

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ НА ОСНОВІ ЗМІСТУ ДЛЯ
ВДОСКОНАЛЕННЯ РІВНЯ ЖИТТЯ ЛЮДЕЙ ПОХИЛОГО ВІКУ
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE CONTEXT-
AWARE RECOMMENDATION SYSTEM TO IMPROVE THE LIVING
STANDARD OF OLDER PEOPLE

Виконав: студент II курсу, групи КНТ-119м

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Гулько Е. В.

(прізвище та ініціали)

Керівник Гладкова О.М.

(прізвище та ініціали)

Рецензент Морщавка С. В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет Інститут інформатики та радіоелектроніки, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Д.Т.Н., професор
С. О. Субботін
« » 20 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гулько Ернесту Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація рекомендаційної системи на основі змісту для вдосконалення рівня життя людей похилого віку. Research and Software Implementation of the Context-Aware Recommendation System to Improve the Living Standard of Older People

керівник проєкту (роботи) Гладкова Ольга Миколаївна, к.т.н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «03» листопада 2020 року №301

2. Строк подання студентом проєкту (роботи) 11 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1 Аналіз предметної області, 2 Аналіз та вибір інструментів для розробки застосунку, 3 Розробка архітектури програми, 4 Керівництво програміста, 5 Керівництво оператора, 6 Економіко-організаційна частина, 7 Охорона праці та безпека у надзвичайних ситуаціях, Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Комплект слайдів

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	Гладкова О.М., доцент		
6	Пожуєва Т.О., професор		
7	Коробко О. В., ст. викладач		
Нормоконтролер	Камінська Ж. К., асистент		

7. Дата видачі завдання «04» вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1.	Визначення основної теми магістерської роботи	1 тиждень	
2.	Аналіз предметної області	1 тиждень	Розділи 1,2 ПЗ
3.	Розробка та затвердження технічного завдання	2 тиждень	Розділ 3 ПЗ
4.	Проектування структури системи	2 тиждень	Розділ 3 ПЗ
5.	Розробка схеми функціонування системи	3-4 тиждень	Розділ 4 ПЗ
6.	Створення програмного коду системи	4-5 тиждень	Розділ 4,5 ПЗ
7.	Розробка розділів пояснювальної записки	5-6 тиждень	Розділи 6,7 та Додатки до ПЗ
8.	Розробка слайдів презентації	6-7 тиждень	
9.	Захист кваліфікаційної роботи магістра	8 тиждень	

Студент

(підпис)

Гулько Е. В.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Гладкова О. М.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
120 с., 51 рис., 10 табл., 14 формул, 2 додатка, 34 джерела.

РЕКОМЕНДАЦІЙНИЙ АЛГОРИТМ, ЕЛЕКТРОННЕ ЗДОРОВ'Я, ANDROID ЗАСТОСУНОК, ІОТ, СИСТЕМА ДОПОМОГИ ЛЮДЯМ ПОХИЛОГО ВІКУ

Об'єкт дослідження – системи покращення рівню життя людей похилого віку.

Предмет дослідження – система надання рекомендацій на основі контекстного методу людям похилого віку або осіб які того потребують.

Мета роботи – розробка алгоритму та програмного продукту, що реалізує систему надання рекомендацій користувачам.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- дослідити потреби користувачів сегменту охорони здоров'я;
- проаналізувати існуючі системи покращення життя людей похилого віку;
- розробити мобільний застосунок для надання рекомендацій людям похилого віку.

В результаті розроблено та програмно реалізовано систему надання рекомендацій на основі контекстного методу людям похилого віку або для осіб, які того потребують.

Розроблена програма отримує від користувача для роботи програми дані під час реєстрації та оброблюючи дані надає рекомендації щодо запобігання вживання заборонених ліків або їжі, надання повідомлень щодо ліків або їжі, що рекомендовано регулярно вживати, надання інформації щодо стану зайвої ваги та виводить усю актуальну інформацію на дисплей.

ABSTRACT

Master thesis: 120 pages, 51 figures, 10 tables, 14 formulas, 2 appendixes, 34 sources.

RECOMMENDATION ALGORITHM, E-HEALTH, ANDROID APPLICATION, IOT, ELDERLY CARE PLATFORM

Object of study are systems for improving the living standards of the elderly.

Subject of study is a system of providing recommendations based on the contextual method for the elderly or those who need it.

The purpose of the work is development of an algorithm and a software product that implements a system for providing recommendations to users.

In order to achieve the goal, the following tasks must be performed:

- explore the needs of modern users of the healthcare segment;
- analyze existing systems for improving the lives of the elderly;
- develop mobile application that can solve the problem.

As a result, the system of providing recommendations based on the contextual method for the elderly or people who need it has been researched, developed and programmatically implemented.

The developed program receives from the user useful data for the operation of the program during user registration process and data processing provides recommendations on how to prevent the use of illicit drugs or food, provide reports on drugs or food that are recommended to use regularly, provide information on overweight and displays all relevant information on display.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ	9
ВСТУП.....	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	12
1.1 Аналіз поточного стану Інтернет речей	12
1.2 Огляд існуючих програмних засобів для вирішення завдання	15
1.2.1 Опис аналогу «Health Pal»	15
1.2.2 Опис аналогу «Remente».....	16
1.2.3 Опис аналогу «Health Nutrition & Fitness»	16
1.2.4 Опис аналогу «Eufylife».....	17
1.2.5 Результати аналізу та порівняння існуючих аналогів.....	17
1.3 Постановка завдання проблеми та її вирішення.....	18
1.3.1 Системні вимоги	19
1.3.2 Планування проєкту	20
1.3.3 Аналіз системи.....	22
1.4 Висновки до розділу 1	22
2 АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТІВ І ЗАСОБІВ ДЛЯ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ.....	23
2.1 Огляд особливостей платформи.....	23
2.2 Огляд особливостей середовища розробки.....	24
2.3 Огляд особливостей мови програмування	25
2.4 Огляд особливостей рекомендаційних методів	26
2.5 Висновки за розділом	27
3 ДИЗАЙН ТА РЕАЛІЗАЦІЯ СИСТЕМИ.....	28
3.1 Використані бібліотеки	28
3.2 Застосунок для Android смартфона	30
3.3 Програмне забезпечення для другорядного пристрою	31
3.4 Виконання системи.....	32

3.5 Апаратні компоненти	35
3.5.1 Допоміжний пристрій.....	36
3.5.2 Основний пристрій	36
3.6 Функціонування мобільного застосунку	37
3.7 Система мультиаккаунтності.....	41
3.8 Реєстрація	44
3.9 Головні функції системи	51
3.9.1 Нагадування про ліки	52
3.9.2 Система рекомендацій.....	53
3.10 Дизайн бази даних	55
3.11 Майбутня робота.....	56
3.11.1 Облікові записи та реєстрація	57
3.11.2 Екстрене повідомлення	57
3.11.3 Рекомендації на основі місцезнаходження	58
3.11.4 Система досягнень.....	58
3.11.5 Гео-зони	58
3.12 Висновки до розділу 3	59
4 КЕРІВНИЦТВО ПРОГРАМІСТА	60
4.1 Призначення та умови використання мобільного застосунку	60
4.2 Характеристики мобільного застосунку.....	60
4.3 Звертання до мобільного застосунку	61
4.4 Вхідні та вихідні дані	61
4.5 Повідомлення та обробка помилкових ситуацій	62
4.6 Висновки до розділу 4	63
5 КЕРІВНИЦТВО ОПЕРАТОРА.....	64
5.1 Призначення мобільного застосунку	64
5.2 Умови використання мобільного застосунку	64
5.3 Виконання мобільного застосунку.....	64
5.4 Приклад роботи з мобільним застосунком	65

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА	69
6.1 Концепція економічного обґрунтування	69
6.2 Планування розробки програмного продукту	70
6.3 Розрахунок основної заробітної плати	72
6.4 Розрахунок додаткової заробітної плати	73
6.5 Єдиний соціальний внесок.....	73
6.6 Визначення витрат на матеріали	73
6.7 Витрати на спеціальне обладнання	74
6.8 Інші прямі витрати.....	78
6.9 Розрахунок накладних витрат	79
7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ	80
7.1 Аналіз потенційних небезпек	80
7.2 Заходи із забезпечення безпеки.....	81
7.3 Заходи з забезпечення виробничої санітарії та гігієни праці	84
7.4 Заходи безпеки у надзвичайних ситуаціях.....	89
7.4.1 Заходи з пожежної безпеки.....	89
7.4.2 Єдина державна система цивільного захисту, її складові та режими функціонування.....	93
ВИСНОВКИ	97
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ	98
ДОДАТОК А	103
ДОДАТОК Б.....	111

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

IoT	– Internet of Things;
UML	– уніфікована мова моделювання;
ІКТ	– інформаційно-комунікаційні технології;
ОС	– операційна система;
ПЗ	– програмне забезпечення.

ВСТУП

Старіння населення - це внутрішній процес демографічного переходу, який представляє зміну високої смертності та народжуваності на низькому та контрольованому рівнях. Зниження народжуваності, що супроводжується поступовим збільшенням тривалості життя людей, мають прямий вплив на віковий склад населення, зменшуючи кількість людей молодшого віку та поширюючи верстви похилого віку [1]. З огляду на такий ріст населення похилого віку, соціальні системи та системи охорони здоров'я породжують більш вимогливі виклики для задоволення потреб цього сегменту населення [2]. Зіткнувшись із цими проблемами, міста та пристрої повинні надавати послуги, які можуть динамічно пристосовуватися до потреб їх жителів та робити речі більш комфортними та дружніми для людей похилого віку [3].

Через значний прогрес у розвитку електронних пристроїв та швидке розгортання комунікаційних мереж змінюється спосіб взаємодії у містах [4], таким чином, досягаючи визначення розумних міст. Концепція розумних міст почала розвиватися в минулому столітті як міська модель, заснована на технології, яка дозволить нам зіткнутися з великими проблемами, які почали хвилювати міста нашої планети: покращити енергетичну ефективність, зменшити забруднюючі викиди та спробувати змінити кліматичні зміни [5]. Сьогодні термін «розумні міста» є новим виразом, головна мета якого - покращити якість життя своїх громадян. Таким чином, розумні міста можна визначити як ті міста, які використовують ІКТ, щоб зробити свою критичну інфраструктуру, її компоненти та державні послуги більш інтерактивними, ефективними та місцем, де її мешканці знають про все це переваги [6].

Завдяки зближенню телематичних послуг та великій кількості програм для мобільних пристроїв, переваги, пропоновані Smart Cities у соціальній галузі, різноманітні. У цих містах часто спостерігають, як люди, незалежно від їх віку, використовують смартфони, щоб бути в курсі того, що відбувається навколо них в режимі реального часу, насолоджуючись інтерактивними

послугами та збагачуючи свій досвід під час різних видів діяльності (екскурсії музеями, парками, церкви, торгові центри тощо). Концепція розумних міст стосується нової технологічної тенденції, яка швидко зростає в галузі сучасних бездротових комунікацій, метою якої є об'єднання об'єктів, які повсюдно пов'язані з мережею та Інтернетом, створюючи цифрове взаємозв'язок [7]. У цьому Інтернеті майбутнього передбачається не лише зв'язок між все більшою кількістю людей, а й цифровий підхід, в якому в ідеалі все може бути пов'язане. Від технологічного пристрою до фізичних об'єктів, які зазвичай не мали цього зв'язку як пристрої, і взагалі всього, чим можна керувати чи маніпулювати [8]. Використання IoT у розумних містах може призвести до позитивних змін у житті людини. Завдяки стрімкому зростанню та популярності розумних медичних пристроїв, портативності біомедичних датчиків та використанню хмарних обчислень для аналізу великих обсягів даних, це різко змінює надання медичних послуг.

Ідея полягає в тому, щоб пов'язати всі доступні послуги розумного міста, eHealth та розумні рекомендації зі світом мобільних пристроїв та технологій IoT, щоб мати можливість надати користувачеві мобільний застосунок, який використовує всі привілеї IoT для надання рекомендацій та для автоматизований моніторинг таких речей, як харчування, ліки, розваги тощо. Основна мета цієї платформи - сприяти фізичному та соціальному обслуговуванню людей похилого віку, надаючи розумні рекомендації та нагадування на основі історії хвороби, місцезнаходження, віку, статі, ваги, та поточні захворювання користувача. Система також може надати користувачам можливість відстежувати деякі показники здоров'я, такі як частота серцебиття, рівень цукру, насичення крові киснем.

Метою цієї платформи є комплексне забезпечення набору розумних послуг, що покращують якість життя та соціальну інтеграцію людей похилого віку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз поточного стану Інтернет речей

У світі не існує загальноприйнятого визначення Інтернету речей. Насправді існує багато різних груп людей, які визначили цей термін, хоча його початкове використання приписується експерту з цифрових інновацій на ім'я Кевін Ештон. У всіх визначеннях ми отримуємо загальну думку, що перша версія Інтернету стосувалася даних, створених людьми, тоді як наступна версія стосувалась даних, створених речами, тому її називають Інтернетом речей.

Існує багато визначень для Інтернету речей. Нижче наводяться деякі визначення: IoT, як правило, визначали як «динамічну глобальну мережеву інфраструктуру з можливостями самоконфігурування на основі стандартів та сумісних протоколів зв'язку; фізичні та віртуальні "речі" в IoT мають ідентичності та атрибути і здатні використовувати інтелектуальні інтерфейси та бути інтегрованими як інформаційна мережа".

Мета IoT - підвищити функції першої версії Інтернету та зробити її більш корисною. За допомогою IoT користувачі можуть обмінюватися як інформацією, наданою людьми, що міститься в базах даних, так і інформацією, наданою речами у фізичному світі. IoT можна описати як зв'язок фізичних речей з Інтернетом та між собою для різних корисних цілей за допомогою різних інтелектуальних технологій, створюючи розумну екосистему всепроникаючих обчислень. Це також можна описати як включення вбудованого інтелекту в окремі об'єкти, які можуть помітити зміни у своєму фізичному стані.

Загальним визначенням IoT є те, що комп'ютери, датчики та об'єкти взаємодіють між собою та обробляють дані, тому ми можемо стверджувати, що IoT - це нова технологічна система, поєднана з низкою інформаційних технологій.

Інтернет речей поєднує різні технології в напівавтономну мережу. Він підключає окремі пристрої до мережі та між собою. У мережі також є системи контролерів (програмне забезпечення та послуги), які виконують функції головного мозку системи для обробки даних шляхом аналізу та використання даних, зібраних підключеними пристроями, для прийняття рішень та ініціювання дій з тих самих чи інших пристроїв.

Основною метою IoT є надання людям можливості однозначно ідентифікувати, вказувати, отримувати доступ і контролювати речі в будь-який час і в будь-якому місці за допомогою Інтернету. Взаємозв'язані мережі пристроїв можуть призвести до великої кількості інтелектуальних та автономних додатків та послуг, що приносить значні особисті, професійні та економічні вигоди.

Розумні середовища спрямовані на використання багатих комбінацій невеликих обчислювальних вузлів для ідентифікації та надання персоналізованих послуг користувачеві під час їх взаємодії та обміну інформацією з навколишнім середовищем. Технологію IoT можна застосовувати для створення розумних будинків, щоб забезпечити інтелект, комфорт та покращити якість нашого життя.

Інтегроване середовище розробки. Для розробки мобільного застосунку для догляду за старшими було обрано інтегроване середовище розробки "Android Studio".

Це середовище розробки базувалося на середовищі розробки IntelliJ IDEA, яке досі доступне як окремий програмний продукт. Основними перевагами Android Studio є безкоштовна форма доступу та статус офіційного середовища для розробки Android-застосунків.

Android Studio постійно оновлюється, додаючи функціональність для полегшення процесу розробки. На сьогодні у програмному середовищі доступні такі функції [9]:

- вдосконалений редактор макету. Він працює з технологією "ти отримуєш те, що бачиш", і дозволяє розробнику працювати з елементами користувацького інтерфейсу способом "перетягування";

- складання застосунків за допомогою системи "Gradle". Він побудований на принципах Apache Ant та Apache Maven, але замість використання XML-подібної форми презентації конфігурації проекту "Gradle" використовує мови Kotlin та Groovy;

- різні типи збірок програм та генерація декількох файлів .apk;

- засоби рефакторингу програмного коду;

- аналізатор статистичного коду, який дозволяє розробникам знаходити проблеми з продуктивністю, несумісність версій тощо;

- вбудований застосунок "ProGuard", призначений для оптимізації та затемнення програмного коду. Існує також утиліта для електронного підпису застосунку;

- підтримка розробки застосунків для ОС Wear віртуальний пристрій Android (емулятор).

Більшість існуючих на ринку застосунків не містять послуг гео-зонування, а також моніторингу стану здоров'я. Наприклад, мобільна система охорони здоров'я за замовчуванням, яка зазвичай попередньо інстальована на смартфонах, не включає послугу відстеження місцезнаходження, хоча вона вбудована в смартфони. Ця система моніторингу пацієнта використовує відстеження місцезнаходження та попередження, якщо пацієнт має смартфон.

У деяких інших системах [10] біосенсиори імплантуються в тіло пацієнта, якому може знадобитися хірургічна процедура та ускладнення технічного обслуговування та проблеми з цінами.

З попередніх досліджень та досвіду [11] відомо, що використання складного пристрою та технічне обслуговування є клопотом для людей похилого віку. Як наслідок, такі системи моніторингу можуть ускладнитися [12].

Ця система забезпечує міст між пацієнтами з технологічними проблемами та розумним життям. Завдяки простим функціям кнопок на допоміжному пристрої та РК-дисплеї (мобільному телефоні) легко керувати різними функціоналами, не потребуючи ніякого навчання чи допомоги.

Більше того, допоміжні пристрої, які можна носити, вже можуть отримувати повідомлення від основного пристрою із вмістом підпрограм. Однак інші функції мають бути реалізовані в майбутньому.

1.2 Огляд існуючих програмних засобів для вирішення завдання

Проблема створення сервісу для покращення рівню життя людей похилого віку постала паралельно з оцифровуванням сервісів та навколишнього середовища, тому вже існують певні рішення цієї проблеми. Під час роботи над дипломною кваліфікаційною роботою магістра були проаналізовані найбільш актуальні та популярні рішення. Аналіз та порівняння приведено в наступних підрозділах.

1.2.1 Опис аналогу «Health Pal»

Health Pal має всі функції, які ви коли-небудь вважали б необхідними, щоб підтримувати здоровий спосіб життя. Від лічильника кроків та нагадувань про дієту протягом дня до відстежування вживання їжі та фізичних вправ - програма Health Pal - це щоденний супутній інструмент, який полегшує вашу подорож до цілісно здорового способу життя. Тут зберігається інформація про ваш раціон, ваш фізичний стан та багато інших джерел здоров'я в одному місці.

До недоліків можна віднести те, що даний застосунок не має функції крокоміру, та значно спрощений асортимент пропонованих функцій. А деякі функції взагалі доступні лише після їх придбання. Даний застосунок не

приділяє стільки уваги до літньої частини населення, а більше орієнтований на досягнення користувача у фітнесі.

1.2.2 Опис аналогу «Remente»

Бути здоровим - це більше, ніж просто правильно харчуватися, пити достатньо води та добре спати - це також стосується налагодження розуму. Застосунок Remente надає вам безліч ресурсів, які допоможуть шукати у вашому житті щастя та здійснення, із встановленням цілей, щоденним інструментом планування щоденних завдань та довгострокових цілей, а також письмовими та візуальними функціями, які допоможуть вам відстежувати свої почуття детальні способи, які можуть допомогти вам краще зрозуміти, що приносить мету вашого життя.

До недоліків можна віднести те, що даний застосунок не має функції крокоміру, та значно спрощений асортимент пропонованих функцій. А деякі функції взагалі доступні лише після їх придбання. І хоча даний застосунок приділяє певну увагу до літньої частини населення, він все ще більше орієнтований на покращення рівня життя користувача та вироблення хороших звичок.

1.2.3 Опис аналогу «Health Nutrition & Fitness»

Спроба активного харчування та схуднення може здатися математикою, коли ви намагаєтесь розбити макроси, проаналізувати інгредієнти або, здавалося б, розрахувати кожен калорію. Ця програма допоможе вам зрозуміти, як вибір вашого загального раціону, а не фіксація певних поживних речовин, впливає на ваше здоров'я та споживання їжі. Він надає детальну інформацію про користь багатьох здорових продуктів для вегетаріанців та м'ясоїдів. Це також дозволяє вам розрахувати свій ІМТ та інші виміри тіла, щоб побачити, як зміни у вашому раціоні призводять до позитивних чи негативних результатів для здоров'я.

До недоліків можна віднести те, що даний застосунок не має функції крокоміру, та значно спрощений асортимент пропонованих функцій. А деякі функції взагалі доступні лише після їх придбання. Загалом даний застосунок не приділяє уваги конкретно до літньої частини населення, він все ще більше орієнтований на покращення рівня харчування.

1.2.4 Опис аналогу «Eufylife»

Застосунок Eufylife є супутником розумної шкали Eufylife та інших оздоровчих продуктів Eufylife, які допоможуть вам легше та безперешкодно відстежувати своє здоров'я та свою вагу протягом дня. Він синхронізується з вашими продуктами Eufylife, а також із вашим застосунком Apple Health, завдяки чому ви завжди маєте найновіший знімок ваги. Це може допомогти вам прийняти обдумані рішення щодо того, що ви їсте, коли ви тренуєтесь, і як ви плануєте підтримувати свою вагу на здоровому рівні для вашого типу фігури.

До недоліків можна віднести те, що даний застосунок не має функції крокоміру та відстеження досягнень та має значно спрощений асортимент пропонованих функцій. А деякі функції взагалі доступні лише після їх придбання. І хоча даний застосунок приділяє певну увагу до літньої частини населення, він все ще більше орієнтований на покращення рівня життя користувача та вироблення хороших звичок.

1.2.5 Результати аналізу та порівняння існуючих аналогів

Результати аналізу та порівняння вище вказаних аналогів зазначено у таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих рішень з розроблюваною системою.

	Розроблений застосунок	Eufylife	Health Pal	Remente	Health Nutrition Fitness
Безкоштовне завантаження	так	так	так	так	так
Не має платних функцій	так	так	ні	ні	ні
Доступність на Android	так	так	так	так	так
Доступність на iOS	ні	так	ні	так	ні
Має крокомір	ні	ні	ні	ні	ні
Має рутини	так	ні	ні	ні	ні
База даних алергій	так	ні	ні	ні	ні
Мультиаккаунтність	так	ні	ні	ні	ні
Аккаунт лікаря	так	ні	ні	ні	ні
Рекомендації ліків	так	ні	ні	ні	ні
Рекомендації їжі	так	ні	так	так	так
Рекомендації щодо ваги	так	так	ні	ні	так
Персоналізація профілю	так	ні	ні	ні	ні
Ознайомчий тур	так	ні	ні	ні	ні
Обмеження доступу	так	ні	ні	ні	ні
База даних ліків	так	ні	ні	ні	ні
База даних захворювань	так	ні	ні	ні	ні
Розклад прийому ліків	так	ні	ні	ні	ні
Система досягнень	ні	так	так	так	ні

1.3 Постановка завдання проблеми та її вирішення

Ця робота спрямована на розробку моделі, прототипу та простого мобільного застосунку для активної системи рекомендацій у режимі реального

часу в середовищі Інтернету речей (IoT) для підтримки потреб літніх людей та надання їм пропозицій та рекомендацій на основі їхнього контексту, які спрямовані на поліпшення якості їх життя.

Ця робота стосується програмного підходу IoT в галузі охорони здоров'я. Поширені випадки використання включають вимірювання частоти серцебиття, контроль дієти та контроль прийняття ліків, що приймаються за допомогою графіків, що використовуються у застосунку, є прикладом часто використовуваних функцій програми. Однак головна увага в цій тезі полягає у допомозі людям похилого віку організувати своє життя. Завдяки розробленому програмному забезпеченню людина може контролювати графіки прийняття ліків, доступність ліків, дієту та багато іншого, що створено для полегшення життя людині, яка цього потребує. Більш конкретно, розглядаються такі основні пристрої, як смарт-годинники та смартфони Android.

На сьогоднішній день усі живуть напруженим життям, наприклад, людині доводиться приймати різну кількість таблеток у різний час, і пропуск цих дій може коштувати йому життя. Для полегшення таких процесів, як прийом ліків та перевірка наявності ліків вдома, було вирішено створити програмне забезпечення, яке допоможе людям впоратися з цими справами. Якщо наше повсякденне життя стане легшим і менш складним, це буде краще, ніж золота ера технологій.

1.3.1 Системні вимоги

Основними вимогами до цього проєкту були: програмне забезпечення можна встановлювати на пристрої, що працюють від операційної системи Android.

Повинні бути 2 пристрої: мобільний застосунок з основним застосунком та додатковий носимий пристрій (годинник), який можна підключити до

мобільного телефону. Носимий на руці смарт-годинник повинен мати можливість отримувати рекомендації, тому він повинен мати дисплей.

Також були подані запити щодо системи мультиаккаунтності. Було зрозуміло, що повинна існувати система з кількома обліковими записами, яка зможе створити принаймні одного лікаря та одного пацієнта.

1.3.2 Планування проєкту

У цій роботі було вирішено використовувати гнучке планування як основний метод планування проєкту. Для реалізації всього було обмежена кількість часу, тому цей метод був найбільш підходящим для даної дипломної кваліфікаційної роботи магістра. Він оцінює роботу за допомогою автономних робочих одиниць, які називаються ітераціями або спринтами. Спринти - це періоди певної кількості часу, як правило, 1-3 тижні, коли команда або окрема особа зосереджується на невеликому наборі робочих завдань і має на меті їх завершити. Швидке планування визначає, які завдання виконуються в кожному спринті, і створює повторюваний процес, щоб допомогти командам або окремій людині дізнатися, скільки вони можуть досягти.

У цій дипломній кваліфікаційній роботі магістра робота була розділена на три спринти: перший спринт – аналіз проєкту. Другий спринт - розробка додатків. Третій спринт - опис програмного забезпечення та документація.

Після деяких досліджень у цій галузі було прийнято таке рішення: метод “Scrum” більше підходить для команд, а “Agile” можна використовувати для одного розробника. Отже, саме Agile було взято до уваги.

