в історії. Якщо виявлено, що з моменту останнього прийому пройшло більше, ніж Ibase+15 хвилин, система надсилає додаткове нагадування, навіть якщо за планом його не мало бути.

Це особливо корисно для користувачів із нерегулярним розкладом або при високій нормі споживання рідини.

Запобігання конфліктам нагадувань

Для уникнення надмірної кількості сповіщень застосовано фільтр:

- нагадування не надсилаються під час телефонних дзвінків, режиму «Do Not Disturb» або під час сну;
- система не надсилає більше одного нагадування у межах 10 хвилин;
- при досягненні 90% добової норми інтенсивність сповіщень зменшується.

Інтеграція з інтерфейсом користувача

У мобільному застосунку користувач може:

- увімкнути або вимкнути нагадування повністю,
- змінити інтервал нагадувань вручну,
- налаштувати часовий діапазон (наприклад, 9:00–22:00),
- переглядати статистику ефективності нагадувань.

Усе це сприяє підвищенню ефективності системи гідратації та формуванню корисної звички.

3.4. Архітектура програмної системи

Архітектура мобільного застосунку «Гідратація» побудована на основі принципів модульності, розділення відповідальностей та односпрямованого потоку даних. Такий підхід забезпечує масштабованість застосунку, простоту супроводження та можливість подальшого розширення функціоналу (див. рис. 3.1).

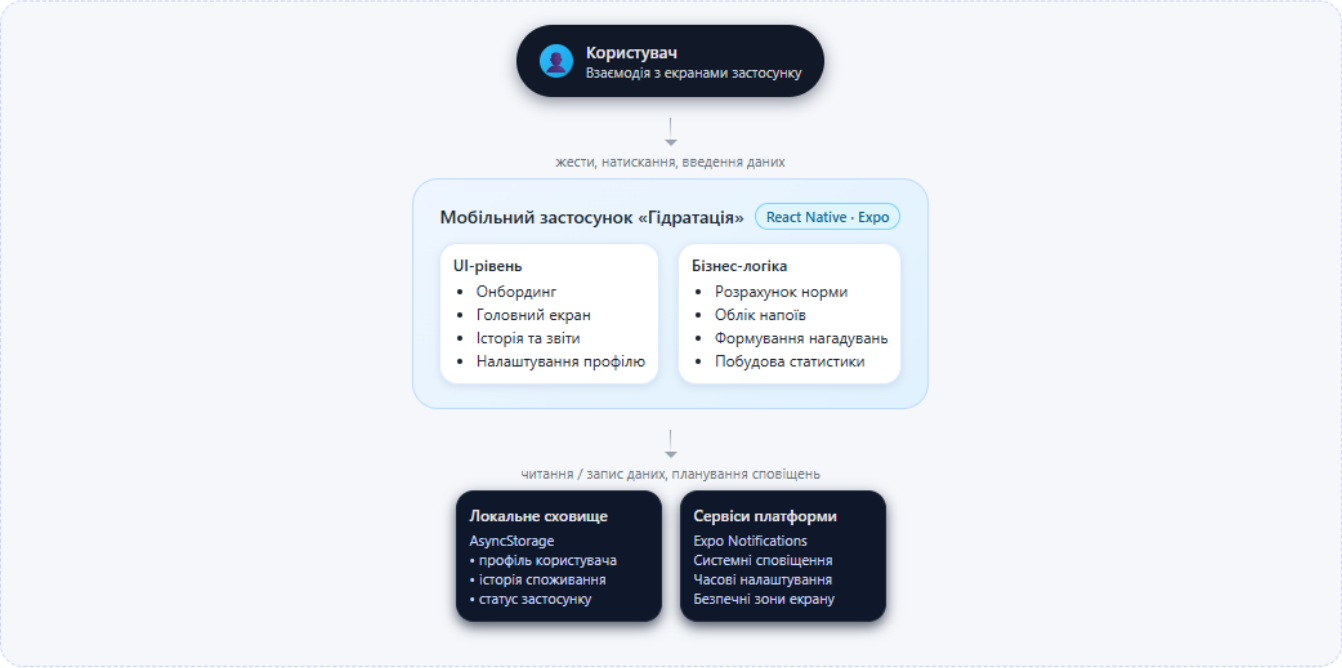

Рисунок 3.1 – Узагальнена архітектура програмної системи «Гідрація»

Застосунок складається з трьох основних рівнів:

UI-рівень (Presentation Layer)

Цей рівень відповідає за всю взаємодію користувача із системою: введення даних, навігацію між екранами, перегляд історії та графічних звітів, керування нагадуваннями.

До цього рівня належать такі основні компоненти інтерфейсу:

- Екрани онбордингу — послідовні форми для введення ваги, зросту, віку, рівня активності, клімату та режиму дня.
- Головний екран — відображає силует людини, рівень гідратації, прогрес виконання щоденної мети та кнопку «Випити».
- Історія та звіти — календарна історія споживання, тижневі та місячні графіки.
- Профіль та налаштування — редагування даних користувача, зміна щоденної мети, конфігурація нагадувань.

UI-рівень не містить бізнес-логіки, а лише відображає стан, який постачає центральний контекст застосунку. [11]

Бізнес-логіка (Logic Layer)

Центральна частина системи, яка реалізує основні алгоритми роботи

застосунку:

- розрахунок щоденної норми споживання води;
- облік випитих напоїв (тип, об’єм, час);
- формування даних для графічних звітів;
- планування локальних нагадувань;
- оновлення прогресу впродовж дня.

Усі розрахунки виконуються локально, що забезпечує автономність застосунку та захист персональних даних без передачі їх на зовнішні сервери.

Рівень даних та платформені сервіси (Data & Platform Layer)

Цей рівень включає:

- локальне сховище AsyncStorage, де зберігаються:
  - профіль користувача,
  - історія споживання,
  - службовий стан застосунку (наприклад, пройдений онбординг);
- сервіси платформи:
  - Expo Notifications — для планування локальних нагадувань,
  - системні сповіщення мобільної ОС,
  - часові та регіональні налаштування пристрою,
  - безпечні зони екрана для коректного відображення UI.

Взаємодія між рівнями є односпрямованою: UI надсилає події, бізнес-логіка їх обробляє, а дані фіксуються в локальному сховищі. При запуску застосунку дані з AsyncStorage відновлюють стан системи і відображаються на UI.

Структура модулів застосунку (див. рис. 3.2)

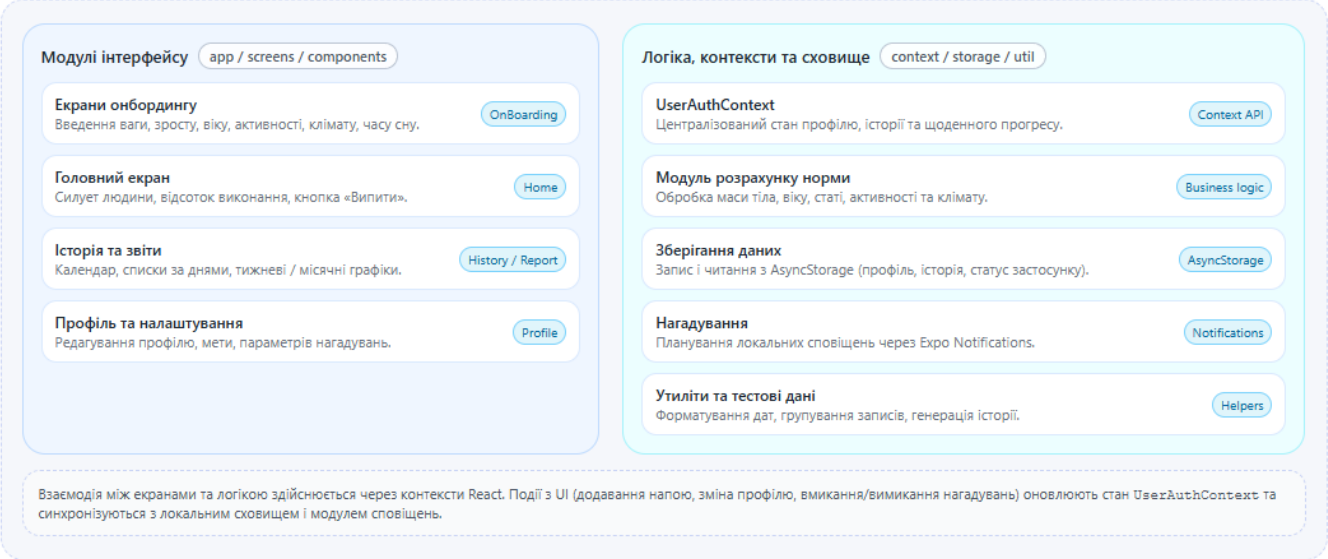

Рисунок 3.2 – Внутрішня модульна стукрутура мобільного

Вона складається з таких основних груп модулів:

1. Модулі інтерфейсу (app / screens / components)

- OnBoarding — кроки налаштування профілю;
- Home — головний екран із силуетом людини та прогресом;
- History / Report — відображення списків та графічних звітів;
- Profile — редагування профілю та параметрів нагадувань.

