Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка мобільної системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу»

Виконав: студент IV курсу
групи 2ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Саєцький Д.Р.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Черноволик Г.О.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Колесник І. С.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ_____
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
О. Н. Романюк _____
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Саєцькому Денис Руслановичу

1. Тема роботи – Розробка мобільної системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу.

Керівник роботи: Черноволик Галина Олександрівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: методи відстеження фізичної активності – алгоритми обчислення калорій та органічних речовин; моделі – моделі трекінгу активності користувачів; графічний режим – сучасні інтерфейсні компоненти з використанням XML для побудови UI; вихідні дані для вимірювання – дані про фізичну активність, споживання калорій, заплановані та виконані завдання; дані про особисті параметри користувачів, такі як вік, стать, вага, рівень активності, цілі щодо здорового способу життя та ефективності використання робочого часу.

4. Зміст текстової частини: вступ; аналіз стану мобільних систем для підтримки здорового способу життя й підвищення ефективності використання робочого часу; розробка методу, моделі та алгоритму мобільної системи; розробка програмних модулів мобільної системи; тестування програми; висновки; список використаних джерел; додатки; ілюстративна частина

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність розробки; мета, предмет, об'єкт дослідження; постановка задачі: новизна; практична цінність отриманих результатів; існуючі аналоги; метод адаптивного планувальника завдань (ч.1); метод адаптивного планувальника завдань (ч.2); блок-схема методу планувальника розкладу завдань; модель персоналізованої

системи рекомендації у вигляді діаграми діяльності; алгоритм роботи модуля трекера калорій; діаграма діяльності мобільної системи обрахунків кількість калорій для споживання; система діаграми класів; загальний алгоритм роботи програмного застосунку; показ функціоналу; використані технології; апробація та публікація результатів роботи; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата			
		Завдання видав		Завдання прийняв	
1-4	Черноволик Г.О., к.т.н., доцент каф ПЗ	13.03.24		03.06.24	

7. Дата видачі завдання 13 березня 2024 р

КАЛЕНДАРНИЙ ПЛАН

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз підходів до підвищення ефективності використання робочого часу та підтримки здорового способу життя за допомогою мобільних додатків.	14.03.24 – 22.03.24	вик.
2	Розробка методів моніторингу фізичної активності та управління робочими завданнями.	23.03.24 – 24.03.24	вик.
3	Розробка алгоритмів адаптивного планування завдань та надання рекомендацій.	25.03.24 – 22.04.24	вик.
4	Розробка мобільного програмного забезпечення	23.04.24 – 10.05.24	вик.
5	Тестування та валідація мобільного додатку.	11.05.24 – 24.05.24	вик.
6	Оформлення матеріалів до захисту БДР	25.05.24 – 30.05.24	вик.

Студент _____ **Д.Р. Сасцький**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **Г.О. Черноволик**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 65 сторінок формату А4, на яких є 25 рисунка, 2 таблиці, список використаних джерел містить 24 найменування.

У бакалаврській дипломній роботі проведено детальний аналіз методів і засобів підтримки здорового способу життя та підвищення продуктивності робочого часу за допомогою мобільних систем. Сформульовано мету досліджень – підвищення продуктивності використання робочого часу та підтримка здорового способу життя за рахунок впровадження мобільної системи, що інтегрує трекінг фізичної активності, контроль харчування та оптимізацію робочого графіку.

Запропоновано методи трекінгу фізичної активності з використанням мобільних пристроїв та зовнішніх сенсорів. Розроблено систему контролю харчування з можливістю індивідуального планування дієти та інтеграції технік тайм-менеджменту для оптимізації робочого часу.

Розроблено прототип мобільної системи, проведено тестування та оцінку ефективності системи в реальних умовах для аналізу впливу на здоров'я та продуктивність користувачів. Отримані в бакалаврській дипломній роботі результати можна використати для створення інноваційних мобільних додатків, що сприяють збереженню здоров'я та підвищенню продуктивності.

Ключові слова: здоровий спосіб життя, мобільні системи, трекінг фізичної активності, тайм-менеджмент.

ABSTRACT

The bachelor's thesis consists of 65 A4 pages, containing 25 figures, 2 tables, and a reference list with 24 sources.

The bachelor's thesis provides a detailed analysis of methods and tools for supporting a healthy lifestyle and enhancing work productivity through mobile systems. The research goal is formulated as improving work time productivity and supporting a healthy lifestyle by implementing a mobile system that integrates physical activity tracking, sleep monitoring, nutrition control, and work schedule optimization.

Proposed methods include tracking physical activity using mobile devices and external sensors, as well as implementing sleep monitoring and quality analysis functions. A nutrition control system has been developed, enabling individualized diet planning, along with the integration of time management techniques for optimizing work time.

A prototype of the mobile system was developed, and its effectiveness was tested and evaluated in real conditions to analyze its impact on users' health and productivity. The results obtained in the bachelor's thesis can be used to create innovative mobile applications that promote health preservation and productivity enhancement.

Keywords: healthy lifestyle, mobile systems, physical activity tracking, time management.

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ РОЗВИТКУ МОБІЛЬНИХ СИСТЕМ ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ Й ПІДВИШЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ РОБОЧОГО ЧАСУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ 9
1.1 Аналіз стану мобільних систем для підтримки здорового способу життя й підвищення ефективності використання робочого часу 9
1.2 Порівняльний аналіз аналогів 10
1.3 Аналіз методів розв'язання задачі 17
1.4 Постановка задач для розробки мобільної системи 18
1.5 Висновки 19
2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМУ МОБІЛЬНОЇ СИСТЕМИ 20
2.1 Аналіз даних 20
2.2 Розробка моделі системи 21
2.3 Розробка методу адаптивного планувальника завдань 25
2.4 Розробка загального алгоритму роботи системи 30
2.5 Висновки 32
3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ МОБІЛЬНОЇ СИСТЕМИ 33
3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації мобільної системи 33
3.2 Розробка модуля трекера споживання калорій 37
3.3 Розробка модуля обрахування кількості калорій та органічних речовин для споживання 40
3.4 Розробка модуля підвищення ефективності використання робочого часу. 41
3.5 Розробка дизайну інтерфейсу мобільної системи 43
3.6 Висновки 51
4 ТЕСТУВАННЯ ПРОГРАМИ 52
4.1 Тестування системи 52
4.2 Розробка інструкції користувача 58
4.3 Висновки 60
ВИСНОВКИ 61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63
ДОДАТКИ 66
Додаток А – Технічне завдання ..... **Error! Bookmark not defined.**
Додаток Б – Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень ..... **Error! Bookmark not defined.**

Додаток В – Лістинг програми 72
Додаток Г – Ілюстративний матеріал..... 99

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі швидкого темпу життя і високих вимог до продуктивності праці, збереження здоров'я та ефективне використання робочого часу набувають особливої ваги. Вплив стресу, сидячого способу життя, недостатньої фізичної активності та неправильного харчування негативно позначаються як на фізичному, так і на психічному здоров'ї людей. Це, у свою чергу, призводить до зниження продуктивності та якості життя загалом.

Останні дослідження свідчать про те, що підтримка здорового способу життя безпосередньо впливає на працездатність та ефективність виконання професійних обов'язків [1]. Наприклад, регулярні фізичні вправи, збалансоване харчування та належний відпочинок не лише покращують фізичний стан, але й підвищують концентрацію, знижують рівень стресу та підвищують загальний тонус організму.

З огляду на ці фактори, тема розробки мобільної системи для підтримки здорового способу життя й підвищення продуктивності використання робочого часу є надзвичайно актуальною. Мобільні додатки стали невід'ємною частиною повсякденного життя багатьох людей. Вони пропонують зручний і ефективний спосіб моніторингу здоров'я, планування фізичних активностей, контролю за харчуванням та управління робочим часом.

Розробка такої системи спрямована на вирішення низки важливих завдань. По-перше, це сприяння здоровому способу життя через інтеграцію різноманітних функцій для відстеження та аналізу фізичної активності та харчування. По-друге, це підвищення продуктивності через оптимізацію робочого часу та планування завдань, враховуючи індивідуальні особливості користувача.

Для реалізації мобільної системи, яка підтримує здоровий спосіб життя та підвищує продуктивність використання робочого часу, необхідно застосувати

комплексний підхід, що включає різні методи та технології. Одним із ключових методів є інтеграція трекінгу фізичної активності. Використовуючи датчики, такі як акселерометри та гіроскопи, можна точно вимірювати кількість кроків, пройдену відстань, спалені калорії та інші показники активності. Крім того, система може інтегруватися з популярними фітнес-трекерами та смарт-годинниками, що значно розширить її функціональні можливості та зробить використання більш зручним для користувачів.

Контроль за харчуванням – ще одна важлива функція системи. Реалізація цієї функції можлива через використання баз даних харчових продуктів та їхніх харчових цінностей, а також через вбудовані сканери штрих-кодів та QR-кодів, які дозволяють швидко вводити дані про спожиті продукти. Додаток може запропонувати користувачеві різні дієтичні плани, що враховують його індивідуальні потреби, мету (наприклад, схуднення чи набір ваги), а також алергени та інші особисті обмеження.

Оптимізація робочого часу та підвищення продуктивності досягаються за рахунок інтеграції з календарями, нагадуваннями та системами управління завданнями. Система може аналізувати розклад користувача, рекомендувати оптимальні періоди для виконання завдань з урахуванням рівня енергії та продуктивності впродовж дня. Важливим є використання технік тайм-менеджменту, таких як метод Pomodoro або GTD (Getting Things Done), що дозволяють ефективно організувати робочий процес та уникнути вигорання.

Таким чином, мобільна система для підтримки здорового способу життя та підвищення продуктивності використання робочого часу має бути комплексною та багатофункціональною. Вона повинна інтегрувати сучасні технології та методи, щоб надавати користувачам максимально корисні та персоналізовані рекомендації. Це сприятиме не лише покращенню фізичного та психічного здоров'я, але й підвищенню загальної якості життя.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення продуктивності використання робочого часу та підтримка здорового способу життя за рахунок впровадження мобільної системи, що інтегрує трекінг фізичної активності, контроль харчування та оптимізацію робочого графіку, що дозволяє покращити фізичний стан та активність користувача.

Задачі дослідження:

- аналіз існуючих мобільних додатків і систем для підтримки здорового способу життя та підвищення продуктивності;
- розробка методу, моделі та алгоритмів роботи системи;
- розробка модуля трекінгу фізичної активності;
- створення системи контролю харчування з можливістю індивідуального планування дієти;
- інтеграція технік тайм-менеджменту для оптимізації робочого часу та підвищення продуктивності;
- розробка алгоритмів персоналізації рекомендацій на основі аналізу зібраних даних;
- проведення тестування та оцінка ефективності розробленої мобільної системи в реальних умовах.

Об’єкт дослідження – процес розробки мобільної системи для підтримки здорового способу життя та підвищення продуктивності використання робочого часу.

Предмет дослідження – методи та засоби розробки мобільної системи для підтримки здорового способу життя та підвищення продуктивності використання робочого часу.

Методи дослідження. У процесі досліджень використовувались:

- методи теорії алгоритмів для розробки персоналізованих рекомендацій;
- методи обробки сигналів для трекінгу фізичної активності;
- методи програмування для створення прототипу мобільної системи;
- методи тестування та оцінки ефективності системи в реальних умовах для аналізу впливу на здоров'я та продуктивність користувачів.

Новизна отриманих результатів:

1. Подальшого розвитку набув метод адаптивного планування, який, на відміну від існуючих, полягає в автоматичному коригуванні розкладу завдань з урахуванням втомленості та продуктивності користувача, використовуючи локальну базу даних SQLite, яка забезпечує швидкий доступ до даних і високу продуктивність їх обробки, що значно прискорює процес адаптації розкладу.

2. Подальшого розвитку набула персоналізована модель формування рекомендацій для покращення здорового способу життя, яка, на відміну від існуючих, використовує бібліотеку Retrofit для інтеграції з REST API, щоб отримати додаткові дані, а також бібліотеку Room для ефективної роботи з базою даних, що значно підвищує ефективність і точність рекомендацій, роблячи процес підтримки здорового способу життя більш ефективним та персоналізованим.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає в можливості використання розробленої мобільної системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу користувача.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У друкованій роботі [2], опублікованій у співавторстві, автору належать такі результати: порівняльний аналіз аналогів й алгоритм роботи мобільної системи підтримки

здорового способу життя й підвищення продуктивності використання робочого часу.

Апробація матеріалів бакалаврської дипломної роботи. Матеріали бакалаврської дипломної роботи доповідалися на конференції: «LIII Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024.

Публікації. Результати дослідження були опубліковані в матеріалах Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024 [2].

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 24 найменування, 5 додатків. Робота містить 34 ілюстрації та 3 таблиці

1 АНАЛІЗ РОЗВИТКУ МОБІЛЬНИХ СИСТЕМ ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ Й ПІДВИШЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ РОБОЧОГО ЧАСУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану мобільних систем для підтримки здорового способу життя й підвищення ефективності використання робочого часу

На сьогоднішній день на ринку існує ряд мобільних додатків, спрямованих на підвищення ефективності використання робочого часу та підтримку здорового способу життя. Ці додатки можуть включати функціональність, що дозволяє відстежувати фізичну активність, планувати завдання, керувати часом, а також надавати рекомендації щодо покращення продуктивності та досягнення поставлених цілей. Проте, не дивлячись на наявність різноманітних додатків [3], багато з них мають обмежену функціональність або недостатньо персоналізовані для конкретних потреб користувача. Деякі можуть бути складні у використанні або не надавати достатньої зворотного зв'язку з користувачем. Також важливо враховувати різноманітність мобільних платформ, які вимагають адаптації додатків під різні операційні системи.

Переваги використання мобільних систем для підвищення ефективності використання робочого часу виявляються на кількох рівнях. По-перше, такі додатки надають користувачам зручний і доступний інструмент для ведення контролю за їхнім робочим часом та продуктивністю в будь-який час і в будь-якому місці. Вони дозволяють відстежувати різноманітні показники, такі як виконані завдання, заплановані події, кількість витраченого часу на різні види діяльності, а також аналізувати їх у зручному для користувача форматі.

По-друге, такі системи можуть надавати персоналізовані рекомендації та поради щодо поліпшення ефективності роботи на підставі зібраних даних. Це може включати в себе індивідуалізовані плани роботи, рекомендації щодо

розподілу часу та пріоритетів, спеціально розроблені для конкретного користувача.

Крім того, мобільні системи стимулюють мотивацію [4] та відповідальність користувачів за їхню продуктивність. Подавання нагадувань, встановлення цілей та моніторинг прогресу допомагають створити систему позитивних стимулів для досягнення робочих цілей. Загалом, використання мобільних систем для підвищення ефективності використання робочого часу дозволяє зробити процес управління часом більш доступним, ефективним і мотивуючим для користувачів, що веде до покращення їхньої загальної продуктивності та якості життя.

Отже, аналіз стану мобільних систем для підвищення ефективності використання робочого часу підкреслює значущість і необхідність подальшої розробки та вдосконалення таких додатків. Вони є не лише зручним інструментом для користувачів, але й потужним інструментом для популяризації ефективного використання робочого часу та підтримки продуктивності. Розробка ефективної та інноваційної мобільної системи для підвищення ефективності використання робочого часу стає важливим кроком у напрямку покращення продуктивності та здорового способу життя суспільства.

1.2 Порівняльний аналіз аналогів

Використання мобільних систем для підвищення ефективності використання робочого часу та підтримки здорового способу життя набуває все більшої популярності в сучасному світі. З розвитком технологій та поширенням смартфонів, люди все частіше використовують мобільні додатки для планування свого робочого часу, підвищення продуктивності та моніторингу свого здоров'я. Розвиток цих систем є відповіддю на зростаючий інтерес суспільства до ефективного використання часу та підтримки здорового способу життя. В умовах сучасного життя, коли багато людей стикаються з проблемами тайм-менеджменту та перевантаженості роботою, мобільні додатки можуть стати

невід'ємною частиною стратегії підвищення продуктивності та збереження здоров'я..До найбільш відомих рішень відносять:

- MyFitnessPal;
- Strava;
- Toggl;
- Headspace;
- Lifesum.

Прикладом зручного та популярного додатку для ведення контролю над здоров'ям і фізичною активністю є MyFitnessPal (рис. 1.1). Цей додаток став дуже популярним серед користувачів завдяки своїй простоті використання та багатому функціоналу. Він дозволяє користувачам вести щоденник здорового харчування, відстежувати спожиті калорії, білки, жири та вуглеводи, а також встановлювати цілі для досягнення [5]. Крім того, MyFitnessPal має велику базу даних щодо продуктів і страв, що дозволяє легко вносити інформацію про спожиту їжу.

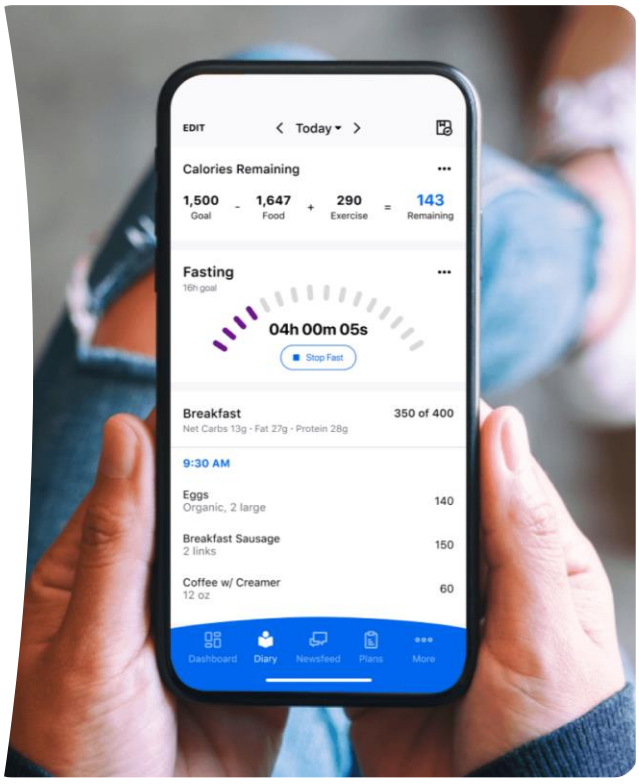

Рисунок 1.1 – Приклад роботи застосунку MyFitnessPal

Крім того, серед популярних додатків для здорового способу життя варто відзначити Strava (рис. 1.2). Цей додаток спеціалізується на відстеженні фізичної активності, особливо серед велосипедистів, бігунів та інших спортсменів. Strava дозволяє відстежувати дистанції, тривалість тренувань, швидкість та інші параметри, а також ділитися результатами з іншими користувачами та долучатися до спільноти спортсменів [6]. Цей додаток допомагає мотивувати користувачів до досягнення спортивних цілей і покращення фізичної форми. Завдяки своїй соціальній складовій, Strava створює спільноту спортсменів, яка надихає до досягнень і підтримки один одного. Користувачі можуть обмінюватися своїми тренуваннями, планами та досягненнями, а також створювати та долучатися до викликів і групових тренувань. Це стимулює спортивну активність та забезпечує підтримку та мотивацію у будь-якій фізичній діяльності.

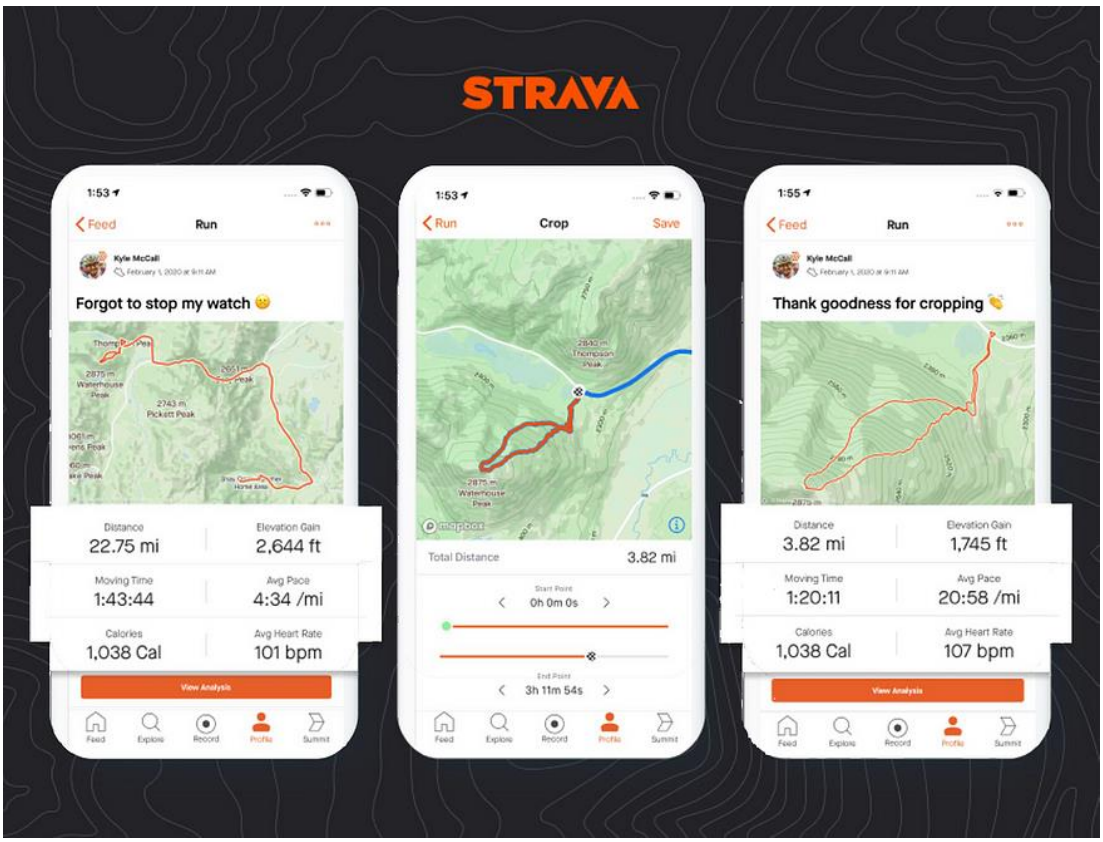

Рисунок 1.2 – Приклад роботи застосунку Strava

Серед аналогів мобільних систем для підвищення продуктивності робочого часу можна виділити Toggl (рис. 1.3). Toggl [7] – це популярний інструмент для відстеження робочого часу, який допомагає користувачам ефективно керувати своїм часом і завданнями.