План розробки додатків був розділений на три основні категорії: аналіз раніше доступних функцій, функцій, які слід додати, і майбутня робота. Це показано на рисунку 1.1.

План розробки		
Попередньо доступний функціонал	Розроблені функції	Буде розроблено у майбутньому
База ліків Розпорядок Розклади База алергій Ознайомчий тур	Аккаунт Лікаря Рекомендації щодо їжі Розширена реєстрація Рекомендації щодо лік Загальні рекомендації	Крокомір Надзвичайна ситуація Досягнення користувачів Відстеження руху та розташування Записи рівня цукру Записи поточної ваги Вимір частоти серцебиття Запис частоти серцебиття Вимір насиченості крові киснем Запис насиченості крові киснем Пошук найближчої аптеки Найближче розважальне місце

Рисунок 1.1 – План розробки застосунку

Також у цій роботі дотримувалась стратегія життєвого циклу розробки програмного забезпечення (рис. 1.2) , тому вона також була реалізована в парі з методами управління проєктами. „Співпраця” між цими двома напрямками досліджень показана на рисунку 1.3.



Рисунок 1.2 – Життєвий цикл розробки програмного забезпечення цього проєкту

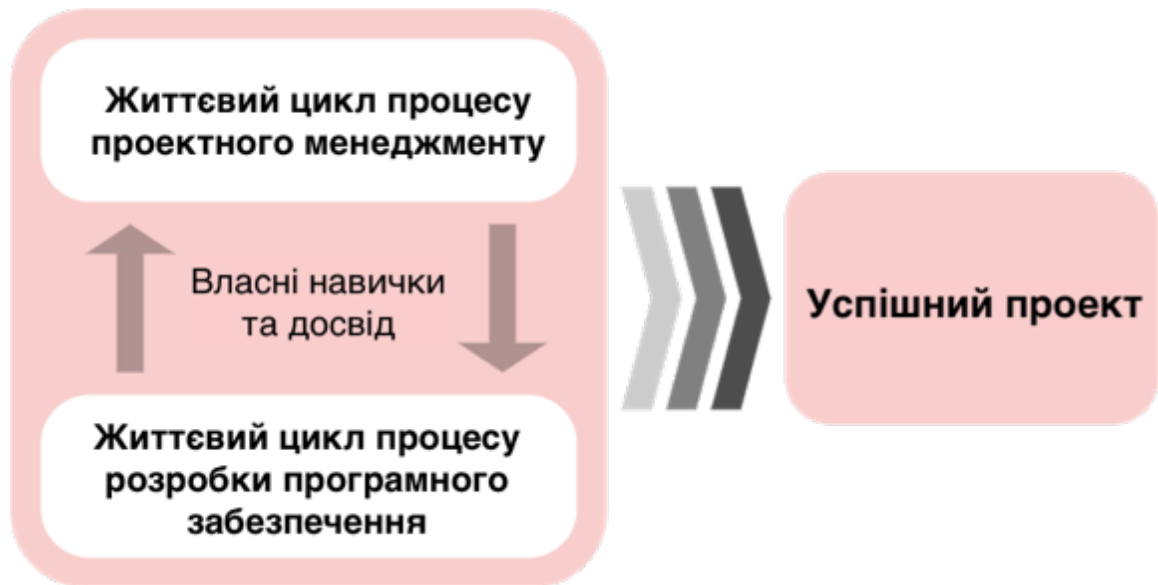


Рисунок 1.3 – Потреби для успішного виконання проєкту

1.3.3 Аналіз системи

Для реалізації всього, що було перелічено у вимогах, було зроблено кілька простих кроків. Найпершим кроком було вирішення, яку мову розробки слід використовувати в цій роботі. Другим кроком був аналіз усіх необхідних функцій, якщо їх можна реалізувати на цій платформі. Потім було створено бачення структури та знайдено відповідний проєкт з відкритим кодом, який можна успішно розширити [19]. Наступним кроком був вибір інтегрованого середовища розвитку.

1.4 Висновки до розділу 1

Було розглянуто існуючі програмні засоби, які виконують аналогічні завдання, або схожі за типом, були взяті до уваги їх переваги та недоліки, описана мета дипломної кваліфікаційної роботи магістра. До цього ж був описаний функціонал мобільного застосунку, що розроблюється, та його основні принципи.

2 АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТІВ І ЗАСОБІВ ДЛЯ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Огляд особливостей платформи

Застосунки для мобільних пристроїв, які працюють під управлінням ОС Android, складаються з так званих «операцій», з якими можуть взаємодіяти користувачі для виконання всіляких дій. Для кожної операції присвоюється екран для відображення відповідного інтерфейсу для користувача.

Як правило, застосунок складається з декількох операцій, які слабо пов'язані один з одним. Зазвичай одна з операцій розроблюваного застосунку визначається як головна, і відображається користувачам при першому запуску застосунку. Варто відзначити, що кожна операція може запустити іншу для виконання різноманітних дій.

Порівняння платформ для розробки наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння платформ для розробки.

Критерій	Android	iOS
Поширеність серед людей похилого віку	Висока	Низька
Доступність застосунків	Висока	Низька
Наявність вбудованих сенсорів у пристрої	Часто	Відсутні

Таким чином було обрано саме платформу Android для розробки застосунку, як платформу, яка більше підходить для реалізації поставленої мети.

2.2 Огляд особливостей середовища розробки

Для розробки мобільного застосунку системи надання рекомендацій на основі контекстного методу людям похилого віку або осіб які того потребують, обрано інтегроване середовище розробки «Android Studio».

Обране середовище розробки було засноване на базі середовища розробки «IntelliJ IDEA». Ця програма і досі випускається як окремий програмний продукт. Головними перевагами «Android Studio» є безкоштовна форма доступу і статус офіційного середовища розробки для Android-застосунків любого типу та для любого пристрою с Android ОС.

Для середовища Android Studio постійно випускаються оновлення, додаючи функціонал для того, щоб зробити процес розробки зручніше та зрозуміліше для розробників початкового рівня. Наразі у ньому доступні наступні функції:

- розширений редактор макетів, який працює за технологію «що бачиш, те і отримаєш». Він дозволяє працювати з елементами інтерфейсу користувача методом «Drag-and-drop»;
- збірка застосунків за допомогою системи «Gradle». Така система була побудована на принципах Apache Ant та Apache Maven, але замість використання XML-подібної форми представлення конфігурації проєкту – «Gradle» використовує мови Kotlin та Groovy [9];
- різноманітні види збірок для застосунку та генерація декількох .apk файлів;
- доступний інструментарій для рефакторингу програмного коду;
- надано статистичний аналізатор коду, що дозволяє знаходити проблеми продуктивності або несумісності версій та інше;
- є вбудований застосунок «ProGuard», який призначений для оптимізації та обфускації розроблюваного програмного коду. Також є утиліта для підписування власних застосунків розробника;

- підтримка розробки застосунків для платформ Wear OS та Android TV;
- емулятор Android, або віртуальний пристрій.

Таблиця 2.2 – Порівняння можливих середовищ розробки.

	Android Studio	Visual Studio
Вбудований емулятор Android пристроїв	Так	Так
Нативна підтримка мов програмування	Так	Ні
Нативна підтримка Gradle	Так	Ні
Вбудована підтримка баз даних SQLite	Так	Ні

Вибір середовища для розробки застосунку грає ключову роль у подальшій зручності написання та обслуговування програмного засобу.

2.3 Огляд особливостей мови програмування

Для написання мобільного застосунку для надання рекомендацій щодо здоров'я людям похилого віку та іншим користувачам на основі контекстного методу обрано такі мови програмування, як Java та Kotlin.

Результати аналізу та порівняння можливих мов програмування зазначено у таблиці 2.3.

Таблиця 2.3 – Порівняння можливих мов програмування.

	Java	Kotlin
Спільнота	Велика	Не велика
Кількість бібліотек	Багато	Не багато
Синтаксис	Менш зручний	Зручніше

Вибір мови програмування у даному випадку зіграв не значну роль, оскільки обидві мови були використані при розробці ПЗ.

2.4 Огляд особливостей рекомендаційних методів

Обираючи метод надання рекомендацій було виявлено два найпоширеніших методи. Метод випадкового підбору може повноцінно використовуватися у даному застосунку, але його ефективність для поставлених цілей для розробки, значно нижча за контекстний метод надання рекомендацій.

Контекстний метод має змогу проаналізувати користувача саме за тими параметрами, що було запрограмовано розробником та можуть бути оброблені лише ті дані, які погодився дати о собі користувач.

Порівняння цих двох методів наведено на таблиці 2.4.

Таблиця 2.4 – Порівняння методів надання рекомендацій.

	Контекстний метод	Випадковий підбір
Бере до уваги вік	Так	Ні
Бере до уваги недуги	Так	Ні
Бере до уваги вагу	Так	Ні
Бере до уваги зріст	Так	Ні

Контекстний метод рекомендацій базується на реєстрації користувача. Чим менше надано даних, тим менше рекомендацій має змогу дати система. Контекстний метод було обрано як найвлучніший у своїх результатах та відносно не складний у реалізації. Єдиний недолік такого методу, це якщо користувач повністю відкажеться від надання інформації о собі, тоді програмі доведеться не надавати рекомендацій взагалі, так як перш за все цей метод базується на наданих та доступних даних.

2.5 Висновки за розділом

У цьому розділі був обґрунтований вибір платформи для розробки, виходячи з чого було зроблено обґрунтування мови програмування та здійснено обґрунтування вибору середовища розробки. Було визначено як розроблявся мобільний застосунок, основні підходи та принципи розробки. Також визначено який метод надання рекомендацій підходить найбільше до вирішення поставленої задачі.

3 ДИЗАЙН ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Використані бібліотеки

Використання бібліотек у сучасному процесі розробки застосунків - чи не головне, і насправді запорука успіху. Використання бібліотек може заощадити величезну кількість часу в процесі розробки застосунків. Розробникам не потрібно створювати застосунок з нуля, все просто потрібно скласти з існуючих фрагментів. Існує безліч бібліотек, що використовуються в процесі розробки програми, але основні бібліотеки виділено нижче.

JUnit - для реалізації тесту програми. JUnit - це простий фреймворк для написання повторюваних тестів. Це приклад архітектури xUnit для модульних модулів тестування.

Jsoup - для розбору HTML. Jsoup - це бібліотека Java для роботи з реальним HTML. Він забезпечує дуже зручний API для отримання URL-адрес та вилучення та обробки даних, використовуючи найкращі з DOM-методів HTML5 та селектори CSS. Jsoup реалізує специфікацію WHATWG HTML5 і аналізує HTML до тієї ж DOM, що і сучасні браузері. Видалити та проаналізувати HTML з URL-адреси, файлу або рядка; F Ind та видавати дані, використовуючи DOM обходу або CSS селектори; м аніпулювати елементами HTML, атрибутами та текстом; с нахилити вміст, поданий користувачем, до безпечного білого списку, щоб запобігти атакам XSS; о виводить прибирання HTML. Jsoup призначений для роботи з усіма різновидами HTML, що зустрічаються в дикій природі; від первозданної та валідаційної, до недійсної суп- мітки; jsoup створить розумне дерево синтаксичного аналізу [13].

Zxing - для підтримки всього, що стосується QR-кодів. ZXing («перетин зебри») - це бібліотека для обробки зображень зі штрих-кодом, реалізована на Java, з портами на інші мови. Він підтримує одновимірний продукт, одновимірні промислові та двовимірні штрих-коди [14].

Joda-time - для обробки часу. Joda-Time забезпечує якісну заміну класів дати та часу Java [15].

Butterknife - для зручності прив'язки ресурсів.

Caldroid - для кращого перегляду календаря. Caldroid - це фрагмент, який відображає календар із датами за місяць. Caldroid можна використовувати як вбудований фрагмент або як діалоговий фрагмент. Користувачі також можуть проводити пальцем вліво / вправо, щоб перейти до різних місяців. Налаштувати зовнішній вигляд Caldroid за допомогою власної теми дуже просто. У Caldroid є дві теми за замовчуванням (світла та темна). Розробник може також запропонувати власну тему на основі цих тем за замовчуванням. Caldroid повністю локалізований. Розробник може налаштувати день початку тижня для різних країн. Наприклад, за замовчуванням календар починається в неділю [16].

RoundedImageView - для круглих аватарів, що використовуються для представлення користувачів. Це швидкий ImageView (і для малювання), який підтримує закруглені кути (і овали або кола). Він підтримує безліч додаткових функцій, включаючи овали, округлі прямокутники, ScaleTypes і TileModes. Є багато способів створити закруглені кути в android, але це найшвидший і найкращий із відомих, оскільки він не створює копію оригінального растрового зображення, не використовує clipPath, який не є апаратно прискореним і не згладжується, не використовує setXfermode, щоб відрізати растрове зображення та двічі намалювати на полотні [17].

MaterialDrawer - для лівого висувного меню, яке містить усі основні функції програми. MaterialDrawer підтримує отримання зображень із URL-адрес та встановлення їх для піктограм профілю. Оскільки MaterialDrawer не містить бібліотеки ImageLoading, розробник може вибрати власну реалізацію. Це потрібно зробити перед завантаженням першого зображення через URL [18].

У мобільному додатку використовувались такі компоненти Android:

- AppCompat - надає зворотну сумісність;

- RecyclerView - розширення, яке дозволяє розробнику створити віджет з однойменною назвою, який є вдосконаленою та гнучкішою версією віджета ListView;

- навігація – допомагає здійснити навігацію, від простих натискань кнопок до більш складних моделей, таких як панелі програм та навігаційна панель;

- фрагмент – поведінка або частина користувацького інтерфейсу в FragmentActivity, дозволяє розробнику програми поєднувати декілька фрагментів для створення багатопанельного інтерфейсу та повторного використання фрагмента.

3.2 Застосунок для Android смартфона

Мобільні застосунки для мобільних пристроїв під управлінням Android складаються з так званих "операцій", за допомогою яких користувачі можуть взаємодіяти, виконуючи різні дії. Для кожної операції призначається екран для підкреслення відповідного інтерфейсу користувача. Як правило, застосунок складається з декількох операцій, які слабо пов'язані між собою. Зазвичай одна з операцій програми визначається "основним" і відображається користувачеві при першому запуску програми. У свою чергу, кожна операція може запускати іншу для виконання різних дій.

Прикладне програмне забезпечення побудовано з використанням інтегрованого середовища розробки Android Studio і передбачає використання цього продукту на телефонах, керованих системою Android. Комплекс програмного та апаратного забезпечення, розроблений з використанням принципів IoT, фактично призначений для допомоги людям похилого віку у:

- дотриманні дієти;
- моніторингу їх фізичної активності;
- моніторингу інших показників, пов'язаних зі здоров'ям;
- вибору місць для відвідування;

- вибору аптечних магазинів для відвідування;
- вживання таблеток вчасно;
- зберігання активності протягом дня, якщо це потрібно;
- отримання миттєвого відгуку у разі надзвичайної ситуації;
- залишатися мотивованими цілями або системою балів;
- уникати вживання заборонених ліків.

Всі дії обробляються ефективно. Застосунок зробить це, отримуючи дані від датчиків на годинниках, поєднуючи їх з даними, які надаються при реєстрації користувачів, попередження про лихо та сповіщення, передані з пристрою. Прикладне програмне забезпечення розділене на кілька видів діяльності з різними функціональними можливостями.

3.3 Програмне забезпечення для другорядного пристрою

Програмне забезпечення застосунку для пристрою, який можна носити, також створене з використанням інтегрованого середовища розробки Android Studio і передбачає використання цього продукту на годинниках, які мають сенсорний екран і керовані системою Android.

Комплекс програмного та апаратного забезпечення, розроблений з використанням принципів IoT, насправді призначений для забезпечення читання інформації (показників стану здоров'я) та передачі її до головного пристрою - мобільного телефону. Носимий пристрій не буде робити ніяких обчислень, всі внутрішні обчислення будуть виконуватися на стороні телефону. Допоміжний пристрій не тільки збиратиме та надсилатиме інформацію, але й братиме участь у отриманні рекомендацій та обробці повідомлень у майбутньому.

Перелік основних функцій допоміжного пристрою:

- отримувати сповіщення;
- отримувати рекомендації;

- надсилати дані з акселерометра;
- надсилати дані з біометричних сенсорів.

3.4 Виконання системи

Запропонована система складається з двох окремих систем, які в поєднанні надають послуги з охорони здоров'я та рекомендацій. Система побудована з урахуванням усіх основ моделей IoT. Структура програми з точки зору IoT продемонстрована на рисунку 3.1. Функції системи згадані нижче.

- мобільний телефон з ОС Android із встановленим основним програмним забезпеченням;
- Android годинник підключено до телефону;
- реєстрація у застосунку шляхом надання даних про користувача та лікаря показана на рисунку 3.2.

Рисунок 3.2 представляє бажаний процес реєстрації, але додавання телефонних номерів для екстрених дзвінків, а також системи екстрених дзвінків буде розроблено лише в майбутньому. Це описано в розділі «Майбутня робота».

Основна програма складатиметься з наступних екранів та функцій: основні екрани, основні функції, щоденні вимірювання. Впроваджено лише перші п'ять основних екранів, усі інші функції та екрани є частиною майбутньої роботи.



Рисунок 3.1 – IoT структура застосунку

Процес реєстрації	
Користувач	Лікар
Ім'я	Ім'я
Вік	E-mail
Номер телефону	Адрес
Захворювання	Номер телефону
Стать	
Вага	
Зріст	
E-mail	
Адрес	
Світлина аккаунту	
Фон аккаунту	
Можна створити розклад по замовчуванням	

Рисунок 3.2 – Зміст реєстраційної форми

На рисунку 3.3 представлена система з усіма очікуваними характеристиками для кращого бачення кінцевого продукту.



Рисунок 3.3 – Основні екрани та функції кінцевого продукту

Функції, засновані на системі рекомендацій, показані на рисунку 3.4, рекомендації на основі даних датчиків GPS показані для кращого бачення кінцевого продукту, оскільки їх планується впровадити лише в майбутньому.

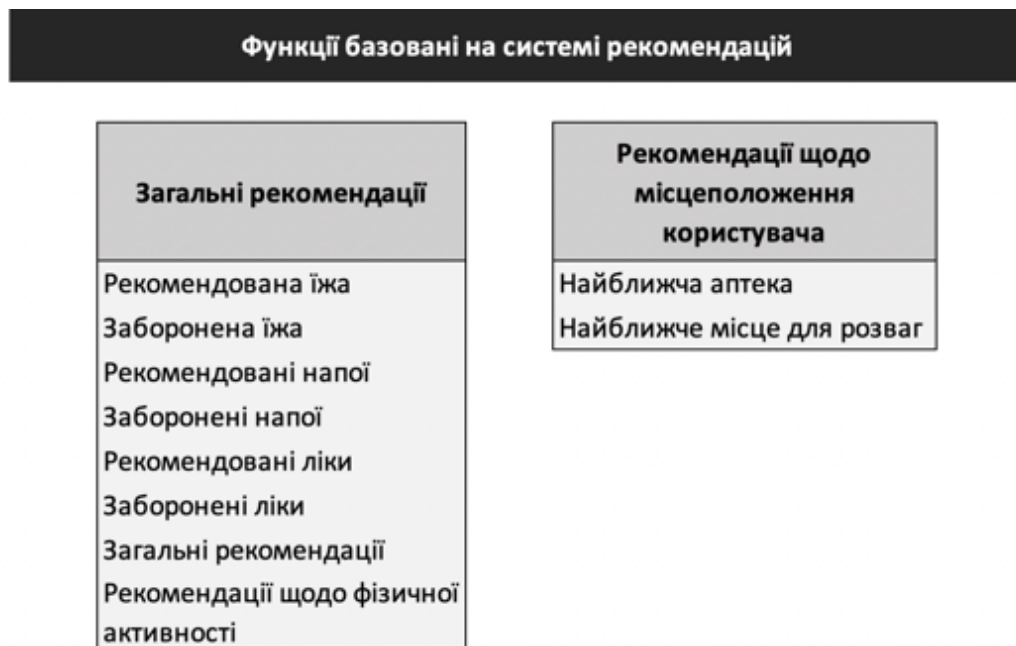


Рисунок 3.4 – Функції що засновані на рекомендаційній системі

Варіант використання, показаний на рисунку 3.5, представляє дії користувача програми відразу після знаходження цієї програми на ринку застосунків.

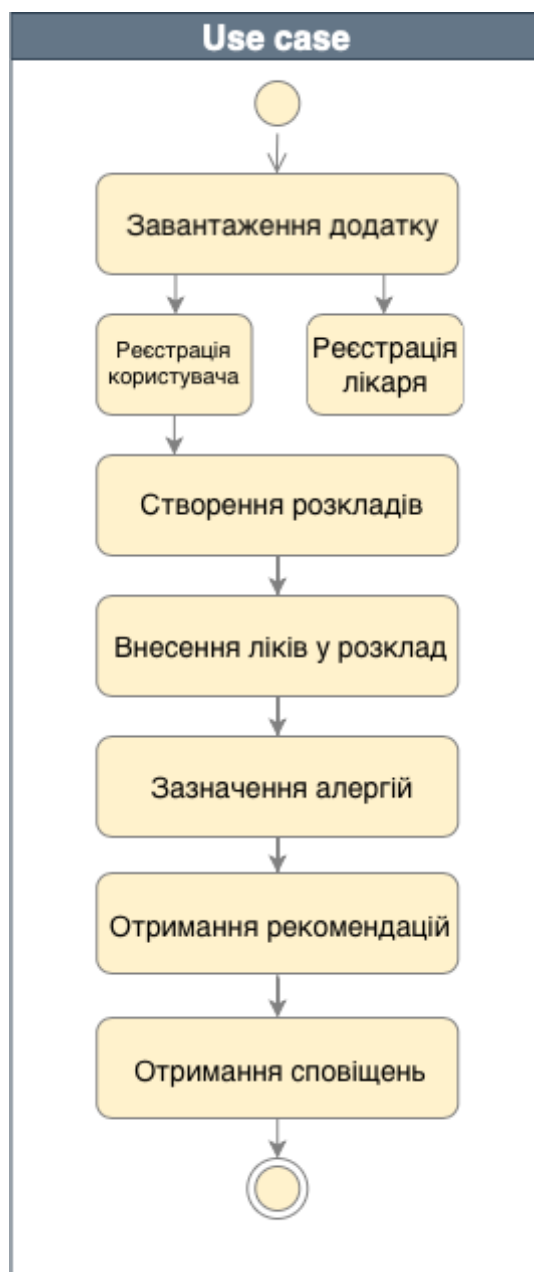


Рисунок 3.5 – Діаграма використання

3.5 Апаратні компоненти

Для впровадження цієї системи в роботу лише два пристрої: мобільний пристрій та носяться пристрій. Обома пристроями повинна керувати

операційна система Android. Вимоги до обладнання будуть описані далі в цьому розділі.

3.5.1 Допоміжний пристрій

Для забезпечення постійного моніторингу дані в режимі реального часу збираються за допомогою декількох датчиків за допомогою допоміжного пристрою, який можна носити, наприклад, розумних годинників із сенсорним екраном, керованих ОС Android. Для розрахунку загального стану людей похилого віку використано датчик пульсу, датчик температури, акселерометр, зумер, кнопки та модуль GPS.

Датчик пульсу - це біометричний датчик виявлення серцевого ритму, який також може вимірювати насиченість крові киснем. Для лікаря або медичного спеціаліста показники температури тіла дають значні дані, які можуть виявити захворювання. На екрані є інтерактивна кнопка, яка використовується як чек-пост для екстрених дзвінків. Акселерометр використовується для виявлення загального руху пацієнтів. Усі датчики в комплексі контролюватимуть стан здоров'я літньої людини. Регульований та легко знімний гумовий ремінець надає пристосуванню можливість лягти на зап'ястя пацієнта.

3.5.2 Основний пристрій

Для того, щоб забезпечити всі розрахунки, забезпечити підключення до Інтернету та зручний інтерфейс, мобільна частина програми буде найкращим вибором для використання. Телефон може легко надати рекомендаційну функціональність програми.

Якщо в телефоні є датчик частоти серцебиття, то такі розрахунки можна проводити без використання наручних годинників. Смартфон може забезпечувати постійний моніторинг, збір даних у режимі реального часу може використовувати кілька датчиків, а також, смартфон повинен мати

можливість підключатися до допоміжного пристрою, який можна носити, а саме до розумних годинників із сенсорним екраном, керованих Android. Bluetooth модуль буде використовуватися для забезпечення безперервного з'єднання між телефоном і смарт годинником. Мобільний телефон бажано повинен мати модуль GPS.

Для розрахунку загального стану людей похилого віку використано датчик пульсу, датчик температури, акселерометр, зумер, кнопки та модуль GPS. Датчик пульсу - це біометричний датчик виявлення серцевого ритму, який також може вимірювати насиченість крові киснем.

Смартфон повинен мати можливість обробляти екстрені дзвінки. На сьогодні цю дію можна зробити або за допомогою VoIP, або за допомогою звичайного дзвінка за допомогою SIM-карти телефону. Акселерометр використовується для виявлення загального руху пацієнтів.

Усі датчики в комплексі контролюватимуть стан здоров'я літньої людини.

3.6 Функціонування мобільного застосунку

Коли користувач відкриває програму вперше - користувацька робота за допомогою цієї програми почнеться з швидкого огляду. Його можна або повністю пропустити, або користувач може переключатися між слайдами з корисною інформацією щодо використання цієї програми.

Робота мобільного застосунку починається з відображення його домашнього екрану, який є екраном з п'ятьма основними функціями всього програмного забезпечення. Головний екран дозволяє користувачеві знайти потрібну йому функцію, просто натиснувши на піктограму, екран змінить фрагмент, не змінюючи всі дії. Наприклад, на рисунку 3.6) показано макет домашньої сторінки.