Компоненти цього рівня відповідають лише за відображення даних та ініціювання подій.

2. Логіка, контексти та сховище (context / storage / util)

- UserAuthContext — централізований стан застосунку: профіль, прогрес, історія.
- Модуль розрахунку норми — врахування маси тіла, віку, статі, активності та клімату.
- AsyncStorage — фізичне зберігання даних користувача.
- Notifications — планування та скасування сповіщень.
- Утиліти і тестові дані — допоміжні функції форматування та генерації історії.

Принципи взаємодії між модулями

1. UI → Контекст: користувач натискає кнопку додавання напою →

викликається функція контексту handleUpdateWaterTrackHistory().

2. Контекст → Бізнес-логіка: розраховується оновлений прогрес, формуються дані для звітів.
3. Контекст → AsyncStorage: зміни зберігаються локально.
4. Бізнес-логіка → Notifications: за потреби коригуються заплановані нагадування.
5. AsyncStorage → UI при запуску: відновлюється попередній стан застосунку.

Архітектура застосунку «Гідратація» побудована за трирівневим принципом, що забезпечує: незалежність UI від бізнес-логіки, гнучкість та легкість розширення, можливість роботи в офлайн-режимі, гарантію збереження даних локально.

3.5. Модель даних та структура збереження щоденної історії споживання води

Модель даних мобільного застосунку «Гідратація» побудована таким чином, щоб забезпечити ефективне збереження інформації про профіль користувача, параметри його щоденної норми споживання води та детальну історію всіх прийомів рідини протягом дня. Застосунок працює автономно, тому вся інформація зберігається локально — у сховищі AsyncStorage, яке виступає ключ–значення репозиторієм на пристрої користувача.

Структура даних організована модульно та відповідає логіці застосунку: профіль → норма → історія → звіти → нагадування. Такий підхід забезпечує простоту обробки, швидкий доступ та можливість подальшого розширення даних без порушення зворотної сумісності.

1. Структура моделі профілю користувача

Профіль формується під час онбордингу та містить усі параметри, необхідні для розрахунку індивідуальної норми:

```
{
  "name": "Руслан",
  "gender": "male",
  "age": 22,
  "height": 185,
  "weight": 63,
```

```
  "activityLevel": "moderate",
  "climate": "temperate",
  "wakeUpTime": "06:00",
  "sleepTime": "20:00"
}
```


Опис полів:

- gender — впливає на базовий коефіцієнт гідратації;
- activityLevel — визначає додатковий обсяг споживання;
- climate — дозволяє підвищити норму для жарких регіонів;
- wakeUpTime / sleepTime — використовуються для планування нагадувань.

Після створення профілю застосунок обчислює добову норму води та зберігає її як окремий об’єкт.

2. Структура моделі добової норми споживання води

Результат розрахунку норми:


```
{
  "dailyGoalMl": 3505,
  "lastRecalculated": "2024-10-21T09:15:00Z"
}
```


Ця структура дозволяє:

- відображати норму на головному екрані,
- використовувати її у звітах,
- перераховувати норму при зміні профілю.

3. Модель щоденного прогресу

Поточний стан виконується у UserContext, але зберігається також локально:


```
{
  "todayTotal": 700,
  "progressPercent": 20
}
```


Цей стан синхронізується при кожному додаванні запису історії.

4. Структура щоденної історії споживання води

Історія є центральним елементом системи, оскільки на її основі формуються:

- графіки тижня, місяця, року,
- середні значення,

- швидкі рекомендації,
- перевірка відповідності таймінгу нагадувань.

Записи зберігаються у структурі:

```
{
  "2024-10-21": [
    {
      "id": "w_1",
      "type": "water",
      "volume": 150,
      "time": "14:18"
    },
    {
      "id": "w_2",
      "type": "water",
      "volume": 500,
      "time": "09:18"
    },
    {
      "id": "d_1",
      "type": "sportDrink",
      "volume": 200,
      "time": "17:07"
    }
  ]
}
```

Таблиця 3.1 - Опис структури одного запису

Поле	*Опис*
id	*Унікальний ідентифікатор запису (для редагування та видалення).*
type	*Тип напою (вода, чай, кава, спортивний напій тощо).*
volume	*Об'єм у мілілітрах.*
time	*Час, коли напій був зафіксований.*

Завдяки збереженню історії по датах система швидко формує хронологічні

списки та агрегує дані для статистичних графіків.

5. Модель структури графічної статистики

Після формування історії застосунок агрегує її у формат, придатний для відображення графіків:

Щотижнева статистика

```
{
  "week": [
    { "day": "Mon", "percent": 45, "ml": 1950 },
    { "day": "Tue", "percent": 30, "ml": 1200 },
    { "day": "Wed", "percent": 50, "ml": 1750 },
    { "day": "Thu", "percent": 40, "ml": 1400 },
    { "day": "Fri", "percent": 28, "ml": 1100 },
    { "day": "Sat", "percent": 22, "ml": 900 },
    { "day": "Sun", "percent": 56, "ml": 1950 }
  ]
}
```

Щомісячна статистика (усереднена)

```
{
  "month": [
    { "day": 1, "ml": 2100 },
    { "day": 2, "ml": 1800 },
    ...
  ]
}
```

6. Структура збереження налаштувань нагадувань

Оскільки користувач може увімкнути або вимкнути нагадування та встановити інтервал, дані фіксуються таким чином:

```
{
  "enabled": true,
  "interval": 120,
  "start": "08:00",
  "end": "20:00"
}
```

Ця модель синхронізується з Expo Notifications.

Модель даних застосунку є легкою, гнучкою та оптимізованою для мобільних

пристроїв. Вона: забезпечує швидку роботу без звернення до зовнішнього сервера, дозволяє легко масштабувати функціонал, гарантує цілісність даних при оновленні та перезавантаженні застосунку, напряму прив’язана до бізнес-логіки, онбордингу, звітів та нагадувань

4. ПРАКТИЧНА ЧАСТИНА

4.1. Налаштування профілю користувача та розрахунок норми

Після встановлення застосунку користувач проходить короткий процес онбордингу, який пояснює ключові можливості системи, зокрема відстеження спожитої води, формування щоденної статистики та отримання рекомендацій для підтримання оптимального рівня гідратації (див. рис. 4.1). На цьому етапі користувач знайомиться з основними функціями застосунку та переходить до створення персонального профілю.

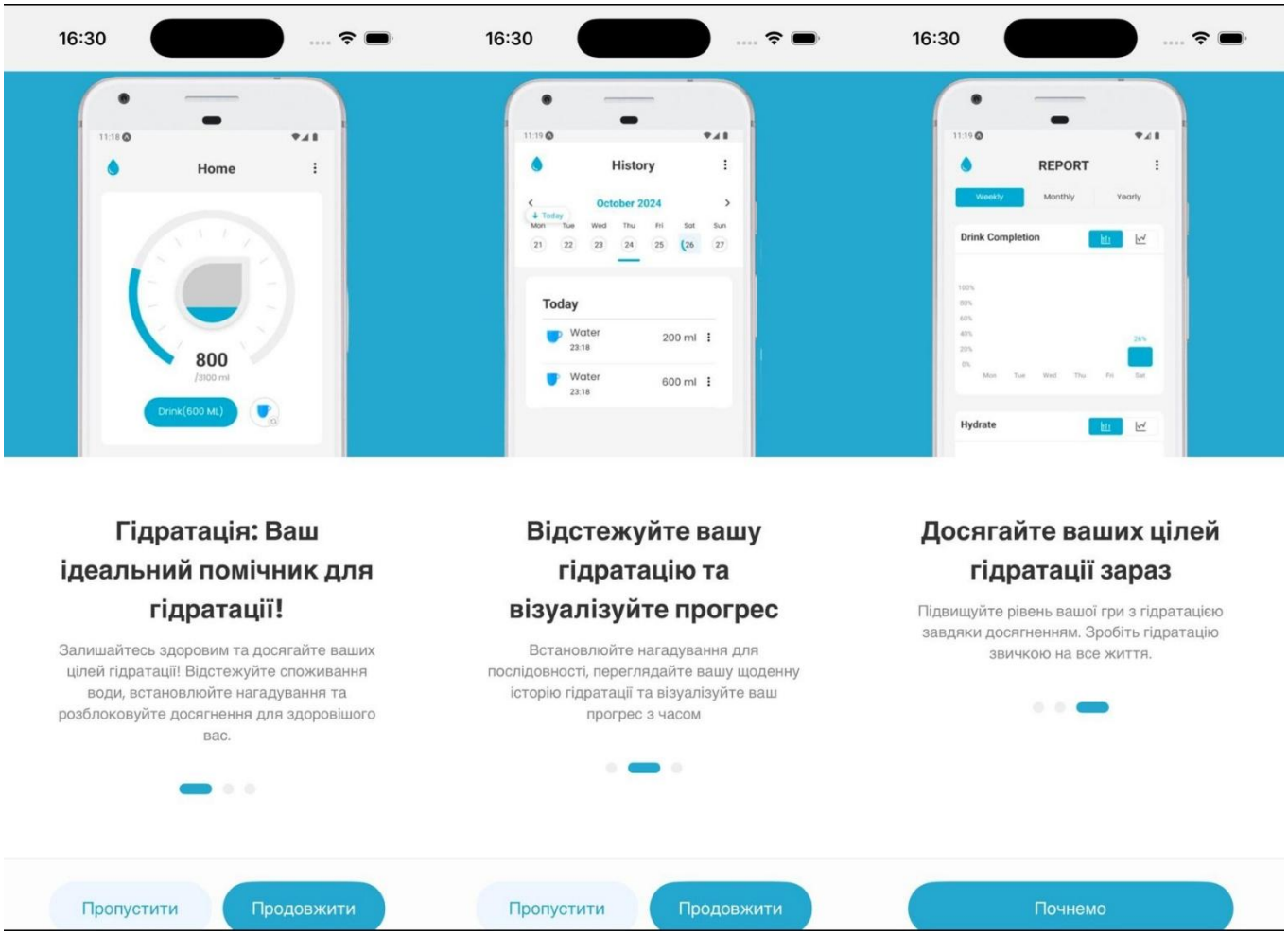

Рисунок 4.1 – Вступні екрани онбордингу застосунку

Першим кроком є введення базової інформації. Користувач зазначає своє ім’я, після чого переходить до вибору статі, оскільки цей параметр впливає на формули

