МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«Мобільна система самоконтролю при позбавленні від шкідливої звички паління»

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма «**Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 42

_____Юшкевич О. С._____
(прізвище та ініціали)

Керівник _____Циганок В. В._____
(прізвище та ініціали)
_____доктор технічних наук_____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.
_____
"___"_____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Юшкевич Олександр Сергійович
(прізвище, ім'я, по батькові)

1. **Тема проекту (роботи)**
«*Мобільна система самоконтролю при позбавленні від шкідливої звички паління*»

затверджена протоколом засідання кафедри від « 23 » грудня 2021 р. № 4

2. **Термін здачі студентом закінченого проекту** *29 травня 2022 року*
3. **Вихідні дані до проекту**
Технічне завдання

4. **Зміст розрахунково-пояснювальної записки**

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Аналіз якості та тестування програмного забезпечення
4) Розгортання та впровадження програмного забезпечення

5. **Перелік презентаційного матеріалу**
1) Схема структурна варіантів використань
2) Креслення вигляду екранних форм
3) Схема структурна класів програмного забезпечення

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Циганок В. В /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Юшкевич О.С. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	*Вивчення рекомендованої літератури*	*10.04.2022*	
2.	*Аналіз існуючих методів розв'язання задачі*	*14.04.2022*	
3.	*Постановка та формалізація задачі*	*18.04.2022*	
4.	*Аналіз вимог до програмного забезпечення*	*20.04.2022*	
5.	*Алгоритмізація задачі*	*23.04.2022*	
6.	*Моделювання програмного забезпечення*	*25.04.2022*	
7.	*Обґрунтування використовуваних технічних засобів*	*28.04.2022*	
8.	*Розробка архітектури програмного забезпечення*	*01.05.2022*	
9.	*Розробка програмного забезпечення*	*02.05.2022*	
10.	*Налагодження програми*	*12.05.2022*	
11.	*Виконання графічних документів*	*13.05.2022*	
12.	*Оформлення пояснювальної записки*	*15.05.2022*	
13.	*Подання ДП на попередній захист*	*31.05.2022*	

Студент-дипломник _____ / Юшкевич О.С. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Циганок В.В. /
(підпис) (ПІБ)

АНОТАЦІЯ

Структура та обсяг роботи : Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 16 рисунків, 47 таблиць, 8 джерел.

Об'єкт дослідження : розробка мобільного застосування для індивідуальної боротьби з тютюнопалінням «SmokeBreak».

Мета дипломного проекту : підвищення ефективності подолання шкідливої звички тютюнопаління, засноване на щоденному відслідковуванні власного прогресу боротьби з залежністю.

У першому розділі пояснювальної записки проаналізовано предметну область, відомі аналоги, визначено варіанти використання програмного забезпечення, розроблено функціональні і нефункціональні вимоги.

У другому розділі пояснювальної записки розроблено загальну архітектуру програмного забезпечення, змодельовано структуру програмного коду, описано модель бази даних.

У третьому розділі пояснювальної записки описано план тестування програмного забезпечення на предмет якості, наведено контрольні варіанти тестування.

У четвертому розділі описано процес розгортання та запуск програмного забезпечення, а також керівництво користувача.

КЛЮЧОВІ СЛОВА : МОБІЛЬНЕ ЗАСТОСУВАННЯ, MVP, ЗДОРОВ'Я, ЗАЛЕЖНІСТЬ, ANDROID, REALM

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of six sections, contains 16 figures, 47 tables, 8 sources.

The third section describes the process of testing software quality. The fourth section describes the process of deployment of the software and the management of a non-resident

Object of research : the development of the mobile application for personal smoking control «SmokeBreak».

Diploma project purpose : elevation of the efficiency of overcoming the harmful habit of tobacco smoking, based on daily monitoring of personal progress of fighting the addiction.

The first section of explainatory note contains an analysis of the subject area, known analogues, the definition of use cases of the software, the development of functional and non-functional requirements.

The second section of explainatory note contains a description of developing the software architecture, modeling of the programming code structure and defining the database model.

The third section of explainatory note contains a description of a software testing plan on the subject of quality, a definition of test cases.

The fourth section of explainatory note contains the description of deploying and starting the mobile aplication and a user guide.

KEY WORDS: MOBILE APPLICATION, MVP, HEALTH, ADDICTION, ANDROID, REALM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Тютюн – рід однорічних та багаторічних рослин сімейства пасльонових, що містять в собі нікотин, який має стимулюючу дію на організм.

Тютюнопаління – процес вдихання продуктів тління висушеного листя тютюну, в основному у вигляді цигарок. Регулярне тютюнопаління завдає шкоди здоров'ю курця та викликає тютюнову залежність.

BPMN (англ. Business Process Model and Notation) – це стандарт, що надає нотації для визначення та конструювання бізнес-процесів.

MVP (англ. Modal-View-Presenter) – архітектурний шаблон, суть якого полягає у відділенні бізнес логіки від візуального представлення даних.

UI (англ. User Interface) – інтерфейс користувача програмного забезпечення, засіб взаємодії людини з інформаційною системою.

NoSQL – (англ. Not only SQL) – модель бази даних, що реалізує зберігання даних за допомогою механізмів, що відрізняються від табличних відношень у реляційних базах даних.

Фреймворк (англ. Framework – програмний каркас) – комплекс готових програмних рішень, що є системоутворюючим для розробки програмного забезпечення.

API (англ. Application Programming Interface) – це визначення протоколів взаємодії різних систем програмного забезпечення між собою.

APK (англ. Android Package) – формат файлів збірки для операційних систем на базі платформи Android.

ВСТУП

Тютюнопаління є однією з найпоширеніших шкідливих звичок, нарівні з вживанням алкоголю та наркотичних речовин. Кинути курити — одне із кращих рішень у житті кожного курця. Вже з першого дня відмови від цигарок організм починає відновлюватися, а ризики виникнення захворювань, асоційованих з курінням, зменшуються. Чим раніше курець прийме рішення кинути тютюн, тим краще.

За даними Українського центру контролю за курінням, близько 25% дорослих українців є щоденними курцями (42,2% чоловіків та 9,4% жінок). В той час як влада регулярно впроваджує нові законодавчі ініціативи з метою обмеження тютюнопаління серед населення, такі як обмеження куріння в громадських місцях, зменшення маркетингової привабливості та цінової доступності тютюнових виробів, тенденція залишається сталою, і кількість курців не зменшується, і щоденно від проблем зі здоров'ям, викликаних тютюнопалінням, гине 230 українців, 85 тисяч українців - щороку. Це вказує на обмежену ефективність законодавчого контролю за тютюном.

Така поширеність проблеми тютюнопаління зумовлена багатьма факторами, серед яких найважливішими є суспільна толерантність до явища тютюнопаління та його інтегрованість в культурне життя українців. Так, в свідомості більшої частини населення, тютюнопаління серед дорослих сприймається не як проблема, яка потребує заходів вирішення, а як особисте рішення кожної людини на рівні способу життя. Вживання тютюну є невід'ємною частиною суспільного життя, прийнятним вважається зображення паління в масовому медіа, персонажами кінофільмів, знаменитостями та політиками. Ще однією запорукою поширеності вживання тютюну є його доступність в різному вигляді : сигарети, електронні приводи нагріву, нюхальний та жувальний тютюн, що збільшує шанси на те, що тютюн знайде свого споживача, та зменшує ефективність законодавчих ініціатив з його контролю. Явище тютюнопаління настільки

укорінилось в нашому суспільстві, що більша частина молодих українців до моменту настання повноліття хоча б раз спробує тютюн у одному з його багатьох виглядів, і кожен – зіткнеться з тютюнопалінням в якості пасивного курця.

Всеосяжна поширеність тютюну в суспільстві переводить проблему з площини «як запобігти доступу українців до тютюну» (оскільки попит на тютюн завжди буде створювати пропозицію, якими б не були жорсткими обмеження) в проблему індивідуального вибору «як допомогти нашим співгромадянам-курцям впоратися з залежністю» (тому що лише усвідомлюючи проблему, людина може позбавитись шкідливої звички).

Однією з головних перепон на шляху подолання залежності є механізм її формування : так, кинути палити настільки ж важко, наскільки легко почати. Саме тому важливо розуміти, як людська психіка взаємодіє з тютюном і як ця взаємодія переростає в згубну залежність. Справа в тім, що робота людського мозку побудована на причинно-наслідковому зв’язку стимулів та реакцій, і в природньому стані людина не піддається впливу таких речовин як нікотин в тютюні та етанол в алкоголі. Ці речовини є джерелом швидкого та доступного задоволення, і мають вплив «супер-стимулу» для мозку. Саме тому формування залежності відбувається стрімкими темпами. З іншого боку, будь-який стимул має тенденцію до послаблення реакції на нього при повторенні (іншими словами, людина звикає до всього), тому для отримання того ж задоволення, мозок потребує більших та частіших доз стимулу. Саме так формується залежність. Людський мозок є дуже гнучким механізмом, тому для самої людини процес залежності залишається непоміченим, оскільки мозок застосовує реакцію раціоналізації. Іншими словами, підсвідомість людини шукає все нові і нові приводи, щоб переконати її свідомість випалити ще одну цигарку, купити ще одну пачку, аж поки людина не перетворюється в щоденного курця. Через механізм раціоналізації курець не усвідомлює як стає курцем, а потім завдяки

йому ж шукає і знаходить причини не кидати згубну звичку. Саме тому методи боротьби з тютюнопалінням, побудовані на раптовому припиненні та утриманні від паління за допомогою лише сили волі не є ефективними, оскільки навіть раціональна частина свідомості курця грає проти нього в щоденній боротьбі зі спокусою зірватися і отримати свою дозу нікотину.

З іншого боку, позитивний підхід, заснований на щоденній роботі над собою і поступовій відмові від тютюну, довів свою ефективність, оскільки дозволяє курцю позбавитись від пастки супер-стимулу та раціоналізації.

Метою даної роботи є підвищення ефективності подолання шкідливої звички тютюнопаління, заснована на щоденному відслідковуванні власного прогресу в боротьбі з залежністю.

Завданням даної роботи є розробка мобільного застосунку для індивідуальної допомоги в боротьбі з тютюнопалінням «SmokeBreak». Цей застосунок покликаний підготувати курця до відмови від тютюнопаління, а також відслідковувати прогрес та переваги відмови у вигляді статистики, досягнень. Мобільний додаток дозволить курцю не втрачати мотивацію у боротьбі з шкідливою звичкою, надасть поради та навчальні матеріали, як ефективніше відмовитися від паління, та які переваги для життя курця матиме позбавлення від тютюну.

Результатом роботи є система, доступна будь-якому користувачеві, що задовільняє увесь функціонал технічного завдання.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

За визначенням вільної енциклопедії Вікіпедії, тютюнопаління – шкідлива звичка вдихання диму диму тліючого висушеного листя тютюну. Компоненти тютюнового диму, такі як нікотин, викликають стан тимчасової ейфорії, запаморочення, і ведуть до розвитку тютюнової залежності, яка негативно впливає на життя курця, його оточуючих та довкілля.

Тютюнопаління спричиняє численні захворювання, серед них серцево-судинні захворювання, онкологічні та респіраторні захворювання, від яких щороку гине 85 000 українців. Тютюнопаління є великою проблемою суспільного масштабу, оскільки 10% загальних смертей в світі пов’язують з проблемами, викликаними цією шкідливою звичкою.

В Україні боротьба з тютюнопалінням зводиться до законодавчих ініціатив, покликаних знизити маркетингову привабливість та економічну доступність тютюнових виробів для населення, але ці заходи неефективні в боротьбі з проблемою поширеності паління серед населення, оскільки не допомагають курцям позбутися шкідливої звички. В Україні не розвинута культура роботи з залежними, вони піддаються стигматизації, а проблема поширеності тютюнопаління серед населення не вивчається та ігнорується.

Індивідуальна боротьба з тютюнопалінням – це комплекс заходів та методик, що допомагає курцеві побороти свою залежність від вживання тютюнових виробів. Індивідуальний підхід до проблеми паління допомагає курцю усвідомити, як тютюнопаління згубно впливає на його життя, прийняти рішення про відмову від тютюнових виробів, не втрачати мотивацію в боротьбі з залежністю.

Після аналізу проблеми поширеності тютюнопаління серед населення України та методик ефективної боротьби з палінням, було прийнято рішення розробити мобільний додаток індивідуальної боротьби з курінням. Формат мобільного додатку було обрано через доступність та простоту у

використанні звичайним користувачем, оскільки мобільний телефон завжди під рукою у курця, що дозволить контролювати і боротися з залежністю у повсякденному житті.

1.2 Змістовний опис і аналіз предметної області

Для аналізу предметної області індивідуальної боротьби з тютюнопалінням було досліджено проблеми, які заважають курцеві позбавитися залежності, та методики, які допомогають їх вирішити.

Основною проблемою, що заважає курцеві позбавитися залежності є страх перед відмовою від тютюну. Залежний усвідомлює, що процес боротьби з залежністю є складним і тривалим, тому підсвідомо шукає причини продовжувати свою шкідливу звичку, або, хоч і прийнявши рішення про відмову від цигарок, відтягують цей момент. Позбавитися від цього страху можна лише усвідомленням залежності курця від тютюну та шкоди, яку він наносить власному здоров'ю та здоров'ю оточуючих його людей.

Іншою проблемою, що стоїть між курцем і поверненням до здорового образу життя, є недостатність сили волі. У решти такі спроби закінчуються нервовими зривами, відчаєм та поверненням до нікотинової пастки. Тому для ефективної боротьби з залежністю середньостатистичній людині необхідні методи і засоби, що зможуть полегшити стан психологічної залежності та звести її до нуля.

Є багато методик для подолання тютюнової залежності, деякі з них потребують втручання дипломованого лікаря-спеціаліста. Однак, не зважаючи на те, що одного вольового зусилля курця буває недостатньо, твердження, що кинути палити самостійно, без допомоги спеціалістів, неможливо, є не невірним. Звернувшись до методів індивідуальної боротьби з тютюнопалінням, подолати залежність може кожен.

На рисунку 1 зображена схема подолання психологічної залежності від тютюнопаління, представлена в книзі Аллена Карра «Легкий спосіб кинути палити». Ця методика побудована на подоланні страху перед процесом

розлучення з цигарками, та побудові розуміння, що задоволення від паління – це міф. Метод Аллена Карра довів свою ефективність, з його допомогою змогло подолати залежність 95% бажаючих. В рамках проекту дослідницький інтерес ця методика представляє тому що дозволяє побороти залежність самостійно і є повністю безпечною для організму.

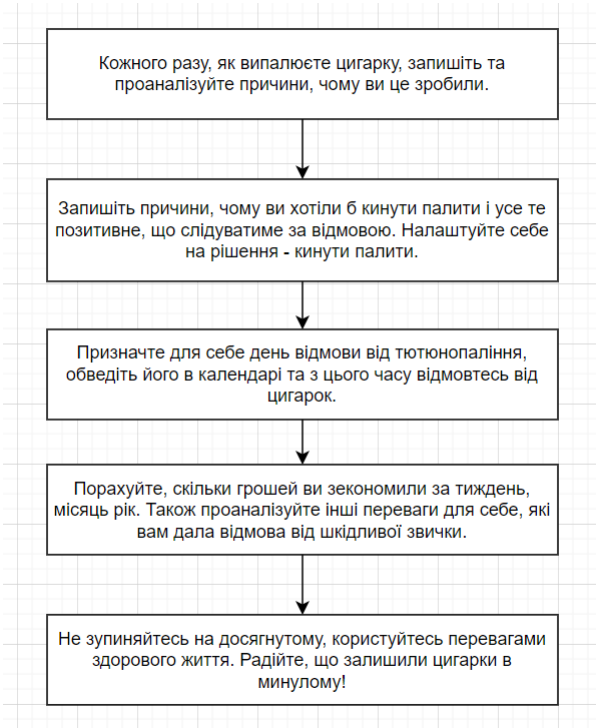

Рисунок 1.1 – Схема самостійного подолання тютюнової залежності

Методики боротьби з залежністю можна поділити на два типа : методи одночасної відмови від паління та методи поступової відмови. Перший тип краще підійде для людей з розвинутою силою волі, проте підвищується ризик зірватися, в той час як другий тип довзоляє організму курця поступово призвичаїтись до відмови від тютюну, але потребує щоденного планування та роботи над собою.

Після аналізу методів індивідуальної боротьби з тютюновою залежністю було виділено найпрактичніші :

- Відслідковування прогресу відмови та користі для здоров’я. Маючи наявний приклад того, який шлях пройшов колишній курець від відмови та які переваги для здоров’я означає життя без тютюну, курець буде мати більше мотивації не повертатись до шкідливої звички. Цей метод ефективний для курців, які вирішили кинути палити одразу.
- Ведення щоденника паління. Ставлячи для себе цілі та виконуючи їх, курець матиме змогу відслідковувати свій поступовий прогрес на шляху до повного подолання психологічної залежності від цигарок. Також дозволяє записувати причини, через які людина палить цигарки та проаналізувати їх. Цей спосіб підійде курцям, що планують подолати залежність поступово.
- Довідкові матеріали. Важливо, щоб курець гарно усвідомлював шкоду, яку наносить його здоров’ю тютюнопаління, і користь для здоров’я, яку принесе відмова від тютюну.
- Винагорода за досягнення. Замінивши задоволення від тютюнопаління відчуттям гордості за прогрес в подоланні залежності, можна послабити негативні ефекти синдрому відмови.

Повне позбавлення залежності від тютюнопаління повністю залежить від мотивації курця, проте за допомогою методик індивідуальної боротьби з палінням можна зробити цей процес набагато ефективнішим.

В процесі боротьби з тютюновою залежністю доцільним помічником стане мобільний застосунок, який дозволить ефективно відслідковувати ваш прогрес, стане джерелом довідки методів ефективної відмови від паління та надасть мотивацію не повертатись до згубної звички.

Повне позбавлення залежності від тютюнопаління повністю залежить від мотивації курця, проте за допомогою методик індивідуальної боротьби з палінням можна зробити цей процес набагато ефективнішим.

В процесі боротьби з тютюновою залежністю доцільним помічником стане мобільний застосунок, який дозволить ефективно відслідковувати ваш прогрес, стане джерелом довідки методів ефективної відмови від паління та надасть мотивацію не повертатись до згубної звички.

1.3 Аналіз успішних IT-проектів

Сьогодні популярність мобільних застосувань зростає все більше і більше, вони широко використовуються в будь-якій сфері сучасного життя, від розваг до освіти. Щодня мільйони людей встановлюють та використовують мобільні застосування.

Упродовж багатьох років популярними мобільними операційними системами були Android від Google, iOS від Apple та Windows Phone від Microsoft.

Розглянемо кілька аналогів продуктів із магазину застосувань Google Play. Багато програм для боротьби з залежностями, представлених на ринку, надають користувачам змогу відслідковувати свій прогрес, переглянути переваги для здоров’я, отримати мотивацію не повертатись до шкідливої звички та дізнатись корисні матеріали, що допоможуть їм ефективніше побороти залежність. Розглянемо наступні існуючі мобільні застосування : Не П’ю, I’m sober, QuitNow!, Flamy.

1.3.1 Аналіз існуючих програмних продуктів

Не п’ю

Додаток, що допомагає людям в боротьбі з алкоголем. Є можливість встановлювання цілей і стеження за їх досягненням, відслідковування поліпшенням здоров'я. Можливе використання без реєстрації.

Наведено корпус статтей з описом захворювань, на виникнення яких впливає алкоголь, порадами тим, хто бореться з алкоголізмом, перевагами тверезого життя.

Серед функціоналу також наявний калькулятор алкоголю в крові за введеними даними за певний період часу.

Додаток відмінно справляється зі своєю функцією, проте підійде лише для алкозалежних користувачів.

Мова програми - російська, є платний контент.

На рисунку 1.2 наведені скріншоти застосунку Не п’ю.

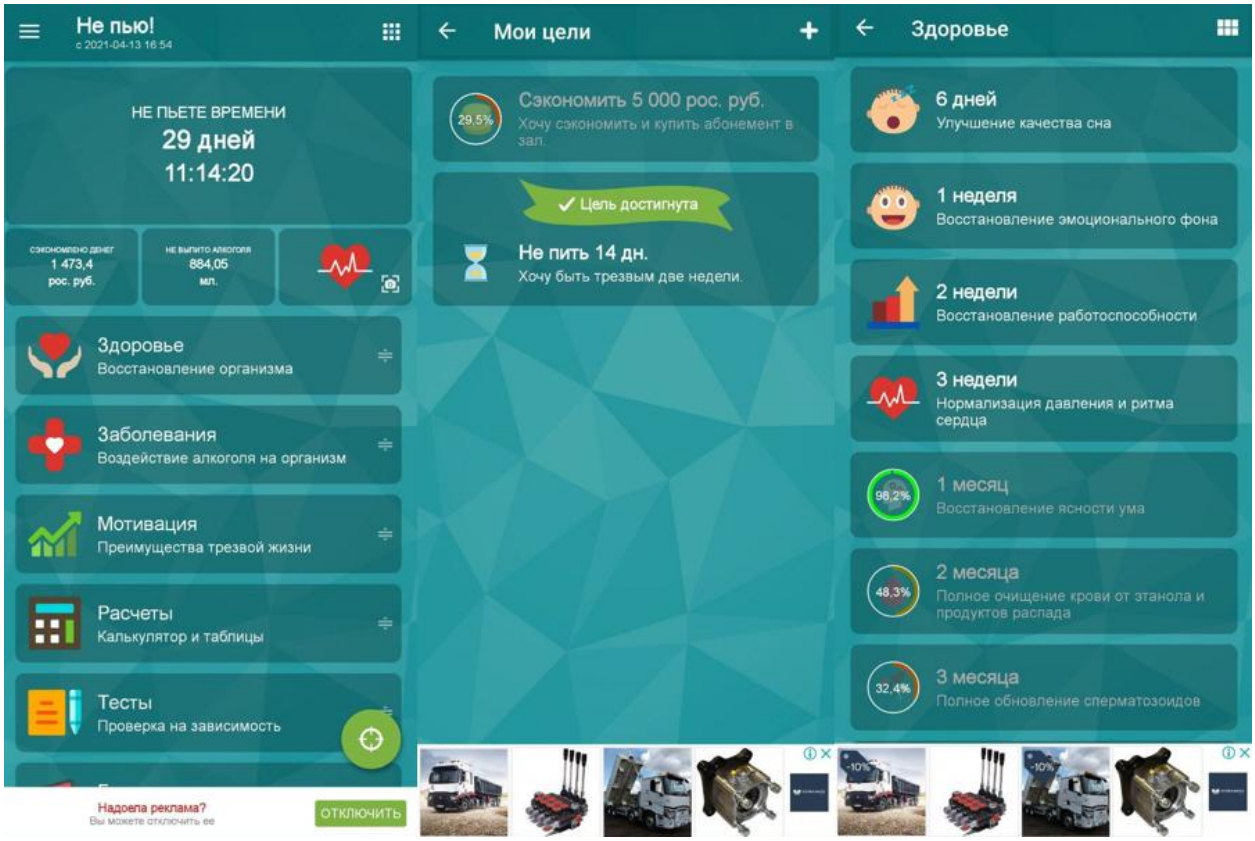

Рисунок 1.2 – Скріншоти застосунку «Не п’ю»

I’m sober

Додаток, що допомагає людям боротися з широким списком залежностей, навіть з декількома одночасно. В цьому є його основна перевага – універсальність, і недолік – відсутність конкретики.

Програма дозволяє відслідковувати прогрес та користь для здоров’я з часу останнього зриву, але оскілька основна орієнтація застосунку – наркозалежні, не передбачено можливість поступової відмови від паління та відслідковування історії зривів.

Застосунок дозволяє давати обіцянки, що нагадують користувачеві про важливість боротьби з залежністю, надає мотивуючі матеріали, що будуть відкриватися користувачеві по мірі користування та доступ до спільноти залежних, яке, однак не диференційоване за типами залежності, тому курець та залежний від амфетаміну спілкуватимуться в спільній групі.

Програма має приємний, зручний для користування дизайн з можливістю налаштування.

Для доступу до повного функціоналу необхідно оформити платну підписку, є реклама.

Мова програми – на вибір, однак спільнота майже повністю англомовна. Українська мова відсутня.

На рисунку 1.3 наведені скріншоти застосунку «I’m sober».

