МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

МОБІЛЬНИЙ ЗАСТОСУНОК ПІДТРИМКИ ВОДНОГО БАЛАНСУ В ОРГАНІЗМІ ЛЮДИНИ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810324

Виконала студентка 4-го курсу, групи 402
Т. О. Удовик
«___» червня 2022 р.

Керівник: старший викладач
С. В. Дворецька
«___» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студентці групи 402 факультету комп'ютерних наук Удовик Тетяні Олександрівні.

1. Тема кваліфікаційної роботи «Мобільний застосунок підтримки водного балансу в організмі людини».

Керівник роботи Дворецька Світлана Володимирівна, старший викладач.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «07» грудня 2021 р. № 318

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: вхідна інформація про користувача.

4. Перелік питань, що підлягають розробці:

– аналіз предметної області та існуючих аналогів;

– огляд технологій та методів для розробки мобільного застосунку;

– проєктування застосунку;

– програмна реалізація мобільного застосунку;

– проведення тестування застосунку;

– аналіз результатів розробки.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Аналіз вимог до організації заходів техніки безпеки та охорони праці під час роботи за комп’ютером».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Старший викладач кафедри екології Макарова О.В.	

Керівник роботи __старший викладач Дворецька С. В._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Удовик Т. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «20» листопада 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Мобільний застосунок підтримки водного балансу в організмі людини.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2021	30.10.2021	Виконано
2	Отримання завдання на виконання БКР	20.11.2021	22.11.2021	Виконано
3	Складання календарного плану	05.12.2021	07.12.2021	Виконано
4	Отримання завдання на переддипломну практику	20.05.2022	20.05.2022	Виконано
5	Проходження переддипломної практики, написання практичної частини БКР	23.05.2022	04.06.2022	Виконано
6	Розробка звіту з переддипломної практики	04.06.2022	06.06.2022	Виконано
7	Початок виконання БКР: збір матеріалу та даних, написання аналітичної частини БКР	28.02.2022 та 06.06.2022	27.03.2022 та 19.06.2022	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	31.05.2022	Виконано
9	Розробка звіту з переддипломної практики	01.06.2022	03.06.2022	Виконано
10	Доробка та остаточне оформлення БКР	02.06.2022	20.06.2022	Виконано
11	Подання БКР рецензенту	16.06.2022	18.06.2022	Виконано
12	Подання БКР, електронні копії та інші необхідні документи до захисту	20.06.2022	20.06.2022	Виконано
13	Захист БКР перед екзаменаційною комісією	27.06.2022	27.06.2022	Виконано

Розробила студентка _____Удовик Т.О._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи __старший викладач Дворецька С.В._____ _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студентки групи 402
ЧНУ ім. Петра Могили
Удовик Тетяни Олександрівни
«Мобільний застосунок підтримки водного балансу в організмі людини»

Об’єктом дослідження даної роботи є процес підтримки водного балансу в організмі людини.

Предметом дослідження є технології розробки мобільних застосунків.

Метою роботи є покращення водного балансу та самопочуття користувачів за рахунок систематичного вживання необхідної норми води, шляхом розробки мобільного застосунку на базі операційної системи Android.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, п’яти розділів та висновків.

У першому розділі був проведений аналіз предметної сфери та наявних аналогів застосунків для підтримки водного балансу в організмі людини.

У другому розділі наведено опис технологій для розробки мобільного застосунку.

У третьому розділі було створено структуру мобільного застосунку, а також виконано прототипування та розробку дизайну.

У четвертому розділі було описано архітектуру та програмну реалізацію застосунку, написано керівництво користувача.

В результаті розроблено мобільний застосунок підтримки водного балансу в організмі людини.

Бакалаврська кваліфікаційна робота містить: сторінок – 92, таблиць – 7, рисунків – 32, джерел – 27, додатків – 1.

Ключові слова: машинне навчання, модель лінійної регресії, Android OS, Android Studio, Kotlin, Python, TensorFlow.

ABSTRACT

To the bachelor's qualification work by the student of group 402 of
Petro Mohyla Black Sea National University
Udovyk Tetiana Oleksandrivna
«Mobile application for maintaining water balance in the human body»

The object of research of this work is the process of maintaining water balance in the human body.

The subject of research is the technology of mobile application development.

The aim of the work is to improve the water balance and well-being of users through the systematic use of the required amount of water by developing a mobile application based on the Android operating system.

The work consists of a professional section and a special section on labor protection. The explanatory note consists of an introduction, five chapters and conclusions.

The first section analyzes the subject area and existing analogues of applications for control and maintenance of water balance in the human body.

The second section reviews technologies for the mobile application.

In the third section, the structure of the mobile application was created, as well as prototyping and design development.

The fourth section described the architecture and software implementation of the application, wrote a user manual.

As a result, a mobile application for maintaining water balance in the human body has been developed.

The bachelor's thesis contains: pages – 92, tables – 7, images – 32, references – 27, supplements – 1.

Keywords: machine learning, linear regression model, Android OS, Android Studio, Kotlin, Python, TensorFlow.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ДСанПіН	–	Державні Санітарні Правила і Норми
ЕОМ	–	Електронна обчислювальна машина
КЗ	–	Комп’ютерні засоби
КК	–	Користувач комп’ютера
ПК	–	Персональний комп’ютер
API	–	Application Programming Interface
HIG	–	Human Interface Guidelines
IDE	–	Integrated Development Environment
IOS	–	iPhone Operating System
MVC	–	Model-View-Controller
MVP	–	Model-View-Presenter
MVVM	–	Model-View-ViewModel
OS	–	Operating System
UI	–	User Interface
UX	–	User Experience
XML	–	Extensible Markup Language

Пояснювальна записка

до кваліфікаційної роботи

на тему:

МОБІЛЬНИЙ ЗАСТОСУНОК ПІДТРИМКИ ВОДНОГО БАЛАНСУ В ОРГАНІЗМІ ЛЮДИНИ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810324

Виконала студентка 4-го курсу, групи 402
Т. О. Удовик
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Керівник: *старший викладач*
(наук. ступінь, вчене звання)
С. В. Дворецька
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

ВСТУП

Життя людини безпосередньо залежить від багатьох факторів, найважливішим з яких є стан здоров'я її організму. Щоб бути здоровим, потрібно докласти багато зусиль. А насамперед треба контролювати та підтримувати водний баланс свого організму. Вода допомагає запобігти більшості захворювань, особливо онкологічних. На думку фахівців, чим більше води людина споживає, тим більше рідини виділяється організмом. Разом з нею виходять збудники, що викликають захворювання, а також канцерогени, що сприяють появі раку. Також знижується розумова діяльність в результаті порушення водного балансу. Часто виникає депресія, а також зменшується рівень захисних механізмів організму, що є основною причиною появи простудних захворювань. Порушення водного балансу безпосередньо впливає на фізичне та психічне здоров'я. Рівень води має вирішальне значення для функціонування всіх систем організму людини.

Таким чином, **актуальність** теми даної випускної кваліфікаційної роботи є безсумнівною і полягає в необхідності підтримки водного балансу людини, який безпосередньо впливає на наше фізичне самопочуття та когнітивні здібності.

Метою роботи є покращення водного балансу та самопочуття користувачів за рахунок систематичного вживання необхідної норми води, шляхом розробки мобільного застосунку на базі операційної системи Android.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- виконати аналіз предметної сфери;
- дослідити та проаналізувати наявні аналоги мобільних застосунків для підтримки водного балансу;
- сформувати вимоги до функціоналу розроблювального застосунку;
- розглянути принципи розробки Android-застосунків;

– обрати технології та методи розробки мобільного застосунку підтримки водного балансу в організмі людини;

– спроєктувати застосунок;

– реалізувати застосунок.

Об’єктом дослідження даної роботи є процес підтримки водного балансу в організмі людини.

Предметом дослідження є технології розробки мобільних застосунків.

Технології, що мають бути використані у роботі – мова програмування Kotlin, платформа Android, мова програмування Python та технологія TensorFlow.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ. ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної сфери

Для покращення стану здоров’я потрібно, як було сказано вище, контролювати водний баланс. Але що воно таке і як його підтримувати?

Водний баланс — це процес, в якому вода надходить і виходить з організму. Метою підтримки водного балансу є підтримання рівня гідратації в організмі, запобігаючи зневодненню. Зневоднення може викликати проблеми зі здоров’ям, включаючи головний біль, запаморочення, втому, нудоту та блювоту. Це також може підвищити ризик інфекцій, особливо тих, які поширюються від однієї людини до іншої повітряно-крапельним шляхом, наприклад, при пневмонії або туберкульозі [1].

Тіло людини складається з води. Точніше, клітини людини приблизно на 60% складаються з води. Вода в організмі допомагає організму функціонувати та виконувати різноманітні функції. Наприклад, вода допомагає переносити хімічні речовини в клітину та з неї, підтримує органи та допомагає підтримувати клітини здоровими. Вода також допомагає переносити поживні речовини до клітин, регулює температуру тіла та запобігає накопиченню продуктів життєдіяльності [2].

Важливо підтримувати водний баланс в організмі людини, оскільки це допомагає підтримувати належне функціонування організму. Наприклад, якщо порушується водний баланс, організм може не виконувати свої нормальні функції. Наприклад, якщо в клітинах порушується водний баланс, клітина може не виконувати функції, які вона повинна виконувати. Так само, якщо водний баланс порушується в м’язах, м’язи можуть не працювати належним чином.

Також важливо підтримувати водний баланс в організмі людини, оскільки це сприяє збереженню здоров’я організму. Наприклад, якщо в клітинах порушується водний баланс, клітина може почати гинути. Аналогічно, якщо водний баланс порушується в м’язах, м’язи можуть почати виснажуватися. Якщо

в органах порушується водний баланс, орган може почати дегенерувати. Одним словом, якщо порушується водний баланс у всьому організмі, організм може почати страждати від різних проблем зі здоров’ям.

Багато людей п’ють воду лише тоді, коли відчувають спрагу. Це велика помилка. Спрага вказує на зневоднення. Незважаючи на те, що зневоднення незначне, воно все ж має сильний вплив на організм. Не можна пити багато води під час сніданку, обіду, вечері та після вечері. Це значно знизить концентрацію шлункового соку і порушить процес травлення. Рекомендується пити воду між прийомами їжі. Людині потрібна достатня кількість води, щоб вижити, і вона повинна постійно збільшувати її кількість в організмі. Це дуже важливо, оскільки вода є важливою частиною організму. Водний баланс визначає різницю між водою, яка надходить в організм і виробляється організмом під час певних метаболічних процесів, і водою, яку ми втрачаємо з організму разом з потом і повітрям, яким дихаємо.

Щоб зберегти своє здоров'я, потрібно вживати стільки води, скільки можна втратити. Це може здатися складним, але досить краще зрозуміти функціонування нашого організму, щоб наблизитись до кількості води, яку потрібно споживати щодня. Вода — це життя на планеті. Вживання рідини допоможе збільшити продуктивність, дозволить покращити здоров'я та життєвий тонус.

Отже, що станеться з організмом, якщо випити склянку води на голодний шлунок.

1. Починається обмін речовин. Вода підтримує обмін речовин, що позитивно впливає на багато систем організму. Лікарі та дієтологи рекомендують випивати склянку води, щоб розбудити організм і запустити обмін речовин.

2. Профілактика простудних захворювань. Вода розчиняє і видаляє слиз у верхніх дихальних шляхах. Одної склянки достатньо, щоб полегшити симптоми застуди: кашель і біль у горлі.

3. Дезінтоксикація організму. Вода сприяє виведенню токсичних речовин та застійних рідин. Якщо багато пити рідини, то підвищується температура тіла, це спосіб рясного потовиділення та виведення токсинів.

4. Уповільниться процес старіння. Вода уповільнює старіння шкіри. Якщо регулярно підтримувати баланс рідини, клітини шкіри зберігають свою еластичність та пружність, завдяки чому шкіра стає гладкою та сяючою.

5. Поліпшиться робота внутрішніх систем організму. Регулярний прийом покращує роботу судин і кровоносної систем організму. Вода руйнує жирові відкладення по всій нервовій системі і, таким чином, починає краще працювати. Цей ефект був підтверджений клінічними випробуваннями.

6. Покращиться структура волосся.

7. Зміцнюється імунітет.

Поради для відновлення водного балансу організму:

– одна склянка за 30 хвилин до сніданку для нормальної роботи травлення;

– півтори – дві склянки за кілька годин після сніданку;

– одна склянка за 30 хвилин до обіду;

– півтори-дві склянки за кілька годин протягом дня;

– одна склянка за 30 хвилин до вечері;

– одна склянка після вечері;

– одна склянка перед сном;

– цей метод можливий за відсутності протипоказань;

– якщо таке регулярне пиття води незвичне і важке, можна спробувати продовжити цей процес протягом дня на кілька ковтків.

Збалансоване споживання води покращує метаболізм. Також за рахунок підтримки водного балансу в організмі не накопичуються токсини, тобто не відбувається перевантаження печінки і нирок. Ваша шкіра стане дуже гладкою та пружною зі здоровим тоном.

1.2 Огляд та аналіз наявних аналогів застосунків для контролю та підтримки водного балансу

У даній роботі буде розглянуто аналоги серед мобільних застосунків підтримки водного балансу в організмі людини. Всі ці вже існуючі програмні продукти по-різному реалізовані, а також мають різний бюджет. Вони допомагають користувачам слідкувати за станом свого організму та за рівнем балансу води.

Мобільний застосунок «Моя вода». Застосунок для підтримки водного балансу в організмі людини, який допоможе пити воду постійно та надає функціонал підрахунку норми рідини і нагадування, коли саме треба випити склянку води. Містить в собі статистику щодо дотримання норми води упродовж тижня, а також динаміку зміни ваги людини.

Функції програми:

- калькулятор необхідної кількості води;
- нагадування;
- статистика у вигляді графіків;
- поради як пити більше води;
- мотивуючі досягнення.

Переваги:

- багатофункціональність;
- підходить для більшості смартфонів;
- не потребує підключення до Інтернету;
- наявність автоматизації.

Недоліки:

- наявність реклами;
- незручність;
- немає здатності коригування норми води залежно від погодних умов та фізичної активності.

Приклади інтерфейсу застосунку зображені на рис. 1.1.

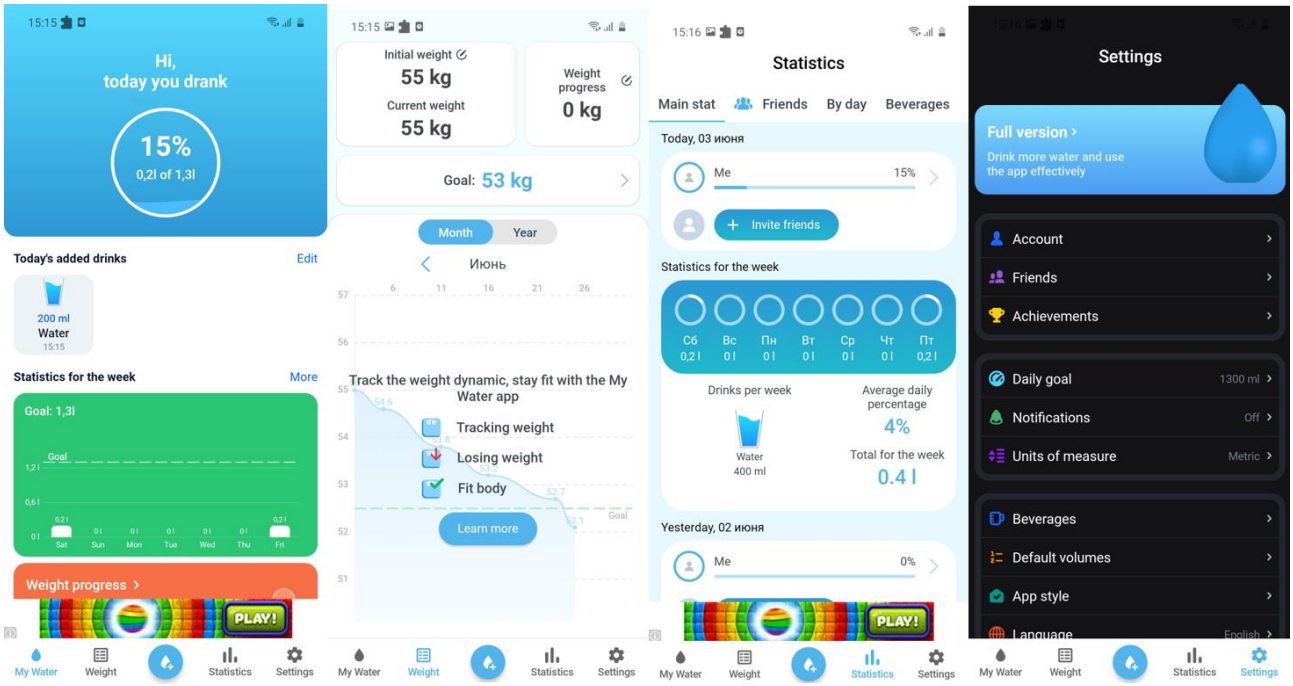

Рисунок 1.1 — Інтерфейс застосунку «Моя вода»

Мобільний застосунок «BeWet: Нагадування пити воду». Зручний застосунок для охочих схуднути, покращити свій метаболізм, а також підтримувати водний баланс власного тіла. Повідомлення можна ввімкнути автоматично та встановити ручні налаштування часу, коли повинні приходити повідомлення пити воду.

Функції програми:

– вибір напоїв з їхніми індексами гідратації для вірного підрахунку насичення водою;

– контроль ваги;

– підрахунок щоденної норми води;

– статистика у вигляді графіків.

Переваги:

– багатофункціональність;

– підходить для більшості смартфонів;

– коригування норми води залежно від погодних умов та фізичної активності;

– не потребує підключення до Інтернету;

– зручність;

– гарний інтерфейс;

– синхронізація із Google Fit;

– повідомлення про те, коли саме потрібно випити воду;

– наявність автоматизації.

Недоліки:

– наявність реклами.

Приклади інтерфейсу застосунку зображені на рис. 1.2.

Рисунок 1.2 — Інтерфейс застосунку «BeWet: Нагадування пити воду»

Мобільний застосунок «Water Drink». Мобільний застосунок «Water Drink» — трекер води, який забезпечує необхідний водний баланс в організмі

людини. Це система розумних оповіщень, яка включає велику кількість різних функцій.

Цей застосунок використовується для того, аби:

– пити воду для схуднення;
– позбавлятися токсинів;
– очищати шкіру;
– поповнювати мінерали та знижувати стрес;
– захищати сечову систему;
– підвищувати запас енергії.

Функції програми:

– повідомлення про те, коли саме потрібно випити воду;
– детальна мотивуюча статистика;
– інформація про погоду;
– вибір стандартних користувацьких напоїв;
– створення цілі по гідратації.

Переваги:

– багатофункціональність;
– підходить для більшості смартфонів;
– коригування норми води залежно від погодних умов та фізичної активності;
– не потребує підключення до Інтернету;
– повідомлення про те, коли саме потрібно випити воду;
– наявність автоматизації.

Недоліки:

– наявність реклами;
– незручність;
– наявність багів.

Приклади інтерфейсу застосунку зображені на рис. 1.3.

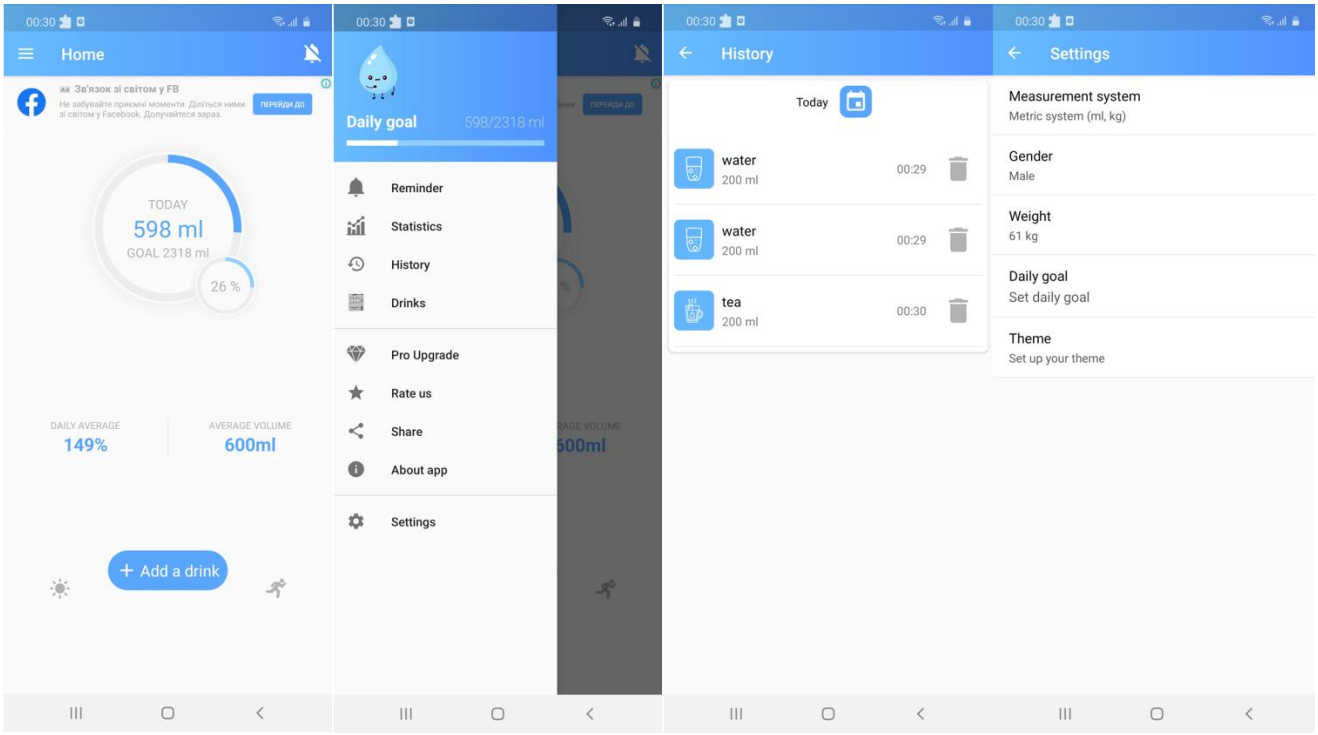

Рисунок 1.3 — Інтерфейс застосунку «Water Drink»

Мобільний застосунок «Водокотик». Цей мобільний застосунок служить для нагадування пити воду, підтримки водного балансу та як трекер води.

«Водокотик» — дуже простий у використанні застосунок, який має всі необхідні функції для формування корисної звички пити воду.

Також програма надсилає нагадування, щоб точно не забути попити, а також надсилає повідомлення-компліменти за кожну випиту склянку води.

Функції програми:

– розрахунок необхідної норми води з урахуванням статі та ваги;

– коригування норми води залежно від погодних умов та фізичної активності;

– повідомлення про те, коли саме треба пити воду;

– детальна статистика та історія.

Переваги:

– багатофункціональність;

– підходить для більшості смартфонів;

– не потребує підключення до Інтернету;

– коригування норми води залежно від погодних умов та фізичної активності;

– повідомлення про те, коли саме потрібно випити воду;

– система підтримує різні одиниці вимірювання для зручності;

– наявність автоматизації.

Недоліки:

– наявність реклами.

Приклади інтерфейсу застосунку зображені на рис. 1.4.

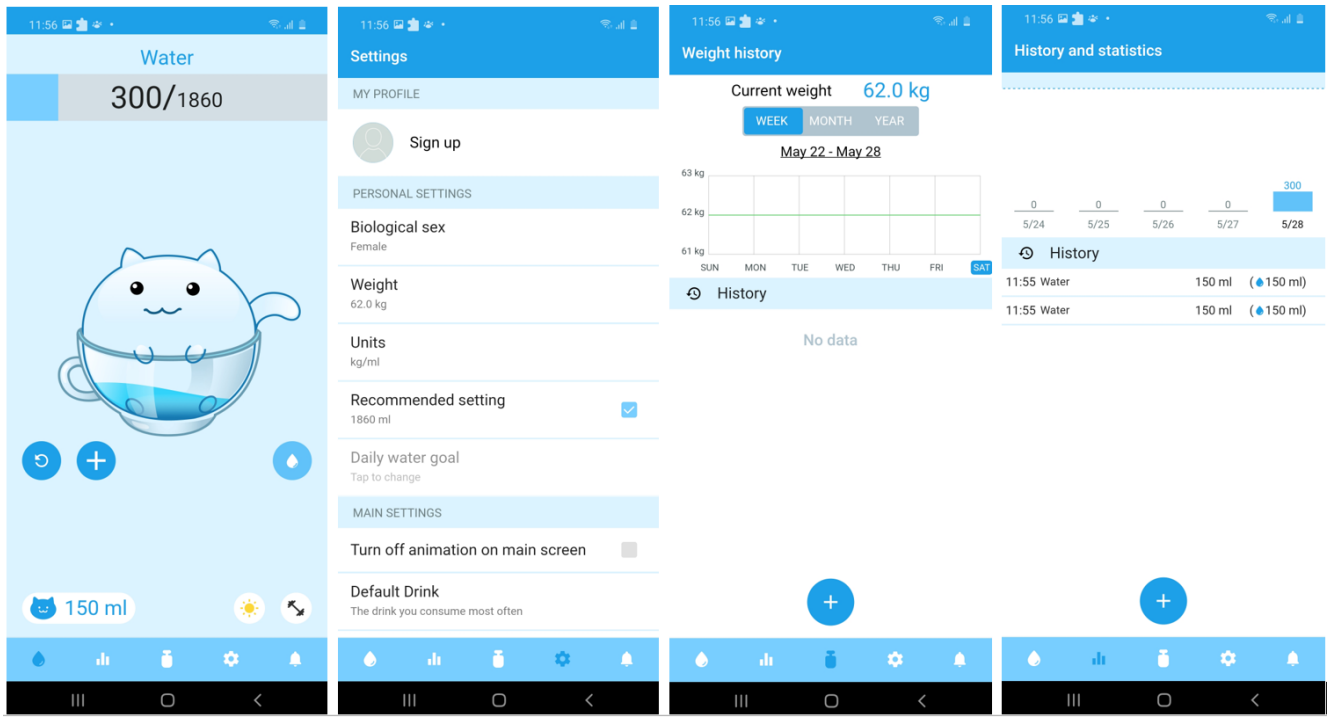

Рисунок 1.4 — Інтерфейс застосунку «Водокотик»

Мобільний застосунок «Aqualert». Розраховує денну норму питної води на основі ваших фізичних параметрів, активності, певних фізіологічних процесів та погоди. Наприклад, ви можете відзначити в налаштуваннях, що вагітні або годуєте грудьми. Або що сьогодні було страшенно спекотно — все це враховуватиметься при розрахунку норми.