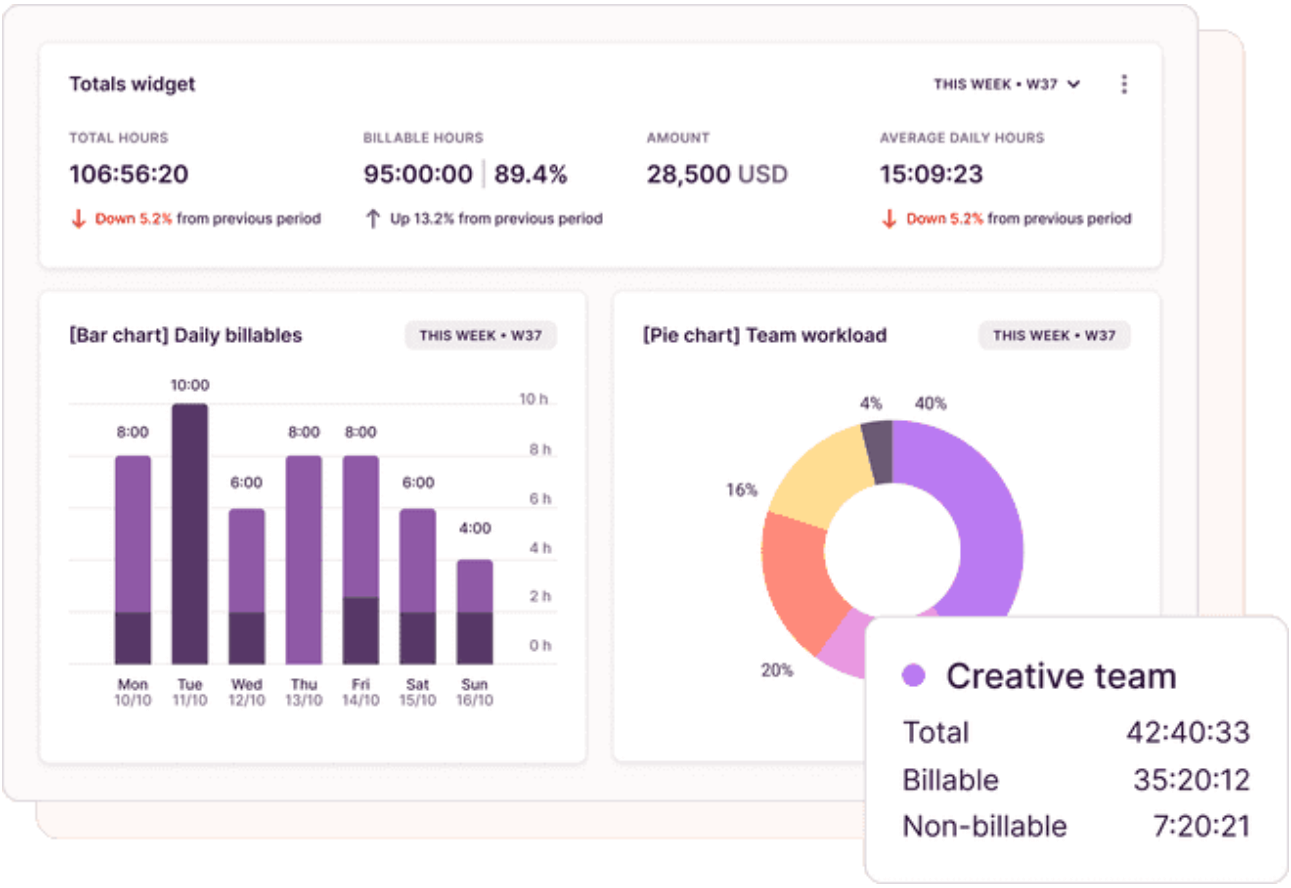

Рисунок 1.3 – Приклад роботи застосунку Toggl

Headspace (рис. 1.4) - ще один популярний додаток, який, хоч і спеціалізується на медитації, але також відіграє важливу роль у підтримці здорового способу життя [8]. Цей додаток пропонує користувачам різні медитаційні програми та уроки для зниження стресу, покращення сну та загального психологічного благополуччя. Headspace допомагає користувачам зосередитися на їхніх почуттях та емоціях, що сприяє підвищенню рівня їхнього здоров'я та щастя. Headspace не лише надає користувачам можливість практикувати медитацію, але й використовує науково обґрунтовані методи для

підвищення психічного здоров'я. Через регулярні медитаційні сесії та навчальні курси, Headspace допомагає користувачам знижувати рівень стресу, поліпшувати концентрацію, заспокоювати розум та зберігати емоційний баланс.

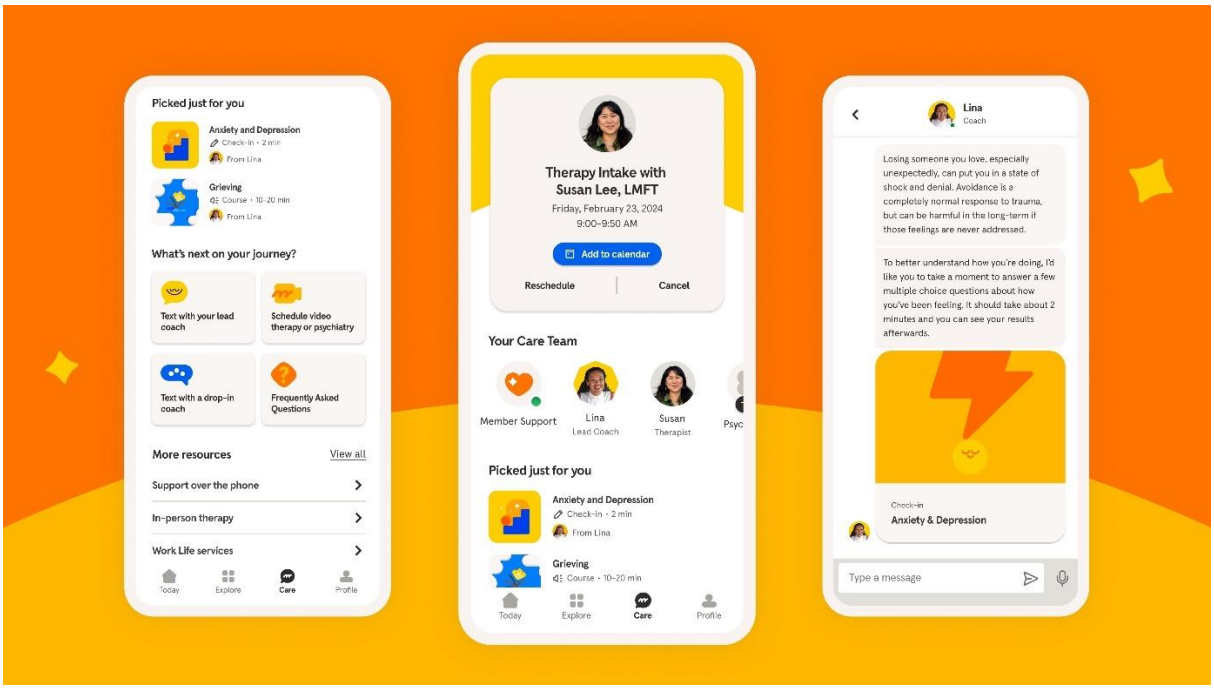

Рисунок 1.4 – Приклад роботи застосунку Headspace

Lifesum (рис. 1.5) - це ще один додаток, який зосереджується на підтримці здорового способу життя, проте він акцентується на аспектах харчування. За допомогою Lifesum користувачі можуть вести щоденник харчування, відстежувати спожиті калорії та поживні речовини, а також отримувати персоналізовані рекомендації щодо здорового харчування та досягнення цілей фітнесу [9]. Додаток також надає користувачам інформацію про продукти та страви, що допомагає їм зробити більш обізнаний вибір щодо свого харчування. Lifesum не лише веде облік харчування, але й надає користувачам індивідуалізовані поради та рекомендації щодо поліпшення харчових звичок. Завдяки аналізу споживаних продуктів і калорій, Lifesum може пропонувати персоналізовані рекомендації з питань харчування, що допомагає користувачам збалансувати своє харчування та досягати своїх цілей з фітнесу та здоров'я.

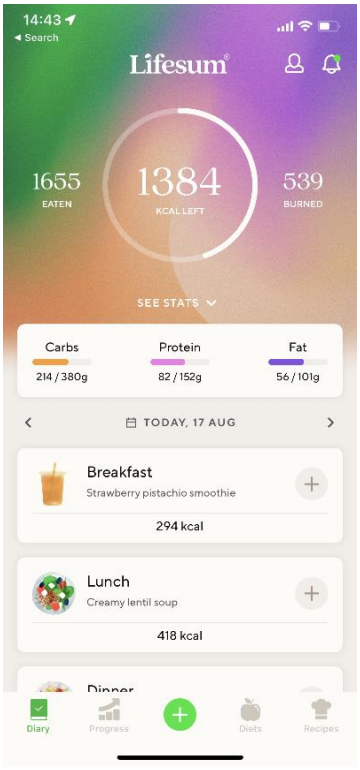

Рисунок 1.5 – Приклад роботи застосунку Lifesum

Розробка програмного забезпечення для підвищення ефективності використання робочого часу та підтримки здорового способу життя вимагає злагодженого поєднання досвіду в програмуванні, розуміння основ тайм-менеджменту та навичок у дизайні інтерфейсів користувача. Це передбачає створення програмних модулів, які забезпечать планування завдань, відстеження прогресу, управління робочим часом та аналіз зібраних даних. Ключовим етапом є розробка зручного та інтуїтивно зрозумілого інтерфейсу, який сприятиме мотивації користувачів до досягнення своїх робочих цілей, підвищення продуктивності та підтримки здорового способу життя.

Результати порівняння аналогів зведено в табл. 1.1.

Критерії	MyFitnessPal	Strava	Toggl	Headspace	Lifesum	Власна розробка
Ведення щоденника харчування	+	-	-	-	+	+

Продовження табл. 1.1.

Критерії	MyFitnessPal	Strava	Toggl	Headspace	Lifesum	Власна розробка
Відстеження прогресу	+	+	+	-	+	+
Управління робочим часом	-	-	+	+	-	+
Соціальна взаємодія	-	+	+	-	-	+
Медитаційні та психологічні вправи	-	-	-	+	-	+
Загальна оцінка	40%	40%	60%	40%	60%	100%

Відповідно до таблиці порівняльних характеристик (табл.1.1), розробка власної мобільної системи для підвищення ефективності використання робочого часу та підтримки здорового способу життя є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити користувачам більш повний та зручний інструмент для управління робочим часом і підтримки здорового способу життя.

Отже, розробка мобільної системи для підвищення ефективності використання робочого часу та підтримки здорового способу життя є актуальною та перспективною задачею в сучасному світі. На основі проведеного аналізу існуючих додатків було виявлено ряд недоліків у їхньому функціоналі та можливостях. Розроблений продукт має потенціал усунути ці недоліки та надати користувачам інноваційний та зручний інструмент для досягнення своїх цілей у плані продуктивності та здоров'я. Цей проєкт відкриває нові перспективи у сфері ефективного використання робочого часу та може стати важливим кроком у напрямку покращення якості життя користувачів.

1.3 Аналіз методів розв’язання задачі

Створення програмних засобів для підтримки здорового способу життя з використанням мобільних систем потребує уважного аналізу доступних методів. Це включає в себе визначення найбільш ефективних та оптимальних стратегій для реалізації функцій, пов'язаних із відстеженням фізичної активності, контролем харчування та психологічною підтримкою. У даному розділі проводиться аналіз різних методів розробки програмних модулів для мобільних систем, зокрема їхніх переваг та недоліків з урахуванням вимог до забезпечення зручного та ефективного користувацького інтерфейсу.

Один із варіантів вирішення проблеми створення програмного засобу для підтримки здорового способу життя може включати в себе використання гібридного підходу. Цей підхід передбачає поєднання функціональності з різних джерел інформації та ресурсів для забезпечення максимальної користі для користувачів. На приклад, програмний засіб може використовувати дані зі сенсорів мобільних пристроїв для відстеження фізичної активності користувача, таких як кроки, відстань, серцевий ритм тощо. Також можна інтегрувати базу даних щодо харчування, щоб дозволити користувачам вести щоденник харчування та отримувати рекомендації з покращення дієти. Психологічну підтримку можна забезпечити за допомогою медитаційних вправ та порад щодо зняття стресу, які можуть бути вбудовані безпосередньо в додаток. Гібридний підхід дозволяє використовувати найкращі аспекти різних рішень, що вже існують, та поєднувати їх для створення більш повного та ефективного програмного засобу для користувачів.

Ще одним варіантом вирішення проблеми може бути розробка інтегрованої мобільної платформи з використанням штучного інтелекту (ШІ). Цей підхід передбачає створення програмного забезпечення, яке здатне аналізувати поведінку користувача, надавати персоналізовані рекомендації та автоматично адаптуватися до індивідуальних потреб. Розробка інтегрованої мобільної платформи з використанням ШІ може забезпечити більш ефективну та

персоналізовану підтримку здорового способу життя для користувачів, що відповідає сучасним вимогам у сфері здоров'я та фітнесу.

Інший варіант вирішення проблеми може полягати в створенні інтерактивної платформи з використанням блокчейн технології [10]. Цей підхід може забезпечити високий рівень безпеки та приватності даних користувачів, а також створити механізми стимулювання здорового способу життя через використання криптовалют. Платформа може дозволити користувачам збирати та зберігати свої дані про фізичну активність, харчування та психологічне благополуччя у безпечному та недоступному для зміни форматі блокчейн. Крім того, використання криптовалют може стимулювати користувачів до досягнення своїх здорових цілей через нагороди за досягнуті результати. Створення інтерактивної платформи на базі блокчейн технології може забезпечити не лише безпеку та приватність даних користувачів, але і стимулювати їх до активності та досягнення своїх здорових цілей через використання інноваційних механізмів мотивації.

Розглянувши переваги та недоліки варіантів для розробки програмного забезпечення для підтримки здорового способу життя, було прийнято рішення обрати гібридний підхід. Переваги полягають у можливості комбінувати найкращі аспекти різних рішень, що вже існують, та поєднувати їх у єдиний комплексний програмний засіб. Цей підхід дозволяє використовувати функціонал з різних джерел інформації та ресурсів, що забезпечує більш широкий спектр можливостей для користувачів.

1.4 Постановка задач для розробки мобільної системи

Проаналізувавши переваги та недоліки існуючих мобільних систем для підтримки здорового способу життя, було визначено наступні завдання, які необхідно виконати для успішної розробки програмного забезпечення:

- розробити метод, модель та алгоритми роботи системи;

- розробити модуль відстеження споживання калорій;
- створити модуль для обрахування кількості калорій та органічних речовин для споживання;
- розробити модуль планування завдань, що дозволить користувачам ефективно розподіляти час на виконання робочих задач;
- розробити інтерфейс для взаємодії користувачів з програмним забезпеченням, забезпечивши його зручність та ефективність;
- провести тестування програмного забезпечення згідно з поставленими задачами.

1.5 Висновки

У першому розділі було розглянуто стан мобільних систем для підвищення ефективності використання робочого часу та підтримки здорового способу життя. Був проведений аналіз відомих аналогів, таких як: MyFitnessPal, Strava, Toggl, Headspace та Lifesum.

Порівнявши переваги та недоліки мобільних систем, були створені завдання для розробки власної мобільної системи для підвищення ефективності використання робочого часу та підтримки здорового способу життя. Ці завдання включають розробку модулів для ведення щоденника харчування, відстеження фізичних активностей, планування завдань, надання персоналізованих рекомендацій для користувачів, створення зручного інтерфейсу користувача та забезпечення стабільної роботи програмного забезпечення. Кожне завдання є важливим для забезпечення ефективної та корисної мобільної системи для користувачів. Таким чином, було доведено доцільність розробки власного продукту. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки власних програмних засобів.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМУ МОБІЛЬНОЇ СИСТЕМИ

2.1 Аналіз даних

Для успішної розробки мобільної системи для підвищення ефективності використання робочого часу та підтримки здорового способу життя необхідно ретельно вивчити та обробити вхідні дані.

Вхідні дані:

– дані про фізичну активність користувачів, що включають кількість кроків, витрачений час на тренування, пройдену відстань;

– інформація про харчування, що містить записи про спожиті продукти, калорійність та харчову цінність;

– дані про розклад та виконання робочих завдань, що включають заплановані події, дедлайни та статус виконання завдань;

– опитування та фідбек користувачів для визначення їхніх потреб та очікувань від системи.

Спочатку необхідно провести детальне ознайомлення з цими даними. Це включає аналіз їхніх форматів, структур та джерел. Наприклад, дані про фізичну активність можуть надходити з трекерів активності або бути введеними вручну користувачами. Інформація про харчування може бути отримана з баз даних продуктів або введена користувачами. Дані про розклад завдань можуть надходити з календарних додатків або бути введеними вручну.

Після ознайомлення з даними проводиться їх попередня обробка. Це включає перевірку на наявність помилок, відсутніх або пошкоджених записів, а також валідацію форматів. Наприклад, дані про фізичну активність та харчування мають бути перевірені на правильність введення. Необхідно впевнитися, що всі записи містять необхідну інформацію і відповідають встановленим стандартам.

Одним із важливих етапів аналізу даних є визначення потреб користувачів та встановлення вимог до функціональності системи. Це включає проведення опитувань та дослідження цільової аудиторії для визначення їхніх очікувань та вимог до системи. На основі зібраної інформації можна встановити пріоритети функцій, які мають бути реалізовані у системі.

Крім того, важливо провести аналіз конкурентних рішень на ринку, таких як MyFitnessPal, Strava, Toggl, Headspace та Lifesum. Це дозволяє визначити сильні та слабкі сторони існуючих додатків і врахувати їх при розробці власного продукту.

Після завершення аналізу даних, наступним кроком є створення концептуальної моделі мобільної системи. Модель повинна відображати компоненти системи та показувати, як вони взаємодіють між собою. Для наочності структуру та організацію системи можна представити у вигляді діаграм, схем або інших візуальних засобів. Це дозволить чітко зрозуміти, як функціонують окремі частини системи та їх взаємозв'язки.

2.2 Розробка моделі системи

Для кращого розуміння роботи модулів мобільної системи для підтримки здорового способу життя було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 2.1). Ця діаграма ілюструє послідовність дій користувача під час взаємодії з додатком [11].

Персоналізована модель рекомендацій для покращення здорового способу життя представляє собою інноваційний підхід, який використовує дані про активність користувача, споживання калорій та інші параметри для створення індивідуальних рекомендацій. Це дозволяє підтримувати здоровий спосіб життя більш ефективно, ніж за допомогою загальних рекомендацій.

Персоналізована модель рекомендацій базується на збиранні детальних даних про користувача. Ці дані включають фізичну активність: кількість кроків,

витрачений час на тренування, пройдена відстань, типи фізичної активності біг, ходьба, плавання, частота тренувань. Споживання калорій: записи про спожиті продукти, їх калорійність, склад – білки, жири, вуглеводи, час прийому їжі. Інші параметри: вік, стать, вага, зріст, індивідуальні цілі – схуднення, набір ваги, підтримка ваги, рівень стресу.

Після збору даних модель обробляє їх за допомогою певних правил та алгоритмів, які базуються на науково обґрунтованих методах та рекомендаціях щодо здорового способу життя. Ось деякі з них:

1. Алгоритми для аналізу фізичної активності:

1.1. Крокомір: визначає загальну кількість кроків, пройдених за день, і порівнює з рекомендованими значеннями (наприклад, 10 000 кроків на день).

1.2. Калькулятор витрачених калорій: обчислює кількість витрачених калорій на основі типу та тривалості фізичної активності.