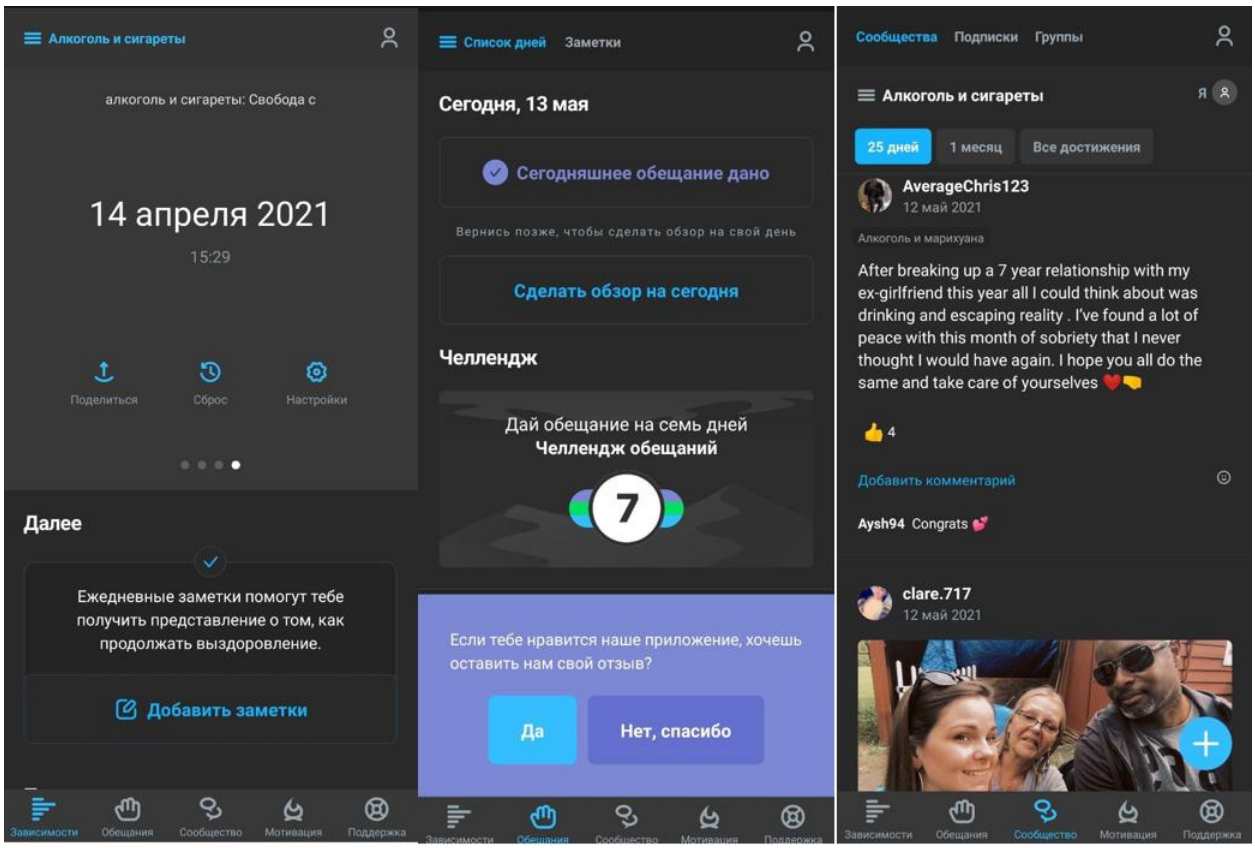

Рисунок 1.3 – Скріншоти застосунку «I’m sober»

QuitNow!

Додаток, що допомагає людям кинути палити.

Необхідна реєстрація, яка дає доступ до спільноти курців, які діляться враженнями та порадами з приводу відмови від паління.

Є система досягнень, відслідковування кількості днів після відмови від паління, кількості невипалених цигарок, зекономлених грошей.

Однак для доступу до чату і повного функціоналу необхідне оформлення платної підписки, користування без реєстрації не є можливим.

Додаток добре справляється з своєю функцією, проте підійде лише для курців, що планують відмовитися від тютюнопаління одразу, можливості поступової відмови та відстеження історії зривів немає.

Мова програми – на вибір, українська відсутня.

На рисунку 1.4 наведені скріншоти застосунку «Quit Now!».

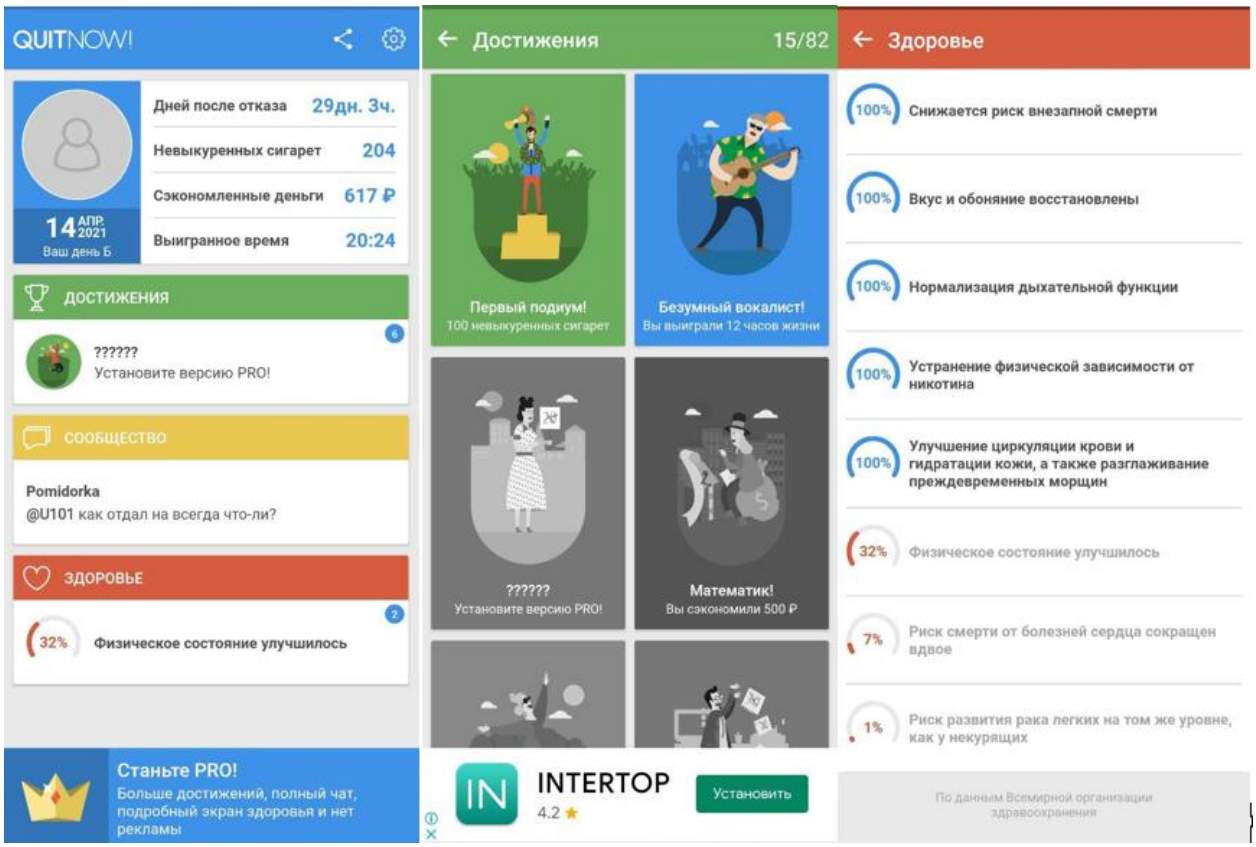

Рисунок 1.3 – Скріншоти застосунку «I’m sober»

Flamy

Додаток, що допомагає людям кинути палити.

Необхідна реєстрація для доступу до функціоналу.

Програма дозволяє встановити дату коли користувач планує кинути палити і надає поради як підготуватись до цього рішення, після цього на основі введених курцем даних про залежність відслідковує статус його реабілітації після залежності та надає такі дані, як час з початку відмови від тютюну, кількість непалених цигарок, зекономлених грошей та часу.

Застосунок реалізує гнучку систему досягнень та можливість поділитися досягненнями через соціальні мережі. Досягненнями підкріплені фактами про покращення здоров’я курця після відмови на протязі часу. Також є можливість запрошувати друзів курців та кидати виклики, щоб користувачі могли допомогати одне одному в боротьбі з залежністю.

Мова програми – на вибір. Українська відсутня.

На рисунку 1.5 наведені скріншоти застосунку «Flamy».

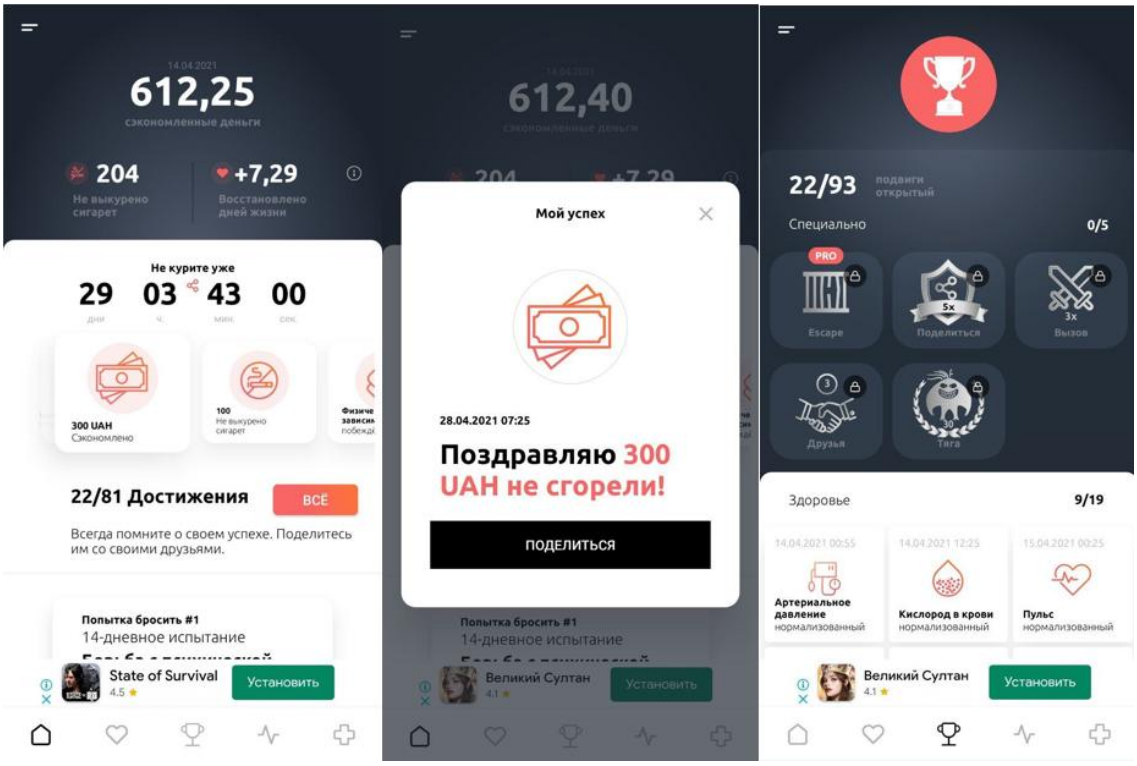

Рисунок 1.4 – Скріншоти застосунку «Flamy»

Результатом аналізу успішних ІТ-проектів є висновок про наявність у застосунках, поширених на ринку таких недоліків, як обмежена функціональність, відсутність української мови, необхідність передплати за користування, не передбаченість можливості поступової відмови від паління користувачем-курцем, необхідність підключення до мережі Інтернет, наявність реклами. У застосунку, що розробляється під час даної дипломної роботи, був зроблений акцент на вирішення проблем, що виникають при роботі з продуктами-аналогами.

1.4 Аналіз вимог до програмного забезпечення

Для визначення вимог до програмного забезпечення необхідно вияснити ролі користувачів та їх можливості в системі. Система повинна містити наступні типи користувачів:

– користувач.

В програмному забезпеченні не буде передбачено створення адміністративних ролей тож кожен користувач буде мати однакову роль і зможе мати доступ до усієї функціональності.

1.4.1 Розроблення функціональних вимог

Користувач має можливість заповнити профіль курця (UC01), редагувати профіль курця(UC02), додати нову ціль до списку цілей (UC03), переглянути список цілей з прогресом їх виконання (UC04), видалити ціль зі списку цілей (UC05), переглянути свій прогрес відмови від паління, в тому числі статистику по показникам та досягнення користі для здоров’я (UC06), додати новий зрив в щоденник паління (UC07), переглянути щоденник паління з інформацією по дням та зривам (UC08), обрати та пройти тест з переліку тестувань (UC09), обрати та переглянути допис з довідкових матеріалів за категорією (UC10).

Варіанти використання, які передбачені в даному застосунку наведено в таблицях 1-10.

Таблиця 1.1 - Опис варіанту використання UC01

Атрибут	**Значення**
Use Case ID	UC01
Use Case Name	Заповнення профілю курця
Use Case Description	Цей варіант використання описує механізм заповнення користувачем профілю курця
Primary Actors	Користувач
Secondary Actors	-
Preconditions	-
Postconditions	Введені дані користувачем збережені до профілю курця
Trigger	Користувач вперше запускає додаток.
Main flow	1) Застосунок демонструє вікно профілю курця з полями : дата і час останньої випаленої цигарки, середня кількість цигарок в день (шт), ціна пачки (грн), кількість цигарок в пачці (шт), вміст смоли (мг/циг), вміст нікотину (мг/циг). 2) Користувач заповнює значення показників. 3) Користувач натискає кнопку «Підтвердити». 4) Застосунок переходить на вікно «Головна». 5) Варіант виконання завершує свою роботу.
Alternative flows	-
Exception flows	-
Result	Введені дані користувачем збережені до профілю курця

Таблиця 1.2 - Опис варіанту використання UC02

Атрибут	Значення
Use Case ID	UC02

Use Case Name	Редагування профілю курця
Use Case Description	Цей варіант використання описує механізм редагування користувачем профілю курця
Primary Actors	Користувач
Secondary Actors	-
Preconditions	Користувач заповнив профіль курця
Postconditions	Введені дані користувачем збережені до профілю курця
Trigger	Користувач обирає в меню застосунку вкладку «Профіль курця».
Main flow	1) Застосунок демонструє вікно профілю курця з значеннями : дата і час останньої випаленої цигарки, середня кількість цигарок в день (шт), ціна пачки (грн), кількість цигарок в пачці (шт), вміст смоли (мг/циг), вміст нікотину (мг/циг). В ці поля попередньо занесені дані, введені користувачем під час заповнення або останнього редагування профілю курця. 2) Користувач редагує значення. 3) Користувач натискає кнопку «Підтвердити». 4) Застосунок демонструє повідомлення про те, що дані профілю курця успішно оновлені. 5) Варіант виконання завершує свою роботу.
Alternative flows	-
Exception flows	-

Продовження таблиці 1.2

Result	Введені дані користувачем збережені до профілю

	курця

Таблиця 1.3 - Опис варіанту використання UC03

Атрибут	**Значення**
Use Case ID	UC03
Use Case Name	Додання нової цілі
Use Case Description	Цей варіант використання описує механізм додання нової цілі в список цілей
Primary Actors	Користувач
Secondary Actors	-
Preconditions	-
Postconditions	Нова ціль додана до списку цілей
Trigger	Користувач переходить на сторінку «Мої цілі» та натискає кнопку «Додати ціль».
Main flow	1) Застосунок демонструє вікно створення цілі. 2) Користувач обирає категорію з переліку : час відмови, кількість невипалених цигарок, зекономлені кошти. 3) Застосунок демонструє вікно створення нової цілі з заголовком цілі та полем вводу, формат яких залежить від обраної категорії (час відмови – роки/місяці/дні, кількість невипалених цигарок – шт., зекономлені кошти – грн). Це поле вводу є попередньо заповненим і не може бути попрожнім. Також вікно містить порожнє поле вводу для опису цілі, яке Користувач може заповнити за бажанням.

Продовження таблиці 1.3

	4) Користувач встановлює цільове значення. 5) Користувач натискає кнопку «Додати». 6) Застосунок додає створену ціль до списку цілей. 7) Застосунок переходить на вікно «Мої цілі». Варіант виконання завершує свою роботу.
Alternative flows	-
Exception flows	-
Result	Нова ціль додана до списку цілей

Таблиця 1.4 - Опис варіанту використання UC04

Атрибут	**Значення**
Use Case ID	UC04
Use Case Name	Перегляд списку цілей
Use Case Description	Цей варіант використання описує механізм перегляду списку цілей
Primary Actors	Користувач
Secondary Actors	-
Preconditions	Користувач заповнив профіль курця
Postconditions	Користувач переглядає список цілей з прогресом їх виконання
Trigger	Користувач переходить на сторінку «Мої цілі»
Main flow	1) Застосунок демонструє вікно зі списком доданих цілей. Список містить ячейки з заголовком цілі, описом, якщо Користувач вводив його при додаванні, та прогресом виконання у вигляді відсотків від 0 до 100. Прогрес виконання

Продовження таблиці 1.4

	обраховується на основі данних, введених в профіль курця. 2) Користувач натискає на ячейку з окремою ціллю. 3) Застосунок демонструє модальне вікно з заголовком цілі, повним описом, іконкою, що відповідає відповідній категорії цілі, та прогрес виконання у вигляді відсотків від 0 до 100, Також модальне вікно містить кнопку «Видалити». Варіант виконання завершує свою роботу.
Alternative flows	1a. Додаток не містить доданих цілей 1) Застосунок демонструє повідомлення, що список цілей порожній і пропозицією додати нову ціль. 2) Відбувається варіант використання UC03.
Exception flows	-
Result	Користувач переглядає список цілей з прогресом їх виконання

Таблиця 1.5 - Опис варіанту використання UC05

Атрибут	**Значення**
Use Case ID	UC05
Use Case Name	Видалення цілі зі списку цілей
Use Case Description	Цей варіант використання описує механізм видаленні цілі зі списку цілей
Primary Actors	Користувач
Secondary Actors	-
Preconditions	Користувач заповнив профіль курця, список цілей містить цілі.

Продовження таблиці 1.5

Postconditions	Ціль видалена зі списку цілей
Trigger	Користувач переходить на сторінку «Мої цілі»
Main flow	1) Застосунок демонструє вікно зі списком доданих цілей та прогресом їх виконання. 2) Користувач натискає на ячейку з окремою ціллю. 3) Застосунок демонструє модальне вікно, яке містить кнопку «Видалити». 4) Користувач натискає кнопку «Видалити». 5) Застосунок видаляє обрану ціль з списку цілей. 6) Застосунок прибирає з екрану модальне вікно фокус повертається до вікна зі списком цілей. 7) Варіант виконання завершує свою роботу.
Alternative flows	6а. Користувач натискає на область за межами модального вікна. 1) Керування переходить на крок 8 основного потоку
Exception flows	-
Result	Ціль видалена зі списку цілей

Таблиця 1.6 - Опис варіанту використання UC06

Атрибут	**Значення**
Use Case ID	UC06
Use Case Name	Перегляд прогресу відмови від паління
Use Case Description	Цей варіант використання описує механізм перегляду прогресу відмови від паління
Primary Actors	Користувач
Secondary Actors	-

Продовження таблиці 1.6

Preconditions	Користувач заповнив профіль курця.
Postconditions	Користувач переглядає свій прогрес відмови від паління.
Trigger	Користувач переходить на сторінку «Головна»
Main flow	1) Застосунок демонструє вікно з дашбордом, який відображає дату і час останньої випаленої цигарки, час з моменту відмови, кількість невипалених цигарок, обсяг зекономлених коштів, обсяг зекономленого часу. Ці показники прогресу обчислюються на основі даних, заповнених Користувачем в профілі курця. 2) Користувач натискає на кнопку «Статистика». 3) Застосунок демонструє вікно зі списком середніх значень показників відмови від паління за весь час на основі історії зривів, прогресу на момент зриву та поточні значення показників відмови. 4) Користувач натискає кнопку «Досягнення». 5) Застосунок демонструє вікно із досягненнями - списком переваг для здоров’я та відповідним їм часом відмови від паління. На основі дати останньої випаленої цигарки, досягнення можуть бути розблокованими або заблокованими, в останньому випадку також відображається прогрес отримання досягнення. 6) Варіант виконання завершує свою роботу.
Alternative flows	-
Exception flows	-

Продовження таблиці 1.6

Result	Користувач переглядає свій прогрес відмови від паління.

Таблиця 1.7 - Опис варіанту використання UC07

Атрибут	**Значення**
Use Case ID	UC07
Use Case Name	Додання зриву в щоденник паління
Use Case Description	Цей варіант використання описує механізм додання зриву в щоденник паління
Primary Actors	Користувач
Secondary Actors	-
Preconditions	Користувач заповнив профіль курця.
Postconditions	Новий зрив збережено до щоденнику паління, дату останньої випаленої цигарки в профілі курця оновлено, дашборд показників прогресу відмови від паління оновлено.
Trigger	Користувач переходить на сторінку «Головна»
Main flow	1) Застосунок демонструє вікно з лічильником часу з дати останньої випаленої цигарки, що заповнена Користувачем в профілі курця або встановлена доданням зриву раніше. 2) Користувач натискає на кнопку «Скинути лічильник». 3) Застосунок демонструє модальне вікно з запитом на підтвердження додання зриву та обнулення лічильнику часу відмови. Також вікно містить поле

Продовження таблиці 1.7

	вводу для причини зриву, яке Користувач може заповнити за бажанням. 4) Користувач натискає кнопку «Підтвердити». 5) Застосунок зберігає зрив до щоденнику паління. 6) Застосунок оновлює дату останньої випаленої цигарки в профілі курця на поточну. 7) Застосунок прибирає модальне вікно, фокус переноситься на вікно «Головна». Дашборд з показниками прогресу відмови від паління тепер оновлено згідно з датою останньої випаленої цигарки. 8) Варіант виконання завершує свою роботу.
Alternative flows	4a. Користувач натискає на кнопку «Відмінити». 1) Керування переходить на крок 7 основного потоку.
Exception flows	-
Result	Новий зрив збережено до щоденнику паління, дату останньої випаленої цигарки в профілі курця оновлено, дашборд показників прогресу відмови від паління оновлено.

Таблиця 1.8 - Опис варіанту використання UC08

Атрибут	**Значення**
Use Case ID	UC08
Use Case Name	Перегляд щоденнику паління
Use Case Description	Цей варіант використання описує механізм переглядц щоденнику паління

Продовження таблиці 1.8

Primary Actors	Користувач
Secondary Actors	-
Preconditions	Користувач заповнив профіль курця.
Postconditions	Користувач переглянув щоденник паління
Trigger	Користувач переходить на сторінку «Щоденник паління» в меню
Main flow	1) Застосунок демонструє вікно, на якому зображений календар щоденнику паління за поточний місяць. Дні, коли не було відмічено нових зривів, позначено зеленим, дні, коли кількість зривів менше за 3- жовтим, дні, за які було випалено більше цигарок ніж 3 – червоним. 2) Користувач обирає окремий день в календарі. 3) Застосунок демонструє панель з інформацією по зривам за обраний день - список зривів, якщо такі наявні. Список зривів містить такі дані : час зриву, причина зриву, якщо така була вказана. 4) Варіант виконання завершує свою роботу.
Alternative flows	2a. Користувач обирає день, записів за який немає в щоденнику паління. 1) Застосунок демонструє панель з повідомленням, що за обраний день немає даних. 2) Керування переходить на крок 1 основного потоку.
Exception flows	-
Result	Користувач переглянув щоденник паління.