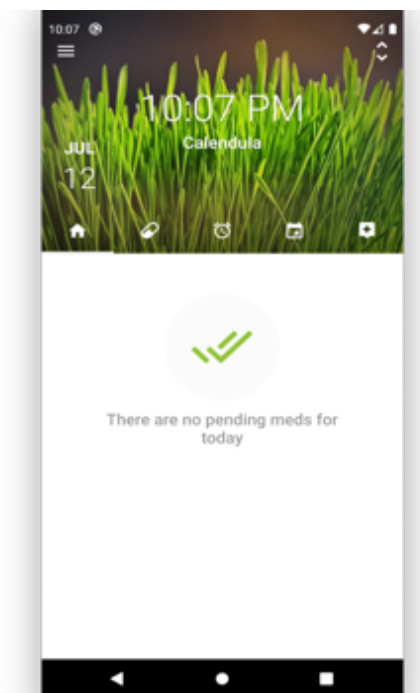


Рисунок 3.6 – Головний екран програми

На рисунку 3.7 представлена діаграма класів діяльності головної сторінки застосунку та класів, які вона реалізує.

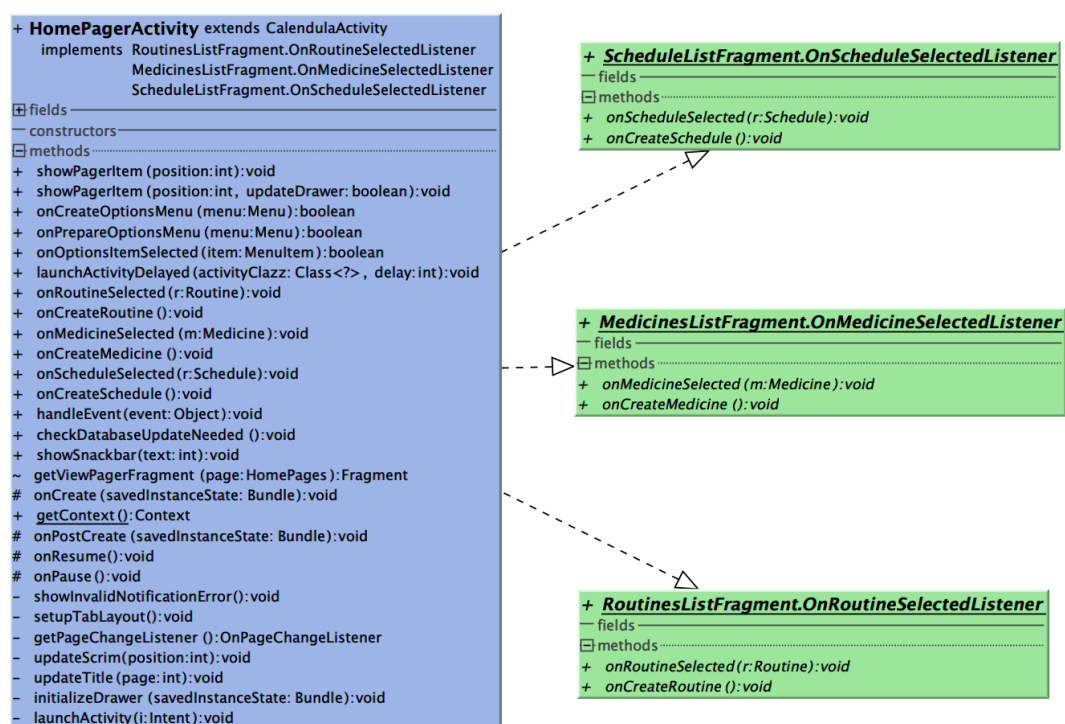


Рисунок 3.7 – Схема класів діяльності домашньої сторінки та класів, які вона реалізує

Щоб отримати доступ до всіх доступних функцій, користувач повинен провести пальцем праворуч, щоб з'явилося ліве меню зі списком усіх доступних функцій. Доступні деякі додаткові функції, наприклад, зміна даних користувача, швидка зміна користувача, створення нового користувача та список усіх облікових записів, включаючи обліковий запис лікаря. Ящик дозволяє користувачеві вибрати потрібний йому обліковий запис для подальшого перегляду рекомендацій та процедур, доступних для цього конкретного облікового запису. Діяльність ящика та класи, які він реалізує, представлені на рисунку 3.8 разом із відповідною кнопкою.

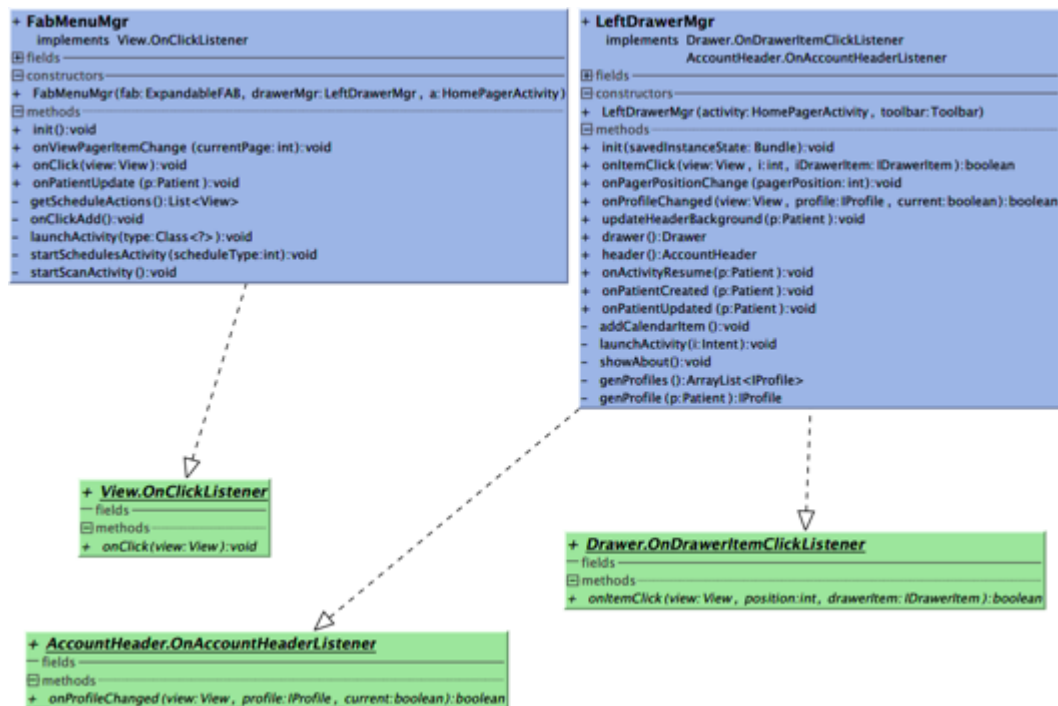


Рисунок 3.8 – Діаграма класу лівого меню та плаваючої кнопки дії та класи, які вони реалізують

Коли користувач переглядає, наприклад, екран рекомендацій, а потім хоче змінити обліковий запис за допомогою швидкої зміни у верхній частині меню (рисунок 3.9) - рекомендації пацієнта або будь-який інший екран із цього списку буде оновлено негайно.

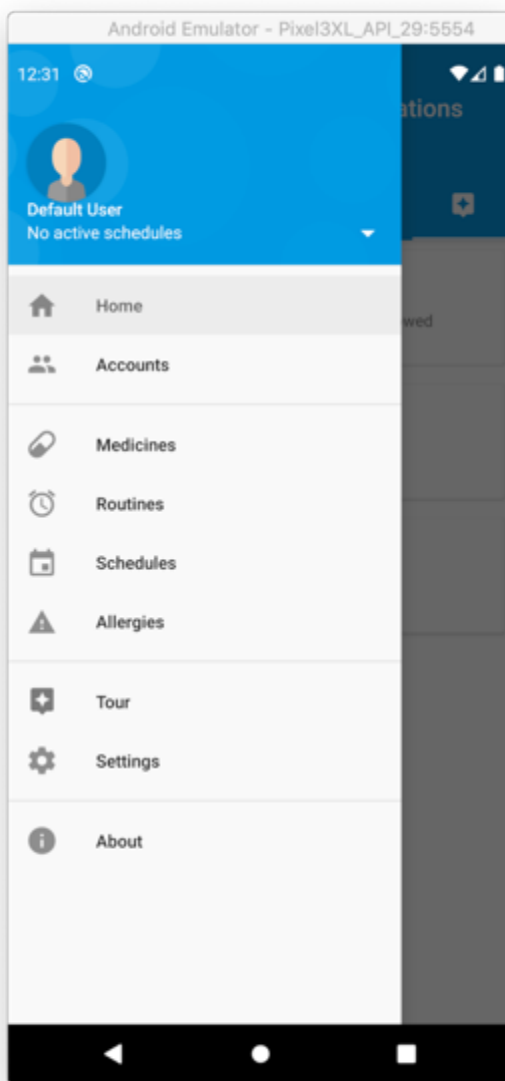


Рисунок 3.9 – Вид лівого меню та спосіб, яким він закриває вигляд фрагмента рекомендацій

Є деякі функції конфіденційності, які можна знайти в розділі "Налаштування", наприклад, встановити ПІН-код програми, щоб при відкритті програми пацієнт повинен був довести, що саме він може переглянути всю використану інформацію у цій програмі.

Ще одна особливість знаходиться в розділі "Налаштування" - база даних ліків. Користувач може вибрати базу даних ліків, яку він хоче, для подальшої ідентифікації всіх назв лікарських засобів, які використовуються у процедурах.

3.7 Система мультиаккаунтності

Мультиаккаунтність означає, що ця програма може не тільки створювати та переключатися між користувачами (пацієнтами), але також має розділ "Лікар" у селекторі облікових записів.

Усі доступні функції облікового запису лікаря скорочені, але функціонал можна продовжити у майбутньому. Розділ «лікар» не має однієї з ключових особливостей, які доступні для пацієнтів - мультиаккаунтної системи.

Для лікаря вирізані не тільки певні функції, але й наявна інформація. Існує порівняння доступної інформації між пацієнтом та лікарем, показане на рисунку 3.10.

	Користувач	Лікар
Можна встановити вік	✓	×
Можна встановити ім'я	✓	✓
Має доступ до розкладу	✓	×
Можна встановити фон профілю	✓	×
Може отримувати рекомендації	✓	×
Можна встановити аватар	✓	×
Можна додати email	✓	✓
Можна встановити адресу	✓	✓
Можна встановити зріст	✓	×
Можна встановити вагу	✓	×
Можна встановити хворобу	✓	×
Можна створити декілька аккаунтів	✓	×

Рисунок 3.10 – Порівняння наявної інформації про лікаря та пацієнта в системі

Ідея системи кількох облікових записів полягає у створенні можливості керувати кількома пацієнтами з одного конкретного пристрою. Ця система може бути корисною в сім'ях або якщо пристрій із встановленим програмним забезпеченням розміщений у камері для пацієнтів. Кожен пацієнт, який створює обліковий запис у цій програмі, має все окремо, крім головного лікаря (рисунок 3.11). Це, звичайно, можна змінити в майбутньому.

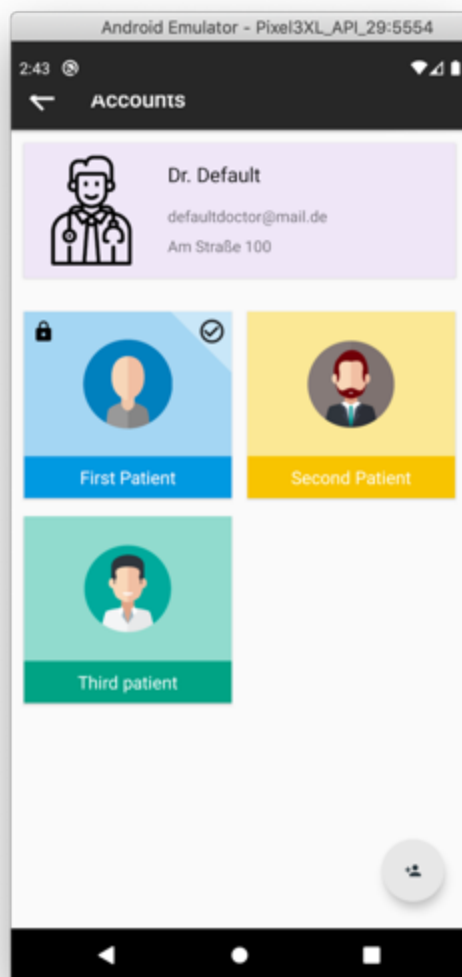


Рисунок 3.11 – Система облікових записів

Існує два сценарії створення нового пацієнта у застосунку. Перший - використовувати ліве меню. Користувач повинен провести пальцем праворуч, щоб з'явилася ліве меню, а потім вибрати розділ «Облікові записи». Після натискання на кнопку «Облікові записи» з'явиться список вже доступних облікових записів (рисунок 3.11), які користувач може перемикаати між собою,

або внизу списку рахунків буде опція «Додати пацієнта». Це сценарій А, який показаний на рисунку 3.12.

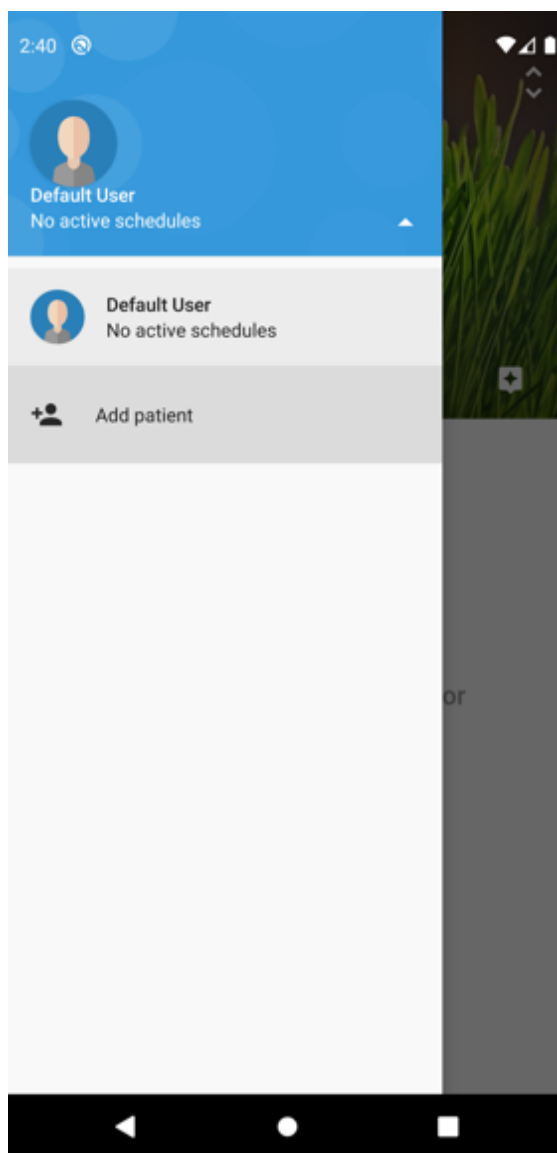


Рисунок 3.12 – Сценарій А

У тому випадку, якщо користувач вже ввійшов на екран списку пацієнтів, немає необхідності використовувати ліве меню для створення нового пацієнта. Користувач може просто використовувати плаваючу кнопку дій, розташовану в нижньому правому куті з іконкою людини з невеликим знаком «плюс». Це призведе безпосередньо до реєстрації нового користувача. Це сценарій створення користувача, який показаний на рисунку 3.13.

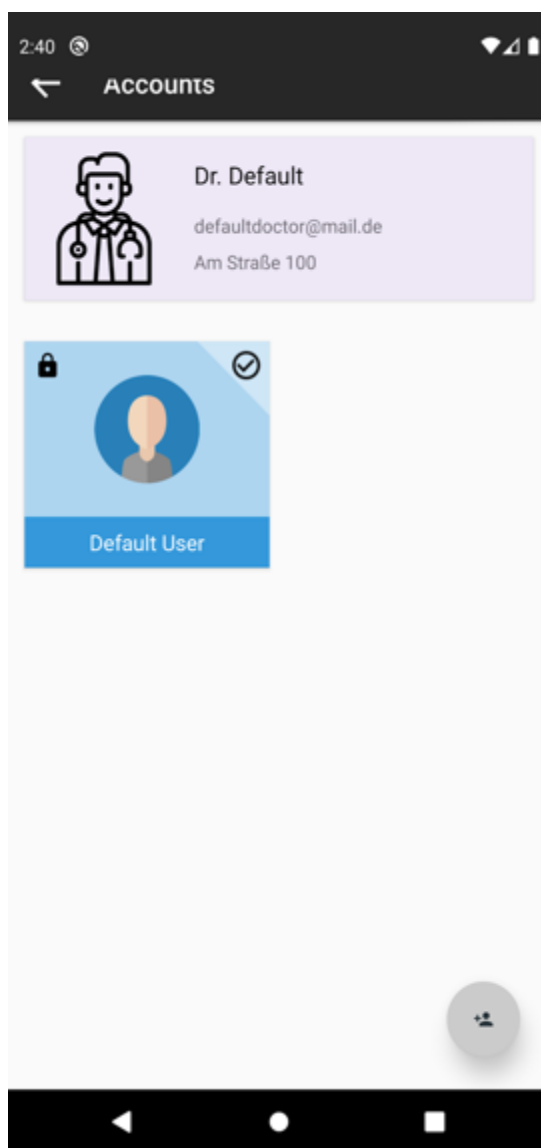


Рисунок 3.13 – Сценарій Б

3.8 Реєстрація

Під час процесу реєстрації збирається основна інформація про користувача, щоб на основі даних, введених користувачем, програма могла обробляти дані, і стало можливим робити рекомендації та відображати корисну інформацію. Персоналізований вміст є ключовою особливістю програми та є найважливішою частиною програми. Отже, зрозуміло, що без даних, введених користувачем, ми втратимо весь сенс програми.

Список основних речей, які повинен надати користувач у процесі реєстрації:

– повне ім'я - таким чином, це може бути використано, наприклад, для того, як лікар може ідентифікувати користувача (це поле є обов'язковим, але немає жодних обмежень щодо того, що користувач бажає ввести, якщо пацієнт вважає за краще показувати лише своє ім'я, лише прізвище чи вигаданий псевдонім - програма не забороняє цього);

– вік - це поле не є обов'язковим і не впливає на рекомендації пацієнта (але надалі програма зможе проаналізувати ризики, пов'язані з віком людини, та визначити, до якої вікової категорії належить пацієнт, наприклад, дитина, дорослий до 50 років, дорослий старше 50 років);

– вага, зріст - показник ваги використовується в парі з показником зросту (дані, наведені в цих полях, обробляються для обчислення відношення ваги до висоти, залежно від коефіцієнта, програма надасть користувачеві рекомендації щодо того, як це може вплинути на його здоров'я та що рекомендується робити, щоб, наприклад, набрати вагу або схуднути, якщо ці поля не будуть заповнені інформацією при реєстрації, то система не надаватиме жодних рекомендацій користувачеві щодо цього співвідношення);

– хвороба - цей показник можна вибрати зі списку доступних захворювань, які програма може розпізнати (залежно від наданих даних, програма надасть користувачеві рекомендації щодо того, як це може вплинути на його здоров'я, і що рекомендується робити, буде декілька рекомендацій, а не лише одна, це потрібно для окремих рекомендацій, що описують заборонені дії та дії, які рекомендується робити щодо хвороби пацієнта);

– електронна пошта – ми живемо у двадцять першому столітті, і адреси електронної пошти є майже головним, як постачальники програм або лікарі можуть зв'язуватися з пацієнтами за допомогою Інтернет (це конкретне поле не бере участі в жодному з процесів, що продаються через застосунок, але, звичайно, його можна використовувати вручну, надалі застосунок може використовувати ці дані для розсилки рекламних роликів або спеціальних пропозицій користувачам);

– адреса – це поле використовується як дані, надані користувачем щодо його домашньої місцевої адреси (це конкретне поле не бере участі в жодному з процесів, що продаються через застосунок, але, звичайно, його можна використовувати вручну).

Надалі застосунок може використовувати ці дані для формування рекомендацій на основі місцезнаходження, наприклад, показуючи найближчу аптеку як підказку. Список усіх рекомендацій, що базуються на розташуванні, наведено в розділі "Рекомендації, що базуються на місцезнаходженні".

Цей список можна поповнити безліччю інших корисних даних, наприклад - персональний страховий номер, мобільний телефон тощо. Усі функції, які збираються розробити, описані в розділі "Майбутня робота".

Випадок використання реєстрації користувача представлений на рисунку 3.14.

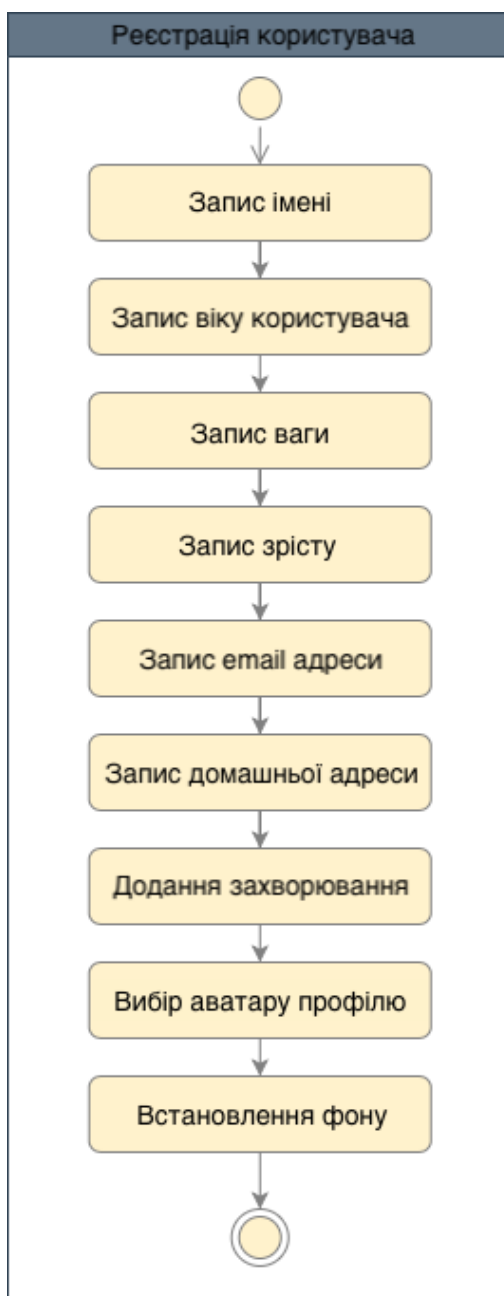


Рисунок 3.14 – Покрокова реєстрація пацієнта в заявці

Реєстрація лікаря - це не повністю, але трохи інший процес, який вимагає менше терпіння через менший обсяг даних, який потрібно надати, і цей крок відрізняється від самого початку. Це показано на рисунку 3.15.

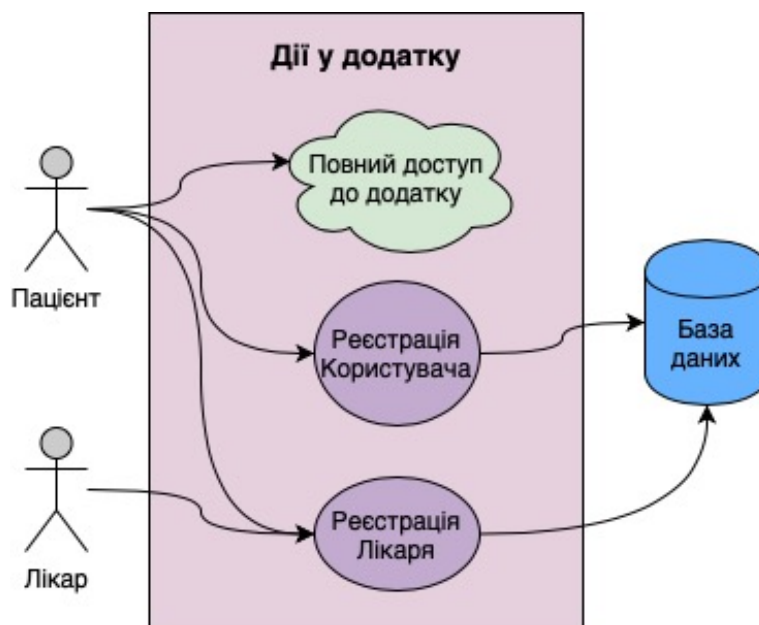


Рисунок 3.15 – Різниця у доступі до реєстрації між лікарем та користувачем

Інтерфейс реєстрації також відрізняється. Інтерфейс реєстрації користувача (макет) зображено на рисунку 3.16.

Екран реєстрації користувачів також містить одну конкретну дію - це селектор хвороби. Натиснувши кнопку “Вибрати”, з’явиться спливаюче діалогове вікно, де користувач може вибрати одну із захворювань, що є у списку. Цю дію потрібно затвердити, натиснувши кнопку “Вибрати” внизу праворуч у спливаючому діалоговому вікні. Всі описані вище дії представлені на рисунку 3.17.

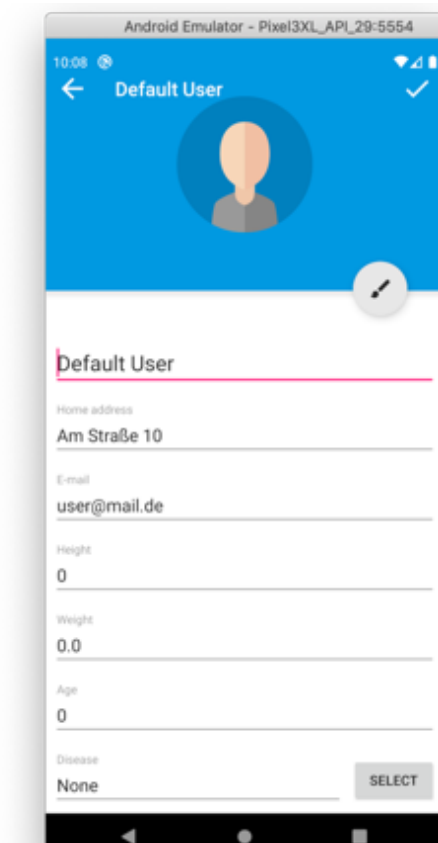


Рисунок 3.16 – Інтерфейс реєстрації користувача

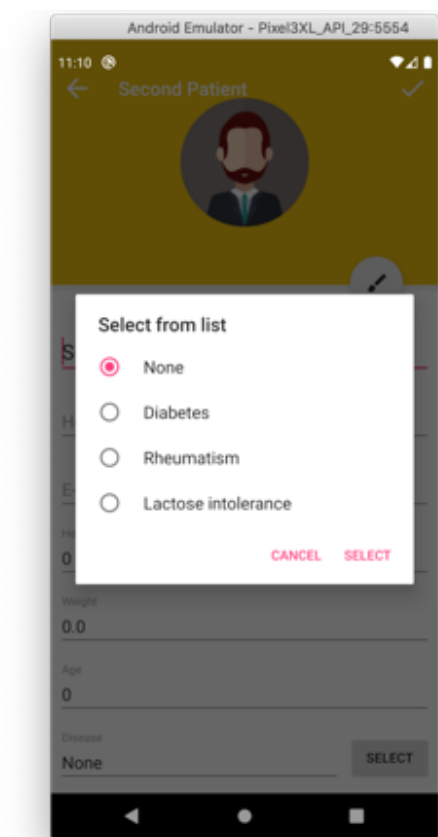


Рисунок 3.17 – Спливаюче діалогове вікно вибору захворювання

Інтерфейс реєстрації лікаря показаний на рисунку 3.18. Цей процес дуже простий і може бути завершений за 20 секунд. Користувачеві навіть не потрібно вибирати аватар для лікаря. Цю форму може заповнити пацієнт або особисто лікар.