1.3. Трекер тренувань: відстежує частоту та інтенсивність тренувань, порівнюючи їх із встановленими цілями.

2. Алгоритми для аналізу споживання калорій:

2.1. Калькулятор калорій: підраховує загальну кількість спожитих калорій на основі введених даних про харчування.

2.2. БЖУ-баланс: аналізує співвідношення білків, жирів і вуглеводів у раціоні користувача і порівнює з рекомендованими значеннями.

2.3. Трекер прийому їжі: відстежує час та частоту прийомів їжі, допомагаючи уникати переїдання або пропуску прийомів їжі.

3. Алгоритми для врахування індивідуальних параметрів:

3.1. Розрахунок основного обміну речовин (BMR): використовує формули, такі як формула Харріса-Бенедикта, для визначення кількості калорій, які спалює тіло у стані спокою.

3.2. Розрахунок загальної добової витрати енергії (TDEE): визначає загальну кількість калорій, які користувач повинен споживати щодня, з урахуванням рівня фізичної активності.

3.3. Індивідуальні цілі: враховує особисті цілі користувача (схуднення, набір ваги, підтримка ваги) для налаштування рекомендацій.

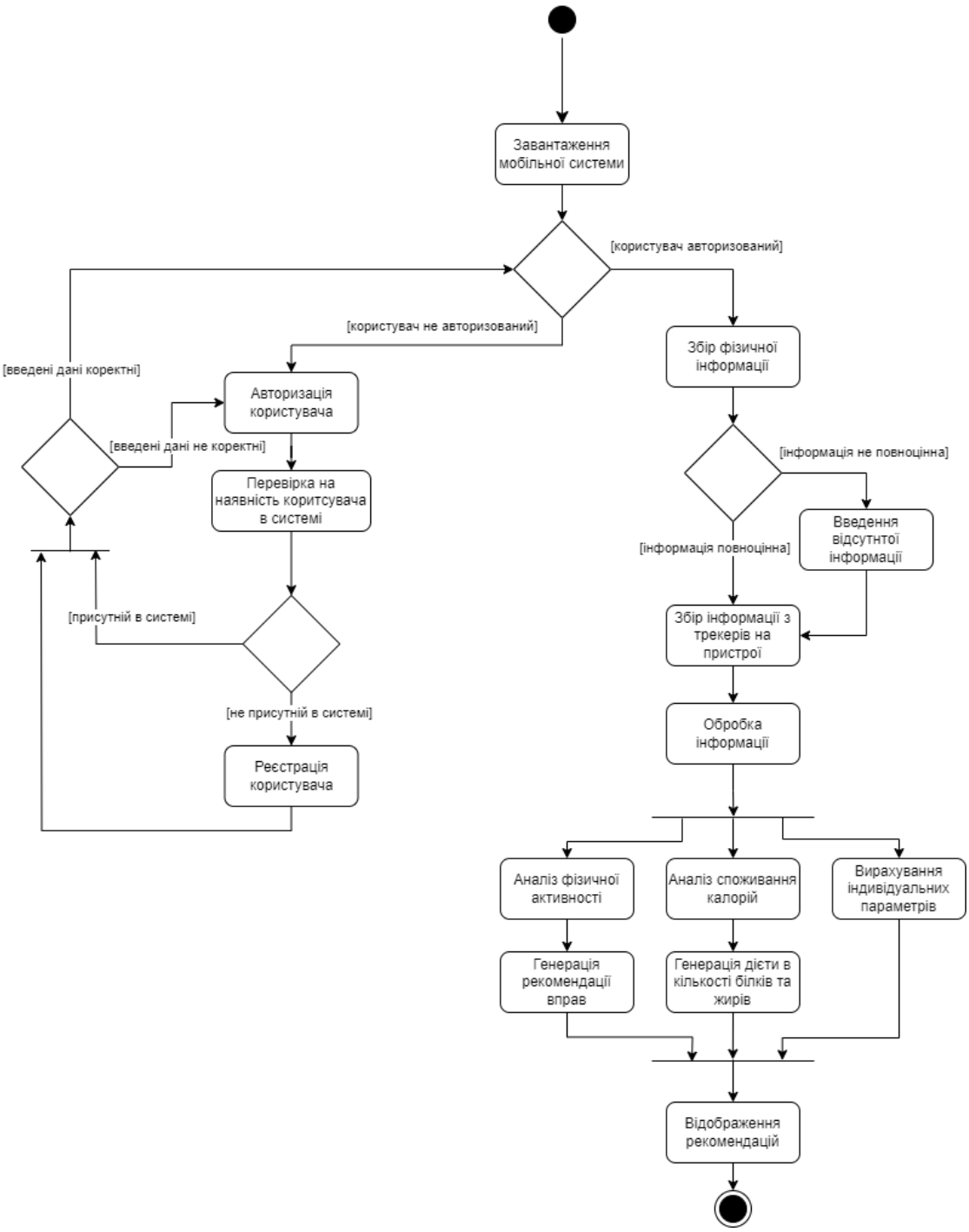

Рисунок 2.1 – Модель системи у вигляді діаграми діяльності

На основі оброблених даних модель генерує індивідуальні рекомендації. Ці рекомендації можуть включати пропозиції щодо збільшення або зменшення фізичної активності, зміна типу активності наприклад, додавання силових вправ до кардіо-тренувань, рекомендації щодо частоти та тривалості тренувань, рекомендації щодо споживання певних продуктів, корекція дієти для досягнення збільшення білків для набору м'язової маси, зменшення жирів для схуднення, пропозиції щодо режиму прийому їжі.

Рекомендації відображаються у зручному інтерфейсі користувача, який дозволяє легко взаємодіяти з системою. Інтерфейс включає:

1. Щоденні, тижневі та місячні звіти: графічне відображення фізичної активності, споживання калорій та досягнення цілей.
2. Персоналізовані повідомлення: сповіщення та нагадування про виконання тренувань, прийом їжі, досягнення цілей.
3. Інтерактивні функції: можливість користувачам вводити нові дані, коригувати свої цілі, отримувати зворотний зв'язок від системи.

Модель постійно вдосконалюється на основі нових даних. Кожен новий введений користувачем запис допомагає системі точніше налаштовувати рекомендації. Це дозволяє системі надавати все більш точні та ефективні рекомендації.

Персоналізована модель рекомендацій є потужним інструментом для підтримки здорового способу життя. Використання індивідуальних даних користувача дозволяє створювати рекомендації, які точно відповідають його потребам і цілям, що робить процес підтримки здоров'я більш ефективним та мотивуючим.

2.3 Розробка методу адаптивного планувальника завдань

Метод адаптивного планування завдань на основі фізичної та робочої активності являє собою інноваційне рішення, яке дозволяє автоматично коригувати розклад завдань з урахуванням втомленості та продуктивності користувача. Це забезпечує оптимізацію часу виконання завдань та підтримання балансу між роботою та відпочинком, що є критично важливим для підтримки високого рівня продуктивності та загального здоров'я користувача.

Метод включає такий функціонал:

1. Метод адаптивного планування завдань починається з авторизації користувача. Після введення облікових даних користувача, дані перевіряються на сервері за допомогою класу UserAuthService. У разі успішної авторизації токен зберігається у SharedPreferences для подальшого використання.
2. Після цього створюється інстанція бази даних SQLite через клас DatabaseHelper. DatabaseHelper містить методи для створення таблиць, таких як createTables(), які включають tasks, physical_activity, nutrition, sleep_quality та stress_levels.
3. Далі, збираються дані про фізичну активність користувача. Дані можуть надходити з трекера активності або вводитись вручну користувачем. Клас ActivityTracker зберігає ці дані у таблицю physical_activity за допомогою методу saveActivityData(), використовуючи змінні steps, distance, activeMinutes, activityType.
4. Після цього користувач вводить інформацію про своє харчування через інтерфейс додатку. Клас NutritionTracker зберігає дані про спожиті продукти у таблицю nutrition за допомогою методу saveNutritionData(), використовуючи змінні foodItem, calories, protein, fat, carbohydrates.
5. Користувач також вводить рівень стресу через опитувальник, і клас StressEvaluator зберігає ці дані у таблицю stress_levels за допомогою методу saveStressData(), використовуючи змінні stressLevel, date.

6. Наступним кроком користувач створює або оновлює завдання через інтерфейс додатку. Клас TaskManager зберігає дані про завдання у таблицю tasks за допомогою методів addTask() та updateTask(), використовуючи змінні taskId, taskName, description, dueDate, status, priority.
7. Потім дані про продуктивність та втомленість зчитуються з таблиць tasks, physical_activity та sleep_quality. Клас PerformanceAnalyzer використовує метод analyzePerformance() для розрахунку продуктивності та втомленості, використовуючи змінні taskCompletionTime, activeMinutes, sleepQuality, fatigueLevel.
8. Алгоритм адаптивного планування, реалізований у класі ScheduleAdjuster, коригує розклад завдань за допомогою методу adjustSchedule(), використовуючи дані про продуктивність та втомленість для оптимізації розкладу. Використовуються змінні currentTaskList, fatigueLevel, productivityScore, adjustedSchedule.
9. Відкоригований розклад завдань відображається користувачу через інтерфейс додатку, який оновлюється класом UIUpdater за допомогою методу updateUI(). Зміни зберігаються у базу даних через методи класу DatabaseHelper.
10. Дані про активність та завдання оновлюються у режимі реального часу через клас RealTimeUpdater. Метод syncData() синхронізує дані з локальної бази даних SQLite із сервером, якщо є підключення до Інтернету, використовуючи змінні localData, serverData.
11. Нарешті, клас NotificationService надсилає користувачу регулярні нагадування про заплановані завдання та необхідність фізичної активності за допомогою методів sendTaskReminder() та sendActivityReminder(). Цей метод дозволяє забезпечити адаптивне планування завдань з урахуванням фізичної активності та продуктивності користувача, використовуючи локальну базу даних

SQLite для швидкого доступу до даних та їхньої обробки в режимі реального часу.

Для кращого сприйняття методу розроблено алгоритм роботи планувальника завдань, блок-схема якого наведена на рис. 2.2.

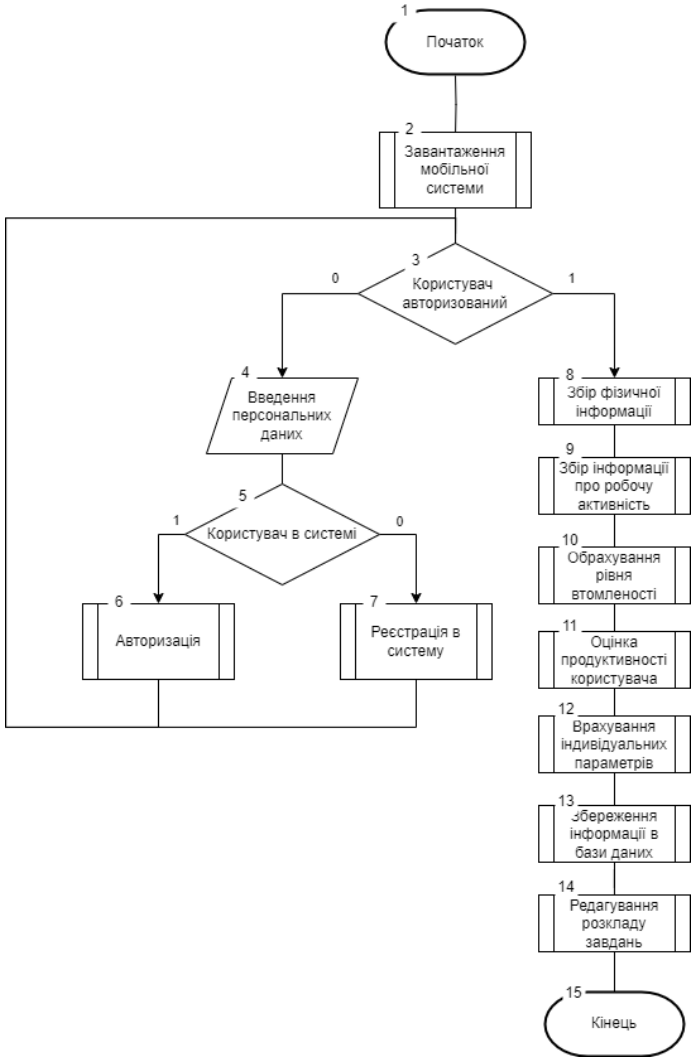

Рисунок 2.2 – Блок-схема алгоритму планувальника розкладу завдань

Робота алгоритму починається зі збору даних про фізичну активність користувача, які включають кількість кроків, витрачений час на тренування, пройдену відстань та типи фізичної активності. Ці дані зберігаються в базі даних системи і постійно оновлюються. Алгоритм також отримує інформацію про робочу активність користувача, яка включає заплановані завдання, дедлайни,

статус виконання завдань та інші параметри, що відображають робоче навантаження.

Після збору даних алгоритм аналізує фізичну активність користувача для визначення рівня втомленості. Наприклад, якщо користувач виявляє високу фізичну активність протягом дня, алгоритм може визнати його втомленим і запропонувати зменшити кількість робочих завдань або перенести їх на інший час. Аналогічно, якщо користувач має низький рівень фізичної активності, алгоритм може рекомендувати включення фізичних вправ у розклад для покращення загального стану здоров'я.

Важливою частиною роботи алгоритму є оцінка продуктивності користувача. Алгоритм аналізує історію виконання завдань, зокрема час, витрачений на виконання кожного завдання, та його успішність. Якщо користувач регулярно виконує завдання швидко та ефективно, алгоритм може запропонувати збільшити кількість завдань або підвищити їхню складність. Навпаки, якщо продуктивність знижується, алгоритм може рекомендувати перерви або зміну типу завдань для уникнення перевтоми.

Алгоритм також враховує індивідуальні параметри користувача, такі як вік, стать, вага, рівень втомленості. Ці дані зберігаються в базі даних і використовуються для налаштування рекомендацій. Наприклад, для користувачів старшого віку алгоритм може рекомендувати більше часу для відпочинку між завданнями, а для молодших користувачів - активніші фізичні вправи.

Алгоритм адаптивного планувальника завдань використовує дані з таблиць (рис. 2.3) physical_activity, work_tasks, sleep_quality, nutrition та stress_levels для автоматичного коригування розкладу завдань.

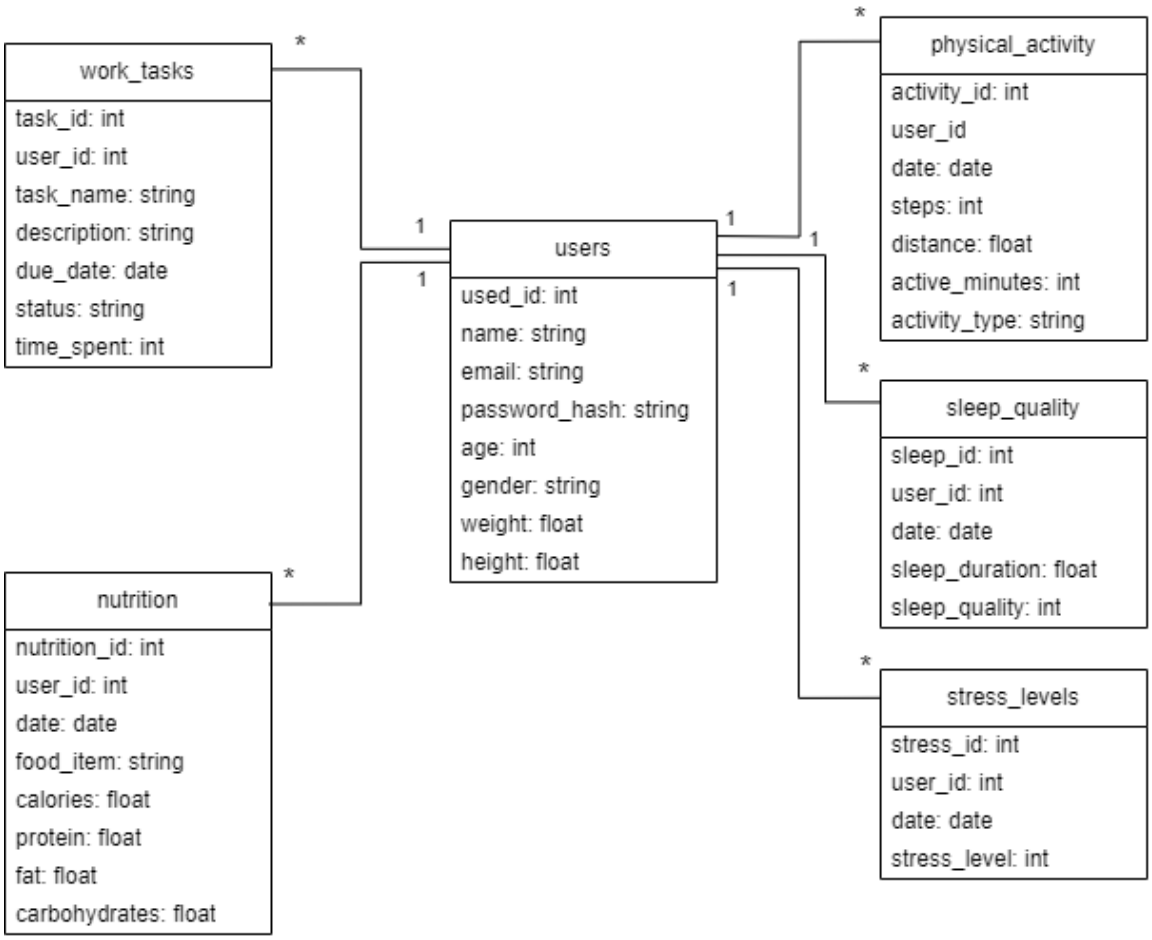

Рисунок 2.3 – Таблиці бази даних системи

Поле physical_activity включає дані про кількість кроків, час тренувань, типи активності та інші параметри. Поле work_tasks містить інформацію про заплановані завдання, дедлайни та статус виконання. Поле fatigue_level відображає розрахунковий рівень втомленості користувача, яке базується на аналізі фізичної активності та робочого навантаження. Поле productivity включає дані про ефективність виконання завдань, зокрема час та успішність їх виконання. Поле user_profile містить інформацію про вік, стать, вагу та інші індивідуальні параметри користувача. Поле stress_level містить дані про рівень стресу, який може впливати на здатність користувача виконувати завдання ефективно.

Коли користувач вводить нові дані про свою фізичну активність (кількість кроків, пройдену відстань, час активності) або про своє харчування (споживані калорії, білки, жири, вуглеводи), ці дані зберігаються в базі даних.

На основі цих даних алгоритм розраховує рівень втомленості користувача та продуктивність виконання завдань. Якщо користувач виконує багато фізичної активності та високий рівень втомленості, алгоритм може зменшити кількість робочих завдань або перенести їх на інший час, щоб уникнути перевтоми. Навпаки, якщо користувач має низький рівень фізичної активності та низький рівень стресу, алгоритм може збільшити кількість завдань або додати нові, щоб підвищити продуктивність.

Таким чином, алгоритм адаптивного планувальника завдань використовує комплексний підхід до управління часом, враховуючи фізичну активність, харчування та рівень втомленості користувача для оптимізації його розкладу завдань.

2.4 Розробка загального алгоритму роботи системи

Для розробки мобільної системи, що сприяє здоровому способу життя, необхідно створити загальний алгоритм функціонування, який визначатиме роботу застосунку (рис. 2.4). За цим алгоритмом початковий екран застосунку вітає користувача, пропонуючи авторизацію. У разі відсутності облікового запису користувача, він має можливість створити профіль, введенням особистих даних. Після успішної авторизації користувач потрапляє у головне меню, де він може обрати подальші дії. Користувач має можливість налаштувати свій профіль, перейшовши на відповідну сторінку, де може внести зміни або додаткові дані. Також доступний варіант запуску таймера для тренування, який пропонує простий таймер на одну годину.