Таблиця 1.9 - Опис варіанту використання UC09

Атрибут	**Значення**
Use Case ID	UC09
Use Case Name	Вибір тесту для проходження
Use Case Description	Цей варіант використання описує механізм вибору тесту для проходження.
Primary Actors	Користувач
Secondary Actors	-
Preconditions	-
Postconditions	Користувач отримав результат тестування згідно з його відповідями
Trigger	Користувач переходить на сторінку «Тестування» в меню
Main flow	1) Застосунок демонструє список тестів на вибір . 2) Користувач обирає один з трьох тестів. 3) Застосунок демонструє вікно з запитанням та варіантами відповіді на них. 4) Користувач обирає варіант відповіді та натискає кнопку «Далі». 5) Застосунок демонструє наступне запитання, доки Користувач не відповість на останнє питання. 6) Застосунок демонструє користувачу результат проходження тесту. 7) Варіант виконання завершує свою роботу.
Alternative flows	4а) Користувач натискає кнопку «Назад». 1) Керування переходить на крок 1 основного потоку.
Exception flows	-

Продовження таблиці 1.9

Result	Результат проходження тестування користувачем

Таблиця 1.10 - Опис варіанту використання UC10

Атрибут	**Значення**
Use Case ID	UC10
Use Case Name	Перегляд розділу довідки
Use Case Description	Цей варіант використання описує механізм перегляду розділу довідки.
Primary Actors	Користувач
Secondary Actors	-
Preconditions	-
Postconditions	Користувач переглянув розділ довідки.
Trigger	Користувач переходить на сторінку «Довідка» в меню
Main flow	1) Застосунок демонструє перелік розділів статей з відповідною проблематикою : методи відмови, факти про паління, поради для тих, хто бажає кинути палити. 2) Користувач обирає розділ довідки. 3) Застосунок демонструє перелік дописів за обраною категорією з заголовком і коротким описом. 4) Користувач обирає допис і натискає на нього. 5) Застосунок демонструє модальне вікно з заголовком допису та повним текстом статті. 6) Користувач закриває модальне вікно натиском на кнопку «Закрити». 7) Варіант виконання завершує свою роботу.
Alternative flows	3а) Користувач натискає кнопку «Назад».

Продовження таблиці 1.10

	1) Керування переходить на крок 1 основного потоку.
Exception flows	-
Result	Перегляд Користувачем розділу довідки.

Функціональні вимоги – це вимоги до ПЗ, які описують внутрішню роботу системи. В даному застосунку було визначено такі функціональні вимоги, що зазначені в таблицях 1.11 – 1.26 :

Таблиця 1.11. - Опис функціональної вимоги REQ01

ID	REQ01
Name	Збереження даних профілю курця до бази даних
Description	Система повинна надавати можливість користувачу вказати дату останньої випаленої цигарки, середню кількість випалюваних цигарок в день, вартість однієї пачки цигарок, кількість цигарок в пачці, та вміст смол в одній цигарці, та зберігати вказані значення в базі даних.

Таблиця 1.12. - Опис функціональної вимоги REQ02

ID	REQ02
Name	Збереження цілей до бази даних
Description	Система повинна надавати можливість користувачу створення нових цілей за одним з трьох категорій : кількість не випалених цигарок в день, об’єм зекономлених коштів, об’єм зекономленого часу, встановлювати критерій виконання цілі в залежності від категорії, додавати опис до цілі, та зберігати ці дані в базі даних.

Таблиця 1.13. - Опис функціональної вимоги REQ03

ID	REQ03
Name	Перегляд прогресу виконання цілей
Description	Система повинна надавати можливість користувачу перегляду списку створених цілей з їх описом, прогресом та статусом виконання, які обраховується застосунком на основі даних профілю курця та дати останньої випаленої цигарки.

Таблиця 1.14. - Опис функціональної вимоги REQ04

ID	REQ04
Name	Отримання сповіщення про виконання цілі
Description	При завершенні прогресу виконання цілі, система має надіслати сповіщення користувачеві.

Таблиця 1.15. - Опис функціональної вимоги REQ05

ID	REQ05
Name	Видалення цілі з бази даних
Description	Система повинна надавати можливість користувачу видаляти цілі зі списку цілей, таким чином видаляючи їх з бази даних.

Таблиця 1.16. - Опис функціональної вимоги REQ06

ID	REQ06
Name	Перегляд прогресу відмови від куріння
Description	Система повинна надавати можливість користувачу переглянути його прогрес відмови від паління : кількість невипалених цигарок, кількість зекономлених коштів,

Продовження таблиці 1.16

	кількість зекономленого часу та кількість неспожитих смол, який обраховується на основі даних профілю курця та дати останньої випаленої цигарки.

Таблиця 1.17. - Опис функціональної вимоги REQ07

ID	REQ07
Name	Ведення статистики по показникам прогресу відмови
Description	Система повинна вести облік статистики по показникам прогреу відмови : середній час відмови, середній об’єм зекономлених коштів, середній об’єм зекономленого часу та середній обсяг неспожитих смол, зберігати їх до бази даних та надавати користувачеві можливість перегляду.

Таблиця 1.18. - Опис функціональної вимоги REQ08

ID	REQ08
Name	Відслідковування досягнень користі для здоров’я
Description	Система повинна містити перелік переваг для здоров’я від відмови від паління та вести їх статус та прогрес виконання на основі дати останньої випаленої цигарки.

Таблиця 1.19. - Опис функціональної вимоги REQ09

ID	REQ09
Name	Оновлення дати останньої випаленої цигарки
Description	Система повинна надавати можливість зміни дати останньої випаленої цигарки за допомогою додавання нового зриву та вручну в профілі курця.

Таблиця 1.20. - Опис функціональної вимоги REQ10

ID	REQ10
Name	Додання зриву до щоденнику паління
Description	Система повинна надавати можливість користувачеві додати новий зрив та його короткий опис, який можна буде переглянути в щоденнику паління.

Таблиця 1.21. - Опис функціональної вимоги REQ11

ID	REQ11
Name	Ведення обліку випалених цигарок за день
Description	Система повинна вести облік випалених цигарок по дням.

Таблиця 1.22. - Опис функціональної вимоги REQ12

ID	REQ12
Name	Перегляд щоденника паління
Description	Система повинна надавати користувачеві можливість перегляду зривів з їх описом за обраним днем в щоденнику.

Таблиця 1.23. - Опис функціональної вимоги REQ13

ID	REQ13
Name	Забезпечення вибору тестування з переліку тестів
Description	Система повинна надавати користувачеві можливість обрати тест для проходження з переліку тестів.

Таблиця 1.24. - Опис функціональної вимоги REQ14

ID	REQ14
Name	Обробка тестових відповідей

Продовження таблиці 1.24

Description	Система повинна надавати користувачеві можливість обрати варіант відповіді на тестове запитання та обробляти результати тесту з виводом

Таблиця 1.25. - Опис функціональної вимоги REQ15

ID	REQ15
Name	Забезпечення вибору розділу з каталогу довідкової інформації
Description	Система повинна надавати користувачеві можливість обрати розділ довідки з трьох категорій : методи відмови від паління, факти про паління, поради для тих, хто бажає кинути палити.

Таблиця 1.26. - Опис функціональної вимоги REQ17

ID	REQ16
Name	Імпорт дописів
Description	Система повинна завантажувати дописи за обраною категорією з відповідної таблиці в базі даних.

Таблиця 1.27. - Матриця трасування між функціональними вимогами та варіантами використання

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10
REQ01	■	■								
REQ02			■							
REQ03				■						
REQ04				■						
REQ05					■					
REQ06						■				
REQ07						■				
REQ08						■				
REQ09							■			
REQ10	■	■					■			
REQ11								■		
REQ12								■		
REQ13									■	
REQ14									■	
REQ15										■
REQ16										■

1.4.2 Розроблення нефункціональних вимог

До розроблюваного програмного забезпечення висуваються наступні нефункціональні вимоги :

- система має бути реалізована з використанням мови програмування Java. Мінімальна версія – Java 8;
- версія ОС для мобільних додатків Android 4.1 та вище;

– мінімальна версія Andoid SDK – 30;
– локалізація інтерфейсу українською мовою;
– в якості системи управляння бази даних об’єктів повинен застосовуватись Realm;

1.4.3 Постановка комплексу задач модулю

Основним призначенням розроблюваного програмного забезпечення є створення застосунку покликананого підготувати курця до відмови від тютюнопаління, а також відслідковувати прогрес та переваги відмови у вигляді статистики, досягнень.

Мета – застосувати набуті в процесі навчання комплексі знання різних технічних дисциплін для побудови мобільного застосування, який допоможе курцю кинути паління, надасть мотивацію та інструменти відслідковування власного прогресу, а також зробить процес позбавлення шкідливої звички інформативним та інтерактивним.

Система має надавати користувачеві можливість виконувати наступні задачі :

– Заповнення профілю курця.
– Редагування профілю курця.
– Додавання нових цілей.
– Відслідковування прогресу виконання цілей.
– Видалення існуючих цілей.
– Перегляд прогресу відмови від паління.
– Перегляд статистики по показникам прогресу відмови від паління та досягнення користі для здоров’я.
– Додання нового зриву в щоденник паління.
– Перегляд щоденнику паління.
– Проходження тестів з переліку тестувань.
– Переглядь довідкових матеріалів за категорією.

Мобільний застосунок мусить працювати на пристроях з операційною системою на базі Android

1.5 Висновки до розділу

В рамках першого розділу було розглянуто особливості предметної області, опис сучасних методів вирішення поставленої проблематики. Розглянуто характеристики і недоліки найближчих по своїй суті продуктів. Сформульовано функціональні і нефункціональні вимоги до програмного забезпечення.

Поширення шкідливої звички тютюнопаління серед молодого та дорослого населення України є актуальною проблемою, яка не має простого вирішення, оскільки не може бути вирішена ні законодавчими ініціативами, ні традиційними методами боротьби з залежностями. Ця проблема є комплексною, оскільки у кожного курця своя історія залежності, що має індивідуальну специфіку, і в багатьох випадках не може бути подолана курцем вольовим зусиллям та без допомоги. Тому було поставлено за мету розробку мобільного додатку індивідуальної боротьби з тютюнопалінням «SmokeBreak», який би допоміг курцям у подоланні залежності, розвинути мотивацію та відслідковувати прогресс, оскільки лише за допомогою вмотивованої щоденної боротьби з залежністю можна досягти успіху в її подоланні. Формат мобільного додатку був обраний оскільки мобільний телефон завжди присутній у житті курця і він зможе користуватись застосунком будь де.

В першому розділі була описана специфіка тютюнової залежності та відомі способи її подолання.

Було проведено аналіз існуючих програмних продуктів за предметною областю, були визначені їх переваги та недоліки. На основі аналізу було визначено проблеми, що виникають при роботі з додатками, що вже присутні на ринку програмного забезпечення, та зроблено висновок щодо необхідності вирішення цих проблем в розроблюваному програмному продукті.

На основі проведеного аналізу предметної області були сформовані варіанти використання, функціональні та нефункціональні вимоги до розроблюваного програмного продукту. На основі порівняння функціональних вимог та можливих сценаріїв використання була створена матриця залежності між вимогами застосунку і варіантами використання, яка в свою чергу показала , що створений розроблюваний програмний продукт повністю задовільняє висунуті до нього вимоги.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для створення якісного програмного забезпечення необхідно попереднє детальне моделювання його бізнес процесів і архітектури. Для проектування бізнес-процесів зручно використовувати методологію створення діаграм BPMN.

BPMN (англ. Business Process Model and Notation) – це система умовних позначень для моделювання бізнес-процесів. Дана нотація дає можливість визначати складну семантику бізнес-процесів

Основною задачею BPMN -діаграм є моделювання бізнес-процесів, що є проміжним етапом між формалізацією/візуалізацією та втіленням бізнес процесу. Така нотація представляє собою опис графічних елементів, які використовуються для побудови схеми протікання бізнес-процесу. При цьому модель бізнес-процесу повинна бути зрозумілою для всіх користувачів.

У ході аналізу та моделювання програмного забезпечення, було виділено декілька основних бізнес-процесів:

- додання нової цілі до списку цілей;
- додання нового зриву до щоденнику паління;
- проходження тестування на вибір;
- використання розділу довідки.

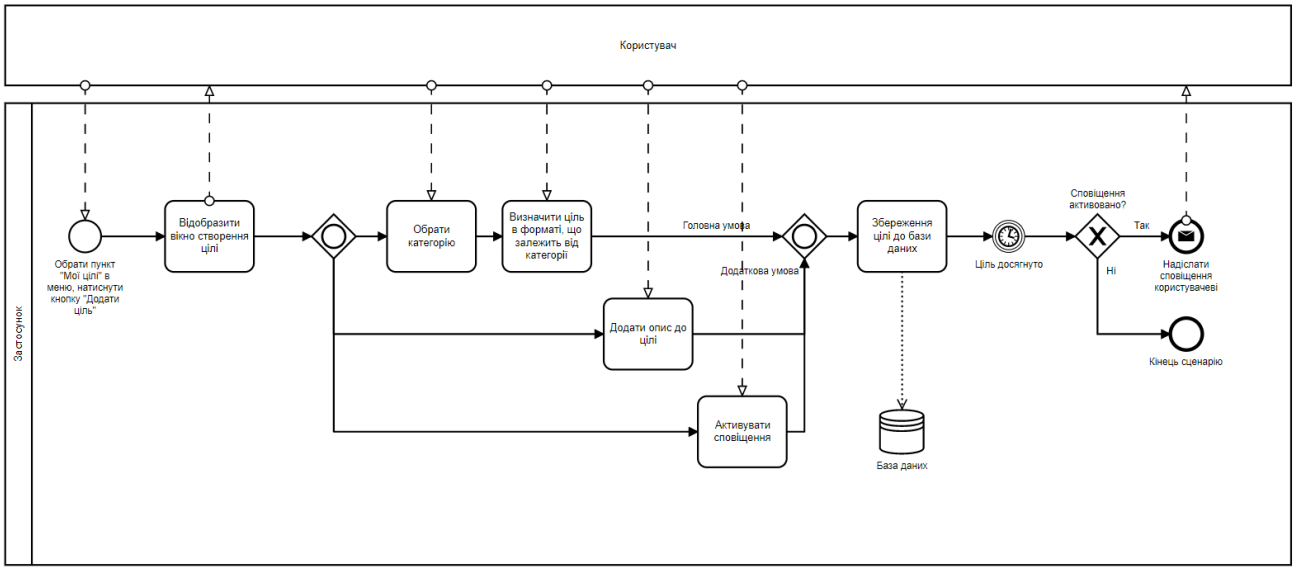

Рисунок 2.1 BPMN-діаграма для опису бізнес-процесу додання нової цілі

Послідовний опис бізнес-процесу додання нової цілі до списку цілей :

- користувач інаціалізує виконання сценарію, перейшовши до вікна «Мої цілі» та натискаючи кнопку «Додати ціль»;
- додаток відображає вікно додання нової цілі;
- користувач обирає категорію, задає цільове значення в форматі, що залежить від категорії;
- користувач додає короткий опис цілі за бажанням;
- користувач активує/деактивує сповіщення за бажанням;
- користувач натискає кнопку «Підтвердити»;
- додаток зберігає нову ціль до списку цілей та починає відслідковувати прогрес її виконання на основі дати останньої випаленої цигарки;
- коли ціль досягнуто, у випадку, якщо користувач при створенні цілі активував сповіщення, додаток надішле йому оповіщення на телефон;
- кінець сценарію.

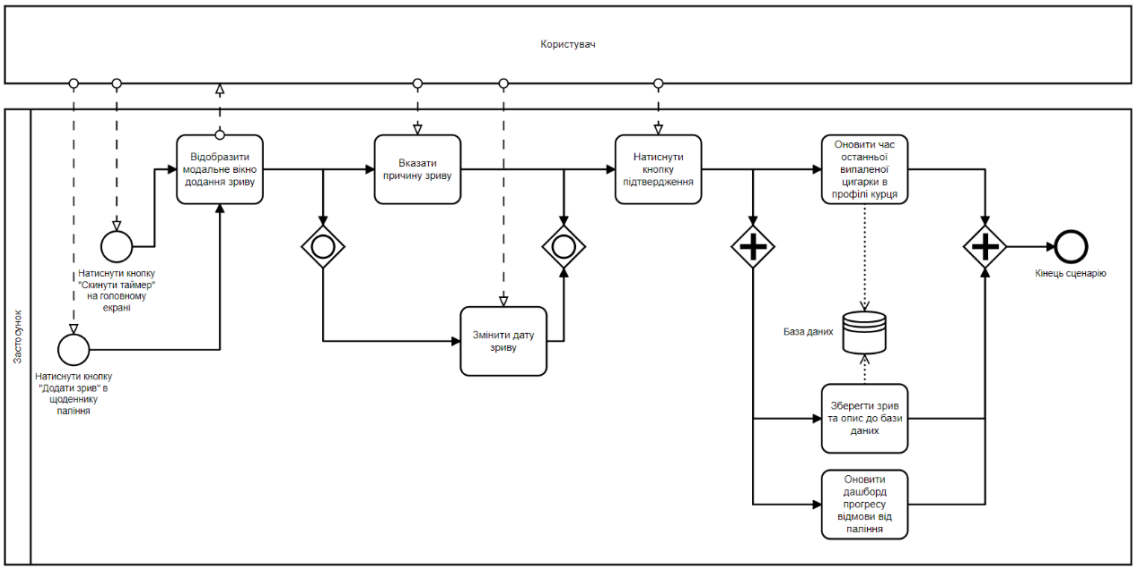

Рисунок 2.2 BPMN-діаграма для опису бізнес-процесу додання зриву

Послідовний опис бізнес-процесу додання нового зриву до щоденнику паління :

- користувач ініціалізує бізнес-процес, натискаючи на кнопку «Скинути таймер» на головному екрані або на кнопку «Додати зрив» на екрані щоденника паління;
- додаток відображає модальне вікно додання нового зриву з поточною датою;
- користувач записує причину зриву в текстове поле модального вікна за бажанням;
- користувач змінює поточну дату і час на обрану в календарі та циферблаті за бажанням;
- користувач натискає кнопку «Підтвердити»;
- додаток зберігає новий зрив з описом причини та датою до бази данних;
- додаток оновлює дату останньої випаленої цигарки;
- додаток оновлює показники відмови від паління на дашборді головного екрану;
- кінець сценарію;

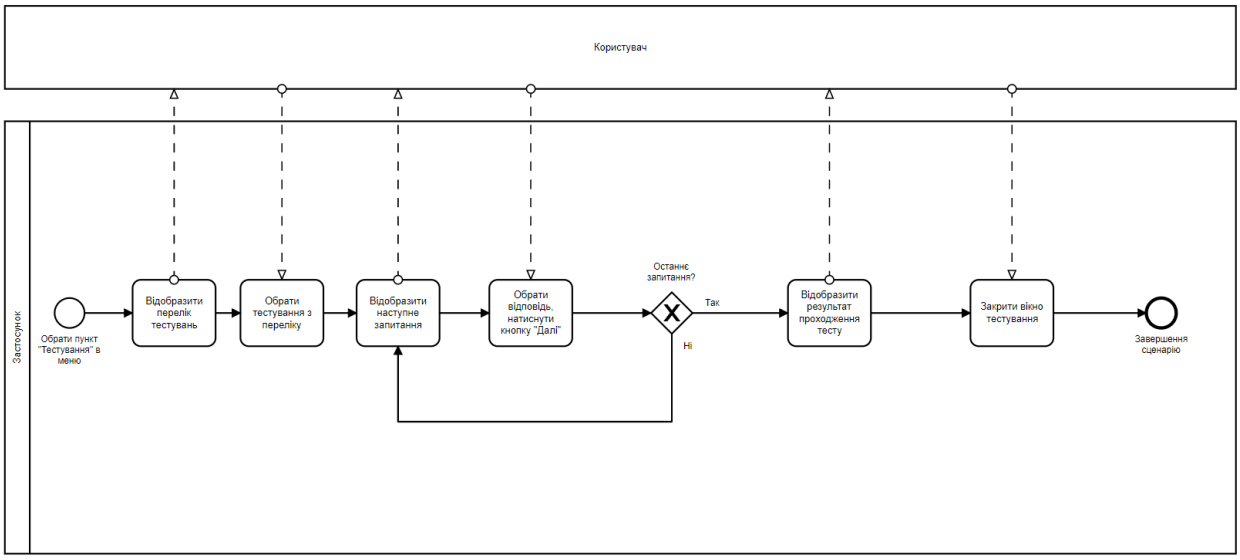

Рисунок 2.3 BPMN-діаграма для опису бізнес-процесу проходження тестування на вибір

Послідовний опис бізнес-процесу проходження тестування на вибір :

- користувач ініцалізує виконання сценарію, перейшовши на сторінку «Тестування»;
- додаток відображає список тестів для проходження;
- користувач обирає з переліку тест, який він бажає пройти;
- якщо запитання не закінчились, додаток відображає наступне запитання;
- користувач обирає варіант відповіді з переліку та натискає кнопку «Далі»;
- якщо запитання закінчились, додаток відображає результат тестування : кількість набраних балів та відповідний кількості балів опис результату з порадою для курця;
- користувач закриває вікно тестування, натискаючи на кнопку «Назад», повертаючись до вікна переліку тестувань для проходження;
- кінець сценарію;

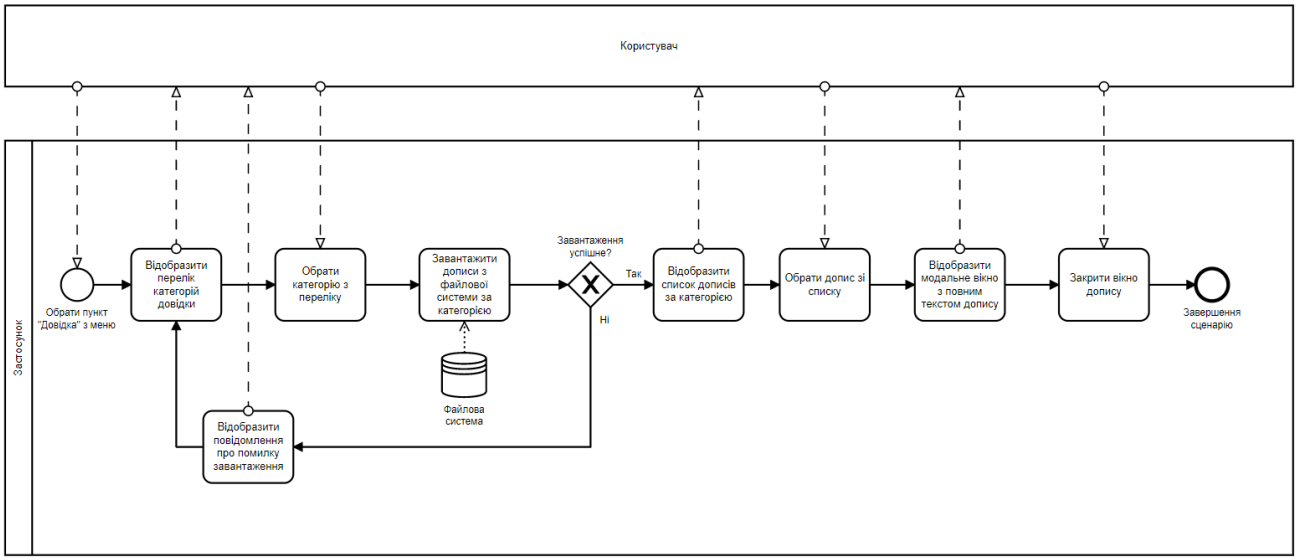

Рисунок 2.4 BPMN-діаграма для опису бізнес-процесу використання розділу довідки

Послідовний опис бізнес-використання розділу довідки :

- користувач ініціалізує сценарій, перейшовши на сторінку «Довідка»
- застосунок відображає список категорій довідки;
- користувач обирає категорію дописів для перегляду з переліку;
- застосунок завантажує дописи за обраною категорією з файлової системи.
- користувач обирає допис зі списку;
- застосунок відображає модальне вікно з заголовком та повним текстом допису, доступним для пролистування користувачем догори та донизу;
- користувач переглядає повний текст допису, закриває модальне вікно натисканням на кнопку «Зрозуміло»l
- користувач закриває перелік дописів, повернувшись на вікно категорій довідки;
- кінець сценарію.

2.2 Архітектура програмного забезпечення

Для того, щоб спроектувати програмний продукт, потрібно обрати архітектуру програмного забезпечення.

В якості архітектурного паттерну програмного забезпечення для мобільного застосунку індивідуальної боротьби з тютюнопалінням «SmokeBreak» було обрано паттерн MVP.

Шаблон MVP (англ. Model-View-Presenter, укр. Модель-Представлення-Представник) — архітектурний патерн, який походить від MVC, в основному використовується задля побудови призначеного для користувача інтерфейсу. Цей патерн описує, яким чином можна відділити логіку інтерфейсу від самого UI та окремо від даних. За рахунок такого поділу підвищується можливість повторного використання та гнучкість системи

Схему патерну MVP зображено на рисунку 2.5.