Рисунок 3.18 – Інтерфейс реєстрації лікаря

Схема класу з екраном відомостей про лікаря, екраном даних про користувача, списком облікових записів та класом, який вони поширюють, показана на рисунку 3.19. Існує також діалогове вікно захворювання, яке

з'являється, коли користувач натискає кнопку вибору захворювання в процесі реєстрації.



Рисунок 3.19 – Діаграма класів діяльності зі списку аккаунтів та діяльності з деталями аккаунтів

3.9 Головні функції системи

Як і будь-яке інше типове програмне забезпечення, ця програма складається з різних програмних модулів - класів. Буде описано кілька основних модулів: системи рекомендацій та нагадування про ліки.

3.9.1 Нагадування про ліки

Користувачі можуть не лише встановити щоденне нагадування про споживання ліків, але також можна встановити кількість необхідних порцій певного ліки. Ця процедура буде виглядати так: користувач відкриває застосунок, розташований у телефоні, і вибирає вкладку з розкладами. За замовчуванням доступні 3 варіанти: час сніданку (9:00), обідній час (13:00) та час вечері (21:00). Цей список відображено на рисунку 3.20. Можливо також створення нового часу. Це продемонстровано на рисунку 3.21. Користувач вибирає, які ліки слід приймати і коли. Користувачі також можуть вказати, скільки таблеток потрібно прийняти. Спливаюче сповіщення зі звуком з'явиться одночасно в застосунку, розташованому на телефоні, і в застосунку, який знаходиться на годиннику. Повідомлення повідомляє, які та скільки таблеток слід приймати. Для користувача будуть доступні наступні варіанти: користувач підтверджує, що ліки було прийнято, користувач відкладає на пізніший час (наприклад, відкладає на 10 хвилин) або користувач відмовляється від цього конкретного прийому.

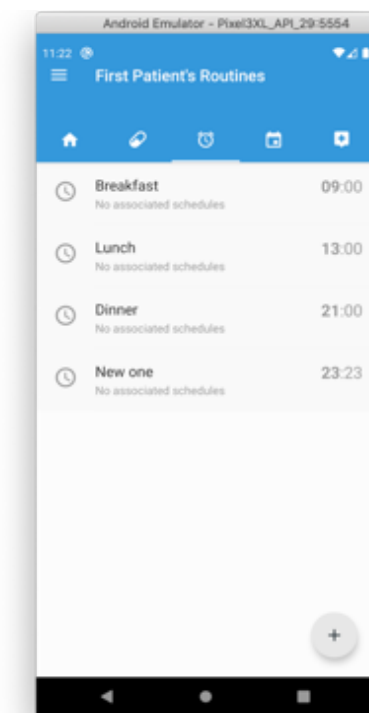


Рисунок 3.20 – Процес створення нагадування про нові ліки

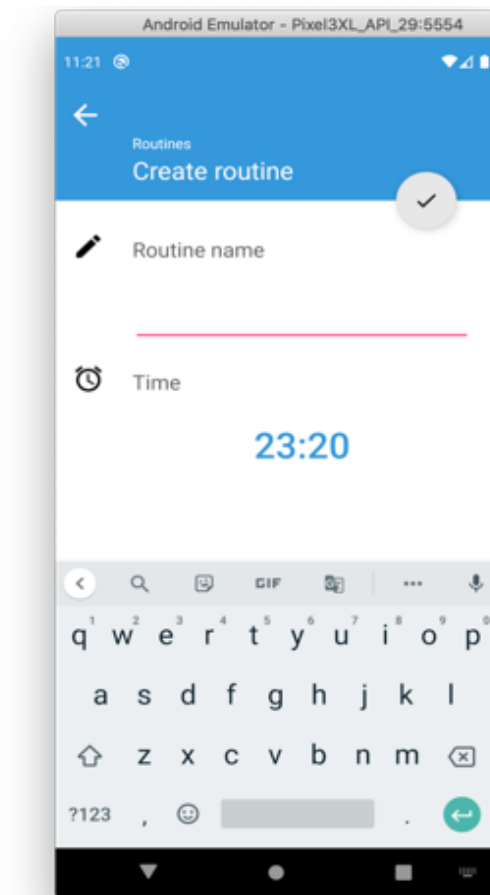


Рисунок 3.21 – Список нагадувань

3.9.2 Система рекомендацій

Система рекомендацій є однією з ключових особливостей програми. Майже всі процеси, що відбуваються в застосунку, покликані забезпечити систему рекомендацій достатньою кількістю даних для прийняття рішення про те, яку нову корисну інформацію пацієнт точно хоче отримати. Приклад рекомендацій для “Користувача за замовчуванням” показано на рисунку 3.22.

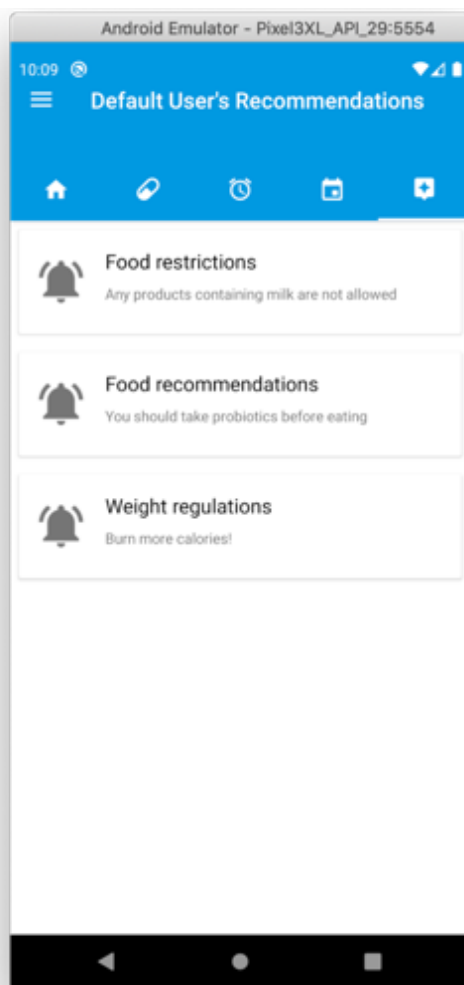


Рисунок 3.22 – Екран із доступними рекомендаціями

Якщо користувач може бачити рекомендації на вкладці рекомендацій на головному екрані, це означає, що він правильно зареєструвався в програмі, і його здоров'я чи дієта потребують поліпшення. У випадку, якщо користувач не завершив “процес реєстрації”, програмне забезпечення не зможе розрахувати рекомендації через відсутність даних, що стосуються особливо цього користувача.

Рисунок 3.23 є прикладом, який показує, що процес реєстрації пройшов неправильно для цього пацієнта.

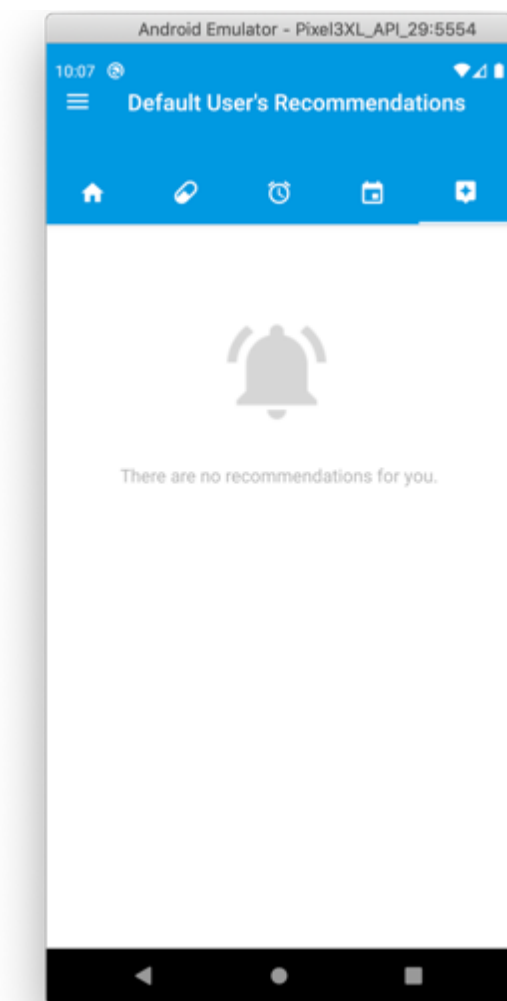


Рисунок 3.23 – Порожній екран рекомендацій

3.10 Дизайн бази даних

У цьому застосунку база даних використовується для зберігання всіх даних користувачів та лікаря. За замовчуванням, коли створюється новий користувач або лікар, у кожному полі реєстраційної форми відображаються зразки даних.

База даних SQLite використовується для зберігання всіх даних про користувачів та лікаря. Структура баз даних та класів, які вони поширюють, показана на рисунку 3.24. Також показані всі методи, з яких складається база даних.

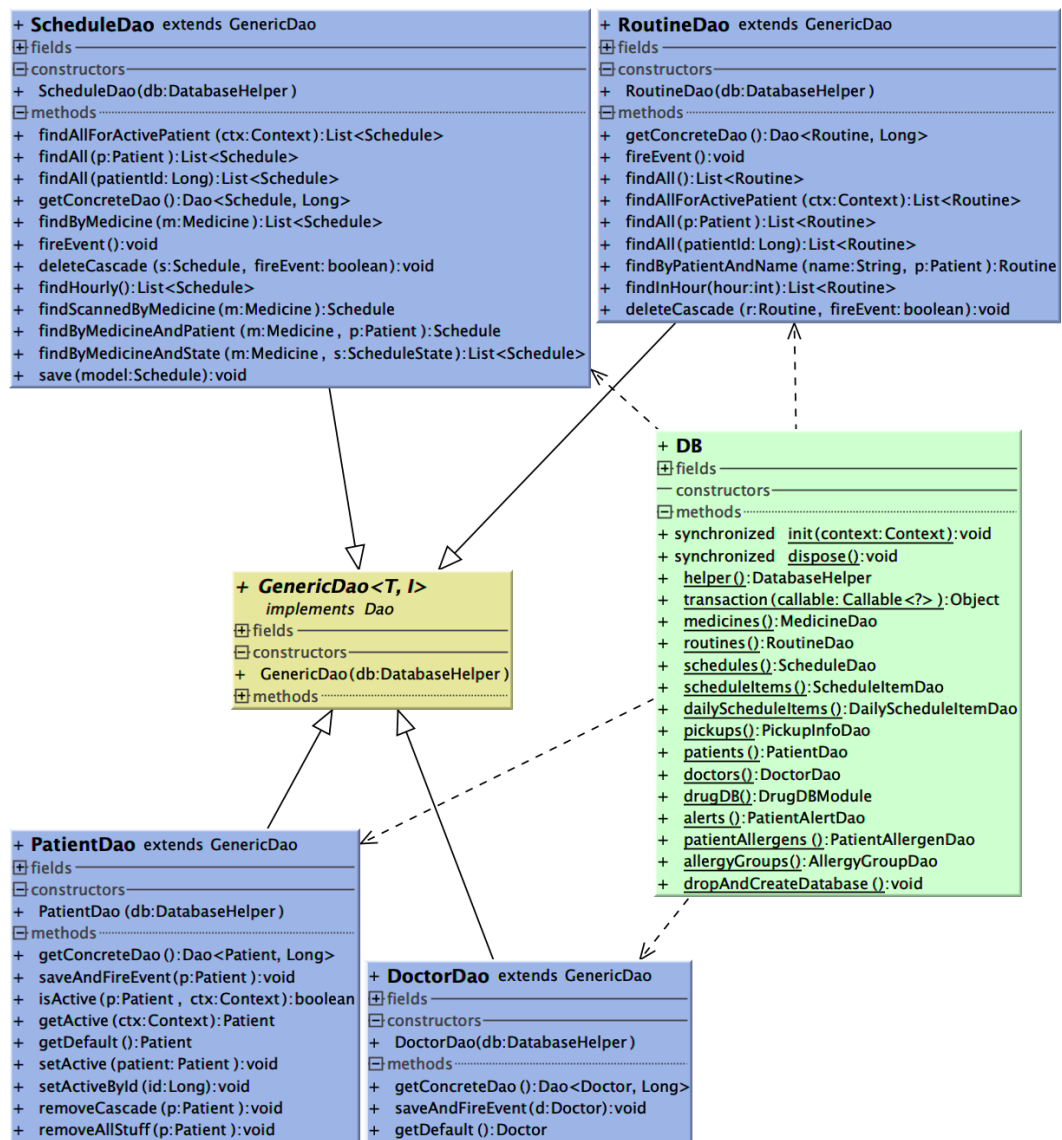


Рисунок 3.24 – Діаграма класів структури бази даних та полів бази даних

3.11 Майбутня робота

Загальне завдання майбутньої роботи - перевірити стан людей похилого віку та полегшити їхнє життя. Завдання полягає у розробці системи, придатної для всіх класів користувачів. Незважаючи на те, що система є повною і готова до використання, існують потенційні напрямки розвитку, які забезпечують більш доступну та зручну систему. Деякі напрямки вдосконалення обговорюються наступним чином.

З метою профілактики фізичних проблем у майбутньому та забезпечення загальної фізичної форми, лікарі рекомендують людям похилого

віку регулярно гуляти. Оскільки ідеальною метою були люди похилого віку, гіпотетична структура була використана для того, щоб запропонувати дії в залежності від наявних технологій. За допомогою датчиків система розрахує та визначить, які дії переважніші для пацієнта на даний момент.

3.11.1 Облікові записи та реєстрація

Аналізуючи особливості, які можуть знадобитися пацієнтам у майбутньому, логічно припустити, що функції облікового запису лікаря потрібно розширити таким чином, щоб лікар міг додавати лікарські засоби та процедури до пацієнта в обліковому записі лікаря.

Крім того, деякі поля можна додати до процесу реєстрації пацієнтів або лікаря, щоб мати можливість зібрати більше інформації, щоб обробити її таким чином, щоб програма могла запропонувати більше корисних функцій. Наприклад, було б корисно, якщо б лікар міг завантажити або взяти в застосунку свою справжню фотографію та додати її до профілю.

3.11.2 Екстрене повідомлення

За допомогою простих кнопок на корпусі або на екрані користувач може надсилати сповіщення на заздалегідь визначений контактний номер державної служби або родича.

У наш час більшість людей похилого віку вважають складним використання розумних пристроїв. Враховуючи це, ми розробили спрощену систему, коли годинники надсилатимуть попереджувальне повідомлення після натискання кнопки, ця дія відбуватиметься одночасно з даними, взятими з модуля GPS, тому дані про події будуть записані і міститимуть дату, час, та місцезнаходження. Номер контакту та рядок попереджувального повідомлення можна зберегти в пам'яті пристрою.

3.11.3 Рекомендації на основі місцезнаходження

Основна ідея цієї послуги - рекомендувати користувачеві місця, які можуть бути привабливими. Не зручно рекомендувати місця, розташовані за тисячі кілометрів від користувача. Отже, виходячи з цього припущення, необхідна реалізація системи, яка може розпізнавати місцезнаходження користувача.

Передбачається, що сповіщення, що містять рекомендації щодо розташування, будуть відображатися на одній вкладці з усіма іншими рекомендаціями, але міститимуть додатковий текст, наприклад, посилання на місце, яке рекомендується.

3.11.4 Система досягнень

Ідея впровадження цієї конкретної системи полягає в мотивації користувачів. Якщо програма зможе надсилати сповіщення користувачеві про покращення попередніх результатів, вона може спонукати користувача до дій, які, ймовірно, покращать стан здоров'я пацієнта. Якщо застосунок вирішить, що зайва активність шкодить здоров'ю пацієнта, тоді вона не призведе до такої категорії «досягнень». Наприклад, якщо пацієнт має будь-які обмеження щодо ніг - програма визнає це ситуацією, в якій вона не повинна надавати користувачеві "цілі" щодо ходьби або бігу.

3.11.5 Гео-зони

Визначення місця перебування людей похилого віку в реальному часі буде відстежуватися за допомогою GPS. Система відстеження місцеположення може бути використана для реалізації гео-зон для певної місцевості. Система може використовувати телефонний модуль GSM для забезпечення зв'язку та передачі даних через мобільні мережі. Паралельно модуль також буде допомагати у відстеженні місцезнаходження та гео-

зонуванні. GSM - це такий пристрій, який підтримує дзвінки, послуги SMS, інтернет-послуги та доступний на кожному смартфоні [20].

3.12 Висновки до розділу 3

У цьому розділі описано та обґрунтовано основні рішення щодо функціонування мобільного застосунку для перегляду екранів застосунку. Розроблено функціональну схему програми. Детально описано основні етапи роботи алгоритму програми.

Також описані використані у проєкті бібліотеки для створення програмного продукту. До цього ж була описана взаємодія застосунку з базою даних.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови використання мобільного застосунку

Розроблений мобільний застосунок для надання рекомендацій на основі аналізу даних користувачів, та забезпечення відповідності усім вимогам людей похилого віку.

Він дозволяє користувачеві:

- отримати зручний спосіб контролювати прийом ліків;
- переглянути персональні рекомендації та проаналізувати їх;
- переглянути контактні дані свого лікаря;
- контролювати кількість та тип наявних ліків.

Для роботи з даним застосунком необхідно мати мобільний пристрій, що функціонує під управлінням ОС Android версії 5.0 або вище, і має 80Мб вільного простору у внутрішньому сховищі. Але рекомендованою є версія ОС Android не нижча за версію 8.0, адже нижчі версії винятково не підтримують деяких елементів інтерфейсу користувача.

Застосунок розроблений за допомогою мов програмування Java та Kotlin середовища розробки Android Studio.

4.2 Характеристики мобільного застосунку

Час роботи застосунку залежить від конфігурації мобільного пристрою, на якому він виконується. Під час роботи застосунок виводить рекомендації на основі контексту користувача, надає регулярні нагадування щодо прийняття ліків, дозволяє створювати нові рутини та контролює залишки ліків.

Оскільки середовищем розробки було обране середовище Android Studio, то проєкт має модульну архітектуру, яка складається з наступних екранів: екран рутин, екран надання рекомендацій, екран розкладів, екран

керування обліковими записами, екран туру програми, головний екран, екран налаштувань, екран вибору баз даних алергій, екран вибору баз даних ліків.

Застосунок може бути запущеним паралельно з іншими Android-застосунками, адже він підтримує режим з паралельним відображенням двох застосунків.

4.3 Звертання до мобільного застосунку

Для того щоб запустити даний застосунок, треба встановити його на мобільний пристрій, виконавши відповідний файл з розширенням .apk. Також є можливість публікації програмного продукту у магазин застосунків «Play Market».

Завантажити застосунок Calendula можна натиснувши на його піктограму в списку застосунків.

4.4 Вхідні та вихідні дані

Вхідні дані - це дані або параметри, отримані від введення даних користувачем, а у майбутньому й з датчиків пристроїв. Мобільний застосунок також пропонує користувачеві використовувати функції, щодо ліків, які слід поповнити або взяти.

Вхідні дані надходять із двох різних джерел: мобільного застосунка та у майбутньому, зі допоміжного пристрою. Список усіх входів показаний на рисунку 4.1.

Вихідними даними є сповіщення про зміни в профілі пацієнта або створені екстрені телефонні дзвінки або надіслані SMS. Також є звукові виходи, які йдуть із сповіщеннями. Список усіх виходів показаний на рисунку 4.1.

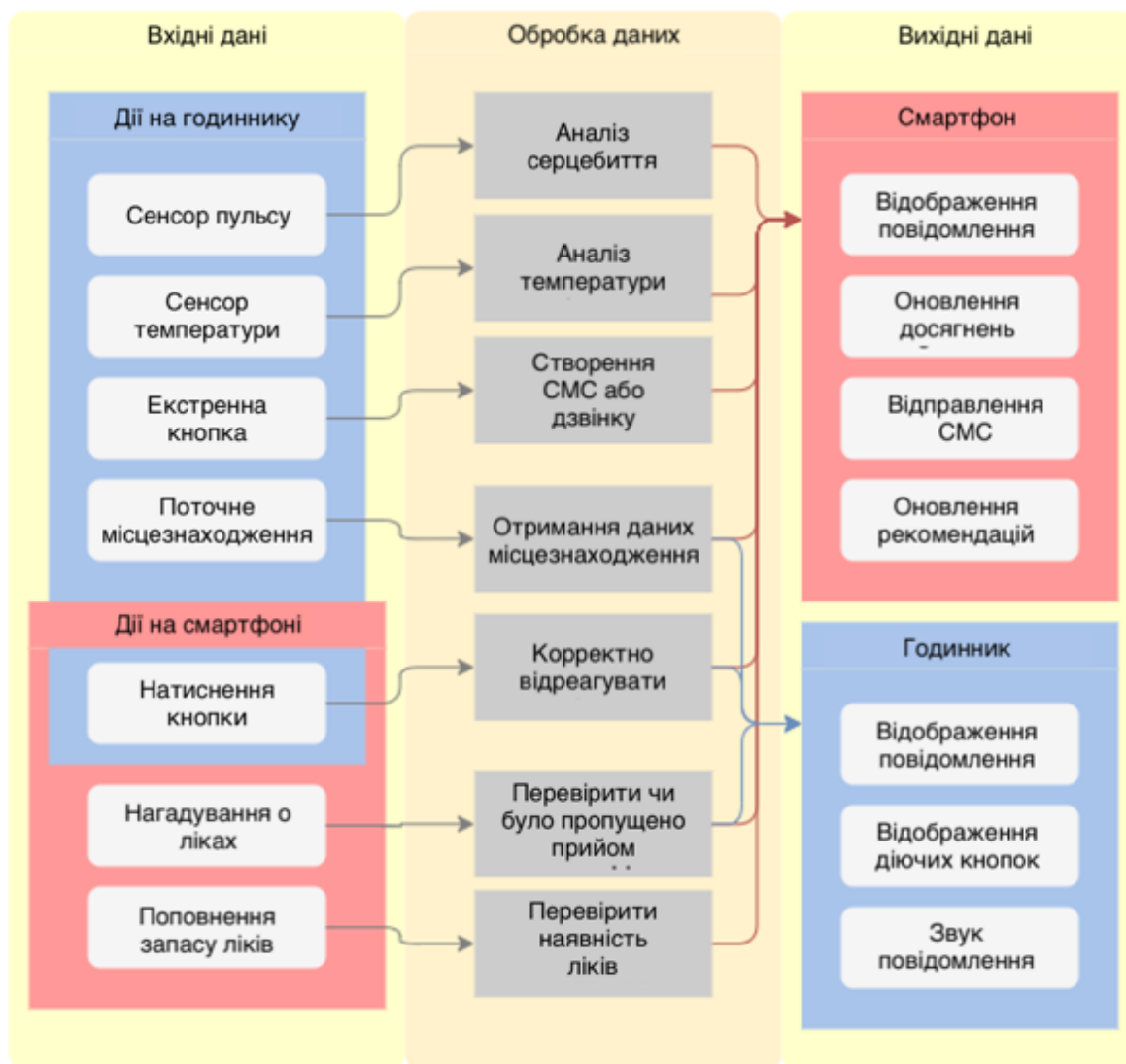


Рисунок 4.1 – Вхідні дані, обробка та вихідні дані системи

4.5 Повідомлення та обробка помилкових ситуацій

Розроблений застосунок гарантує його стійке функціонування. Виняткові ситуації можуть виникати при введенні неправильних символів у поле для введення захворювань користувача.

У зв'язку з цим у мобільному застосунку натискання на кнопку для вибору захворювання відкриває лист доступних захворювань, що дозволяє ввести тільки існуючу хворобу. Але навіть якщо це поле якимось чином сфальсифікувати – програма просто не надасть рекомендації щодо неіснуючої хвороби.

Це означає, що даний мобільний застосунок має широкий функціонал, і в той же час пред'являє достатньо невисокі вимоги до апаратного забезпечення, адже близько 95% усіх користувачів мають пристрій, що працює на ОС Android принаймні версії 5.0 і близько 60% мають пристрій з ОС Android версії 8.0 або вище. Також він забезпечує стійку роботу і обробку помилкових ситуацій.

4.6 Висновки до розділу 4

У ході роботи над розділом було описано розроблений програмний комплекс. Також детально описано взаємодію між реалізованою системою для надання рекомендацій людям похилого віку та іншим користувачам на основі контексту та програмістом. Даний розділ дасть змогу програмісту розібратися до дрібниць із принципом як даний застосунок було реалізовано, що надасть йому змогу успішно модифікувати та налаштувати розроблений застосунок.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення мобільного застосунку

Розроблений мобільний застосунок для надання рекомендацій на основі аналізу даних користувачів, та забезпечення відповідності усім вимогам людей похилого віку.

Він дозволяє користувачеві:

- отримати зручний спосіб контролювати прийом ліків;
- переглянути персональні рекомендації та проаналізувати їх;
- переглянути контактні дані свого лікаря;
- контролювати кількість та тип наявних ліків.