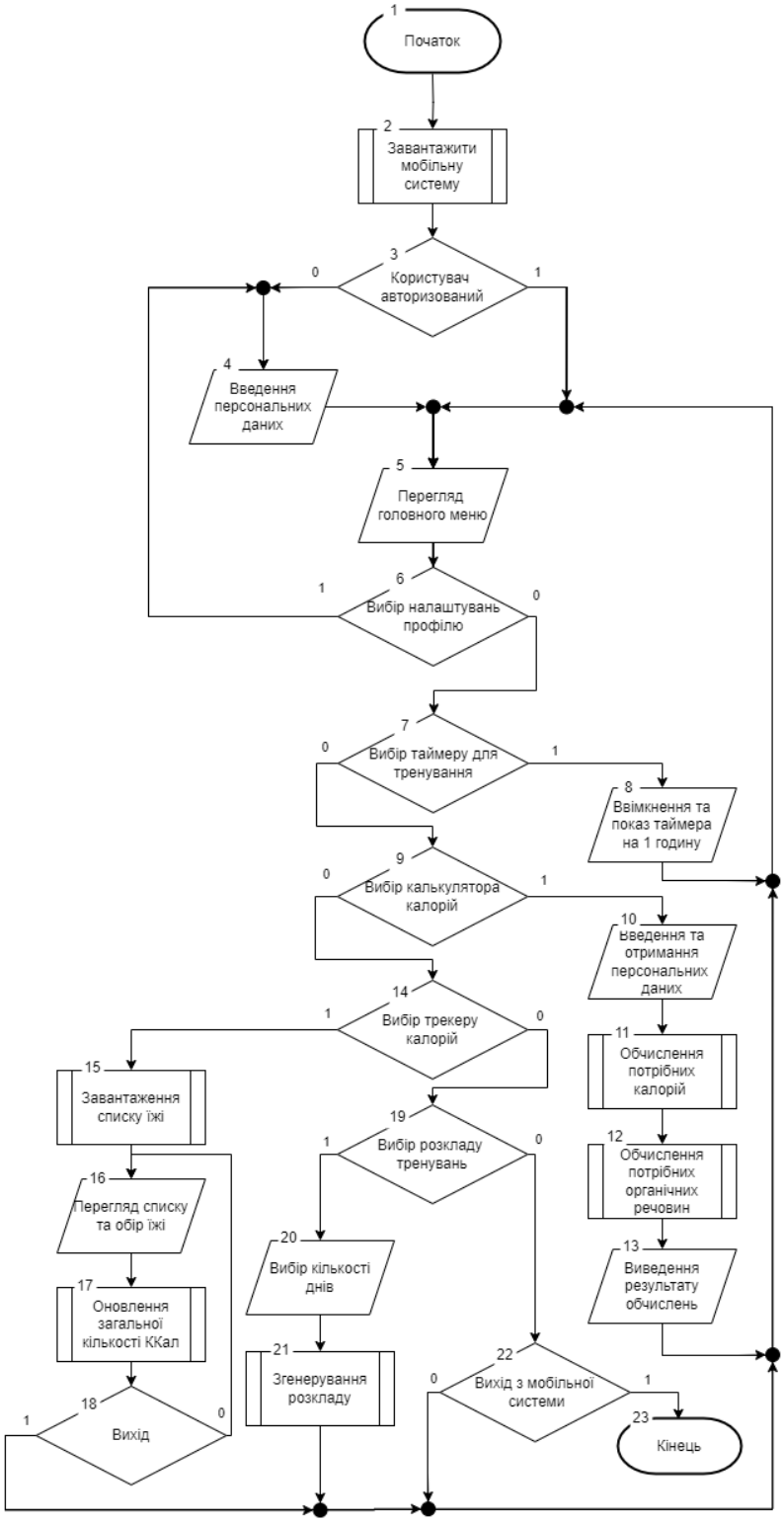

Рисунок 2.4 – Алгоритм роботи програмного застосунку

Крім того, користувач може скористатися калькулятором калорій, вводячи або надаючи дані з профілю для обчислення кількості калорій та органічних речовин для їх споживання. Додатково, користувач має можливість

використовувати трекер калорій, де представлений список страв, з яких він може обрати необхідну кількість для перегляду загальної кількості калорій. Останнім пунктом меню є вибір розкладу тренувань, який генерується після обрання кількості днів, і відображає рекомендовані вправи та активності на кожен день.

2.5 Висновки

У другому розділі було розглянуто детальний аналіз вхідних даних, необхідних для успішної розробки мобільної системи для підвищення ефективності використання робочого часу та підтримки здорового способу життя. Було розроблено концептуальну модель системи, включаючи персоналізовану модель рекомендацій та алгоритм адаптивного планування завдань, які забезпечують індивідуальний підхід до користувача. До моделі та алгоритму були створені UML-діаграму та блок-схему. Було розроблено метод адаптивного планування завдань, який використовує локальну базу даних SQLite для автоматичного коригування розкладу завдань з урахуванням втомленості та продуктивності користувача Також були розроблені алгоритми персоналізованих рекомендацій, які використовують бібліотеку Retrofit для інтеграції з REST API для отримання додаткових даних, а також бібліотеку Room для ефективної роботи з базою даних. Крім того, було розроблено алгоритм загальної роботи додатку, який включає функції авторизації користувача, моніторингу фізичної активності, та планування робочих завдань.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ МОБІЛЬНОЇ СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації мобільної системи

При розробці програмного забезпечення для підвищення ефективності використання робочого часу та підтримки здорового способу життя важливо обирати найбільш оптимальні технології для кожного модуля. Однією з основних технологій, яка використовується у розробці Android-додатків [12], є мова програмування Kotlin. Kotlin є стандартною мовою для розробки Android-додатків і забезпечує широкі можливості для створення функціональних та ефективних рішень. Завдяки Kotlin, можна легко інтегрувати різноманітні функції, такі як трекінг фізичної активності, управління розкладом завдань, моніторинг харчування та інші, що дозволяє створювати додатки, які максимально відповідають потребам користувачів у покращенні їхнього способу життя та продуктивності.

Layout Editor [13] у Android Studio є потужним інструментом для розробки графічного інтерфейсу користувача. Він надає зручний спосіб розміщення та налаштування різноманітних елементів інтерфейсу без прямого втручання в XML-код. Цей інструмент дозволяє розміщувати елементи за допомогою перетягування та операцій з розміщенням, що значно спрощує процес розробки. З Layout Editor можна швидко створювати складні інтерфейси, використовуючи вбудовані шаблони та компоненти. Крім того, він надає можливість переглядати інтерфейс у різних режимах, таких як режим дизайну, режим тексту або режим прев'ю на пристрої. Дана технологія підтримує різні версії Android, що дозволяєпереглядати та налаштовувати інтерфейс для різних пристроїв та роздільної здатності екрану. Він автоматично генерує відповідний XML-код, що дозволяє швидко переглядати зміни в реальному часі. Layout Editor також підтримує візуальну настройку атрибутів елементів, таких як розмір, кольори,

шрифти та інші. Це дозволяє зручно налаштовувати вигляд та поведінку елементів інтерфейсу без необхідності вручну введення значень.

SQLite - це легковага вбудована база даних, яка широко використовується в Android-розробці для зберігання та управління даними. Вона надає зручний спосіб створення та використання локальних баз даних без потреби встановлення додаткових серверів чи конфігурації складних середовищ. SQLite дозволяє розробникам створювати бази даних безпосередньо на мобільному пристрої, що робить його ідеальним вибором для мобільних додатків, що потребують локального зберігання даних. Він підтримує різні типи даних, включаючи текстові рядки, числа, дати та бінарні дані. SQLite використовує мову SQL для взаємодії з базою даних, що робить його зрозумілим та зручним для багатьох розробників. Він підтримує стандартні операції SQL, такі як вибірка, вставка, оновлення та видалення даних, що дозволяє ефективно взаємодіяти з даними в базі.

Android SDK – це набір інструментів для розробки програмного забезпечення для платформи Android [14]. Він включає в себе різноманітні ресурси, бібліотеки, засоби розробки та документацію, які дозволяють створювати різноманітні мобільні додатки для пристроїв, що працюють на операційній системі Android. Крім того, Android SDK містить SDK Manager, який дозволяє керувати версіями Android SDK та залежними від них компонентами, такими як платформи Android, засоби підтримки та системи зображень. Це дозволяє легко оновлювати своє середовище розробки та використовувати останні функції та покращення, які надаються Google для розробки додатків для Android.

Поєднання цих технологій дозволяє розробити мобільну систему, яка має високу продуктивність, ефективність та надійність. Використання Android SDK надає доступ до широкого спектру функціональності платформи Android, SQLite дозволяє зручно та ефективно зберігати та управляти даними на мобільних пристроях, забезпечуючи високу швидкість та надійність операцій з базою

даних. З допомогою Layout editor можна легко створювати та налаштовувати макети екранів, розміщати різноманітні елементи і керувати їх властивостями.

Отже, розробка програмних модулів для мобільної системи для підтримки здорового способу життя за допомогою SQLite, Android SDK та Layout Editor має вирішальне значення для створення інноваційного та користувацько-орієнтованого додатку. Таке поєднання технологій дозволяє ефективно використовувати ресурси мобільного пристрою, забезпечуючи оптимальну продуктивність та ефективність мобільної системи.

Для реалізації програмного продукту для підвищення ефективності використання робочого часу та підтримки здорового способу життя використовуються Kotlin [15] та Android Studio [16]. Kotlin є мовою програмування, яка широко використовується для розробки мобільних додатків на платформі Android. Вона володіє великим спектром функціональності та добре підтримується Android-середовищем. Android Studio є інтегрованою середою розробки (IDE), призначеною спеціально для розробки Android-додатків. Вона надає розробникам широкий набір інструментів для створення, тестування та налагодження додатків, що прискорює процес розробки. Android Studio має вбудовану підтримку для Kotlin, а також ряд інших мов програмування, що дозволяє розробникам вибирати найбільш зручний для них інструмент.

Використання Kotlin та Android Studio дозволяє розробникам створювати потужні та ефективні мобільні додатки для підвищення ефективності використання робочого часу та підтримки здорового способу життя на платформі Android. Ці інструменти забезпечують швидкий та зручний спосіб створення додатків з багатою функціональністю та дружнім інтерфейсом користувача.

Розробка програмних модулів для реалізації мобільної системи підвищення ефективності використання робочого часу та підтримки здорового способу життя передбачає кілька етапів. По-перше, потрібно створити модуль,

що відповідає за ведення щоденника харчування. Цей модуль повинен дозволяти користувачам реєструвати споживані продукти, вводити кількість та склад продуктів, а також автоматично розраховувати кількість калорій. Далі необхідно розробити модуль для відстеження фізичних активностей. Цей модуль має вимірювати показники, такі як кроки, відстань та час тренувань, а також надавати користувачам зручні інструменти для аналізу їхньої фізичної активності. Також слід розробити модуль планування завдань, який дозволить користувачам ефективно розподіляти час на виконання робочих задач.

До цього ще потрібно розробити модуль для надання персоналізованих рекомендацій. Цей модуль має аналізувати дані про харчування та фізичну активність користувачів і на їхній основі надавати рекомендації з покращення здорового способу життя та ефективності використання робочого часу. Нарешті, потрібно створити модуль інтерфейсу користувача, який забезпечить зручну та привабливу взаємодію з розробленими функціями. Цей модуль має включати у себе графічний інтерфейс, а також можливості для персоналізації та налаштувань, щоб користувачі могли максимально комфортно використовувати систему.

Розробка цих програмних модулів вимагає глибокого розуміння принципів розробки мобільних додатків для платформи Android. Для ефективної розробки та тестування використовується Android Studio, інтегрована середа розробки, яка надає розробникам широкий спектр інструментів для створення високоякісних додатків. Для реалізації цих модулів будуть використані різноманітні інструменти та бібліотеки програмування, зокрема розширений набір Android SDK, який включає в себе багато корисних компонентів для розробки мобільних додатків. Основні компоненти Android SDK, такі як Activity [17], Fragment, RecyclerView [18], інструменти для роботи з базою даних SQLite [19].

Отже, розробка програмних модулів для підвищення ефективності використання робочого часу та підтримки здорового способу життя на Kotlin та Android Studio є комплексним завданням, яке включає кілька етапів. Модулі

повинні бути ретельно спроектовані, реалізовані та протестовані, щоб забезпечити їхню ефективність, надійність та відповідність вимогам користувачів. Маючи необхідні інструменти та досвід, можна створити високоякісний та функціонально повний мобільний додаток для підвищення ефективності використання робочого часу та підтримки здорового способу життя, який відповідатиме потребам сучасних користувачів.

3.2 Розробка модуля трекера споживання калорій

Однією з головних функцій мобільної системи для підтримки здорового способу життя є трекер споживання страв з урахуванням калорій. Першим кроком у розробці цього модуля є побудова інтерфейсу користувача за допомогою XML. Це включає створення макету екрану, додавання текстових полів, кнопок та пустого списку, який буде використовуватися для відображення страв.

Далі необхідно приєднати цей інтерфейс до відповідного класу, щоб додати необхідний функціонал. Це включає обробку подій кнопок, контроль за текстовими полями та управління списком страв. При натисканні на контейнер зі стравою, програма додає калорії натиснутої користувачем страви.

Такий підхід дозволяє створити зручний та ефективний інтерфейс для користувача, який дозволяє легко вести облік спожитих страв та контролювати калорійні витрати.

Використання XML для побудови інтерфейсу та зв'язування його з Kotlin-класами дозволяє розробникам ефективно організовувати та управляти функціональністю додатку. Цей підхід до розробки дозволяє забезпечити чітку структуру та поділ функціональності програми на окремі компоненти. XML визначає вигляд елементів інтерфейсу, включаючи їх розміщення та властивості.

XML може використовуватися для опису анімацій, переходів між екранами та інших візуальних ефектів. Це робить можливим створення динамічних та привабливих інтерфейсів, які покращують користувацький досвід.

Наступним кроком є програмування функціоналу додавання страв в список та їх відображення на інтерфейсі. Це може бути досягнуто за допомогою Kotlin-коду, який додає нові елементи до списку та оновлює інтерфейс для відображення нової інформації.

Після додавання страв до списку, можливо налаштувати обробники подій для взаємодії з елементами списку – при натисканні на певний елемент списку знаходиться кількість калорій страви цього елементу та додається до загальної кількості.

Для більшої ясності та розуміння принципів роботи модуля трекера калорій було розроблено додатковий алгоритм (рис. 3.1). За цим алгоритмом спочатку завантажується відповідний екран трекера калорій.

Після цього з бази даних витягуються інформація про страви, і за допомогою циклу створюється список, в якому кожна страва представлена окремим контейнером з необхідними даними: назвою страви, кількістю калорій та зображенням. Користувачу показується цей список, і загальна кількість калорій оновлюється автоматично.

Далі користувач має можливість натиснути на контейнер, щоб додати калорії цієї страви до загальної суми, або натиснути кнопку "Рестарт", щоб скинути загальну кількість калорій до початкового значення. Також він може залишити модуль, якщо вважає, що завершив роботу з трекером калорій.

Кожна страва відображається з інформацією про кількість калорій, що допомагає користувачам керувати своїм харчуванням та визначати, чи вони вже досягли своєї цільової кількості калорій на день. Така деталізація інформації робить використання модуля трекера калорій більш зручним та ефективним для користувачів.

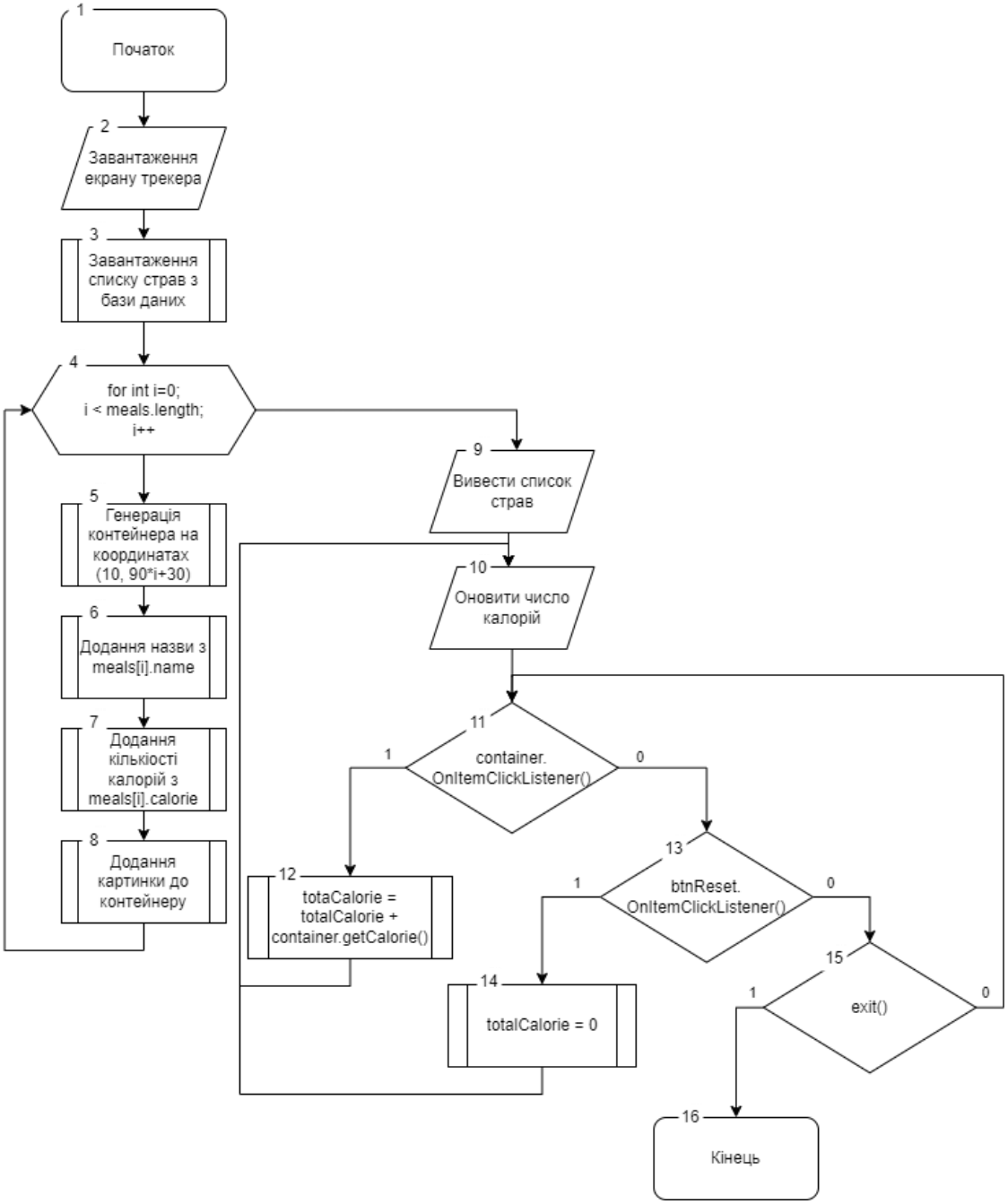

Рисунок 3.1 – Алгоритм роботи модуля трекера калорій

Цей алгоритм взаємодіє з локальною базою даних, щоб забезпечити користувачеві доступ до списку страв. Під час роботи програми, він використовує дані з цієї бази даних, щоб завантажити і відобразити доступні страви для вибору користувачем. Після завантаження алгоритм вставляє ці страви в список, який користувач може переглядати та вибирати.

3.3 Розробка модуля обрахування кількості калорій та органічних речовин для споживання

Головною метою модуля для обрахунку кількості калорій та органічних речовин для споживання є точність та ефективність використання визначених формул. Щоб впевнитися у відповідності результатів здоровим нормам, необхідно чітко використовувати встановлені методи обрахунку. Для досягнення цієї мети спочатку створюється інтерфейс за допомогою XML, що містить в собі поля для введення даних та поля, в які буде виведена інформація про кількість калорій та органічних речовин, необхідних для споживання.

Для кращого розуміння роботи модуля обрахунків кількості калорій було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 3.2).

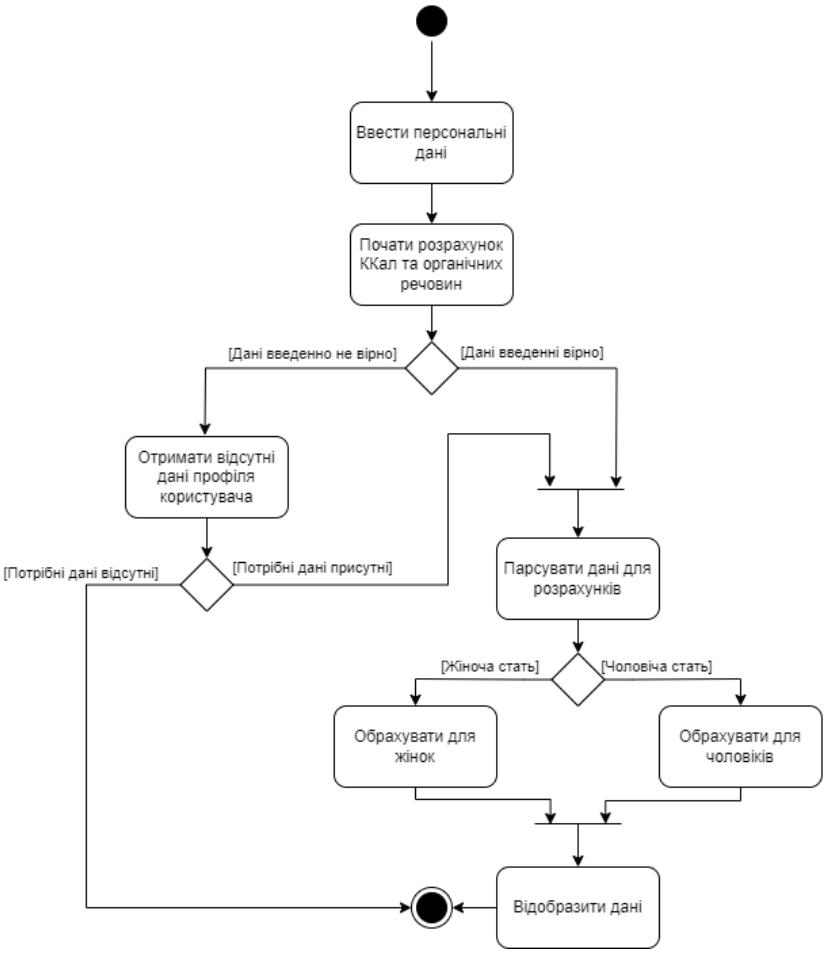

Рисунок 3.2 – Діаграма діяльності мобільної системи

Розробка діаграми діяльності була виконана у програмному забезпеченні Draw.io [20]. Цей інструмент дозволив зобразити послідовність кроків і операцій, необхідних для досягнення певного функціонального результату в системі.