Функції:

– калькулятор щоденного прийому води, який враховує стать, вагу та рівень активності;

– повідомлення та нагадування, коли пити воду;

– автоматичний режим сну, в якому нагадування не турбуватимуть посеред ночі;

– графічне зображення балансу води у вашому організмі та щоденного споживання;

– розраховує, скільки порцій води залишилося випити;

– легко додати та видалити порцію;

– історія споживання у вигляді таблиці;

– автоматичні нагадування про те, чому варто пити воду і коли ви п'єте більше, ніж потрібно.

Переваги:

– багатофункціональність;

– підходить для більшості смартфонів;

– не потребує підключення до Інтернету;

– повідомлення про те, коли саме потрібно випити воду;

– можна вказати особливості свого фізичного стану (вагітність тощо) і програма врахує це при розрахунку норми води;

– синхронізація із Google Fit;

– наявність автоматизації.

Недоліки:

– наявність реклами;

– незручний, поганий інтерфейс;

– немає здатності коригування норми води залежно від погодних умов та фізичної активності.

Приклади інтерфейсу застосунку зображені на рис. 1.5.

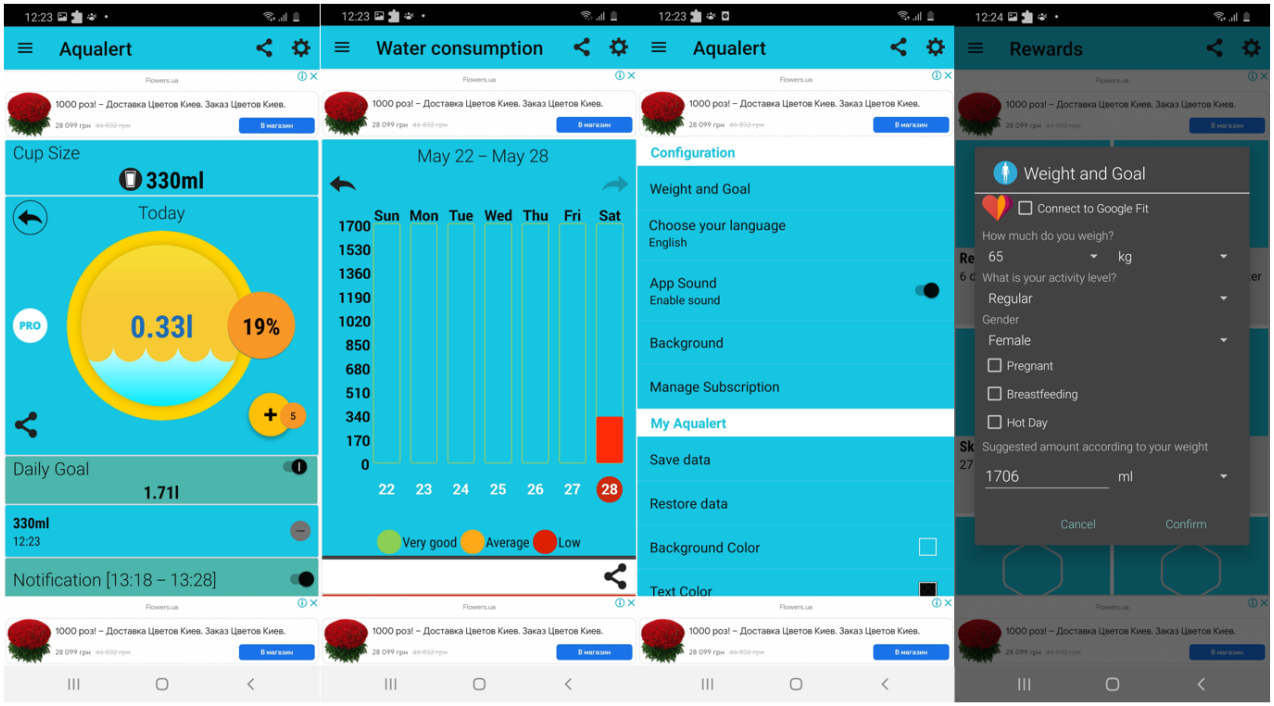

Рисунок 1.5 — Інтерфейс застосунку «Aqualert»

Порівняння оглянутих рішень. Оглянуті рішення порівняно за такими критеріями, як багатофункціональність, зручність використання, автоматизація, необхідність підключення до Інтернету та коригування норми води в залежності від фізичної активності. Було створено таблицю порівняння оглянутих рішень (таблиця 1.1).

Таблиця 1.1 — Порівняння існуючих рішень

Критерій / Рішення	Необхідність підключення до Інтернету	Автоматизація	Багато-функціональність	Зручність	Коригування норми води в залежності від фізичної активності
Мобільний застосунок «Моя вода»	–	+	+	–	–
Мобільний застосунок «BeWet»	–	+	+	+	+
Мобільний застосунок «Water Drink»	–	+	+	–	+
Мобільний застосунок «Водокіт»	–	+	+	+	+
Мобільний застосунок «Aqualert»	–	+	+	–	–

1.3 Постановка задачі

У результаті огляду та аналізу існуючих рішень було виявлено, що існуючі мобільні застосунки мають як переваги, так і недоліки. Тому було вирішено створити власний застосунок, при розробці якого будуть враховані плюси та мінуси конкурентів, який буде нагадувати користувачеві, коли саме потрібно пити воду, вираховуватиме необхідну щоденну норму води для кожної людини, виходячи з її індивідуальних параметрів, буде показувати загальну оцінку самопочуття людини на основі показників випитої кількості води людиною впродовж певного проміжку часу. А також матиме простий та зручний інтерфейс, а найголовніше — буде актуальним для тих, хто хоче покращити стан свого здоров’я завдяки щоденному контролю за кількістю випитої рідини.

Тому метою бакалаврської кваліфікаційної роботи є покращення водного балансу та самопочуття користувачів за рахунок систематичного вживання необхідної норми води, шляхом розробки мобільного застосунку на базі операційної системи Android.

Розробка мобільного застосунку підтримки водного балансу в організмі людини буде реалізована мовою програмування Kotlin із використання програмного середовища Android Studio. Розроблений застосунок буде нагадувати користувачеві, коли саме потрібно пити воду, вираховуватиме необхідну щоденну норму води для кожної людини, виходячи з індивідуальних параметрів, а також буде показувати загальну оцінку самопочуття людини на основі показників випитої кількості води впродовж доби завдяки реалізації моделі лінійної регресії на мові програмування Python із використанням технології TensorFlow та TensorFlow Lite.

Список функціональних вимог:

– автоматизація розрахунку щоденного прийому води, який враховуватиме вагу людини та час фізичної активності;

– автоматизація виведення повідомлень, коли пити воду, враховуючи режим дня користувача;

– реалізація кругової шкали прогресу для контролю балансу води в організмі;

– реалізація легкого додавання порцій води;

– реалізація виведення прогнозованої оцінки фізичного самопочуття людини шляхом створення простої моделі лінійної регресії.

Список нефункціональних вимог:

– версія Android 6.0 та вище;

– англійська версія користувацького інтерфейсу;

– застосунок повинен мати зручний та зрозумілий інтерфейс.

Висновки до розділу 1

Розглянуто, що таке водний баланс, яким чином його підтримувати та які наслідки за собою веде недостатнє споживання рідини людиною. Також було описано предметну сферу застосунків для контролю кількості випитої води. Проведено аналіз ринку мобільних застосунків для підтримки водного балансу в організмі людини. Виявлено відповідні переваги та недоліки у вже існуючих програмних застосунках та створено таблицю порівняння оглянутих рішень за такими критеріями, як багатофункціональність, зручність використання, автоматизація, необхідність підключення до Інтернету та коригування норми води в залежності від фізичної активності.

У ході аналізу оглянутих рішень можна підсумувати:

– оглянуті мобільні застосунки є продуктами, які мають багатофункціональну реалізацію із автоматизованим коригуванням підрахунку щоденної норми води та графіками, які відображають статистику випитої кількості води користувачем протягом певного проміжку часу;

– застосунки майже в повному обсязі виконують всі поставлені задачі та функції без підключення до Інтернету;

– значна частина розглянутих застосунків незручна у використанні.

Сформовано постановку задачі для розроблювального застосунку. Наведено список функціональних та нефункціональних вимог для майбутнього продукту.

2 ТЕХНОЛОГІЇ ТА МЕТОДИ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ

2.1 Типи мобільних застосунків

Розробка мобільних програм — це процес проєктування, а також створення мобільних застосунків для пристроїв, наприклад для смартфонів, планшетів тощо. Розробники мобільного програмного забезпечення мають також враховувати дуже широкий спектр конфігурацій та технічних характеристик внаслідок значної конкуренції в розробці програмного забезпечення, а також їх зміни на різних платформах [3]. Проєктування мобільних застосунків постійно зростає, також зростають доходи та кількість робочих місць серед розробників.

Проєктування мобільних програм є все більш важливим для більшості компаній. Адже люди проводять приблизно 90% власного часу за мобільним телефоном.

В процесі розробки мобільного застосунку треба обрати тип для, який буде доцільно застосовувати при розробці продукту.

Якщо говорити про типи мобільних застосунків, то всі вони поділяються на 3 категорії: нативні, веб та гібридні (рис. 2.1).

Рисунок 2.1 — Типи мобільних застосунків

Нативний застосунок — це застосунок, який розроблений для якоїсь конкретної платформи, наприклад, iOS чи Android. При розробці нативного застосунку враховується специфіка тої чи іншої платформи.

Переваги нативних застосунків.

1. Їх легко освоїти. Інтерфейс користувача та графічний компонент рідних програм дотримуються ідеології дизайну, яка вбудована в операційну систему. Розташування анімації, палітри кольорів, а також елементів керування сприймається природним. Тому освоїти застосунок такого типу досить легко, і його використання не викличе дисонансу з наявним досвідом користувача.

2. Висока швидкість. Вбудовані програми оптимізовані для конкретної мобільної операційної системи, що робить їх швидкими та надзвичайно стабільними. Крім того, дані таких застосунків переважно зберігаються на пристрої, що також збільшує їх швидкість і зменшує залежність від Інтернету.

3. Широкі можливості. Завдяки доступу до можливостей операційної системи, нативні застосунки досить просто інтегруються між собою.

Вебзастосунок — це застосунок, який розроблений на JavaScript, HTML та CSS, для виконання такої програми треба встановлений і налаштований браузер мобільного пристрою із виходом в Інтернет.

Переваги вебзастосунків.

1. Здатність до повторного використання коду на різноманітних платформах.

2. Не має значних вимог до використання апаратних засобів, або до графіки.

3. Також мають великий вибір різних фреймворків, інструментів, які спрощують процес розробки.

Гібридні застосунки — це універсальні застосунки, які створено для багатьох платформ одразу.

Можна перефразувати, що гібридний застосунок – вебсайт у звичному форматі мобільного забезпечення.

Переваги гібридних застосунків.

1. Універсальність. Такі застосунки призначені для декількох платформ, тому вони просто адаптуються під кожну. Гібридні застосунки націлені на значно більшу аудиторію потенційних користувачів.

2. Менша вартість розробки. Такий тип застосунків простіший під час розробки, тому їх собівартість менша, ніж собівартість у нативного забезпечення. Ця особливість робить застосунки такого типу гарним рішенням для багатьох компаній, які мають незначний стартовий бюджет.

3. Також досить швидко виходять на ринок. Проста розробка гібридних застоунків надає ще таку особливість, як термін розробки, саме це дозволяє компаніям відносно швидко випустити власний продукт і також отримати прибуток.

Отже, для досягнення поставленої мети доцільна розробка нативного мобільного застосунку, за рахунок вищенаведених переваг даного типу.

2.2 Огляд сучасних мобільних операційних систем

На даний час існують багато різних мобільних операційних систем (ОС), але всі вони мають різну популярність. Якщо застосунок не кросплатформовий, то розробка для вузького кола користувачів може заздалегідь зробити його провальним. Тому що чим більше користувачів будуть працювати з програмою, тим більше буде зацікавленості в ній. Тому важливо вибрати операційну систему для мобільного застосунку.

На сьогодні, можна виокремити дві найбільш поширені мобільні ОС, на які припадає понад 99% ринку, такі як iOS та Android.

iOS — операційна система, яка була розроблена Apple Inc. для своїх пристроїв iPhone, iPod Touch та iPad. Перша версія iOS була випущена 12 жовтня 2007 року. У порівнянні з іншими платформами, такими як ОС Android або Windows Phone, які надають більше можливостей налаштування та доступність

магазину сторонніх програм, iOS обмежує свободу користувачів на користь централізованого контролю з боку Apple.

Використання операційної системи iOS має багато переваг. По-перше, дуже зручна у використанні. Крім того, що система проста у використанні, вона також ефективна і швидка. Це робить її ідеальною для мобільних пристроїв, таких як смартфони та планшети. Крім того, безпечна. Це пояснюється тим, що вона використовує ряд функцій безпеки, таких як паролі та шифрування, для підтримки конфіденційності користувачів. Ще одна перевага iOS полягає в тому, що вона сумісна з широким спектром пристроїв. Це означає, що її можна використовувати на різних пристроях, таких як смартфони, планшети, комп'ютери та навіть комп'ютери Mac. Нарешті, iOS доступна за ціною. Це пояснюється тим, що вона використовує прості та ефективні методи керування даними користувачів. Все це робить її привабливим варіантом для користувачів.

Однак є і недоліки, пов'язані з iOS. Одним з таких недоліків є те, що вона несумісна з широким спектром пристроїв. Це означає, що вона не підходить для використання на ряді пристроїв. Крім того, вона не доступна різними мовами. Нарешті, вона не зручна для людей, які не знайомі з технологіями.

Але все ж iOS має багато переваг, зокрема зручність, ефективність і швидкість.

Android — це мобільна операційна система, що належить компанії Google та розроблена базуючись на ядрі Linux. На відміну від iOS, Android є відкритою системою. Android – це дуже популярна мобільна операційна система. Android був задуманий Енді Рубіном, який розробляв його в Google з 2002 по 2005 рік як дослідницький проєкт. Вона конкурувала з iOS, яка була створена Скоттом Форстолом і Джонатаном Айвом з Apple Inc. Розробка цих двох систем почалася окремо, але в кінцевому підсумку була дуже схожою через їхню природу з відкритим кодом. iOS виграла, оскільки її швидше засвоїли великі виробники телефонів, а магазин застосунків дозволив розробникам швидко заробляти гроші.

Android і iOS є двома найпопулярнішими операційними системами у світі. Обидва вони мають свої унікальні особливості та переваги, які роблять їх ідеальними для різних цілей.

Android популярний для мобільних пристроїв, таких як телефони та планшети. Це багатозадачна операційна система, що означає, що користувач може запускати кілька програм одночасно. Це корисно для людей, які хочуть використовувати свій телефон для кількох завдань. iOS в основному використовується на iPhone та iPad. Це закрита система, що означає, що користувачі обмежені продуктами, доступними в App Store. Це корисно для людей, яким потрібна конкретна програма, а не різноманітні варіанти.

Важливо персоналізувати операційну систему відповідно до своїх потреб. Якщо людині потрібна універсальна система, їй варто вибрати Android. Якщо вони хочуть систему, яка обмежена з точки зору вибору застосунків, вони повинні вибрати iOS.

Після проведення аналізу найбільш популярних мобільних операційних ситсема було вирішено, що **розробка буде виконуватися під пристрої з ОС Android**, тому що вона найбільш відповідає заявленим вимогам.

2.3 Середовище розробки Android Studio

Щоб створити програму для ОС Android, необхідно вибрати середовище розробки. На даний момент існує кілька популярних середовищ розробки — IDE (Integrated Development Enviroment) під платформу Android: Android Studio, IntelliJ IDEA, Eclipse, AIDE і фреймфорк Xamarin в Visual Studio.

Для розробки програми було обрано Android Studio. Android Studio — це потужне та універсальне середовище розробки, яке допомагає розробникам створювати високоякісні програми для Android. Завдяки інтуїтивно зрозумілому інтерфейсу користувача, надійним функціям та широкому набору інструментів і плагінів Android Studio робить розробку мобільних програм простішою, ніж будь-коли. Які ж переваги використання Android Studio як основної платформи

для розробки застосунків Android. Перш за все, за допомогою Android Studio можна швидко й легко розробляти програми для Android, практично не потребуючи попереднього досвіду програмування. Крім того, витончений дизайн та інтуїтивно зрозумілий макет інтерфейсу розробника програми спрощують навігацію в процесах створення програми без додаткового навчання чи допомоги експерта-дизайнера.

Android Studio має багато переваг, головна з яких — простота використання. Розробники без значного досвіду можуть швидко освоїти основи роботи із цим середовищем, дотримуючись початкових покрокових інструкцій, а досвідчені розробники оцінять гнучкість і розширюваність функцій Android Studio. Вони включають вбудовану підтримку версій і розгалуження, комплексний інструмент аналізу коду, а також різноманітні рефакторинги та перевірки коду, які допомагають виявляти та виправляти помилки.

Розпочати роботу з Android Studio легко. Спочатку треба встановити програмне забезпечення Android Studio на комп’ютер. Потім дотримуватись початкових інструкцій, щоб дізнатися про основи розробки Android. Отримавши базове розуміння мови програмування Android та основ розробки Android, можна починати працювати над проєктом.

2.4 Вибір мови програмування

Вибрана IDE підтримує роботу з двома мовами для програмування Android програм, а саме — Java та Kotlin.

Java і Kotlin — дві дуже популярні мови програмування, які користуються великою популярністю в останні роки. Обидві вони мають свій набір переваг і недоліків, які слід враховувати, перш ніж приймати рішення щодо того, використовувати їх чи ні. Java відома своєю надійністю, тоді як Kotlin вважається більш лаконічним. Однак Java критикують за повільність порівняно з новими мовами, такими як Python або Javascript, а Kotlin страждає від обмеженої підтримки на різних платформах, включаючи Android, iOS та Windows.

Java є найпопулярнішою і широко використовуваною мовою програмування у світі. Вона має довгу і багату спадщину, яка сягає 1995 року. Популярність Java пояснюється її широкою екосистемою бібліотек та інструментів, що робить її дуже універсальною. Іншою причиною популярності Java є її продуктивність. Java швидка, потужна та надійна.

Одним з основних недоліків Java є те, що новачкам може бути важко вчитися. Синтаксис багатослівний, і на перший погляд код може здатися складним. Однак, після того, як звикнути до синтаксису Java, писати код стає дуже легко. Java також відома своєю надійністю. Він має довгу історію використання в критично важливих програмах.

Kotlin — це нова мова програмування, яка була розроблена JetBrains у 2017 році. Kotlin розроблено, щоб зробити програмування на Java більш стислим і легким для читання. Kotlin також має багатоплатформну сумісність і працює на платформах Java і Android. Kotlin оптимізовано для швидкості, що робить його хорошим вибором для розробки власних мобільних застосунків [4].

Під час виконання бакалаврської кваліфікаційної роботи для розробки застосунку **було вибрано мову програмування Kotlin**. Адже ця мова має багато переваг.

1. Лаконічність. Це є одним із основних факторів, через які чимало розробників використовує саме Kotlin.

2. Kotlin null-безпечний. Винахідники мови зробили все, аби NullPointerException залишилися в минулому.

3. Функції-розширення. Kotlin дозволяє змінити функціонал існуючих класів без наслідування цих класів. Це дуже зручний інструмент, адже він дозволяє підвищити читабельність коду.

4. Kotlin має відкритий код. Код Kotlin відкритий для розробників, і його впровадження у проєкт також є безкоштовним. Відкритий вихідний код полегшує пошук проблем. Розробники Kotlin прислухаються до інших розробників і вносять корегування, які запропоновані спільнотою.

5. Легко вчити. Багато хто говорить, що Kotlin досить простий у вивченні та що це мова програмування для початківців.

Як мова побудови макетів програми буде використовуватись XML. XML— це мова опису даних, яка відома як мова розмітки. У програмах під платформу Android інтерфейс (макети) завантажується з файлів з розширенням .xml. А ще використовується для роботи у Web. У нього є ряд переваг, а саме: легко читається людиною або комп'ютером, при передачі з одного на інший комп'ютер не виникне ніяких проблем, тому що зберігається у простому текстовому файлі, а ще не фіксує розмітку, тобто дозволяє створювати її самим залежно від особливостей області, а також від потреб.

2.5 Метод лінійної регресії в машинному навчанні

Під час розробки мобільного застосунку також буде використовуватись метод лінійної регресії для представлення залежності самопочуття людини від щоденної випитої норми води та інших індивідуальних показників. То що ж це за метод та для чого він використовується?

У статистиці та машинному навчанні лінійна регресія — це контрольований метод навчання, який використовується для аналізу зв’язку між вхідними змінними (x) і вихідними змінними (y). Мета лінійної регресії полягає в тому, щоб знайти математичну функцію, яка найкраще прогнозує y від x, використовуючи лише відомі вхідні дані. Лінійні регресії часто використовуються в аналізі часових рядів, де ми хочемо зрозуміти, як зміни x впливають на зміни y з часом [5].

Якщо розглядається залежність між однією вхідною та однією вихідною змінними, має місце проста лінійна регресія. Для цього визначається рівняння регресії та будується відповідна пряма, вона відома як лінія регресії (рис. 2.2).

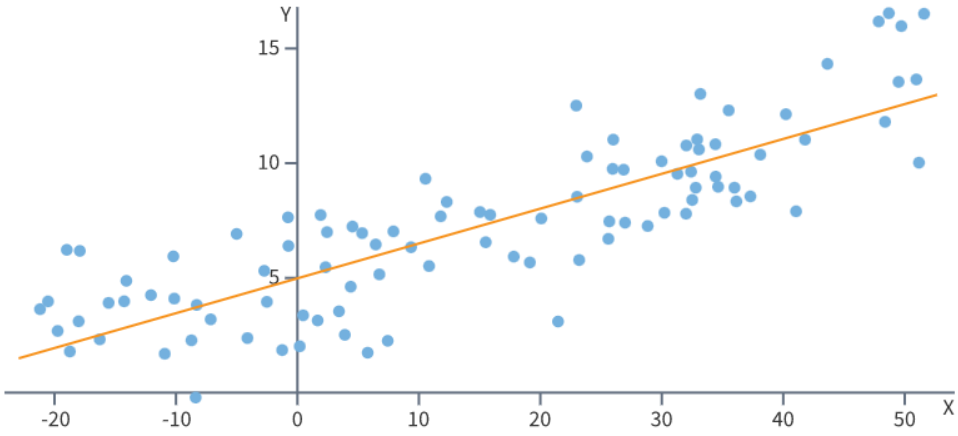

Рисунок 2.2 — Лінійна регресія

Лінійна регресія є широко використовуваною технікою машинного навчання, яка використовується для досягнення бажаного результату [6]. Це проста математична функція, яка допомагає дослідникам передбачити майбутні значення однієї або кількох змінних на основі минулих значень. Для обчислення лінійної регресії використовується спрощене рівняння 2.1.

$$y = a + b \times x, \tag{2.1}$$

де y – прогнозовані результати;

a і b – коефіцієнти рівняння регресії;

x – вхідні змінні рівняння.

Однією з причин використання лінійної регресії в машинному навчанні є те, що це універсальний інструмент. На відміну від деяких інших моделей машинного навчання, лінійна регресія здатна відповідати широкому діапазону розподілів даних. Це робить її хорошим вибором для моделювання нелінійних зв’язків. Крім того, модель лінійної регресії відносно легко створити та налаштувати. Якщо прогноз занадто низький або високий, можна налаштувати коефіцієнти рівняння регресії таким чином, щоб отримати кращий прогноз.