5.2 Умови використання мобільного застосунку

Для роботи з даним застосунком необхідно мати мобільний пристрій, що функціонує під управлінням ОС Android версії 5.0 або вище, і має 80Мб вільного простору на внутрішньому сховищі. Але рекомендованою є версія ОС Android не нижча за 8.0, адже нижчі версії не підтримують деяких елементів інтерфейсу користувача.

5.3 Виконання мобільного застосунку

Для того щоб запустити даний застосунок, треба встановити його на мобільний пристрій, виконавши відповідний файл з розширенням .apk. Також є можливість публікації програмного продукту у магазин застосунків «Play Market».

Завантажити застосунок Calendula можна натиснувши на його піктограму в списку застосунків.

Для того щоб знайти потрібний профіль користувача треба на висувному меню обрати пункт «Облікові записи», та обрати профіль. При

виборі одного з записів на головному екрані відображається інформація, що стосується лише обраного користувача.

Якщо облікового запису, що цікавить користувача, немає серед наведених у списку, він може натиснути на кнопку «Створити новий запис». Після натискання відобразиться форма реєстрації, де відображатимуться всі поля які буде необхідно заповнити щоб мати змогу отримувати рекомендації.

Для навігації по застосунку можна користуватися переважно висувним меню. Воно надає можливість перейти до основних екранів програми.

5.4 Приклад роботи з мобільним застосунком

Після запуску мобільного застосунку Calendula користувач бачить головний екран (рис. 5.1), на якому користувач має змогу знайти поточні прийняття ліків, та пустий екран якщо людина не приймає ліків на регулярній основі.

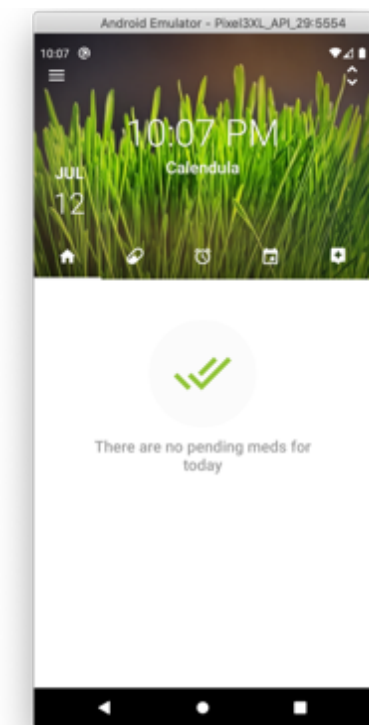


Рисунок 5.1 – Головний екран

Цей екран надає користувачеві наступні можливості:

- побачити наявні рутини прийняття ліків;
- натискання на один з елементів відкриває екран перегляду та налаштування відповідної рутини;
- кнопка підтвердження прийняття.

Екран реєстрації користувача (рис 5.2) складається з полей, які потрібно заповнити даними користувача. Форма складається з наступних полів:

- повне ім'я користувача;
- домашня адреса;
- адреса електронної пошти;
- зріст користувача;
- вага користувача;
- вік користувача;
- захворювання;
- маркер створення рутин за замовченням.

Слід зауважити, що вибір захворювання здійснюється за допомогою натиснення на кнопку вибору, за допомогою якої викликається список доступних для обрання захворювань, також у цьому списку є пункт що захворювань немає.

Завдяки функціоналу екрану облікових записів (рис 5.3) користувач отримує наступні можливості:

- обрати свій обліковий запис;
- створити новий обліковий запис зі своїми даними;
- переглянути обліковий запис лікаря та модифікувати його.

Для переходу до головного екрану достатньо натиснути на кнопку повернення.

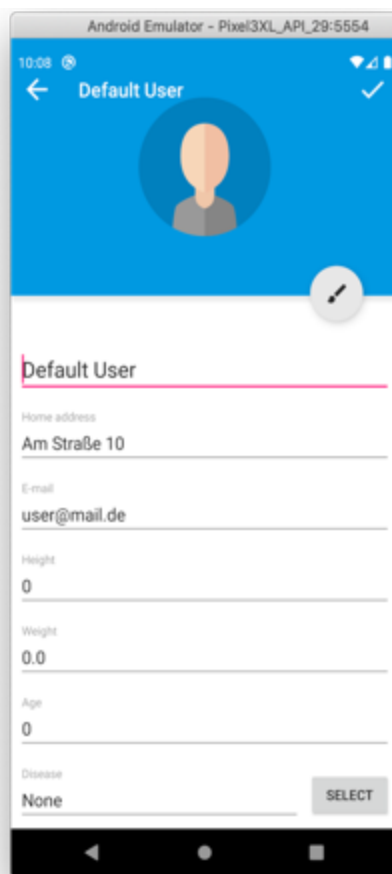


Рисунок 5.2 – Экран реєстрації користувача

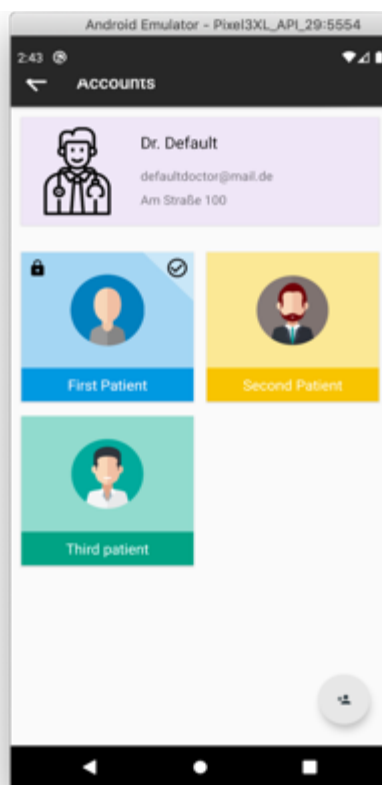


Рисунок 5.3 – Экран облікових записів

Екран перегляду рекомендацій (рис 5.4) відображає користувачеві повідомлення щодо його рекомендованих обмежень щодо їжі, ваги та медикаментів, що вживаються.

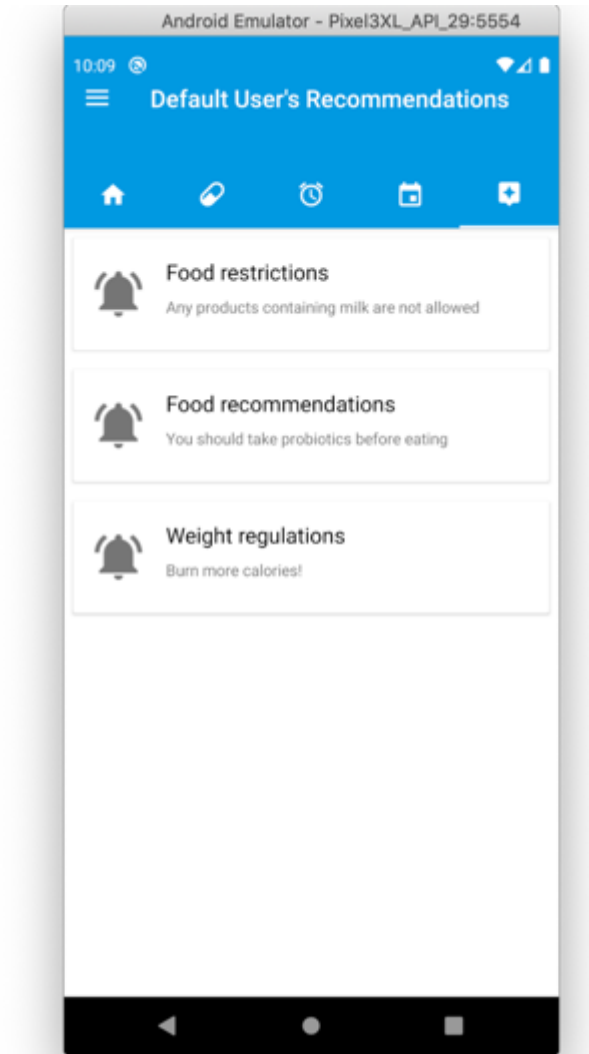


Рисунок 5.4 – Екран рекомендацій

У середній частині інтерфейсу користувача присутня панель навігації. За замовчуванням користувач перебуває у першому пункті. Натиснувши на пункт «Рекомендації», користувач потрапляє до екрану перегляду рекомендацій.

Програма має зручний, дружній інтерфейс, який дозволяє недосвідченому користувачу з легкістю розібратися з можливостями даного програмного забезпечення.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

Даний розділ присвячено для приведення економічного обґрунтування створення ПЗ, оцінка затрат на його створення, розрахунок економічної ефективності та терміну окупності [21].

6.1 Концепція економічного обґрунтування

Метою даної роботи є розробка алгоритму та програмного продукту, що реалізує систему надання рекомендацій користувачам. Основним завданням є створення зручного у використанні застосунку для забезпечення надання рекомендацій на основі контекстного методу людям похилого віку або осіб які того потребують.

Кількість людей похилого віку у світі швидко зростає. З цієї причини необхідно мати більше інструментів для догляду, розваг та взаємодії цієї частини населення. У цьому документі представлена архітектура, яка буде використовувати IOTCARS („Система рекомендацій, що базується на контексті для покращення якості життя”, це система контролю моніторингу здоров’я, зосереджена на загальному добробуті людей похилого віку.

Система надає декілька функціональних можливостей, таких як носяться пристосування для моніторингу загального стану здоров’я людини та прикладне програмне забезпечення Android, яке буде задіяне у догляді за людьми похилого віку. Носимий пристрій має збирати інформацію за допомогою датчиків і передавати її в основний застосунок на мобільному телефоні. Більш того, він складається з деяких загальних функцій, таких як нагадування про ліки, рекомендовані та заборонені: ліки, їжа, напої, та у майбутньому - локатор найближчої аптеки. За допомогою модуля GPS користувач зможе отримувати рекомендації щодо місць для відвідування відповідно до місцезнаходження.

Метою цієї платформи є комплексне забезпечення набору розумних послуг, що покращують якість життя та соціальну інтеграцію людей похилого віку.

Якщо розглядати конкурентів, то слід відзначити найбільш популярного конкурента на сьогоднішній день – застосунок Samsung Health. Це сучасне комплексне рішення, але воно не покриває усіх потреб людей літнього віку. Таким чином при виході на ринок прямих конкурентів на сьогоднішній день не існує.

6.2 Планування розробки програмного продукту

Для розрахунку витрат необхідно визначити етапи робіт та їх тривалість. У розробці беруть участь програміст протягом 3 місяців та консультант протягом 0,15 місяця. Розробка ПЗ починається тридцять першого липня та повинна бути закінчена до двадцять п'ятого листопада 2020 року.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, людей.

Нижче, у таблиці 6.1 наведена трудомісткість проєктування розроблюваної системи.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування роботи	Трудомісткість		Виконавці	Тривалість, дні
	люд.- дні	% до підсумку		
Аналіз предметної області	4	6,15%	Програміст	4
Узгодження результатів аналізу	4	6,15%	Програміст Консультант	2
Визначення вимог до програмного продукту	2	3,08%	Програміст	2
Узгодження вимог	3	4,61%	Програміст Консультант	3
Документування вимог	5	7,69%	Програміст	5
Проектування структури програми	13	20%	Програміст	13
Розробка схеми функціонування програми	7	10,77%	Програміст	7
Створення програмного коду	17	26,15%	Програміст	17
Тестування та налагодження програмного продукту	2	3,08%	Програміст	2
Складання програмної документації	8	12,3%	Програміст	8
Підсумок	65	100		63

За даними таблиці 6.1 складається зведений стрічковий графік розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення строків початку виконання всі види роботи, а навпаки – календарний період їх виконання. Даний графік наведений на рисунку 6.1. Також слід ввести додаткові позначення, що наведені у нижній частині рисунку 6.1.



Рисунок 6.1 – Зведений стрічковий графік планування розробки програмного продукту

6.3 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих у розробці програми. Розрахунок зарплати ведеться на підставі даних про трудомісткість, поданих у таблиці 6.2.

Таблиця 6.2 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	14000	3	42000
Консультант	1	17000	0,15	2550
Підсумок	2			44550

6.4 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою (6.2):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1. \quad (6.2)$$

У нашому випадку отримаємо:

$$ЗП_{\text{дод}} = 44550 \times 0,1 = 4455 \text{ грн.}$$

6.5 Єдиний соціальний внесок

Єдиний соціальний внесок становить 22% і береться від основної та додаткової заробітної плати (формула 6.3):

$$ВТ_{\text{соц}} = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22. \quad (6.3)$$

Далі отримаємо:

$$ВТ_{\text{соц}} = (44550 + 4455) \times 0,22 = 10781,10 \text{ грн.}$$

6.6 Визначення витрат на матеріали

Використовується 3 найменування матеріалів: диск DVD+RW – 15 грн; картридж – 100 грн, і папір – 100 грн (1 упаковка).

Витрати на матеріали вираховуються за наступною формулою (6.4):

$$ВТ_{\text{мат}} = \sum_{i=1}^n (Ц_i \cdot N_i \cdot (1 + K_{\text{т.з}}) - Ц_{io} \cdot N_{io}), \quad (6.4)$$

де $BT_{\text{мат}}$ – це витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

$K_{\text{т.з}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн;

N_i – потреба в i -му матеріалі, напівфабрикати і комплектуючому;

C_{io} – ціна зворотних відходів i -го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

$$C_{io} = 0; N_{io} = 0; K_{\text{т.з}} = 0,05.$$

$$BT_{\text{мат}} = (1 + 0,05) \cdot (15 + 100 + 100) = 225,75 \text{ грн.}$$

Разом, витрати на матеріали становлять 225,75 грн.

6.7 Витрати на спеціальне обладнання

Основною робочою машиною є персональний комп'ютер в наступній конфігурації: Macbook Pro, Intel Core i5, 8 ГБ RAM, 256 ГБ SSD, Intel Iris Plus. Виходячи з даних, наданих виробником обладнання, а так само знятими системними показниками можна припустити, що середнє енергоспоживання персонального комп'ютера при подібній проектній роботі складає 60 Ватт. Початкова ціна робочої машини – 55000 грн.

Амортизаційні відрахування визначають за формулою (6.5) [22]:

$$A = \Phi_6 \cdot \frac{H_a}{100}, \quad (6.5)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, %.

Норма амортизації для ПК становить 25%.

Балансова вартість обчислювальної техніки становить:

$$\Phi_6 = 20000 \text{ грн.}$$

Тоді отримаємо амортизаційні відрахування:

$$A = 20000 \times 0,25 = 5000 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію та допоміжні матеріали, формула (6.6):

$$C_e = N_n \cdot \Phi_{\text{еф}} \cdot K_{\text{зч}} \cdot K_{\text{зп}} \cdot C_{\text{ел.е}}, \quad (6.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{\text{зч}}$ – середній коефіцієнт завантаження по часу;

$K_{\text{зп}}$ – коефіцієнт завантаження по потужності;

$C_{\text{ел.е}}$ – ціна одного кВт·год електроенергії, грн/(кВт·год).

Номінальна потужність робочої машини – 0,06 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і по потужності рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,80 грн.

Отримуємо ціну експлуатації обладнання:

$$C_e = 0,06 \times 1800 \times 0,9 \times 0,6 \times 2,80 = 163,30 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.7):

$$ЗП_{\text{обсл}} = ФЗП_{\text{р}} \cdot (1 + K_{\text{відр}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.7)$$

де $ФЗП_{\text{р}}$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн;

$K_{\text{відр}}$ – коефіцієнт, що враховує відрахування на соціальне страхування й в інші фонди;

$t_{\text{обсл}}$ – час протягом року, необхідне на технічне обслуговування ЕОМ, год/рік;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік.

Місячна заробітна плата обслуговуючого персоналу становить 5100 грн., а річний фонд заробітної плати відповідно дорівнює 61200 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год/рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що в рік становить 12 годин.

Отримуємо:

$$ЗП_{\text{обсл}} = 61200 \times (1 + 0,22) \times \frac{12}{1750} = 511,98 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і становить 600 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 6275,28 грн., 5% від суми становлять 313.76 грн.

Розраховані статті витрат на утримання та експлуатацію обладнання внесено до таблиці 6.3.

Таблиця 6.3 – Кошторис витрат на утримання та експлуатацію обладнання.

Найменування статей витрат	Сума, грн.
Амортизація обладнання	5000
Експлуатація обладнання (крім витрат на поточний ремонт)	163,30
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	511,98
Поточний ремонт обладнання	600
Інші витрати	313,76
Разом	6589,04

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{\text{мо}} = B_{\text{екс}} \cdot t_{\text{мо}}, \quad (6.8)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн;

$B_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн/машино-год;

$t_{\text{мо}}$ – машинний час цифровий ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію устаткування (ЕОМ)» (табл. 6.4) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$B_{\text{екс}} = \frac{6589,04}{1800} = 3,67 \text{ грн / машино-год}$$

ЕОМ експлуатується 63 дні в одну зміну, що становить в сумі 504 годин. Таким чином, витрати на оплату машинного часу складуть

$$C_{mo} = 3,67 \times 504 = 1849,68 \text{ грн.}$$

6.8 Інші прямі витрати

До інших прямих витрат включаються витрати на використовуване при розробці системи комерційне ПЗ:

- пайове ПЗ, що використовується постійно при роботі ПК (Mac OS) – 0 грн. без ПДВ;
- цільове ПЗ, що для даної конкретної задачі (Microsoft Office) – 2600 грн. без ПДВ.

$$S_{\text{пай.ПО}} = \frac{C_{\text{ПОMacOS}} \times T_{\text{КТС}}}{\Phi_{\text{еф.КТС}} \times T_{\text{с.ПО}}}, \quad (6.9)$$

$$S_{\text{цел.ПО}} = C_{\text{ПОMSOffice}}, \quad (6.10)$$

де $S_{\text{пай.ПО}}$ – витрати на пайове ПЗ при розробці ПЗ з розрахунків, грн;

$S_{\text{цел.ПО}}$ – витрати на цільове ПЗ, що купується виключно для розробки ПЗ з розрахунків, грн;

$C_{\text{ПОMacOS}}$ – ціна за MacOS (без ПДВ), грн;

$C_{\text{ПОMSOffice}}$ – ціна за MS Office (без ПДВ), грн;

$T_{\text{КТС}}$ – машинний час КТС, необхідний користувачеві для розробки системи, машино-год/рік;

$\Phi_{\text{еф.КТС}}$ – річний ефективний фонд часу роботи КТС, машино-год/рік;

$T_{\text{с.ПО}}$ – термін служби пайового ПЗ, років.

$$S_{\text{пай.ПО}} = \frac{0 \times 504}{1800 \times 5} = 0$$

$$S_{\text{цел.ПО}} = 2600 + 0 = 2600$$

$$S_{\Sigma} = 0 + 2600 = 2600 \text{ грн.}$$

6.9 Розрахунок накладних витрат

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т. д.), утримання та експлуатацію будівель.

Накладні витрати включаються у вартість розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати складають 40% до основної заробітної плати розробників, що становить 17820 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведені в таблиці 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми.

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	44550	54,14
Додаткова заробітна плата	4455	5,41
Відрахування на соцстрах та ін.	10781,10	13,1
Матеріали та комплектуючі	225,75	0,27
Витрати на спец. обладнання	1849,68	2,24
Інші прямі витрати	2600	3,16
Накладні витрати	17820	21,68
Разом	82281,53	100
ПДВ	16456,31	
Разом (з ПДВ)	98737,84	

Таким чином, вартість проекту оцінюється в 98737,84 грн.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

З сучасним розвитком науки та техніки всі сфери виробництва, змінюють предмети праці, технології, якість, методи зберігання та обробки інформації. До того ж удосконалення безпечності та універсальності місця праці. Та на сьогоднішній день у багатьох сферах виробництва місце праці працівника залишається небезпечним або шкідливим для його здоров'я.

При роботі, в офісному та серверному приміщенні людину оточують: пристрої, обладнання зв'язку та інші засоби, що використовують досягнення радіоелектроніки, є генераторами цілої низки електромагнітних випромінювань. На сьогодні вплив на організм людини електромагнітного випромінювання ще не зовсім вивчений [23].

Методологічною основою охорони праці є системний підхід до вивчення організації праці з точки зору її безпеки, функціонування засобів взаємодії людини з середовищем, аналізу фізичних, хімічних, біологічних, психологічних та соціальних факторів безпеки виробничого процесу та їх особливостей впливу на організм.

Додержання правил щодо техніки безпеки і виробничої санітарії залежить не тільки від виконання власником або уповноваженим ним органом своїх обов'язків, а й від того, наскільки кожний працівник обізнаний і виконує їх під час роботи.

Програмний комплекс, що розробляється для розробки додатку з системою контекстних рекомендацій, передбачає роботу в обчислювальному центрі невеликої команди програмістів, що складається з 4 чоловік. Розміри приміщення $11 \times 5 \times 3.5 \text{ м}$.

Площа приміщення становить $11 \times 5 = 55 \text{ м}^2$, а об'єм – $11 \times 5 \times 3.5 = 192.5 \text{ м}^3$.

Небезпечні та шкідливі виробничі фактори за природою виникнення поділяються на такі групи:

- фізичні (підвищена і знижена температура повітря, надмірна загазованість та запиленість повітря та ін.);
- біологічні (підвищений вміст у повітрі мікроорганізмів та ін.), хімічні (вміст у робочій зоні шкідливих речовин та ін.);
- психофізіологічні (нервово-емоційні перевантаження, розумове напруження та ін.).

Також потенційна небезпеки може бути пов'язана з порушеннями правил пожежної безпеки, наприклад: проблеми в організації заходів з пожежної безпеки, що можуть призвести до швидкого розповсюдження пожежі.

У даному розділі розглянуто наступні питання:

- пожежна безпека;
- виробниче освітлення;
- безпечність технологічного обладнання та процесу;
- виробничі випромінювання;
- безпека щодо організації робочих місць;
- електробезпека;
- шум у робочій зоні;
- мікроклімат та склад повітря робочої зони.

7.2 Заходи із забезпечення безпеки

При роботі в обчислювальному центрі головним небезпечним фактором виступає електрика.

Приміщення обчислювального центру, за ступенем небезпеки ураження електричним струмом, відноситься до приміщень без підвищеної небезпеки.

Питання правильного електроживлення при об'єднанні комп'ютерів в локальні мережі мають важливе значення як для забезпечення електробезпеки користувачів, так і для безаварійної і безперебійної роботи обладнання.

Заходи щодо забезпечення електробезпеки розроблено відповідно до стандартів [23].

Обладнання, що використовується в приміщенні обчислювального центру є споживачем електроенергії, що живиться від змінного струму 220 В, частотою струму 50 Гц від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000В закритого виконання.

Комп'ютери мають власні блоки живлення, які перетворюють змінний струм мережі у постійний струм (5 В, 12 В), комп'ютери використовують блоки живлення потужністю 60 – 750 Вт.

Заради зменшення ймовірності ураження людини електричним струмом живлення всіх електричних пристроїв здійснюється спеціальним кабелем, корпуси всіх електричних пристроїв виготовляються з неструмопровідних матеріалів.

Забезпечення безпечності у приміщенні обумовлено тим, що всі кабелі сховані в спеціальні коробки, штепсельні розетки розташовані на висоті одного та 0,2 метри від підлоги, вимикачі на стінах знаходяться на висоті 1,2 метра від підлоги, електропроводка трифазна, чотирижильна, з подвійною ізоляцією, електрична мережа вмикається і вимикається за допомогою пускової апаратури (рубильником). Відповідно до міжнародного стандарту СЕ зазначені міри електробезпеки відповідають вимогам, пропонованим до побутових приладів.

Відповідно до стандартів [24] в приміщенні розроблено заходи щодо забезпечення електробезпеки.

Також для запобігання ураження людини електричним струмом в приміщенні використовуються:

- захисне заземлення;

- окремі розетки по 220 В для кожного робочого місця з обчислювальним пристроєм;
- триполюсні вилки та розетки, для підключення обчислювальних пристроїв до струму, у яких відповідний полюс з'єднаний з шиною заземлення;
- мережевий фільтр до якого підключений кожен блок живлення обчислювального пристрою або периферійного пристрою. Шунтування високочастотних перешкод живлячої мережі на землю через дріт захисного заземлення обумовлено конденсаторами;
- провід на загальний контор при з'єднанні пристроїв для ліквідування проблеми різниці потенціалів;
- гумова підлога, що надійно ізолює.

Згідно з "Правилами улаштування електроустановок" опір захисного заземлення в обчислювальному центрі становить 2.5 Ом (за стандартом – не більше 4 Ом). Згідно з стандартом [25] в приміщенні встановлено автоматичні вимикачі, пристрій заземлення контуру.

Втома та тунельний синдром також є розповсюдженим небезпечним фактором для здоров'я люди при роботі в обчислювальному центрі.

Для профілактики втоми робітників застосовуються специфічні методи, до яких можна віднести засоби відновлення функціонального стану зорового та опорно-рухового апарату, зменшення гіподинамії, підсилення мозкового кровообігу, оптимізацію розумової діяльності.

Для профілактики втоми та запобігання статистичного навантаження робітників встановлені такі правила організації робочого місця:

- оптимальна висота клавіатури від підлоги – 65-75 см;
- можливість регулювання висоти і нахилу клавіатури (відстань від поверхні стола до середини клавіатури – не більше 30 мм, кут підйому клавіатури – від 2 град до 15 град);
- наявність у клавіатури підставки для рук та килимка для миші з захистом від тунельного синдрому;

- наявність стільця або крісла з підлокітниками;
- рекомендовано використовувати перерви в роботі 5 хв. через кожні 40 хвилин та 15 хв. перерву кожні 2 години.

7.3 Заходи з забезпечення виробничої санітарії та гігієни праці

Заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного ПК з ВДТ розроблені відповідно до вимог Державних санітарних норм та правил розроблені заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного обчислювальними пристроями з ВДТ [26]-[28].

Так як робота програмістів в обчислювальному центрі виконуються сидячи і не потребують фізичного напруження, то вони відносять до типу операторської роботи, даний тип роботи відносить до категорії 1а. Робоче місце програміста є постійним, робота проводиться у холодний та теплий періоди року тому рекомендується підтримувати оптимальні показники мікроклімату.