Крім того, створюється клас Recommendation, який взаємодіє з інтерфейсом та виконує обрахунки з використанням введених користувачем даних, таких як стать та інші особисті показники. Важливою особливістю цього модуля є його здатність до адаптації під індивідуальні потреби користувача. Він враховує різні фактори, такі як вік, стать, рівень активності та особисті цілі, і надає користувачу збалансовані рекомендації щодо споживання калорій та поживних речовин. Такий підхід дозволяє індивідуалізувати досвід використання додатку та забезпечує оптимальні результати для кожного користувача.

Даний клас Recommendation використовує визначені формули [21], що враховують стать користувача та інші особисті характеристики. Для чоловіків викликається функція maleCheck().

Для жінок викликається функція femaleCheck(), яка в результаті видасть чуть нижче значення калорій, оскільки враховується відмінність у фізіологічних потребах. Ця функція базується на наукових дослідженнях та стандартних рекомендаціях з харчування, спеціально адаптованих для жіночого організму. Враховуючи такі фактори, як метаболізм, вага, вік і рівень активності, ця функція дозволяє точніше визначити потрібну кількість калорій для жінки, щоб забезпечити оптимальне функціонування її організму.

3.4 Розробка модуля підвищення ефективності використання робочого часу

Модуль для підвищення ефективності використання робочого часу є ключовим компонентом мобільної системи, спрямованої на оптимізацію робочих процесів та підтримку здорового способу життя. Модуль поєднує

функціональність управління завданнями, моніторингу продуктивності та адаптивного планування, що дозволяє користувачам ефективно розподіляти свій час і зберігати баланс між роботою та відпочинком.

Для кращого розуміння роботи модуля було зроблено модель роботи системи на прикладі блок-схеми (рис. 3.3).

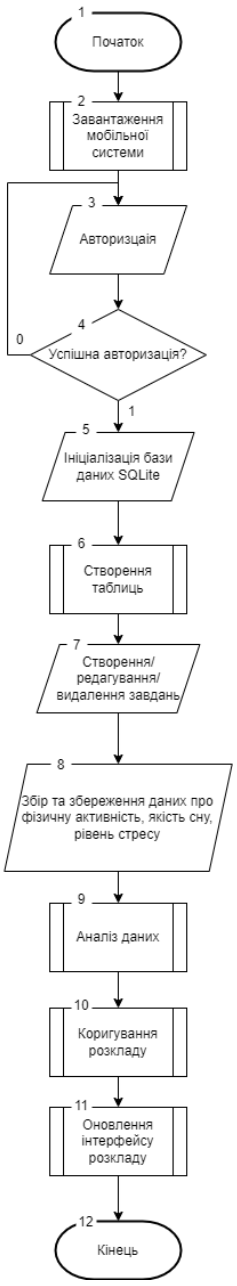

Рисунок 3.3 – Блок-схема алгоритму модуля для підвищення ефективності використання робочого часу

Користувачі можуть створювати, редагувати та видаляти робочі завдання. Кожне завдання має такі атрибути, як назва, опис, дедлайн, статус виконання та пріоритет. Це дозволяє організовувати робочий процес і слідкувати за виконанням завдань. Система відстежує час, витрачений на виконання кожного завдання, та оцінює продуктивність користувача. Дані про продуктивність зберігаються у базі даних і використовуються для аналізу ефективності роботи. Алгоритм адаптивного планування завдань коригує розклад з урахуванням фізичної активності, рівня втомленості та продуктивності користувача. Це забезпечує оптимізацію робочого процесу, запобігає перевантаженню та підтримує баланс між роботою та відпочинком.

Модуль взаємодіє з базою даних системи, використовуючи такі таблиці:

- work_tasks: Зберігає дані про робочі завдання користувачів, включаючи назву, опис, дедлайн, статус та час, витрачений на виконання;
- physical_activity: Відображає дані про фізичну активність користувачів, які використовуються для визначення рівня втомленості та адаптації розкладу;
- stress_levels: Зберігає дані про рівень стресу користувачів, що також враховується при плануванні завдань.

Модуль для підвищення ефективності використання робочого часу є потужним інструментом для оптимізації робочих процесів та підтримки здорового способу життя. Використовуючи дані про фізичну активність та рівень стресу, алгоритм адаптивного планування дозволяє індивідуалізувати розклад завдань, підвищуючи продуктивність та зменшуючи ризик перевтоми.

3.5 Розробка дизайну інтерфейсу мобільної системи

Під час створення інтерфейсу програмного забезпечення було акцентовано увагу на його простоті та зручності для користувача [22].

Основним кольором інтерфейсу обрано колір «Білий пар»(#F6F6F2), оскільки даний колір є приємним на око. Допоміжними кольорами обрано

кольори «Пудрово-синій» (#BADFE7) та його відтінок «Місячний камінь-синій» (#6FB3B8), оскільки вони контрастують основному кольору.

При відкритті мобільного додатку, користувача зустрічає екран з назвою та мотиваційна цитата (рис. 3.4), також відбувається невелика анімація пересунення тексту та з’явлення кнопки «РОЗПОЧАТИ».

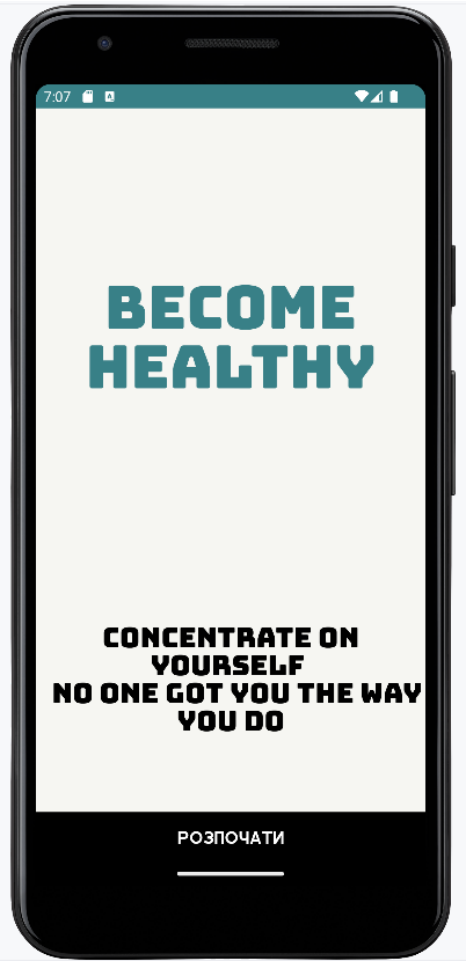

Рисунок 3.4 – Екран завантаження

Було імплементовано екран для реєстрації користувачів, для запису початкових даних для їх подальшої обробки (рис. 3.5). На даному екрані користувачу необхідно ввести його персональні данні: ім’я – для назви профілю, вік, зріст, вага, стать – для обчислень та пошту – для прив’язки. Після створенню профіля користувача зустрічає головний екран (рис. 3.6),

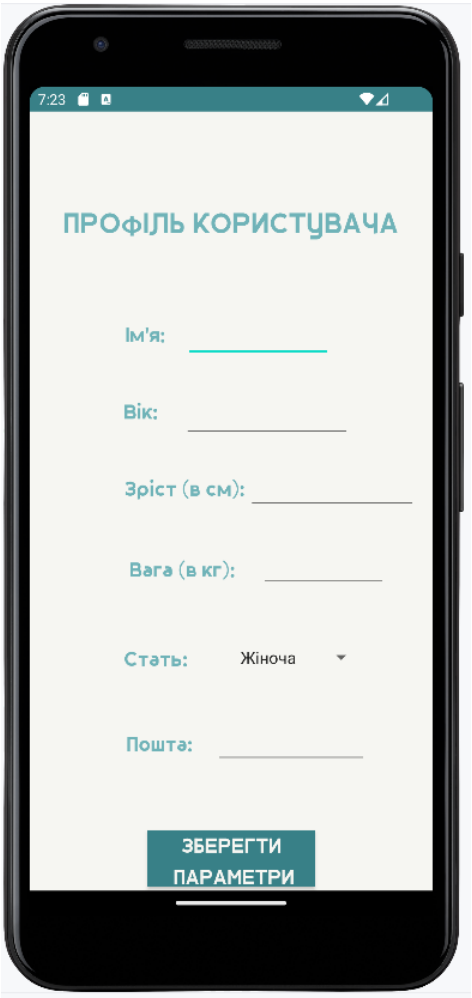

Рисунок 3.5 – Екран створення профілю

Після натискання кнопки «Розрахувати», програма використовує введені користувачем дані, такі як вага, зріст, вік та інші параметри, і застосовує вбудовані формули для обчислення калорій. Цей процес передбачає складні математичні розрахунки, які враховують індивідуальні характеристики користувача та визначають кількість калорій, які йому потрібно споживати протягом дня. Такий підхід дозволяє точно визначити денну норму калорій для користувача, враховуючи його унікальні фізіологічні особливості.

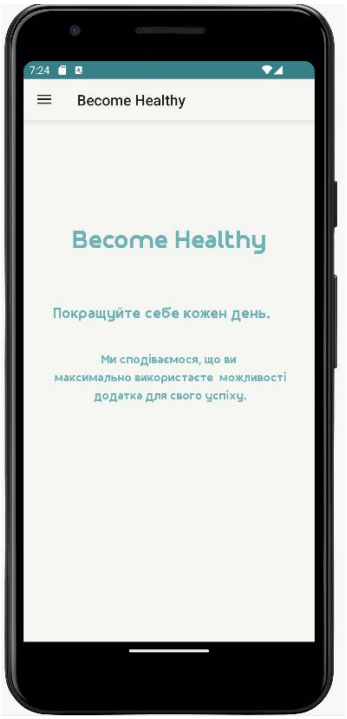

Рисунок 3.6 – Головний екран

В головному меню користувач може перейти до інших екранів за допомогою меню (рис. 3.7).

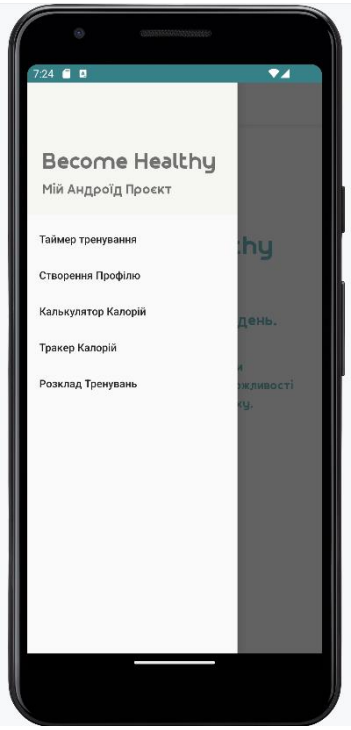

Рисунок 3.7 – Екран з меню

В меню користувач може обрати один з п’яти пунктів, які відкриють вікна з розробленими модулями. Одним з таких є простий Таймер для тренування який користувач може активувати і перейти на екран з таймером налаштованим на 1 годину (рис. 3.8). За допомогою таймера можна дізнатися скільки залишилось часу до кінця підходу, коли розпочнеться та закінчиться перерва між підходами, скільки залишилось часу до кінця тренування кардіо. Кнопка нижче була створена для користувачів, які випадково ввели не ті данні, або вони вже змінилися і потрібно їх оновити.

Рисунок 3.8 – Екран таймера для тренувань

Наступним пунктом у меню додатка є калькулятор калорій, який відображається на екрані (рис. 3.9). Цей інструмент призначений для використання раніше введених користувачем даних з метою визначення денного раціону калорій та органічних речовин, таких як вуглеводи, білки та жири. Крім

того, до цього модулю було додано додатковий функціонал - тракер калорій (рис. 3.10), який дозволяє користувачу вести облік кількості спожитих калорій через прийом їжі. Користувач може вибирати продукти зі списку, щоб дізнатися кількість калорій, що містяться в кожному з них, і враховувати це при плануванні їжі на день. Це дозволяє користувачеві контролювати свій харчовий раціон та досягати бажаних результатів у збереженні або зниженні ваги.

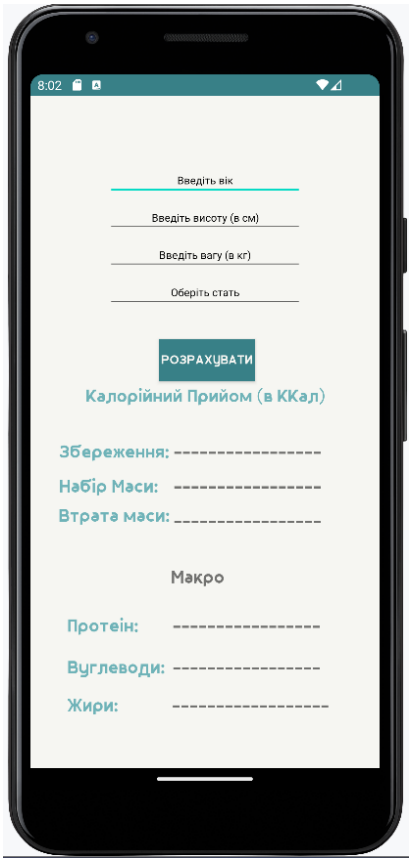

Рисунок 3.9 – Екран калькулятора калорій

Після натискання кнопки «Розрахувати», програма починає обробку введених користувачем даних, використовуючи вбудовані алгоритми та формули для обчислення калорій. Цей процес включає в себе аналіз введених параметрів, таких як вага, зріст, вік, рівень фізичної активності тощо, та їх подальше застосування до визначених математичних моделей для отримання результату

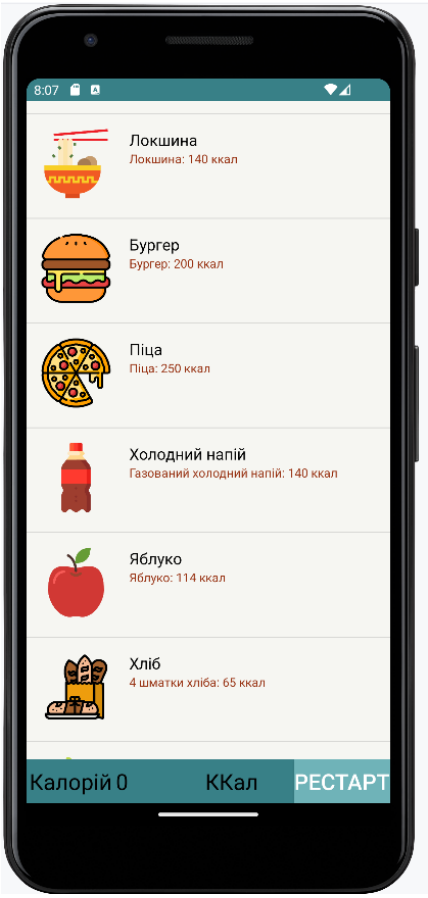

Рисунок 3.10 – Екран тракера калорій

Останнім модулем є розклад тренувань (рис. 3.11), що надає користувачеві можливість планувати свої тренування в зручний для нього спосіб. На цьому екрані користувач має можливість вибрати рівень інтенсивності тренувань, відповідно до своїх потреб та можливостей. Чим більше днів обрано для тренувань, тим менш інтенсивним буде розклад, щоб забезпечити зручність та комфорт під час фізичних навантажень. Після вибору, користувач перенаправляється на екран з покроковими інструкціями, які допомагають йому планувати та виконувати вправи протягом дня. Такий підхід дозволяє зберігати систематичність та налагоджувати режим тренувань відповідно до індивідуальних потреб користувача.

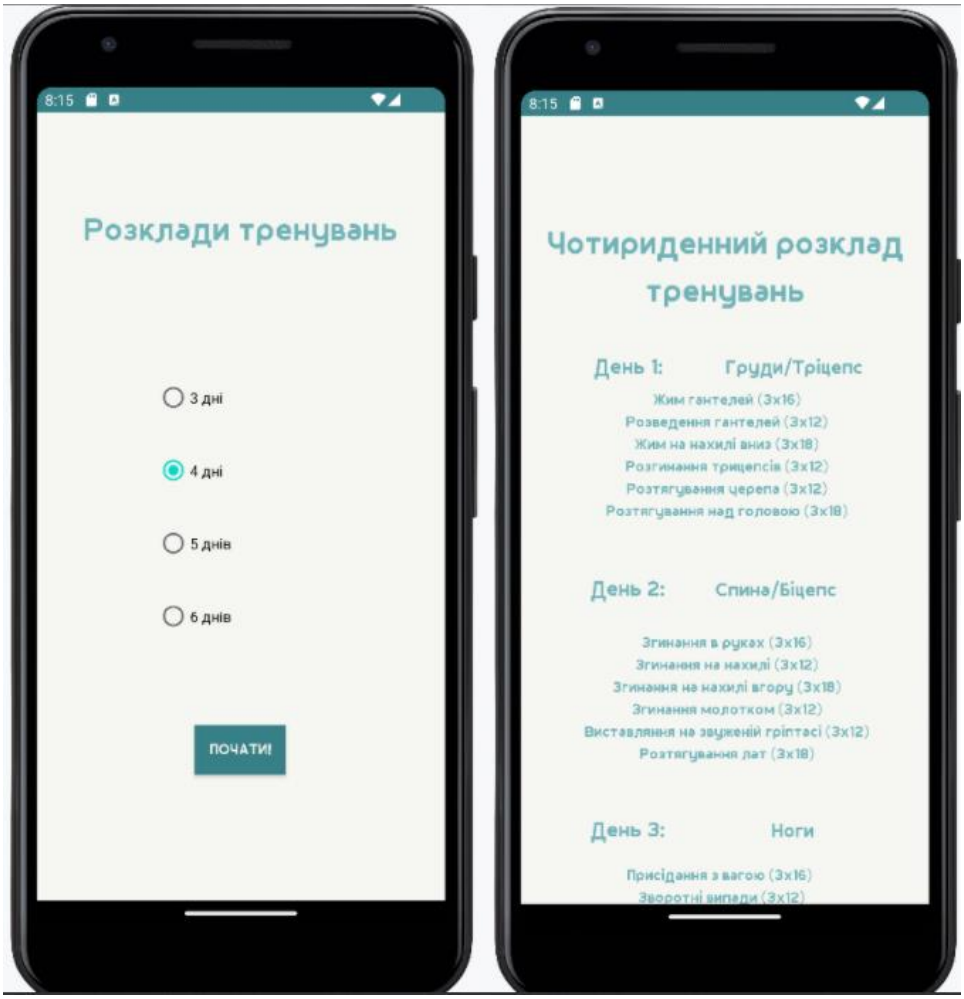

Рисунок 3.11 – Екрани вибору розкладу та розкладу тренування

Розробка графічного інтерфейсу програмного застосунку була проведена у середовищі розробки Android Studio з використанням технології XML. XML забезпечує можливість описувати структуру та зовнішній вигляд елементів інтерфейсу, а також їх взаємодію, зберігаючи дані у текстовому форматі. Ця технологія дозволяє ефективно організовувати інтерфейс, розташовуючи елементи на екрані та встановлюючи їх взаємодію з користувачем за допомогою атрибутів та подій. Використання Android Studio дозволяє швидко розробляти та тестувати інтерфейс, забезпечуючи оптимальну продуктивність та сумісність з різними пристроями на платформі Android.

3.6 Висновки

У третьому розділі детально розглянуто вибір технологій, мови програмування та інструментів, що застосовувалися під час розробки програмних модулів. Після ретельного аналізу вимог і можливостей, було вирішено використовувати Android Studio як основне середовище розробки, а також мову програмування Kotlin. Використання Android Studio надало змогу ефективно створювати та управляти проектом завдяки його широкому набору інструментів. Окрім цього, для реалізації графічної частини додатку був обраний Layout Editor, що дозволило зручно розміщувати елементи на екрані та налаштовувати їх відповідно до потреб користувача. Для збереження та управління даними було обрано SQLite, легка та потужна реляційна система управління базою даних (СУБД), яка ідеально підходить для мобільних додатків. SQLite забезпечив надійне зберігання і доступ до інформації про страви та калорійні обчислення, забезпечуючи оптимальну швидкість та ефективність додатку.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування мобільних систем на платформі Android – це процес перевірки роботи програмного забезпечення з метою виявлення помилок, недоліків та неполадок в функціональності та взаємодії з користувачем. Цей процес включає в себе проведення різних видів тестів, які охоплюють різні аспекти додатку, від його інтерфейсу та коректності роботи функцій до забезпечення безпеки та стійкості.