2.6 TensorFlow як технологія для вивчення штучного інтелекту

TensorFlow — це бібліотека програмного забезпечення з відкритим вихідним кодом для програмування потоків даних, орієнтована на паралельні обчислення на CPU та GPU. Спочатку TensorFlow був розроблений дослідниками Google Brain, підрозділу Alphabet Inc., в основному для досліджень машинного навчання, але з тих пір він використовувався для розробки інших застосунків, таких як чат-боти, самокеровані автомобілі, обробка природної мови тощо. Використання TensorFlow продовжує швидко розширюватися, тому що філософія дизайну, що лежить в основі бібліотеки, дозволяє легко впроваджувати нові функції, зберігаючи при цьому старі функції належним чином. Зараз дослідники використовують TensorFlow у дослідженнях штучного інтелекту (AI), комп'ютерного зору (CV) та глибокого навчання (DL).

TensorFlow можна використовувати для навчання моделей штучного інтелекту [7]. Його також можна використовувати для попередньої обробки та аналізу даних перед подачею їх у моделі AI [8]. Ця попередня обробка може включати такі речі, як зменшення шуму, нормалізація даних та виправлення зміщення. TensorFlow має багато потенційних застосувань, включаючи навчання та попередню обробку даних, оцінку продуктивності та підвищення точності.

Існує також TensorFlow Lite — просте рішення TensorFlow для мобільних і вбудованих пристроїв. Це дозволяє дуже швидко і з невеликим двійковим розміром розгортати на пристроях користувачів рішення, орієнтовані на ML, але підтримує обмежену кількість систем. Апаратне прискорення також підтримується через API нейронних мереж Android.

Після вивчення моделі зазвичай потрібно її протестувати. На етапі розробки це можна зробити через CLI (інтерфейс командного рядка). Однак, якщо вам потрібно перейти до фази розгортання, вам потрібно запустити модель в Інтернеті та Android. Для Інтернету ви можете запускати моделі за допомогою

Tensorflow.js або за допомогою фреймворків Flask і Django. Коли справа доходить до Android, було б важко запустити моделі на Android, оскільки зазвичай для цього потрібно більше оперативної пам’яті та багато обмежень. Для цього Google розробляє міні-API під назвою TensorFlow-Lite. За допомогою Tensorflow Lite API ви можете надати модель ML для будь-якого застосунку Android. Архітектуру роботи TensorFlow Lite можна побачити на рис. 2.3.

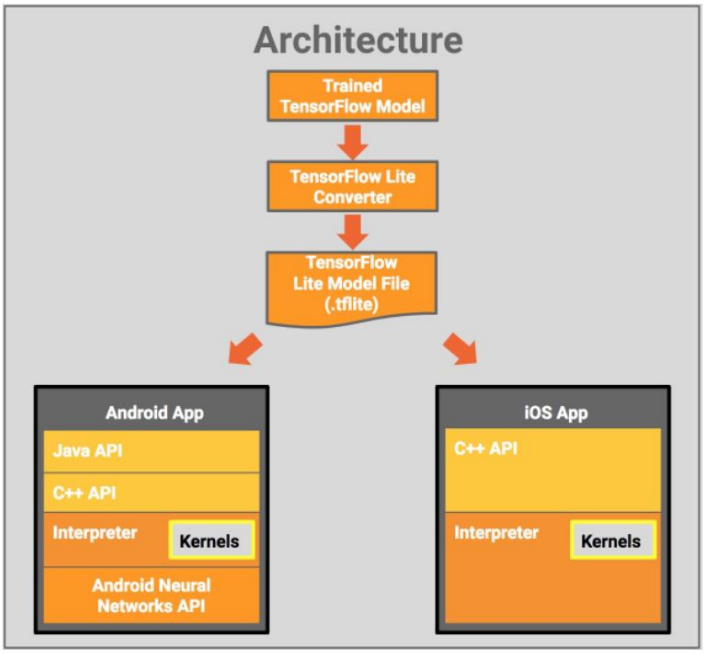

Рисунок 2.3 — Архітектура роботи TensorFlow Lite

Архітектура роботи включає 4 етапи.

1. Навчання, а потім збереження моделі Tensorflow. Спочатку потрібно навчити модель за допомогою фреймворку Keras, а також треба зберегти її у форматі .H5 або .PB, аби завантажувати модель в будь-який час.

2. Створення конвертора TensorFlow Lite. Конвертер TensorFlow Lite – це інструмент, доступний у вигляді API Python, який перетворює навчені моделі TensorFlow у формат TensorFlow Lite. Також він може ввести оптимізацію.

3. Перетворення TensorFlow Lite Converter у формат .tflite. Необхідно перетворити цей об’єкт у формат tflite, який надалі використовуватиметься в програмі для Android.

4. Розгортання .tflite в Android Studio, а потім виконання висновку. Тепер ми будемо використовувати Tensorflow Interpreter API в студії Android для запуску моделі .tflite з даними для отримання результатів.

Саме так можна запускати моделі ML за допомогою Tensorflow Lite API, виконавши етапи, які наведені вище.

Висновки до розділу 2

Проаналізувавши наведені технології та методи для реалізації поставленої задачі, обрано і затверджено наступний стек програмних засобів реалізації і зроблено такі висновки:

– розглянуто та обрано тип для мобільного застосунку, а саме — нативний, за рахунок переваг даного типу;

– розглянуто і обрано мову програмування Kotlin, її синтаксис, методи програмування на цій мові;

– розглянуто Android OS, як цільову платформу для розробки;

– обрано інтегроване середовище розробки Android Studio, яке допоможе зручно та швидко створити інтерфейс для мобільного застосунку. Було вказано функції та переваги цієї IDE;

– розглянуто метод лінійної регресії та технологію TensorFlow, а також архітектуру роботи TensorFlow Lite.

3 ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ. РОЗРОБКА ДИЗАЙНУ

3.1 Структура мобільного застосунку

Проєктування — це дуже важливий процес під час створення мобільного застосунку [9]. Адже перше враження про продукт залежить від того, наскільки він зручний у використанні та звичайно ж від зовнішнього вигляду. Із досліджень відомо, що більше ніж 75% людей завантажують застосунок, відкривають його лише раз і після цього не повертаються до нього. Аби створити дійсно унікальний продукт, який буде користуватися попитом серед потенційних користувачів, треба пам’ятати про важливі моменти в процесі його проєктування та не нехтуватими ними.

Важливими етапами проєктування мобільних застосунків є створення структури та прототипування інтерфейсів.

Візуалізація загальної структури розроблювального продукту відіграє значну роль в проєктуванні мобільних застосунків. На рисунку 3.1 показано діаграму, яка відображає доступні дії користувача під час роботи із системою та алгоритм використання інформаційної системи.

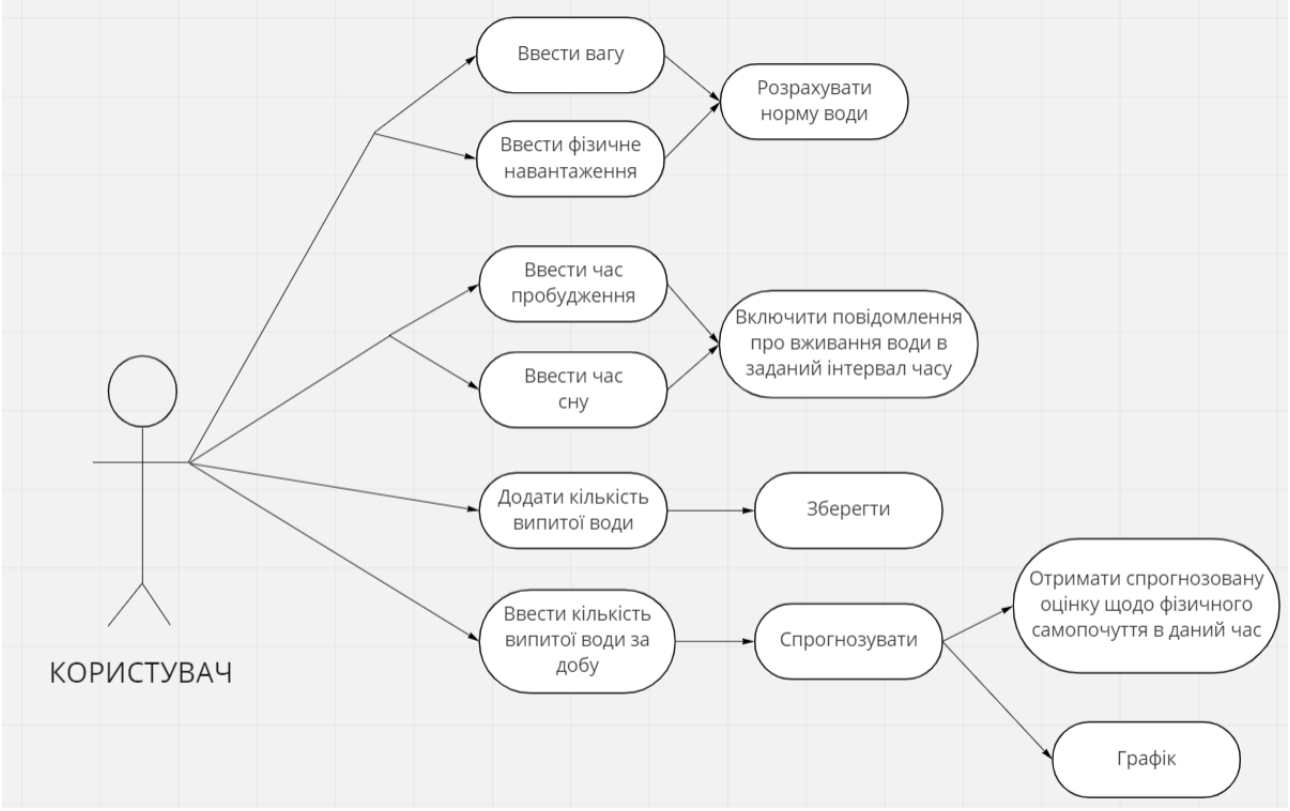

Рисунок 3.1 – Діаграма використання мобільного звстосунку

Структурна діаграма мобільного застосунку описує внутрішню будову модулів з яких складається дана інформаційна система та показує зв’язок між цими модулями. Створену діаграму зображено на рисунку 3.2.

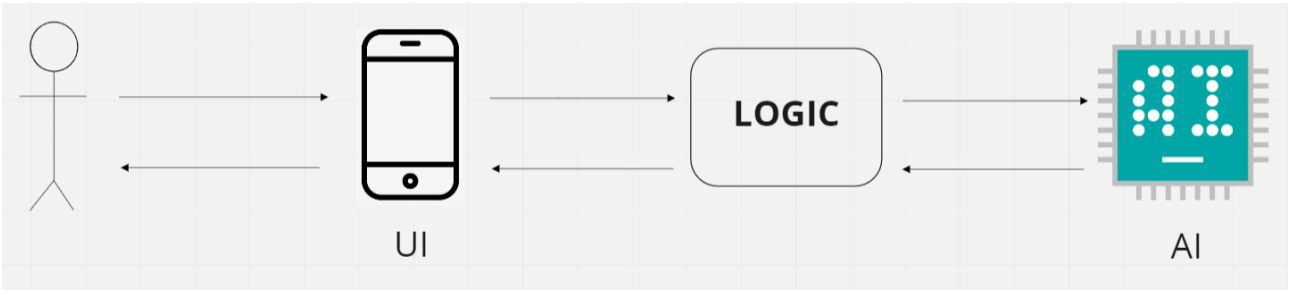

Рисунок 3.2 – Структурна діаграма мобільного застосунку

3.2 Прототипування мобільного застосунку

Прототипування є одним з ключових етапів розробки мобільних застосунків. Прототип — це модель, яка описує ключові функції та концепцію інформаційної системи.

Етап прототипування починається після визначення потенційних користувачів майбутнього продукту, який розробляється, тобто визначення цільової аудиторії. А також після формування проблем, які застосунок буде вирішувати. Саме такий підхід визначає основний функціонал та концепцію програмного забезпечення, знижує ризики та дозволяє зберегти бюджет та дорогоцінний час розробників, який вони б витратили на реалізацію неперевірених на практиці рішень.

Залежно від задач, які мають вирішувати прототипи, їх поділяють на різні типи, а саме: концептуальні, інтерактивні, каркасні, анімовані та макетні.

Концептуальний прототип — це схематичне зображення майбутніх екранів, що створюється на початкових етапах розробки мобільного застосунку. При створенні нового інтерфейсу програми потрібно завжди створювати концептуальний прототип. Адже саме цей метод допомагає вирішити чимало питань, які стосуються зручності використання майбутнього застосунку. Також цей тип прототипування добре підходить для того, аби протестувати ідеї, оскільки основні елементи екранів можна намалювати за досить короткий час.

Концептуальне прототипування дуже важливо, коли справа доходить до перенесення на екрани застосунків користувацьких сценаріїв. Саме завдяки цьому, розроблювальний продукт стає прототипом кінцевого результату.

Інтерактивний прототип створюється на основі екранів, які є результатом етапу створення концептуального прототипу. Таким чином, прототип виглядає досить реалістичним і підходить для тестування на кінцевих користувачах.

Цей тип прототипування слід використовувати саме в тому випадку, якщо треба змоделювати користувацький сценарій, для прикладу це може бути реєстрація в застосунку. Або ж потрібно показати команді розробників, на якому саме етапі реалізації знаходиться продукт чи коротко продемонструвати керівництву, над чим саме зараз триває робота. Цей спосіб може добре вплинути на оцінку менеджерів та зарахується як плюс для всієї команди.

Каркасний прототип відображає екрани майбутнього продукту, які містять ключові елементи інтерфейсу, а саме: значки, поля, кнопки. Цей тип прототипування надає візуальну індикацію програми, показуючи найважливіші деталі, розташування та розміри елементів.

Каркаси працюють аналогічно із цифровим кресленням продукту. Вони показують основні структури застосунків, в тому числі і макети, ключові сторінки, потоки сторінок, форми та архітектури. Каркаси дозволяють зосередитися саме на елементах макро інтерфейсу, ігноруючи дрібні деталі.

Анімований прототип — це прототип на найвищому рівні. Адже рух – це можливість втілити життя в інтерфейс. Серед прототипів такого типу є такі, що здатні досить точно імітувати роботу реального застосунку таким чином, що розрізнити такий прототип від готового продукту досить важко. Основну перевагу реалізації анімованого прототипу видно з його назви. На цій стадії розробки дизайнер думає про дужевагомий аспект в UX дизайні, а саме про взаємодію людини із інформаційною системою, візуалізація якої здійснюється за допомогою анімації.

Створення анімованого прототипу вимагає знання спеціальних інструментів, які націлені на роботу з анімацією. Чимало із цих програм здатні підтримувати пошаровий імпорт із векторного графічного редактору Sketch для macOS, саме це забезпечує роботу не з екранами в цілому, а з окремими елементами. Такий прототип можна подивитися на мобільному пристрої. Створення анімованого прототипу займає більше часу, який витрачається на продумування деталей і реалізації, але результат отримується бездоганним.

Макетний прототип майже повністю відображає майбутній застосунок, але не включає в себе інтерактивні дії. Можна сказати, що макети — це статичні каркаси, які включають в себе точну деталізацію візуальних деталей та інтерфейсу — текст, кнопки, кольори, зображення. Макети мобільних застосунків домогають дизайнерам досить легко тестувати різноманітні візуальні деталі.

Протягом проєктування застосунку створюють і змішаний тип прототипу, який також дозволяє досягнути основної мети у прототипуванні.

Під час виконання бакалаврської кваліфікаційної роботи створено концептуальний прототип на папері для того, щоб структурувати всі ідеї, які плануються реалізовуватися в інформаційній системі. Це допомогло візуалізувати значну частину функціоналу та проаналізувати, яким чином користувач буде взаємодіяти із мобільним застосунком.

Після цього було розроблено прототип змішаного типу, який включає в собі ознаки інтерактивного та каскадного прототипів та концепцію UX дизайну. Створений прототип можна побачити на рисунках 3.3, 3.4.

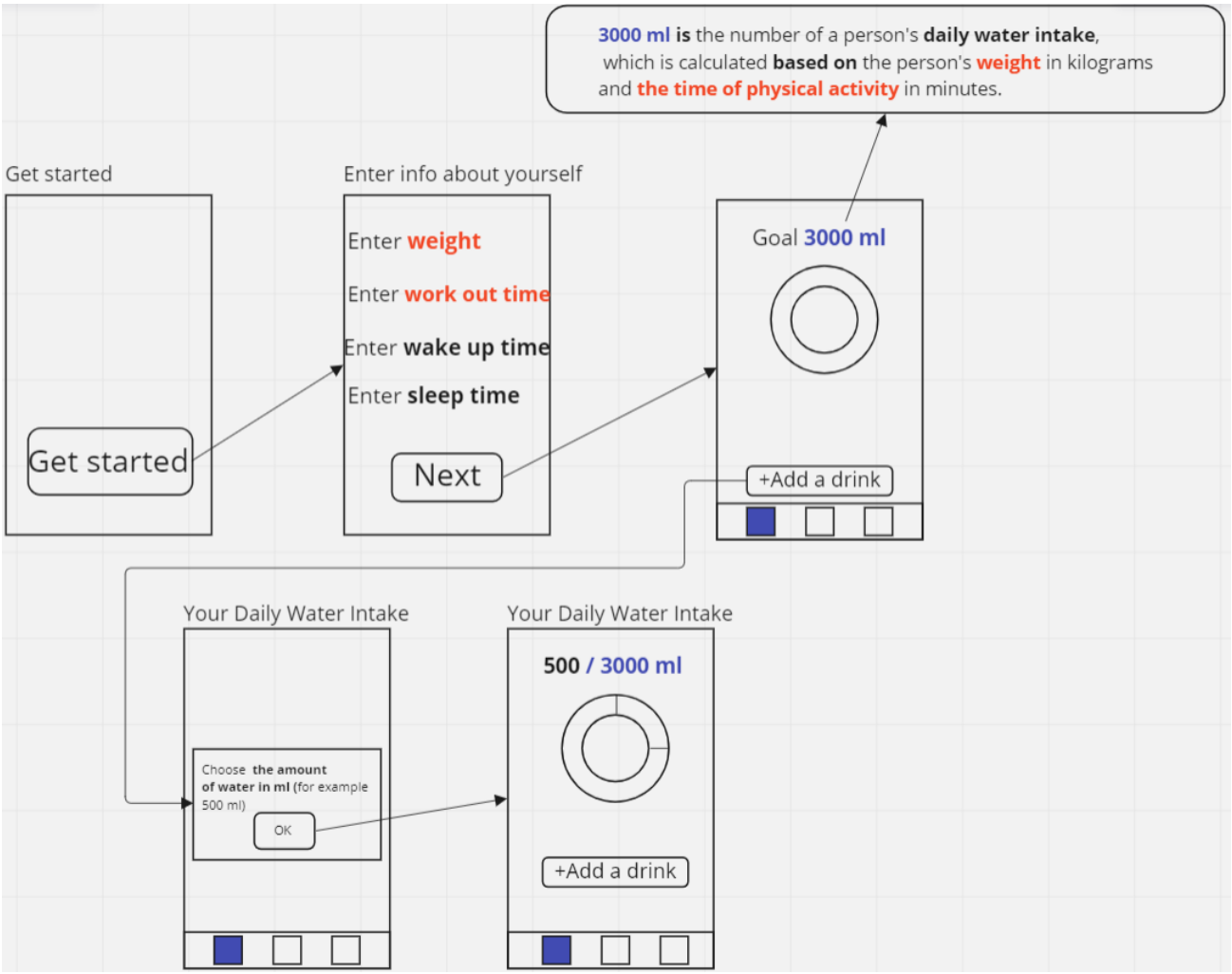

Рисунок 3.3 — Прототип мобільного застосунку

Як показано на даному прототипі, взаємодія користувача із мобільним застосунком починається з показу екрану старту, за допомогою якого користувач переходить на екран введення потрібної інформації про себе для подальших розрахунків. Після введення необхідних даних з’являється головний екран, де розташована кругова шкала прогресу та інформація про те, яку норму води користувачу потрібно випивати за добу. Також на головному екрані знаходиться кнопка додавання порції води та нижня панель навігації, за рахунок якої можливий перехід на інші екрани застосунку.

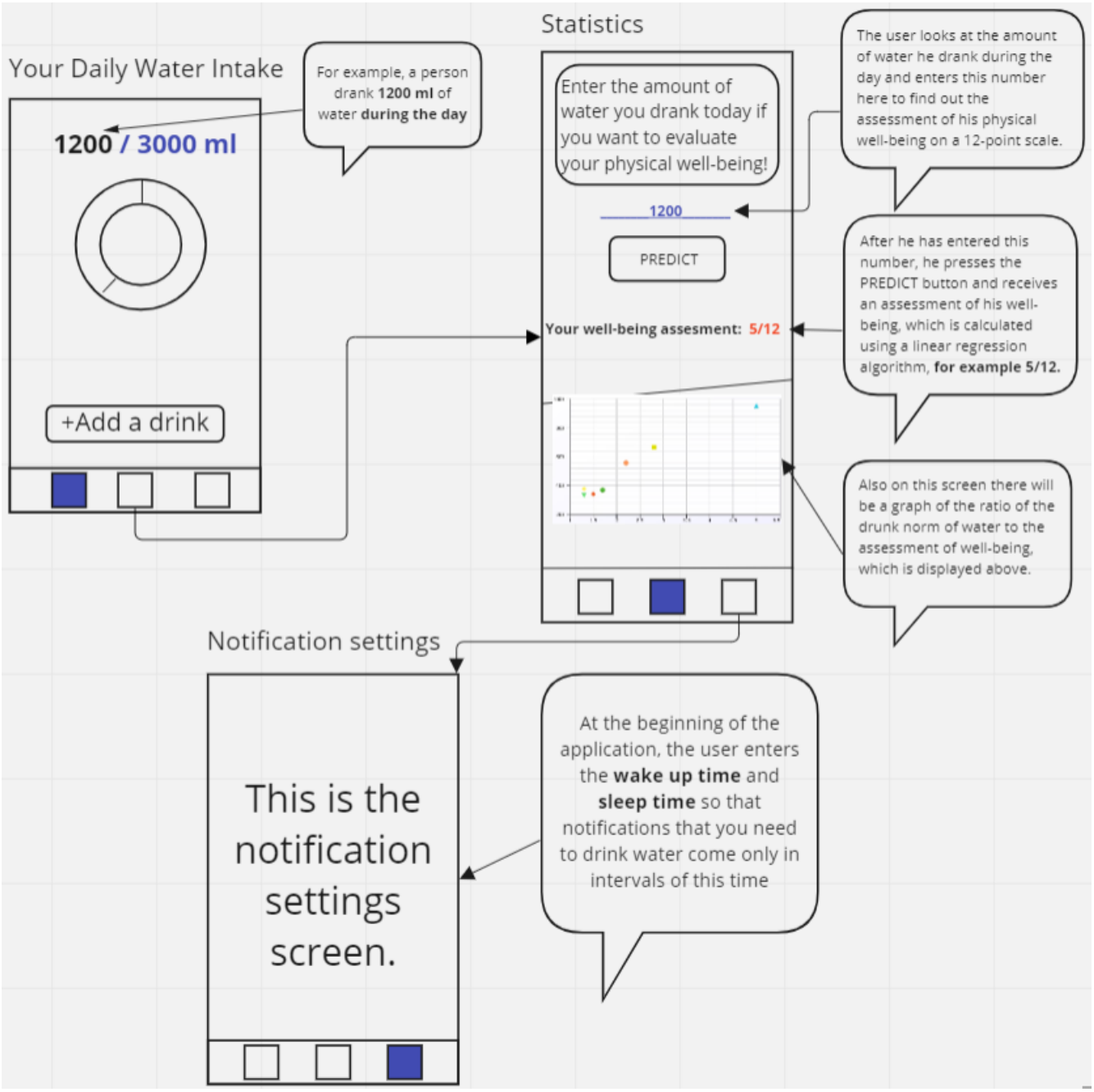

Рисунок 3.4 — Прототип мобільного застосунку

Також на прототипі зображено перехід з головного екрану на екран статистики та прогнозування. Після введення кількості води, яку людина випила протягом доби, користувач має натиснути кнопку «Predict», після чого на екрані з’явиться прогнозована оцінка фізичного самопочуття людини за дванадцятибальною шкалою, а також лінійна діаграма, яка відображає лінійну залежність випитої кількості людиною води за добу до оцінки її самопочуття.

Ще на прототипі показано перехід із екрану прогнозування на екран налаштування сповіщень, які будуть приходити користувачу в межах того інтервалу часу, який він задасть в полях індивідуального часу пробудження та часу сну.

Після створення загальної структури та стадії прототипування застосунку можна перейти до розробки дизайну.