Для забезпечення оптимального рівня параметрів повітряного виробничого середовища використовуємо стандарти і норми (табл. 7.1) [29]-[30].

Таблиця 7.1 – Норми мікроклімату в приміщенні.

Період Року	Категорія робіт	Температура повітря, °C	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодний	Легка	22–24	40–60	0,1
Теплий	Легка	23–25	40–60	0,1-0,2

Оптимальні рівні позитивних (n+) і негативних (n-) іонів у повітрі приміщення з ПК нормовані згідно зі стандартом [28] становлять: n+=1500-30000 (шт. на 1см³); n- = 3000-5000 (шт. на 1см³). Підтримка оптимального

рівня легких позитивних і негативних аероіонів у повітрі на робочих місцях забезпечено за допомогою біполярних коронних аероіонізаторів.

В обчислювальному центрі передбачено: устрій системи водяного опалення приміщення для забезпечення необхідної температури повітря в холодний період року відповідно до стандартів [31].

Проведені дослідження мікрокліматичних умов на комп'ютеризованих робочих місцях показали, що зимою значення відносної вологості повітря часто можуть нижчими за встановлені норми і становити в середньому 30–40%. Це приводить не лише до надмірного висихання слизових оболонок очей, носа, горла, а й до нагромадження зарядів статичної електрики, що утворюються в процесі роботи комп'ютера. Для усунення таких проблем в приміщенні передбачена наявність 4 зволожувачів повітря, які допомагають підняти вологість до рекомендованих 60% для усунення таких проблем. Кількість зволожувачів зумовлена тим, що один такий апарат розраховано на приміщення площею до 15 м².

У теплий період року для підтримки нормальної температури повітря в приміщенні використовуються три кондиціонери типу «спліт-система». Два кондиціонера розташовані у протилежних частинах та один по центру обчислювального центру дозволяють підтримувати температуру повітря не вище 24°C. Кількість кондиціонерів зумовлена тим, що один такий апарат розраховано на кондиціонування приміщення площею до 18 м².

Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація. Рівні звуку, рівні звукового тиску в октавних смугах частот, та еквівалентні рівні звуку на робочих місцях у приміщення нормуються згідно з стандартом [26]. Оптимальний рів шуму в приміщення забезпечується за допомогою:

- використання блоків живлення ПК з вентиляторами на гумових підвісках;
- використання більш сучасного обладнання;
- розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;
- відмова від жорстких дисків на користь твердотільних накопичувачів;
- використання пасивних систем охолодження ПК.

Вимоги до освітлення регламентовані ДБН В.2.5-28:2018 [32], згідно з якими освітлення у приміщеннях з ЕОМ суміщене, при якому недостатнє за нормами природне освітлення доповнене штучним. Природне освітлення бічне, одностороннє.

Згідно з вимогами регламентованими в ДБН В.2.5-28:2018 [32] В обчислювальному центрі коефіцієнт природної освітленості (КПО) має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду). Вікна в приміщенні обладнанні сонцезахисними металевими жалюзі з системою регульована для забезпечення відносної постійності природного освітлення. коефіцієнт відбиття жалюзі становить 0,7. Робочі місця робітників розташовано так, щоб у поле зору робітника не потрапляли світлі поверхні світильників та вікон.

Найбільш прийнятними для приміщень, де працюють ПК, є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (тепло-білого світла) потужністю 20, 40 або 80 Вт. Для освітлення приміщення розміром $11 \times 5 \times 3.5\text{ м}$ освітленістю $E_p = 400\text{ лк}$, коефіцієнтом відбиття стелі 70% і стін 50% розрахуємо кількість ламп необхідних для освітлення використовуючи люмінесцентні лампи типу ЛБ в світильниках ЛПО.

Далі наведено формулу (7.1), для знаходження індексу приміщення:

$$i = \frac{A \times B}{h \times (A + B)} = \frac{11 \times 5}{(3,5 - 0,7) \times (11 + 5)} = 0,98 \quad (7.1)$$

де A – довжина приміщення;

B – ширина приміщення;

H – висота приміщення.

Приймаємо коефіцієнт запасу $k = 1,6$ і коефіцієнт нерівномірності освітлення за формулою (7.2):

$$z = \frac{E_{cp}}{E_{min}} = 1.1 \quad (7.2)$$

При індексі $i = 0,98$ отримаємо коефіцієнт використання світлового потоку $\eta = 35\%$.

В два ряди ($N_p = 2$) розміщуємо світильники.

Використовуючи вище зазначені параметри розраховуємо необхідний світловий потік ламп в кожному ряду за формулою (7.3):

$$\Phi_p = \frac{E_n \times S \times z \times k}{N_p \times \eta} = \frac{400 \times 55 \times 1,1 \times 1,6}{2 \times 0,35} = 55314 \text{ лм} \quad (7.3)$$

Далі розраховуємо кількість необхідних світильників за формулою (7.4), якщо в світильники встановити по дві лампи ЛБ ($n = 2$) потужністю 65 Вт і світловим потоком $\Phi_n = 4500 \text{ лм}$:

$$N_p = \frac{\Phi_p}{n \times \Phi_n} = \frac{55314}{2 \times 4500} \approx 7 \quad (7.4)$$

Таким чином, для освітлення приміщення використовуємо 14 світильників, розташованих по 7 у кожному ряду.

Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури, використовуються сонцезахисні пристрої, на вікнах встановлені жалюзі або штори.

Організація та обладнання робочого місця в робочу приміщені має відповідати ергономічним вимогам з урахуванням особливостей трудовою дальності.

В приміщені обчислювального центру одночасно працюють 4 чоловіка, кожний робітник має свій:

- обчислювальний пристрій, пристрій вводу (клавіатура, миш та інше), виводу (дисплей, принтер, навушники та інше) та набір периферійних пристроїв (принтер, сканер та інше);
- робочу поверхню-стіл;
- ортопедичне крісло;
- тумбу.

Також кожний робітник може скористатися шафою для одягу, шафою з приладами, двома тумбами призначеними для використанню всіма робітниками.

Робоче місце працівника має площу 13,75 м², а об'єм – по 48,13 м³.

Приміщення має природне та штучне освітлення. Робочі місця розташовано так, щоб природне світло падало збоку. Робочі місця та взаємне розташування всіх його елементів відповідати антропометричним, фізичним і психологічним вимогам.

Також всі робочі місця організовані за такими правилами організації робочого місця:

- всім робітникам надані ергономічні та зручних миші і клавіатури;
- висота столу забезпечує розташування клавіатури від підлоги – 65-75 см, що позитивно впливає на працездатність робітника;
- кожна клавіатура має можливість міняти висоти і нахилу клавіатури;
- кожна клавіатура має підставку для рук;

- всі килимки для миші мають захист від тунельного синдрому (спеціальний виступ забезпечує правильне положення кисті);
- екрани монітору знаходяться на оптимальній відстані 700 мм від працівника;
- кожний стілець має підлокітник.

Загальний план обчислювального центру та робочих місць наведено на рисунку 7.1.

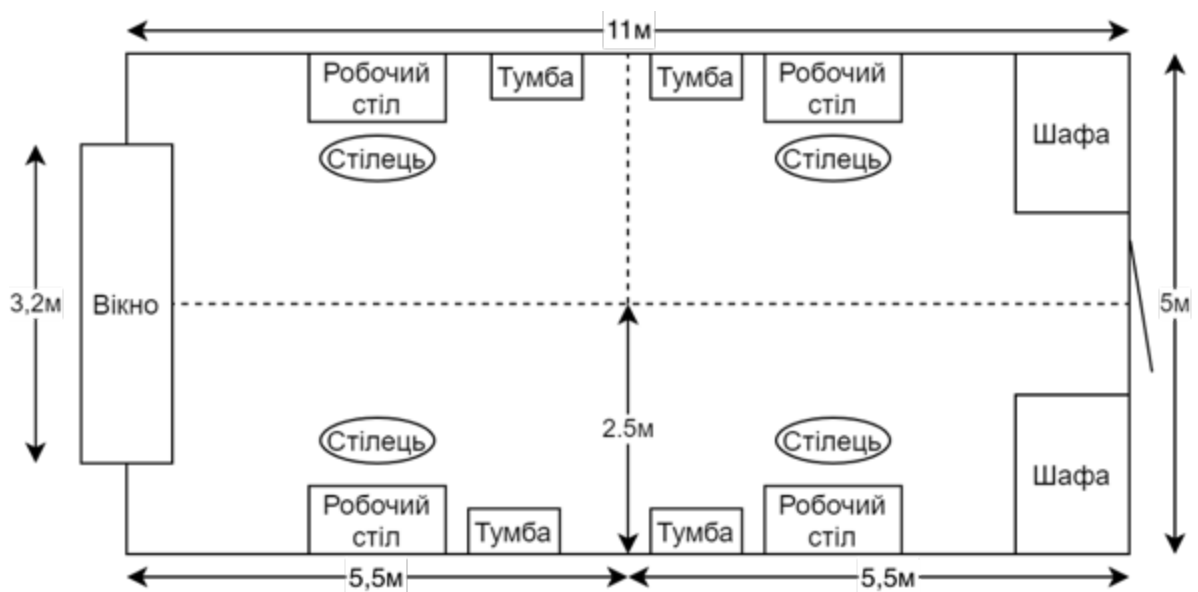


Рисунок 7.1 – Загальна схема обчислювального центру

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Пожежі в обчислювальному центрі становлять особливу небезпеку, тому що пов'язані з великими матеріальними втратами. Так як в приміщенні обчислювального центру є всі три основні чинники, необхідні для виникнення пожежі, а саме: горючі речовини, окислення і джерела запалювання.

До ймовірних причин виникнення пожежі можна віднести несправність електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з ладу чи експлуатації несправного електроустаткування, порушення правил протипожежної безпеки тощо.

Для забезпечення пожежної безпеки проводилась пожежна профілактика, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі.

Для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціонування. Тому кисень, як окислювач процесів горіння, є в усіх придатних місцях обчислювального центру.

Обчислювальний центр, згідно [32] відноситься до категорії «П-Па», а клас можливої пожежі, визначається, як «А».

Для ліквідації пожежі у початковій стадії її розвитку силами персоналу об'єктів застосовуються первинні засоби пожежогасіння. До засобів гасіння пожежі, призначених для локалізації невеликих загорянь, відносяться пожежні стовбури, внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри і т. п.

Так як приміщення (дослідницької лабораторії, конструкторського бюро, тощо) що обладнане ПК з ВДТ має площу 55 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» [33] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 3 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

Усі працівники обчислювального центру повинні проходити протипожежний інструктаж під час прийняття на роботу і в процесі праці, а також проходити перевірку знань з питань пожежної безпеки. Евакуаційні виходи і проходи утримуються постійно вільними, легкодоступними та не захаращуються.

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогне-димо-затримувальних пристроїв. Під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі.

Фільтри припливно-витяжної вентиляції слід чистити згідно з затвердженим графіком, але не рідше двох разів на рік. Роботи з ремонту вузлів (блоків) ЕОМ, як правило, проводити не дозволяється. Їх необхідно проводити в окремому приміщенні (майстерні). У разі необхідності проведення ремонту або технічного обслуговування ЕОМ безпосередньо в залі допускається мати не більше 0,5 л легкозаймистих рідин у небиткій, щільно закритій тарі. Для промивання деталей необхідно застосовувати, як правило, негорючі миючі препарати. Чарунки та інші знімні пристрої допускається промивати горючими рідинами тільки в спеціальному приміщенні, обладнаному припливно-витяжною вентиляцією. Відповідальний за пожежну безпеку приміщень обчислювального центру зобов'язаний негайно сповістити технічну службу у разі виявлення незадовільного технічного стану димових та теплових сповіщувачів автоматичної пожежної сигналізації, а також про несправності автоматичної установки об'ємного (газового) пожежогасіння.

Забороняється працівникам проводити ремонт вузлів ЕОМ. Ремонтні роботи необхідно проводити в окремому, спеціалізованому для ремонту приміщенні.

Для своєчасного попередження пожеж та підвищення оперативності реагування при їх виникненні у приміщенні використовується такий комплекс заходів та правил:

- обов'язковий інструктаж персоналу з питань охорони праці;
- зокрема, правила пожежної безпеки у приміщеннях з ЕОМ;

- наявність системи автоматичної пожежної сигналізації з димовими пожежними оповісниками;
- наявність шляхів евакуації при виникненні пожежі;
- заборона залишати без нагляду електричну апаратуру, що використовується;
- заборона використання відкритого вогню у приміщенні;
- заборона зберігати постійно в залах ЕОМ носії інформації, запасні блоки та деталі в кількості більшій, ніж необхідно для поточного використання;
- заборона користуватися груповими розетками на горючій панелі;
- заборона використовувати килими та доріжки із синтетичних матеріалів;
- заборона ставити на вікна глухі ґрати (в разі потреби встановлюються ґрати, що розкриваються зсередини);
- заборона застосовувати електронагрівальні побутові прилади;
- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

Усі працівники обчислювального центру повинні вміти привести в дію (в разі потреби) первинні засоби пожежогасіння. При виявленні ознак пожежі працівник повинен:

- негайно повідомити про це службу порятунку за номером телефону – 101 (112);
- вжити заходів щодо евакуації людей та матеріальних цінностей;
- вжити заходів до вжити заходів гасіння пожежі з використанням наявних вогнегасників та інших засобів пожежогасіння.

Керівник підрозділу при виникненні пожежі зобов'язаний:

- упевнитися що викликані рятувальні служби;
- відключити подачу електроенергії в приміщення;
- проінформувати про пожежу керівництво;

- перевірити чи людей оповіщено про пожежу;
- організувати евакуацію працівників (за умови загрози життю людей)
- забезпечити дотримання техніки безпеки працівниками, які беруть участь у гасінні пожежі;
- організувати зустріч підрозділів Державної пожежної охорони;
- надати допомогу підрозділам Державної пожежної охорони під час локалізації та ліквідації пожежі;
- забезпечити безперешкодний доступ до місця виникнення пожежі пожежно-рятувальним підрозділам.

7.4.2 Єдина державна система цивільного захисту, її складові та режими функціонування

Забезпечення реалізації державної політики у сфері цивільного захисту здійснюється єдиною державною системою цивільного захисту, яка складається з функціональних і територіальних підсистем та їх ланок.

Положення про єдину державну систему цивільного захисту, типові положення про функціональну і територіальну підсистеми затверджуються Кабінетом Міністрів України.

Основними завданнями єдиної державної системи цивільного захисту є:

- забезпечення готовності міністерств та інших центральних та місцевих органів виконавчої влади, органів місцевого самоврядування, підпорядкованих їм сил і засобів до дій, спрямованих на запобігання і реагування на надзвичайні ситуації;
- забезпечення реалізації заходів щодо запобігання виникненню надзвичайних ситуацій;
- навчання населення щодо поведінки та дій у разі виникнення надзвичайної ситуації;

- виконання державних цільових програм, спрямованих на запобігання надзвичайним ситуаціям, забезпечення сталого функціонування підприємств, установ та організацій, зменшення можливих матеріальних втрат;
- опрацювання інформації про надзвичайні ситуації, видання інформаційних матеріалів з питань захисту населення і територій від наслідків надзвичайних ситуацій;
- прогнозування і оцінка соціально-економічних наслідків надзвичайних ситуацій, визначення на основі прогнозу потреби в силах, засобах, матеріальних та фінансових ресурсах;
- створення, раціональне збереження і використання резерву матеріальних та фінансових ресурсів, необхідних для запобігання і реагування на надзвичайні ситуації;
- оповіщення населення про загрозу та виникнення надзвичайних ситуацій, своєчасне та достовірне інформування про фактичну обстановку і вжиті заходи;
- захист населення у разі виникнення надзвичайних ситуацій;
- проведення рятувальних та інших невідкладних робіт щодо ліквідації наслідків надзвичайних ситуацій, організація життєзабезпечення постраждалого населення;
- пом'якшення можливих наслідків надзвичайних ситуацій у разі їх виникнення;
- здійснення заходів щодо соціального захисту постраждалого населення;
- реалізація визначених законом прав у сфері захисту населення від наслідків надзвичайних ситуацій, в тому числі осіб (чи їх сімей), що брали безпосередню участь у ліквідації цих ситуацій;
- інші завдання, визначені законом.

Функціональні підсистеми єдиної державної системи цивільного захисту (далі - функціональні підсистеми) створюються центральними органами виконавчої влади у відповідній сфері суспільного життя.

Положення про функціональні підсистеми розробляються на підставі типового положення про таку підсистему і затверджуються центральними органами виконавчої влади, що їх створили, за погодженням із центральним органом виконавчої влади, який забезпечує формування та реалізує державну політику у сфері цивільного захисту.

У разі якщо діяльність центральних органів виконавчої влади спрямовується і координується Кабінетом Міністрів України через відповідного міністра, зазначені положення затверджуються такими міністрами за погодженням із центральним органом виконавчої влади, який забезпечує формування та реалізує державну політику у сфері цивільного захисту.

Перелік центральних органів виконавчої влади, що створюють функціональні підсистеми, визначається Положенням про єдину державну систему цивільного захисту.

Безпосереднє керівництво функціональною підсистемою покладається на керівника органу, суб'єкта господарювання, що створив таку підсистему.

До складу функціональних підсистем входять органи управління та підпорядковані їм сили цивільного захисту, відповідні суб'єкти господарювання, які виконують завдання цивільного захисту.

Єдина державна система залежно від масштабів і особливостей надзвичайної ситуації, що прогнозується або виникла, функціонує у режимах: повсякденного функціонування, підвищеної готовності, надзвичайної ситуації або надзвичайного стану.

Режим повсякденного функціонування єдиної державної системи цивільного захисту встановлюється за умов нормальної виробничо-промислової, радіаційної, хімічної, сейсмічної, гідрогеологічної,

гідрометеорологічної, техногенної та пожежної обстановки та за відсутності епідемій, епізоотій, епіфітотій.

Режим підвищеної готовності впроваджується у разі загрози виникнення надзвичайної ситуації за рішенням відповідно Кабінету Міністрів України, Ради міністрів Автономної Республіки Крим, обласних, Київської чи Севастопольської міських державних адміністрацій для єдиної державної системи цивільного захисту у повному обсязі або частково для окремих її територіальних підсистем тимчасово встановлюється режим підвищеної готовності.

Режим надзвичайної ситуації встановлюється у разі виникнення надзвичайної ситуації за рішенням відповідно Кабінету Міністрів України, Ради міністрів Автономної Республіки Крим, обласних, Київської чи Севастопольської міських державних адміністрацій для єдиної державної системи цивільного захисту у повному обсязі або частково для окремих її територіальних підсистем тимчасово встановлюється режим надзвичайної ситуації.

Режим надзвичайного стану для єдиної державної системи цивільного захисту у повному обсязі або частково для окремих її територіальних підсистем тимчасово встановлюється у межах території, на якій введено правовий режим надзвичайного стану відповідно до Закону України "Про правовий режим надзвичайного стану" [34].

ВИСНОВКИ

Система розроблена для інтеграції охорони здоров'я разом із функціональними можливостями моніторингу і, нарешті, для задоволення потреб користувачів.

Запропонована система розроблена з акцентом на ефективність та доступність. У цій системі користувачі без комп'ютера та Інтернету також можуть використовувати систему через портативний пристрій та застосунок для Android. Користувачі, які мають старі телефони Android, також зможуть завантажити та використовувати програму, оскільки мінімальні вимоги відповідають старим пристроям.

Середня вартість проживання для людей похилого віку вища з урахуванням загальних проблем здоров'я, плати лікаря, плати за медичне обстеження, витрат на ліки тощо. Більше того, людині важко одночасно доглядати за своїми літніми родичами та підтримувати роботу. Система була розроблена враховуючи всі ці чинники, для задоволення потреб користувачів.

За темою дипломної кваліфікаційної роботи магістра було опубліковано одні тези доповіді на X Міжнародній науково-практичній конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій» (м. Запоріжжя, 7-9 жовтня 2020 року).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. El apoyo familiar de las personas de edad, en Europa: contrastes e implicaciones [Electronic source]. – Access mode: https://repositorio.cepal.org/bitstream/handle/11362/12755/np77219249_es.pdf?sequence=1&isAllowed=y
2. Complex IoT Systems as Enablers for Smart Homes in a Smart City Vision [Electronic source]. – Access mode: <https://www.mdpi.com/1424-8220/16/11/1840/htm>
3. Pervasive Computing Research on Aging Disability and Independence [Electronic source]. – Access mode: [http://www.research.lancs.ac.uk/portal/en/publications/pervasive-computing-research-on-aging-disability-and-independence\(cd589958-5105-4c34-a7de-eda7e0dca2b0\).html](http://www.research.lancs.ac.uk/portal/en/publications/pervasive-computing-research-on-aging-disability-and-independence(cd589958-5105-4c34-a7de-eda7e0dca2b0).html)
4. Pervasive Social Computing: augmenting five facets of human intelligence [Electronic source]. – Access mode: <https://core.ac.uk/download/pdf/191447122.pdf>
5. Los nuevos roles de las ciudades inteligentes [Electronic source]. – Access mode: <https://www.uoc.edu/uocpapers/5/esp/editorial.html>
6. Smart Cities: un primer paso hacia la internet de las cosas [Electronic source]. – Access mode: <https://www.aeiciberseguridad.es/descargas/categoria6/9704170.pdf>
7. Integration of Cloud Computing and Internet of Things [Electronic source]. – Access mode: https://books.google.de/books?id=wCi-DwAAQBAJ&pg=PA370&lpg=PA370&dq=A.+Botta,+W.+De+Donato,+V.+Persico,+and+A.+Pescapè,+“Integration+of+cloud+computing+and+internet+of+things:+a+survey,”+Future+Generation+Computer+Systems&source=bl&ots=Lag3W_Cu0l&sig=ACfU3U1B4ZIIfKCcDErtq_d0NChlpGMiLQ&hl=en&sa=X&ved=2ahUKEwiNrPCZlsztAhWZBGMBHfqFBr4Q6AEwAHoECAEQAg#v=onepage&q

=A.%20Botta%2C%20W.%20De%20Donato%2C%20V.%20Persico%2C%20and
%20A.%20Pescape%20%2C%20“Integration%20of%20cloud%20computing%20
and%20internet%20of%20things%3A%20a%20survey%2C”%20Future%20Gener
ation%20Computer%20Systems&f=false

8. Learning Internet of Things Security “Hands-on” [Electronic source]. –
Access mode: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=919708

9. Android Studio [Electronic source]. – Access mode:
<https://developer.android.com/about>

10. SmartAssist - Wireless Sensor Networks for Unobtrusive Health
Monitoring [Electronic source]. – Access mode: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-678/BMI10-08.pdf>

11. Elderly people at home: technological help in everyday activities
[Electronic source]. – Access mode:
https://www.researchgate.net/publication/4177586_Elderly_people_at_home_Technological_help_in_everyday_activities

12. Analyzing the Elderly Users’ Adoption of Smart-Home Services
[Electronic source]. – Access mode:
<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8463455>

13. JSOUP [Electronic source]. – Access mode: <https://jsoup.org>

14. ZXing [Electronic source]. – Access mode:
<https://opensource.google/projects/zxing>

15. JodaTime [Electronic source]. – Access mode:
<https://www.joda.org/joda-time>

16. Caldroid [Electronic source]. – Access mode:
<https://github.com/roomorama/Caldroid>

17. Round Image View [Electronic source]. – Access mode:
<https://github.com/vinc3m1/RoundedImageView>

18. Material Drawer [Electronic source]. – Access mode:
<https://github.com/mikepenz/MaterialDrawer>

19. Calendula [Electronic source]. – Access mode: <https://github.com/citiususc/calendula>

20. Self Sufficient Healthcare Gadget for Elderly [Electronic source]. – Access mode: <https://ieeexplore.ieee.org/document/9002467>

21. Методичні вказівки по економічному обґрунтуванню дипломних проектів для студентів спеціальності 8.090801 «Мікроелектроніка та напівпровідникові прилади» [Текст] / Уклали В.О. Варинська, Т.А. Антоненко, – Запоріжжя: ЗНТУ, 2002 – 26 с.

22. Коробко О.В. Методичні вказівки до виконання розділу «Охорона праці та безпека у надзвичайних ситуаціях» у дипломних проектах (роботах) студентів–магістрів спеціальностей 121 Інженерія програмного забезпечення (спеціалізація: інженерія програмного забезпечення), 121 Інженерія програмного забезпечення (спеціалізація: програмне забезпечення систем), 122 Комп'ютерні науки (спеціалізація: інформаційні технології проектування), 122 Комп'ютерні науки (спеціалізація: системи штучного інтелекту), 123 Комп'ютерна інженерія (спеціалізація: комп'ютерні системи та мережі), 123 Комп'ютерна інженерія (спеціалізація: спеціалізовані комп'ютерні системи), 124 Системний аналіз (спеціалізація: інтелектуальні технології та прийняття рішень в складних системах) усіх форм навчання [Текст] / Укл.: О.В. Коробко – Запоріжжя: ЗНТУ, 2018 – 27с.

23. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом: ДСТУ Б В.2.5-82:2016. – [Чинний від 2017-04-01]. – К.: ДП «УкрНДНЦ», 2016. – 109 с. – (Державний Стандарт України)

24. Правила безпечної експлуатації електроустановок: НПАОП 40.1-1.01-97. – [Чинний від 1997-10-06]. – К.: Держнаглядохоронпраці, 1997. – 97 с. – (Нормативно-правовий акт охорони праці)

25. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПіН

3.3.2.007-98. – Чинний від 1998-12-10. – К. : МОЗ України, 1998. – 110 с. – (Державні санітарні правила та норми)

26. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу: Державні санітарні норми та правила від 30.05.2014 р. № z0472-14. Редакція від: 30.05.2014

27. «Правила охорони праці під час експлуатації електронно-обчислювальних машин»: НПАОП 0.00-1.28-10 Наказ Держгірпромнагляду від 26.03.2010 р. № 65

28. Санітарні норми мікроклімату виробничих приміщень: ДСН 3.3.6.042-99. – [Чинний від 1999-12-01]. – К. : МОЗ України, 1999. – 106 с. – (Державні санітарні норми)

29. Загальні санітарно-гігієнічні вимоги до повітря робочої зони: ГОСТ 12.1.005-88. - [Чинний від 1989-01-01]. – Міністерство охорони здоров'я СРСР, 1988. – 101 с. – (Міжнародний стандарт)

30. Опалення, вентиляція та кондиціонування: ДБН В.2.5-67:2013. – [Чинний від 2014-01-01]. – К.: Мінрегіонбуд України, 2013. – 149 с. – (Державні будівельні норми України)

31. Природне і штучне освітлення: ДБН В.2.5-28-2018. – [Чинний з 2019-03-01]. – К. : Мінрегіон України, 2018. – 133 с. – (Державні будівельні норми України)

32. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою: ДСТУ Б В.1.1-36:2016. – [Чинний від 2017-01-01]. – К.: Мінрегіонбуд України, 2016. – 66 с. – (Національний стандарт України)

33. Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги: ДСТУ 4297:2004. – [Чинний від 2004-10-01]. – УкрНДПБ МНС України, 2004. – 74 с. – (Національний стандарт України)

34. Єдина державна система цивільного захисту [Електрон. ресурс]. -
Режим доступу: <http://www.civilniy.org.ua/pages/rozdil-ii--edina-derzhavna-sistema-tsivilnogo-zakhistu.html>

ДОДАТОК А
Текст програми

Клас FabMenuMgr

```

public class FabMenuMgr implements
View.OnClickListener {

    private LeftDrawerMgr drawerMgr;
    private ExpandableFAB fabMenu;
    private HomePagerActivity activity;
    private List<View> fabMenuSubViews;
    private List<FloatingActionButton> fabMenuButtons;
    private int currentPage = 0;

    public FabMenuMgr(ExpandableFAB fab,
LeftDrawerMgr drawerMgr, HomePagerActivity a) {
        this.fabMenu = fab;
        this.activity = a;
        this.drawerMgr = drawerMgr;
    }

    public void init() {
        fabMenu.setOnClickListener(this);
        fabMenu.setImageDrawable(new
IconicsDrawable(activity)
            .icon(GoogleMaterial.Icon.gmd_plus)
            .paddingDp(5)
            .sizeDp(24)
            .colorRes(R.color.fab_default_icon_color));
        fabMenu.setSubViews(getScheduleActions());
        onViewPagerItemChange(0);
    }

    public void onViewPagerItemChange(int currentPage)
    {

        this.currentPage = currentPage;
        HomePages page =
HomePages.values()[currentPage];

        switch (page) {
            case HOME:
            case RECOMMENDATIONS:
                Intent intent = new
Intent(HomePagerActivity.getContext(),
RequiredActivity.class);
                HomePagerActivity.getContext().startActivity(intent);
                fabMenu.hide();
                break;
            case ROUTINES:
            case MEDICINES:
            case SCHEDULES:
                fabMenu.collapse();
                fabMenu.show();
                break;
        }
    }

    @Override
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.add_button:
                onClickAdd();
                break;
            case R.id.fab_action_routines_button:

startSchedulesActivity(ScheduleTypeFragment.TYPE_RO
UTINES);

                break;
            case R.id.fab_action_interval_button:

startSchedulesActivity(ScheduleTypeFragment.TYPE_H
OURLY);
                break;
            case R.id.fab_action_period_button:

startSchedulesActivity(ScheduleTypeFragment.TYPE_PE
RIOD);
                break;
            case R.id.fab_action_qr_button:
                startScanActivity();
                break;
        }
    }

    public void onPatientUpdate(Patient p) {
        for (FloatingActionButton fabMenuButton :
fabMenuButtons) {

            fabMenuButton.setBackgroundTintList(ColorStateList.val
ueOf(p.getColor()));
        }
    }

    private List<View> getScheduleActions() {
        fabMenuSubViews = new ArrayList<>();
        fabMenuButtons = new ArrayList<>();

        final View fabActionIntervalView =
activity.findViewById(R.id.fab_action_interval);
        final View fabActionRoutinesView =
activity.findViewById(R.id.fab_action_routines);
        final View fabActionPeriodView =
activity.findViewById(R.id.fab_action_period);

        fabMenuSubViews.add(fabActionPeriodView);
        fabMenuSubViews.add(fabActionIntervalView);
        fabMenuSubViews.add(fabActionRoutinesView);

        final FloatingActionButton fabActionInterval =
(FloatingActionButton)
activity.findViewById(R.id.fab_action_interval_button);
        final FloatingActionButton fabActionRoutines =
(FloatingActionButton)
activity.findViewById(R.id.fab_action_routines_button);
        final FloatingActionButton fabActionPeriod =
(FloatingActionButton)
activity.findViewById(R.id.fab_action_period_button);

        fabActionInterval.setOnClickListener(this);
        fabActionRoutines.setOnClickListener(this);
        fabActionPeriod.setOnClickListener(this);

        fabActionInterval.setBackgroundTintList(ColorStateList.v
alueOf(DB.patients().getActive(activity).getColor()));

        fabActionRoutines.setBackgroundTintList(ColorStateList.
valueOf(DB.patients().getActive(activity).getColor()));

        fabActionPeriod.setBackgroundTintList(ColorStateList.va
lueOf(DB.patients().getActive(activity).getColor()));

        fabMenuButtons.add(fabActionPeriod);
    }
}

```

```

        fabMenuButtons.add(fabActionInterval);
        fabMenuButtons.add(fabActionRoutines);

        return fabMenuSubViews;
    }

    private void onClickAdd() {
        HomePages page =
        HomePages.values()[currentPage];
        switch (page) {
            case HOME:
                return;
            case ROUTINES:
                launchActivity(RoutinesActivity.class);
                break;
            case MEDICINES:
                launchActivity(MedicinesActivity.class);
                break;
            case SCHEDULES:
                if
                (!PreferenceUtils.getBoolean(PreferenceKeys.SCHEDUL
                ES_HELP_SHOWN, false)) {

                    activity.launchActivityDelayed(SchedulesHelpActivity.cl
                    ass, 600);
                }
                fabMenu.toggle();
                break;
        }
    }

```

Клас HomePagerActivity

```

public class HomePagerActivity extends
    CalendulaActivity implements
        RoutinesListFragment.OnRoutineSelectedListener,
        MedicinesListFragment.OnMedicineSelectedListener,
        ScheduleListFragment.OnScheduleSelectedListener {

    public static final int
    REQ_CODE_EXTERNAL_STORAGE = 10;
    private static final String TAG =
    "HomePagerActivity";

    public static Context mContext;

    @MenuRes
    private static final int[] MENU_ITEMS = {
        R.id.action_sort, R.id.action_expand,
        R.id.action_calendar, R.id.action_schedules_help
    };

    @BindView(R.id.appbar)
    public AppBarLayout appBarLayout;
    @BindView(R.id.collapsing_toolbar)
    CollapsingToolbarLayout toolbarLayout;
    @BindView(R.id.add_button)
    ExpandableFAB fab;
    @BindView(R.id.user_info_fragment)
    View userInfoFragment;
    @BindView(R.id.main_content)
    CoordinatorLayout coordinatorLayout;
    // @BindView(R.id.fab_menu)
    // FloatingActionsMenu addButton;

```

```

    }

    private void launchActivity(Class<?> type) {
        activity.startActivity(new Intent(activity, type));
        activity.overridePendingTransition(0, 0);
    }

    private void startSchedulesActivity(int scheduleType)
    {
        Intent i = new Intent(activity,
        ScheduleCreationActivity.class);
        i.putExtra("scheduleType", scheduleType);
        activity.startActivity(i);
        activity.overridePendingTransition(0, 0);
        fabMenu.collapse();
    }

    private void startScanActivity() {
        Intent i = new Intent(activity, ScanActivity.class);
        i.putExtra("after_scan_pkg",
        activity.getPackageName());
        i.putExtra("after_scan_cls",
        ConfirmSchedulesActivity.class.getName());
        activity.startActivity(i);
        activity.overridePendingTransition(0, 0);
        fabMenu.collapse();
    }
}

```

```

@BindView(R.id.toolbar_title)
TextView toolbarTitle;
@BindView(R.id.container)
ViewPager mViewPager;
@BindView(R.id.sliding_tabs)
TabLayout tabLayout;

```

```

private boolean appBarLayoutExpanded = true;
private boolean active = false;
private Drawable icAgendaMore;
private Drawable icAgendaLess;
private FabMenuMgr fabMgr;
private HomeProfileMgr homeProfileMgr;
private HomePageAdapter mSectionsPagerAdapter;
private LeftDrawerMgr drawerMgr;
private Patient activePatient;
private int pendingRefresh = -2;
private Queue<Object> pendingEvents = new
LinkedList<>();
private Handler handler;

private SparseArray<MenuItem> menuItems;

@ColorInt
private int previousColor = -1;

public void showPagerItem(int position) {
    showPagerItem(position, true);
}

public void showPagerItem(int position, boolean
updateDrawer) {
    if (position >= 0 && position <

```

```

mViewPager.getChildCount()) {
    mViewPager.setCurrentItem(position);
    if (updateDrawer) {
        drawerMgr.onPagerPositionChange(position);
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {

    // Inflate the menu; this adds items to the action bar if
    it is present.
    getMenuInflater().inflate(R.menu.home, menu);
    menuItems = new SparseArray<>(menu.size());
    for (int menuItem : MENU_ITEMS) {
        menuItems.put(menuItem,
            menu.findItem(menuItem));
    }

    menuItems.get(R.id.action_sort).setIcon(IconUtils.icon(ge
tApplicationContext(),
CommunityMaterial.Icon.cmd_sort, R.color.white));
    return true;
}

@Override
public boolean onPrepareOptionsMenu(Menu menu) {

    final int pageNum = mViewPager.getCurrentItem();
    final HomePages page =
HomePages.getPage(pageNum);

    // hide all items first
    for (int menuItem : MENU_ITEMS) {
        menuItems.get(menuItem).setVisible(false);
    }

    // show items relevant to the current page
    switch (page) {
        case HOME:
            final boolean expanded =
((DailyAgendaFragment)
getViewPagerFragment(HomePages.HOME)).isExpanded
();

            menuItems.get(R.id.action_expand).setVisible(true);

            menuItems.get(R.id.action_expand).setIcon(!expanded ?
icAgendaMore : icAgendaLess);
            break;
            case MEDICINES:

            menuItems.get(R.id.action_sort).setVisible(true);
            break;
            case SCHEDULES:

            menuItems.get(R.id.action_schedules_help).setVisible(tr
ue);
            break;
    } // tut perekl
    return super.onPrepareOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem
item) {

```

```

switch (item.getItemId()) {
    case R.id.action_calendar:
        startActivity(new Intent(this,
CalendarActivity.class));
        return true;
    case R.id.action_expand:

        final boolean expanded =
((DailyAgendaFragment)
getViewPagerFragment(HomePages.HOME)).isExpanded
();

        appBarLayout.setExpanded(expanded);

        boolean delay = appBarLayoutExpanded &&
!expanded || !appBarLayoutExpanded && expanded;

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                ((DailyAgendaFragment)
getViewPagerFragment(HomePages.HOME)).toggleView
Mode();
            }
        }, delay ? 500 : 0);

        item.setIcon(expanded ? icAgendaMore :
icAgendaLess);
        return true;
        case R.id.action_schedules_help:
            launchActivity(new Intent(this,
SchedulesHelpActivity.class));
            return true;
        case R.id.action_sort:
            ((MedicinesListFragment)
getViewPagerFragment(HomePages.MEDICINES)).toggl
eSort();
            return true;
    }

    return super.onOptionsItemSelected(item);
}

public void launchActivityDelayed(final Class<?>
activityClazz, int delay) {
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            startActivity(new
Intent(HomePageActivity.this, activityClazz));
            overridePendingTransition(0, 0);
        }
    }, delay);
}

@Override
public void onRoutineSelected(Routine r) {
    Intent i = new Intent(this, RoutinesActivity.class);

    i.putExtra(CalendarApp.INTENT_EXTRA_ROUTINE_
ID, r.getId());
    launchActivity(i);
}

@Override

```

```

public void onCreateRoutine() {
    //do nothing
}

@Override
public void onMedicineSelected(Medicine m) {
    Intent i = new Intent(this,
        MedicineInfoActivity.class);

    i.putExtra(CalendulaApp.INTENT_EXTRA_MEDICINE
        _ID, m.getId());
    launchActivity(i);
}

@Override
public void onCreateMedicine() {

    //do nothing
}

@Override
public void onScheduleSelected(Schedule r) {
    Intent i = new Intent(this,
        ScheduleCreationActivity.class);

    i.putExtra(CalendulaApp.INTENT_EXTRA_SCHEDULE
        _ID, r.getId());
    launchActivity(i);
}

@Override
public void onCreateSchedule() {

}

// Method called from the event bus
@Subscribe
public void handleEvent(final Object event) {
    if (active) {
        handler.post(new Runnable() {
            @Override
            public void run() {

                if (event instanceof
                    PersistenceEvents.ModelCreateOrUpdateEvent) {

                    PersistenceEvents.ModelCreateOrUpdateEvent
                        modelCreateOrUpdateEvent =
                        (PersistenceEvents.ModelCreateOrUpdateEvent) event;
                    LogUtil.d(TAG, "handleEvent: " +
                        modelCreateOrUpdateEvent.clazz.getName());
                    ((DailyAgendaFragment)
                        getViewPagerFragment(HomePages.HOME)).notifyData
                        Change();

                    ((RoutinesListFragment)
                        getViewPagerFragment(HomePages.ROUTINES)).notify
                        DataChange();

                    ((MedicinesListFragment)
                        getViewPagerFragment(HomePages.MEDICINES)).notif
                        yDataChange();

                    ((ScheduleListFragment)
                        getViewPagerFragment(HomePages.SCHEDULES)).notif
                        yDataChange();
                } else if (event instanceof
                    PersistenceEvents.IntakeConfirmedEvent) {
                    // dismiss "take all" button, update
                    checkboxes

```

```

                    ((DailyAgendaFragment)
                        getViewPagerFragment(HomePages.HOME)).notifyData
                        Change();

                    // stock info may need to be updated
                    ((MedicinesListFragment)
                        getViewPagerFragment(HomePages.MEDICINES)).notif
                        yDataChange();
                } else if (event instanceof
                    PersistenceEvents.ActiveUserChangeEvent) {
                    activePatient =
                    ((PersistenceEvents.ActiveUserChangeEvent)
                        event).patient;

                    updateTitle(mViewPager.getCurrentItem());

                    toolbarLayout.setContentScrimColor(activePatient.getC
                        olor());

                    fabMgr.onPatientUpdate(activePatient);
                } else if (event instanceof
                    PersistenceEvents.UserUpdateEvent) {
                    Patient p =
                    ((PersistenceEvents.UserUpdateEvent) event).patient;
                    ((DailyAgendaFragment)
                        getViewPagerFragment(HomePages.HOME)).onUserUpd
                        ate();

                    drawerMgr.onPatientUpdated(p);
                    if (DB.patients().isActive(p,
                        HomePagerActivity.this)) {
                        activePatient = p;

                        updateTitle(mViewPager.getCurrentItem());

                        toolbarLayout.setContentScrimColor(activePatient.getC
                            olor());

                        fabMgr.onPatientUpdate(activePatient);
                    }
                } else if (event instanceof
                    PersistenceEvents.UserCreateEvent) {
                    Patient created =
                    ((PersistenceEvents.UserCreateEvent) event).patient;
                    drawerMgr.onPatientCreated(created);
                } else if (event instanceof
                    HomeProfileMgr.BackgroundUpdatedEvent) {
                    ((DailyAgendaFragment)
                        getViewPagerFragment(HomePages.HOME)).refresh();
                } else if (event instanceof
                    ConfirmActivity.ConfirmStateChangeEvent) {
                    pendingRefresh =
                    ((ConfirmActivity.ConfirmStateChangeEvent)
                        event).position;
                    ((DailyAgendaFragment)
                        getViewPagerFragment(HomePages.HOME)).refreshPosit
                        ion(pendingRefresh);
                } else if (event instanceof
                    DailyAgenda.AgendaUpdatedEvent) {
                    ((DailyAgendaFragment)
                        getViewPagerFragment(HomePages.HOME)).notifyData
                        Change();

                    homeProfileMgr.updateDate();
                } else if (event instanceof
                    StockRunningOutEvent) {
                    final StockRunningOutEvent sro =
                    (StockRunningOutEvent) event;
                    handler.postDelayed(new Runnable() {
                        @Override
                        public void run() {

```

```

StockDisplayUtils.showStockRunningOutDialog(HomePa
gerActivity.this, sro.m, sro.days);
    }
    }, 1000);
    } else if (event instanceof
PersistenceEvents.DatabaseUpdateEvent) {
        checkDatabaseUpdateNeeded();
    }
    }
    });
    } else {
        pendingEvents.add(event);
    }
}

    public void checkDatabaseUpdateNeeded() {
        final boolean needPrompt =
PreferenceUtils.getBoolean(PreferenceKeys.DRUGDB_D
B_PROMPT, false);
        if (needPrompt) {
            new MaterialStyledDialog.Builder(this)
                .setStyle(Style.HEADER_WITH_ICON)
                .setIcon(IconUtils.icon(this,
CommunityMaterial.Icon.cmd_database, R.color.white,
100))
                .setHeaderColor(R.color.android_blue)
                .withDialogAnimation(true)

.setTitle(R.string.enable_prescriptions_dialog_title)

.setDescription(R.string.reinstall_prescriptions_dialog_m
essage)

                .setCancelable(false)

.setPositiveText(getString(R.string.dialog_yes_option))
                .onPositive(new
MaterialDialog.SingleButtonCallback() {
                    @Override
                    public void onClick(@NonNull
MaterialDialog dialog, @NonNull DialogAction which) {
                        Intent i = new
Intent(HomePagerActivity.this,
CalendulaSettingsActivity.class);

i.putExtra(CalendulaSettingsActivity.EXTRA_SHOW_D
B_DIALOG, true);
                        startActivity(i);

PreferenceUtils.edit().remove(PreferenceKeys.DRUGDB_
DB_PROMPT.key()).apply();
                    }
                })
                .setNegativeText(R.string.dialog_no_option)
                .onNegative(new
MaterialDialog.SingleButtonCallback() {
                    @Override
                    public void onClick(@NonNull
MaterialDialog dialog, @NonNull DialogAction which) {
                        dialog.cancel();

PreferenceUtils.edit().remove(PreferenceKeys.DRUGDB_
DB_PROMPT.key()).apply();
                    }
                })
                .show();
        }
    }
}

```

```

    }

    public void showSnackbar(@StringRes final int text) {
        Snack.show(text, coordinatorLayout,
Snackbar.LENGTH_SHORT);
    }

    Fragment getViewPagerFragment(HomePages page) {
        String tag =
FragmentUtils.makeViewPagerFragmentName(R.id.conta
iner, page.ordinal());
        return
getSupportFragmentManager().findFragmentByTag(tag);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = getBaseContext();
        ButterKnife.bind(this);

        setupToolbar(null, Color.TRANSPARENT);
        initializeDrawer(savedInstanceState);
        setupStatusBar(Color.TRANSPARENT);
        subscribeToEvents();
        handler = new Handler();
        mSectionsPagerAdapter = new
HomePageAdapter(getSupportFragmentManager(), this,
this);

        mViewPager.setAdapter(mSectionsPagerAdapter);

mViewPager.addOnPageChangeListener(getPageChange
Listener());
        mViewPager.setOffscreenPageLimit(5);

        // Set up home profile
        homeProfileMgr = new HomeProfileMgr();
        homeProfileMgr.init(userInfoFragment, this);

        activePatient = DB.patients().getActive(this);
        updateScrim(0);

        fabMgr = new FabMenuMgr(fab, drawerMgr,
this);
        fabMgr.init();

        fabMgr.onPatientUpdate(activePatient);

        // Setup the tabLayout
        setupTabLayout();

        AppBarLayout.OnOffsetChangedListener mListener
= new AppBarLayout.OnOffsetChangedListener() {
            @Override
            public void onOffsetChanged(AppBarLayout
appBarLayout, int verticalOffset) {
                if (((toolbarLayout.getHeight() + verticalOffset)
< (1.8 *
ViewCompat.getMinimumHeight(toolbarLayout))) {
                    homeProfileMgr.onCollapse();
                    toolbarTitle.animate().alpha(1);
                }
            }
        }
    }
}

```

```

        appBarLayoutExpanded = false;
        LogUtil.d(TAG, "OnCollapse");
    } else {
        appBarLayoutExpanded = true;
        if (mViewPager.getCurrentItem() == 0) {
            toolbarTitle.animate().alpha(0);
        }
        homeProfileMgr.onExpand();
        LogUtil.d(TAG, "OnExpand");
    }
}

}

appBarLayout.addOnOffsetChangedListener(mListener);

icAgendaLess = new IconicsDrawable(this)

.icon(CommunityMaterial.Icon.cmd_unfold_less_hori-
    zontal)
    .color(Color.WHITE)
    .sizeDp(24);

icAgendaMore = new IconicsDrawable(this)

.icon(CommunityMaterial.Icon.cmd_unfold_more_hori-
    zontal)
    .color(Color.WHITE)
    .sizeDp(24);

if
(!PreferenceUtils.getBoolean(PreferenceKeys.HOME_IN-
    TRO_SHOWN, false)) {
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            launchActivity(new
            Intent(HomePageActivity.this,
            MaterialIntroActivity.class));
        }
    }, 500);
}

if (getIntent() != null &&
getIntent().getBooleanExtra("invalid_notification_error",
false)) {
    Toast.makeText(this, "Error",
    Toast.LENGTH_SHORT).show();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            showInvalidNotificationError();
        }
    }, 500);
}

checkDatabaseUpdateNeeded();
}

public static Context getContext() {
    return mContext;
}

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

@Override
protected void onResume() {
    super.onResume();
    Patient p = DB.patients().getActive(this);
    drawerMgr.onActivityResume(p);
    active = true;

    // process pending events
    while (!pendingEvents.isEmpty()) {
        LogUtil.d(TAG, "Processing pending event...");
        handleEvent(pendingEvents.poll());
    }
}

@Override
protected void onPause() {
    active = false;
    super.onPause();
}

private void showInvalidNotificationError() {

    final boolean expanded = ((DailyAgendaFragment)
    getViewPagerFragment(HomePages.HOME)).isExpanded
    ();

    new AlertDialog.Builder(this)
        .setTitle(R.string.notification_error_title)
        .setMessage(R.string.notification_error_msg)
        .setCancelable(true)
        .setIcon(IconUtils.icon(this,
        CommunityMaterial.Icon.cmd_bug, R.color.black))
        .setPositiveButton(R.string.tutorial_understood,
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog,
            int which) {
                dialog.dismiss();
                if (!expanded) {
                    appBarLayout.setExpanded(expanded);
                }
            }
        })
        .show();
}

menuItems.get(R.id.action_expand).setIcon(expanded ?
icAgendaMore : icAgendaLess);
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        ((DailyAgendaFragment)
        getViewPagerFragment(HomePages.HOME)).toggleView
        Mode();
    }
}, 200);
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        ((DailyAgendaFragment)
        getViewPagerFragment(HomePages.HOME)).scrollTo(D
        ateTime.now());
    }
}, 600);
}
}
}).create().show();
}
}

```

```

private void setupTabLayout() {

    tabLayout.setupWithViewPager(mViewPager);

    for (int i = 0; i < tabLayout.getTabCount(); i++) {

        Drawable icon = new IconicsDrawable(this)
            .icon(HomePages.values()[i].icon)
            .alpha(80)
            .paddingDp(2)
            .color(Color.WHITE)
            .sizeDp(24);

        tabLayout.getTabAt(i).setIcon(icon);
    }

    private ViewPager.OnPageChangeListener
    getPageChangeListener() {
        return new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageScrolled(int position, float
            positionOffset, int positionOffsetPixels) {

            }

            @Override
            public void onPageSelected(int position) {
                updateTitle(position);
                updateScrim(position);
                fabMgr.onViewPagerItemChange(position);
                if (position == HomePages.HOME.ordinal()
                && !(DailyAgendaFragment
                getViewPagerFragment(HomePages.HOME)).isExpanded
                ()) {
                    appBarLayout.setExpanded(true);
                    CoordinatorLayout.LayoutParams
                    layoutParams = (CoordinatorLayout.LayoutParams)
                    appBarLayout.getLayoutParams();
                    ((DisableableAppBarLayoutBehavior)
                    layoutParams.getBehavior()).setEnabled(true);
                } else {
                    appBarLayout.setExpanded(false);
                    CoordinatorLayout.LayoutParams
                    layoutParams = (CoordinatorLayout.LayoutParams)
                    appBarLayout.getLayoutParams();
                    ((DisableableAppBarLayoutBehavior)
                    layoutParams.getBehavior()).setEnabled(false);
                }

                invalidateOptionsMenu();
            }

            @Override
            public void onPageScrollStateChanged(int state) {

                }
            }
        };

        private void updateScrim(int position) {
            @ColorInt final int color = (position ==
            HomePages.HOME.ordinal()) ?
            ContextCompat.getColor(this, R.color.transparent_black)
            : activePatient.getColor();

            if (previousColor != -1) {
                ValueAnimator colorAnimation =
                ValueAnimator.ofObject(new ArgbEvaluator(),
                previousColor, color);
                colorAnimation.setDuration(250); // milliseconds
                colorAnimation.addUpdateListener(new
                ValueAnimator.AnimatorUpdateListener() {

                    @Override
                    public void
                    onAnimationUpdate(ValueAnimator animator) {
                        toolbarLayout.setContentScrimColor((int)
                        animator.getAnimatedValue());
                    }

                });
                colorAnimation.start();
            } else {
                toolbarLayout.setContentScrimColor(color);
            }
            previousColor = color;
        }

        private void updateTitle(int page) {
            String title;
            if (page == HomePages.HOME.ordinal()) {
                title = getString(R.string.app_name);
            } else {
                title =
                getString(R.string.relation_user_possession_thing,
                activePatient.getName(),
                getString(HomePages.values()[page].title));
            }

            toolbarTitle.setText(title);
        }

        private void initializeDrawer(Bundle
        savedInstanceState) {
            drawerMgr = new LeftDrawerMgr(this, toolbar);
            drawerMgr.init(savedInstanceState);
        }

        private void launchActivity(Intent i) {
            startActivity(i);
            this.overridePendingTransition(0, 0);
        }
    }
}