Після встановлення тестової версії програми на мобільний пристрій Android розпочинається процес функціонального тестування, що є важливою частиною процесу розробки та випробувань програмного забезпечення. Цей етап включає в себе проведення різних видів тестів, спрямованих на перевірку коректності та ефективності роботи додатку.

Під час функціонального тестування виконуються різноманітні тести, спрямовані на перевірку різних аспектів програми. Одним із найважливіших є тести на введення та виведення даних, що перевіряють правильність обробки вхідних даних програмою та коректність їх виведення. Також проводяться тести на реакцію програми на негативні вхідні дані, спрямовані на виявлення та виправлення можливих помилок та виключень під час роботи програми.

Крім того, важливим аспектом функціонального тестування є проведення тестів на відповідність бізнес-логіки програми. Ці тести дозволяють перевірити, чи працює програма відповідно до вимог та очікувань щодо її функціональності та поведінки.

Тестування відмов є ключовим етапом у процесі випробувань мобільних систем на Android. Цей етап спрямований на виявлення потенційних дефектів та неполадок у програмному забезпеченні, що можуть виникнути в реальних умовах експлуатації.Під час тестування відмов проводиться аналіз реакції

програмного забезпечення на непередбачені ситуації та виняткові випадки. Це включає в себе штучне введення неправильних даних, спроби взаємодії з програмою в умовах, які можуть виникнути у реальному середовищі користувача, а також тестування реакції програми на відмови та помилки.

Одним з основних завдань тестування відмов є перевірка здатності програмного забезпечення відновлювати роботу після виникнення відмов та відновлення коректного стану. Це важливо для забезпечення стабільної та безперебійної роботи додатку у реальних умовах експлуатації. Таке тестування дозволяє ідентифікувати слабкі місця в програмному коді та виправити їх до випуску продукту на ринок.

Тестування сумісності є важливим етапом [23] у процесі випробувань мобільних систем на Android. Це спрямоване на перевірку взаємодії програмного забезпечення з різними операційними системами, версіями програм та апаратними платформами. Під час тестування сумісності проводиться перевірка роботи програми на різних конфігураціях та середовищах. Це включає в себе перевірку сумісності з різними версіями операційної системи Android, різними моделями та типами пристроїв, а також з різними версіями програмного забезпечення, що встановлені на пристрої. Основні завдання тестування сумісності полягають у виявленні та виправленні можливих конфліктів, проблем або несправностей, які можуть виникнути під час взаємодії програми з різними компонентами або іншими програмами, що встановлені на пристрої. Також важливо переконатися, що програма працює стабільно та ефективно на різних пристроях із різними характеристиками та налаштуваннями.

Під час тестування програмного забезпечення проводиться перевірка реакції програми на невірний формат даних для авторизації користувача. Кожен з вхідних параметрів, таких як ім'я, зріст, вік, вага та адреса електронної пошти, піддається детальній перевірці. Якщо будь-яке з цих полів має недопустимі значення або їх відсутність, програма відображає відповідне повідомлення та позначає відповідне поле червоним кольором, щоб вказати на помилку.

Наприклад, якщо ім'я користувача відсутнє або невірного формату, програма позначає це поле червоним кольором та відображає повідомлення про необхідність введення коректних даних (рис. 4.1). Також, дані про зріст, вік та вагу перевіряються на відсутність та на те, щоб вони не мали від'ємне значення (рис. 4.2).

Рисунок 4.1 – Спроба авторизації без введення імені

Під час перевірки адреси електронної пошти програма використовує регулярні вирази, щоб здійснити валідацію введеного тексту. Регулярні вирази -

це шаблони, що дозволяють визначити правильний формат для адреси електронної пошти. Вони перевіряють, чи відповідає введений текст цьому шаблону. Під час валідації адреси електронної пошти програма перевіряє, чи присутні введені дані символи "@", "." (рис. 4.3), а також інші обов'язкові складові електронної адреси. Якщо будь-яка з цих умов не виконується, програма вважає введену адресу електронної пошти недійсною і відображає відповідне повідомлення про помилку. Такий підхід допомагає запобігти введенню некоректних адрес електронної пошти і забезпечити коректну роботу програми. Крім того, валідація адреси електронної пошти за допомогою регулярних виразів є стандартною практикою в програмуванні і дозволяє забезпечити високий рівень безпеки та надійності вводу даних користувачем.

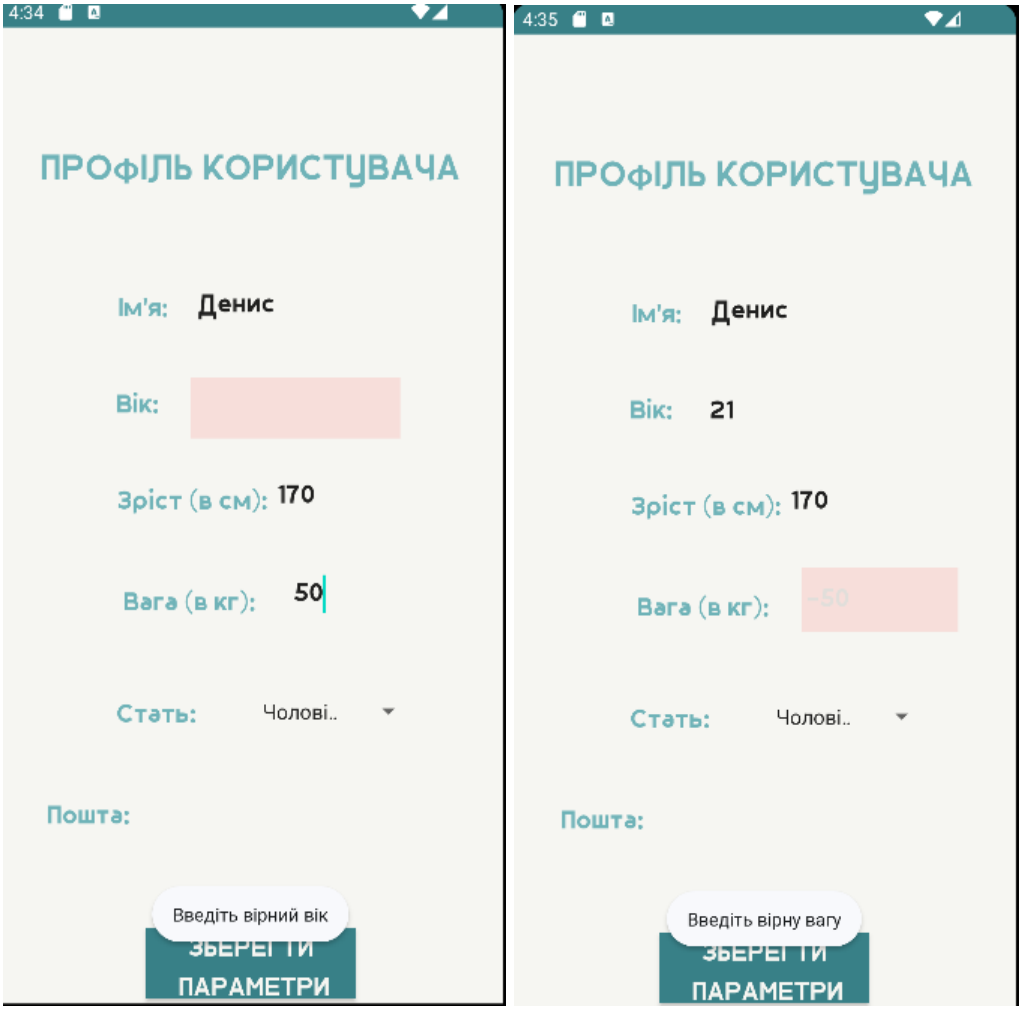

Рисунок 4.2 – Спроба авторизації з невірним віком та вагою

Використання кольорової позначки червоним кольором для помилкових полів допомагає користувачам швидко знайти та виправити неправильно введені дані, забезпечуючи більшу точність та надійність роботи програмного забезпечення.

Рисунок 4.3 – Спроба авторизації з невірною поштою

Останнім етап тестування є розклад тренувань, що надає користувачеві можливість планувати свої тренування в зручний для нього спосіб. На цьому екрані користувач має можливість вибрати рівень інтенсивності тренувань,

відповідно до своїх потреб та можливостей. Чим більше днів обрано для тренувань, тим менш інтенсивним буде розклад, щоб забезпечити зручність та комфорт під час фізичних навантажень. Після вибору, користувач перенаправляється на екран з покроковими інструкціями, які допомагають йому планувати та виконувати вправи протягом дня. Після завершення реалізації модуля, проводився тестувальний процес, який включав у себе перевірку коректності відображення розкладу (рис. 4.4), правильності підбору інтенсивності тренувань, а також перевірку функціональності кожного елементу розкладу.

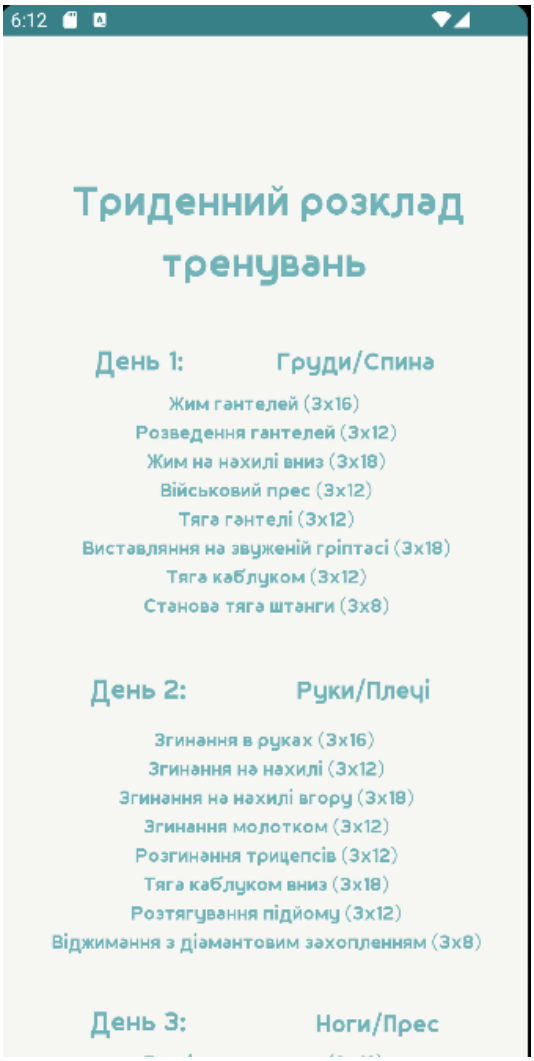

Рисунок 4.4 – Перевірка коректності відображення розкладу

Такий підхід дозволяє зберігати систематичність та налагоджувати режим тренувань відповідно до індивідуальних потреб користувача.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації розміщено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	**Рекомендовано**	**Мінімально**
Операційна система	Android 10.0	Android 7.0
Процесор	Octa-core 2.0 GHz	Quad-core 1.2 GHz
Оперативна пам'ять	4 ГБ	2 ГБ
Внутрішня пам'ять	64 ГБ	16 ГБ
Підтримка GPS	Так	Так

Після запуску платформи користувач повинен авторизуватися в системі та перейти на головний екран. В результаті він отримає доступ до основних функцій системи, таких як введення особистих даних (ім'я, вік, стать, вага, висота), вибір цілей (збереження, набір або скидання ваги), створення та перегляд розкладу тренувань, ведення щоденника споживання їжі та перегляд статистики. При цьому буде забезпечено зручний та інтуїтивно зрозумілий інтерфейс, що дозволить користувачеві легко орієнтуватися в функціях програми та використовувати їх для досягнення своїх цілей з підтримки здорового способу життя. Користувач може налаштувати свій профіль, перейшовши на відповідну сторінку, де може вносити зміни до основних даних або додавати додаткову інформацію. Це дозволяє кожному користувачеві персоналізувати свій досвід використання системи, враховуючи його особисті потреби та цілі. Додатково, є

можливість запуску таймера для тренування, що надає простий і зручний інструмент для відслідковування часу під час фізичних вправ. Такий таймер може бути корисним для тих, хто прагне контролювати тривалість своїх тренувань та планувати їх відповідно до власного графіка.

Крім того, доступний калькулятор калорій (рис. 4.5) надає можливість обчислити кількість калорій та основних поживних речовин для забезпечення відповідної харчової раціону. Користувач може ввести власні дані або скористатися інформацією зі свого профілю для точних розрахунків.

Рисунок 4.5 – Екран розрахунку калорій та органічних речовин

Трекер калорій надає можливість відстежувати розклад харчування та кількість спожитих калорій. З його допомогою користувач може обирати страви зі списку та переглядати загальну кількість калорій, спожитих протягом дня. На останньому етапі, користувач має можливість вибору розкладу тренувань, який автоматично генерується після вибору кількості днів. Цей розклад містить

рекомендації з вправ та активностей на кожен день, що допомагає користувачеві планувати свої фізичні навантаження і виконувати їх систематично.

4.3 Висновки

У четвертому розділі було проведено тестування програмних модулів мобільної системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу. Було перевірено реакцію програмних засобів на помилкові ситуації, валідацію полів введення інформації. Також було розроблено інструкцію користувача по встановленню та початковій експлуатації програмного продукту.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено мобільну системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу. Для розробки системи був використаний Android Studio. Було використано методичні вказівки до виконання бакалаврських дипломних робіт [24].

Під час реалізації застосунку був проведений аналіз сучасного стану питання, включаючи дослідження основних аналогів програмного рішення. Було виявлено недоліки цих аналогів та проведено порівняння з розробленим власним продуктом. На основі цього порівняння були сформульовані основні завдання для виконання бакалаврської дипломної роботи.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування Kotlin, а також технологій SQLite та Layout Editor.

Відповідно до поставлених задач було:

- проаналізовано існуючі мобільні додатків і систем для підтримки здорового способу життя та підвищення продуктивності;
- розроблено методи, модель та алгоритми роботи системи;
- розроблено методи трекінгу фізичної активності;
- створено системи контролю харчування з можливістю індивідуального планування дієти;
- інтегровано техніки тайм-менеджменту для оптимізації робочого часу та підвищення продуктивності;
- розроблено алгоритм персоналізації рекомендацій на основі аналізу зібраних даних;
- проведено тестування та оцінка ефективності розробленої мобільної системи в реальних умовах.

Було розроблено схеми загального алгоритму роботи мобільної системи та модуля трекера калорій. Також було розроблено модель системи з використанням UML діаграм.

Також було розроблено алгоритм адаптивного планування завдань на основі фізичної та робочої активності. Цей алгоритм дозволяє автоматично коригувати розклад завдань з урахуванням рівня втомленості та продуктивності користувача, що забезпечує оптимізацію часу виконання завдань та підтримання балансу між роботою та відпочинком.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Здоровий спосіб життя. Його вплив на здоров’я та працездатність людини [Електронний ресурс] Режим доступу до ресурсу: https://studies.in.ua/bjd-lapin/1151-zdoroviy-sposb-zhittya-yogo-vpliv-na-zdorovya-ta-pracezdatnst-lyudini.html (дата звернення: 04.04.2024)
2. Саєцький Д.Р. Розробка мобільної системи для підтримки здорового способу життя й підвищення продуктивності використання робочого часу / Г. О. Черноволик В. В. Войтко Л. М. Круподьорова А. В. Денисюк Д. Р. Саєцький // Матеріали Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії, Вінниця, 2024. [Електронний ресурс] Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20513/16971 (дата звернення 04.04.2024)
3. A systematic review of intention to use fitness apps [Електронний ресурс] Режим доступу до ресурсу: https://www.nature.com/articles/s41599-023-02011-3 (дата звернення: 07.04.2024)
4. Can health apps motivate people to exercise more? [Електронний ресурс] Режим доступу до ресурсу: https://globalsportmatters.com/health/2018/12/14/can-health-apps-motivate-people-to- exercise-more/ (дата звернення: 07.04.2024)
5. MyFitnessPal [Електронний ресурс] Режим доступу до ресурсу: https://www.myfitnesspal.com (дата звернення: 08.04.2024)
6. Strava [Електронний ресурс] Режим доступу до ресурсу: https://www.strava.com
7. Toggl[Електронний ресурс] Режим доступу до ресурсу: https://toggl.com
8. HeadSpace [Електронний ресурс] Режим доступу до ресурсу: https://www.headspace.com

9. Lifesum [Електронний ресурс] Режим доступу до ресурсу: https://lifesum.com
10. Impact Of Blockchain In Fitness Industry [Електронний ресурс] Режим доступу до ресурсу: https://www.code-brew.com/impact-of-blockchain-in-fitness-industry/
11. Діаграма діяльності [Електронний ресурс] Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Діаграма_діяльності
12. Філліпс, Б., Стюарт, К., Марсікано, К. Android Programming: The Big Nerd Ranch Guide. Атланта: Big Nerd Ranch, 2019. 624 с.
13. Develop a UI with Views [Електронний ресурс] Режим доступу до ресурсу: https://developer.android.com/studio/write/layout-editor
14. Android SDK[Електронний ресурс] Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Android_SDK
15. Скін, Дж., Ґрінгалх, Д. Kotlin Programming: The Big Nerd Ranch Guide. Атланта: Big Nerd Ranch, 2018. 480 с.
16. Android Studio Tutorial[Електронний ресурс] Режим доступу до ресурсу: https://www.geeksforgeeks.org/android-studio-tutorial/
17. Activity [Електронний ресурс] Режим доступу до ресурсу: https://developer.android.com/reference/android/app/Activity (дата звернення: 18.04.2024)
18. Recycler View[Електронний ресурс] Режим доступу до ресурсу: https://developer.android.com/reference/kotlin/androidx/recyclerview/widget/RecyclerView (дата звернення: 18.04.2024)
19. SQLite[Електронний ресурс] Режим доступу до ресурсу: https://developer.android.com/training/data-storage/sqlite (дата звернення: 18.04.2024)
20. Draw.io[Електронний ресурс] Режим доступу до ресурсу: https://www.drawio.com (дата звернення: 22.04.2024)

21. Calories to Cut[Електронний ресурс] Режим доступу до ресурсу: https://www.maxinutrition.com/sports/bodybuilding/Calories-to-Cut/ (дата звернення: 25.04.2024)
22. Джонсон, Джеф, Фінн, Кейт. Designing User Interfaces for an Aging Population: Towards Universal Design. Амстердам: Morgan Kaufmann, 2017. 304 с.
23. Тестування сумісності – що це таке, типи, процес, характеристики, інструменти та інше! [Електронний ресурс] Режим доступу до ресурсу: https://www.zaptest.com/uk/тестування-сумісності-що-це-таке-типи (дата звернення: 26.04.2024)
24. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

67

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
" 12 " березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка мобільної системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. Г,О. Черноволик
" 12 " березня 2024 р.

Виконав:
студент гр. 2ПІ-20Б Д.Р. Саєцький
" 12 " березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка мобільної системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу».

Галузь застосування – охорона здоров'я, особистісний тайм-менеджмент.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від 11 березнч 2024 р. ректора по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності використання робочого часу та підтримка здорового способу життя за рахунок впровадження мобільної системи, що інтегрує трекінг фізичної активності, контроль харчування та оптимізацію робочого графіку, що дозволяє покращити фізичний стан та активність користувача.

Призначення роботи – розробка мобільної системи для покращення здоров’я та тайм-менеджменту.

4 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Філліпс, Б., Стюарт, К., Марсікано, К. Android Programming: The Big Nerd Ranch Guide. Атланта: Big Nerd Ranch, 2019. 624 с.
2. Скін, Дж., Ґрінгалх, Д. Kotlin Programming: The Big Nerd Ranch Guide. Атланта: Big Nerd Ranch, 2018. 480 с.

3. Джонсон, Джеф, Фінн, Кейт. Designing User Interfaces for an Aging Population: Towards Universal Design. Амстердам: Morgan Kaufmann, 2017. 304 с.

5. Технічні вимоги

Основні технології розробки: мова програмування Kotlin, платформа розробки Android Studio. Вхідні дані: дані про фізичну активність (кількість кроків, витрачений час на тренування, пройдена відстань, типи активності); інформація про харчування (записи про спожиті продукти, калорійність, білки, жири, вуглеводи); дані про розклад та виконання робочих завдань (заплановані події, дедлайни, статус виконання завдань); опитування та фідбек користувачів. Системні вимоги: операційна система Android версії 6.0 (Marshmallow) або новіше; мінімальний об'єм оперативної пам'яті – 2 ГБ; необхідний простір для встановлення додатку – 100 МБ.

6. Конструктивні вимоги.

Конструкція мобільної системи повинна бути інтуїтивно зрозумілою та зручною для користувачів, відповідати сучасним вимогам дизайну та ергономіки. Система повинна бути легкою в налаштуванні та використанні, забезпечувати швидкий доступ до основних функцій.