3.3 Розробка дизайну мобільного застосунку

Досвід користувача (UX) і дизайн інтерфейсу користувача (UI) є двома важливими аспектами будь-якого цифрового продукту. Перший завжди орієнтований на клієнтський досвід. Це включає в себе розробку інтерфейсу та взаємодій, які будуть присутні в майбутьому застосунку. Дизайн UX/UI – це процес, який починається зі створення каркасів або прототипів, які допоможуть зрозуміти користувацький інтерфейс, яким чином користувач буде взаємодіяти з ним та як цей інтерфейс буде виглядати. Саме це було виконано та описано в попередньому підрозділі роботи.

Після цього переходять власне на етап створення дизайну користувацького інтерфейсу. Цей етап зосереджений на дизайні та стилі інтерфейсу користувача. Але для того, аби створити гарний інтерфейс, потрібно дотримуватись відповідних правил та рекомендацій.

Google Material Design та Human Interface Guidelines. Перед тим, як розробляти дизайн для мобільного застосунку треба уважно ознайомитися із гайдлайнами. Що це таке і навіщо воно потрібно? Уявимо, що користувач звертається до застосунку для того, аби якомога швидше вирішити свою проблему. Це отримується зробити лише при умові, якщо продукт має продуманий інтерфейс із зрозумілою для людини системою навігації. Звісно, кожен інтерфейс має бути унікальним, зі своєю “родзинкою”. Але при його розробці дизайнери повинні дотримуватись загальноприйнятих правил та рекомендацій, які прописані саме в гайдлайнах.

Гайдлайни — це правила, принципи та рекомендації від розробників операційної системи, платформи, завдяки яким усі цифрові продукти виглядають одноманітно стосовно створеної навігації, елементів інтерфейсу, використаної типографіки та стилів.

На сьогоднішній день існує два гайда для найвідоміших платформ, а саме — Google Material Design System, який призначений для операційної системи Android та Apple Human Interface Guidelines під iOS [10].

Material Design — це філософія дизайну, розроблена Google, яка прагне створити користувацькі інтерфейси, які легко зрозуміти та побачити. Material Design підкреслює важливість ясності, простоти та стриманості, щоб користувачі могли зосередитися на поставленому завданні. Мета матеріального дизайну – не тільки зробити так, аби інтерфейси виглядати чудово, але й зробити їх максимально реалістичними. Material Design мав вплив протягом усієї розробки Android, впливаючи як на програми, створені для телефонів, так і на програми, що використовуються у веб-браузерах.

Material Design важливий, оскільки він полегшує використання програм і вебсайтів. На сьогодні люди мають більше пристроїв для використання, і не всі пристрої мають однаковий інтерфейс користувача. Якщо продукти компанії створюють продукти з користувацьким інтерфейсом, який простий у використанні на всіх пристроях, люди з більшою ймовірністю скористаються застосунком або вебсайтом саме такої компанії. Також Material Design візуально привабливий, він використовує геометричні форми, сміливі кольори та прості лінії для створення привабливого інтерфейсу. Адже клієнтам потрібні програми та вебсайти, які добре виглядають і прості у використанні.

Щоб створити проєкт на основі системи Google Material Design, спочатку потрібно зрозуміти, що це за система і що вона означає. Після цього треба вирішити, як можна використовувати її у своєму проєкті. Нарешті, необхідно створити проєкт, використовуючи доступні інструменти Material Design.

Якщо говорити про другий гайд, а саме про Human Interface Guidelines (HIG), то можна сказати, що це документ, який викладає принципи проєктування користувацьких інтерфейсів у продуктах Apple. HIG був вперше випущений в 1997 році і з тих пір кілька разів переглядався, щоб внести зміни в технології, тенденції дизайну інтерфейсу та відгуки клієнтів. HIG надає загальні вказівки щодо проєктування користувацьких інтерфейсів на всіх платформах, включаючи настільні, мобільні пристрої, планшети, годинники і побутова техніка — і охоплює такі теми, як методи введення, візуальне представлення інформації в меню та діалогових вікнах, використання типографіки та інтервалів у інтерфейсі програми або вебсайту.

HIG — це набір найкращих практик і рекомендацій для всіх типів інтерфейсів користувача, від простих екранів до складних діалогових вікон. Конкретні правила охоплюють все, від текстових міток до розташування елементів керування. Крім того, HIG встановлює стандарти дизайну кнопок, вікон та інших елементів. Дотримуючись інструкцій, дизайнери можуть створювати інтерфейси, які прості у використанні та виглядають професійно. Хоча HIG не є законом, проте це набір рекомендацій, якого дотримуються багато дизайнерів при створенні застосунків. Адже HIG полегшує користувачам розуміння та використання програм.

Під час виконання роботи було прийнято рішення розробляти застосунок для платформи Android, тому відповідним гайдом для створення якісного дизайну було вибрано саме Material Design.

Дизайн користувацького інтерфейсу (UI design). При проєктуванні та розробці мобільного застосунку важливо пам’ятати саме про користувацький інтерфейс (UI). Дизайн користувацького інтерфейсу — це процес проєктування видимого середовища програми для користувача, включаючи меню, екрани та діалогові вікна. На додаток до візуальних аспектів дизайну користувацького інтерфейсу, зручність використання також відіграє важливу роль у створенні якісних та гарних інтерфейсів.

Проєктування користувацького інтерфейсу — це процес розробки інтерфейсу для інформаційної системи. Процес створення UI-дизайну передбачає використання програмного забезпечення для створення інтерфейсів для різноманітних застосунків. Це програмне забезпечення можна використовувати для різних цілей, наприклад, для розробки інтерфейсів веб-сайтів, створення інтерфейсів для комп'ютерних ігор та мобільних застосунків. UI-дизайн передбачає використання різноманітних загальних елементів, таких як кнопки, меню, вікна та значки. Ці елементи об'єднані, щоб створити цілісний і візуально привабливий інтерфейс користувача для програм. Також розробка UI-дизайну включає в себе створення кольорів, стилів, вибір шрифтів, які будуть присутні в розроблювальному застосунку. Інтерфейс інформаційної системи має бути естетично приємним. Також на етапі розробки UI-дизайну потрібно пам'ятати про те, що інтерфейс має бути легким та приємним у користуванні. Створений дизайн має має бути орієнтованим на користувача та відображати задану тематику продукту.

Результат етапу розробки дизайну мобільного застосунку має включати в себе готовий UI Kit та дизайн інтерфейсу. UI Kit — це загальний набір елементів на основі яких побудовано програмний продукт. Також це можна назвати своєрідною візуалізацією всії елементів окремо на кожному етапі взаємодії людини з інтерфейсом. Набір інтерфейсу користувача — це набір усіх елементів, на яких побудовано інтерфейс користувача (елементи інтерфейсу користувача) вашого веб-сайту. По суті, це візуалізація кожного з елементів на кожному етапі взаємодії користувача з інтерфейсом.

Набір елементів інтерфейсу користувача потрібен для стандартизації, спрощення, а також комплексного підходу до реалізації проєктів. Він не тільки дає змогу швидко та ефективно оформлювати кожен екран застосунку з використанням різних елементів та деталей, але й є основним конструкторським документом проєкту, до якого звертаються верстальники, дизайнери, розробники на всіх етапах розробки проєкту.

Було створено UI Kit для розроблювального застосунку, який показано на рисунку 3.5.

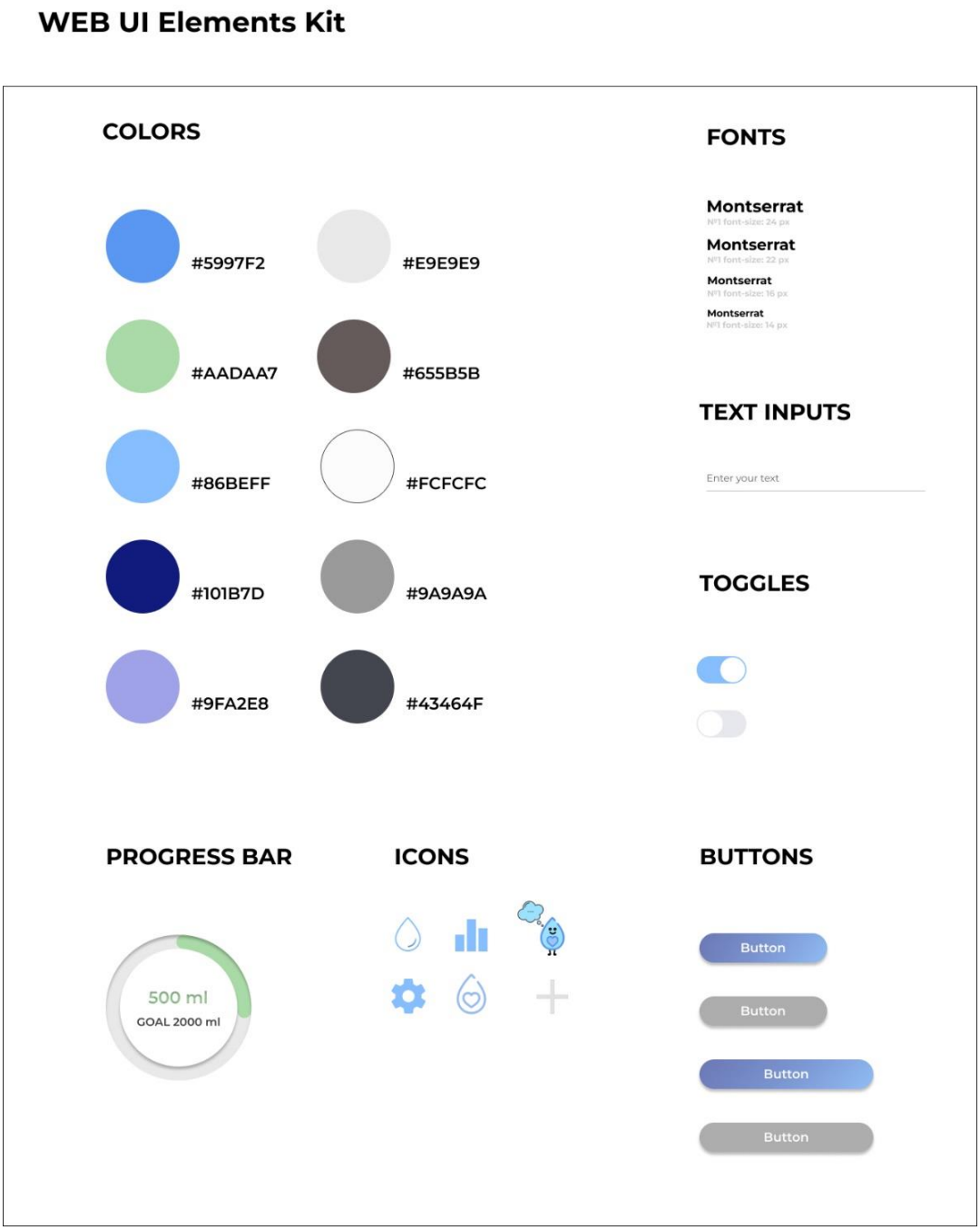

Рисунок 3.5 — UI Kit для мобільного застосунку «Droplet»

На основі раніше показаного прототипу майбутнього продукту та UI Kit, було розроблено відповідний дизайн для мобільного застосунку, який зображено на рисунках 3.6, 3.7.

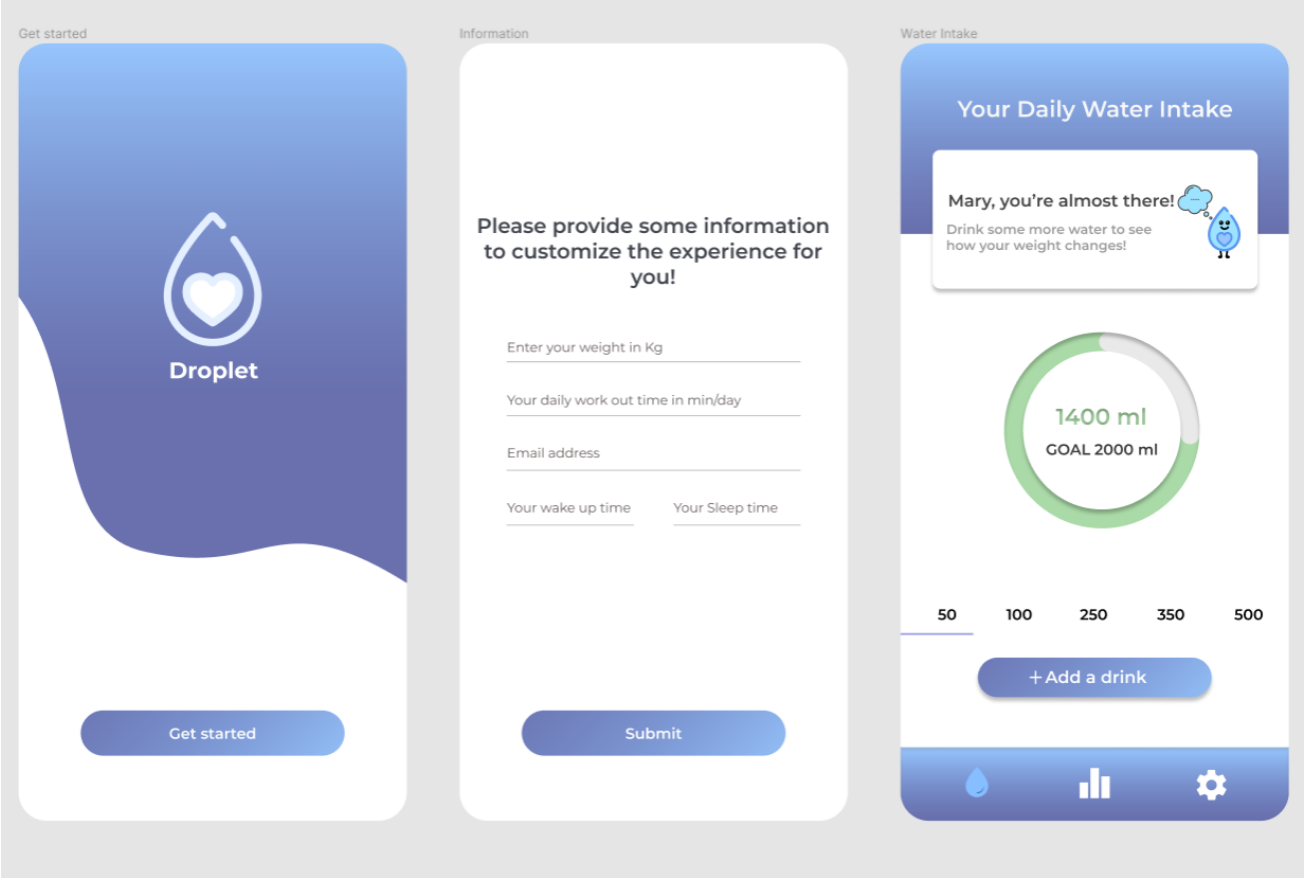

Рисунок 3.6 — Створений дизайн інтерфейсу мобільного застосунку «Droplet»

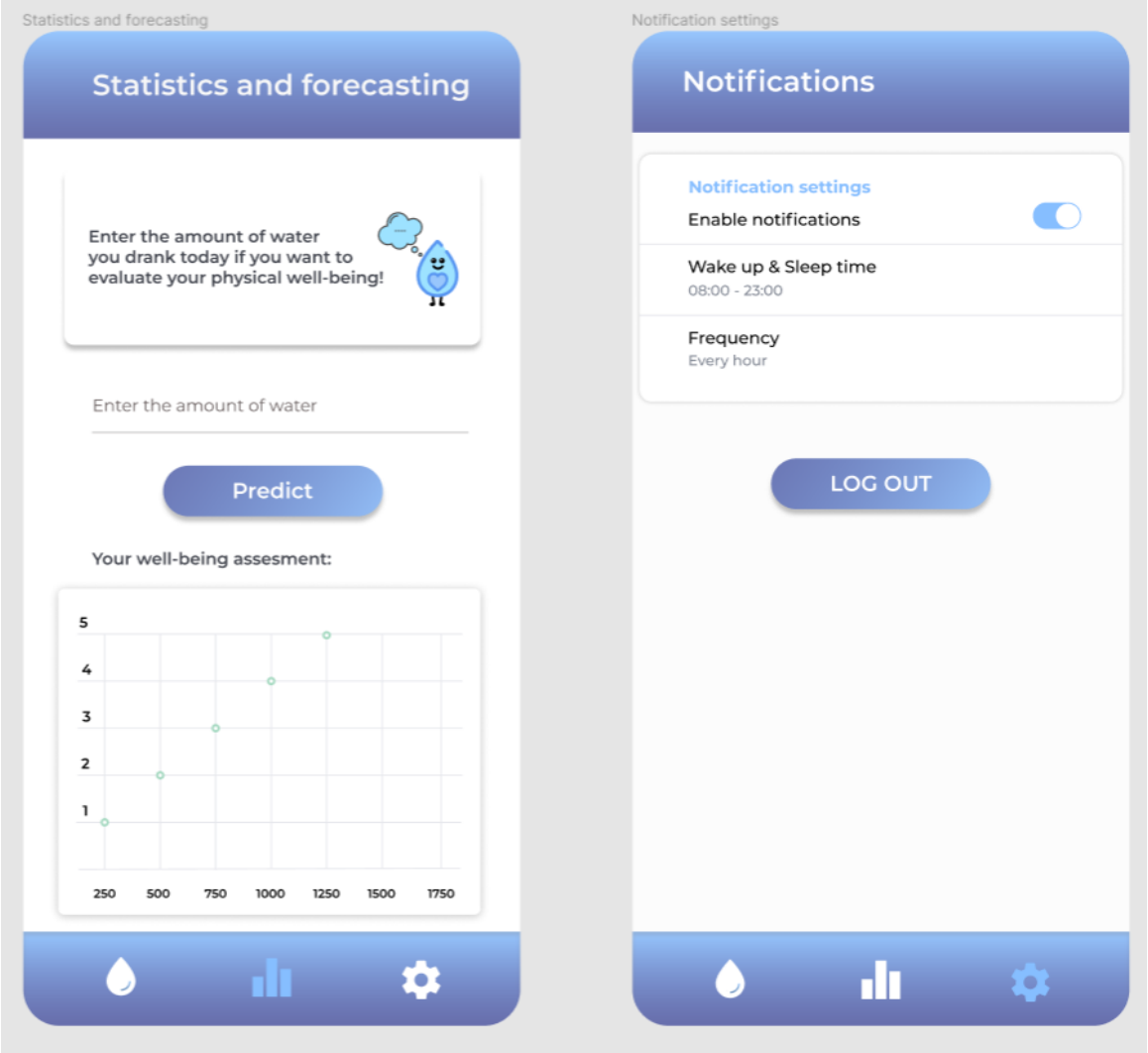

Рисунок 3.7 — Створений дизайн інтерфейсу мобільного застосунку «Droplet»

Висновки до розділу 3

Було розглянуто основні етапи проєктування мобільного застосунку. Створено та описано структуру розроблювального продукту, реалізовано діаграму, яка відображає доступні дії користувача під час роботи із системою. Також було показано структурну діаграму мобільного застосунку, яка описує внутрішню будову модулів з яких складається дана інформаційна система та показує зв’язок між цими модулями. Також було описано, яку роль в розробці програми відіграє прототипування та які бувають типи прототипів, їх особливості використання на практиці.

Розроблено відповідний прототип для застосунку, який включає в собі ознаки інтерактивного та каскадного прототипів та концепцію UX дизайну. Після цього було проаналізовано процес розробки дизайну мобільних інтерфейсів, популярні гайдлайни — Material Design і HIG, їхні особливості та переваги використання. Також під час виконання третього розділу роботи було створено UI Kit та сам дизайн інтерфейсу для розроблювального застосунку.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ

4.1 Архітектура мобільного застосунку

У більшості випадків мобільні застосунки мають визначену архітектуру. Серед них найпопулярнішими є MVC, MVP та MVVM. Нові архітектури з'являються у зв'язку із зростанням складності систем та з підвищенням вимог. Саме через це і зростає архітектурна складність. Для розроблювального застосунку буде використовуватися такий шаблон проєктування архітектури як Model - View - ViewModel (MVVM). Схема шаблону MVVM зображена на рисунку 4.1.

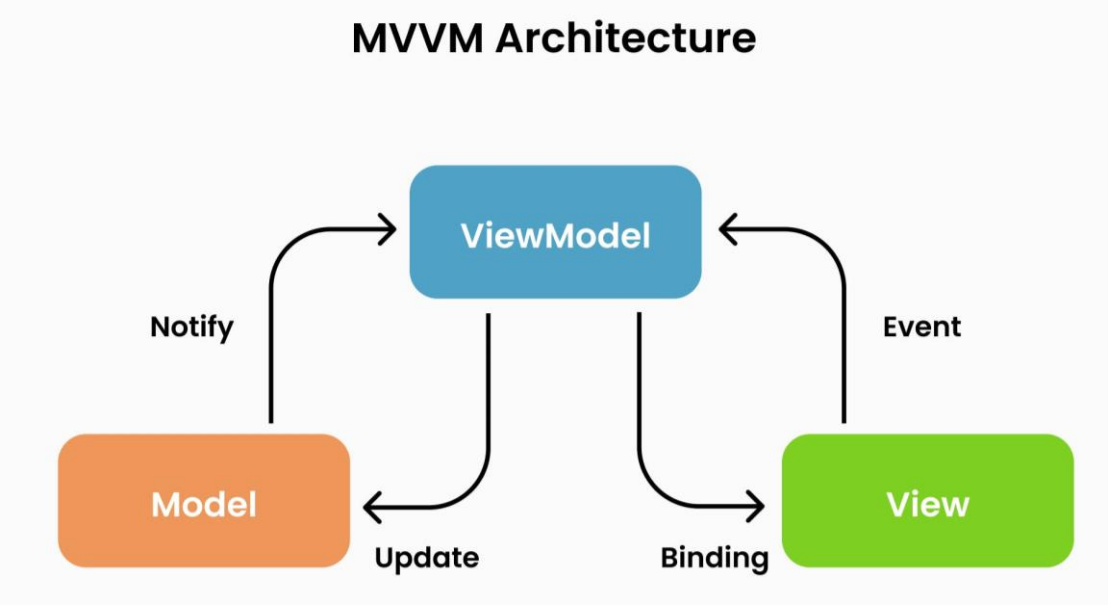

Рисунок 4.1 — Схема шаблону MVVM до проєктування архітектури

MVVM — це паттерн, який також є концепцією програмної архітектури, що використовує ViewModel у якості посередника між Model та View.

Model описує використані дані в застосунку. ModelView відповідає за представлення даних з Model, а також перетворення форматів даних, але не впливає на саму зміну елементів View, відповідаючого за інтерфейс користувача.

Відмінністю підходу MVVM від MVP є те, що MVP є застарілою архітектурою, тому для MVVM зв'язування View з ViewModel відбувається автоматично, а для MVP описаний зв'язок треба програмувати, адже ніяких компонентів від Google для неї немає.

Треба відмітити те, що архітектура в даній роботі була використана для подальшої підтримки масштабування проєкту, саме задля цього потрібно було закласти відповідну архітертуру MVVM із самого початку.

4.2 Опис програмної реалізації

Android застосунки складаються із самостійних структурних одиниць, які називаються компонентами. Кожен компонент є унікальним елементом стуктури системи. Для того, аби запустити компоненти система повинна знати, що всі вони існують. Саме для цього служить спеціальний файл AndroidManifest.xml. У цьому файлі потрібно об’являти всі компоненти, що містяться в проєкті, а також можна вказувати потрібні для застосунку уповноваження користувача.

Activity — це компонент застосунку, який пов’язаний з екраном (layout), а ще відповідає за поведінку, яка демонструється на цьому екрані. Зазвичай, для того, аби представити layout, потрібно використовувати файл з розширенням .xml. Структура усіх застосунків складається із екранів, які взаємодіють один з одним.

Під час того, як користувач запускає мобільний застосунок, виводиться перша Activity, яка вважається головною, вона має назву MainActivity. Саме з неї можливий подальший перехід на інші екрани застосунку.

Activity може бути створена (created), запущена (started), також відновлена (resumed), призупинена (paused), повністю зупинена (stopped) або ж знищена (destroyed). Це ті стани, в яких може перебувати компонент Activity.

Окрім Activity в Android-розробці також все частіше використовують Fragment. Fragment можна назвати подібністю Activity, яку можна підключати в різні частини застосунку.

Під час виконання бакалаврської кваліфікаційної роботи використовувались саме фрагменти, адже вони більш зручні для реалізації переходу між екранами, яка була виконана за допомогою нижньої панелі навігації. Але в проєкті є також і Activity, але вона слугує в якості хосту для всіх фрагментів та не приймає участі в архітектурі.

Нижче наведено короткий опис усіх фрагментів, які було створено для розробленого застосунку.

SplashFragment — пустий екран з логотипом, який з'являється першим при запуску застосунку.

IntroFragment — екран старту, саме з нього відбувається подальший перехід на екран введення інформації про користувача.

ParamsFragment — екран із полями для введення інформації про користувача.

WaterControlFragment — головний екран застосунку, через який можна переходити на всі інші екрани за допомогою нижньої панелі навігації.

StatisticFragment — екран з прогнозуванням оцінки самопочуття людини та з відображенням статистики через лінійну діаграму.