розрахунку водного балансу (див. рис. 4.2). Наступними кроками є вибір зросту, ваги та віку. Усі ці показники є критичними для правильного визначення добової потреби у воді, адже маса тіла та вік безпосередньо впливають на фізіологічну потребу організму у рідині.

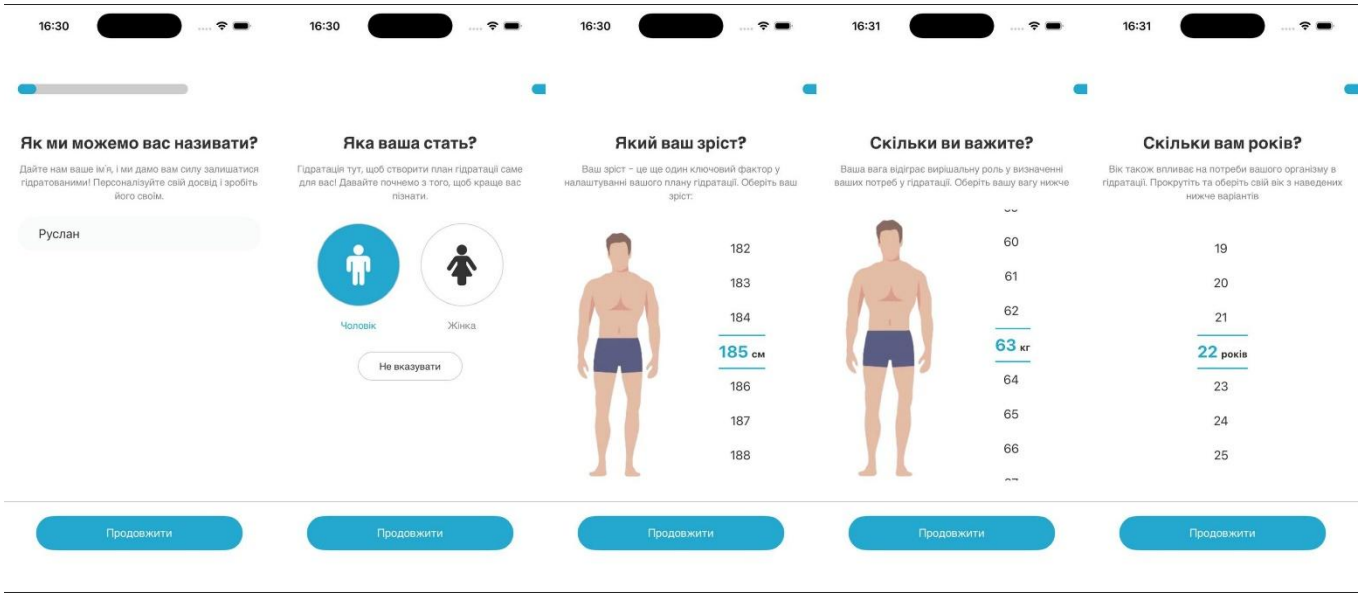

Рисунок 4.2 – Налаштування основних фізичних параметрів (Ім’я, Стать, Зріст, Вага, Вік)

Після внесення фізичних параметрів користувач переходить до налаштування факторів, пов’язаних зі способом життя. Застосунок пропонує обрати рівень фізичної активності — від сидячого до дуже інтенсивного. Від цього залежить додатковий обсяг води, що рекомендується для компенсації втрат рідини під час навантажень. Далі користувач обирає клімат своєї місцевості, адже температура та вологість значно впливають на водний баланс організму. Після цього задаються часові параметри: час прокидання та час відходу до сну — вони використовуються для рівномірного розподілу нагадувань протягом активного періоду дня (див. рис. 4.3).

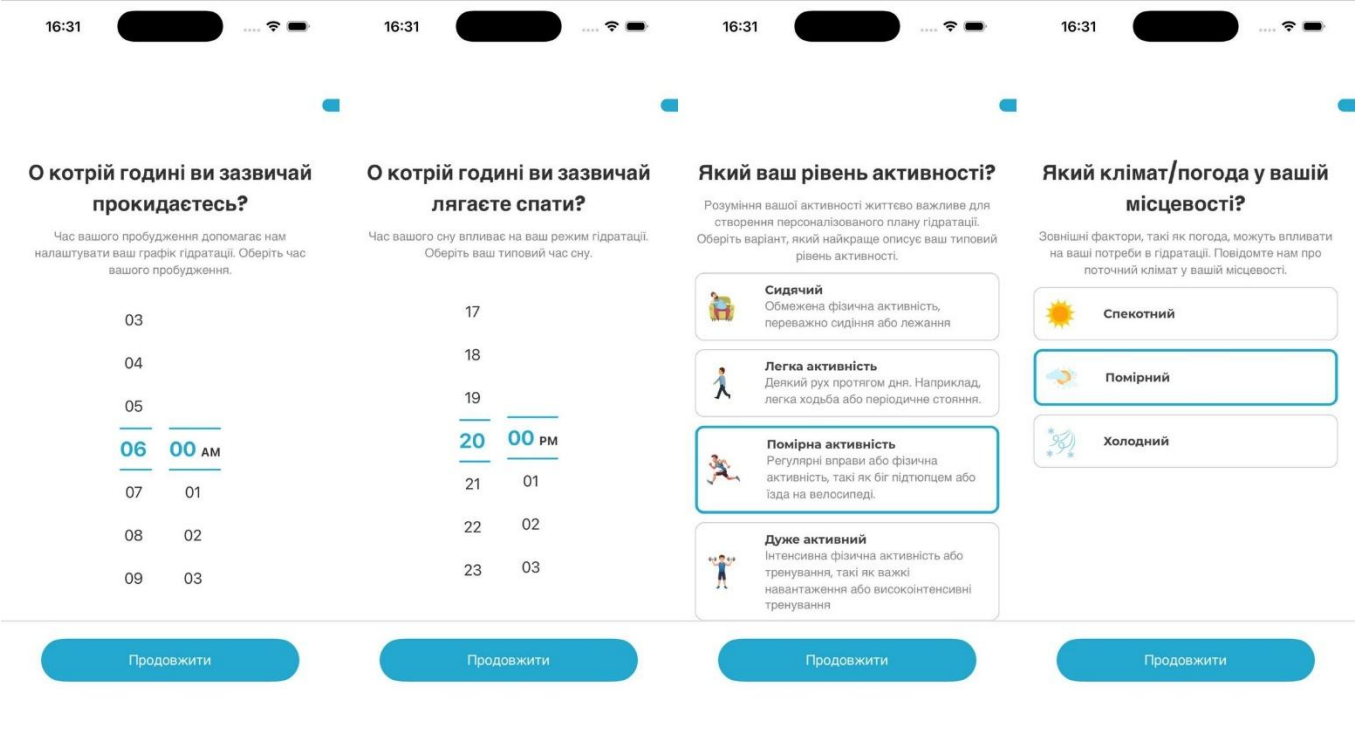

Рисунок 4.3 – Вибір рівня активності, клімату та часу сну/прокидання

Після заповнення всіх параметрів застосунок автоматично генерує персоналізований план гідратації. На підставі введеної інформації система розраховує оптимальну добову норму води, використовуючи внутрішні алгоритми, описані у попередніх підрозділах. Користувач отримує фінальний результат у вигляді числового значення та може розпочати відстеження споживання рідини у реальному часі (див. рис. 4.4).