Графічна та текстова документація повинна відповідати діючим стандартам України, бути чіткою, детальною та доступною для користувачів.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

– пояснювальна записка до БДР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз підходів до підвищення ефективності використання робочого часу та підтримки здорового способу життя за допомогою мобільних додатків.	14.03.24 – 22.03.24
2	Розробка методів моніторингу фізичної активності та управління робочими завданнями.	23.03.24 – 24.03.24
3	Розробка алгоритмів адаптивного планування завдань та надання рекомендацій.	25.03.24 – 22.04.24
4	Розробка мобільного програмного забезпечення	23.04.24 – 10.05.24
5	Тестування та валідація мобільного додатку.	11.05.24 – 24.05.24
6	Оформлення матеріалів до захисту БДР	25.05.24 – 30.05.24

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

71

Додаток Б – ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка мобільної системи для підтримки здорового способу життя й підвищення ефективності використання робочого часу»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Черноволик Г.О.

Оригінальність	96.3%
Схожість	3.7%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку: _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи: _____ Саєцький Д.Р.

Керівник роботи: _____ Черноволик Г.О.

Додаток В – Лістинг програми

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity
{
   TextView title , subtitle;
   Button start;
   ImageView img1,img2;
   View load_bar,load_bar2;
   Animation anim_img1,anim_title,btn_anim,load_progress,load_stop;

   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      anim_img1= AnimationUtils.loadAnimation(this,R.anim.animimage1);
      anim_title=AnimationUtils.loadAnimation(this,R.anim.title);
      btn_anim=AnimationUtils.loadAnimation(this, R.anim.btnanim);
      load_progress=AnimationUtils.loadAnimation(this, R.anim.progress);
```

```
    load_stop=AnimationUtils.loadAnimation(this,R.anim.progress_stop);

    title=(TextView) findViewById(R.id.title_txt);
    subtitle=(TextView) findViewById(R.id.subtitle_txt);
    start=(Button) findViewById(R.id.btn1);
    img1=(ImageView)findViewById(R.id.img1);
    img2=(ImageView)findViewById(R.id.img2);
    load_bar=(View)findViewById(R.id.load_bar);
    load_bar2=(View)findViewById(R.id.load_bar2);

    img1.startAnimation(anim_img1);
    img2.startAnimation(anim_img1);
    title.startAnimation(anim_title);
    subtitle.startAnimation(anim_title);
    start.startAnimation(btn_anim);
    load_bar.startAnimation(load_progress);
    load_bar2.startAnimation(load_stop);

    start.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            openUserPage();
        }
    });



}

public void openUserPage()
{
    Intent intent = new Intent(this, UserProfile.class);
    startActivity(intent);
}
```

```
}


import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.widget.Toast;

import com.google.android.material.navigation.NavigationView;

public class HomePage extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener
{

    Toolbar toolbar;
    NavigationView nav;
    ActionBarDrawerToggle toggle;
    DrawerLayout drawerLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_page);



        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
```

```
        nav = (NavigationView) findViewById(R.id.nav_view);
        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        toggle = new ActionBarDrawerToggle(this, drawerLayout, toolbar, R.string.Open,
R.string.Close);
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();

        nav.setNavigationItemSelectedListener(this);

    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();

        if (id==R.id.nav_timer)
        {
            Intent intent1 = new Intent(HomePage.this, Timer.class);
            startActivity(intent1);
        }
        else if (id==R.id.nav_profile)
        {
            Intent intent2 = new Intent(HomePage.this, UserProfile.class);
            startActivity(intent2);
        }

        else if(id== R.id.nav_calc) {
            Intent intent3 = new Intent(HomePage.this, Recommendation.class);
            startActivity(intent3);
        }

        else if (id==R.id.nav_tracker)
```

```
        {
            Intent intent4 = new Intent(HomePage.this, FoodCalories.class);
            startActivity(intent4);
        }


        else if(id==R.id.nav_splits) {
            Intent intent5 = new Intent(HomePage.this, Splits.class);
            startActivity(intent5);
        }
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        drawerLayout.closeDrawer(GravityCompat.START);
        return true;
    }
}




public class Profiles {
    String name;
    Integer age;
    Float height;
    Float weight;
    String gender;
    String email;

    public Profiles()
    {

    }

    public String getName() {
        return name;
```

```
}

public void setName(String name) {
   this.name = name;
}

public Integer getAge() {
   return age;
}

public void setAge(Integer age) {
   this.age = age;
}

public Float getHeight() {
   return height;
}

public void setHeight(Float height) {
   this.height = height;
}

public Float getWeight() {
   return weight;
}

public void setWeight(Float weight) {
   this.weight = weight;
}

public String getGender() {
   return gender;
}
```

```
    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.CountDownTimer;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Locale;

public class Timer extends AppCompatActivity {

    TextView timerValue;
    static final long START_TIME_IN_MILLIS = 3600000;
    CountDownTimer countDownTimer;
    boolean timerRunning;
    long timeLeftInMillis = START_TIME_IN_MILLIS;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_timer);

        timerValue = (TextView)findViewById(R.id.Timer);

        startTimer();
    }

    private void startTimer()
    {
        countDownTimer = new CountDownTimer(timeLeftInMillis, 1000) {
            @Override
            public void onTick(long millisUntilFinished) {
                timeLeftInMillis=millisUntilFinished;
                updateCountdownTimer();
            }

            @Override
            public void onFinish() {
                Toast.makeText(getApplicationContext(), "Workout Complete!",
Toast.LENGTH_SHORT).show();

            }
        }.start();
        timerRunning = true;
    }

    private void updateCountdownTimer() {
        int minutes = (int) (timeLeftInMillis/1000) / 60;
        int seconds = (int) (timeLeftInMillis/1000) % 60;
```

```
        String timeLeft = String.format(Locale.getDefault(),"%02d:%02d",minutes,seconds);
        timerValue.setText(timeLeft);
    }
}
```

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;

public class Splits extends AppCompatActivity
{
    RadioGroup radioGroup;
    RadioButton rButton1,rButton2,rButton3,rButton4;
    Button go;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splits);
        radioGroup = (RadioGroup) findViewById(R.id.radioGroup1);
        rButton1 = (RadioButton) findViewById(R.id.button1);
        rButton2 = (RadioButton) findViewById(R.id.button2);
        rButton3 = (RadioButton) findViewById(R.id.button3);
        rButton4 = (RadioButton) findViewById(R.id.button4);
```

```
        go = (Button) findViewById(R.id.go_btn);

        go.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (rButton1.isChecked()) {
                    Intent intent = new Intent(getApplicationContext(),ThreeDaySplit.class);
                    startActivity(intent);
                } else if (rButton2.isChecked()) {
                    Intent intent = new Intent(getApplicationContext(),FourDaySplit.class);
                    startActivity(intent);
                } else if (rButton3.isChecked()) {
                    Intent intent = new Intent(getApplicationContext(),FiveDaySplit.class);
                    startActivity(intent);
                } else if (rButton4.isChecked()) {
                    Intent intent = new Intent(getApplicationContext(),SixDaySplit.class);
                    startActivity(intent);
                }
            }
        });
    }
}
```

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
```

```
import android.widget.TextView;
import androidx.annotation.NonNull;
import org.jetbrains.annotations.NotNull;
import org.jetbrains.annotations.Nullable;
import java.net.CookieHandler;
import java.util.List;
public class food_adapter extends ArrayAdapter {
    private Context mCtx;
    private int resources;
    private List items;

    @NotNull
    public View getView(int position, @Nullable View convertView, @NotNull ViewGroup parent)
{
        LayoutInflater var = LayoutInflater.from(this.mCtx);
        LayoutInflater layoutInflater = var;
        View var10 = layoutInflater.inflate(this.resources, (ViewGroup)null);
        View view = var10;
        var10 = view.findViewById(R.id.food_image);
        ImageView imageView = (ImageView)var10;
        var10 = view.findViewById(R.id.foodNameText);
        TextView titleTextView = (TextView)var10;
        var10 = view.findViewById(R.id.foodDescription);
        TextView despTextView = (TextView)var10;
        model mItem = (model) this.items.get(position);
        imageView.setImageDrawable(this.mCtx.getResources().getDrawable(mItem.getImg()));
        titleTextView.setText((CharSequence)mItem.getTitle());
        despTextView.setText((CharSequence)mItem.getDescription());
        return view;
    }
    @NotNull
    public final Context getMCtx() {
        return this.mCtx;
    }
```

```
    public final void setMCtx(@NotNull Context var1) {
        this.mCtx = var1;
    }
    public final int getResources() {
        return this.resources;
    }
    public final void setResources(int var1) {
        this.resources = var1;
    }
    @NotNull
    public final List getItems() {
        return this.items;
    }
    public final void setItems(@NotNull List var1) {
        this.items = var1;
    }
    public food_adapter(@NotNull Context mCtx, int resources, @NotNull List items) {
        super(mCtx, resources, items);
        this.mCtx = mCtx;
        this.resources = resources;
        this.items = items;
    }
}
```

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.Button;
```

```
import android.widget.ListView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;


public class FoodCalories  extends AppCompatActivity {
   private HashMap _$_findCachedViewById;


   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.setContentView(R.layout.activity_food_calories);
      ListView listview = this.findViewById(R.id.listView);
      boolean var4 = false;
      List list = (List) (new ArrayList());
      TextView calorieText = this.findViewById(R.id.calorieText);
      Button reset = this.findViewById(R.id.reset);
      final TextView value = this.findViewById(R.id.valueCal);



      list.add(new model("Рис", "Простий рис: 200 ккал", android.R.drawable.alert_light_frame));
      list.add(new model("Чапаті", "2 Прості чапаті: 100 ккал", R.drawable.roti_canai));
      list.add(new model("Даль", "Проста даль: 50 ккал", R.drawable.dal2));
      list.add(new model("Салат", "Салат з міксом: 50 ккал", R.drawable.salad));
      list.add(new model("Локшина", "Локшина: 140 ккал", R.drawable.noodles));
```

```
        list.add(new model("Бургер", "Бургер: 200 ккал", R.drawable.hamburger));
        list.add(new model("Піца", "Піца: 250 ккал", R.drawable.pizza));
        list.add(new model("Холодний напій", "Газований холодний напій: 140 ккал",
R.drawable.soft_drink));
        list.add(new model("Яблуко", "Яблуко: 114 ккал", R.drawable.apple));
        list.add(new model("Хліб", "4 шматки хліба: 65 ккал", R.drawable.baguette));
        list.add(new model("Торт", "1 шматок торта: 132 ккал", R.drawable.cake));
        list.add(new model("Мафін", "Мафін: 47 ккал", R.drawable.cupcake));
        list.add(new model("Морква", "Морква: 30 ккал", R.drawable.carrot));
        list.add(new model("Курка", "Курка: 220 ккал", R.drawable.chicken_leg));
        list.add(new model("Шоколад", "Шоколадна плитка: 200 ккал", R.drawable.chocolate));
        list.add(new model("Пончик", "Пончик: 110 ккал", R.drawable.donut));
        list.add(new model("Ладду", "Ладду: 170 ккал", R.drawable.fazuelos));
        list.add(new model("Картопляні хрустки", "Картопляні хрустки: 155 ккал",
R.drawable.french_fries));
        list.add(new model("Апельсин", "Апельсин: 52 ккал", R.drawable.fruit));
        list.add(new model("Хот-дог", "Хот-дог: 95 ккал", R.drawable.hot_dog));
        list.add(new model("Роли", "Рол: 68 ккал", R.drawable.kebab));
        list.add(new model("Чіпси з картоплі", "Чіпси з картоплі: 155 ккал",
R.drawable.potato_chips));
        list.add(new model("Сандей", "Сандей: 110 ккал", R.drawable.sundae));
        list.add(new model("Морозиво", "Морозиво: 100 ккал", R.drawable.parfait));
        list.add(new model("Полуниця", "Полуниця: 110 ккал", R.drawable.strawberry));
        list.add(new model("М'ясо", "250 г червоного м'яса: 300 ккал", R.drawable.meat));
        list.add(new model("Кічді", "Кічді: 70 ккал", R.drawable.vegan_food));



        listview.setAdapter(new food_adapter(this, R.layout.listview_row, list));
        ((ListView)this.findViewById(R.id.listView)).setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            public final void onItemClick(@NotNull AdapterView parent, @NotNull View view, int
position, long id) {
                TextView var10000;
```

```
            TextView var10001;
            String var6;
            boolean var7;

            if (position == 0) {
               var10000 = value;
               var10001 = value;
               var6 = var10001.getText().toString();
               var7 = false;
               var10000.setText(String.valueOf(Integer.parseInt(var6) + 200));
            }
         }

      });
      reset.setOnClickListener(new OnClickListener() {
         public final void onClick(View it) {
            value.setText("0");
         }
      });
   }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView 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"
   tools:context=".FourDaySplit"
   android:background="@color/colorPrimary">
```

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/backGroundColor"
    tools:layout_editor_absoluteX="0dp"
    tools:layout_editor_absoluteY="0dp">

    <TextView
        android:id="@+id/textView13"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:fontFamily="@font/cloick"
        android:text="Чотириденний розклад тренувань"
        android:textAlignment="center"
        android:textColor="@color/textColor"
        android:textSize="30sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <TextView
        android:id="@+id/textView14"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="34dp"
        android:fontFamily="@font/cloick"
        android:text="День 1:"
        android:textColor="@color/textColor"
        android:textSize="20dp"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.216"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="36dp"
        android:fontFamily="@font/cloick"
        android:text="Груди/Тріцепс"
        android:textColor="@color/textColor"
        android:textSize="18dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.484"
        app:layout_constraintStart_toEndOf="@+id/textView14"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />

    <TextView
        android:layout_width="249dp"
        android:layout_height="156dp"
        android:layout_marginTop="70dp"
        android:fontFamily="@font/cloick"
        android:text=" Жим гантелей (3x16)\n Розведення гантелей (3x12)\n Жим на нахилі
вниз (3x18)\n Розгинання трицепсів (3x12)\n Розтягування черепа (3x12)\n Розтягування над
головою (3x18)\n"
        android:textAlignment="center"
        android:textColor="@color/textColor"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />

    <TextView
        android:id="@+id/textView10"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="248dp"
        android:fontFamily="@font/cloick"
        android:text="День 2:"
        android:textColor="@color/textColor"
        android:textSize="20dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.209"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="250dp"
        android:fontFamily="@font/cloick"
        android:text="Спина/Біцепс"
        android:textColor="@color/textColor"
        android:textSize="18dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.351"
        app:layout_constraintStart_toEndOf="@+id/textView10"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />


    <TextView
        android:layout_width="294dp"
        android:layout_height="141dp"
        android:layout_marginTop="304dp"
        android:fontFamily="@font/cloick"
        android:text=" Згинання в руках (3x16)\n Згинання на нахилі (3x12)\n Згинання на
нахилі вгору (3x18)\n Згинання молотком (3x12)\n Виставляння на звуженій гріптасі (3x12)\n
Розтягування лат (3x18)\n"
```

```
    android:textAlignment="center"
    android:textColor="@color/textColor"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView13" />

<TextView
    android:id="@+id/textView11"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="480dp"
    android:fontFamily="@font/cloick"
    android:text="День 3:"
    android:textColor="@color/textColor"
    android:textSize="20dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.209"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView13" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="482dp"
    android:fontFamily="@font/cloick"
    android:text="Ноги"
    android:textColor="@color/textColor"
    android:textSize="18dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.48"
    app:layout_constraintStart_toEndOf="@+id/textView10"
    app:layout_constraintTop_toBottomOf="@+id/textView13" />

<TextView
```

```
        android:layout_width="261dp"
        android:layout_height="95dp"
        android:layout_marginTop="528dp"
        android:fontFamily="@font/cloick"
        android:text="Присідання з вагою (3x16)\n Зворотні випади (3x12)\n Бічні випади
(3x18)\n Сумо-присідання (3x12)\n"
        android:textAlignment="center"
        android:textColor="@color/textColor"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.46"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />


    <TextView
        android:id="@+id/textView17"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="640dp"
        android:fontFamily="@font/cloick"
        android:text="День 4:"
        android:textColor="@color/textColor"
        android:textSize="20dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.179"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="645dp"
        android:fontFamily="@font/cloick"
        android:text="Прес/Плечі"
```

```
        android:textColor="@color/textColor"
        android:textSize="18dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.348"
        app:layout_constraintStart_toEndOf="@+id/textView10"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />

    <TextView
        android:layout_width="261dp"
        android:layout_height="95dp"
        android:layout_marginTop="680dp"
        android:fontFamily="@font/cloick"
        android:text="Піднімання ніг у коліні (3x16)\n Бокові скручування (3x12)\n Прес
Арнольда (3x18)\n Військовий прес (3x12)\n"
        android:textAlignment="center"
        android:textColor="@color/textColor"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.46"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView13" />


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

```
import org.jetbrains.annotations.NotNull;
public final class model {
  @NotNull
  private final String title;
```

```
    @NotNull
    private final String description;
    private final int img;
    @NotNull
    public final String getTitle() {
        return this.title;
    }
    @NotNull
    public final String getDescription() {
        return this.description;
    }
    public final int getImg() {
        return this.img;
    }
    public model(@NotNull String title, @NotNull String description, int img) {
        super();
        this.title = title;
        this.description = description;
        this.img = img;
    }
}


import android.os.Bundle;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.CalendarView;
import android.view.ViewGroup.LayoutParams;
import android.view.View;
import android.graphics.Color;
import android.view.Gravity;
import android.graphics.Typeface;
```

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.res.ResourcesCompat;


import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

public class ScheduleActivity extends AppCompatActivity {

    private ActivityScheduleBinding binding;
    private TableLayout tableLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityScheduleBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        tableLayout = binding.tableLayout;

        binding.calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener()
{
            @Override
            public void onSelectedDayChange(CalendarView view, int year, int month, int
dayOfMonth) {
                Calendar selectedDate = Calendar.getInstance();
                selectedDate.set(year, month, dayOfMonth);

                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
```

```
            String formattedDate = sdf.format(selectedDate.getTime());

            Toast.makeText(ScheduleActivity.this, "Selected Date: " + formattedDate,
Toast.LENGTH_SHORT).show();
            populateScheduleForDate(formattedDate);
        }
    });

    // Initialize with today's date
    Calendar today = Calendar.getInstance();
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
    String formattedDate = sdf.format(today.getTime());
    populateScheduleForDate(formattedDate);
}

private void populateScheduleForDate(String date) {
    tableLayout.removeAllViews(); // Clear previous rows

    Map<Integer, String> events = generateEventsForDate(date);

    for (int i = 0; i < 24; i++) {
        if (i > 0) {
            // Add a spacer view between rows
            View spacer = new View(this);
            TableLayout.LayoutParams spacerParams = new TableLayout.LayoutParams(
                    LayoutParams.MATCH_PARENT,
                    16 // Height of the spacer
            );
            spacer.setLayoutParams(spacerParams);
            tableLayout.addView(spacer);
        }

        TableRow tableRow = new TableRow(this);
        TableRow.LayoutParams params = new TableRow.LayoutParams(
```

```
                LayoutParams.MATCH_PARENT,
                LayoutParams.WRAP_CONTENT
            );
            tableRow.setLayoutParams(params);
            tableRow.setBackgroundResource(R.drawable.rounded_border);

            TextView hourTextView = new TextView(this);
            hourTextView.setLayoutParams(new TableRow.LayoutParams(0, 200, 1f)); // Set height
            hourTextView.setText(String.format(Locale.getDefault(), "%02d:00 -\n%02d:00", i, i + 1));
            hourTextView.setGravity(Gravity.CENTER);
            hourTextView.setTextColor(Color.BLACK);
            hourTextView.setTypeface(ResourcesCompat.getFont(this, R.font.cloick),
Typeface.BOLD); // Apply custom font
            hourTextView.setTextSize(18); // Set font size
            hourTextView.setPadding(8, 8, 8, 8);

            TextView scheduleTextView = new TextView(this);
            scheduleTextView.setLayoutParams(new TableRow.LayoutParams(0,
LayoutParams.WRAP_CONTENT, 3f));
            scheduleTextView.setText(events.getOrDefault(i, "Free Time"));
            scheduleTextView.setGravity(Gravity.CENTER_VERTICAL);
            scheduleTextView.setTextColor(Color.BLUE); // Set color
            scheduleTextView.setTypeface(ResourcesCompat.getFont(this, R.font.cloick)); // Apply
custom font
            scheduleTextView.setTextSize(16); // Set font size
            scheduleTextView.setPadding(8, 8, 8, 8);

            tableRow.addView(hourTextView);
            tableRow.addView(scheduleTextView);

            tableLayout.addView(tableRow);
        }
    }
```

```
private Map<Integer, String> generateEventsForDate(String date) {
    Map<Integer, String> events = new HashMap<>();

    // Define durations for different activities
    int sleepDuration = 8; // hours
    int workDuration = 8; // hours
    int breakDuration = 1; // hour
    int workoutDuration = 1; // hour
    int mealDuration = 1; // hour
    int leisureDuration = 2; // hours

    // Fill in the schedule dynamically
    int hour = 0;

    // Sleep
    for (int i = 0; i < sleepDuration; i++) {
        events.put(hour++, "Сон");
    }

    // Morning workout
    events.put(hour++, "Фізична активність");

    events.put(hour++, "Сніданок");

    for (int i = 0; i < workDuration / 2; i++) {
        events.put(hour++, "Робота");
    }

    events.put(hour++, "Обід");

    for (int i = 0; i < workDuration / 2; i++) {
        events.put(hour++, "Робота");
    }
```