```

ДОДАТОК Б
Слайди презентації

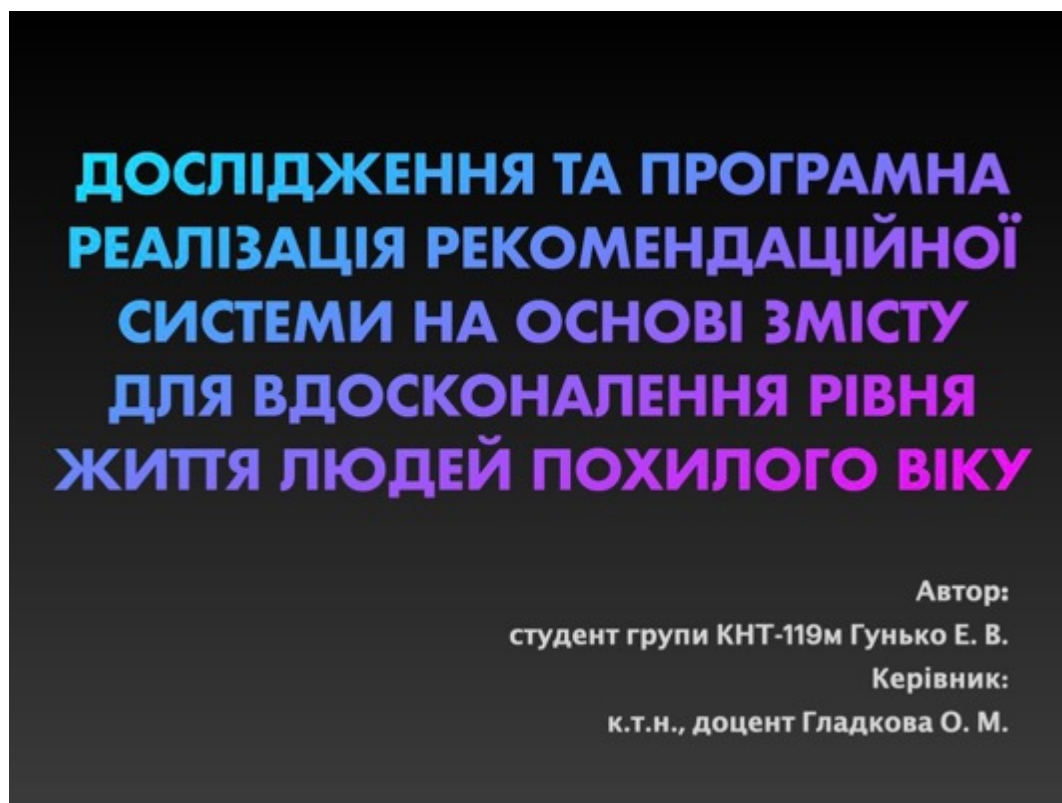


Рисунок Б.1 – Титульна сторінка

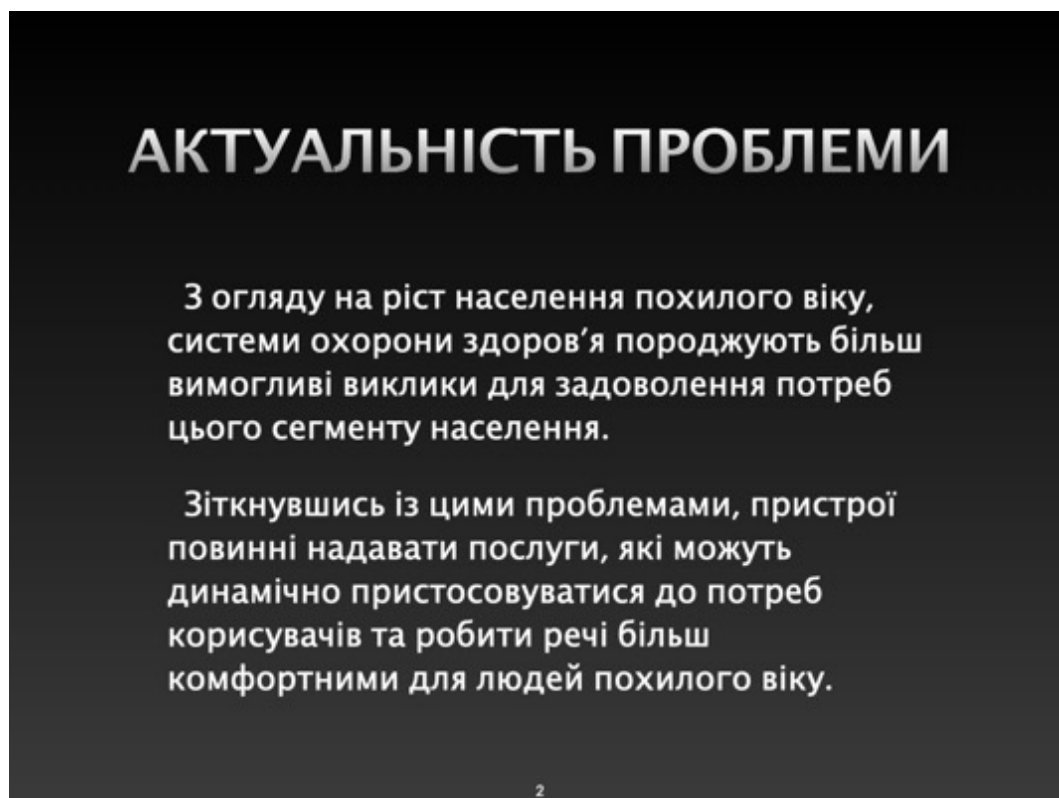


Рисунок Б.2 – Актуальність проблеми

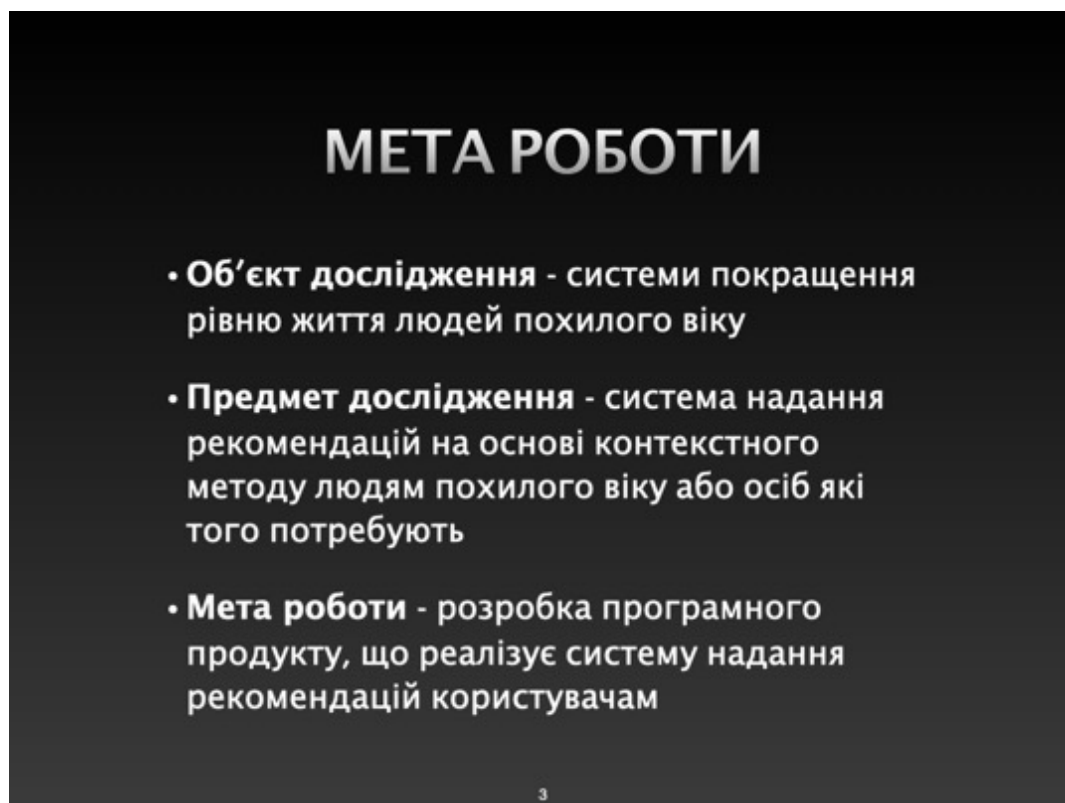


Рисунок Б.3 – Мета роботи

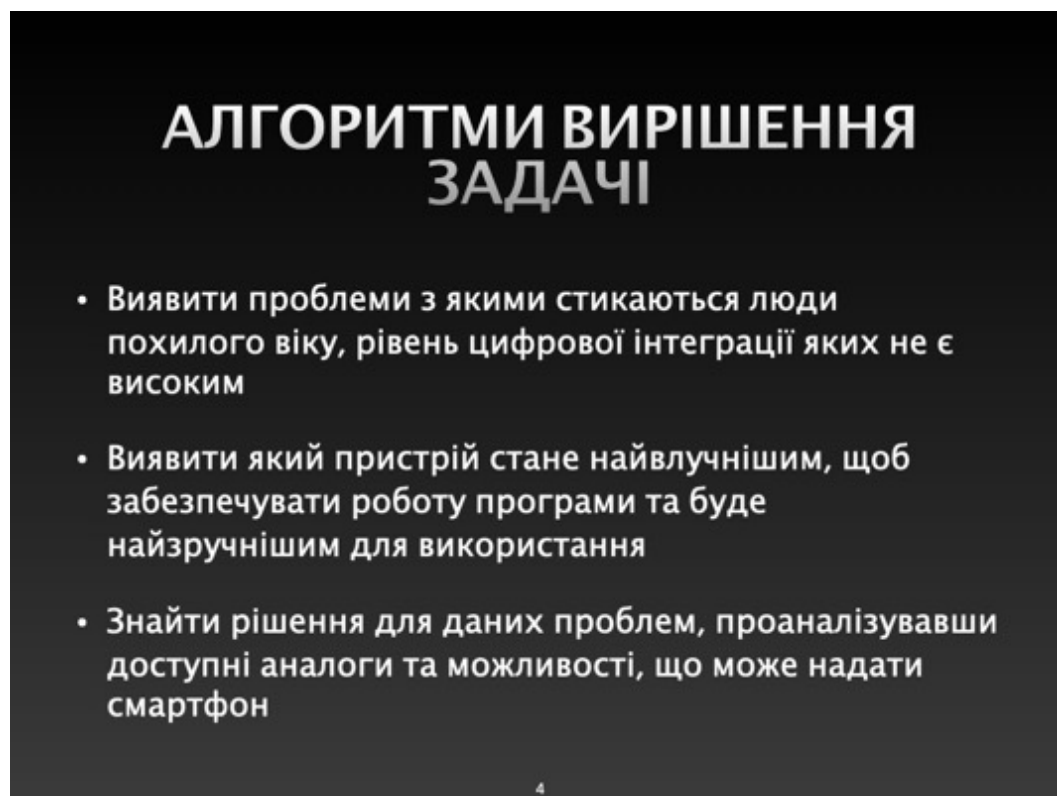


Рисунок Б.4 – Алгоритми вирішення задачі

АНАЛІЗ АНАЛОГІВ

- Було виявлено та взято до уваги перелік функцій, які не має жоден з конкурентів розроблюваного застосунку
- Було прийняте рішення щодо цільової аудиторії

	Samsung Health	ExyLife	Health Pal	Remente	Health Nutrition Fitness
Безкоштов. завантаження	✓	✓	✓	✓	✓
Не має платних функцій	✓	✓	✗	✗	✗
Доступність на Android	✓	✓	✓	✓	✓
Доступність на iOS	✗	✓	✗	✓	✗
Макс. шагомір	✓	✗	✗	✗	✗
Рутини	✗	✗	✗	✗	✗
База даних алергій	✗	✗	✗	✗	✗
Мультиязычність	✗	✗	✗	✗	✗
Акселерометр	✗	✗	✗	✗	✗
Рекомендації щодо ліків	✗	✗	✗	✗	✗
Рекомендації щодо їжі	✓	✗	✓	✓	✓
Рекомендації щодо ваги	✓	✓	✗	✗	✓
Кастомізація профілю	✓	✗	✗	✗	✗
Онлайн-тур	✗	✗	✗	✗	✗
Обмеження доступу	✗	✗	✗	✗	✗
База даних ліків	✗	✗	✗	✗	✗
База даних захворювань	✗	✗	✗	✗	✗
Розклад прийому ліків	✗	✗	✗	✗	✗
Система досягнень	✓	✓	✓	✓	✗

5

Рисунок Б.5 – Аналіз аналогів

АНАЛІЗ ПЛАТФОРМ

Найвідоміша дуель у світі смартфонів у якій в даному випадку перемогу отримала платформа Android.

	Android	iOS
Поширеність серед людей похилого віку	Висока	Низька
Доступність додатків	Висока	Низька
Наявність вбудованих сенсорів	Часто	Відсутні

6

Рисунок Б.6 – Аналіз платформ



Рисунок Б.7 – Обґрунтування вибору

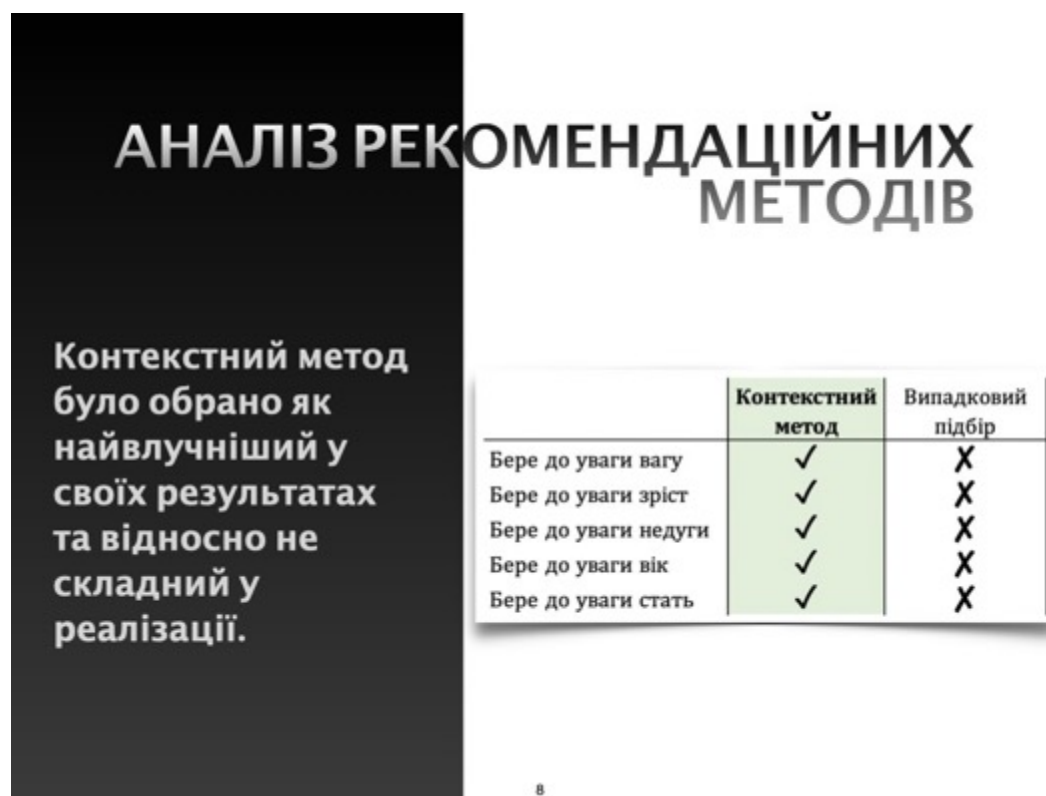


Рисунок Б.8 – Аналіз рекомендаційних методів

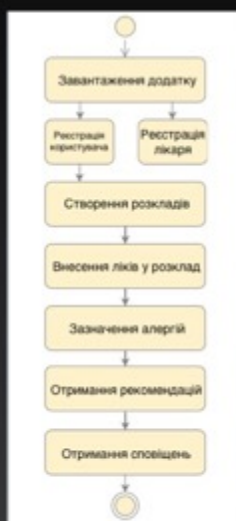
СТРУКТУРА ЗАСТОСУНКУ



9

Рисунок Б.9 – Структура застосунку

ДІАГРАМА ВИКОРИСТАННЯ

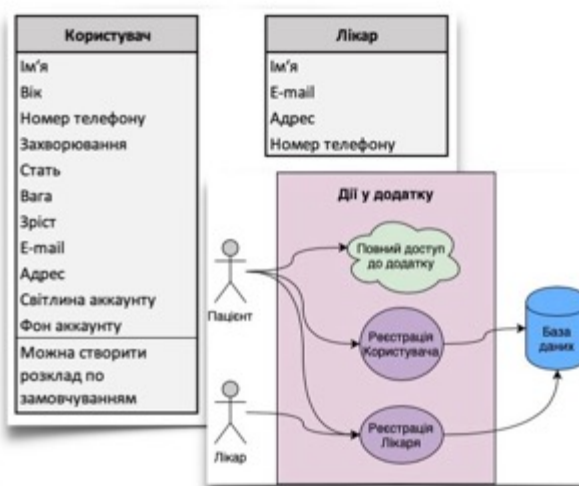


10

Рисунок Б.10 – Діаграма використання

РЕЄСТРАЦІЯ У ЗАСТОСУНКУ

Застосунок має широкий функціонал, але реєстрація має одну з головних ролей, оскільки за допомогою отриманих даних система надає рекомендації користувачу.



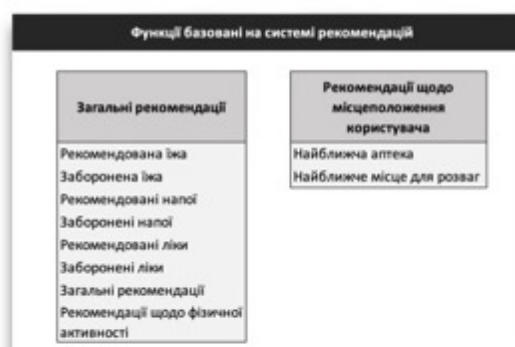
11

Рисунок Б.11 – Реєстрація у застосунку

ОГЛЯД РОЗРОБКИ

Контекстний метод рекомендацій базується на реєстрації користувача.

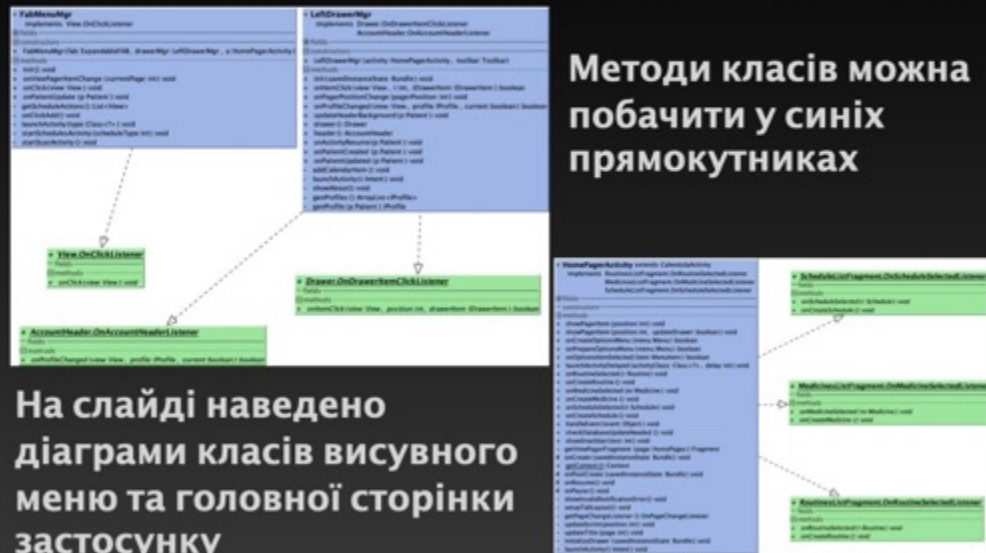
Чим більше надано даних, тим більше рекомендацій дає система.



12

Рисунок Б.12 – Огляд розробки

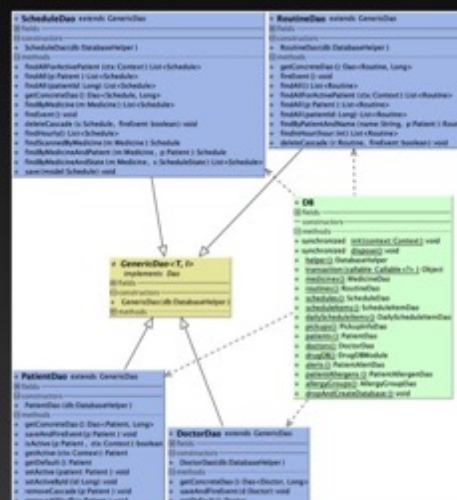
ОГЛЯД ДІАГРАМ КЛАСІВ



13

Рисунок Б.13 – Огляд діаграм класів

ОГЛЯД СТРУКТУРИ БАЗИ ДАНИХ



14

Рисунок Б.14 – Огляд структури бази даних

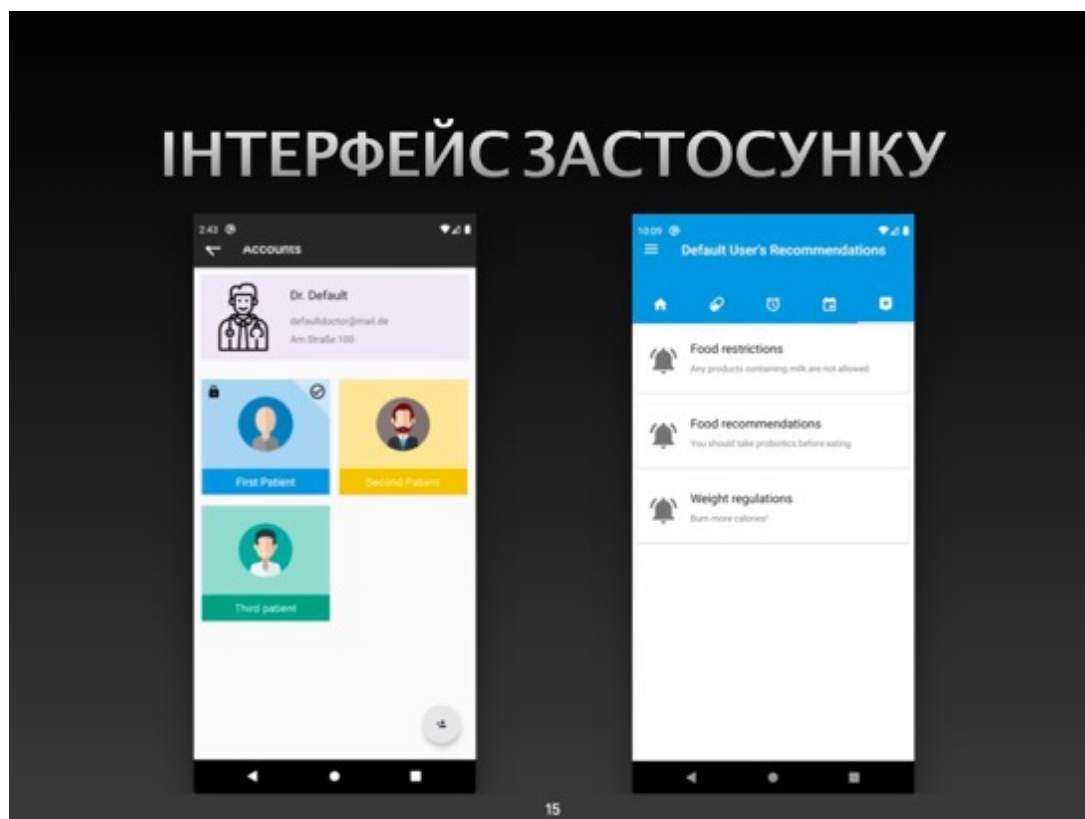


Рисунок Б.15 – Інтерфейс застосунку

ПЕРЕВАГИ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

	Розроблений додаток	Samsung Health	Eufylife	Health Pal	Remente	Health Nutrition Fitness
Безкоштов. завантаження	✓	✓	✓	✓	✓	✓
Не має платних функцій	✓	✓	✓	✗	✗	✗
Доступність на Android	✓	✓	✓	✓	✓	✓
Доступність на iOS	✗	✗	✓	✗	✓	✗
Має шагомір	✗	✓	✗	✗	✗	✗
Рутини	✓	✗	✗	✗	✗	✗
База даних алергій	✓	✗	✗	✗	✗	✗
Мультиакаунтність	✓	✗	✗	✗	✗	✗
Аккаунт лікаря	✓	✗	✗	✗	✗	✗
Рекомендації щодо ліків	✓	✗	✗	✗	✗	✗
Рекомендації щодо їди	✓	✓	✗	✓	✓	✓
Рекомендації щодо ваги	✓	✓	✓	✗	✗	✓
Кастомізація профілю	✓	✓	✗	✗	✗	✗
Онлайн чат	✓	✗	✗	✗	✗	✗
Обмеження доступу	✓	✗	✗	✗	✗	✗
База даних ліків	✓	✗	✗	✗	✗	✗
База даних захворювань	✓	✗	✗	✗	✗	✗
Розклад прийому ліків	✓	✗	✗	✗	✗	✗
Система досягнень	✗	✓	✓	✓	✓	✗

16

Рисунок Б.16 – Переваги розробленого застосунку

ВИСНОВКИ

- Знайдено рішення для проблем, проаналізувавши доступні аналоги та можливості, що може надати смартфон
- Розроблено алгоритм, та програмний продукт, що реалізує систему надання рекомендацій на основі контекстного методу людям похилого віку або осіб які того потребують
- Було опубліковано тези доповіді на X Міжнародній науково-практичній конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій»

17

Рисунок Б.17 – Висновки