SettingsFragment — екран з налаштуваннями повідомлень.

Розрахунок добової норми води. Для розрахунку добової норми води необхідні дані користувача, а саме: вага та час фізичної активності.

Рекомендована норма води позначена як Water, підрахунок виконується за формулою 4.1, яка наведена нижче.

$$\text{Water} = (\text{w} \times 30) + \left(\left(\frac{\text{t}}{30}\right) \times 355\right), \quad (4.1)$$

де w – вага людини в кілограмах;

t – час фізичної активності в хвилинах за день.

Розрахунок проводиться за принципом 30 мілілітрів рідини на кілограм ваги людини, а також додавання 355 мілілітрів води до кожних 30 хвилин фізичних навантажень. Даний алгоритм підрахунку добової норми води було виведено, спираючись на наукові праці Університету Міссурі та Американського коледжу спортивної медицини [11-13].

Програмна реалізація отриманого алгоритму передбачає введення в TextInputEditText параметрів ваги людини та часу фізичної активності. Якщо значення були коректними, то після натискання кнопки «Підтвердити» в змінні weight та workTime записуються дані, які було введено користувачем, а також в ParamsFragment.kt викликається функція calculateIntake(), яка розраховує індивідуальну норму води людини (рис. 4.2). Також після цього відбувається перехід на головний екран застосунку.

```
fun calculateIntake(weight: Int, workTime: Int) {
    val total = ((weight * 30) + ((workTime / 30) * 355))
    saveTotalToPref(total)
}
```

Рисунок 4.2 — Розрахунок добової норми води

Після розрахунків і збереження цільова кількість води, яку користувачу рекомендовано випивати записується в TextView та показується в центрі кругової шкали прогресу в мілілітрах.

Реалізація додавання порцій води. Спочатку було створено кнопку для додавання води, а також контейнер TabLayout в якому міститься група елементів TabItem із значеннями порцій води, яку людина може випивати у мілілітрах, а саме — 50, 100, 250, 350 та 500. Сама операція додававання води програмно реалізована таким чином, що при натисканні кнопки addDrinkButton зчитується обрана позиція Tab і додається відповідне значення кількості води, яке вказано в

TabItem, через функцію handleIntake(). Даний алгоритм описано на рисунку 4.3. Присвоєння значення рівня води в організмі відбувається у функції UpdateChartIntake(). У цій функції до нинішнього значення додається нове обране значення.

```
        addDrinkButton.setOnClickListener { it: View!
            when (tabLayout.selectedTabPosition) {
                0 -> handleIntake( intake: 50)
                1 -> handleIntake( intake: 100)
                2 -> handleIntake( intake: 250)
                3 -> handleIntake( intake: 350)
                4 -> handleIntake( intake: 500)
            }
        }
    }
}

private fun handleIntake(intake: Int) {
    viewModel.increaseTotalIntake(intake)
    updateChartIntake()
}
```

Рисунок 4.3 — Реалізація додавання обраної кількості води

Реалізація кругової шкали прогресу для контролю балансу води в організмі людини. При створенні проєкту було використано потужну бібліотеку DecoView за допомогою якої створюються анімовані кругові діаграми, які потім можна налаштувати в застосунках, створених на основі платформи Android. Інтерактивність кругової шкали прогресу прописано у функціях initChart() та UpdateChartIntake(). Лістинг коду даних функцій зображено на рисунку 4.4.

```
private fun initChart() {
    val total = viewModel.getTotal()
    binding.apply { this: FragmentWaterControlBinding
        val totalItem = SeriesItem.Builder(Color.parseColor( colorString: "#E9E9E9"))
            .setRange( minValue: 0f, total.toFloat(), total.toFloat())
            .build()
        val intakeItem = SeriesItem.Builder(Color.parseColor( colorString: "#AADAA7"))
            .setRange( minValue: 0f, total.toFloat(), initialValue: 0f)
            .build()

        val totalChartIndex: Int = dynamicArcView.addSeries(totalItem)
        intakeChartIndex = dynamicArcView.addSeries(intakeItem)

        dynamicArcView.addEvent(
            DecoEvent.Builder(total.toFloat())
                .setIndex(totalChartIndex)
                .setDuration(0)
                .build()
        )
        updateChartIntake()
    }
}

private fun updateChartIntake() {
    val newIntake = viewModel.getIntake()

    binding.apply { this: FragmentWaterControlBinding
        dynamicArcView.addEvent(
            DecoEvent.Builder(newIntake.toFloat())
                .setIndex(intakeChartIndex!!)
                .setDuration(1000)
                .build()
        )
        waterLevelText.text = "${viewModel.getIntake()} ml"
    }
}
```

Рисунок 4.4 — Реалізація функцій initChart() та updateChartIntake()

При оновленні інтерфейсу викликається функція initChart(). У даній функції загальне значеня норми води записується у змінну totalItem. А значення випитої кількості води записується у змінну intakeItem. У створенній змінні totalChartIndex записується значення випитої води і при виклику функції оновлення даних addEvent() до неї додається нове значення. На основі цих даних будується кругова шкала прогресу.

Функція UpdateChartIntake() оновлює шкалу прогресу балансу води людини. У змінну newIntake записується розраховане значення поточного рівня води, потім ці дані оновлюються в круговій шкалі і змінюється значення випитої кількості води. Відображення випитої кількості води на круговій шкалі прогресу показано на рисунку 4.5.

Рисунок 4.5 — Відображення випитої кількості води на круговій шкалі прогресу

Створення повідомлень. Для того, аби користувач не забував пити воду було реалізовано повідомлення, які б нагадували людини приймати воду протягом доби.

Спочатку було задано вміст повідомлень, який показано у фрагменті коду на рисунку 4.6.

```
val builder = NotificationCompat.Builder(requireContext(), CHANNEL_ID)
    .setSmallIcon(R.drawable.logo_drop)
    .setContentTitle("Droplet")
    .setContentText("It's time for drink water")
    .setPriority(NotificationCompat.PRIORITY_HIGH)
```

Рисунок 4.6 — Встановлення вмісту сповіщень

У фрагменті коду можна помітити, що setSmallIcon() відображає логотип мобільного застосунку, setContentTitle() відповідає за заголовок повідомлення, а setContentText — за основний текст сповіщень. За допомогою setPriority() задається пріоритет нагадувань. Пріоритет визначає рівень нав’язливості повідомлень, які будуть приходити користувачеві. Також потрібно задати ідентифікатор каналу для сумісності із новішими версіями Android [14].

Потім було створено канал сповіщень за допомогою NotificationCompat.Builder об’єкту, шляхом передачі екземпляру NotificationChannel до CreateNotificationChannel().

При цьому, NotificationChannel потребує importance, використовуючи константу із класу NotificationManager. Самей цей параметр вирішує, як саме переривати користувача для повідомлення, яке належить саме цьому каналу [15].

Потім було налаштовано дію натискання повідомлень і встановлено метод setOnClickListener, який прослуховує натискання, в нашому випадку на перемикач Switch. Таким чином, повідомлення з’являються тільки за умови, якщо Switch знаходиться в активованому стані.

На рисунку 4.7 зображено іконку створеного мобільного застосунку із непрочитаним повідомленням.

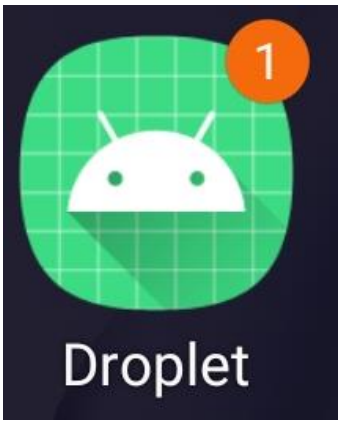

Рисунок 4.7 — Непрочитане повідомлення

На рисунку 4.8 зображено виведене повідомлення із нагадуванням про те, що користувачу потрібно випити воду.

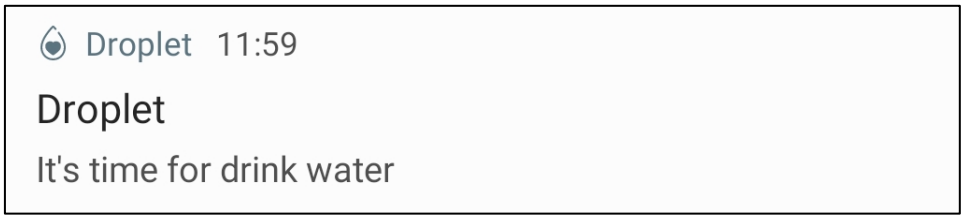

Рисунок 4.8 — Повідомлення із заголовком та текстом

Створення та налаштування простої моделі лінійної регресії та її запуск в Android. Було реалізовано екран застосунку із виведенням прогнозованої лінійної оцінки самопочуття людини за дванадцятибальною шкалою в залежності від введеної користувачем випитої кількості води протягом доби. Для реалізації цієї ідеї було створено модель машинного навчання та виконано її розгортання у застосунку Android за допомогою міні-API, а саме TensorFlow-Lite.

Спочатку було створено просту модель лінійної регресії із навчальними даними у вигляді масивів x та y, де масив x — це множина значень стосовно кількості випитої води людиною за добу, а масив y — множина оцінок самопочуття людини за дванадцятибальною шкалою. Для прикладу розраховано добову норму води для людини вагою 75 кілограм та фізичною активністю 60

хвилин та визначено, що користувачу рекомендовано випивати 2,96 літри води в день. Після чого отриманий результат було округлено до 3 літрів.

Після етапу створення простої моделі лінійної регресії було виконано збереження моделі у файл із розширенням .pbtx. Потім було створено об’єкт TFLite Converter із навченою моделлю машинного навчання.

Після проведення цих дій було перетворено об’єкт в об’єкт tflite та збережено як файл із аналогічним розширенням. Далі було імпортовано створену модель degree.tflite у проєкт в Android Studio. Структуру проєкту після внесених змін показано на рисунку 4.9.

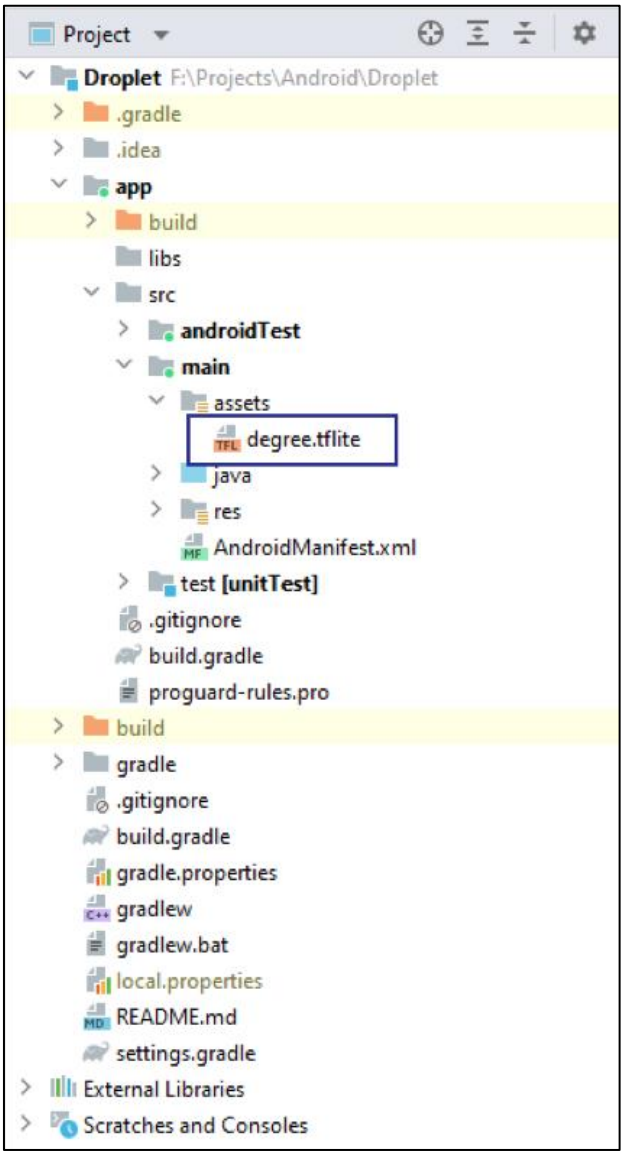

Рисунок 4.9 — Структура проєкту

Додано сценарій у build.gradle (Module: app), який дає змогу не стискати файл із розширенням .tflite під час створення apk застосунку. Також було додано залежність implementation ‘org.tensorflow:tensorflow-lite:+’. Після чого було виконано синхронізацію проєкту та створено структуру fragment_statistic.xml. Після вказаних змін потрібно було додати метод loadModelFile(), який потрібен для завантаження файлу моделі .tflite. Далі було проініціалізовано змінні та імпортовано Interpreter із TensorFlow-Lite API.

Після цього створено об’єкт TFLite Interpreter та метод doInference(), який приймає вхідний рядок за вхідні дані, а потім конвертує це в float масив та використовує метод tflite.run() для того, аби отримати вихідні дані та повернути їх як значення типу float.

Після того, як користувач в поле вводу заносить кількість води, яку було випито протяго доби та натискає кнопку Predict, викликається метод doInference. Цей метод отримує вихідні дані, які відображаються як прогнозована оцінка самопочуття людини у TextView. Приклад отриманої прогнозованої оцінки показано на рисунку 4.10.

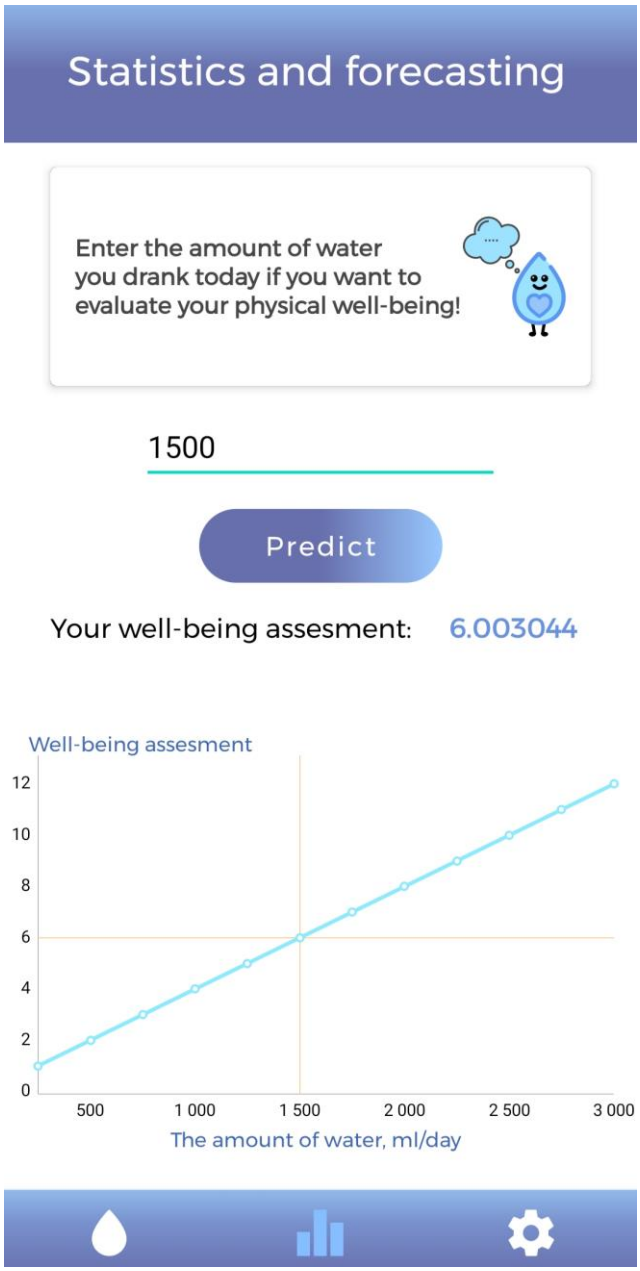

Рисунок 4.10 — Прогнозована оцінка фізичного самопочуття людини в залежності від випитої кількості води протягом доби

Після реалізації виведення прогнозованої оцінки фізичного самопочуття людини, шляхом створення простої моделі лінійної регресіїі та її запуску в Android, було отримано та проаналізовано дані групи людей, а саме — п’яти користувачів (Користувач 1, Користувач 2, Користувач 3, Користувач 4, Користувач 5) мобільного застосунку, вага яких становить 45 кг, 55 кг, 65 кг,

75 кг та 85 кг, а час фізичної активності відповідно — 10 хв, 20 хв, 30 хв, 60 хв та 80 хв за добу.

За параметрами, які вказано вище, розраховано добову норму води для кожного користувача за допомогою розробленого мобільного застосунку та отримано — 1320 мл, 1620 мл, 2400 мл, 3000 мл, 3240 мл відповідно. Спрогнозовані значення оцінки фізичного самопочуття людей за дванадцятибальною шкалою, а також кількість вживаної води за добу протягом тижня користувачами було занесено до таблиць 4.1-4.5.

Таблиця 4.1 — Користувач 1

	День 1	День 2	День 3	День 4	День 5	День 6	День 7
Кількість випитої води за добу, мл	650	550	1200	840	780	900	350
Оцінка фізичного самопочуття	5,96	4,94	10,88	6,98	7,07	8,18	3,09

Таблиця 4.2 — Користувач 2

	День 1	День 2	День 3	День 4	День 5	День 6	День 7
Кількість випитої води за добу, мл	800	1000	1400	500	1200	200	750
Оцінка фізичного самопочуття	5,91	7,39	10,31	3,67	8,89	1,82	5,53

Таблиця 4.3 — Користувач 3

	День 1	День 2	День 3	День 4	День 5	День 6	День 7
Кількість випитої води за добу, мл	2400	200	700	1500	1000	2000	2200
Оцінка фізичного самопочуття	11,98	1,03	3,52	7,5	5,02	9,99	11,01

Таблиця 4.4 — Користувач 4

	День 1	День 2	День 3	День 4	День 5	День 6	День 7
Кількість випитої води за добу, мл	3000	600	1200	1500	1300	2600	2000
Оцінка фізичного самопочуття	12	2,41	4,81	6	5,21	10,4	8

Таблиця 4.5 — Користувач 5

	День 1	День 2	День 3	День 4	День 5	День 6	День 7
Кількість випитої води за добу, мл	800	270	2400	1400	1000	3000	2900
Оцінка фізичного самопочуття	3,12	1,02	8,98	5,28	3,84	11,67	10,98

Спираючись на отримані дані із вищенаведених таблиць, можна помітити закономірність покращення оцінки фізичного самопочуття користувачів із приближенням вживаної кількості води за добу до рекомендованої норми.

4.3 Керівництво користувача

У результаті виконаної роботи було розроблено мобільний застосунок підтримки водного балансу в організмі людини. Було проведено тестування реалізованого продукту та створено керівництво користувача для того, аби описати алгоритм взаємодії людини із застосунком.

Під час запуску реалізованого застосунку на декілька секунд показується екран завантаження із символічним логотипом у вигляді каплі води (рис. 4.11). Після чого з’являється екран входу до мобільного застосунку, який показано на рисунку 4.12.

Рисунок 4.11 — Екран завантаження

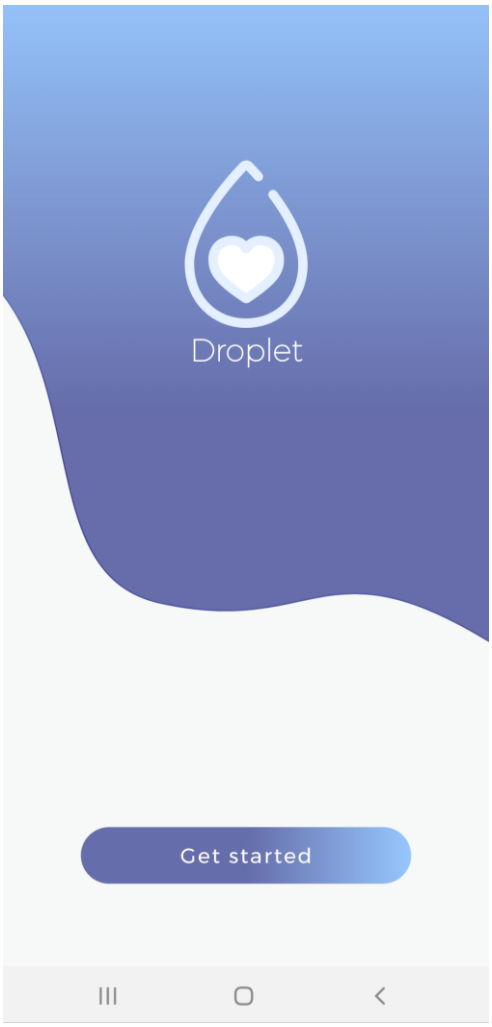

Рисунок 4.12 — Екран старту

Після того, як користувач натисне на кнопку «Get started», відкривається екран для введення індивідіальної інформації про користувача, а саме — вага людини, час фізичної активності, час сну та час пробудження. Даний екран показано на рисунку 4.13 та рисунку 4.14.

Після того, як людина введе коректні дані та натисне кнопку «Submit», відкриється головний екран з круговою шкалою прогресу, який показано на рисунку 4.15.

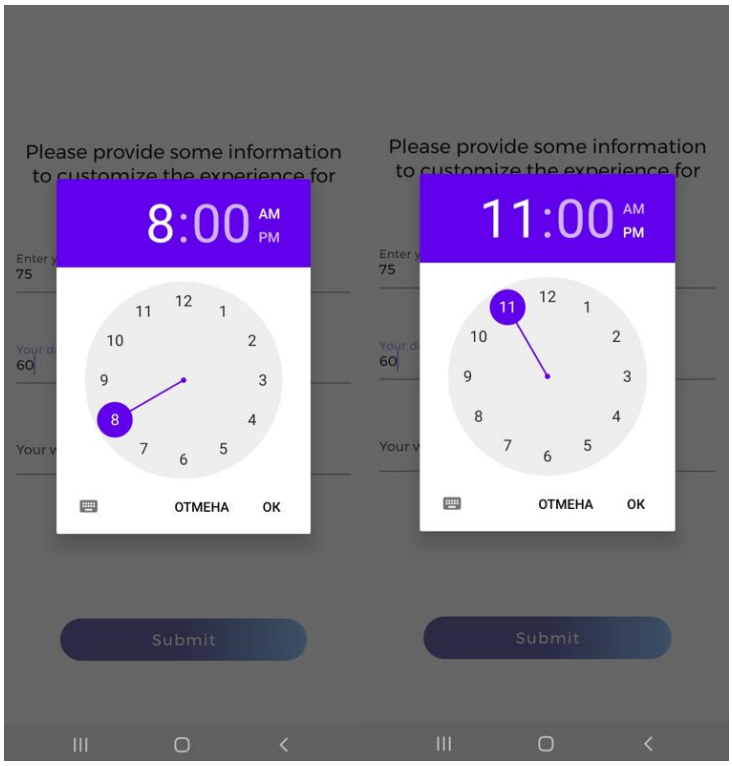

Рисунок 4.13 — Екран введення інформації про користувача

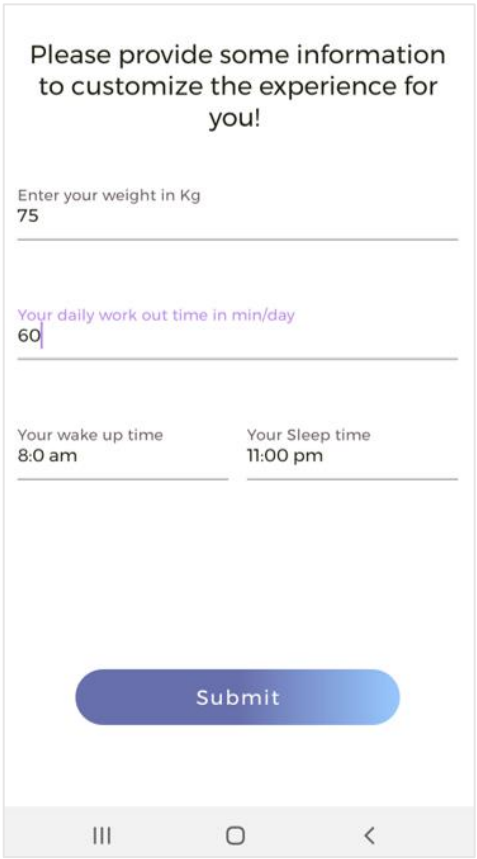

Рисунок 4.14 — Екран введення інформації про користувача

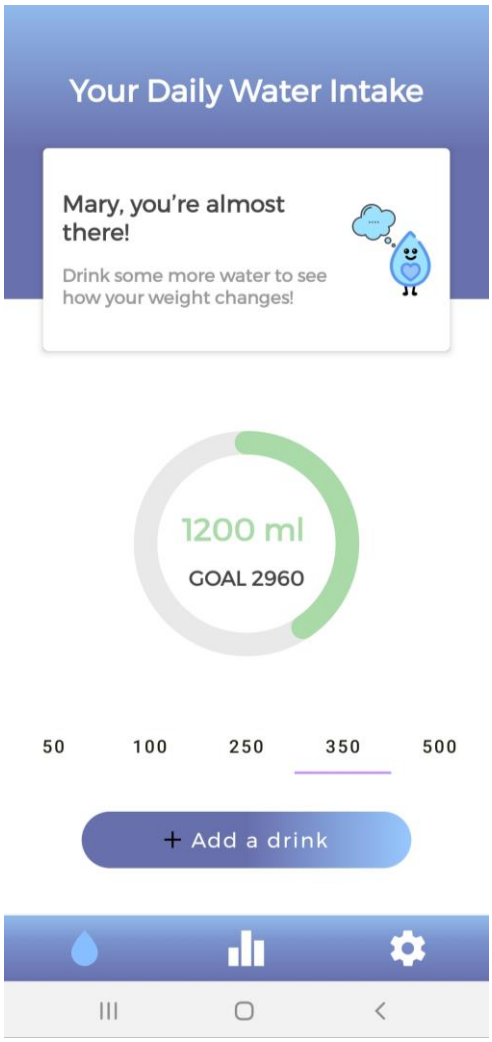

Рисунок 4.15 — Головний екран застосунку

На головному екрані знаходиться кругова шкала прогресу, яка відображає кількість уже випитої людиною води, а також наглядно зображує скільки ще води потрібно випити людині до кінца доби. Також користувач має змогу вибирати підходящу для нього порцію води та додавати її до загальної кількості випитої рідини. Також на головному екрані знаходиться нижня панель навігації, через яку можна здійснювати швидкі переходи між екранами. Це досить зручне та просте у використанні для користувача рішення.