```
        events.put(hour++, "Відпочинок");

        events.put(hour++, "Вечерня");

        for (int i = 0; i < leisureDuration; i++) {
            events.put(hour++, "Відпочинок");
        }

        while (hour < 24) {
            events.put(hour++, "Вільний час");
        }

        return events;
    }
}
```

Додаток Г – Ілюстративний матеріал

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МОБІЛЬНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ Й ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ РОБОЧОГО ЧАСУ

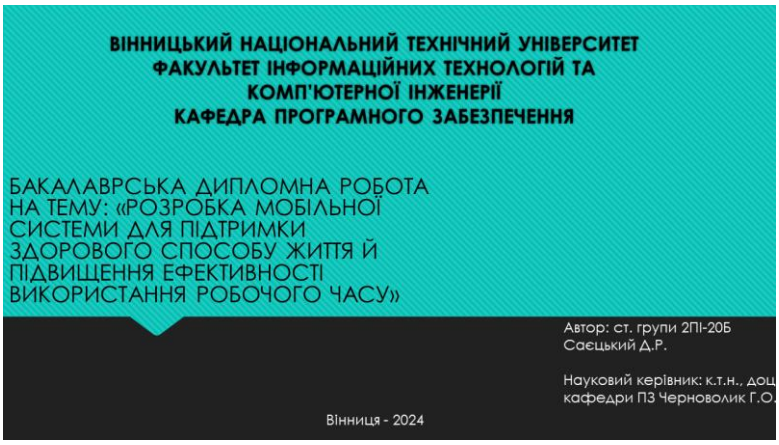

Рисунок Г.1 – Титульний слайд

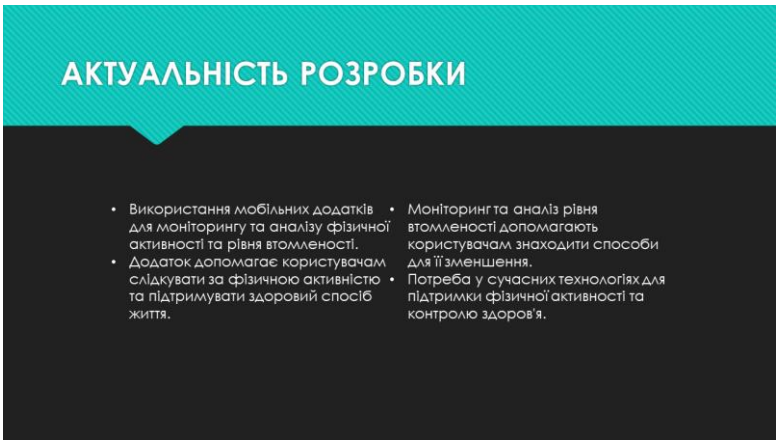

Рисунок Г.2 – Слайд «Актуальність розробки»

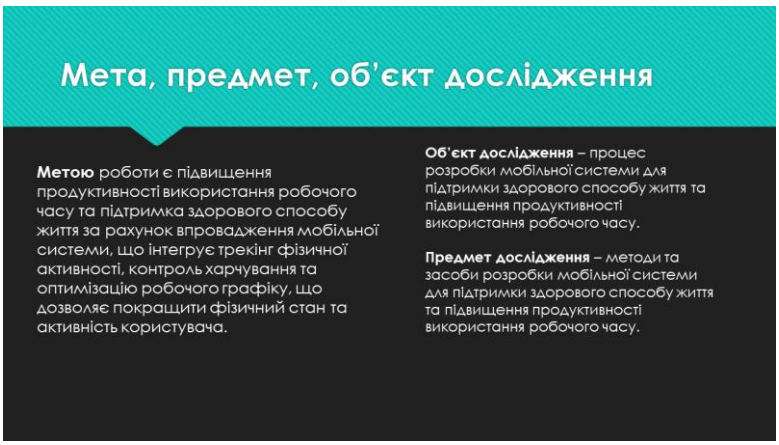

Рисунок Г.3 – Слайд «Мета, предмет та об’єкт дослідження»

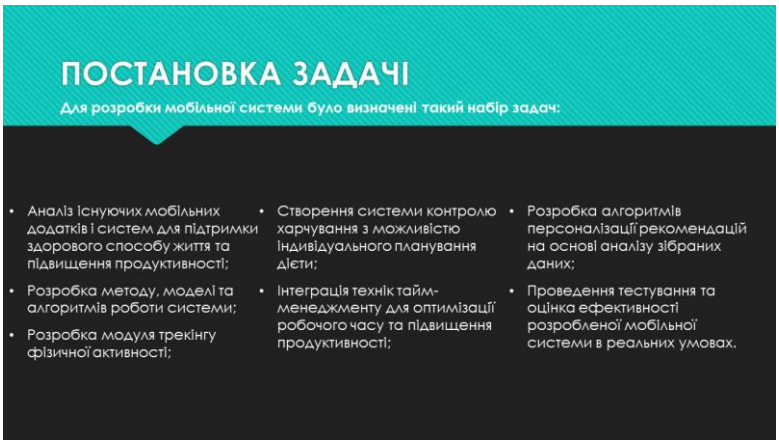

Рисунок Г.4 – Слайд «Постановка задачі»

Рисунок Г.5 – Слайд «Новизна»

Рисунок Г.6 – Слайд «Практична цінність отриманих результатів»

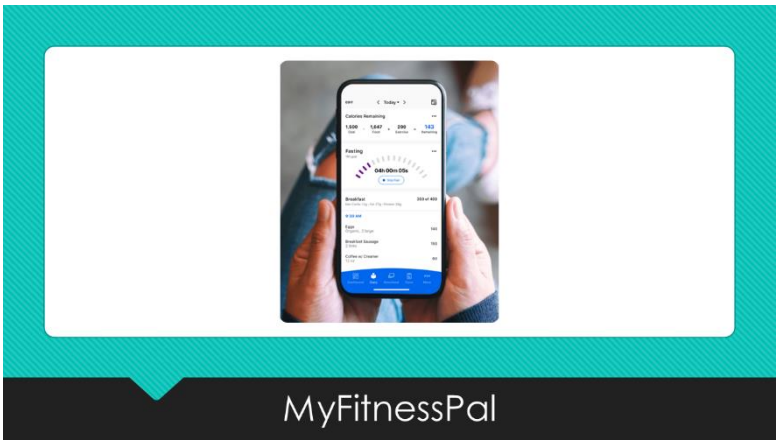

Рисунок Г.7 – Слайд «Аналог MyFitnessPal»

Рисунок Г.8 – Слайд «Аналог Strava»

Рисунок Г.9 – Слайд «Аналог Toggl»

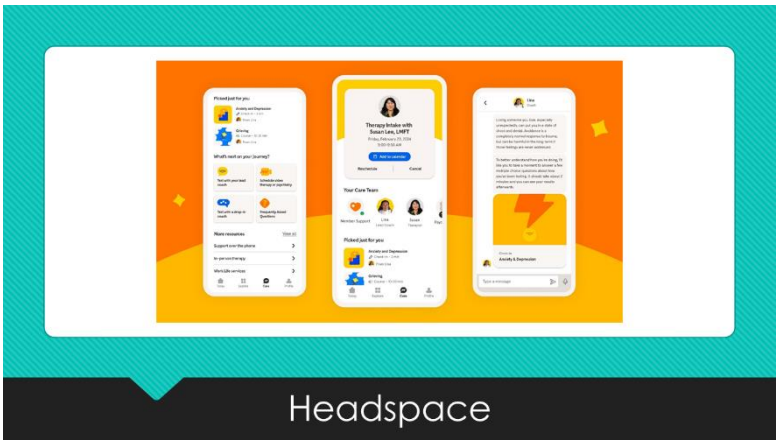

Рисунок Г.10 – Слайд «Аналог Headspace»

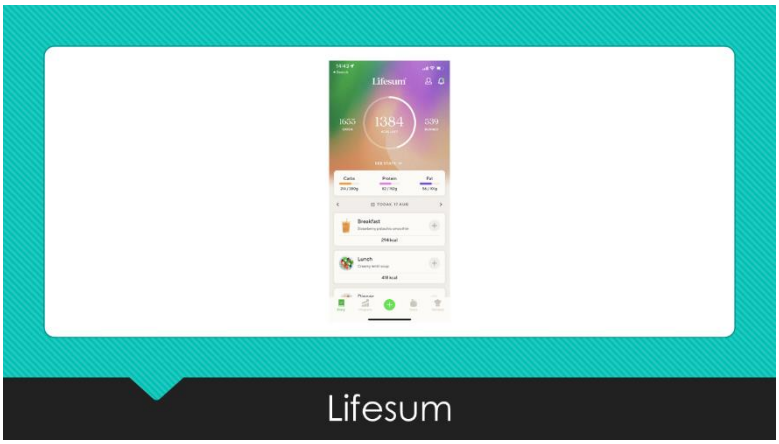

Рисунок Г.11 – Слайд «Аналог Lifesum»

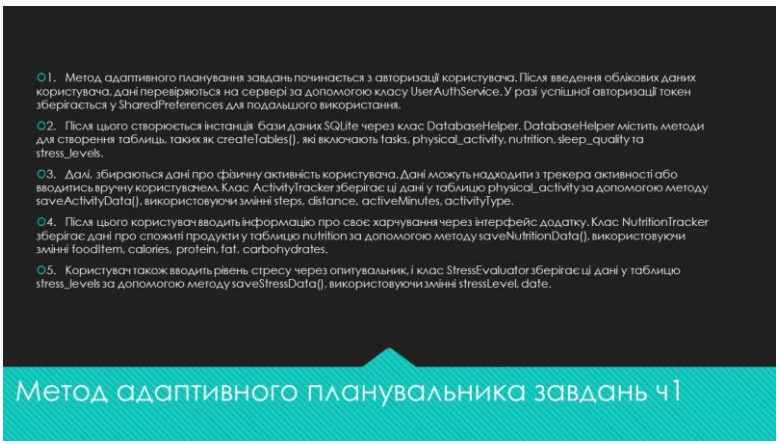

Рисунок Г.12 – Слайд «Метод адаптивного планувальника завдань (ч.1)»

Рисунок Г.13 – Слайд «Метод адаптивного планувальника завдань (ч.2)»

Рисунок Г.14 – Слайд «Блок-схема методу планувальника розкладу завдань»

Рисунок Г.15 – Слайд «Модель персоналізованої системи рекомендації у вигляді діаграми діяльності»

Рисунок Г.16 – Слайд «Алгоритм роботи модуля трекера калорій»

Рисунок Г.17 – Слайд «Діаграма діяльності мобільної системи обрахунків кількості калорій для споживання»

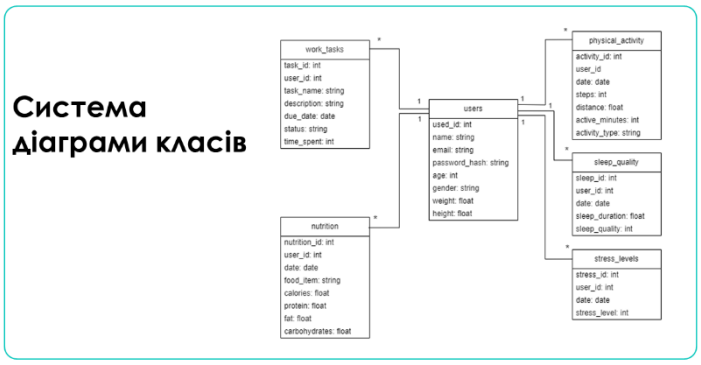

Рисунок Г.18 – Слайд «Система діаграми класів»

Рисунок Г.19 – Слайд «Блок-схема загального алгоритму роботи програмного застосунку»

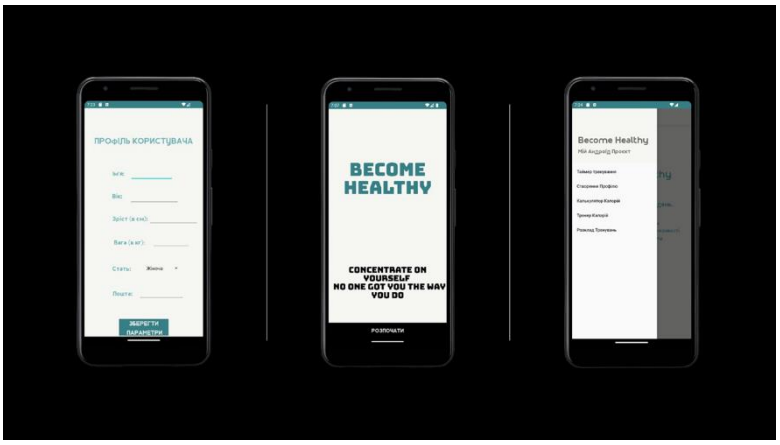

Рисунок Г.20 – Слайд «Функціонал реєстрації, початкового екрана, меню»

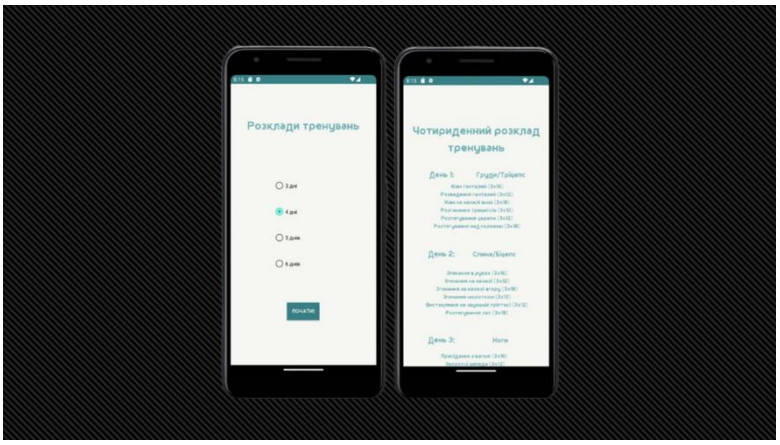

Рисунок Г.21 – Слайд «Функціонал розкладу тренувань»

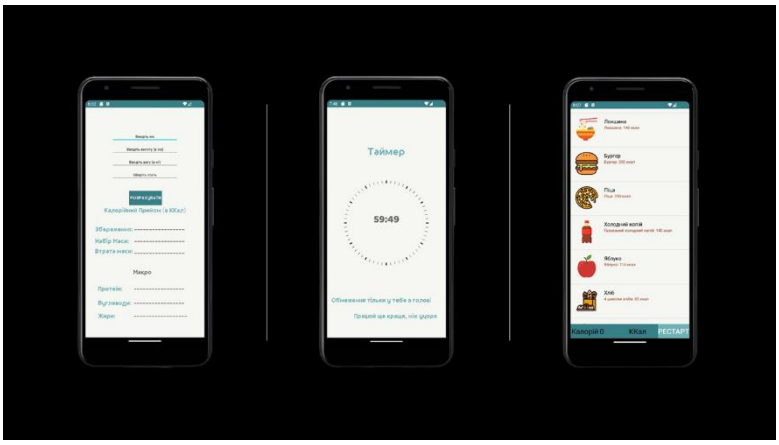

Рисунок Г.22 – Слайд «Функціонал обрахунків калорій, таймеру для фізичної активності»

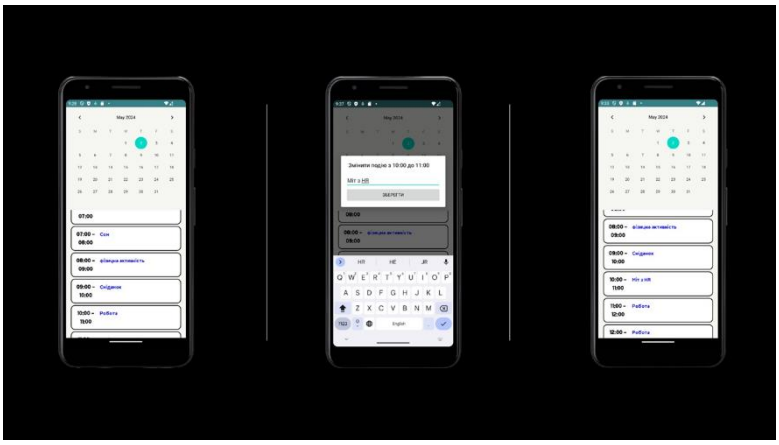

Рисунок Г.23 – Слайд «Функціонал розкладу завдань»

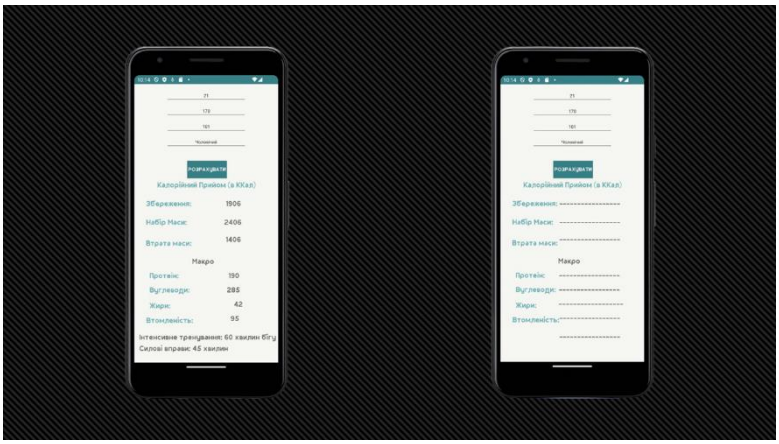

Рисунок Г.24 – Слайд «Функціонал рекомендацій»

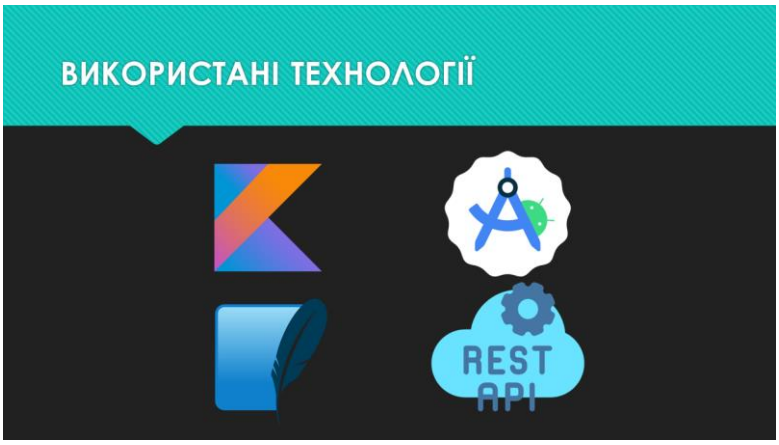

Рисунок Г.25 – Слайд «Використані технології»

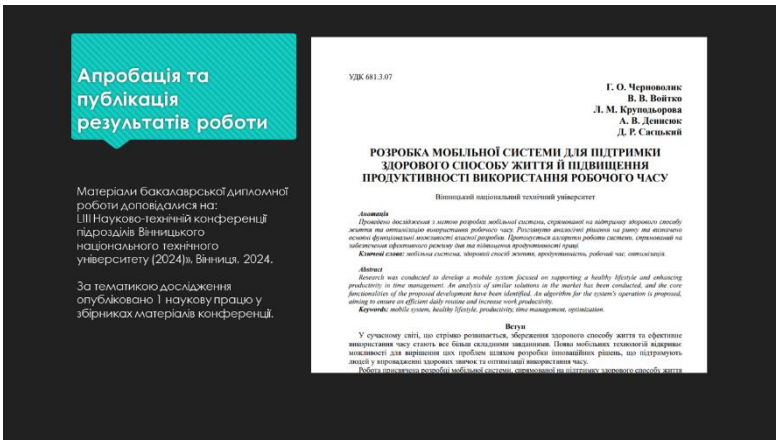

Рисунок Г.26 – Слайд «Апробація та публікація результатів роботи»

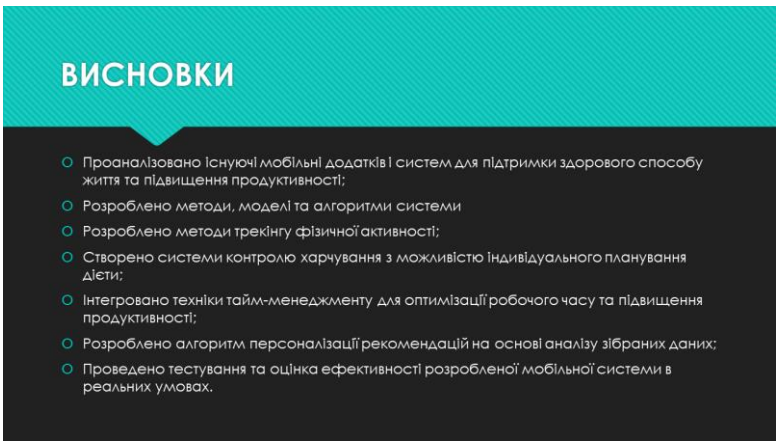

Рисунок Г.27 – Слайд «Висновки»