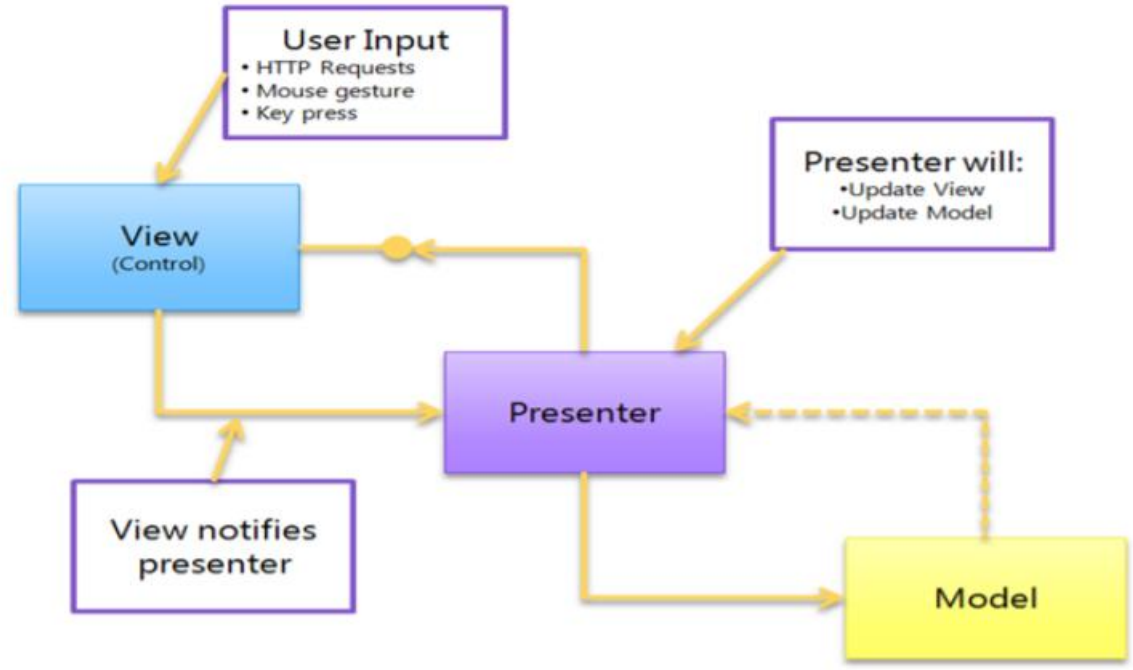

Рисунок 2.5 Схема архітектурного патерну MVP

Принципом MVP є розділення програмної реалізації системи на три головні компоненти: M — Model (Модель), V — View (Представлення), P — Presenter (Представник), таким чином, що редагування будь-якого компонента може відбуватися незалежно.

Модель (Model) — містить знання про предметну область, дані та правила до цих даних, але нічого не знає про контролери та представлення. У моделі відбувається бізнес-логіка роботи застосунку. Модель надає дані які запрошує користувач.

Представлення (View) — надає можливість по-різному відображати дані отримані будь-яким способом від моделі.Представлення — це кінцевий інтерфейс, з яким взаємодіє користувач. Користувач може передавати дані через представлення. Користувач може взаємодіє з елементами Представлення, але коли якась подія, що ініціалізує користувач, зачіпає логіку інтерфейсу, Представлення буде направляти його Представнику.

Представник(Presenter) — містить всю логіку, яка є в користувацькому інтерфейсі і відповідає за синхронізацію моделі данних і її представлення. Коли Представлення повідомляє Представнику, що користувач ініцалізував зміну даних, презентер приймає рішення про оновлення моделі і синхронізує всі зміни між Моделлю і Представленням. Особливістю архітектурного паттерну MVP є те, що Представник не взаємодіє з Представленням напряму, для їх взаємодії використовують інтерфейси, що дозволяє протестувати інтерфейс та логіку застосунку окремо.

Для зберігання даних в додатку було прийнято рішення використати фреймворк Realm.

Realm – no-sql мобільна база даних для Android (Java, Kotlin) та iOS (Objective-C, Swift). Realm була розроблена в якості альтернативи попереднім фреймворкам зберігання даних, які брали за свою основу SQLite. Завдяки тому, що Realm – no-sql база даних, вона має свої переваги в швидкості читання та маніпуляцій з даними. Realm є абсолютно безкоштовною та простою у використанні базою даних, для її встановлення достатньо підключити залежність в файлі Gradle.

Realm не потребує використання ORM фреймворків, вона використовує свій власний механізм персистетності. Для того, шоб клас сутності (entity) міг використовуватись Realm в якості моделі даних, він має наслідувати абстрактний клас RealmObject. Під час компіляції програмного продукту, Realm генерує класи RealmProxy, які використовує для трансфомації моделей даних в об’єкти даних бази даних Realm.

Структурна схема застосування зображена на рисунку 2.6.

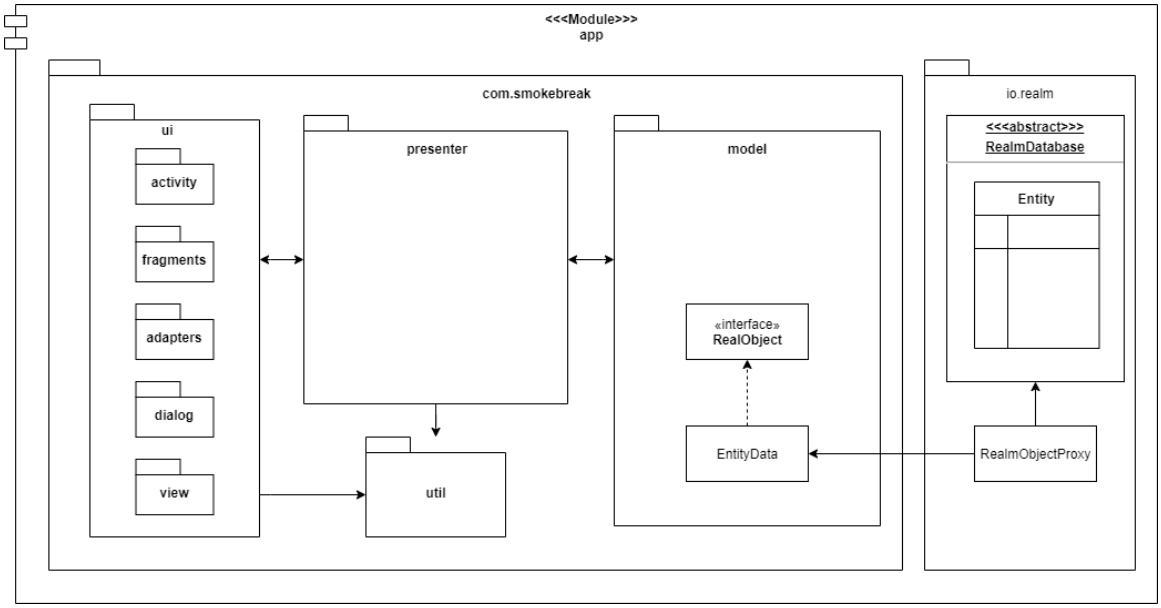

Рисунок 2.6. Структурна схема застосування

2.3 Конструювання програмного забезпечення.

2.3.1 Користувацький інтерфейс

Компоненти користувацького інтерфейсу відповідають за візуальне наповнення додатку.

Активність (Activity) – компонент програмного додатку, що містить користувацький інтефейс, з яким користувач може взаємодіяти. Активність являє собою екран користувача, ініцалізує елементи інтерфейсу та керує фрагментами даного екрану. Активність наслідує клас Activity та імплементує його методи, що дозволяє керувати життєвим циклом

активності. Зазвичай програмний додаток містить кілька актвностей, що пов’язкуються використанням об’єкту класу Intent, який дозволяє здійснювати перехід від однієї активності до іншої. Детальний опис класів активностей застосунку наведений в таблиці 2.1.

Таблиця 2.1 Опис класів активностей

Назва класу	Опис	Основні методи
StartActivity	Активність початкового екрану. Містить лого застосунку. Автоматично передає контроль іншій активності.	onCreate() – якщо профіль користувача є заповненим, передає контроль активності головного екрану, інакше – активності профілю курця. notEmpty() – перевіряє всі поля об’єкту UserData не дорівнюють 0 (чи є профіль заповненим користувачем).
MainActivity	Активність головного екрану. Містить меню застосунку, яке керується статичним класом FragmentPagerAdapter.	setViews() – відображає на екрані пункти меню : список цілей, щоденник паління, головний екран, розділ довідки, модуль тестування. onClick() – при натиску на пункт меню перенаправляє користувача на відповідний екран – компонент типу Fragment.
ProfileActivity	Активність екрану профілю курця. Містить дату останньогї випаленої цигарки, показники	initViews() – відображає на екрані компоненти користувацького інтерфейсу : дату останньої випаленої цигарки, заголовки показників, кнопку підтвердження, створює екземпляр класу ProfilePresenter.

Продовження таблиці 2.1

	профілю курця, кнопки для їх редагування та кнопку підтвердження. Імплементує інтерфейс IProfile для використання об’єктом ProfilePresenter.	showDate() – при натиску на дату останньої випаленої цигарки відображає компонент календарю для редагування дати. showTime() – після того як користувач обрав вибрав дату в календарі, відображає циферблат для вибору часу дня. setData() – встановлює значення показників з об’єкту UserData. onClick() – зберігає введені данні в ProfilePresenter та передає керування активності головного екрану MainActivity.
StatisticsActivity	Активність екрану статистики відмови від паління. Значення оновлюються за викликом екрану.	initViews() – відображає на екрані значення статистики : середній час відмови, середня кількість невипалених цигарок, середній об’єм коштів, середній об’єм зекономленого часу, кнопку «Назад».
HealthActivity	Активність екрану досягнень для здоров’я. Досягнення відображаються за допомогою компоненту RecyclerView та	initViews() – RecyclerView, відображає на екрані кнопку «Назад». loadJSONFromAsset () – завантажує об’єкти HealthData з файлової системи до бази даних Realm. showAdapter() – ініціалізує новий адаптер HealthAdapter об’єктами HealthData з бази даних Realm,

Продовження таблиці 2.1

	оновлюються за викликом екрану.	передає його RecyclerView для відображення.
InfoDetailActivity	Активність екрану розділу довідки. Дописи відображаються за допомогою RecyclerView, InfoDetailAdapter та InfoDetailDialog. Завантажує об’єкти дописів з файлів info_1.json, info_2.json, info_3.json. Змінює відображення в залежності від розділу довідки.	initViews() - створює компонент користувацього інтерфейсу RecyclerView, відображає на екрані кнопку «Назад». loadJSONFromAsset () - завантажує об’єкти InfoData з файлової системи до бази даних Realm. showAdapter() - ініціалізує новий адаптер InfoAdapter об’єктами HealthData з бази даних Realm, передає його RecyclerView для відображення. onInfoClick() – при кліку на елемент списку дописів ініціалізує створення діалогового вікна InfoDetailDialog для перегляду повного тексту допису.
TestActivity	Активність проходження тестування. Завантажує об’єкти тестів з файлів test_1.json, test_2.json, test_3.json. Відображає	loadJSONFromAsset() – завантажити об’єкти TestData з файлової системи до бази даних Realm. showQuestion() – відобразити наступне запитання. setData() – завантажити з бази даних Realm об’єкт TestData. onNext() – підрахувати очки, якщо запитання не закінчились, відобразити

Продовження таблиці 2.1

	запитання, підраховує очки.	наступне запитання, інакше відобразити результат.

Фрагмент (Fragment) – компонент програмного додатку, що за своєю природою є подібним до активності, оскільки теж відповідає за ініціалізацію користувацього інтерфейсу. Фрагмент являє собою частиною екрану або допоміжну діяльність. Важливою відмінністю між фрагментом та активнстю є те, що в той час як активність може існувати незалежно і містити декілька фрагментів, фрагмент завжди є компонентною частиною активності Детальний опис класів фрагментів наведено в таблиці 2.2..

Таблиця 2.2. Опис класів фрагментів

Назва класу	Опис	Основні методи
MainFragment	Фрагмент головного екрану. Імплементує інтерфейс IMain для використання презентером MainPresenter, який оновлює значення відліку, круга прогресу та показників дашборду щосекунди. Містить кнопки	initViews() – відображає компоненти користувацього інтерфейсу : таймер, час, дашборд з показниками відмови від паління : кількість невипалених цигарок, об’єми зекономлених коштів та часу, неспожитих смол, кнопки переходу до екранів статистики відмови та досягнень для здоров’я. onClick() – при натиску на таймер викликає діалог AddJournalDialog, при натиску на кнопки статистики та досягнень передає контроль відповідним активностям.

Продовження таблиці 2.2

	статистики та досягнень.	add() – доручає презентеру MainPresenter додати новий зрив до щоденнику паління.
GoalFragment	Фрагмент екрану списку цілей. Реалізує інтерфейс IGoal для використання презентером GoalPresenter. Значення показників прогресу виконання цілей оновлюються з викликом екрану.	onClick() – викликає діалог AddGoalDialog при натиску на кнопку «+». onGoalClick() – викликає діалог GoalInfoDialog при натиску на елемент цілі в списку цілей. onAdd() – при доданні нової цілі через діалог AddGoalDialog повідомити презентер GoalPresenter про оновлення списку. showList() – відображає список об’єктів GoalData за допомогою RecyclerView та GoalAdapter.
JournalFragment	Фрагмент екрану щоденнику паління. Реалізує інтерфейс IJournal для використання презентером JournalPresenter. Об’єкти даних JournalData відображаються в списку за	initView() – відображає компоненти користувацьког інтерфейсу : календар, кнопку «Додати зрив», ініціалізує RecyclerView, декорує календар, встановлює на нього прослуховувачі. prepareCalendar() – встановлює поточну дату в календарі, позначає дні в календарі в залежності від кількості зривів за окремий день.

	допомогою	

Продовження таблиці 2.2

	RecyclerView та JournalAdapter та оновлюються презентером.	add() – викликає діалог AddJournalDialog, передає йому поточний або обраний день. onClick() – доручає презентеру JournalPresenter додати новий зрив до щоденнику зривів. onJournalClick() – при натиску на зрив викликає діальог JournalInfoDialog.
InfoMainFragment	Фрагмент екрану списку розділів довідки. Містить розділи «Методи відмови від паління», «50 фактів про паління», «50 порад тим, хто бажає кинути палити».	initViews() – відображає кнопки для переходу до першого, другого та третього розділу довідки, прив’язує до них підписи та іконки. onClick() – при натиску на кнопку викликає InfoDetailActivity в залежності від обраного розділу.
TestFragment	Фрагмент екрану списку тестувань для проходження. Містить наступні тести «Тест Фагерстрема», «Чи готові в кинути	initViews() - відображає кнопки для переходу до першого, другого та третього розділу довідки, прив’язує до них підписи та іконки. onClick() - при натиску на кнопку викликає TestActivity в залежності від обраного розділу.

	палити?» та	

Продовження таблиці 2.2

	«Визначення особливостей паління».	
TestDetailFragment	Фрагмент екрану проходження тестування. Містить текст запитання та варіанти відповідей, життєвий цикл та бізнес-логіка керується активністю TestActivity.	newInstance() – створює об’єкт Bundle з значеннями номерів тесту та питання, а також загальної суми балів. initView() – відображає на екрані тест запитання та варіанти відповідей. setData() – завантажує об’єкти даних TestData з бази даних Realm. showResult() – відображає набрану суму балів, максимальну кількість балів та текст результату в залежності від суми балів.

Адаптер (Adapter) – компонент програмного додатку, що використовується для зв’язування даних з елементом управління. Призначення адаптеру – відображати представлення елементів для контейнера. В даному застосунку адаптери використовуються елементом RecyclerView для відображення сутностей списку. Детальний опис класів адаптерів наведено в таблиці 2.3

Таблиця 2.3. Опис класів адаптерів

Назва класу	Опис	Основні методи

GoalAdapter	Адаптер відображення цілі. Використовується	checkItem() – відображає іконку, надпис та цільове

Продовження таблиці 2.3

	RecyclerView фрагменту GoalFragment для відображення списку цілей.	значення в залежності від категорії цілі. calculateProgress() – обрахувати прогрес виконання цілі на основі поточного часу.
JournalAdapter	Адаптер відображення зриву. Використовується RecyclerView фрагменту JournalFragment для відображення списку зривів.	onCreateViewHolder() – створює новий об’єкт ViewHolder з заголовком та описом. onBindViewHolder() – заповнює заголовок та опис ViewHolder датою та причиною зриву з об’єкту даних JournalData.
HealthAdapter	Адаптер відображення досягнення для здоров’я. Використовується RecyclerView активності HealthActivity для відображення списку досягнень для здоров’я.	onCreateViewHolder() - створює новий об’єкт ViewHolder з іконкою, заголовком та описом. calculateProgress() – обрахувати прогрес виконання досягнення на основі поточного часу.
InfoAdapter	Адаптер відображення допису довідки.	onCreateViewHolder() - створює новий об’єкт

	Використовується RecyclerView активності	ViewHolder, заголовком та описом.

Продовження таблиці 2.3

	InfoDetailActivity для відображення списку дописів за обраною категорією.	onBindViewHolder() – заповнює заголовок та опис ViewHolder заголовком та описом з об’єкту InfoData. InfoViewHolder.checkNext() – змінює відображення допису в списку в залежності від обраної категорії довідки.

Діалог, діалогове вікно (Dialog) – компонент програмного додатку, що викликається активністю і використовується для більш гнучкого представлення користувацького інтерфейсу та взаємодії з користувачем. Є окремим випадком фрагменту. Детальний опис класів діалогових вікон наведено в таблиці 2.4.

Таблиця 2.4. Опис класів діалогів

Назва класу	Опис	Основні методи
AddGoalDialog	Діалогове вікно додання нової цілі. Містить такі категорії цілі : «Невипалені цигарки», «Зекономлені	initView() – відобразити компоненти користувацього інтерфейсу діалогу : категорії цілі, кнопки редагування цільового значення, показники вимірювання, текстове поле, кнопка «Додати». onClick() – обробляє подію натиску

	кошти» та «Зекономлений час», показники вимірювання : 100шт/10шт/1шт, 100грн/10грн/1грн, 1м/1д/1год.	на візуальний компонент : редагує цільове значення, підтверджує додання цілі. prepareChoosers() – підписує показники вимірювання в залежності від обраної категорії. resetValue() – скидає цільове значення при зміні категорії.
GoalInfoDialog	Діалого вікно відображення цілі. Містить іконку категорії, постановку цілі та повний текстовий опис, а також кнопку «Видалити».	initView() – відображає компоненти візуального інтерфейсу діалогу : іконку, цільове значення, опис цілі, кнопку «Видалити». onClick() – при натиску на кнопку «Видалити» видаляє ціль зі списку. setData() – заповнює візуальні компоненти значеннями з об’єкту GoalData в залежності від категорії.
AddJournalDialog	Діалого вікно додавання зриву. Містить дату зриву, текстове поле для введення причини зриву, і кнопку «Додати».	initView() - – відображає компоненти користувацього інтерфейсу діалогу : поле дати та часу, текстове поле опису причини зриву, кнопку «Додати». onClick() – при натиску на кнопку «Додати» додає новий зрив до щоденнику паління. showDate() – відображає віконце з календар для вибору дати зриву. showTime() – відображає віконце з циферблатом для вибору часу

		зриву. setInitialDateTime() – встановлює поточний час.

Продовження таблиці 2.4

JournalInfoDialog	Діалогове вікно перегляду зриву. Містить дату зриву та повний текст опису причини зриву.	initView() - відображає компоненти користувацього інтерфейсу діалогу : дату зриву, опис причини зриву, кнопку «Назад». onClick() – закриває вікно діалогу. setData() – заповює візуальні компоненти значеннями з об’єкту JournalData.

Представлення, віджет (View) – компонент програмного додатку, що реалізує логіку використання певного графічного елементу користувацького інтерфейсу. Детальний опис класів представлень наведено в таблиці 2.5.

Таблиця 2.5 Опис класів представлень

Назва класу	Опис	Основні методи
ProfileChooseView	Візуальний компонент збільшення/зменшення цільового значення.	initViews() – відобразити візуальні компоненти : кнопку «-», кнопку «+», заголовок, значення, опис. setData() – ініціалізувати значення візуальних компонентів. onClick() – при натиску на кнонки збільшує/зменшує цільове значення.
ProfileDataView	Візуальний	initViews() - відобразити візуальні

	компонент відображення значень показників відмови від паління	компоненти : іконку, підпис, значення. setData() – ініцуалізувати візуальні компоненти.

Продовження таблиці 2.5

AnswersView	Візуальний компонент вибору варіанту відповіді.	setData() – ініціалізує список варіантів відповіді з радіо-кнопками.

2.3.2 Бізнес-логіка

Презентер, представник (Presenter) – це компонент програмного продукту, що пов’язує користувацький інтерфейс з моделлю даних. Презентер містить логіку, винесену з активності до окремого класу. Презентер обробляє зміни, внесені користувачем і згідно них оновлює представлення. Детальний опис класів презентерів описано в таблиці 2.6.

Таблиця 2.6. Опис класів презентерів

Назва класу	Опис	Основні методи
MainPresenter	Презентер головного екрану. Використовує інтерфейс IMain для керування фрагментом MainFragment.	prepareDate() – обраховує час відмови від паління. startTimer() – щосекунди оновлює прогресс таймеру та показники відмови від паління. addJournal() – додає новий зрив до бази данних Realm, оновлює дату останньої випаленої цигарки.
ProfilePresenter	Презентер екрану профілю курця.	checkProfileData() – якщо користувач заповнював профіль,

	Використовує інтерфейс IProfile для керування	завантажити данні з бази данних Realm, інакше встановити значення за замовчуванням.

Продовження таблицы 2.6

	активністю ProfileActivity.	saveData() – зберегти значення профілю курця до бази данних Realm.
GoalPresenter	Презентер екрану списку цілей. Використовує інтерфейс IGoal для керування фрагментом GoalFragment.	checkGoal() – якщо в базі даних Realm є цілі, відображає їх, інакше повідомлення про те, що цілі відсутні. addGoal() – додає в базу даних Realm нову ціль. deleteGoal() – видаляє з бази даних Realm обрану ціль. calculateFinish() – обраховує дату закінчення виконання цілі на основі даних профілю курця.
JournalPresenter	Презентер щоденнику паління. Використовує інтерфейс IJournal для керування фрагментом JournalFragment.	getJournalByDate() – завантажує з бази даних Realm зриви за обрану дату. notifyAdapter() – оновляє JournalAdapter, коли обрано інший день або місяць в календарі. addJournal() – додати в базу даних Realm новий зрив. getMonthJournalData() – завантажує з бази даних Realm зриви за

		обраний місяць.

Інтерфейс – це окремий випадок абстрактного класу програми, що містить декларацію методів, але не їх реалізацію. Презентери пов'язані з компонентами користувацького інтерфейсу за допомогою інтерфейсу. Для цього ці компоненти повинні імплементувати інтерфейс та його методи. Такий підхід дозволяє відділити бізнес-логіку та представлення даних додатковим шаром абстракції, що полегшує тестування кожного елементу окремо. Детальний опис класів інтерфейсів наведено в таблиці 3.7.

Таблиця 3.7. Опис класів інтерфейсів

Назва інтерфейсу	Опис	Абстрактні методи
IMain	Інтерфейс, що використовується презентером MainPresenter для керування фрагментом MainFragment.	setProgress() – встановлює значення шкали прогресу. setDateNoSmoke() – встановлює часу відмови від паління. setCig() – встановлює значення кількості невипалених цигарок. setMoney() – встановлює значення об’єм зекономлених коштів. setTime() – встановлює значення об’єму зекономленого часу. setSmolle() – встановлює значення об’єму неспожитих смол.
IProfile	Інтерфейс, що	setData() – встановлює значення

	використовується презентером ProfilePresenter для керування фрагментом ProfileActivity.	профілю курця : дату останньої випаленої цигарки, кількість цигарок в день, кількість цигарок в пачці, вартість пачки, об’єм смол в цигарці.

Продовження таблицы 2.7

IGoal	Інтерфейс, що використовується презентером GoalPresenter для керування фрагментом GoalFragment.	showEmpty() – відображає повідомлення про те, що список цілей порожній. showList() – відображає список цілей.
IJournal	Інтерфейс, що використовується презентером JournalPresenter для керування фрагментом JournalFragment.	showEmpty() – відображає повідомлення про те, що за обраний день зривів не було. showList() – відображає список зривів за обраний день.

2.3.3 Допоміжні класи

Утилітарні класи – це класи, що реалізують певну універсальну логіку, яка часто використовується в різноманітних класах, тому її зручніше винести в окрему сутність.