Після того, як людина натисне на іконку у вигляді діаграми, то перед нею з’явиться екран із прогнозуванням оцінки фізичного самопочуття в залежності від введеної кількості випитої води за добу, його можна побачити на

попередньому рисунку 4.10. А нижче виведеної оцінки буде зображено лінійну діаграму, яка наглядно показує лінійну залежність між випитою кількістю води та самопочуттям людини.

Якщо користувач натисне на іконку із зображенням шестерні, то з’явиться екран налаштувань повідомлень, який показано на рисунку 4.16. Повідомлення можна включити, якщо натиснути на switch напроти напису «Enable notifications».

Також можна вийти із застосунку, натиснувши на кнопку «LOG OUT».

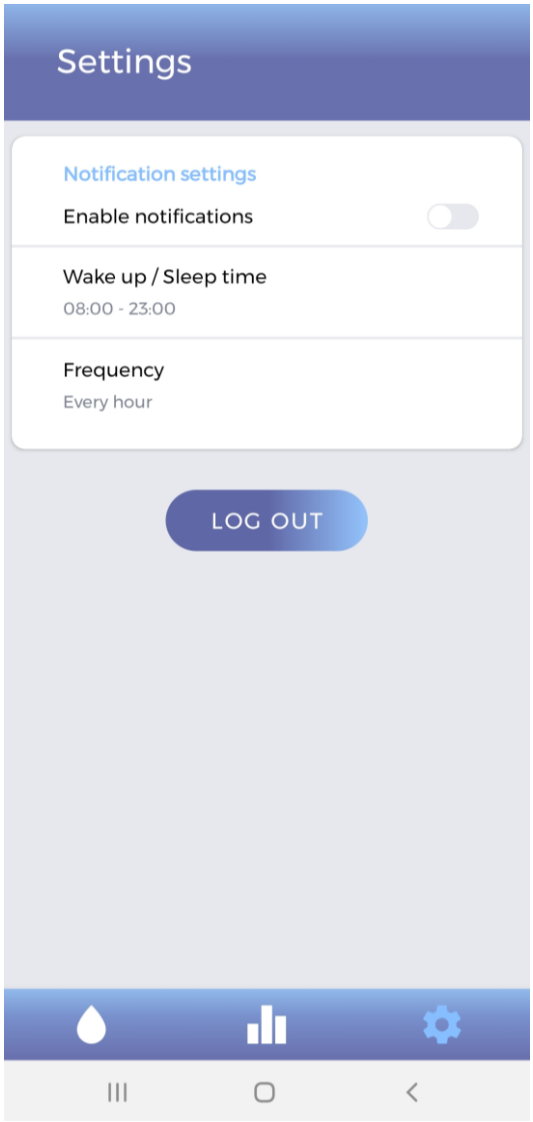

Рисунок 4.16 — Екран налаштувань повідомлень

Висновки до розділу 4

Обрано та описано архітектуру для мобільного застосунку, а саме — MVVM. Було показано принцип роботи даної архітектури, а також було сказано про те, чим MVVM відрізняється від інших паттернів, які також є концепцією програмної архітектури.

Також під час виконання даного розділу була описана програмна реалізація розробленого для бакалаврської кваліфікаційної роботи застосунку підтримки водного балансу в організмі людини. Було описано реалізацію розрахунку добової норми води за індивідуальними параметрами людини. Також було описано та проаналізовано реалізацію додавання порцій води та кругової шкали прогресу, яка відображає кількість вже випитої людиною рідини. Ще було описано програмну реалізацію виведення повідомлень із нагадуванням пити воду та створення і налаштування простої моделі лінійної регресії, а також її запуск в Android.

Було створено керівництво користувача, аби описати алгоритм переходів між екранами мобільного застосунку, а також детальну взаємодію людини із застосунком.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

МОБІЛЬНИЙ ЗАСТОСУНОК ПІДТРИМКИ ВОДНОГО БАЛАНСУ В ОРГАНІЗМІ ЛЮДИНИ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810324

Виконала студентка 4-го курсу, групи 402
Т. О. Удовик
(підпис, ініціали та прізвище)
«___»_____ 2022 р.

Консультант *старший викладач*
(наук. ступінь, вчене звання)
О. В. Макарова
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

5 АНАЛІЗ ВИМОГ ДО ОРГАНІЗАЦІЇ ЗАХОДІВ ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ ПІД ЧАС РОБОТИ ЗА КОМП'ЮТЕРОМ

Всеохоплююча інформатизація суспільства як глобальний, загальносвітовий процес та багатопланові зміни в організації праці спонукає багатьох людей використовувати різноманітні засоби електронно-обчислювальної (комп'ютерної) техніки та опановувати відповідні інформатичні технології.

Комп'ютер - нова реальність в нашому житті, і людина в силу своєї еволюції не пристосована до постійної роботи з ним. Упровадження комп'ютерних засобів (КЗ) в усі сфери життєдіяльності людини виявило не тільки позитивні, а й негативні наслідки їх використання, про що свідчать скарги користувачів комп'ютера (КК). Зокрема, щодо погіршення власного здоров'я та зниження функціональності організму.

Досить багато проблем зі здоров'ям можуть виникнути через використання комп'ютера. Наприклад, проблеми із зором, проблеми зі спиною, також від комп'ютера надходить електромагнітне випромінювання.

Але на сьогодні комп'ютер все одно залишається необхідною річчю при роботі вдома чи на підприємствах, тому людям потрібно знати правила роботи та безпеки під час роботи за ПК, вміти організовувати своє робоче місце та зменшувати негативний вплив під час довгострокової роботи.

Метою даного розділу бакалаврської роботи є створення безпечних і здорових умов праці на робочому місці або у виробниочму приміщенні, для досягнення якої були виділені необхідні завдання:

- організація та обладнання робочого місця;
- створення необхідних умов освітлення;
- формування правил безпеки під час роботи з персональним комп'ютером;
- визначення режиму праці та відпочинку за комп'ютером.

5.1 Організація та обладнання робочого місця

Під час розміщенні робочих столів з комп'ютерами необхідно дотримуватися певних правил:

– відстань між бічними поверхнями персональних комп'ютерів 1,2 м.;

– відстань від тильної поверхні одного персонального комп'ютера до екрана іншого – 2,5 м.

За необхідності особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2м.

Конструкція робочого місця користувача персонального комп'ютера повинна забезпечувати підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу повинна відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розташування на робочій поверхні обладнання, яке використовується (дисплея, клавіатури, принтера) і документів [16]. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, а глибина – 800-1000мм).

В робочому столі повинен бути простір для ніг заввишки не менше ніж 600мм, завширшки не менше ніж 500мм, завглибшки (на рівні колін) не менше ніж 450мм, на рівні простягнутої ноги не менше ніж 650мм. Робочий стілець повинен бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим. Регулювання за кожним із параметрів має здійснюватися незалежно, легко і надійно фіксуватися. Шаг регулювання елементів стільця має становити: для лінійних розмірів – 15-20мм, для кутових – 2-5 градусів. Зусилля регулювання має не перевищувати 20Н. Висота поверхні сидіння повинна регулюватися в межах 400-

500мм, а ширина і глибина становити не менше ніж 400мм. Кут нахилу сидіння — до 15 градусів вперед і до 5 градусів назад. Висота спинки стільця має становити (300+-20) мм, ширина — не менше ніж 380 мм, радіус кривизни горизонтальної площини — 400мм. Кут нахилу спинки має регулюватися в межах 1-30 градусів від вертикального положення. Відстань від спинки до переднього краю сидіння має регулюватися в межах 260-400мм. Для зниження статичного напруження м'язів верхніх кінцівок слід використовувати стаціонарні або змінні підлокітники завдовжки не менше ніж 250мм, завширшки 50-70мм, що регулюються за висотою над сидінням у межах 230-260мм і відстанню між підлокітниками в межах 350-500мм. Поверхня сидіння і спинки стільця має бути напівм'якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується. Робоче місце має бути обладнане підставкою для ніг завширшки не менше ніж 300мм, завглибшки не менше ніж 400мм, що регулюється за висотою в межах до 150мм і за кутом нахилу опорної поверхні підставки до 20 градусів. Підставка повинна мати рифлену поверхню і бортик по передньому краю заввишки 10мм.

Робочі місця необхідно розташовувати відносно світових прорізів таким чином, щоб природне світло падало переважно з лівого боку. Монітор повинен розташовуватися на оптимальній відстані від очей користувача, що становить 600-700мм, але не ближче ніж за 600мм з урахуванням розміру літерно-цифрових знаків і символів. Розташування екрана монітору повинно забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 градусів до нормальної лінії погляду працівника. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, звернутого до працюючого.

У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-15 градусів. Висота середнього рядка клавіш не повинна перевищувати 30мм. Поверхня клавіатури має бути матовою з коефіцієнтом

відбиття 0,4. Розташування пристрою введення — виведення інформації має забезпечувати добру видимість монітору, зручність ручного керування в зоні досяжності моторного поля і за висотою – 900-1300мм, за шириною 400-500мм. Під матричні принтери потрібно підкладати вібраційні килимки для гасіння вібрації та шуму.

Робоче місце з персональним комп'ютером слід обладнати пюпітром для документів, що легко переміщуються.

Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосування приекранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, які пройшли випробування в акредитованих лабораторіях та мають щорічний гігієнічний сертифікат.

5.2 Створення необхідних умов освітлення

Серед факторів зовнішнього середовища, які впливають на організм людини в процесі праці, світло займає одне з перших місць. Адже відомо, що майже 90% всієї інформації про навколишнє середовище людина одержує через органи зору. Під час здійснення будь-якої трудової діяльності втомлюваність очей, в основному, залежить від напруженості процесів, які супроводжують зорове сприйняття. До таких процесів відносяться адаптація, акомодація і конвергенція.

Світло впливає не тільки на функцію органів зору, а й на діяльність організму в цілому.

При поганому освітленні людина швидко втомлюється, працює менш продуктивно, зростає потенційна небезпека помилкових дій і нещасних випадків.

Згідно зі статистичними даними 5% травм можна пояснити недостатнім або нераціональним освітленням, а в 20% воно сприяє виникненню травм. Нарешті, погане освітлення може призвести до професійних захворювань, наприклад, до таких, як робоча міопія (короткозорість) спазм акомодації.

Для створення оптимальних умов зорової роботи слід враховувати не тільки кількість і якість освітлення, а й кольорове оточення. При світлому забарвленні інтер'єру завдяки збільшенню кількості відбитого світла рівень освітленості підвищується на 20-40% (при той же потужності джерел світла) різкість тіней зменшується, покращується рівномірність освітлення.

При надмірній яскравості джерел світла та оточуючих предметів може статися засліплення працівника. Нерівномірність освітлення та неоднакова яскравість навколишніх предметів призводять до частої преадаптації очі під час виконання роботи і, як наслідок цього - до швидкої втоми органів зору. Тому поверхні, добре освітлюються і знаходяться в полі зору, краще фарбувати в кольори середньої світлості, коефіцієнт відображення яких знаходиться в межах 0,3 - 0,6. Бажано, щоб поверхні були матові [17].

Раціональне освітлення робочих місць створює сприятливі умови для забезпечення безпечної праці, поліпшення якості продукції та підвищення продуктивності праці, впливає психологічно на людину, створює гарний настрій: бадьорий у працюючих [17]. Спектральний склад світла, зумовлюючи колірне сприйняття, може викликати забруднюючою дію і підсилювати відчуття тепла (оранжево-червона) частину спектру, або, заспокійливу дію (жовто-зелений колір) посилювати гальмують процеси (синьо-фіолетово частина спектра).

Освітлення виробничих приміщень характеризується кількісними та якісними показниками.

До основних кількісних показників відносяться: світловий потік, сила світла, яскравість і освітленість. До основних якісних показників зорових умов роботи можна віднести: фон, контраст між об'єктом і фоном, видимість.

Виробниче освітлення може виконується природне і штучне світло, а також може бути поєднаним.

За способом виконання системами штучного освітлення є: загальна, коли освітлювальні прилади повинні бути розташовані таким чином, щоб забезпечити досить рівномірну освітленість в зоні робіт і проходах; комбінована, коли крім

загального освітлення встановлюються світильники місцевого освітлення для створення більш високих рівнів освітленості на робочих місцях, де виконується напружена зорова робота. Користуватися в приміщеннях тільки приладами місцевого освітлення заборонено.

Комбіноване освітлення рекомендується для приміщень з роботами розрядів I-ОЛ.. При комбінованому освітленні загальне освітлення має становити 10% всієї норми освітленості, але не менше 150 лк і не більше 500 лк (для газорозрядних ламп).

Штучне освітлення передбачається використовувати у всіх виробничих і побутових приміщеннях, де недостатньо природного світла, і також для освітлення приміщень в темну пору року. При утворенні штучного освітлення необхідно забезпечити комфортний гігієнічні умови для зорової роботи і одночасно враховувати економічні показники.

В області освітлювальних приладів однією з новинок сьогодні є світлодіодні лампи. За принципом пристрою дане джерело освітлення принципово відрізняється від ламп розжарювання і люмінесцентних ламп, а за багатьма характеристиками - істотно їх перевершує.

Існує безліч різних конструкцій світлодіодних ламп: з прихованим радіатором, з розсіювачем або без, з різним розташуванням і кількістю світлодіодів, проте принципово світлодіодна лампа складається з, власне, світлодіодів, блоку живлення і радіатора охолодження.

Основні параметри світлодіодна лампа визначається, перш за все, характеристики сам світлодіоди, їх розташування, ефективність блок живлення, ефективність тепловідведення і світлопропускання розсіювач.

Розглянемо основні переваги світлодіодних ламп.

1. Світлодіодні лампи мають терміном служби до 100 000 годин, що в десятки разів більше, ніж у ламп розжарювання і люмінесцентних ламп.

2. Сучасні світлодіоди мають показник світловіддачі понад 100 лм / Вт. Це робить світлодіодні лампи кращого варіанту для вирішення завдань енергозбереження.

3. Світлодіодні лампи не мають затримки включення, як, наприклад, люмінесцентні лампи. Час включення для світлодіодних ламп зазвичай становить не більше 1 мілісекунди.

4. У світлодіодних лампах не міститься шкідливих речовин, їх легко утилізувати, не завдаючи шкоди екології.

5. Світлодіодні лампи зберігають стабільні світлові і колірні характеристики протягом усього терміну служби, світловий потік з плином часу практично не знижується.

6. У світлодіодних ламп відсутнє мерехтіння, властивий, наприклад, люмінесцентні лампи.

7. Світлові характеристики світлодіодних ламп практично не залежить від температури навколишнього середовища. Особливо гарні світлодіодні лампи в умовах низьких температур.

8. Світлодіодні лампи забезпечують високу контрастність і високий коефіцієнт передачі кольору Ra, що особливо важливо для застосування світлодіодних ламп в якості вуличних світильників.

9. Світлодіодні лампи можна застосовувати в умовах вібрації, що відкриває широкий спектр їх використання в якості світлодіодних ліхтарів на транспорті.

Єдиною перешкодою на шляху масового впровадження світлодіодних ламп поки що є їх ціна. Однак розрахунки показують, що вже зараз світлодіодні лампи окупаються протягом 2-3 років, а при новому будівництві, при необхідності плати за підключення потужностей навіть швидше. До того ж, з розвитком технологій ціни на світлодіодах знижується, в той час як електроенергія неухильно дорожчає. Все це робить світлодіодні лампи найперспективнішим джерелом світла на сьогоднішньому ринку.

Грецьке слово «люкс» означає світло, тому прилад, який вимірює освітленість у люксах (лк), називається люксметром (рис. 5.1). Вимірювальний пристрій використовується як у внутрішньому просторі приміщень, так і на відкритому повітрі. Для чого ж потрібен цей прилад і навіщо робити заміри освітленості? Вченими давно доведено, що дія слабкого або навпаки занадто сильного світла, що негативно впливає на головний мозок, викликаючи негативну реакцію та інших органів. Недостатнє освітлення у приміщеннях призводить до зниження працездатності, з'являється сонливість, увага розсіюється.

Рисунок 5.1 – Прилад для вимірювання світла (люксметр)

Показник освітленості для офісів становить в середньому 200-300 лк.

Розрахунок освітлення приміщення. Вхідними даними є розміри кімнати 7 метрів у довжину та 4 метри у ширину, висота приміщення 3 метри, мінімальна освітленість 750 лк, вид лампи світлодіод.

Відстань між світильниками або рядами світильників, (L'): визначають залежно від типу кривої сили світла світильника, представлено в таблиці 5.2.

Таблиця 5.2 – Відстань між світильниками

Тип кривої сили світла світильника	Відстань між світильниками або рядами світильників, L
К	(0,4–0,7) × h
Г	(0,8–1,1) × h
Д	(1,4–1,6) × h
Л	(1,6–1,8) × h
М	(1,8–2,6) × h

Для офісних приміщень рекомендується обирати варіант Г. Відстань до зони роботи 0,9 м, довжина звису лампи 0,2 м.

Відстань між світильниками 1,7 м. Виконаємо розрахунки за формулою 5.1:

$$h = H - (hp + hc), \quad (5.1)$$

де h - висота установки світильників над робочою поверхнею;

Н - висота приміщення, м;

hр - відстань робочої поверхні від підлоги, м;

hс - відстань (звис) світильника від стелі, м.

Відстань від крайніх світильників до стін приймають в межах $(0{,}3 - 0{,}5) \times$ L. Тоді відстань від крайніх ламп до стін 0,68 м. Кількість ламп в ряду для приміщення можна вирахувати з формули 5.2:

$$N = \frac{(\mathrm{A} - (0{,}68 \times 2))}{L}, \quad (5.2)$$

де N- кількість ламп в ряду для приміщення;

L - відстань між світильниками.

Таким чином, кількість ламп дорівнює 3,317. Округлимо це значення до 4 ламп у ряду, для відповідної ширини у 4 метра, це значення треба збільшити у $4/1{,}7$ разів, округлимо його до 2. У цілому отримаємо для освітлення приміщення нам потрібно 8 ламп. Цей розрахунок відповідає вимогам, щодо відстані між світильниками.

Перевірка наміченого варіанту освітлення нормативним вимогам. Для цього можна задіяти два методи: метод коефіцієнта використання і точковий метод. Метод коефіцієнта використання грунтується на визначенні відношення світлового потоку, що подає на розрахункову поверхню, по усьому світловому потоку, що випромінюється світильниками. Метод коефіцієнта використання можна задіяти у площі приміщення, без виділення яких-небудь окремих точок або зон, тобто для розрахунку загального рівномірного освітлення приміщень з досить світлими стелями, стінами і підлогою і за відсутності істотних затемнень. На методі коефіцієнта використання основані спрощені методи розрахунку освітлення по питомій потужності, Вт/м2, і деякі інші, що мають меншу точність у випадках, коли освітленість повинна створюватися в горизонтальній площині по усій площі приміщення, без виділення яких-небудь окремих точок або зон, т. е. для розрахунку загального рівномірного освітлення приміщень з досить світлими стелями, стінами і підлогою і за відсутності істотних затемнень. На методі коефіцієнта використання ґрунтовані спрощені методи розрахунку освітлення по питомій потужності, Вт/м2, і деякі інші, що мають меншу точність.

Для розрахунку освітленості виробничих приміщень рекомендується метод коефіцієнта використання. По цьому методу необхідна величина світлового потоку ламп в кожному світильнику визначається по формулі 5.3:

$$\Phi = \frac{E \times S \times K \times Z}{N \times \eta}, \qquad (5.3)$$

де E - задана мінімальна освітленість, лк;

K - коефіцієнт запасу;

S - освітлювана площа, м^2;

z - коефіцієнт нерівномірності освітлення;

N - число світильників;

η - коефіцієнт використання світлового потоку в долях одиниці.

Вимірювання світлового потоку від джерела світла провадиться за допомогою спеціальних приладів - сферичних фотометрів (рис. 5.2). Світловий потік в світлодіодах варіюється в середньому від 80 до 150 Лм з 1 Вт.

Сферичний фотометр — прилад, що є сферою з внутрішнім покриттям, що має коефіцієнт відображення, близький до 1. Досліджуване джерело світла поміщається в центр сфери і за допомогою фотоелемента, вмонтованого в стінку сфери і покритого фільтром з кривою пропускання, що дорівнює кривій спектральної чутливості ока, вимірюється сигнал, пропорційний освітленості фотоелемента, яка, у свою чергу, в даному пристрої пропорційна світловому потоку джерела світла.

Рисунок 5.2 – Прилад для вимірювання світлового потоку

Індекс приміщення знаходять за формулою 5.4.

$$\mathrm{i} = \frac{\mathrm{A} \times \mathrm{B}}{\mathrm{h} \times (\mathrm{A}+\mathrm{B})}, \tag{5.4}$$

де A і B - довжина і ширина приміщення, м.

Таким чином, індекс приміщення становить 1,4. Потужність освітлювальної установки визначають за формулою 5.5.

$$\mathrm{P}y = \mathrm{P}l \times N, \tag{5.5}$$

де Pl - потужність вибраної стандартної лампи.

Параметр η вираховуються за допомогою значень і та коефіцієнтів відбиття від підлоги, стелі та стін. Три останні мають табличні значення і дорівнюють: $\mathrm{Pc1} = 50,\ \mathrm{Pc} = 50$ та $\mathrm{Pp} = 10$. Треба вказати, що Pc1 – коефіцієнт стелі, Pc – коефіцієнт стін, а Pp –коеціцієнт підлоги.

Отже, дивлячись на ці дані можна сказати, що η = 87%. Коефіцієнт запасу для світлодіодних ламп дорівнює 1. Коефіцієнт нерівномірності освітлення дорівнює 1.1. Освітлювана площа 28 м^2. Тоді світловий потік Ф для 1 лампи буде 3300 лм.

Була обрана лампа LED E27 34W 162pcs NW T80 SMD2835, яка показана на рис 5.2. Ця лампа має світовий потік у 3400 лм, що задовольняє обмеження. Загальна потужність Ру буде дорівнювати 272W.

Рисунок 5.2 – Світлодіодна лампа

Рекомендації для комфортності і безпеки праці у приміщеннях. Будівлі та приміщення, в яких розміщені робочі місця, мусять відповідати вимогам нормативно-технічної та експлуатаційної документації виробника персональних комп'ютерів ДСанПіН 3.3.2-007-98 та Правил [18]. Будівлі та приміщення, де розміщені робочі місця операторів, мають бути не нижче другого ступеня вогнестійкості [19]. Всі будівлі і приміщення, в яких знаходяться робочі місця, повинні бути визначено відповідно до класу зони згідно з НПАОП 40.1-1.01-97 [20]. Таке позначення повинно бути нанесено на вхідних дверях кожного приміщення. Не дозволяється розташування приміщень з робочими місцями у підвалах і цокольних поверхах. Забороненим є розташування приміщень категорій А і Б, а також виробництв з мокрими технологічними процесами поруч з приміщеннями, в яких розташовуються робочі місця, а також над ними чи під ними. При цьому площа приміщення має бути не менше 6,0 кв. м. із розрахунку на одне робоче місце, а об'єм – не менше 20,0 куб. м.

Віконні прорізи приміщень для роботи з персональними комп'ютерами повинні бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки. Для внутрішнього оздоблення приміщень, в яких знаходяться комп'ютери необхідно використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6. Покриття підлоги повинно бути матовим з коефіцієнтом відбиття 0,3-0,5. Поверхня підлоги має бути рівною, неслизькою, з антистатичними властивостями. Забороняється для оздоблення інтер'єру приміщень з персональними комп'ютерами використовувати полімерні матеріали (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини. Полімерні матеріали для внутрішнього оздоблення приміщень з персональними комп'ютерами можуть використовуватися при наявності дозволу органів та установ державної санітарно-епідеміологічної служби. Приміщення можуть обладнуватися шафами

для зберігання документів, магнітних дисків, полицями, стелажами, тумбами тощо з урахуванням вимог до площі приміщень [21].