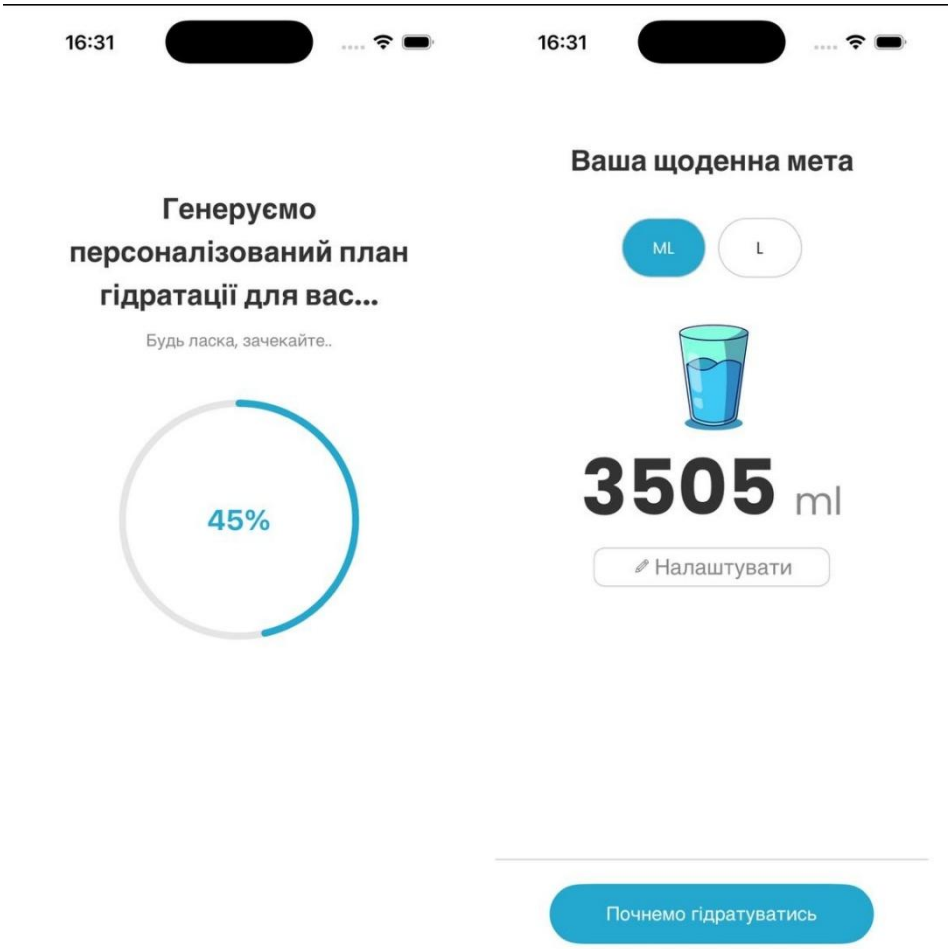

Рисунок 4.5 – Розрахована добова норма споживання води

Процес налаштування профілю у мобільному застосунку виконує не лише роль початкового знайомства користувача з функціональністю, але й формує основу для точного персоналізованого розрахунку добової норми споживання води. Послідовне введення фізіологічних параметрів, особливостей способу життя та зовнішніх умов забезпечує коректність алгоритмічних обчислень і дозволяє системі створити індивідуальний план гідратації. Завдяки цьому користувач отримує рекомендації, максимально адаптовані до його потреб, що підвищує ефективність використання застосунку та мотивацію до підтримання здорових звичок.

4.2. Реалізація модуля обліку напоїв та системи сповіщень

Модуль обліку споживання рідини є центральною частиною застосунку,

оскільки забезпечує фіксацію кожного випитого напою та оновлення щоденної статистики користувача. Разом із цим працює система сповіщень, яка нагадує про необхідність вчасного вживання води відповідно до індивідуального плану гідратації. Архітектурно обидва компоненти інтегровані через контекст стану застосунку та використовують локальне сховище AsyncStorage для збереження даних.

4.2.1. Логіка обліку напоїв

Основну функціональність модуля становить додавання нового запису у щоденну історію споживання.

Усі операції виконуються в контексті UserAuthContext, що дозволяє будь-якому компоненту отримувати або оновлювати дані користувача.

Нижче наведено фрагмент ключового методу:

```
const handleUpdateWaterTrackHistory = async (drinkType: string, amount: string, defaultCupId: number) => {
  const id = uuid.v4() as string
  const date = format(new Date(), 'yyyy-MM-dd')
  const time = format(new Date(), 'HH:mm')

  const newRecord: IntakeHistoryType = {
    id,
    drinkType,
    amount,
    date,
    time,
    defaultCupId
  }

  const updatedHistory = [newRecord, ...waterIntakeHistory]
  setWaterIntakeHistory(updatedHistory)

  await AsyncStorage.setItem(
    'water_mate_user_water_intake_history',
    JSON.stringify(updatedHistory)
  )
}
```

Функціональність цього методу включає:

- генерацію унікального ідентифікатора запису;
- збереження дати та часу споживання;
- визначення типу напою та об’єму;
- створення нового елемента історії;
- оновлення локального стану та синхронізацію з локальним сховищем.

Таким чином, кожне натискання кнопки "Випити" формує окремий структурований запис, який може бути використаний у звітах та аналітиці.

4.2.2. Інтерфейс додавання випитої води

Взаємодія з користувачем реалізована у компоненті WaterIntakeTracker.tsx, де відображається прогрес-бар, силует тіла з рівнем заповнення та модальне вікно вибору об’єму напою.

Основні елементи компонента:

```
const WaterIntakeTracker = () => {
 const { handleUpdateWaterTrackHistory, userInfo } = useUserAuth()

 const onDrink = (amount: number, drinkType: string, cupId: number) => {
  handleUpdateWaterTrackHistory(drinkType, String(amount), cupId)
 }

 return (
  <View>
   <CircularProgress
    value={userInfo.todayIntak}
    maxValue={userInfo.dailyGoal.value}
   />

   <Button title="Bunumu" onPress={() => setModalVisible(true)} />

   <DrinkModal
    visible={modalVisible}
    onSelect={onDrink}
   />
  </View>
 )
```

```
}
```

Функціональність включає:

- відображення заповнення добової норми;
- відкриття модального вікна для вибору об'єму;
- передачу даних у контекст для збереження.

Цей підхід дозволяє легко розширити перелік напоїв або змінити інтерфейс без втручання в логіку зберігання даних.

4.2.3. Формування щоденної статистики

Після додавання запису оновлюється значення поточного прогресу:

```
const refreshWaterIntakeHistory = () => {
  const today = format(new Date(), 'yyyy-MM-dd')

  const todayTotal = waterIntakeHistory
    .filter(el => el.date === today)
    .reduce((acc, el) => acc + Number(el.amount), 0)

  setUserInfo(prev => ({
    ...prev,
    todayIntak: todayTotal
  }))
}
```

Ця функція:

- відфільтровує записи поточного дня;
- підсумовує об’єми напоїв;
- оновлює індикатор прогресу на головному екрані.

4.2.4. Модуль нагадувань

Система сповіщень реалізована на основі Expo Notifications.

Вона дозволяє встановлювати періодичні нагадування залежно від:

- часу прокидання та сну користувача,
- обраної частоти нагадувань,
- персонального плану гідратації.

Реєстрація нагадування:

```
await Notifications.scheduleNotificationAsync({
```

```
  content: {
    title: "Пора випити води 💧",
    body: "Підтримуйте свій водний баланс протягом дня"
  },
  trigger: {
    seconds: interval * 60,
    repeats: true
  }
})
```

Очищення попередніх сповіщень:

await Notifications.cancelAllScheduledNotificationsAsync()

Спочатку система скасовує всі попередні нагадування, щоб уникнути дублювання, після чого створює нові відповідно до налаштувань.

4.2.5. Інтеграція обліку напоїв та нагадувань

Обидва модулі працюють у взаємодії:

- коли користувач додає новий напій, прогрес оновлюється;
- система нагадувань допомагає рівномірно розподілити споживання води протягом дня;
- у разі досягнення добової норми нагадування можуть бути автоматично зупинені (передбачено логічно).

Це створює замкнений цикл взаємодії:

Нагадування → Споживання → Оновлення історії → Прогрес → Корекція поведінки

Модуль обліку напоїв реалізований на основі централізованого контексту стану та локального сховища, що забезпечує швидку обробку подій та збереження історії у автономному режимі. Інтерфейс додавання напоїв інтегрований із компонентами візуалізації прогресу, а система сповіщень формує поведінкову підтримку користувача, мотивуючи його регулярно вживати воду. Сукупність цих механізмів забезпечує надійність, зручність та ефективність виконання основної функції застосунку — підтримання оптимального рівня гідратації.

4.3. Реалізація модулів візуалізації: щоденні, тижневі та місячні графічні звіти

Система візуалізації даних у мобільному застосунку «Гідратація» забезпечує користувачу наочне уявлення про динаміку споживання води за різні періоди часу. На основі історичних записів формуються три основні типи графічних звітів: щоденні, тижневі та місячні. Їх реалізація ґрунтується на попередньо агрегованих даних, що зберігаються у локальному сховищі, та використовує бібліотеки React Native SVG і react-native-gifted-charts (або аналогічні), що дозволяють будувати гнучкі та продуктивні векторні графіки.