Таблиця 2.8 Таблиця утилітарних класів

Назва класу	Опис	Основні методи
UserUtils	Утилітарний клас, що використовується для обрахування	calculateProgress() – обчислює час відмови від паління getCigSaved () – обчислює кількість

	показників прогресу відмови від паління на основі данних профілю курця.	невипалених цигарок getMoneySaved() – обчислює об'єм зекономлених коштів getTimeSaved() – обчислює об'єм зекономленого час

Продовження таблиці 2.8

		getSmollSaved() – обчислює об'єм неспожитих смол
CalendarUtils	Утилітарний клас, що використовується для маніпуляцій з часом.	getFirsttTimeStampOfCurrentMonth() – обчислює першу відмітку часу поточного місяцю getLastTimeStampOfCurrentMonth() – обчислює останню відмітку часу поточного місяцю getFirstTimeStampOfCurrentDay() - обчислює першу відмітку часу поточного дня getLastTimeStampOfCurrentDay() - обчислює останню відмітку часу поточного місяцю getStrDate() – повертає дату в формі рядку

2.3.4 Модель даних

В якості моделі даних застосунок використовує класи-сутності, що наслідують абстрактний клас RealmObject. Realm трансформує класи даних в сутності бази даних і навпаки. Також Realm може завантажувати в базу даних не тільки моделі, створені під час роботи застосунку, але й з файлової

системи, наприклад, файлу JSON. Детальний опис моделей даних застосунку наведено в таблиці 2.9.

Таблиця 2.9 Опис структури моделей даних

Модель даних	Структура моделі	Опис

Продовження таблиці 2.9

UserData	{
	id : int,
	lastSmokeDate : long,
	cigarettesInDay : int,
	cigarettesInBox : int,
	pricePerBox : int,
	smolleSize : float
	}
		Структура даних профілю курця. Містить в собі id, дату останньої випаленої цигарки, кількість цигарок в день, кількість цигарок в пачці, ціну пачки, об’єм смол в пачці.

HealthData	{
	estimate : long,
	description : String
	}
		Структуру даних досягнення для здоров’я. Містить в собі оцінку часу виконання та опис досягнення.

GoalData	{
	startDate : long,
	finishDate : long,
	description : String,
	type : int,
	count : int,
	time : long
	}
		Структура даних цілі. Містить в собі дату початку, дату закінчення, текстовий опис, тип категорії, цільове значення та час виконання.

JournalData	{ date : long,	Структура даних зриву. Містить в собі відмітку часу та

	description : String }	текстовий опис.
InfoData	{ id : int, name : String, infoDetailList :	Структуру даних розділу довідки. Містить id, назву, список дописів.

Продовження таблиці 2.9

	RealmList<InfoDetailData> }	

InfoDetailData	{
	title : String,
	description : String
	}
		Структура даних допису розділу довідки. Містить заголовок та текстовий опис.

TestData	{
	id : int,
	name : String,
	questionList : RealmList<QuestionData>,
	resultList : RealmList<ResultData>
	}
		Структура даних тестування для проходження. Містить id, назву, список запитань та список результатів.

QuestionData	{
	id : int,
	question : String,
	answerList : RealmList<AnswerData>
	}
		Структура даних запитання тесту. Містить id, текст запитання, список варіантів відповіді.

AnswerData	{	Структура даних варіанту

	answer : String, points : int }	відповіді тесту. Містить текст варіанту відповіді та кількість очків, що нараховуються.
ResultData	{ result : String, min : int,	Структура даних результату проходження тестування. Містить текст результату,

Продовження таблиці 2.9

	max : int }	мінімальний та максимальний поріг балів.

На рисунку 2.7 зображена схема моделей даних.

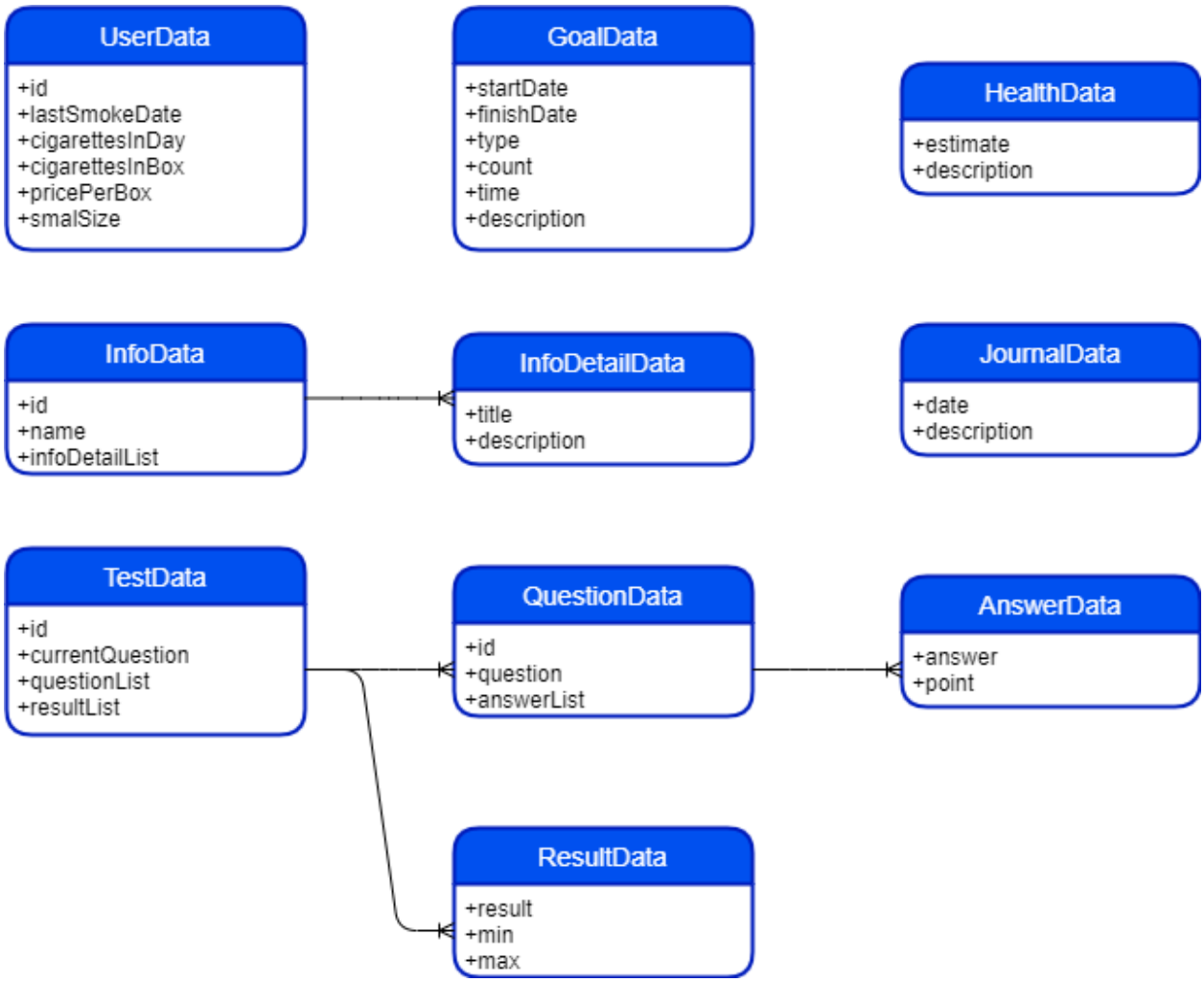

Рисунок 2.7 Схема моделей даних.

2.4 Аналіз безпеки данних

Для захисту даних мобільного застосунку індивідуальної боротьби з тютюнопалінням «SmokeBreak» потрібно провести процедуру обфускації.

Обфускація – це процес заплутування програмного коду, в результаті якого програмний продукт зберігає свою функціональність, але аналізу його структури компонентів та даних стає більш заплутаним. Обфускація необхідна для ускладнення процесу декомпіляції вихідного коду та/або приховування даних користувачів від третіх осіб.

Для обфускації було прийнято рішення використати фреймворк ProGuard з його інтеграцію з базою даних Realm.

2.5 Висновки до розділу

В даному розділі було проведено моделювання та конструювання мобільного додатку для індивідуальної боротьби з тютюнопалінням «SmokeBreak».

Були проаналізовані бізнес-процеси застосунку, наведений опис з використанням діаграм BPMN.

Була змодельована архітектура застосунку – розробка буде проводитись за архітектурним патерном MVP. Також для збереження моделей даних застосунку було обрано фреймворк Realm.

Були описані класи та компоненти програмного застосунку та їх методи.

Була наведена структура моделей даних бази даних Realm.

Був описаний процес захисту даних програмного застосунку – для цього було прийнято рішення провести процедуру обфускації.

За результатами моделювання та конструювання можна зробити висновок, що застосунок готовий для розробки та подальшого тестування.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПЗ

3.1 Аналіз якості ПЗ

Під час розробки додатку та протягом його експлуатації можуть виникати різноманітні помилки, як критичні, що повністю унеможливлюють роботу програмного забезпечення, так і незначні, які, тим не менш, погіршують враження від застосунку. Більшість багів та дефектів програмного забезпечення стають очевидними у процесі розробки, однак є і такі, що з'являються, коли додаток вже експлуатується користувачами, вони можуть призвести до втрати коштів та клієнтів замовником, а їх виправлення коштує дуже дорого. Баги сповільнюють команду та вимагають додаткових ресурсів для їх вирішення, саме тому контролювати якість програмного забезпечення потрібно з моменту початку розробки, щоб забезпечити належний рівень якості програмного забезпечення.

Якість програмного забезпечення – це ступінь відповідності ПЗ заданим вимогам або очікуванням користувача. Тестування програмного забезпечення - це спосіб гарантувати якість продукту, перевірити відповідність заявлених до продукту вимог і реалізованої функціональності, здійснений шляхом спостереження за його роботою в спеціально заданих ситуаціях на обмеженому наборі тестів. Тестування є невід’ємним етапом у процесі розробки будь-якого програмного забезпечення, як спосіб підтвердження працездатності продукту. Метою коректного тестування є зниження ризику шляхом активного повторюваного пошуку та вирішення проблем, що могли б найбільш негативно вплинути на користувача програмного забезпечення, що тестується.

В ході тестування задля визначення ступеня якості системи, використовують так звані атрибути якості (характеристики), до яких відносять:

- надійність (reliability);
- готовність (availability);

– безпека (safety);
– зручність роботи (usability);
– ремонтопридатність (maintainability);
– конфіденційність (confidentiality);
– цілісність (integrity).

В сфері інформаційних техтнологій існує багато методів тестування програмного забезпечення та оцінки різноманітних атрибутів якості, але не існує однозначно коректного способу провести тестування програмного забезпечення, тому тестування є по суті дослідницький процес, який вимагає від особи, що проводить тестування, з урахуванням специфіки програмного продукту, що розробляється, та його предметної області.

Тестування мобільного застосунку дещо відрізняється від звичайної програми для ПК, адже в процесі тестування перед тестувальнику необхідно враховувати мобільність пристрою, можливі оновлення платформи операційної системи, перемикання орієнтації екрану, масштабованість елементів користувацького інтерфейсу в залежності від розмірних характеристик мобільного телефону Адже потрібно враховувати мобільність пристрою, можливі оновлення операційної системи, змінення орієнтації екрану, переключення програми в фоновий режим, обмежена кількість пам'яті як основної так і оперативної і так далі.

Аби стратегія тестування була максимально ефективною, варто заздалегідь написати тест-план. Стандарт IEEE 829 дозволяє окреслити стратегію тестування, а також зважити ризики, що можуть виникнути в процесі розробки. Завдання створення тестового плану полягає в наступному:

– визначення об’єкту тестування;
– опис підхідів до тестування;
– визначення функцій, що будуть протестовані;
– визначення тестів, що будуть виконуватися;
– визначення критеріїв проходження тестів.

3.2 Опис процесів тестування

3.2.1 Ідентифікатор тест-плану

SmokeBreak Testing TP_1.0

3.2.2 Вступ до тест-плану

Цей план стосується мобільного застосунку для індивідуальної відмови від тютюнопаління «SmokeBreak», написаному на мові програмування Java для мобільної платформи Android.

Окремо у проекті мають бути перевірені:

- модуль профілю курця;
- модуль списку цілей;
- модуль щоденника паління;
- модуль прогресу відмови від паління;
- модуль довідки;
- модуль тестування.

3.2.3 Об’єкт тестування

Об’єктом тестування є застосунок на мобільний телефон з платформою Andoid.

3.2.4 Компоненти, що тестуються

В ході тестування мають бути протестовані інтерфейсні та функціональні характеристики програмного забезпечення.

Компоненти користувацького інтерфейсу, що мають бути протестовані, включають :

- коректна реакція кнопок на натискання;
- виведення потрібних екранів та вікон;
- коректне відображення данних у текстових полях;
- коректна робота клавіатури.

Функціональні компоненти, що мають бути протестовані в модулі профілю курця, включають :

- відображення профілю курця при першому запуску застосунку;

– заповнення профілю курця значеннями за замовчуванням;
– коректна реакція значень на натиск кнопок;
– збереження данних профілю курця.

Функціональні компоненти, що мають бути протестовані в модулі списку цілей, включають :

– створення нової цілі та її збереження;
– коректне відслідковування прогресу виконання цілей;
– коректна реакція цілей на оновлення дати останньої цигарки;
– видалення цілі зі списку цілей.

Функціональні компоненти, що мають бути протестовані в модулі щоденника паління, включають :

– коректне додання нового зриву з описом;
– перегляд історії щоденника паління за днями;
– коректне відображення днів в календарі щоденника паління.

Функціональні компоненти, що мають бути протестовані в модулі прогресу відмови від паління :

– коректне відображення таймеру відмови від паління;
– коректне скидання таймеру відмови від паління;
– коректне відображення показників відмови від паління;
– коректне відображення досягнень відмови від паління;
– коректне відображення статистики відмови від паління;
– коректна реакція показників на оновлення лічильника.

Функціональні компоненти, що мають бути протестовані в модулі довідки:

– можливість вибору категорії дописів;
– завантаження дописів з бази данних за обраною категорією;
– коректне відображення дописів.

Функціональні компоненти, що мають бути протестовані в модулі тестування від паління :

– можливість вибрати тестування для проходження зі списку тестів;
– коректна обробка вибору варіанту відповіді;
– коректне виведення результатів тестування.

3.2.5 Компоненти, що не підлягають тестуванню

Не підлягає тестуванню наступна функціональність :

– стійкість до навантаження;
– безпека даних;
– швидкодія інтерфейсу;
– швидкодія роботи з базою даних.

3.2.6 Підхід до тестування

Тестування будуть проводитися згідно з описаними завчасно тест кейсами. Кожен тест кейс буде помічено як такий, що пройшов тестування або такий, що провалився. Для тестів, що провалилися, буде занотовано шляхи відтворення виключної ситуації, проаналізовано серйозність і пріоритетість та запропоновано шляхи роз’язання.

3.2.7 Критерії проходження тестів

Успішно пройденим тестування вважається, якщо в ході тестування не було знайдено жодних помилок на усіх рівнях тестування, та виконані усі вимоги, зазначені у технічному завданні. Увесь основний функціонал системи повинен функціонувати, як очікувалося, та окреслено в окремих варіантах тестування.

3.2.8 Критерії призупинення та продовження тестування

Процес тестування призупиняється, якщо в ході тестування був знайдений дефект типу Blocker, що негативно впливає на працездатність усієї системи. Тестування має бути продовжене, коли розробник виправить дефект та наддасть нову версію програмного забезпечення.

3.2.9 Задачі для проведення тестування

Тестування включає в себе наступні задачі :

– затвердження плану тестування;

– опис функціональних вимог та характеристик додатку;
– підготовка програмного забезпечення до тестування;
– успішне проходження тестів.

3.2.10 Технічні потреби

Застосунок повинен бути протестований на пристрої на платформі Android з версією вище 5.0 з технічними характеристиками, зазначеними в технічному завданні.

3.2.11 Інструменти

Тестування проводиться з використанням вбудованого емулятору Andoid Virtual Device середовища розробки Andoid Studio.

3.2.12 Обов'язки

Оскільки додаток є простим програмним забезпеченням, тестуванням та розробкою займається одна людина. Вона несе відповідальність за всі ризики та слідування плану тестування. Основні завдання тестувальника – створити так багато тест кейсів, як можливо, та описати дефекти.

3.2.13 Необхідні компетенції та тренінги

Ознайомлення з функціональними вимогами та характеристиками програмного додатку, особливостями роботи платформи Android та знання підходів то тестування програмного забезпечення.

3.2.14 Розклад

Написання тест-плану має відбутись до початку роботи над проектом, але після визначення вимог до продукту. Тестування має розпочатись паралельно розробці. Фінальна ітерація тестування починається, коли усі основні функції додатку реалізовані та не пізніше, ніж за 3 тижні до дати введення продукту в експлуатацію.

3.2.15 Ризики

В таблиці 3.1 описані основні ризики, та засови їх протидії.

Таблиця 3.1 Ризики при тестуванні

Ризик	Засіб протидії
Зміна вимог	Забезпечення гнучкості та пристусованості ПЗ, можливість масштабування програмних компонентів
Непристосованість програмних компонентів до тестування	Розробка програмного забезпечення з урахуванням необхідності тестування
Помилка роботи емулятора мобільного пристрою AVD	Стирання данних віртуального пристрою, створення нового, відтворення прецеденту для аналізу причини виникнення помилки.
Пошкодження програмних данних	Використання системи контролю версій для відновлення.

3.3 Опис контрольного прикладу

В процесі тестування була перевірена уся функціональність підсистеми. Були відтворені всі варіанти використання, результати представлені у відповідних таблицях:

- заповнення профілю курця (таблиця 3.2);
- редагування профілю курця (таблиця 3.3);
- додання нової цілі (таблиця 3.4);
- перегляд списку цілей та прогресу виконання цілей (таблиця 3.5);
- перегляд прогресу відмови від паління (таблиця 3.6);
- видалення цілі (таблиця 3.7);
- додання зриву в щоденник паління (таблиця 3.8);
- перегляд щоденнику паління (таблиця 3.9);
- вибір тесту для проходження (таблиця 3.10);
- перегляд дописів за категорією (таблиця 3.11);

Таблиця 3.2 Опис тестового прецеденту ТС01

Ідентифікатор тестового прецеденту	TC01
Мета тестування	Перевірка можливості заповнення профілю курця
Передумови	1) Встановити програмний додаток Дочекатися кінця анімації початкового екрану
Вхідні дані	Значення показників профілю курця
Схема проведення тесту	1) Натисканням на кнопки «-» та «+» встановити значення відповідних показників : середня кількість цигарок в день, кількість цигарок в пачці, вартість пачки, кількість смол в цигарці. 2) Натиснути кнопку «Готово».

Продовження таблиці 3.2

	3) Перезапустити додаток. 4) Натиснути на кнопку профілю курця.
Очікуваний результат	Значення, що відображаються в профілі курця, відповідають значенням, що були заповнені Користувачем раніше.
Стан програмного продукту після проведення випробувань	Значення, що відображаються в профілі курця, відповідають значенням, що були заповнені Користувачем раніше.

Таблиця 3.3 Опис тестового прецеденту TC02

Ідентифікатор тестового прецеденту	TC02
Мета тестування	Перевірка можливості редагування профілю курця

Передумови	1) Відтворити тестовий прецедент ТС01
Вхідні дані	Значення показників профілю курця
Схема проведення тесту	1) Натиснути на кнопку профілю курця на головному екрані. 2) Натисканням на кнопки «-» та «+» редагувати значення відповідних показників : середня кількість цигарок в день, кількість цигарок в пачці, вартість пачки, кількість смол в цигарці. 3) Натиснути кнопку «Готово». 4) Натиснути на кнопку профілю курця на головному екрані.

Продовження таблиці 3.3

Очікуваний результат	Значення, що відображаються в профілі курця, відповідають редагованим значенням.
Стан програмного продукту після проведення випробувань	Значення, що відображаються в профілі курця, відповідають редагованим значенням.

Таблиця 3.4 Опис тестового прецеденту ТС03

Ідентифікатор тестового прецеденту	ТС03
Мета тестування	Перевірка можливості додання нової цілі
Передумови	1) Запустити додаток 5) Перейти на вкладку списку цілей

Вхідні дані	Цільове значення відповідної категорії, короткий опис цілі
Схема проведення тесту	1) Натиснути на кнопку додання нової цілі. 6) Обрати категорію для нової цілі. 7) Натисканням на кнопки «-» та «+» встановити цільове значення. 8) В текстовому полі ввести короткий опис цілі. 9) Натиснути кнопку «Додати».
Очікуваний результат	Нова ціль з обраними раніше категорією, цільовим значенням, коротким описом та прогресом в 0% створено і додано до списку цілей.

Продовження таблиці 3.4

Стан програмного продукту після проведення випробувань	Нова ціль з обраними раніше категорією, цільовим значенням, коротким описом та прогресом в 0% створено і додано до списку цілей.

Таблиця 3.5 Опис тестового прецеденту ТС04

Ідентифікатор тестового прецеденту	ТС04
Мета тестування	Перевірка можливості відслідковування списку цілей
Передумови	1) Запустити додаток Встановити значення в профілю курця : середня кількість цигарок в день – 20 шт.

Вхідні дані	Цільове значення за категорією невипалених цигарок
Схема проведення тесту	1) Створити нову ціль : «Не скурити 1 шт. цигарок», за методологією тестового прецеденту ТС03. 10) Повернутися на головний екран. 11) Зачекати 216 секунд. 12) Перейти на сторінку списку цілей.
Очікуваний результат	Створена ціль присутня в списку цілей і має показник прогресу – 5%.
Стан програмного продукту після проведення випробувань	Створена ціль присутня в списку цілей і має показник прогресу – 5%.

Таблиця 3.6 Опис тестового прецеденту ТС05

Ідентифікатор тестового прецеденту	ТС05
Мета тестування	Перевірка можливості видалення цілі зі списку цілей
Передумови	1) Відтворити тестовий прецедент ТС03.
Вхідні дані	Цільове значення відповідної категорії, короткий опис цілі
Схема проведення тесту	1) Натиснути на створену ціль в списку цілей. 13) У відкритому модальному вікні натиснути кнопку «Видалити».
Очікуваний результат	Ціль видалена зі списку цілей.
Стан програмного продукту після	Ціль видалена зі списку цілей.

проведення випробувань	

Таблиця 3.7 Опис тестового прецеденту ТС06

Ідентифікатор тестового прецеденту	ТС06
Мета тестування	Перевірка можливості перегляду прогресу відмови від паління
Передумови	1) Відкрити додаток.
Вхідні дані	Цільове значення відповідної категорії, короткий опис цілі
Схема проведення тесту	1) Перейти у вікно профілю курця. 14) Встановити поточний час як дату останньої випаленої цигарки та заповнити

Продовження таблиці 3.7

	наступні значення : кількість цигарок – 20 штук, кількість цигарок в пачці – 20 штук, вартість однієї пачки – 70 гривень,
Очікуваний результат	Створена ціль видалена зі списку цілей.
Стан програмного продукту після проведення випробувань	Створена ціль видалена зі списку цілей.

Таблиця 3.8 Опис тестового прецеденту ТС07

Ідентифікатор тестового прецеденту	ТС07

Мета тестування	Перевірка можливості додання зриву в щоденник паління
Передумови	1) Відкрити додаток.
Вхідні дані	Опис причини зриву
2) Схема проведення тесту	1) Натиснути на кнопку «Скинути таймер». 2) В модальному вікні вказати причину зриву. 3) Натиснути кнопку «Підтвердити».
Очікуваний результат	На головному екрані обнулився таймер, дата змінилась на поточну, показники відмови від паління оновились.
Стан програмного продукту після проведення випробувань	На головному екрані обнулився таймер, дата змінилась на поточну, показники відмови від паління оновились.

Таблиця 3.9 Опис тестового прецеденту ТС08

Ідентифікатор тестового прецеденту	ТС08
Мета тестування	Перевірка можливості перегляду щоденника паління.
Передумови	1) Відтворити тестовий прецедент ТС07.
Вхідні дані	Опис причини зриву
Схема проведення тесту	1) Перейти до сторінки щоденнику паління. 2) Обрати поточну дату в календарі паління.
Очікуваний результат	В щоденнику паління за поточну дату додано новий зрив з вказаною причиною і часом скидання таймеру. В календарі поточна дата

	позначається жовтим кольором.
Стан програмного продукту після проведення випробувань	В щоденнику паління за поточну дату додано новий зрив з вказаною причиною і часом скидання таймеру. В календарі поточна дата позначається жовтим кольором.