У приміщеннях з джерелами шкідливих виробничих факторів робочі місця операторів повинні розміщуватися в ізольованих кабінах, які обладнані повітрообміном.

Заземлені конструкції, які знаходяться в приміщеннях, де розташовані робочі місця (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), повинні бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу. Приміщення, в яких розміщені робочі місця, повинні бути оснащені системою автоматичної пожежної сигналізації і вогнегасниками згідно з вимогами чинного законодавства України [22]. Проходи до засобів пожежогасіння мають бути вільними.

У приміщеннях, де розташовані робочі місця, необхідно щоденно робити вологе прибирання. Також, ці приміщення мають бути оснащені аптечками першої медичної допомоги, а при них повинні бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження [23].

5.3 Безпека під час роботи з персональним комп’ютером

Перед тим, як включити комп'ютер, необхідно приділити пару хвилин наступним діям [22].

1. Потрібно переконатися в тому, що в зоні досяжності відсутня оголені дроти і різні шнури. Вони не тільки заважають роботі, але і несуть потенційну небезпеку в разі короткого замикання.

2. Не можна починати роботу на техніці з видимим пошкодженням. У разі виявлення тріщини на корпусі або пошкоджень іншого роду, потрібно звернутися за допомогою в сервісний центр. Це ж ставлюся до ПК з несправним індикатор включення / вимикання.

3. Предмети на столі не повинні заважати огляду, користування мишкою і клавіатурою. Поверхня екрану повинна бути абсолютно чистою.

4. На системному блоці не повинно знаходитися жодних предметів, так як в результаті вібрацій може порушитися робота пристрою. Потрібно переконатися в тому, що ніякі сторонні предмети не заважають роботі системи охолодження.

5. Неприпустимо включати персональний комп'ютер в подовжувачі і розетки, в яких відсутня заземлювальна шина.

6. Забороняється починати роботу в приміщеннях з підвищеною вологістю, а також в разі, якщо поруч присутні відкриті джерела вологості (калюжі, мокра підлога). Включити техніку можна лише після повного висихання навколишніх предметів.

7. Неприпустимо часто вмикати і вимикати комп'ютер протягом робочого дня без особливої потреби. Система просто не справляється з необхідністю швидко згортати всі процеси.

5.4 Визначення режиму праці та відпочинку за комп’ютером

При організації праці, що пов’язана з використанням персональних комп’ютерів, для збереження здоров’я працюючих, запобігання професійним захворювання і підтримки працездатності слід передбачити регламентовані перерви для відпочинку [16].

Режими праці і відпочинку мають передбачати додаткові нетривалі перерви в періоди, що передують появі об’єктивних і суб’єктивних ознак стомлення і зниження працездатності. За основну роботу з персональним комп’ютером слід вважати таку, що займає не менше 50% часу впродовж робочої зміни [24]. Відповідно до п. 5.3 ДСанПіН 3.3.2.007-98 протягом дня мають передбачатися [18]:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Тривалість обідньої перерви визначається чинним законодавством про працю і правилами внутрішнього трудового розпорядку [25].

Пунктом 5.8 ДСанПіН 3.3.2.007-98 встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні залежно від характеру праці [26]:

– для розробників програм слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за персональним комп'ютером;

– для операторів персональних комп'ютерів слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за персональним комп'ютером.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви**,** тривалість безперервної роботи з персональним комп'ютером не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин (п. 5.9 та п. 5.10 ДСанПіН 3.3.2.007-98) [18].

З метою зменшення негативного впливу монотонності є доцільним застосовувати чергування операцій обробки тексту і числових даних (зміна змісту роботи), чергування вводу даних та редагування текстів.

Для зниження нервово-емоційного напруження, стомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільні деякі перерви використовувати для виконання комплексу вправ, приклади яких також наведено в ДСанПіН 3.3.2.007-98 [18]. В окремих випадках — при хронічних скаргах працюючих на зорове стомлення, незважаючи на дотримання санітарно-гігієнічних вимог до режимів праці і відпочинку, а також застосування засобів локального захисту очей, допускається індивідуальний підхід до обмеження часу робіт з персональним комп'ютером, зміни характеру праці, чергування з іншими видами діяльності, не пов'язаними з персональним комп'ютером [27].

Висновки до розділу 5

Під час виконання спеціальної частини з охорони праці було проведено аналіз вимог до організації заходів техніки безпеки та охорони праці під час роботи за комп'ютером. Було виконано такі необхідні завдання: організація та обладнання робочого місця, створення необхідних умов освітлення, формування правил безпеки під час роботи з персональним комп'ютером, визначення режиму праці та відпочинку за комп'ютером.

Під час роботи за персональним комп'ютером людина швидко втомлюється та від цього страждає опорно-руховий, зоровий апарат, нервова система и в окремих випадках психічне здоров'я. Тому було сформовані правила безпеки під час роботи з персональним комп'ютером є актуальними на сьогодні. А визначення режиму праці та відпочинку дозволить користувачам підтримувати свою працездатність на високому рівні.Також було описано необхідні заходи для створення відповідних умов освітлення. Адже раціональне освітлення робочих місць створює сприятливі умови для забезпечення безпечної праці, поліпшення якості продукції та підвищення продуктивності праці, впливає психологічно на людину, створює гарний настрій: бадьорий у працюючих.

ВИСНОВКИ

Підтримка водного балансу в організмі людини є важливим процесом, який допомагає нам залишатися здоровими. Тіло людини приблизно на 60% складається з води. Вода необхідна для нашого виживання. Вона відіграє важливу роль у травленні, роботі клітин, секреції гормонів тощо. Порушення водного балансу може призвести до ряду проблем зі здоров'ям.

Підтримуючи водний баланс, ми допомагаємо нашому організму виводити шкідливі токсини та хімічні речовини. Коли водний баланс порушується, ці токсини та хімічні речовини можуть накопичуватися в організмі. Це може призвести до ряду проблем зі здоров'ям, включаючи проблеми з нирками, печінкою та іншими органами людського організму.

Підтримка водного балансу також важлива для нашого загального самопочуття. Порушення водного балансу може призвести до ряду фізичних травм та психологічних проблем. Процес підтримки водного балансу надзвичайно важливий для загального здоров'я та благополуччя людини. Саме тому розробка мобільного застосунку підтримки водного балансу в організмі людини є актуальною. Використання технологій для розробки застосунків та моделей машинного навчання дають змогу реалізовувати дійсно корисні та актуальні на сьогоднішній час продукти, які значно полегшують життя людей та дозволяють вирішувати ті пробеми, які кілька століть тому були для людства максимально гострими.

Метою бакалаврської кваліфікаційної роботи було покращення водного балансу та самопочуття користувачів за рахунок систематичного вживання необхідної норми води, шляхом розробки мобільного застосунку на базі операційної системи Android. У процесі виконання проєкту було розглянено особливості операційної системи Android, мов програмування Kotlin та Python, робочого середовища Android Studio. Також в результаті виконання роботи було створено та навчено модель простої лінійної регресії, задля реалізації

прогнозування оцінки самопочуття людини в залежності від випитої кількості води. Точність прогнозованої оцінки самопочуття користувача складає 99%. Також було описано бібліотеку TensorFlow, яку слід використовувати при розгортанні моделей машинного навчання на мобільних пристроях.

Для досягнення поставленої мети було вирішено такі задачі:

– виконано аналіз предметної сфери;

– досліджено та проаналізовано наявні аналоги мобільних застосунків для підтримки водного балансу;

– сформувано вимоги до функціоналу розроблювального застосунку;

– розглянено принципи розробки Android застосунків;

– обрано технології та методи розробки мобільного застосунку підтримки водного балансу в організмі людини;

– спроєктовано мобільний застосунок;

– реалізовано мобільний застосунок.

Результатом роботи є мобільний застосунок підтримки водного балансу в організмі людини із автоматизованим розрахунком індивідуальної норми води на основі введеної інформації користувачем. Розроблений застосунок надає змогу додавати випиту людиною порцію води та спостерігати за власним прогресом на круговій шкалі прогресу. Також мобільний застосунок виводить повідомлення про нагадування випити воду, прогнозовану оцінку фізичного самопочуття людини в залежності від випитої кількості рідини протягом доби та лінійну діаграму із відповідними заданими значеннями.

Мобільний застосунок розроблено згідно з функціональними вимогами та виконує поставлену задачу бакалаврської роботи, але може бути масштабованим в майбутньому для розширення технічних можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Water Intake, Water Balance and the Elusive Daily Water Requirement. National Library of Medicine: вебсайт. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6315424/ (дата звернення: 10.05.2022).

2. Lawrence E. Armstrong and Evan C. Johnson, Nutrients. стаття: Water Intake, Water Balance and the Elusive Daily Water Requirement, 2018, p. 11. URL: http://www.acsm.org/docs/publications/Roundtable%20on%20Hydration%20and%20Physical%20Activity.pdf.

3. Розробка мобільних застосунків: вебсайт. URL: https://webcase.com.ua/uk/razrabotka-mobilnyh-prilozhenij/ (дата звернення: 05.05.2022).

4. J. Ward, C. Haase. Kotlin Mullets: вебсайт. URL: https://resources.jetbrains.com (дата звернення: 10.05.2022).

5. Introduction to Linear Regression: вебсайт. URL: https://www.vedantu.com/maths/linear-regression (дата звернення: 11.05.2022).

6. Simple Linear Regression Examples. *Blog For Data-Driven Business*: вебсайт. URL: https://www-intellspot-com.cdn.ampproject.org/v/s/www.intellspot.com/linear-regression-examples (дата звернення: 10.05.2022).

7. Написання ML моделі за допомогою Keras: вебсайт. URL: http://mikrotik.kpi.ua/index.php/courses-list/ml/124-using-keras-via-tf-keras-in-tensorflow-lecture-5 (дата звернення: 15.05.2022).

8. Laurence Moroney. AI and Machine Learning for Coders. O'Reilly Media, Inc, 2020. p. 67.

9. Проєктування мобільного застосунку. *Wellsoft IT Company:* вебсайт. URL: https://wellsoft.pro/blog/proektirovanie-mobilnogo-prilozheniya (дата звернення: 15.05.2022).

10. Що таке гайдлайни мобільних застосунків та для чого вони потрібні? *Medium*: вебсайт. URL: https://medium.com/ (дата звернення: 17.05.2022).

11. Casa, D.J., Clarkson, P.M. & Roberts, W.O., MD, FACSM. стаття: American College of Sports Medicine roundtable on hydration and physical activity: Consensus statements, 2005, p. 120. URL: http://www.acsm.org/docs/publications/Roundtable%20on%20Hydration%20and%20Physical%20Activity.pdf.

12. How to calculate how much water you should drink? University of Missouri System: вебсайт. URL: https://www.umsystem.edu/totalrewards/wellness/how-to-calculate-how-much-water-you-should-drink (дата звернення: 17.05.2022).

13. The truth about how much water you should really drink. U. S. News & World Report: вебсайт. URL: https://www.umsystem.edu/totalrewards/wellness/how-to-calculate-how-much-water-you-should-drink (дата звернення: 09.05.2022).

14. Створення повідомлень. *Документація для розробників:* вебсайт. URL: https://developer.android.com/training/notify-user/build-notification (дата звернення: 05.05.2022).

15. Огляд повідомлень на Android. *Gabriel Tanner:* вебсайт. URL: https://gabrieltanner.org/blog/android-notifications-overview/ (дата звернення: 05.05.2022).

16. Жидецький В.Ц. Основи охорони праці: підручник. Київ, 2002.320 с.

17. Природне і штучне освітлення : вебсайт. URL: https://dnaop.com/html/45036/doc (дата звернення: 23.05.2022).

18. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98: Постанова від 10 грудня 1998 р. №7. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення: 29.05.2022).

19. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин : вебсайт. URL: http://zakon3.rada.gov.ua/laws/show/z0382-99 (дата звернення: 24.05.2022).

20. Правила безпечної експуатації електроустановок НПАОП 40.1-1.01-97: вебсайт. URL: https://dnaop.com/ (дата звернення: 29.05.2022).

21. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : вебсайт. URL: http://zakon3.rada.gov.ua/laws/show/z0508-18 (дата звернення: 25.05.2022).

22. ДСТУ 2293-99 Охорона праці. Терміни та визначення основних понять. Київ, 1999. 24 с. (Інформація та документація).

23. Москальова В. М. Основи охорони праці: підручник. Київ, 2005. 666 с.

24. Ткачук К. Н., Халімовський М. О., Зацарний В.В., та інші. Основи охорони праці: підручник. Київ, 2006.444 с.

25. Гандзюк М. П., Желібо Е. П., Халімовський М. О. Основи охорони праці : підручник. Київ, 2003.405 с.

26. Державні санітарні правила і норми роботи з ВДТ ЕОМ ДСанПІН 3.3.2.007-98 : вебсайт. URL: http://mozdocs.kiev.ua/view.php?id=2445 (дата звернення: 23.05.2022).

27. Катренко Л.А., Катренко А. В. Охорона праці в галузі комп'ютинґу: підручник. Львів: "Магнолія 2006", 2012. 544 с.

ДОДАТОК А

Код програмного забезпечення

SplashFragment.kt

```
package com.droplet.ui.splash

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import com.droplet.R
import com.droplet.databinding.FragmentSplashBinding
import com.droplet.ui.base.BaseFragment
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch

class SplashFragment : BaseFragment() {
    private lateinit var binding: FragmentSplashBinding

    private val viewModel by viewModels<SplashViewModel>()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentSplashBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        lifecycleScope.launch {
            delay(1000)
            checkNextScreen()
        }

    }

    private fun checkNextScreen() {
        if (viewModel.checkUserLogIn()) {

findNavController().navigate(R.id.action_splashFragment_to_mainFragment)
        } else {

findNavController().navigate(R.id.action_splashFragment_to_introFragment)
        }
    }

}
```

SplashViewModel

```
package com.droplet.ui.splash

import com.droplet.data.SharedPrefHelper
import com.droplet.ui.base.BaseViewModel

class SplashViewModel: BaseViewModel() {

    fun checkUserLogIn(): Boolean {
        return SharedPrefHelper.getIsLogIn()
    }
}
```

fragment_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_splash"
    tools:context=".ui.splash.SplashFragment">

    <ImageView
        android:id="@+id/droplet_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo_drop" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

IntroFragment.kt

```
package com.droplet.ui.registration.intro

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.findNavController
import com.droplet.R
import com.droplet.databinding.FragmentIntroBinding
import com.droplet.ui.base.BaseFragment
import com.droplet.ui.splash.SplashViewModel

class IntroFragment: BaseFragment() {
    private lateinit var binding: FragmentIntroBinding

    private val viewModel by viewModels<IntroViewModel>()

    override fun onCreateView(
```

```
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentIntroBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        updateUi()
    }

    private fun updateUi() {
        binding.apply {
            btnGetStarted.setOnClickListener {

findNavController().navigate(R.id.action_introFragment_to_paramsFragment)
            }
        }
    }
}
```

IntroViewModel.kt

```
package com.droplet.ui.registration.intro

import com.droplet.ui.base.BaseViewModel

class IntroViewModel: BaseViewModel() {

}
```

fragment_intro.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorWhite"
    tools:context=".ui.registration.intro.IntroFragment">

    <ImageView
        android:id="@+id/background"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:background="@drawable/background_vector"
        app:layout_constraintBottom_toTopOf="@+id/backgroundGuideline"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/logoContainer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:gravity="center_horizontal"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="@+id/background"
    app:layout_constraintEnd_toEndOf="@+id/background"
    app:layout_constraintStart_toStartOf="@+id/background"
    app:layout_constraintTop_toTopOf="@+id/background"
    app:layout_constraintVertical_bias="0.39">

    <ImageView
        android:id="@+id/droplet_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout constraintHorizontal bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.2"
        app:srcCompat="@drawable/logo_drop" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/montserrat_thin"
        android:text="Droplet"
        android:textColor="#FFFFFF"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.483" />
</LinearLayout>


<com.google.android.material.button.MaterialButton
    android:id="@+id/btnGetStarted"
    style="@style/ButtonGradient"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="66dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="66dp"
    android:layout_marginBottom="70dp"
    android:text="Get started"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/background"
    app:layout_constraintVertical_bias="1.0" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/backgroundGuideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.67" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

ParamsFragment.kt

```
package com.droplet.ui.registration.params

import android.app.TimePickerDialog
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.findNavController
import com.droplet.R
import com.droplet.databinding.FragmentParamsBinding
import com.droplet.ui.base.BaseFragment
import com.droplet.utils.Utils.formatTime

class ParamsFragment : BaseFragment() {


    private var weight: String = ""
    private var workTime: String = ""


    private lateinit var binding: FragmentParamsBinding

    private val viewModel by viewModels<ParamsViewModel>()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentParamsBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        initUi()
    }


    private fun initUi() {
        binding.apply {
            etWakeUpTime.setOnClickListener {
                showTimePickerDialog { _, hourOfDay, minute ->
                    onWakeTimeSelected(hourOfDay, minute)
                }
            }
            etSleepTime.setOnClickListener {
                showTimePickerDialog { _, hourOfDay, minute ->
                    onSleepTimeSelected(hourOfDay, minute)
                }
            }

            button.setOnClickListener {
```

```
                weight = etWeight.editText!!.text.toString()
                workTime = etWorkTime.editText!!.text.toString()

                if(weight.isEmpty()){
                    etWeight.error = "Weight is required!"
                }
                else if(workTime.isEmpty()) {
                    etWorkTime.error = "WorkTime is required!"
                }
                else{

                if(weight.toInt() > 200 || weight.toInt() < 40) {
                    etWeight.error = "Please enter the correct weight value"
                }
                else if(workTime.toInt() > 500 || workTime.toInt() < 0) {
                    etWorkTime.error = "Please enter the correct workTime value"
                }
                else {
                    viewModel.calculateIntake(weight.toInt(), workTime.toInt())

findNavController().navigate(R.id.action_paramsFragment_to_waterControlFragment)
                }

            }}
        }
    }

    private fun onWakeTimeSelected(hourOfDay: Int, minute: Int) {
        val wakeTimeFormatted = formatTime(hourOfDay, minute)
        binding.etWakeUpTime.setText(wakeTimeFormatted)
    }

    private fun onSleepTimeSelected(hourOfDay: Int, minute: Int) {
        val sleepTimeFormatted = formatTime(hourOfDay, minute)
        binding.etSleepTime.setText(sleepTimeFormatted)
    }

    private fun showTimePickerDialog(timeSelectedListener:
TimePickerDialog.OnTimeSetListener) {
        val timePicker = TimePickerDialog(
            context,
            R.style.TimePickerTheme,
            timeSelectedListener,
            12,
            10,
            false
        )
        timePicker.show()
    }
}
```

ParamsViewModel.kt

```
package com.droplet.ui.registration.params

import com.droplet.data.SharedPrefHelper
import com.droplet.ui.base.BaseViewModel
```

```
class ParamsViewModel : BaseViewModel() {

    fun calculateIntake(weight: Int, workTime: Int) {
        val total = ((weight * 30) + ((workTime / 30) * 355))
        saveTotalToPref(total)
    }

    private fun saveTotalToPref(total: Int) {
        SharedPrefHelper.saveTotalAmount(total)
        SharedPrefHelper.setIsLogIn(true)
    }

}
```

fragment_params.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    tools:context=".ui.registration.params.ParamsFragment">


    <TextView
        android:id="@+id/textInfo"
        style="@style/TextStyleNormal"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="0dp"
        android:layout_marginEnd="16dp"
        android:text="Please provide some information to customize the
experience for you! "
        android:textAlignment="center"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.495"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/guidelineTop" />


    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/paramsContainer"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textInfo"
        app:layout_constraintVertical_bias="0.100000024">

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/etWeight"
            style="@style/TextInput"
```

```
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:errorEnabled="true"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <com.google.android.material.textfield.TextInputEditText
        style="@style/TextInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/weight_hint"
        android:imeOptions="actionNext"
        android:inputType="numberSigned"
        android:maxLength="3" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/etWorkTime"
    style="@style/TextInput"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:errorEnabled="true"
    android:layout_marginTop="24dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/etWeight">

    <com.google.android.material.textfield.TextInputEditText
        style="@style/TextInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Your daily work out time in min/day"
        android:imeOptions="actionDone"
        android:inputType="numberSigned"
        android:maxLength="3" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/layoutWakeUpTime"
    style="@style/TextInput"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:layout_marginEnd="8dp"
    app:layout_constraintEnd_toStartOf="@+id/layoutSleepTime"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/etWorkTime">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/etWakeUpTime"
        style="@style/TextInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:clickable="true"
        android:focusable="false"
        android:hint="@string/wakeup_hint"
        android:imeOptions="actionNext"
        android:inputType="numberSigned" />
</com.google.android.material.textfield.TextInputLayout>
```

```
        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/layoutSleepTime"
            style="@style/TextInput"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="24dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/layoutWakeUpTime"
            app:layout_constraintTop_toBottomOf="@+id/etWorkTime">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/etSleepTime"
                style="@style/TextInput"
                android:layout width="match parent"
                android:layout_height="wrap_content"
                android:clickable="true"
                android:focusable="false"
                android:hint="@string/sleeping_hint"
                android:imeOptions="actionDone"
                android:inputType="numberSigned" />
        </com.google.android.material.textfield.TextInputLayout>

    </androidx.constraintlayout.widget.ConstraintLayout>

    <com.google.android.material.button.MaterialButton
        android:id="@+id/button"
        style="@style/ButtonGradient"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_marginStart="66dp"
        android:layout_marginEnd="66dp"
        android:layout_marginBottom="70dp"
        android:text="Submit"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guidelineTop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.2" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

WaterControlFragment.kt

```
package com.droplet.ui.watercontrol


import android.graphics.Color
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
```

```
import com.droplet.databinding.FragmentWaterControlBinding
import com.droplet.ui.base.BaseFragment
import com.hookedonplay.decoviewlib.charts.SeriesItem
import com.hookedonplay.decoviewlib.events.DecoEvent


class WaterControlFragment : BaseFragment() {

    private lateinit var binding: FragmentWaterControlBinding

    private val viewModel by viewModels<WaterControlViewModel>()

    private var intakeChartIndex: Int? = null

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentWaterControlBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        updateUi()
    }

    private fun updateUi() {
        binding.apply {
            initChart()

            goalText.text = "GOAL ${viewModel.getTotal()}"

            addDrinkButton.setOnClickListener {
                when (tabLayout.selectedTabPosition) {
                    0 -> handleIntake(50)
                    1 -> handleIntake(100)
                    2 -> handleIntake(250)
                    3 -> handleIntake(350)
                    4 -> handleIntake(500)
                }
            }
        }
    }

    private fun handleIntake(intake: Int) {
        viewModel.increaseTotalIntake(intake)
        updateChartIntake()
    }

    private fun initChart() {
        val total = viewModel.getTotal()
        binding.apply {
            val totalItem = SeriesItem.Builder(Color.parseColor("#E9E9E9"))
                .setRange(0f, total.toFloat(), total.toFloat())
                .build()
            val intakeItem = SeriesItem.Builder(Color.parseColor("#AADAA7"))
                .setRange(0f, total.toFloat(), 0f)
```

```
                .build()

            val totalChartIndex: Int = dynamicArcView.addSeries(totalItem)
            intakeChartIndex = dynamicArcView.addSeries(intakeItem)

            dynamicArcView.addEvent(
                DecoEvent.Builder(total.toFloat())
                    .setIndex(totalChartIndex)
                    .setDuration(0)
                    .build()
            )
            updateChartIntake()
        }
    }

    private fun updateChartIntake() {
        val newIntake = viewModel.getIntake()

        binding.apply {
            dynamicArcView.addEvent(
                DecoEvent.Builder(newIntake.toFloat())
                    .setIndex(intakeChartIndex!!)
                    .setDuration(1000)
                    .build()
            )
            waterLevelText.text = "${viewModel.getIntake()} ml"
        }
    }

}
```

WaterControlViewModel.kt

```
package com.droplet.ui.watercontrol

import com.droplet.data.SharedPrefHelper
import com.droplet.ui.base.BaseViewModel


class WaterControlViewModel : BaseViewModel() {

    fun increaseTotalIntake(amount: Int) {
        val newAmount = SharedPrefHelper.getTotalIntake().plus(amount)
        SharedPrefHelper.saveTotalIntake(newAmount)
    }

    fun getIntake(): Int {
        return SharedPrefHelper.getTotalIntake()
    }

    fun getTotal(): Int {
        return SharedPrefHelper.getTotalAmount()
    }

}
```

fragment_water_control.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:background="#FFFFFF"
tools:context=".ui.watercontrol.WaterControlFragment">


<FrameLayout
    android:id="@+id/frameLayout"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginBottom="60dp"
    android:background="@drawable/background_splash"
    app:layout_constraintBottom_toBottomOf="@+id/topContentContainer"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