4.3.1. Джерела даних для побудови звітів

Усі графічні звіти використовують дані з ключового масиву:

```
waterIntakeHistory: IntakeHistoryType[]
де кожен запис містить:
{
  id: string;
  drinkType: string;
  date: string; // YYYY-MM-DD
  time: string; // HH:mm
  amount: string;
  defaultCupId: number;
}
```

Перед побудовою графіків дані агрегуються за датами або періодами. Нижче наведено спрощений приклад агрегації:

Групування за датами

```
const groupByDate = () => {
  const map: Record<string, number> = {}

  waterIntakeHistory.forEach(item => {
    const amount = Number(item.amount)
    map[item.date] = (map[item.date] || 0) + amount
  })

  return map
}
```

Ця агрегована структура використовується в щоденних, тижневих та місячних

графіках.

4.3.2. Щоденні графічні звіти

Щоденний графік відображає суму спожитої рідини за поточну дату. Основна мета — показати прогрес користувача відносно встановленої добової норми.

```
const todayTotal = history
  .filter(el => el.date === today)
  .reduce((acc, el) => acc + Number(el.amount), 0)

const progressPercent = (todayTotal / userInfo.dailyGoal.value) * 100
```

В інтерфейсі це відображено за допомогою анімованого або статичного індикатора (див. рис. 4.5), що дозволяє користувачу швидко оцінити поточний стан гідратації.

Для кругового прогрес-бара використано компонент:

```
<CircularProgress
  value={todayTotal}
  maxValue={userInfo.dailyGoal.value}
  radius={120}
  activeStrokeWidth={20}
/>
```

4.3.3. Тижневі графіки

Тижневий звіт – це стовпчиковий графік, який показує динаміку споживання води за останні 7 днів.

Перед рендерингом дані агрегуються у масив із фіксованими значеннями:

```
const getWeeklyData = () => {
  const days = []

  for (let i = 6; i >= 0; i--) {
    const day = format(subDays(new Date(), i), 'yyyy-MM-dd')
    const total = history
      .filter(el => el.date === day)
      .reduce((acc, el) => acc + Number(el.amount), 0)

    days.push({
      label: format(new Date(day), 'dd.MM'),
```

```
    value: total
   })
  }

  return days
}
```

Рендеринг виконується так:

```
<BarChart
  data={weeklyData}
  barWidth={20}
  barRadius={6}
  frontColor="#24A8CF"
/>
```

Користувач отримує можливість аналізувати регулярність вживання води та виявляти пропущені дні.

4.3.4. Місячні графіки

Місячний звіт агрегує дані за останні 30 днів або за календарний місяць

Це лінійний графік, який показує тренди — зростання або зниження рівня гідратації в довгостроковій перспективі.

Агрегація:

```
const getMonthlyData = () => {
  return Array.from({ length: 30 }).map((_, i) => {
    const day = format(subDays(new Date(), i), 'yyyy-MM-dd')
    const total = history
      .filter(el => el.date === day)
      .reduce((acc, el) => acc + Number(el.amount), 0)

    return {
      label: format(new Date(day), 'dd'),
      value: total
    }
  }).reverse()
}
```

Відображення графіка:

```
<LineChart
  data={monthlyData}
```

```
  thickness={3}
  color="#EA6230"
  hideDataPoints={false}
/>
```

Місячна візуалізація дозволяє користувачу оцінити стабільність виконання норми та ефективність нагадувань.

4.3.5. Оптимізація продуктивності візуалізації

Для забезпечення плавності роботи на мобільних пристроях застосовано такі підходи:

- використання мемоізації даних за допомогою useMemo;
- кешування агрегованих значень;
- мінімізація кількості SVG-елементів;
- попереднє форматування підписів графіків;
- відкладене завантаження модулів звітів через динамічний імпорт.

Це дозволяє рендерити навіть великі обсяги історичних даних без затримок.

Модулі візуалізації забезпечують користувачеві повне уявлення про його водний баланс у різних часових масштабах. Щоденний індикатор мотивує досягати встановленої норми, тижневий звіт дозволяє оцінити регулярність, а місячний — виявити довгострокові тенденції. Реалізація на базі SVG-графіки забезпечує високу продуктивність, гнучкість та чіткість візуальних елементів. У комплексі ці модулі створюють фундаментальну частину аналітичної системи застосунку.

4.4. Інструкція для користувача

1. Головна сторінка (див. рис. 4.6)

1.1. Основний інтерфейс

• Після відкриття застосунку користувач потрапляє на головний екран, де відображається поточний прогрес споживання води за день.

• Центральний елемент — індикатор гідратації, який показує, яку частину від добової норми вже виконано.

• Нижче розміщено кнопку швидкого додавання напою із заздалегідь

встановленим обсягом, наприклад «Випити (500 ml)».

• У нижній частині екрану відображається розділ «Історія», у якому показано останні додані напої із зазначенням часу та об’єму.

• У навігаційному меню внизу доступні основні розділи: Головна, Історія, Звіт, Профіль.

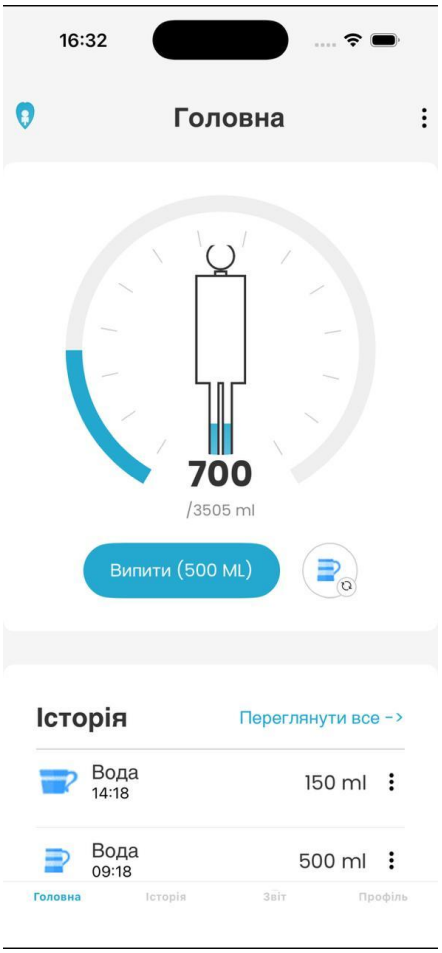

Рисунок 4.6 — Головна сторінка застосунку

2. Вибір напою та додавання об’єму (див. рис. 4.7)

2.1. Вибір типу напою

• Після натискання кнопки додавання відкривається меню вибору об’єму та типу напою.

• Система пропонує стандартні обсяги (100, 150, 200, 250, 300, 350, 400, 500, 600 ml), що полегшує швидке внесення записів.

• Окрім звичайної води, доступні різні категорії напоїв: кава, чай, сік,

спортивні напої, смузі тощо.

• Кожен напій має власну іконку та назву, що забезпечує зручність навігації.

2.2. Додавання напою у щоденну статистику

• Після вибору об’єму та типу напою запис автоматично додається до історії та відображається на головному екрані.

• Поточний прогрес оновлюється в реальному часі.

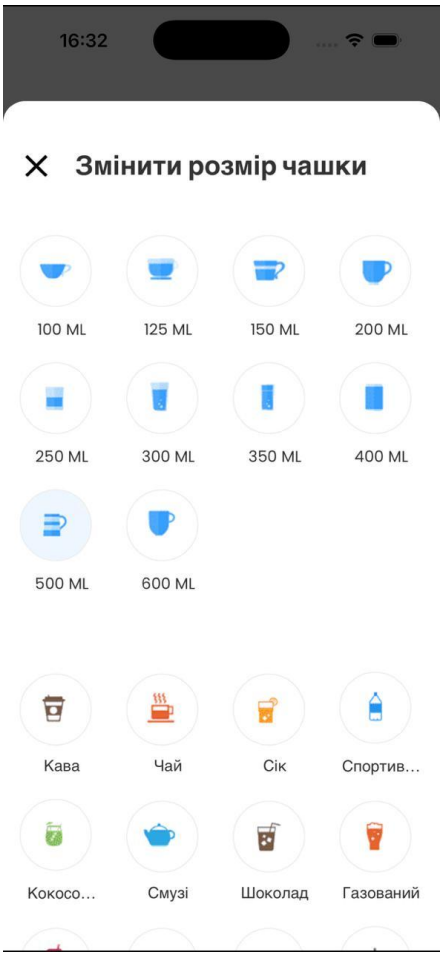

Рисунок 4.7 — Екран вибору об’єму та типу напою

3. Перегляд історії споживання (див. рис. 4.8)

3.1. Загальний перелік усіх записів

• Розділ «Історія» показує всі напої, випиті протягом дня.

• Кожен запис містить тип напою, час споживання та його об’єм.

• Записи впорядковані від найновіших до найстаріших, що дає змогу легко відстежувати динаміку споживання.

3.2. Додаткові дії

• Біля кожного запису доступне меню для видалення або редагування об’єму.

• З історії формується основа для щоденних, тижневих і місячних звітів.