Таблиця 3.10 Опис тестового прецеденту ТС09

Ідентифікатор тестового прецеденту	ТС09
Мета тестування	Перевірка можливості проходження тестування на вибір.
Передумови	1) Відкрити додаток. 15) Перейти до вікна тестування.
Вхідні дані	Варіанти тестування
Схема проведення тесту	1) Обрати з переліку тестування для проходження.

Продовження таблиці 3.10

	2) Обирати варіант відповіді зі списку і натискати кнопку «Далі», доки відображаються запитання.
Очікуваний результат	Відображається результат тестування : кількість набраних балів та відповідний йому текст.
Стан програмного продукту після проведення випробувань	Відображається результат тестування : кількість набраних балів та відповідний йому текст.

Таблиця 3.11 Опис тестового прецеденту ТС10

Ідентифікатор тестового прецеденту	TC10
Мета тестування	Перевірка можливості перегляду дописів за категорією.
Передумови	16) Відкрити додаток. 17) Перейти до вікна довідки.
Вхідні дані	-
Схема проведення тесту	18) Обрати з переліку категорію довідки для перегляду. 19) Обрати зі списку дописів допис на вибір.
Очікуваний результат	Відображається вікно перегляду допису з можливістю пролистання.
Стан програмного продукту після проведення випробувань	Відображається вікно перегляду допису з можливістю пролистання.

3.4 Висновки до розділу

В даному розділі було проведено аналіз якості програмного забезпечення. Було описано, чим являється якість ПЗ та які вона має характеристики. Було описано тест-план, за яким має тестуватися програмне забезпечення. Було визначено ряд тестів, успішне проходження яких повинно гарантувати якість програмного забезпечення. За результатом проведення тестування можна зробити висновок, що додаток успішно проходить вимоги до якості та відповідає поставленим перед ним функціональним вимогам.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розгортання програмного забезпечення можна здійснити кількома способами : за допомогою емулятора мобільного пристрою, на реальному мобільному пристрої та опублікувавши застосування в сервісі-магазині програмних додатків Google Play Market.

4.1.1 Розгортання додатку в емуляторі мобільного пристрою

Для запуску застосунку в емуляторі мобільного пристрою Android потрібно використати засіб середовища розробки Android Studio – AVD Manager. В даному інструменті необхідно створити новий віртуальний пристрій, вказати його конфігурацію : характеристики мобільного пристрою, такі як діагональ дисплею, обсяг оперативної пам’яті, версія Android API, запустити збірку програмного додатку, що автоматично встановить його на віртуальний девайс. Далі можна запустити емулятор та почати роботу з програмним продуктом. Емулятор є повноцінною симуляцією реального мобільного пристрою – так, користувач може гнучко налаштувати всі необхідні йому конфігурації мови, місцевого часу, взаємодіяти з інструментами камери, місцевого сховища, геолокації, мікрофону, звінків, різноманітних кнопок та сенсорів, протестувати роботу додатку при повороті екрану. Також емулятор дозволяє дослідити роботу системи поза процесом застосунку – наприклад щоб протестувати його роботу в фоновому режимі, механіку сповіщень або взаємодію з іншими застосунками віртуального пристрою.

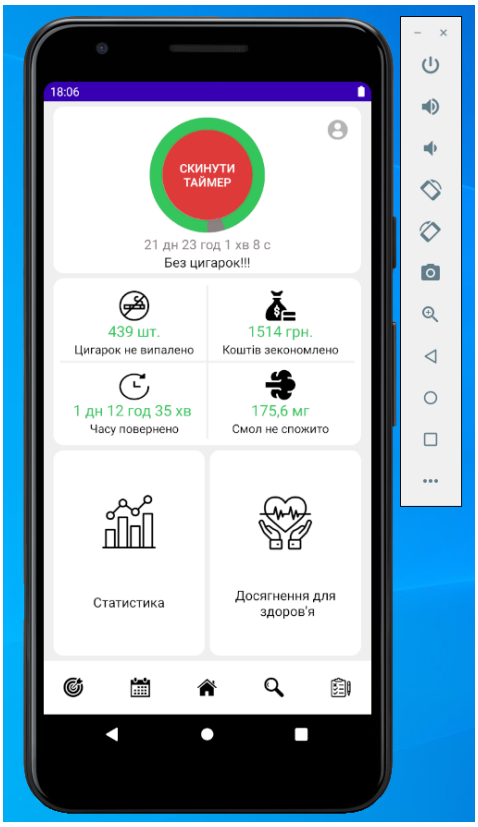

Рисунок 4.1 – Програмний інтерфейс Android Emulator з запущенним додатком SmokeBreak

4.1.2 Розгортання додатку на мобільному пристрої

4.1.2.1 Розгортання файлу збірки APK

Для встановлення програмного продукту на мобільний пристрій Android необхідно створити файл збірки формату APK. Це можна зробити в середовищі розробки Andoid Studio, яке дозволяє розробнику створити файл збірки, який можна завантажити на пристрій, створити новий підписаний ключ, що дозволяє здійснити аутентифікацію розробника, а також визначити сховище ключів застосувань, захищене паролем. Після створення файлу збірки APK необхідно завантажити його на мобільний пристрій через кабель USB або іншим доступним способом. Завантаживши файл збірки на мобільний пристрій, можна встановити програмний додаток. Більшість операційних систем на платформі Android містять інтегрований засіб інсталяції, що дозволяє встановлювати застосування з файлів збірки APK.

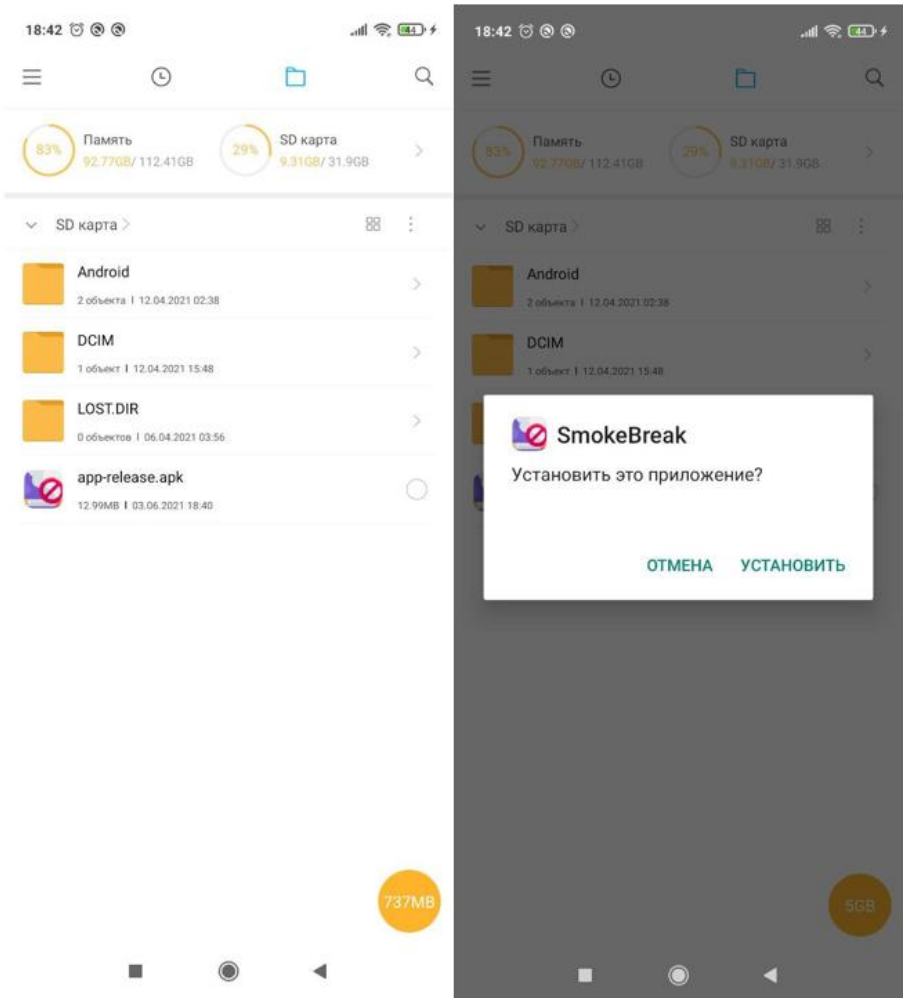

Рисунок 4.2 – Встановлення програмного додатку SmokeBreak з файлу збірки APK на мобільному пристрої Xiaomi Redmi Note 9 Pro

4.1.2.2 Встановлення застосування на мобільний пристрій за допомогою Android Studio через USB.

Для розгортання програмного продукту на мобільний пристрій через USB необхідно активувати режим розробника. В залежності від операційної системи, кроки для відтворення можуть різнитися, але в більшості систем активувати режим розробника можна, натиснувши десять разів на версію операційної системи в налаштуваннях пристрою. Після активації режиму розробника, у вкладці «Для розробників» необхідно увімкнути налаштування «Відладка по USB» та «Встановлення по USB».

Після цього необхідно під’єднати мобільний пристрій до комп’ютера з встановленим середовищем розробки Android Studio за допомогою кабелю USB. Після цього необхідно обрати мобільний телефон в меню пристроїв Android Studio та натиснути «Run». Середовище розробки виконає збірку програмного продукту та встановить його на мобільний пристрій.

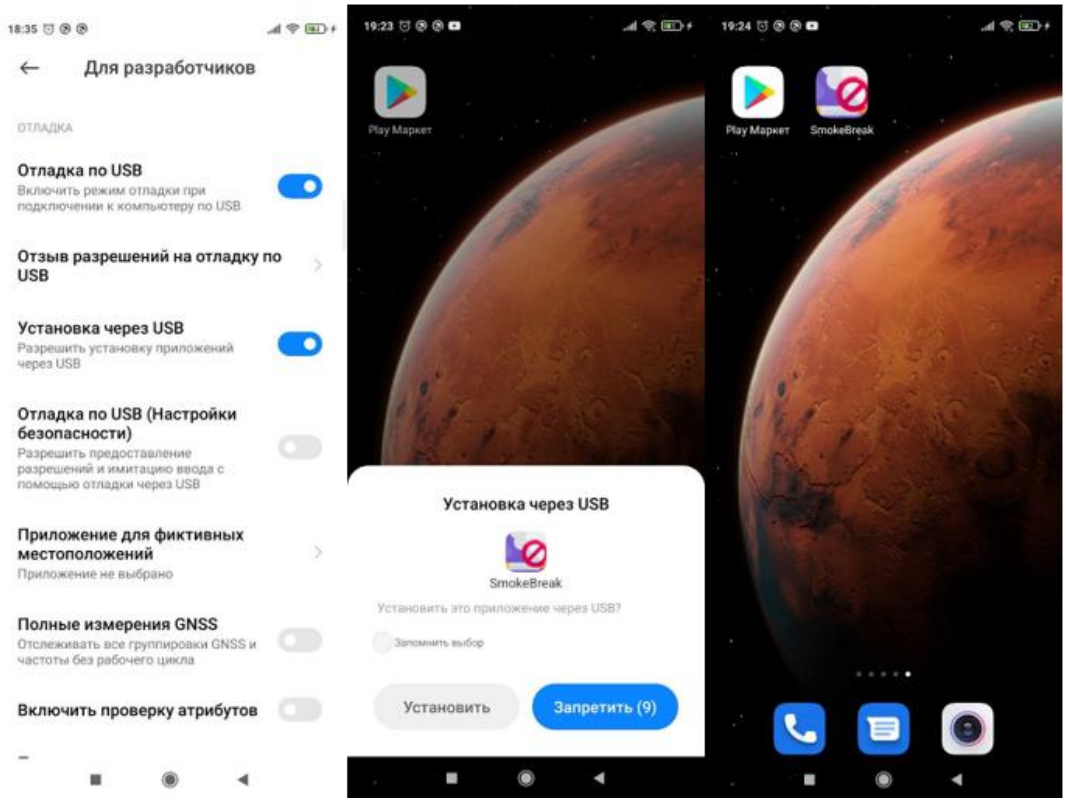

Рисунок 4.3 – Встановлення програмного додатку SmokeBreak на мобільному пристрої Xiaomi Redmi Note 9 Pro за допомогою Android Studio через кабель USB

4.1.3 Публікація застосунку в Google Play Market

Для публікації програмного продукту для загального доступу необхідно викласти його до сервісу Google Play Market. Після публікації застосунок буде доступний для встановлення всіма користувачами без попередньої реєстрації.

Для публікації застосунку необхідно відтворити наступні кроки :

– Створити аккаунт розробника в Google Play DeveloperConsole. Для реєстрації аккаунту розробника необхнідно здійснити одноразовий внесок в 25 долларів та підтвердити свою особистість за допомогою документу, що її підтверджує.

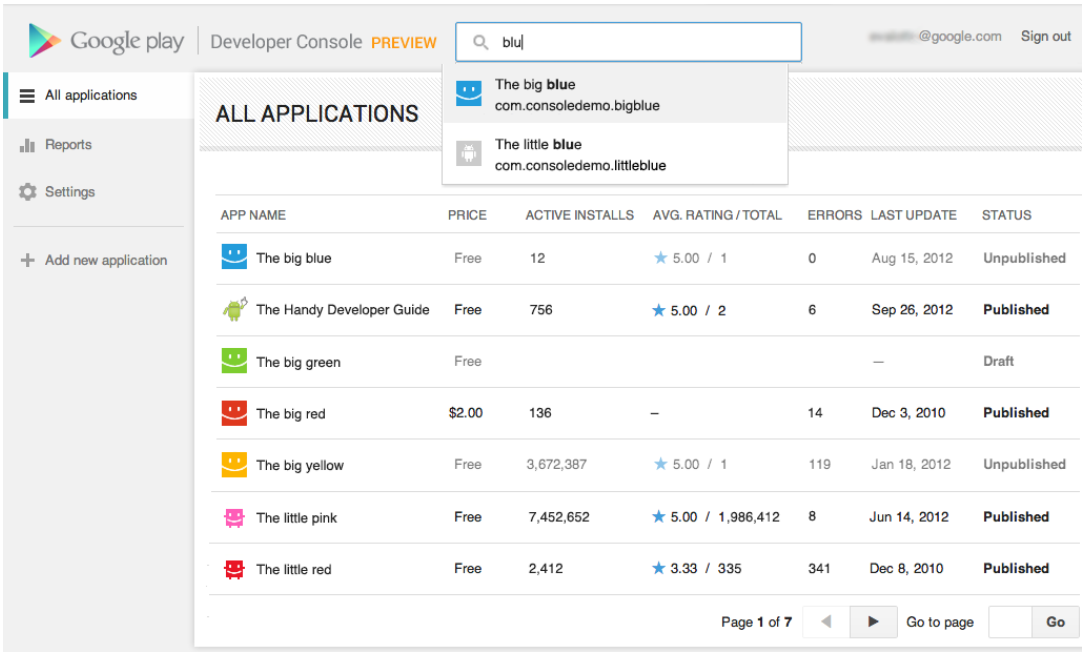

Рисунок 4.4 – Інтерфейс сервісу Google Play Develop Console

– Створити нове застосування. Потрібно заповнити анкету застосування : вказати назву та опис програмного продукту, встановити мову – українська. Оскільки «SmokeBreak» є застосунком індивідуальної боротьби з тютюнопалінням, у вкладці «Вікові обмеження» необхідно вказати вікову категорію від 18 років та підтвердити наявність в застосунку матеріалів, пов’язаних з палінням та вживанням тютюну. Заповнення решти граф анкети є тривіальним.
– Завантажити в створене застосування матеріали застосунку : файл збірки APK з програмним продуктом, іконки для смартфонів, скріншоти, банер. Створити файл збірки можна відтворивши кроки, наведені в підпункті 4.1.2.1.

Рисунок 4.5 – Іконка мобільного застосування для індивідуальної боротьби з тютюнопалінням «SmokeBreak»

– Опублікувати застосунок в сервіс Google Play Market. Коли застосунок пройде модерацію, застосунок стане доступним для встановлення для всіх користувачів сервісу і його можна буде знайти в пошуку за назвою. Модерація займає від кількох днів до місяця.

4.2 Робота з програмним забезпеченням

Детальний опис роботи з програмним забезпеченням подано в документі «Керівництво користувача».

4.3 Висновки до розділу

У рамках даного розділу було наведено методики розгортання програмного забезпечення наступними способами : розгортання в емуляторі мобільного пристрою, розгортання на мобільному пристрої та публікація в Google Play Market. Також було наведено керівництво користувача.

ВИСНОВКИ

В рамках виконання дипломного проєкту було розроблено мобільне застосування для індивідуальної боротьби з тютюнопалінням «SmokeBreak».

Дане застосування є актуальним, зважаючи на поширеність шкідливої звички тютюнопаління серед дорослого населення України, серйозність шкоди для здоров’я курця та складність боротьби з тютюновою залежністю зусиллями сили волі без застосування методів відмови від паління.

В ході аналізу вимог до програмного забезпечення було здійснено аналіз предметної області, огляд існуючих аналогів, на основі чого були сформовані варіанти використання програмного забезпечення, функціональні та нефункціональні вимоги до нього.

На етапі моделювання та конструювання програмного забезпечення було обрано архітектуру застосування та технології розробки, спроектовано структуру програмного коду та змодельовано базу даних. За результатами цього етапу було розроблено програмний продукт.

На етапі аналізу якості та тестування програмного забезпечення було визначено критерії якості проектного застосування та подано план тестування програмного продукту. За результатом проведеного тестування було зроблено висновок, що розроблене застосування відповідає критеріям якості.

Після цього, в ході впровадження та супроводу програмного забезпечення було описано алгоритми встановлення та запуску застосування в емуляторі та мобільному пристрої, а також подано інструкцію використання мобільного застосунку користувачем.

В ході роботи було досягнутно всіх поставлених задач, а розроблене мобільне застосування відповідає поставленим перед ним вимогам.

ПЕРЕЛІК ПОСИЛАНЬ

1) Залежність від паління [Електронний ресурс]: (Стаття) / StopSmoking – Електрон. дан. (1 файл) – 2021 – Режим доступу : http://stopsmoking.org.ua/addiction/zalezhnist-vid-kurinnya/. – Назва з екрану
2) Поширеність паління в Україні [Електронний ресурс]: (Стаття) / Міністерство охорони здоров’я – Електрон. дан. (1 файл) – 2021 – Режим доступу : https://moz.gov.ua/article/news/poshirenist-kurinnja-v-ukraini-zmenshilas-na-20
3) MVP [Електронний ресурс]: (Стаття) / StopSmoking – Електрон. дан. (1 файл) – 2021 – Режим доступу : https://habr.com/ru/post/278769/. – Назва з екрану
4) Realm [Електронний ресурс]: (Стаття) / StopSmoking – Електрон. дан. (1 файл) – 2021 – Режим доступу : https://docs.mongodb.com/realm/get-started/introduction-mobile/. – Назва з екрану
5) NoSQL [Електронний ресурс]: (Стаття) / StopSmoking – Електрон. дан. (1 файл) – 2021 – Режим доступу : https://habr.com/ru/post/152477/. – Назва з екрану
6) Android Emulator [Електронний ресурс]: (Стаття) / StopSmoking – Електрон. дан. (1 файл) – 2021 – Режим доступу : https://developer.android.com/studio/run/emulator – Назва з екрану
7) Lewis, W.E. (2016). Software Testing and Continuous Quality Improvement (3rd ed.). - CRC Press. - pp.92-96 .
8) Android Emulator [Електронний ресурс]: (Стаття) / StopSmoking – Електрон. дан. (1 файл) – 2021 – Режим доступу : https://www.javatpoint.com/how-to-install-apk-on-android – Назва з екрану

ДОДАТОК А. ТЕКСТИ ПРОГРАМНОГО КОДУ

App.java

```
package com.smokebreak;

import android.app.Activity;
import android.app.Application;
import android.content.Context;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.gson.GsonBuilder;

import io.realm.Realm;
import io.realm.RealmConfiguration;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class App extends Application {

   private static App app;

   public static App getApp() {
      return app;
   }

   private Retrofit retrofit;

   private static final String BASE_URL = "https://smoke-break-admin.herokuapp.com/";
   @Override
   public void onCreate() {
      super.onCreate();
      Realm.init(this);
      RealmConfiguration config = new RealmConfiguration.Builder().deleteRealmIfMigrationNeeded()
            .name("smokeBreak")
            .allowWritesOnUiThread(true)
            .build();
      Realm.setDefaultConfiguration(config);

      app = this;

      retrofit = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create(new GsonBuilder().create()))
            .build();
   }

   public void showToastError(String error, Activity root){
      LayoutInflater inflater = LayoutInflater.from(root);
      View layout = inflater.inflate(R.layout.custom_toast_error,
            (ViewGroup)  root.findViewById(R.id.toast_layout_root));

      TextView text = (TextView) layout.findViewById(R.id.tv);
      text.setText(error);
```

```
        Toast toast = new Toast(getApplicationContext());
        toast.setGravity(Gravity.TOP, 0, 0);
        toast.setDuration(Toast.LENGTH_SHORT);
        toast.setView(layout);
        toast.show();
    }

    public Retrofit getRetrofit() {
        return retrofit;
    }
}
```

StartActivity.java

```
package com.smokebreak.ui.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

import com.smokebreak.R;
import com.smokebreak.db.UserData;
import com.smokebreak.presenter.MainPresenter;

import io.realm.Realm;

public class StartActivity extends AppCompatActivity {

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

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                if(Realm.getDefaultInstance().where(UserData.class).findFirst() != null &&
                        notEmpty(Realm.getDefaultInstance().where(UserData.class).findFirst())) {
                    startActivity(new Intent(StartActivity.this, MainActivity.class));
                }else {
                    startActivity(new Intent(StartActivity.this, ProfileActivity.class));
                }
                 finish();
            }
        },1500);
    }

    private boolean notEmpty( UserData userData){
        if (userData.getCigarettesInDay() > 0 &&
                userData.getCigarettesInBox() > 0 &&
                userData.getPricePerBox() > 0 &&
                userData.getSmallSize() > 0.0f)
            return true;

        return false;
    }
}
```

MainActivity.java

```
package com.smokebreak.ui.activity;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;
import androidx.viewpager.widget.PagerAdapter;
import androidx.viewpager.widget.ViewPager;

import android.app.Notification;
import android.app.NotificationManager;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;

import com.smokebreak.R;
import com.smokebreak.notifycation.NotifUtils;
import com.smokebreak.ui.fragments.InfoMainFragment;
import com.smokebreak.ui.fragments.JournalFragment;
import com.smokebreak.ui.fragments.MainFragment;
import com.smokebreak.ui.fragments.GoalFragment;
import com.smokebreak.ui.fragments.TestFragment;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private ViewPager mPager;
    private PagerAdapter pagerAdapter;

    private ImageView ivGoal,ivJournal,ivMain,ivInfo,ivTest;

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

    private void initViews(){
        mPager = findViewById(R.id.pager);
        pagerAdapter = new ScreenSlidePagerAdapter(getSupportFragmentManager());
        mPager.setAdapter(pagerAdapter);
        mPager.setCurrentItem(2);

        ivGoal = findViewById(R.id.ic_Goal);
        ivGoal.setOnClickListener(this);
        ivJournal = findViewById(R.id.ic_journal);
        ivJournal.setOnClickListener(this);
        ivMain = findViewById(R.id.ic_main);
        ivMain.setOnClickListener(this);
        ivInfo = findViewById(R.id.ic_info);
        ivInfo.setOnClickListener(this);
        ivTest = findViewById(R.id.ic_test);
        ivTest.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.ic_Goal:
                mPager.setCurrentItem(0);
                break;
```

```
            case R.id.ic_journal:
                mPager.setCurrentItem(1);
                break;
            case R.id.ic_main:
                mPager.setCurrentItem(2);
                break;
            case R.id.ic_info:
                mPager.setCurrentItem(3);
                break;
            case R.id.ic_test:
                mPager.setCurrentItem(4);
                break;
        }
    }

    private class ScreenSlidePagerAdapter extends FragmentPagerAdapter {

        private static final int NUM_PAGES = 5;

        public ScreenSlidePagerAdapter(FragmentManager fragmentManager) {
            super(fragmentManager);
        }

        @NonNull
        @Override
        public Fragment getItem(int position) {
            switch (position) {
                case 0:
                    return new GoalFragment();
                case 1:
                    return new JournalFragment();
                case 2:
                    return new MainFragment();
                case 3:
                    return new InfoMainFragment();
                case 4:
                    return new TestFragment();
            }
            return null;
        }

        @Override
        public int getCount() {
            return NUM_PAGES;
        }
    }
}
```