</FrameLayout>

<LinearLayout
    android:id="@+id/topContentContainer"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="55dp"
        android:fontFamily="@font/montserrat_regular"
        android:text="Your Daily Water Intake"
        android:textColor="#FFFFFF"
        android:textSize="24sp"
        android:textStyle="bold" />

    <com.google.android.material.card.MaterialCardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="33dp"
        android:layout_marginTop="29dp"
        android:layout_marginEnd="33dp"
        android:layout_marginBottom="16dp"
        app:cardElevation="4dp">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/white">

            <TextView
                android:id="@+id/titleText"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginStart="16dp"
                android:layout_marginTop="32dp"
                android:fontFamily="@font/montserrat_regular"
```

```
                    android:text="Mary, you’re almost there!"
                    android:textColor="#474646"
                    android:textSize="18sp"
                    android:textStyle="bold"
                    app:layout_constraintEnd_toStartOf="@+id/iconDrop"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toTopOf="parent" />

                <TextView
                    android:id="@+id/descriptionText"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="16dp"
                    android:layout_marginTop="10dp"
                    android:layout marginBottom="34dp"
                    android:fontFamily="@font/montserrat_regular"
                    android:text="Drink some more water to see how your weight
changes!"
                    android:textAlignment="textStart"
                    android:textColor="#9A9A9A"
                    android:textSize="14sp"
                    android:textStyle="bold"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintEnd_toStartOf="@+id/iconDrop"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@+id/titleText" />

                <ImageView
                    android:id="@+id/iconDrop"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="32dp"
                    android:layout_marginEnd="16dp"
                    android:layout_marginBottom="32dp"
                    android:src="@drawable/ic_thinking_droplet"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintTop_toTopOf="parent" />
            </androidx.constraintlayout.widget.ConstraintLayout>
        </com.google.android.material.card.MaterialCardView>
    </LinearLayout>

    <TextView
        android:id="@+id/waterLevelText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="26dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="34dp"
        android:fontFamily="@font/montserrat_regular"
        android:text="0 ml"
        android:textAlignment="center"
        android:textColor="#AADAA7"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="@+id/dynamicArcView"
        app:layout_constraintEnd_toEndOf="@+id/dynamicArcView"
        app:layout_constraintStart_toStartOf="@+id/dynamicArcView"
        app:layout_constraintTop_toTopOf="@+id/dynamicArcView" />
```

```
<TextView
    android:id="@+id/goalText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="36dp"
    android:layout_marginEnd="36dp"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="34dp"
    android:fontFamily="@font/montserrat_regular"
    android:textAlignment="center"
    android:text="GOAL 2000 ml"
    android:textColor="#474646"
    android:textSize="16sp"
    android:textStyle="bold"
    app:layout constraintEnd toEndOf="@+id/dynamicArcView"
    app:layout_constraintStart_toStartOf="@+id/dynamicArcView"
    app:layout_constraintTop_toBottomOf="@+id/waterLevelText" />

<com.hookedonplay.decoviewlib.DecoView
    android:id="@+id/dynamicArcView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="110dp"
    android:layout_marginEnd="110dp"
    app:dv_lineWidth="20dp"
    app:layout_constraintBottom_toTopOf="@+id/tabLayout"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/topContentContainer" />


<com.google.android.material.button.MaterialButton
    android:id="@+id/addDrinkButton"
    style="@style/ButtonGradient"
    android:layout_width="0dp"
    android:layout_height="48dp"
    android:layout_marginStart="66dp"
    android:layout_marginEnd="66dp"
    android:layout_marginBottom="40dp"
    android:text="Add a drink"
    android:textColorHint="#000000"
    app:elevation="10dp"
    app:icon="@drawable/ic_add"
    app:iconGravity="textStart"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent" />

<com.google.android.material.tabs.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:background="@color/white"
    app:tabTextColor = "#191F0E"
    android:layout_marginStart="1dp"
    android:layout_marginEnd="1dp"
    android:layout_marginBottom="80dp"
    app:layout_constraintBottom_toBottomOf="@+id/addDrinkButton"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
```

```
        app:layout_constraintStart_toStartOf="parent">

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="50" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="100" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout height="wrap content"
            android:text="250" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="350" />

        <com.google.android.material.tabs.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="500" />
    </com.google.android.material.tabs.TabLayout>


</androidx.constraintlayout.widget.ConstraintLayout>
```

StatisticFragment.kt

```
package com.droplet.ui.statistic

import android.annotation.SuppressLint
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import com.droplet.databinding.FragmentStatisticBinding
import com.droplet.extensions.showToast
import com.droplet.extensions.visible
import com.droplet.ui.base.BaseFragment
import com.github.mikephil.charting.components.XAxis
import com.github.mikephil.charting.data.Entry
import com.github.mikephil.charting.data.LineData
import com.github.mikephil.charting.data.LineDataSet
import org.tensorflow.lite.Interpreter
import java.io.FileInputStream
import java.io.IOException
import java.nio.MappedByteBuffer
import java.nio.channels.FileChannel

class StatisticFragment : BaseFragment() {

    private lateinit var binding: FragmentStatisticBinding

    private val viewModel by viewModels<StatisticViewModel>()
```

```
    private var tflite: Interpreter? = null

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentStatisticBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        initUi()

    }

    @SuppressLint("SetTextI18n")
    private fun initUi() {
        binding.apply {
            chart.data = LineData()
            toolbar.title.text = "Statistics and forecasting"

            try {
                tflite = Interpreter(loadModelFile())
            } catch (ex: Exception) {
                ex.printStackTrace()
            }
            btnPred.setOnClickListener {
                if (et.text.isNotEmpty() && et.text.isNotBlank()) {
                    val prediction = doInference(et.text.toString())
                    println(prediction)
                    hw.text = prediction.toString()

                    drawChart(viewModel.getChartEntries())
                } else {
                    context?.showToast("value can not be empty")
                }

            }
        }
    }



    @Throws(IOException::class)
    private fun loadModelFile(): MappedByteBuffer {
        val fileDescriptor = requireContext().assets!!.openFd("degree.tflite")
        val inputStream = FileInputStream(fileDescriptor.fileDescriptor)
        val fileChannel = inputStream.channel
        val startOffset = fileDescriptor.startOffset
        val declareLength = fileDescriptor.declaredLength
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,
declareLength)
    }


    private fun doInference(inputString: String): Float {
        val inputVal = FloatArray(1)
```

```
        inputVal[0] = inputString.toFloat()
        val output = Array(1) {
            FloatArray(
                1
            )
        }
        tflite?.run(inputVal, output)
        return output[0][0]
    }

    private fun drawChart(entries: List<Entry>) {
        binding.apply {
            chart.visible()

            with(chart) {
                setDrawGridBackground(false)
                description = null
                legend.isEnabled = false
                isDragEnabled = true
                setPinchZoom(false)
                animateX(500)
                setGridBackgroundColor(android.R.color.white)
                setScaleEnabled(true)
            }

            with(chart.xAxis) {
                isEnabled = true
                setCenterAxisLabels(false)
                setAvoidFirstLastClipping(false)
                setDrawLimitLinesBehindData(false)
                setDrawGridLines(false)
                position = XAxis.XAxisPosition.BOTTOM
                textSize = 11f
            }

            with(chart.axisLeft) {
                setDrawGridLines(false)
                textSize = 11f
            }

            with(chart.axisRight) {
                setDrawGridLines(false)
                isEnabled = false
            }

            val dataSets = LineDataSet(entries, "label")

            with(dataSets) {
                setDrawCircles(true)
                setDrawIcons(false)
                lineWidth = 2.5f
                setDrawValues(false)
            }
            val lineData = LineData(dataSets)
            chart.data = lineData
            chart.invalidate()
        }

    }

}
```

StatisticViewModel.kt

```
package com.droplet.ui.statistic

import com.droplet.ui.base.BaseViewModel
import com.github.mikephil.charting.data.Entry


class StatisticViewModel : BaseViewModel() {

    private val dataSet = mapOf(
        250 to 1,
        500 to 2,
        750 to 3,
        1000 to 4,
        1250 to 5,
        1500 to 6,
        1750 to 7,
        2000 to 8,
        2250 to 9,
        2500 to 10,
        2750 to 11,
        3000 to 12
    )

    fun getChartEntries(): List<Entry> {
        return dataSet.entries.map { Entry(it.key.toFloat(), it.value.toFloat())
}
    }
}
```

fragment_statistic.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    tools:context=".ui.statistic.StatisticFragment">


    <include
        android:id="@+id/toolbar"
        layout="@layout/toolbar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <com.google.android.material.card.MaterialCardView
        android:id="@+id/materialCardView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="30dp"
        app:cardElevation="4dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/toolbar">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/white">

            <TextView
                android:id="@+id/titleText"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginStart="16dp"
                android:layout_marginTop="25dp"
                android:layout marginBottom="25dp"
                android:fontFamily="@font/montserrat_regular"
                android:text="Enter the amount of water you drank today if you
want to evaluate your physical well-being!"
                android:textColor="#474646"
                android:textSize="14sp"
                android:textStyle="bold"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toStartOf="@+id/iconDrop"
                app:layout_constraintHorizontal_bias="0.0"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <ImageView
                android:id="@+id/iconDrop"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="32dp"
                android:layout_marginEnd="16dp"
                android:layout_marginBottom="32dp"
                android:src="@drawable/ic_thinking_droplet"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintVertical_bias="1.0" />
        </androidx.constraintlayout.widget.ConstraintLayout>
    </com.google.android.material.card.MaterialCardView>

    <EditText
        android:id="@+id/et"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:hint="Enter Input Value (x)"
        android:inputType="textPersonName"
        android:minHeight="48dp"
        android:textColor="@color/black"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/materialCardView" />

    <Button
        android:id="@+id/btnPred"
        style="@style/ButtonGradient"
        android:layout_width="158dp"
        android:layout_height="48dp"
```

```
        android:layout_marginTop="16dp"
        android:text="Predict"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et" />

    <TextView
        android:id="@+id/textView18"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:fontFamily="@font/montserrat_regular"
        android:text="Your well-being assesment:"
        android:textColor="@color/black"
        android:textSize="16sp"
        app:layout_constraintStart_toStartOf="@+id/materialCardView"
        app:layout_constraintTop_toBottomOf="@+id/btnPred" />

    <TextView
        android:id="@+id/hw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:fontFamily="@font/montserrat_regular"
        android:textColor="#6A91DE"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="@+id/textView18"
        app:layout_constraintStart_toEndOf="@+id/textView18"
        app:layout_constraintTop_toTopOf="@+id/textView18" />


    <com.github.mikephil.charting.charts.LineChart
        android:id="@+id/chart"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp"
        android:minHeight="200dp"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/hw"
        app:layout_constraintVertical_bias="0.0" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

SettingsFragment.kt

```
package com.droplet.ui.settings

import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.os.Build
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
```

```
import android.view.ViewGroup
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.findNavController
import com.droplet.R
import com.droplet.databinding.FragmentSettingsBinding
import com.droplet.ui.base.BaseFragment

class SettingsFragment: BaseFragment() {

    private val CHANNEL_ID = "channel_id_example_01"
    private val notificationId = 101

    private lateinit var binding: FragmentSettingsBinding

    private val viewModel by viewModels<SettingsViewModel>()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentSettingsBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        updateUi()
    }

    private fun updateUi() {
        binding.apply {
            toolbar.title.text = "Settings"
            switchEnableNotification.setOnClickListener {
                sendNotification()
            }
            btnLogOut.setOnClickListener {
                viewModel.logout()
                findNavController().navigate(R.id.nav_graph)
            }
        }
    }
    private fun createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val name = "Notification Title"
            val descriptionText = "Notification Description"
            val importance = NotificationManager.IMPORTANCE_HIGH
            val channel = NotificationChannel(CHANNEL_ID, name,
importance).apply {
                description = descriptionText
            }
            val notificationManager =
                requireActivity().getSystemService(Context.NOTIFICATION_SERVICE)
as NotificationManager
            notificationManager.createNotificationChannel(channel)
        }
    }
```

```
    private fun sendNotification() {
        val builder = NotificationCompat.Builder(requireContext(), CHANNEL_ID)
            .setSmallIcon(R.drawable.logo_drop)
            .setContentTitle("Droplet")
            .setContentText("It's time for drink water")
            .setPriority(NotificationCompat.PRIORITY_HIGH)

        with(NotificationManagerCompat.from(requireContext())) {
            notify(notificationId, builder.build())
        }
    }
}
```

SettingsViewModel.kt

```
package com.droplet.ui.settings

import com.droplet.data.SharedPrefHelper
import com.droplet.ui.base.BaseViewModel

class SettingsViewModel: BaseViewModel() {

    fun logout() {
        SharedPrefHelper.clear()
    }

}
```

fragment_settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    tools:context=".ui.settings.SettingsFragment">

    <include
        android:id="@+id/toolbar"
        layout="@layout/toolbar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="6dp"
        app:layout_constraintTop_toBottomOf="@+id/toolbar"
        android:orientation="vertical">

        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="6dp"
            android:layout_marginTop="6dp"
            android:layout_marginEnd="6dp"
            android:visibility="visible"
            app:cardCornerRadius="10dp"
```

```
tools:visibility="visible">

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/white">

    <androidx.appcompat.widget.SwitchCompat
        android:id="@+id/switchEnableNotification"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="34dp"
        android:minWidth="48dp"
        android:minHeight="48dp"
        android:thumb="@drawable/thumb"
        app:layout_constraintBottom_toBottomOf="@+id/textView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@+id/textView"
        app:track="@drawable/track" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="40dp"
        android:layout_marginTop="18dp"
        android:fontFamily="@font/montserrat_regular"
        android:text="Notification settings"
        android:textColor="#86BEFF"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView"
        style="@style/TextNotificationSetting"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:fontFamily="@font/montserrat_regular"
        android:text="Enable notifications"
        app:layout_constraintStart_toStartOf="@+id/textView2"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <View
        android:id="@+id/view2"
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:layout_marginTop="11dp"
        android:background="#E7E8ED"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <TextView
        android:id="@+id/textView3"
        style="@style/TextNotificationSetting"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="11dp"
```

```
            android:text="Wake up / Sleep time"
            app:layout_constraintStart_toStartOf="@+id/textView2"
            app:layout_constraintTop_toBottomOf="@+id/view2" />

        <TextView
            android:id="@+id/textView3_3"
            style="@style/TextNotificationSetting"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="3dp"
            android:fontFamily="@font/montserrat_regular"
            android:text="08:00 - 23:00"
            android:textColor="#7C828F"
            android:textSize="12sp"
            app:layout constraintStart toStartOf="@+id/textView2"
            app:layout_constraintTop_toBottomOf="@+id/textView3" />

        <View
            android:id="@+id/view3"
            android:layout_width="match_parent"
            android:layout_height="2dp"
            android:layout_marginTop="12dp"
            android:background="#E7E8ED"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3_3" />

        <TextView
            android:id="@+id/textView4"
            style="@style/TextNotificationSetting"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="12dp"
            android:text="Frequency"
            app:layout_constraintStart_toStartOf="@+id/textView2"
            app:layout_constraintTop_toBottomOf="@+id/view3" />

        <TextView
            android:id="@+id/textView15"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="3dp"
            android:layout_marginBottom="27dp"
            android:fontFamily="@font/montserrat_regular"
            android:text="Every hour"
            android:textColor="#7C828F"
            android:textSize="12sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="@+id/textView4"
            app:layout_constraintTop_toBottomOf="@+id/textView4" />

    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>

<Button
    android:id="@+id/btnLogOut"
    style="@style/ButtonGradient"
    android:layout_width="158dp"
    android:layout_height="48dp"
```

```
            android:layout_gravity="center"
            android:layout_marginTop="32dp"
            android:text="LOG OUT"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/toolbar" />

    </LinearLayout>


</androidx.constraintlayout.widget.ConstraintLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.droplet">

    <application
        android:allowBackup="true"
        android:name=".DropletApp"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Droplet">

        <activity android:name=".ui.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>

</manifest>
```

SharedPrefHelper.kt

```
package com.droplet.data

import PREF_KEY
import PREF_LOG_IN_KEY
import PREF_TOTAL_KEY
import PREF_TOTAL_INTAKE
import android.content.Context
import android.content.SharedPreferences

object SharedPrefHelper {
    private lateinit var preferences: SharedPreferences

    fun init(context: Context) {
        preferences = context.getSharedPreferences(PREF_KEY,
Context.MODE_PRIVATE)
    }
```

```
    fun saveTotalAmount(value: Int) {
        preferences.edit().putInt(PREF_TOTAL_KEY, value).apply()
    }

    fun getTotalAmount(): Int {
        return preferences.getInt(PREF_TOTAL_KEY, 0)
    }

    fun setIsLogIn(value: Boolean) {
        preferences.edit().putBoolean(PREF_LOG_IN_KEY, value).apply()
    }

    fun getIsLogIn(): Boolean {
        return preferences.getBoolean(PREF LOG IN KEY, false)
    }

    fun saveTotalIntake(value: Int) {
        preferences.edit().putInt(PREF_TOTAL_INTAKE, value).apply()
    }

    fun getTotalIntake(): Int {
        return preferences.getInt(PREF_TOTAL_INTAKE, 0)
    }

    fun clear() {
        preferences.edit().clear().apply()
    }
}
```

Constants.kt

```
const val TEST_CONSTANT = 0

const val PREF_KEY = "droplet_pref_key"
const val PREF_TOTAL_KEY = "total_key"
const val PREF_LOG_IN_KEY = "log_in_key"
const val PREF_TOTAL_INTAKE = "total_intake_key"
```

ViewExtensions.kt

```
package com.droplet.extensions

import android.content.Context
import android.view.View
import android.widget.Toast

fun View.visible() {
    visibility = View.VISIBLE
}

fun View.gone(shouldBeGone: Boolean = true) {
    if (shouldBeGone) visibility = View.GONE
    else visible()
}

fun Context.showToast(text: String) {
    Toast.makeText(this, text, Toast.LENGTH_SHORT).show()
}
```

MainActivity.kt

```
package com.droplet.ui

import android.os.Bundle
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.NavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.setupWithNavController
import com.droplet.R
import com.droplet.databinding.ActivityMainBinding
import com.droplet.extensions.gone
import com.droplet.extensions.visible

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    private val viewModel by viewModels<MainViewModel>()

    private lateinit var navController: NavController

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        initNavigation()
    }

    private fun initNavigation() {
        binding.apply {

            val navHostFragment = supportFragmentManager
                .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
            navController = navHostFragment.navController
            bottomNavView.setupWithNavController(navController)

            navController.addOnDestinationChangedListener { _, destination, _ ->
                when (destination.id) {
                    R.id.waterControlFragment -> {
                        bottomNavView.visible()
                    }
                    R.id.statisticFragment -> {
                        bottomNavView.visible()
                    }
                    R.id.settingsFragment -> {
                        bottomNavView.visible()
                    }
                    R.id.notificationFragment -> {
                        bottomNavView.visible()
                    }
                    else ->  bottomNavView.gone()
                }
            }
        }
    }
}
```

MainViewModel.kt

```
package com.droplet.ui

import com.droplet.ui.base.BaseViewModel


class MainViewModel: BaseViewModel() {

}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/coordinatorLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:defaultNavHost="true"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/bottomNavView"
        app:navGraph="@navigation/nav_graph" />

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottomNavView"
        android:layout_width="0dp"
        android:layout_height="56dp"
        app:backgroundTint="@null"
       android:backgroundTint="@null"
        android:backgroundTintMode="multiply"
        android:background="@drawable/background_navigation_bar"
        app:itemIconSize="30dp"
        app:labelVisibilityMode="unlabeled"
        app:itemIconTint="@color/bottom_bar_item_color"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/nav_host_fragment"
        app:menu="@menu/menu_items" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Utils.kt

```
package com.droplet.utils

import java.text.SimpleDateFormat
import java.util.*
```

```
object Utils {

    fun formatTime(hourOfDay: Int, minute: Int): String {

        return when {
            hourOfDay == 0 -> {
                if (minute < 10) {
                    "${hourOfDay + 12}:0${minute} am"
                } else {
                    "${hourOfDay + 12}:${minute} am"
                }
            }
            hourOfDay > 12 -> {
                if (minute < 10) {
                    "${hourOfDay - 12}:0${minute} pm"
                } else {
                    "${hourOfDay - 12}:${minute} pm"
                }
            }
            hourOfDay == 12 -> {
                if (minute < 10) {
                    "${hourOfDay}:0${minute} pm"
                } else {
                    "${hourOfDay}:${minute} pm"
                }
            }
            else -> {
                if (minute < 10) {
                    "${hourOfDay}:${minute} am"
                } else {
                    "${hourOfDay}:${minute} am"
                }
            }
        }
    }
}
```

toolbar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/background_splash">

    <TextView
        android:id="@+id/title"
        style="@style/TextStyleNormal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="41dp"
        android:layout_marginTop="27dp"
        android:layout_marginBottom="27dp"
        android:textColor="@color/white"
        android:textSize="24sp"
        tools:text="Notifications" />

</FrameLayout>
```

DropletApp.kt

```
package com.droplet

import android.app.Application
import com.droplet.data.SharedPrefHelper

class DropletApp: Application() {

    override fun onCreate() {
        super.onCreate()
        SharedPrefHelper.init(this)
    }
}
```

nav_graph.xml

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/splashFragment">
    <fragment
        android:id="@+id/paramsFragment"
        android:name="com.droplet.ui.registration.params.ParamsFragment"
        android:label="fragment_params"
        tools:layout="@layout/fragment_params" >
        <action
            android:id="@+id/action_paramsFragment_to_waterControlFragment"
            app:enterAnim="@anim/fade_in"
            app:exitAnim="@anim/fade_out"
            app:popEnterAnim="@anim/fade_in"
            app:popExitAnim="@anim/fade_out"
            app:destination="@id/waterControlFragment" />
    </fragment>
    <fragment
        android:id="@+id/introFragment"
        android:name="com.droplet.ui.registration.intro.IntroFragment"
        android:label="fragment_intro"
        tools:layout="@layout/fragment_intro">
        <action
            android:id="@+id/action_introFragment_to_paramsFragment"
            app:enterAnim="@anim/fade_in"
            app:exitAnim="@anim/fade_out"
            app:popEnterAnim="@anim/fade_in"
            app:popExitAnim="@anim/fade_out"
            app:destination="@id/paramsFragment" />
    </fragment>
    <fragment
        android:id="@+id/splashFragment"
        android:name="com.droplet.ui.splash.SplashFragment"
        android:label="fragment_splash"
        tools:layout="@layout/fragment_splash">
        <action
            android:id="@+id/action_splashFragment_to_introFragment"
            app:destination="@id/introFragment"
            app:enterAnim="@anim/fade_in"
            app:exitAnim="@anim/fade_out"
            app:popEnterAnim="@anim/fade_in"
            app:popExitAnim="@anim/fade_out"
```

```
            app:popUpTo="@id/nav_graph" />
        <action
            android:id="@+id/action_splashFragment_to_mainFragment"
            app:destination="@id/waterControlFragment"
            app:enterAnim="@anim/fade_in"
            app:exitAnim="@anim/fade_out"
            app:popEnterAnim="@anim/fade_in"
            app:popExitAnim="@anim/fade_out"
            app:popUpTo="@id/nav_graph" />
    </fragment>
    <fragment
        android:id="@+id/waterControlFragment"
        android:name="com.droplet.ui.watercontrol.WaterControlFragment"
        android:label="fragment_main"
        tools:layout="@layout/fragment water control" />
    <fragment
        android:id="@+id/statisticFragment"
        android:name="com.droplet.ui.statistic.StatisticFragment"
        android:label="fragment_statistic"
        tools:layout="@layout/fragment_statistic" />

    <fragment
        android:id="@+id/settingsFragment"
        android:name="com.droplet.ui.settings.SettingsFragment"
        android:label="fragment_settings"
        tools:layout="@layout/fragment_settings" />
    <fragment
        android:id="@+id/notificationFragment"
        android:name="com.droplet.ui.notifications.NotificationFragment"
        android:label="fragment_notification"
        tools:layout="@layout/fragment_notification" />
</navigation>
```

model_creation.py

```
#Import the tensorflow libraries which are used to create a sequential model and
add a dense layer
import tensorflow
import tensorflow as tf
import numpy as np
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
#Create random arrays x1,y1 as training data
x1=np.array([250,500,750,1000,1250,1500,1750,2000,2250,2500,2750,3000])
y1=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
#Create a model with activation function as linear for last layer
model=Sequential()
model.add(Dense(10,input_shape=[1],activation='relu'))
model.add(Dense(1,activation='linear'))
#Compile the model with loss function as Mean Squared Error
model.compile(loss='mean_squared_error',optimizer='Adam',metrics=['mse'])
#Fitting the model to 1000 epochs
```

```
model.fit(x1,y1,epochs=1000)
#Predicting the model
model.predict([2400])
#Save the model to saved_model.pbtxt
tensorflow.keras.models.save_model(model, 'saved_model.pbtxt')
#We need to create a TFLite Converter Object from model we created
converter = tensorflow.lite.TFLiteConverter.from_keras_model(model=model)
#Create a tflite model object from TFLite Converter
tfmodel = converter.convert()
#Save TFLite model into a .tflite file
open("degree.tflite","wb").write(tfmodel)
```