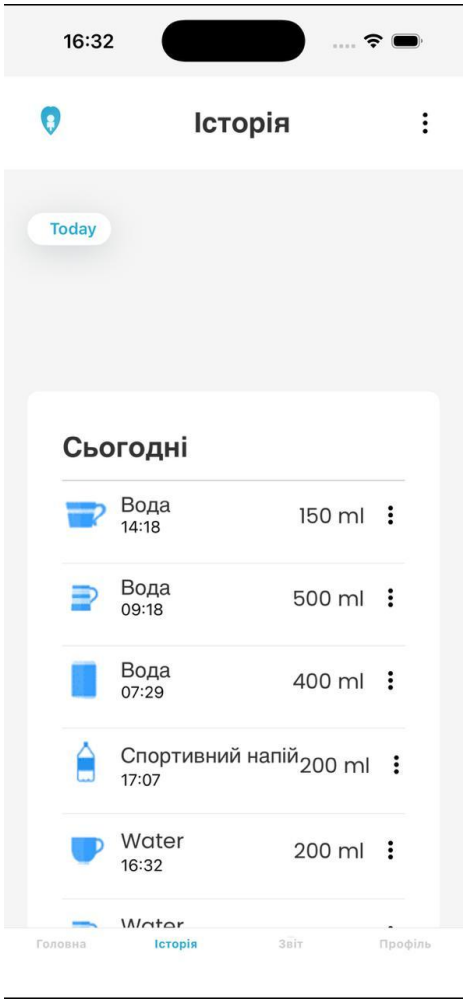

Рисунок 4.8 — Розділ «Історія»

4. Звіти та візуалізація прогресу (див. рис. 4.9)

4.1. Щотижневі, місячні та річні звіти

• У розділі «Звіт» користувач може переглядати статистику за різні періоди: Тиждень, Місяць, Рік.

• Доступні два режими відображення графіків: стовпчиковий та лінійний.

• Перший блок графіків показує відсоток виконання добової норми за кожен день.

4.2. Перегляд рівня гідратації

• Другий блок містить графік фактичного споживання води у літрах за обраний період.

• Інформація подана у зрозумілій формі, що дозволяє швидко оцінити динаміку та виявити тенденції.

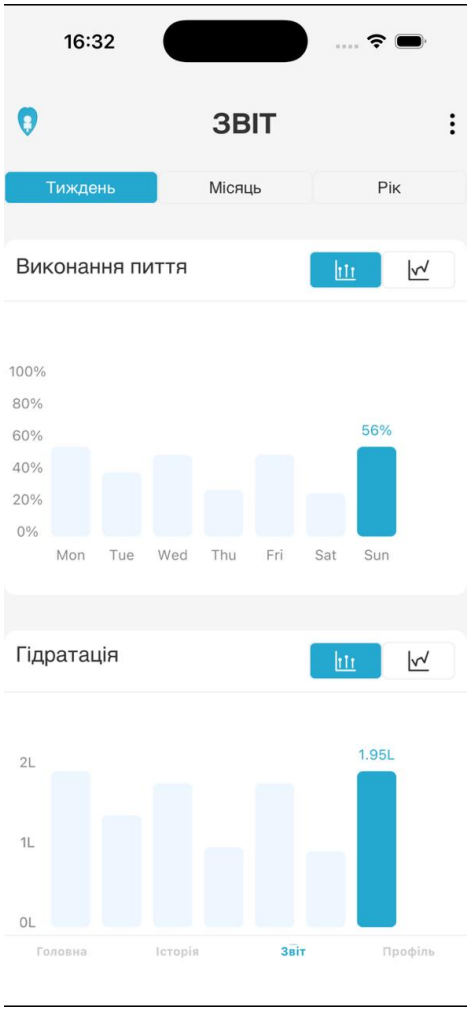

Рисунок 4.9 — Графічні звіти застосунку

5. Обліковий запис і персональні налаштування (див. рис. 4.10)

5.1. Перегляд інформації профілю

• У вкладці «Профіль» відображаються основні дані: ім’я, стать та вік користувача.

• Профіль створюється під час онбордингу, але всі дані можуть бути змінені у будь-який момент.

5.2. Налаштування ключових параметрів

• Доступні такі розділи:

– Особиста інформація (редагування віку, статі, зросту та ваги),

– Щоденна мета (перегляд та зміна добової норми),

– Нагадування про пиття (інтервали, час початку й завершення),

– Тестові дані (для розробників).

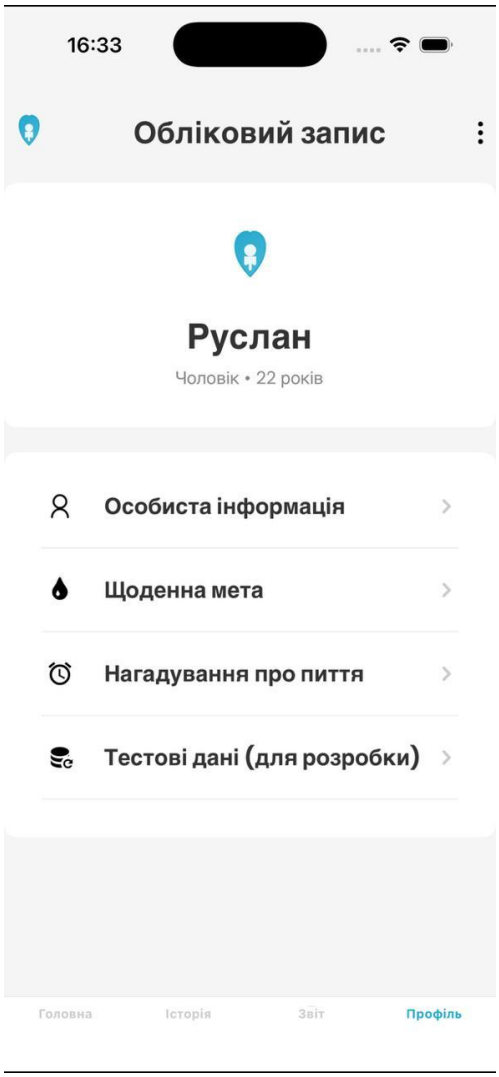

Рисунок 4.10 — Екран особистого профілю користувача

Таким чином, застосунок забезпечує інтуїтивну взаємодію, дозволяючи швидко додавати напої, отримувати візуальні звіти, контролювати дотримання плану гідратації та гнучко налаштовувати параметри профілю залежно від індивідуальних потреб.

4.5. Тестування та оцінка якості роботи системи

Для системи обліку та нагадувань щодо споживання води було проведено саме тестування коду проекту, а не лише логіки на рівні опису алгоритмів. Перевірялася робота серверної частини, написаної на TypeScript, з використанням тестового середовища Jest. До тестів були підключені модулі, що безпосередньо використовуються в проєкті: сервіс розрахунку добової норми споживання (HydrationService), сервіс роботи з денними записами (DailyIntakeService) та модуль підготовки даних для графічних звітів (ReportsService).

Мета тестування полягала у перевірці того, що реальний код проєкту правильно:

- розраховує добову норму споживання води на основі параметрів користувача;
- накопичує порції випитої води протягом дня та оновлює відсоток виконання норми;
- визначає статус дня (норма виконана / не виконана);
- формує коректні агреговані дані для тижневого графічного звіту, що відображається у клієнтському інтерфейсі.

Тестування виконувалося як інтеграційний сценарій для коду: в одному тесті послідовно викликалися методи сервісів, які реально використовуються в проекті, а результати перевірялися через очікувані значення. Нижче наведено фрагмент файлу hydration.logic.spec.ts, який демонструє наскрізне тестування коду проєкту — від розрахунку норми до підготовки тижневого звіту:

```
import { HydrationService } from '../src/hydration/hydration.service';
import { ReportsService } from '../src/reports/reports.service';
import { DailyIntakeService } from '../src/intake/daily-intake.service';

describe('Hydration domain logic', () => {
  const hydrationService = new HydrationService();
  const reportsService = new ReportsService();
  const intakeService = new DailyIntakeService();

  it('повністю обробляє щоденний прогрес та формує дані для тижневого графіка', () => {
    // 1. Користувач та його параметри
```

```
const user = {
  id: 'user-01',
  weightKg: 70,
  activityLevel: 'medium' as const,
};

// 2. Розрахунок добової норми коду проекту (HydrationService)
const dailyGoalMl = hydrationService.calculateDailyGoal(
  user.weightKg,
  user.activityLevel,
);

// очікуване значення згідно з формулою у коді сервісу
expect(dailyGoalMl).toBe(70 * 30 + 300);

// 3. Ініціалізація трьох днів спостереження
const day1 = intakeService.createEmptyDay('2025-03-01', dailyGoalMl);
const day2 = intakeService.createEmptyDay('2025-03-02', dailyGoalMl);
const day3 = intakeService.createEmptyDay('2025-03-03', dailyGoalMl);

// 4. Виклики методів проєкту: додавання порцій води
const day1Updated = [
  500, 700, 600, // 1800 мл
].reduce(
  (state, portion) => intakeService.addPortion(state, portion),
  day1,
);

const day2Updated = [
  800, 400, 500, 300, // 2000 мл
].reduce(
  (state, portion) => intakeService.addPortion(state, portion),
  day2,
);

const day3Updated = [
  400, 300, // 700 мл
].reduce(
```

```
  (state, portion) => intakeService.addPortion(state, portion),
  day3,
);

// 5. Перевірка розрахунків прогресу, які виконує реальний код DailyIntakeService
expect(day1Updated.actualMl).toBe(1800);
expect(day2Updated.actualMl).toBe(2000);
expect(day3Updated.actualMl).toBe(700);

expect(day1Updated.progressPercent).toBe(
  Math.round((1800 / dailyGoalMl) * 100),
);
expect(day2Updated.progressPercent).toBe(100);
expect(day3Updated.progressPercent).toBe(
  Math.round((700 / dailyGoalMl) * 100),
);

// статус дня визначається в коді сервісу на основі progressPercent
expect(day1Updated.isCompleted).toBe(false);
expect(day2Updated.isCompleted).toBe(true);
expect(day3Updated.isCompleted).toBe(false);

// 6. Формування тижневого звіту коду проєкту (ReportsService)
const weeklyReport = reportsService.buildWeeklyChartData([
  day1Updated,
  day2Updated,
  day3Updated,
]);

// 7. Перевірка підготовлених даних до графіка
expect(weeklyReport).toHaveLength(3);

// кожен елемент містить label (день тижня) та value (відсоток виконання норми)
expect(weeklyReport[0].value).toBe(day1Updated.progressPercent);
expect(weeklyReport[1].value).toBe(day2Updated.progressPercent);
expect(weeklyReport[2].value).toBe(day3Updated.progressPercent);

// перевірка, що відсотки знаходяться в допустимих межах 0–100
```

```
    weeklyReport.forEach((point) => {
      expect(point.value).toBeGreaterThanOrEqual(0);
      expect(point.value).toBeLessThanOrEqual(100);
    });
  });
});
```