MainFragment.java

```
package com.smokebreak.ui.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.text.format.DateUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;

import com.smokebreak.R;
import com.smokebreak.presenter.IMain;
import com.smokebreak.presenter.MainPresenter;
import com.smokebreak.ui.activity.HealthActivity;
import com.smokebreak.ui.activity.ProfileActivity;
import com.smokebreak.ui.activity.StatisticsActivity;
import com.smokebreak.ui.dialog.AddJournalDialog;
import com.smokebreak.ui.dialog.InfoDetailDialog;
import com.smokebreak.ui.dialog.JournalInfoDialog;
import com.smokebreak.ui.view.ProfileDataView;
import com.mikhaellopez.circularprogressbar.CircularProgressBar;

import io.realm.Realm;

public class MainFragment extends Fragment implements View.OnClickListener, IMain,
AddJournalDialog.OnAddListener {

    private ImageView ivProfile;
    private ProfileDataView pdvCig, pdvMoney, pdvTime, pdvSmoll;
    private CircularProgressBar progressBar;
    private Button bTimer;
    private TextView tvDateNoSmoke;
    private ConstraintLayout clStatis, clHeath;

    private MainPresenter mainPresenter;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ViewGroup rootView = (ViewGroup) inflater.inflate(
                R.layout.fragment_main, container, false);

        initViews(rootView);

        return rootView;
    }

    @Override
    public void onResume() {
        super.onResume();
        mainPresenter.checkNotification(getContext());
    }

    private void initViews(View rootView){
        ivProfile = rootView.findViewById(R.id.iv_profile);
        ivProfile.setOnClickListener(this);

        progressBar = rootView.findViewById(R.id.circularProgressBar);
        progressBar.setProgressMax(100f);
        progressBar.setProgress(50);



        bTimer = rootView.findViewById(R.id.b_timer);
        bTimer.setOnClickListener(this);
        tvDateNoSmoke = rootView.findViewById(R.id.tv_date_no_smoking);
```

```
        pdvCig = rootView.findViewById(R.id.pdv_cig);
        pdvMoney = rootView.findViewById(R.id.pdv_money);
        pdvTime = rootView.findViewById(R.id.pdv_time);
        pdvSmoll = rootView.findViewById(R.id.pdv_small);

        clStatis = rootView.findViewById(R.id.cl_statistic);
        clStatis.setOnClickListener(this);
        clHeath = rootView.findViewById(R.id.cl_health);
        clHeath.setOnClickListener(this);

        mainPresenter = new MainPresenter(Realm.getDefaultInstance(),this);
        startTimer();

    }

    public void startTimer(){
        mainPresenter.startTimer(getActivity());
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.iv_profile:
                Intent toProf = new Intent(getActivity(), ProfileActivity.class);
                toProf.putExtra("type", "MainFragment");
                startActivity(toProf);
                break;
            case R.id.b_timer:
                AddJournalDialog dialog = AddJournalDialog.newInstance(System.currentTimeMillis());
                dialog.setGoalFragment(this, 0);
                dialog.show(getActivity().getSupportFragmentManager(), "dialog");
                break;
            case R.id.cl_statistic:
                startActivity(new Intent(getActivity(), StatisticsActivity.class));
                break;
            case R.id.cl_health:
//                Realm.getDefaultInstance().close();
                startActivity(new Intent(getActivity(), HealthActivity.class));
                break;
        }
    }

    @Override
    public void setProgress(int progress) {
        progressBar.setProgress(progress);
    }

    @Override
    public void setDateNoSmoke(String date) {
        tvDateNoSmoke.setText(date);
    }

    @Override
    public void setCig(int count) {
        pdvCig.setInt(count);
    }

    @Override
    public void setMoney(int count) {
        pdvMoney.setInt(count);
```

```
    }

    @Override
    public void setTime(String time) {
        pdvTime.setTime(time);
    }

    @Override
    public void setSmolle(float count) {
        pdvSmoll.setFloat(count);
    }

    @Override
    public void add(long date, String desc) {
        mainPresenter.addJournal(date,desc);
        mainPresenter.checkNotification(getContext());
        Fragment page = getActivity().getSupportFragmentManager().findFragmentByTag("android:switcher:" +
R.id.pager + ":" + 1);
        if (page != null) {
            ((JournalFragment)page).prepareCalendar(0,0);
        }
    }
}
```

MainPresenter.java

```
package com.smokebreak.presenter;

import android.app.Activity;
import android.content.Context;
import android.provider.CalendarContract;
import android.util.Log;

import com.smokebreak.R;
import com.smokebreak.db.GoalData;
import com.smokebreak.db.HealthData;
import com.smokebreak.db.JournalData;
import com.smokebreak.db.UserData;
import com.smokebreak.notifycation.NotifUtils;
import com.smokebreak.utils.CalendarUtils;
import com.smokebreak.utils.UserUtils;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Timer;
import java.util.TimerTask;

import io.realm.Realm;
import io.realm.RealmResults;

public class MainPresenter {

    private Realm realm;
    private IMain iMain;

    private UserData userData;

    private Timer timer;

    public MainPresenter(Realm realm, IMain iMain) {
        this.realm = realm;
        this.iMain = iMain;
```

```
}

private void checkProgress(){
    userData = realm.where(UserData.class).equalTo("id",1).findFirst();
    if (userData != null && userData.getLastSmokeDate() != 0){
        calculateProgress();
    }else
        iMain.setProgress(0);
}

private void calculateProgress(){
    iMain.setProgress(UserUtils.calculateProgress(userData));
}

public void startTimer(Activity activity){
    stTimer(activity);
}

private void prepareDate(){
    long diff = System.currentTimeMillis() - 1 - userData.getLastSmokeDate();
    iMain.setDateNoSmoke(CalendarUtils.getStrDate(diff));
}

private void stTimer(Activity activity){
    timer = new Timer();
    TimerTask timerTask = new TimerTask() {
        @Override
        public void run() {
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    checkProgress();
                    prepareDate();
                    checkCig();
                    checkMoney();
                    checkTime();
                    checkSmolle();
                }
            });
        }
    };
    timer.scheduleAtFixedRate(timerTask,0,1000);
}

public void addJournal(long date, String description) {
    realm.beginTransaction();
    JournalData journalData = new JournalData();
    journalData.setDate(date);
    journalData.setDescription(description);
    userData.setLastSmokeDate(date);
    realm.insertOrUpdate(userData);
    realm.insertOrUpdate(journalData);
    checkProgress();
    realm.commitTransaction();

    checkAvg(date);
    calculateAllGoal(date);

}

private void checkAvg(long date){
```

```
        realm.beginTransaction();
        UserData userData = realm.where(UserData.class).equalTo("id", 1).findFirst();
        userData.setAvgSessionTime(UserUtils.getAvgSessionTime(userData,date));
        userData.setAvgCigarettesSaved(UserUtils.getAvgCigarettesSaved(userData));
        userData.setAvgFundsSaved(UserUtils.getAvgFundsSaved(userData));
        userData.setAvgTimeSaved(UserUtils.getAvgTimeSaved(userData));
        userData.setCount(userData.getCount() + 1);
        realm.copyToRealmOrUpdate(userData);
        realm.commitTransaction();
    }

    private void calculateAllGoal(long time){
        RealmResults<GoalData> realmResults = realm.where(GoalData.class).findAll();
        ArrayList<GoalData> dataList = new ArrayList<>();
        dataList.addAll(realm.copyFromRealm(realmResults));

        for (GoalData data : dataList){
            if (!data.isDone()){
                data.setStartDate(time +1);
            }
        }
        realm.beginTransaction();
        realm.copyToRealmOrUpdate(dataList);
        realm.commitTransaction();

    }

    public void checkNotification(Context context){
        RealmResults<GoalData> realmGoal = realm.where(GoalData.class).findAll();
        ArrayList<GoalData> dataGoal = new ArrayList<>();
        dataGoal.addAll(realm.copyFromRealm(realmGoal));

        for (GoalData data: dataGoal){
            if (data.getStartDate() + data.getFinishDate() < Calendar.getInstance().getTimeInMillis()){
                Realm.getDefaultInstance().beginTransaction();
                data.setDone(true);
                Realm.getDefaultInstance().copyToRealmOrUpdate(data);
                Realm.getDefaultInstance().commitTransaction();
            }
            if (!data.isDone())
                new NotifUtils().setAlarm(context, data.getStartDate(),data.getFinishDate(),"Ціль
досягнута!",data.getType());
        }

        RealmResults<HealthData> realmHealth = realm.where(HealthData.class).findAll();
        ArrayList<HealthData> dataHealth = new ArrayList<>();
        dataHealth.addAll(realm.copyFromRealm(realmHealth));

        UserData userData = realm.where(UserData.class).equalTo("id", 1).findFirst();
        for (HealthData data: realmHealth){
                new NotifUtils().setAlarm(context, userData.getLastSmokeDate(),data.getEstimate(),"Нове досягнення
для здоро'я!",3);
        }
    }

    private void checkCig(){
        iMain.setCig(UserUtils.getNotCigCount(userData));
    }

    private void checkMoney(){
        iMain.setMoney(UserUtils.getMoney(userData));
```

```
    }

    private void checkTime(){
        iMain.setTime(CalendarUtils.getStrDate( UserUtils.getTime(userData)));
    }

    private void checkSmolle(){
        iMain.setSmolle(UserUtils.getSmoll(userData));
    }
}
```

IMain.java

```
package com.smokebreak.presenter;

public interface IMain {

    void setProgress(int progress);

    void setDateNoSmoke(String date);

    void setCig(int count);

    void setMoney(int count);

    void setTime(String time);

    void setSmolle(float count);
}
```

ProfileActivity.java

```
package com.smokebreak.ui.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.text.format.DateUtils;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;

import com.smokebreak.App;
import com.smokebreak.R;
import com.smokebreak.presenter.IProfile;
import com.smokebreak.presenter.ProfilePresenter;
import com.smokebreak.db.UserData;
import com.smokebreak.ui.view.ProfileChooseView;

import java.util.Calendar;
import java.util.Date;

import io.realm.Realm;

public class ProfileActivity extends AppCompatActivity implements View.OnClickListener,
        DatePickerDialog.OnDateSetListener, TimePickerDialog.OnTimeSetListener, IProfile {

    private Button bReady;
```

```
private String type;
private TextView tvDate;

private ProfileChooseView pcCigPerDay, pcCigInBox, pcPriceBox, pcSmallSize;

private Calendar dateAndTime= Calendar.getInstance();
private ProfilePresenter profilePresenter;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_profile);
   if (getIntent().getExtras() != null)
      type = getIntent().getExtras().getString("type");

   initViews();

}

private void initViews(){

   bReady = findViewById(R.id.b_ready);
   bReady.setOnClickListener(this);

   tvDate = findViewById(R.id.tv_date);
   tvDate.setOnClickListener(this);
   setInitialDateTime();

   pcCigPerDay = findViewById(R.id.pc_cigarette_per_day);
   pcCigInBox = findViewById(R.id.pc_cigarette_in_box);
   pcPriceBox = findViewById(R.id.pc_price_box);
   pcSmallSize = findViewById(R.id.pc_small_size);

   profilePresenter = new ProfilePresenter(Realm.getDefaultInstance(), this);
   profilePresenter.checkProfileData();
}

@Override
protected void onDestroy() {
   profilePresenter.onDestroy();
   super.onDestroy();
}

@Override
public void onClick(View v) {

   switch (v.getId()){
      case R.id.b_ready:

         if (!profilePresenter.savaData(dateAndTime.getTimeInMillis() ,pcCigPerDay.getInt(),
               pcCigInBox.getInt(),pcPriceBox.getInt(),pcSmallSize.getFloat())){
            showError();
            return;
         }
         if (!TextUtils.isEmpty(type)) {
            onBackPressed();
            return;
         }

         startActivity(new Intent(ProfileActivity.this, MainActivity.class));
         finish();
```

```
            break;

        case R.id.tv_date:
            showDate();
            break;
    }
}

private void showDate() {
    DatePickerDialog dialog = new DatePickerDialog(ProfileActivity.this, ProfileActivity.this,
            dateAndTime.get(Calendar.YEAR),
            dateAndTime.get(Calendar.MONTH),
            dateAndTime.get(Calendar.DAY_OF_MONTH));
    dialog.getDatePicker().setMaxDate(new Date().getTime());
    dialog.show();
}

private void showTime() {
    TimePickerDialog dialog = new TimePickerDialog(ProfileActivity.this, ProfileActivity.this,
            dateAndTime.get(Calendar.HOUR_OF_DAY),
            dateAndTime.get(Calendar.MINUTE), true);
    dialog.show();
}

private void setInitialDateTime() {
    tvDate.setText(DateUtils.formatDateTime(this,
            dateAndTime.getTimeInMillis() ,
            DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_YEAR
                    | DateUtils.FORMAT_SHOW_TIME));
}


@Override
public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
    dateAndTime.set(Calendar.YEAR, year);
    dateAndTime.set(Calendar.MONTH, month);
    dateAndTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);
    showTime();
}

@Override
public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    dateAndTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
    dateAndTime.set(Calendar.MINUTE, minute);
    setInitialDateTime();
}

@Override
public void setData(UserData userData) {
    dateAndTime.setTimeInMillis(userData.getLastSmokeDate());
    setInitialDateTime();
    pcCigPerDay.setInt(userData.getCigarettesInDay());
    pcCigInBox.setInt(userData.getCigarettesInBox());
    pcPriceBox.setInt(userData.getPricePerBox());
    pcSmallSize.setFloat(userData.getSmallSize());
}

private void showError(){
    App.getApp().showToastError("Вкажіть коректні дані",this);
}
```

}

ProfilePresenter.java

```
package com.smokebreak.presenter;

import com.smokebreak.db.GoalData;
import com.smokebreak.db.UserData;
import com.smokebreak.utils.CalendarUtils;

import java.util.ArrayList;

import io.realm.Realm;
import io.realm.RealmResults;

public class ProfilePresenter {

    private Realm realm;
    private UserData userData;
    private IProfile iProfile;

    public ProfilePresenter(Realm realm, IProfile iProfile) {
        this.realm = realm;
        this.iProfile = iProfile;
    }

    public boolean checkProfileData(){
         userData = realm.where(UserData.class).equalTo("id",1).findFirst();
        if (userData == null) {
            setDefault();
            return false;
        }
        iProfile.setData(userData);
        return true;
    }

    public boolean savaData(long lastSmokeDate, int cigPerDay, int cigInBox, int priceBox, float smallSize){

        if (cigPerDay == 0 || cigInBox == 0 || priceBox == 0 || smallSize == 0){
            return false;
        }

        userData =  realm.where(UserData.class).findFirst();
        realm.beginTransaction();
        userData.setLastSmokeDate(lastSmokeDate);
        userData.setCigarettesInBox(cigInBox);
        userData.setCigarettesInDay(cigPerDay);
        userData.setSmallSize(smallSize);
        userData.setPricePerBox(priceBox);
        realm.copyToRealmOrUpdate(userData);
        realm.commitTransaction();
        calculateAllGoal();
        return true;
    }

    private void setDefault(){
        realm.beginTransaction();
        userData = new UserData();
        userData.setLastSmokeDate(System.currentTimeMillis());
        userData.setCigarettesInBox(0);
        userData.setCigarettesInDay(0);
```

```
        userData.setSmallSize(0.0f);
        userData.setPricePerBox(0);
        realm.copyToRealmOrUpdate(userData);
        realm.commitTransaction();
//        iProfile.setData(userData);
    }

    public void onDestroy(){
        if (realm != null)
            realm.close();
    }

    private void calculateAllGoal(){
        RealmResults<GoalData> realmResults = realm.where(GoalData.class).findAll();
        ArrayList<GoalData> dataList = new ArrayList<>();
        dataList.addAll(realm.copyFromRealm(realmResults));

        for (GoalData data : dataList){
            if (!data.isDone()){
                long value = 0;
                if (data.getType() == 2)
                    value =  data.getTime();
                else
                    value = data.getCount();
                data.setFinishDate(calculateFinishGoal(data.getType(),value));
            }
        }
        realm.beginTransaction();
        realm.copyToRealmOrUpdate(dataList);
        realm.commitTransaction();

    }

    private long calculateFinishGoal(int type, long value){
        long finishTime = 0;
        UserData userData = realm.where(UserData.class).equalTo("id",1).findFirst();
        long oneCigTime = CalendarUtils.millisInDay / userData.getCigarettesInDay();
        float priceForCig = (float) userData.getPricePerBox() / userData.getCigarettesInBox();
        if (userData == null)
            return 0;
        switch (type){
            case 0:
                finishTime = oneCigTime * value;
                break;
            case 1:
                finishTime = (long) ((value/priceForCig) * oneCigTime);
                break;
            case 2:
                finishTime = (value / (300000)) * oneCigTime ;
                break;
        }

        return finishTime;
    }


}
}
```

StatisticsActivity.java

```
package com.smokebreak.ui.activity;
```

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

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import com.smokebreak.R;
import com.smokebreak.db.UserData;
import com.smokebreak.utils.CalendarUtils;

import io.realm.Realm;

public class StatisticsActivity extends AppCompatActivity implements View.OnClickListener {

    private ImageView ivBack;
    private Button bBack;
    private TextView tvTimeNoSmok, tvCig, tvMoney, tvTime;

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

    private void initView(){
        ivBack = findViewById(R.id.iv_back);
        ivBack.setOnClickListener(this);
        bBack = findViewById(R.id.b_back);
        bBack.setOnClickListener(this);

        tvTimeNoSmok = findViewById(R.id.tv_time_no_smoke);
        tvCig = findViewById(R.id.tv_cig);
        tvMoney = findViewById(R.id.tv_money);
        tvTime = findViewById(R.id.tv_time);
        setData();
    }

    private void setData(){
        UserData userData = Realm.getDefaultInstance().where(UserData.class).equalTo("id", 1).findFirst();
        tvTimeNoSmok.setText(CalendarUtils.getStrDate(userData.getAvgSessionTime()));
        tvCig.setText(userData.getAvgCigarettesSaved()+" шт.");
        tvMoney.setText(userData.getAvgFundsSaved() +" грн.");
        tvTime.setText(CalendarUtils.getStrDate(userData.getAvgTimeSaved()));
    }

    @Override
    public void onClick(View v) {
        finish();
    }
}
```

HealthActivity.java

```
package com.smokebreak.ui.activity;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ProgressBar;

import com.smokebreak.App;
import com.smokebreak.R;
import com.smokebreak.api.SmokeApi;
import com.smokebreak.db.HealthData;
import com.smokebreak.ui.adapters.HealthAdapter;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import io.realm.Realm;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class HealthActivity extends AppCompatActivity implements View.OnClickListener {

    private FloatingActionButton fab;
    private ImageView ivBack;
    private Button bBack;

    private RecyclerView recyclerView;
    private HealthAdapter healthAdapter;
    private ProgressBar progressBar;

    private Realm realm = Realm.getDefaultInstance();;

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

        initView();
        showAdapter();
    }

    private void initView(){
        recyclerView = findViewById(R.id.rv);
        recyclerView.setHasFixedSize(true);
        LinearLayoutManager llm = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(llm);


        ivBack = findViewById(R.id.iv_back);
        ivBack.setOnClickListener(this);
        bBack = findViewById(R.id.b_back);
        bBack.setOnClickListener(this);

        fab = findViewById(R.id.fab);
        fab.setOnClickListener(this);
```

```
        progressBar = findViewById(R.id.progressBar);

    }

    private void loadJSONFromAsset() {
        try {
            InputStream is = getAssets().open("health.json");
            realm.beginTransaction();
            realm.where(HealthData.class).findAll().deleteAllFromRealm();
            realm.createOrUpdateAllFromJson(HealthData.class,is);
            realm.commitTransaction();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    private void showAdapter(){
        ArrayList<HealthData> dataList = new ArrayList<>();
        dataList.addAll(realm.where(HealthData.class).findAll());
        healthAdapter = new HealthAdapter(dataList);
        recyclerView.setAdapter(healthAdapter);
        recyclerView.setVisibility(View.VISIBLE);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.fab:
                getHealthData();
                break;
            case R.id.iv_back:
            case R.id.b_back:
                finish();
                break;
        }
    }

   private void getHealthData(){
        recyclerView.setVisibility(View.INVISIBLE);
        progressBar.setVisibility(View.VISIBLE);
        App.getApp().getRetrofit().create(SmokeApi.class).getHealths().enqueue(new Callback<List<HealthData>>() {
            @Override
            public void onResponse(Call<List<HealthData>> call, Response<List<HealthData>> response) {
                healthAdapter = new HealthAdapter(response.body());
                realm.beginTransaction();
                realm.insertOrUpdate(response.body());
                realm.commitTransaction();
                recyclerView.setAdapter(healthAdapter);
                recyclerView.setVisibility(View.VISIBLE);
                progressBar.setVisibility(View.INVISIBLE);
            }

            @Override
            public void onFailure(Call<List<HealthData>> call, Throwable t) {

            }
        });
    }
}
```

GoalFragment.java

```
package com.smokebreak.ui.fragments;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.smokebreak.R;
import com.smokebreak.presenter.IGoal;
import com.smokebreak.presenter.GoalPresenter;
import com.smokebreak.ui.adapters.GoalAdapter;
import com.smokebreak.db.GoalData;
import com.smokebreak.ui.dialog.AddGoalDialog;
import com.smokebreak.ui.dialog.GoalInfoDialog;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;

import io.realm.Realm;

public class GoalFragment extends Fragment implements View.OnClickListener,
        AddGoalDialog.OnAddListener, GoalAdapter.OnGoalClickListener, IGoal, GoalInfoDialog.OnDeleteListener
{

    private FloatingActionButton fab;

    private RecyclerView recyclerView;

    private ImageView ivHolder;
    private TextView tvHolder;

    private GoalAdapter adapter;

    private GoalPresenter presenter;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ViewGroup rootView = (ViewGroup) inflater.inflate(
                R.layout.fragment_goal, container, false);
        initViews(rootView);
        return rootView;
    }

    private void initViews(View parent) {

        recyclerView = parent.findViewById(R.id.rv);
        recyclerView.setHasFixedSize(true);
        recyclerView.setVisibility(View.INVISIBLE);
        LinearLayoutManager llm = new LinearLayoutManager(getActivity());
        recyclerView.setLayoutManager(llm);

        ivHolder = parent.findViewById(R.id.iv_holder);
        tvHolder = parent.findViewById(R.id.tv_holder);
```

```
        fab = parent.findViewById(R.id.fab);
        fab.setOnClickListener(this);

        presenter = new GoalPresenter(Realm.getDefaultInstance(), this);
        presenter.checkGoal();
    }

    @Override
    public void onClick(View v) {
        AddGoalDialog dialog = new AddGoalDialog();
        dialog.setGoalFragment(this, 0);
        dialog.show(getActivity().getSupportFragmentManager(), "dialog");
    }


    @Override
    public void onAdd(String inputText, int type, int pcv100, int pcv10, int pcv1) {
        presenter.addGoal(getContext(),inputText, type, pcv100, pcv10, pcv1);
        presenter.checkGoal();
    }

    @Override
    public void showEmpty() {
        recyclerView.setVisibility(View.GONE);
        ivHolder.setVisibility(View.VISIBLE);
        tvHolder.setVisibility(View.VISIBLE);
    }

    @Override
    public void showList(ArrayList<GoalData> arrayList) {
        adapter = new GoalAdapter(arrayList, this);
        recyclerView.setAdapter(adapter);
        adapter.notifyDataSetChanged();
        recyclerView.setVisibility(View.VISIBLE);
        ivHolder.setVisibility(View.GONE);
        tvHolder.setVisibility(View.GONE);
    }

    @Override
    public void onGoalClick(int position, long id) {
        GoalInfoDialog dialog = GoalInfoDialog.newInstance(id);
        dialog.setGoalFragment(this, 0);
        dialog.show(getActivity().getSupportFragmentManager(), "dialog");
    }

    @Override
    public void onDelete(long startTime) {
        presenter.delete(startTime);
    }
}
```

GoalPresenter.java

```
package com.smokebreak.presenter;

import android.content.Context;
import android.util.Log;

import com.smokebreak.db.GoalData;
import com.smokebreak.db.UserData;
import com.smokebreak.notifycation.NotifUtils;
import com.smokebreak.utils.CalendarUtils;
```