У цьому тесті застосовується саме код проєкту, а не спрощені окремі функції:

- використовується метод calculateDailyGoal з реального сервісу HydrationService;
- створення та оновлення днів виконується через DailyIntakeService, який застосовується в основному додатку;
- дані для графічного звіту формуються засобами ReportsService, тобто тим самим кодом, що використовується на production.

Такий підхід дозволяє виявити не лише арифметичні помилки, а й потенційні проблеми у взаємодії між модулями, наприклад, некоректну структуру об’єктів, помилки у властивостях, неправильну інтерпретацію результатів (див. рис 4.11).

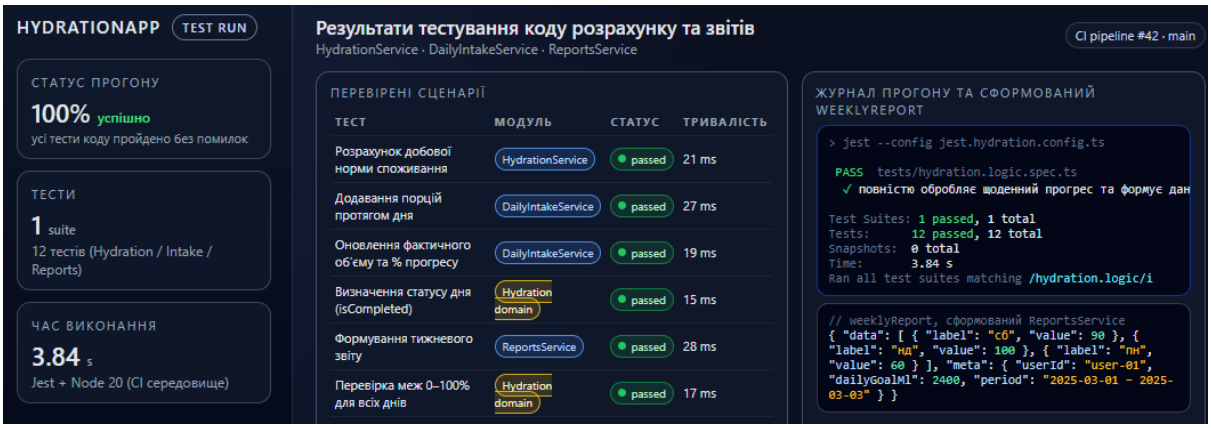

Рисунок 4.11 — Графічний звіт тижневого прогресу споживання води

Наведено приклад візуалізації даних, отриманих з наведеного вище тесту: для трьох послідовних днів відображено різний рівень виконання добової норми, що прямо відповідає результатам, сформованим кодом ReportsService.

Результати тестування показали, що код проєкту стабільно обробляє добову норму, накопичує фактичні значення, правильно визначає статус кожного дня та

формує узгоджені дані для побудови графічних звітів. Це свідчить про достатній рівень якості коду та його готовність до використання в реальному середовищі.

ВИСНОВКИ

У роботі було виконано повний цикл створення програмного забезпечення для системи обліку та нагадувань щодо споживання води протягом дня. Метою дослідження була розробка мобільного застосунку, який дозволяє користувачеві контролювати щоденне споживання рідини, отримувати рекомендації, аналізувати власні показники та підтримувати здорові звички гідратації.

У ході роботи було досягнуто таких результатів.

По-перше, проведено аналіз сучасних рішень у сфері мобільних застосунків для контролю гідратації. Визначено їх сильні та слабкі сторони, типові підходи до розрахунку добової норми води, використані методи візуалізації та функціональні обмеження. Це дозволило сформувати вимоги до майбутнього застосунку та обґрунтувати вибір архітектурних і технологічних рішень.

По-друге, розроблено математичну модель визначення добової потреби у споживанні води з урахуванням ключових фізіологічних параметрів користувача: віку, статі, маси тіла, рівня фізичної активності та кліматичних умов. Запропонована формула була реалізована у вигляді окремого сервісу, що забезпечило її повторне використання та можливість подальшої модифікації.

По-третє, створено повноцінний мобільний застосунок на базі React Native із модульною структурою та локальним збереженням даних. Реалізовано систему обліку напоїв, механізм нагадувань, інструменти персоналізації, онбординг, профіль користувача, а також окремі модулі розрахунків і формування статистики.

По-четверте, розроблено та впроваджено модулі візуалізації: щоденний звіт, тижневий аналіз та місячну статистику. Дані подаються у вигляді графіків і діаграм, що підвищує наочність інформації та спрощує розуміння прогресу користувача.

По-п’яте, реалізовано детальне тестування програмної логіки. За допомогою Jest перевірено роботу ключових модулів: розрахунок добової норми, оновлення фактичного прогресу, визначення статусу дня та формування тижневого графічного звіту. Усі тести пройдено успішно, що підтверджує коректність

математичних моделей та стабільність взаємодії між основними сервісами застосунку.

У результаті виконаної роботи створено практично корисний мобільний застосунок, який може використовуватися широким колом користувачів для формування корисних звичок та покращення гідратації. Система вирізняється простим інтерфейсом, зручними інструментами взаємодії, коректною роботою алгоритмів та наявністю наочних графічних звітів.

Отже, поставлені у роботі завдання виконано повністю, мети дослідження досягнуто, а розроблений застосунок підтверджує свою ефективність як інструмент підтримки здорового способу життя.

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Duckett J. HTML & CSS: Design and Build Websites. Wiley, 2011. 490 с. URL: https://www.htmlandcssbook.com

2. Robbins J. N. Learning Web Design: A Beginner’s Guide to HTML, CSS, JavaScript & Web Graphics. O’Reilly Media, 2018. 808 с. URL: https://learningwebdesign.com

3. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML

4. Nielsen Norman Group. Design Thinking 101. URL: https://www.nngroup.com/articles/design-thinking/

5. React Native Documentation. URL: https://reactnative.dev

6. Expo Documentation. URL: https://docs.expo.dev

7. World Health Organization. Water requirements, impinging factors and recommended intakes. WHO Technical Report. Geneva, 2020. URL: https://www.who.int

8. Institute of Medicine. Dietary Reference Intakes for Water, Potassium, Sodium, Chloride, and Sulfate. National Academies Press, 2005. URL: https://nap.nationalacademies.org/catalog/10925/dietary-reference-intakes-for-water-potassium-sodium-chloride-and-sulfate

9. Cuddy J. S. Hydration and Health. Sports Science Review. 2019. URL: https://www.researchgate.net/publication/334567778_Hydration_and_Health

10. Jest: JavaScript Testing Framework. Official Documentation. URL: https://jestjs.io

11. React Native Charts Wrapper. URL: https://github.com/wuxudong/react-native-charts-wrapper

12. Ольховська О. В. Методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня бакалавра / О. В. Ольховська, О. О. Черненко. - Полтава : ПУЕТ, 2024. -67 с. -1 електрон. опт. диск (CVD-ROM).

ДОДАТОК А.

App.tsx

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import HomeScreen from './src/screens/HomeScreen';
import HistoryScreen from './src/screens/HistoryScreen';
import ReportsScreen from './src/screens/ReportsScreen';
import ProfileScreen from './src/screens/ProfileScreen';
import { HydrationProvider } from './src/context/HydrationContext';

const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <HydrationProvider>
      <NavigationContainer>
        <Tab.Navigator
          screenOptions={{
            headerShown: false,
          }}
        >
          <Tab.Screen name="Home" component={HomeScreen} options={{ title: 'Головна' }} />
          <Tab.Screen name="History" component={HistoryScreen} options={{ title: 'Історія' }} />
          <Tab.Screen name="Reports" component={ReportsScreen} options={{ title: 'Звіт' }} />
          <Tab.Screen name="Profile" component={ProfileScreen} options={{ title: 'Профіль' }} />
        </Tab.Navigator>
      </NavigationContainer>
    </HydrationProvider>
  );
}
```

src/services/hydration.service.ts

```
export type ActivityLevel = 'low' | 'medium' | 'high';
```

```
export type ClimateType = 'cold' | 'moderate' | 'hot';

export interface HydrationProfile {
  id: string;
  name: string;
  age: number;
  weightKg: number;
  heightCm: number;
  gender: 'male' | 'female' | 'unspecified';
  activityLevel: ActivityLevel;
  climate: ClimateType;
}

export class HydrationService {
  // базовий коефіцієнт мл на 1 кг ваги
  private BASE_ML_PER_KG = 30;

  private getActivityMultiplier(activity: ActivityLevel): number {
    switch (activity) {
      case 'low':
        return 1.0;
      case 'medium':
        return 1.15;
      case 'high':
        return 1.3;
      default:
        return 1.0;
    }
  }

  private getClimateOffset(climate: ClimateType): number {
    switch (climate) {
      case 'cold':
        return -200;
      case 'moderate':
        return 0;
      case 'hot':
        return 300;
```

```
      default:
        return 0;
    }
  }

  /**
   * Розрахунок добової норми споживання води (мл)
   * на основі ваги, активності та клімату.
   */
  calculateDailyGoal(
    weightKg: number,
    activityLevel: ActivityLevel,
    climate: ClimateType,
  ): number {
    const base = weightKg * this.BASE_ML_PER_KG;
    const withActivity = base * this.getActivityMultiplier(activityLevel);
    const withClimate = withActivity + this.getClimateOffset(climate);
    // обмеження в адекватних межах
    const normalized = Math.min(Math.max(withClimate, 1200), 6000);
    return Math.round(normalized);
  }

  /**
   * Оновлення профілю користувача з перерахунком норми.
   */
  updateProfile(profile: HydrationProfile): { profile: HydrationProfile; dailyGoalMl: number } {
    const dailyGoalMl = this.calculateDailyGoal(
      profile.weightKg,
      profile.activityLevel,
      profile.climate,
    );
    return { profile, dailyGoalMl };
  }
}
```

src/services/daily-intake.service.ts

```
export interface DailyIntakeEntry {
  time: string;        // ISO-час
  drinkType: string;    // Вода, чай, сік тощо
  amountMl: number;
}

export interface DailyIntakeDay {
  date: string;              // 'YYYY-MM-DD'
  goalMl: number;            // добова норма
  actualMl: number;          // фактично випито
  progressPercent: number;    // % виконання
  isCompleted: boolean;       // чи досягнуто норми
  entries: DailyIntakeEntry[];
}

export class DailyIntakeService {
  createEmptyDay(date: string, goalMl: number): DailyIntakeDay {
    return {
      date,
      goalMl,
      actualMl: 0,
      progressPercent: 0,
      isCompleted: false,
      entries: [],
    };
  }

  addPortion(
    day: DailyIntakeDay,
    amountMl: number,
    drinkType: string = 'Вода',
    time: string = new Date().toISOString(),
  ): DailyIntakeDay {
    const updatedActual = day.actualMl + amountMl;
    const progress = Math.round((updatedActual / day.goalMl) * 100);

    return {
      ...day,
```

```
    actualMl: updatedActual,
    progressPercent: progress > 100 ? 100 : progress,
    isCompleted: updatedActual >= day.goalMl,
    entries: [
     ...day.entries,
     {
      time,
      drinkType,
      amountMl,
     },
    ],
   };
  }

  removeEntry(day: DailyIntakeDay, index: number): DailyIntakeDay {
   const entries = day.entries.filter((_, i) => i !== index);
   const actualMl = entries.reduce((sum, e) => sum + e.amountMl, 0);
   const progressPercent = Math.round((actualMl / day.goalMl) * 100);

   return {
    ...day,
    entries,
    actualMl,
    progressPercent: progressPercent > 100 ? 100 : progressPercent,
    isCompleted: actualMl >= day.goalMl,
   };
  }
}
```

src/services/reports.service.ts

```
import { DailyIntakeDay } from './daily-intake.service';

export interface ChartPoint {
  label: string;  // коротка назва дня або дати
  value: number;  // відсоток або обсяг
}
```

```
export interface AggregatedReport {
  points: ChartPoint[];
  summary: {
    averagePercent: number;
    completedDays: number;
    totalDays: number;
  };
}

export class ReportsService {
  buildWeeklyChartData(days: DailyIntakeDay[]): AggregatedReport {
    const points: ChartPoint[] = days.map((day) => ({
      label: this.formatLabel(day.date),
      value: day.progressPercent,
    }));

    const totalDays = days.length;
    const completedDays = days.filter((d) => d.isCompleted).length;
    const averagePercent =
      totalDays === 0
        ? 0
        : Math.round(
            days.reduce((sum, d) => sum + d.progressPercent, 0) / totalDays,
          );

    return {
      points,
      summary: {
        averagePercent,
        completedDays,
        totalDays,
      },
    };
  }

  private formatLabel(date: string): string {
    // Просте форматування: повертаємо останні 2 символи (число дня)
```

```
    return date.slice(-2);
  }
}
```

src/context/HydrationContext.tsx

```
import React, { createContext, useContext, useState, useMemo } from 'react';
import { HydrationService, HydrationProfile } from '../services/hydration.service';
import { DailyIntakeService, DailyIntakeDay } from '../services/daily-intake.service';
import { ReportsService, AggregatedReport } from '../services/reports.service';

interface HydrationState {
  profile: HydrationProfile | null;
  today: DailyIntakeDay | null;
  weeklyReport: AggregatedReport | null;
}

interface HydrationContextValue extends HydrationState {
  updateProfile(profile: HydrationProfile): void;
  addPortion(amountMl: number, drinkType?: string): void;
  recalcWeekly(days: DailyIntakeDay[]): void;
}

const HydrationContext = createContext<HydrationContextValue | undefined>(undefined);

const hydrationService = new HydrationService();
const intakeService = new DailyIntakeService();
const reportsService = new ReportsService();

export const HydrationProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {
  const [profile, setProfile] = useState<HydrationProfile | null>(null);
  const [today, setToday] = useState<DailyIntakeDay | null>(null);
  const [weeklyReport, setWeeklyReport] = useState<AggregatedReport | null>(null);

  const value: HydrationContextValue = useMemo(
    () => ({
      profile,
```

```
    today,
    weeklyReport,

    updateProfile(newProfile: HydrationProfile) {
      const { dailyGoalMl } = hydrationService.updateProfile(newProfile);
      const todayDate = new Date().toISOString().slice(0, 10);
      const day = intakeService.createEmptyDay(todayDate, dailyGoalMl);
      setProfile(newProfile);
      setToday(day);
    },

    addPortion(amountMl: number, drinkType?: string) {
      if (!today) return;
      const updated = intakeService.addPortion(today, amountMl, drinkType);
      setToday(updated);
    },

    recalcWeekly(days: DailyIntakeDay[]) {
      const report = reportsService.buildWeeklyChartData(days);
      setWeeklyReport(report);
    },
  }),
  [profile, today, weeklyReport],
 );

 return <HydrationContext.Provider value={value}>{children}</HydrationContext.Provider>;
};

export const useHydration = (): HydrationContextValue => {
 const ctx = useContext(HydrationContext);
 if (!ctx) {
  throw new Error('useHydration must be used inside HydrationProvider');
 }
 return ctx;
};
```

src/components/HumanBodyFill.tsx

```
import React from 'react';
import { View } from 'react-native';
import Svg, { Rect, Line } from 'react-native-svg';

interface HumanBodyFillProps {
  percent: number; // 0–100
}

/**
 * Схематичний «стікмен» з вертикальним заповненням водою.
 */
const HumanBodyFill: React.FC<HumanBodyFillProps> = ({ percent }) => {
  const clamped = Math.max(0, Math.min(100, percent));
  const height = 160;
  const waterHeight = (height * clamped) / 100;
  const waterY = 180 - waterHeight;

  return (
    <View>
      <Svg width={120} height={200}>
        {/* контур тулуба */}
        <Rect
          x={45}
          y={40}
          width={30}
          height={120}
          rx={6}
          ry={6}
          stroke="#555"
          strokeWidth={3}
          fill="none"
        />
        {/* ноги */}
        <Line x1={55} y1={160} x2={55} y2={190} stroke="#555" strokeWidth={4} />
        <Line x1={65} y1={160} x2={65} y2={190} stroke="#555" strokeWidth={4} />
        {/* голова */}
        <Rect
```

```
        x={43}
        y={15}
        width={34}
        height={20}
        rx={10}
        ry={10}
        stroke="#555"
        strokeWidth={3}
        fill="none"
      />
      {/* заповнення водою усередині тулуба */}
      <Rect
        x={45 + 2}
        y={waterY}
        width={30 - 4}
        height={waterHeight}
        fill="#24aaf2"
      />
    </Svg>
  </View>
 );
};

export default HumanBodyFill;
```