```
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import io.realm.Realm;
import io.realm.RealmResults;

public class GoalPresenter {

    private Realm realm;
    private IGoal iGoal;

    private List<GoalData> dataList;

    public GoalPresenter(Realm realm, IGoal iGoal) {
        this.realm = realm;
        this.iGoal = iGoal;
    }

    public void checkGoal(){
        RealmResults<GoalData> realmResults = realm.where(GoalData.class).findAll();
        if (realmResults == null || realmResults.isEmpty())
            iGoal.showEmpty();
        else {
            ArrayList<GoalData> dataList = new ArrayList<>();
            dataList.addAll(realm.copyFromRealm(realmResults));
            iGoal.showList(dataList);
        }
    }

    public void addGoal(Context context,String inputText, int type, int pcv100, int pcv10, int pcv1){
        realm.beginTransaction();
        GoalData goalData = new GoalData();
        if (realm.where(GoalData.class).max("id") != null)
            goalData.setId(realm.where(GoalData.class).max("id").intValue() +1);
        else
            goalData.setId(0);
        goalData.setDescription(inputText);
        goalData.setType(type);
        goalData.setStartDate(System.currentTimeMillis());
        goalData.setFinishDate(calculateFinish(type,pcv100,pcv10,pcv1));

        if (type == 2)
            goalData.setTime(pcv100 * CalendarUtils.monthInMilli + pcv10 * CalendarUtils.daysInMilli + pcv1 *
CalendarUtils.hoursInMilli);
        else
            goalData.setCount(pcv100 + pcv10 + pcv1);

        realm.insertOrUpdate(goalData);
        realm.commitTransaction();

        new NotifUtils().setAlarm(context, goalData.getStartDate(),goalData.getFinishDate(),"Ціль
досягнута!",goalData.getType());

    }

    public void delete(long startDate){
        GoalData goalData = realm.where(GoalData.class).equalTo("startDate",startDate).findFirst();

        if (goalData != null){
```

```
            realm.beginTransaction();
            goalData.deleteFromRealm();
            realm.commitTransaction();
        }

        checkGoal();
    }



    private long calculateFinish(int type, int pcv100, int pcv10, int pcv1){
        long finishTime = 0;
        UserData userData = realm.where(UserData.class).equalTo("id",1).findFirst();
        long oneCigTime = CalendarUtils.millisInDay / userData.getCigarettesInDay();
        float priceForCig = (float) userData.getPricePerBox() / userData.getCigarettesInBox();
        if (userData == null)
            return 0;
       switch (type){
           case 0:
               finishTime = oneCigTime * (pcv100 + pcv10 + pcv1);
               break;
           case 1:
               finishTime = (long) (((pcv100 + pcv10 + pcv1)/priceForCig) * oneCigTime);
               break;
           case 2:
               finishTime = ((pcv100 * CalendarUtils.monthInMilli +
                       pcv10 * CalendarUtils.daysInMilli +
                       pcv1 * CalendarUtils.hoursInMilli) / (300000)) * oneCigTime ;
               break;
       }

       return finishTime;
    }
}
```

AddGoalDialog.java

```
package com.smokebreak.ui.dialog;

import android.content.Context;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;

import androidx.fragment.app.DialogFragment;

import com.smokebreak.App;
import com.smokebreak.R;
import com.smokebreak.ui.view.ProfileChooseView;

public class AddGoalDialog extends DialogFragment implements View.OnClickListener {

    private Button bAdd;
```

```
private LinearLayout llSigarets, llMoney, llTime;
private ImageView ivSigaret, ivMoney, ivTime;
private EditText etDesc;

private ProfileChooseView pcv100,pcv10,pcv1;

private OnAddListener listener;

private int type = 0;

public interface OnAddListener {
   void onAdd(String inputText, int type, int pcv100, int pcv10, int pcv1);
}

public AddGoalDialog(){}

@Override
public void onAttach(Context context) {
   super.onAttach(context);
   // Verify that the host activity implements the callback interface
   try {
      // Instantiate the EditNameDialogListener so we can send events to the host
      listener = (OnAddListener) getGoalFragment();
   } catch (ClassCastException e) {
      // The activity doesn't implement the interface, throw exception
      throw new ClassCastException(context.toString()
            + " must implement EditNameDialogListener");
   }
}

public View onCreateView(LayoutInflater inflater, ViewGroup container,
                  Bundle savedInstanceState) {

   View view = inflater.inflate(R.layout.add_goal_dialog, container);

   if (getDialog() != null && getDialog().getWindow() != null) {
      getDialog().getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
      getDialog().getWindow().requestFeature(Window.FEATURE_NO_TITLE);
   }

   initView(view);

   return view;
}

private void initView(View view){
   bAdd = view.findViewById(R.id.b_add);
   bAdd.setOnClickListener(this);

   llSigarets = view.findViewById(R.id.ll_cig);
   llSigarets.setOnClickListener(this);
   llMoney = view.findViewById(R.id.ll_money);
   llMoney.setOnClickListener(this);
   llTime = view.findViewById(R.id.ll_time);
   llTime.setOnClickListener(this);

   ivSigaret = view.findViewById(R.id.iv_cig);
   ivMoney = view.findViewById(R.id.iv_money);
   ivTime = view.findViewById(R.id.iv_time);

   pcv100 = view.findViewById(R.id.pcv_100);
```

```
    pcv100.setTitleGone();
    pcv100.replaceBackground();
    pcv10 = view.findViewById(R.id.pcv_10);
    pcv10.setTitleGone();
    pcv10.replaceBackground();
    pcv1 = view.findViewById(R.id.pcv_1);
    pcv1.setTitleGone();
    pcv1.replaceBackground();

    etDesc = view.findViewById(R.id.edit_text);

    prepareChoosers();
    llSigarets.setBackgroundResource(R.drawable.ll_background_green);
    ivSigaret.setColorFilter(getResources().getColor(R.color.green), PorterDuff.Mode.SRC_IN);
}

@Override
public void onClick(View v) {
    if (v.getId() != R.id.b_add) {
        resetBackground();
        resetValue();
    }
    switch (v.getId()){
        case R.id.ll_cig:
            type = 0;
            prepareChoosers();
            llSigarets.setBackgroundResource(R.drawable.ll_background_green);
            ivSigaret.setColorFilter(getResources().getColor(R.color.green), PorterDuff.Mode.SRC_IN);
            break;
        case R.id.ll_money:
            type = 1;
            prepareChoosers();
            llMoney.setBackgroundResource(R.drawable.ll_background_green);
            ivMoney.setColorFilter(getResources().getColor(R.color.green), PorterDuff.Mode.SRC_IN);
            break;
        case R.id.ll_time:
            type = 2;
            prepareChoosers();
            llTime.setBackgroundResource(R.drawable.ll_background_green);
            ivTime.setColorFilter(getResources().getColor(R.color.green), PorterDuff.Mode.SRC_IN);
            break;
        case R.id.b_add:
            if (pcv100.getInt() == 0 && pcv10.getInt() == 0 && pcv1.getInt() == 0){
                showError();
                return;
            }

            if (listener != null)
                listener.onAdd(etDesc.getText().toString(),type,pcv100.getInt(),pcv10.getInt(),pcv1.getInt());
            dismiss();
            break;
    }
}

private void prepareChoosers(){
    switch (type){
        case 0:
            pcv100.setIntCount(100);
            pcv100.setDesc("100 шт.");
            pcv10.setIntCount(10);
            pcv10.setDesc("10 шт.");
```

```
            pcv1.setIntCount(1);
            pcv1.setDesc("шт.");
            break;
        case 1:
            pcv100.setIntCount(100);
            pcv100.setDesc("100 грн.");
            pcv10.setIntCount(10);
            pcv10.setDesc("10 грн.");
            pcv1.setIntCount(1);
            pcv1.setDesc("грн.");
            break;
        case 2:
            pcv100.setIntCount(1);
            pcv100.setDesc("м.");
            pcv10.setIntCount(1);
            pcv10.setDesc("дн.");
            pcv1.setIntCount(1);
            pcv1.setDesc("год.");
            break;
    }
  }
  private void resetBackground() {
    llSigarets.setBackgroundResource(R.drawable.ll_background_gray);
    llMoney.setBackgroundResource(R.drawable.ll_background_gray);
    llTime.setBackgroundResource(R.drawable.ll_background_gray);

    ivSigaret.setColorFilter(getResources().getColor(R.color.black), PorterDuff.Mode.SRC_IN);
    ivMoney.setColorFilter(getResources().getColor(R.color.black), PorterDuff.Mode.SRC_IN);
    ivTime.setColorFilter(getResources().getColor(R.color.black), PorterDuff.Mode.SRC_IN);
  }

  private void resetValue(){
    pcv100.setInt(0);
    pcv10.setInt(0);
    pcv1.setInt(0);
  }

  private void showError(){
    App.getApp().showToastError("Вкажіть коректні дані",getActivity());
  }
}
```

GoalInfoDialog.java

```
package com.smokebreak.ui.dialog;

import android.content.Context;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.DisplayMetrics;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.fragment.app.DialogFragment;
```

```
import com.smokebreak.R;
import com.smokebreak.db.GoalData;
import com.smokebreak.utils.CalendarUtils;

import io.realm.Realm;

public class GoalInfoDialog extends DialogFragment implements View.OnClickListener {

    private  View view;
    private Button bDelete;
    private ImageView iv;
    private TextView tvGoal,tvDesc;

    private OnDeleteListener listener;

    private GoalData data;

    public interface OnDeleteListener {
        void onDelete(long startTime);
    }

    public GoalInfoDialog(){}

    public static GoalInfoDialog newInstance(long id) {
        GoalInfoDialog f = new GoalInfoDialog();
        // Supply num input as an argument.
        Bundle args = new Bundle();
        args.putLong("id", id);
        f.setArguments(args);

        return f;
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        // Verify that the host activity implements the callback interface
        try {
            // Instantiate the EditNameDialogListener so we can send events to the host
            listener = (OnDeleteListener) getGoalFragment();
        } catch (ClassCastException e) {
            // The activity doesn't implement the interface, throw exception
            throw new ClassCastException(context.toString()
                    + " must implement EditNameDialogListener");
        }
    }

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

         view = inflater.inflate(R.layout.goal_info_dialog, container);

        if (getDialog() != null && getDialog().getWindow() != null) {
            getDialog().getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
            getDialog().getWindow().requestFeature(Window.FEATURE_NO_TITLE);
        }

        initView(view);

        return view;
    }
```

```
    @Override
    public void onResume() {
        super.onResume();
        DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
        getDialog().getWindow().setLayout((int) (displayMetrics.widthPixels*0.8),
                view.getLayoutParams().WRAP_CONTENT);
    }

    private void initView(View view){
        bDelete = view.findViewById(R.id.b_delete);
        bDelete.setOnClickListener(this);

        iv = view.findViewById(R.id.iv);
        tvGoal = view.findViewById(R.id.tv_goal);
        tvDesc = view.findViewById(R.id.tv_desc);

        setData();
    }

    private void setData(){
         data = Realm.getDefaultInstance().where(GoalData.class).
                equalTo("startDate", getArguments().getLong("id")).findFirst();

        if (data != null){
            switch (data.getType()){
                case 0:
                    tvGoal.setText("Не скурити " + data.getCount() + " цг.");
                    iv.setImageResource(R.drawable.ic_no_smoking);
                    break;
                case 1:
                    tvGoal.setText("Зекономити " + data.getCount() + " грн.");
                    iv.setImageResource(R.drawable.ic_money);
                    break;
                case 2:
                    tvGoal.setText("повернути " + CalendarUtils.getStrDate(data.getTime()));
                    iv.setImageResource(R.drawable.ic_time);
                    break;
            }
            if (!TextUtils.isEmpty(data.getDescription())) {
                tvDesc.setText(data.getDescription());
            }else
                tvDesc.setVisibility(View.GONE);
        }
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.b_delete:
                if (listener != null)
                    listener.onDelete(data.getStartDate());
                dismiss();
                break;
        }
    }

}
```

JournalFragment.java

```
package com.smokebreak.ui.fragments;
```

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.smokebreak.R;
import com.smokebreak.presenter.IJournal;
import com.smokebreak.presenter.JournalPresenter;
import com.smokebreak.ui.adapters.JournalAdapter;
import com.smokebreak.db.JournalData;
import com.smokebreak.ui.dialog.AddJournalDialog;
import com.smokebreak.ui.dialog.JournalInfoDialog;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.prolificinteractive.materialcalendarview.CalendarDay;
import com.prolificinteractive.materialcalendarview.DayViewDecorator;
import com.prolificinteractive.materialcalendarview.DayViewFacade;
import com.prolificinteractive.materialcalendarview.MaterialCalendarView;
import com.prolificinteractive.materialcalendarview.OnDateSelectedListener;
import com.prolificinteractive.materialcalendarview.OnMonthChangedListener;

import java.util.ArrayList;
import java.util.Calendar;

import io.realm.Realm;

public class JournalFragment extends Fragment implements View.OnClickListener, IJournal,
        AddJournalDialog.OnAddListener, JournalAdapter.OnJournalClickListener {

    private MaterialCalendarView calendarView;
    private FloatingActionButton fab;
    private ConstraintLayout clHolder;

    private JournalPresenter journalPresenter;

    private RecyclerView recyclerView;
    private JournalAdapter adapter;

    private CalendarDay dateChoose = CalendarDay.from(Calendar.getInstance());;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ViewGroup rootView = (ViewGroup) inflater.inflate(
                R.layout.fragment_journal, container, false);

        initViews(rootView);

        return rootView;
    }

    private void initViews(View parent) {
        calendarView = parent.findViewById(R.id.calendarView);
        calendarView.state().edit().setMaximumDate(dateChoose).commit();
```

```
        clHolder = parent.findViewById(R.id.cl_holder);

        recyclerView = parent.findViewById(R.id.rv);
        recyclerView.setHasFixedSize(true);
        recyclerView.setVisibility(View.INVISIBLE);
        LinearLayoutManager llm = new LinearLayoutManager(getActivity());
        recyclerView.setLayoutManager(llm);

        fab = parent.findViewById(R.id.fab);
        fab.setOnClickListener(this);

        calendarView.setSelectionMode(MaterialCalendarView.SELECTION_MODE_SINGLE);


        journalPresenter = new JournalPresenter(Realm.getDefaultInstance(), this);

        prepareCalendar(0, 0);


        calendarView.addDecorator(new DayViewDecorator() {
            @Override
            public boolean shouldDecorate(CalendarDay day) {
                return hasYellow(day);
            }

            @Override
            public void decorate(DayViewFacade view) {
                view.setBackgroundDrawable(getResources().getDrawable(R.drawable.goal_yellow_selector));
            }
        });

        calendarView.addDecorator(new DayViewDecorator() {
            @Override
            public boolean shouldDecorate(CalendarDay day) {
                return hasRed(day);
            }

            @Override
            public void decorate(DayViewFacade view) {
                view.setBackgroundDrawable(getResources().getDrawable(R.drawable.goal_red_selector));
            }
        });



        calendarView.setOnDateChangedListener(new OnDateSelectedListener() {
            @Override
            public void onDateSelected(@NonNull MaterialCalendarView widget, @NonNull CalendarDay date,
boolean selected) {
                journalPresenter.notifyAdapter(date.getCalendar());
                dateChoose = date;
            }
        });

        calendarView.setOnMonthChangedListener(new OnMonthChangedListener() {
            @Override
            public void onMonthChanged(MaterialCalendarView widget, CalendarDay date) {
                prepareCalendar(date.getMonth(), date.getYear());
            }
        });
```

```
}

@Override
public void onResume() {
   super.onResume();
   prepareCalendar(dateChoose.getMonth(), dateChoose.getYear());
   calendarView.setCurrentDate(dateChoose);
}

public void prepareCalendar(int month, int year) {
   ArrayList<JournalData> arrayListMonth = new ArrayList<JournalData>();
   if (month == 0 && year == 00)
      arrayListMonth = journalPresenter.getMonthJournalData(
            Calendar.getInstance().get(Calendar.MONTH), Calendar.getInstance().get(Calendar.YEAR));
   else
      arrayListMonth = journalPresenter.getMonthJournalData(
            month, year);
   for (JournalData item : arrayListMonth) {
      Calendar calendar = Calendar.getInstance();
      calendar.setTimeInMillis(item.getDate());
      calendarView.setDateSelected(calendar, true);
   }
   calendarView.invalidateDecorators();
}

private boolean hasYellow(CalendarDay day) {
   ArrayList<JournalData> list = journalPresenter.getJournalByDate(day.getCalendar());
   if (list.size() > 0 && list.size() < 3)
      return true;
   else
      return false;
}

private boolean hasRed(CalendarDay day) {
   ArrayList<JournalData> list = journalPresenter.getJournalByDate(day.getCalendar());
   if (list.size() > 2)
      return true;
   else
      return false;
}


@Override
public void onClick(View v) {
   AddJournalDialog dialog;
   if (dateChoose == null)
      dialog = AddJournalDialog.newInstance(System.currentTimeMillis());
   else
      dialog = AddJournalDialog.newInstance(dateChoose.getCalendar().getTimeInMillis());

   dialog.setGoalFragment(this, 0);
   dialog.show(getActivity().getSupportFragmentManager(), "dialog");
}

@Override
public void showEmpty() {
   recyclerView.setVisibility(View.GONE);
   clHolder.setVisibility(View.VISIBLE);
}

@Override
```

```
    public void showList(ArrayList<JournalData> arrayList) {
        adapter = new JournalAdapter(arrayList,this);
        recyclerView.setAdapter(adapter);
        recyclerView.setVisibility(View.VISIBLE);
        clHolder.setVisibility(View.GONE);
    }

    @Override
    public void add(long date, String desc) {
        journalPresenter.addJournal(date, desc);
        prepareCalendar(dateChoose.getMonth(), dateChoose.getYear());
        journalPresenter.notifyAdapter(dateChoose.getCalendar());
        Fragment page = getActivity().getSupportFragmentManager().findFragmentByTag("android:switcher:" +
R.id.pager + ":" + 2);
        if (page != null) {
            ((MainFragment)page).startTimer();
        }
    }

    @Override
    public void onJournalClick(int position, long id) {
        JournalInfoDialog dialog = JournalInfoDialog.newInstance(id);
        dialog.setGoalFragment(this, 0);
        dialog.show(getActivity().getSupportFragmentManager(), "dialog");
    }
}
```

JournalPresenter.java

```
package com.smokebreak.presenter;

import androidx.fragment.app.Fragment;

import com.smokebreak.R;
import com.smokebreak.db.JournalData;
import com.smokebreak.db.UserData;
import com.smokebreak.ui.fragments.JournalFragment;
import com.smokebreak.utils.CalendarUtils;
import com.smokebreak.utils.UserUtils;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import io.realm.Realm;
import io.realm.RealmResults;

public class JournalPresenter {
    private Realm realm;
    private IJournal iJournal;

    public JournalPresenter(Realm realm, IJournal iJournal) {
        this.realm = realm;
        this.iJournal = iJournal;
    }

    public ArrayList<JournalData> getJournalByDate(Calendar day) {
        RealmResults<JournalData> realmResults = realm.where(JournalData.class)
                .greaterThan("date", CalendarUtils.getFirstTimeStampOfCurrentDay(day) - 10).and()
                .lessThan("date", CalendarUtils.getLastTimeStampOfCurrentDay( day)).findAll();
        ArrayList<JournalData> dataList = new ArrayList<>();
        dataList.addAll(realm.copyFromRealm(realmResults));
```

```
    return dataList;
}

public void notifyAdapter(Calendar day){
    RealmResults<JournalData> realmResults = realm.where(JournalData.class)
            .greaterThan("date", CalendarUtils.getFirstTimeStampOfCurrentDay(day) - 10).and()
            .lessThan("date", CalendarUtils.getLastTimeStampOfCurrentDay( day)).findAll();
    ArrayList<JournalData> dataList = new ArrayList<>();
    dataList.addAll(realm.copyFromRealm(realmResults));

    if (dataList.isEmpty())
        iJournal.showEmpty();
    else
        iJournal.showList(dataList);

}

public void addJournal(long date, String description) {

    checkUserSmokeDate(date);
    checkAvg(date);
    realm.beginTransaction();
    JournalData journalData = new JournalData();
    journalData.setDate(date);
    journalData.setDescription(description);
    realm.insertOrUpdate(journalData);
    realm.commitTransaction();

}

private void checkUserSmokeDate(long date){
    UserData userData = realm.where(UserData.class).equalTo("id", 1).findFirst();
    if (userData.getLastSmokeDate() < date) {
        realm.beginTransaction();
        userData.setLastSmokeDate(date);
        realm.copyToRealmOrUpdate(userData);
        realm.commitTransaction();
    }
}

private void checkAvg(long date){
    realm.beginTransaction();
    UserData userData = realm.where(UserData.class).equalTo("id", 1).findFirst();
    userData.setAvgSessionTime(UserUtils.getAvgSessionTime(userData,date));
    userData.setAvgCigarettesSaved(UserUtils.getAvgCigarettesSaved(userData));
    userData.setAvgFundsSaved(UserUtils.getAvgFundsSaved(userData));
    userData.setAvgTimeSaved(UserUtils.getAvgTimeSaved(userData));
    userData.setCount(userData.getCount() + 1);
    realm.copyToRealmOrUpdate(userData);
    realm.commitTransaction();
}

public ArrayList<JournalData> getMonthJournalData(int month, int year) {
    RealmResults<JournalData> realmResults = realm.where(JournalData.class)
            .greaterThan("date", CalendarUtils.getFirstTimeStampOfCurrentMonth(month , year)).and()
            .lessThan("date", CalendarUtils.getLastTimeStampOfCurrentMonth(month , year)).findAll();

    ArrayList<JournalData> dataList = new ArrayList<>();
    dataList.addAll(realm.copyFromRealm(realmResults));
    return dataList;
}
```

}

AddJournalDialog.java

```
package com.smokebreak.ui.dialog;

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.Context;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.format.DateUtils;
import android.util.DisplayMetrics;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.TimePicker;

import androidx.fragment.app.DialogFragment;

import com.smokebreak.R;
import com.smokebreak.ui.activity.ProfileActivity;

import java.util.Calendar;
import java.util.Date;

public class AddJournalDialog extends DialogFragment implements View.OnClickListener,
        DatePickerDialog.OnDateSetListener, TimePickerDialog.OnTimeSetListener{

    private Button bAdd;
    private  View view;

    private TextView tvDate;
    private EditText etDesc;

    private OnAddListener listener;
    private Calendar dateAndTime= Calendar.getInstance();

    public interface OnAddListener {
        void add(long date, String desc);
    }

    public AddJournalDialog(){}

    public static AddJournalDialog newInstance(long date) {
        AddJournalDialog f = new AddJournalDialog();

        // Supply num input as an argument.
        Bundle args = new Bundle();
        args.putLong("date", date);
        f.setArguments(args);

        return f;
```

```
}

@Override
public void onAttach(Context context) {
   super.onAttach(context);
   // Verify that the host activity implements the callback interface
   try {
      // Instantiate the EditNameDialogListener so we can send events to the host
      listener = (OnAddListener) getGoalFragment();
   } catch (ClassCastException e) {
      // The activity doesn't implement the interface, throw exception
      throw new ClassCastException(context.toString()
            + " must implement EditNameDialogListener");
   }
}

public View onCreateView(LayoutInflater inflater, ViewGroup container,
                  Bundle savedInstanceState) {

     view = inflater.inflate(R.layout.disruption_dialog, container);

   if (getDialog() != null && getDialog().getWindow() != null) {
      getDialog().getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
      getDialog().getWindow().requestFeature(Window.FEATURE_NO_TITLE);

   }

   initView(view);

   return view;
}

@Override
public void onResume() {
   super.onResume();
   DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
   getDialog().getWindow().setLayout((int) (displayMetrics.widthPixels*0.8),
         view.getLayoutParams().WRAP_CONTENT);
}

private void initView(View view){

   tvDate = view.findViewById(R.id.tv_date);
   tvDate.setOnClickListener(this);

   dateAndTime.setTimeInMillis(getArguments().getLong("date",System.currentTimeMillis()));

   tvDate.setText(DateUtils.formatDateTime(getActivity(),
         getArguments().getLong("date",System.currentTimeMillis()),
         DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_YEAR
               | DateUtils.FORMAT_SHOW_TIME));

   etDesc = view.findViewById(R.id.edit_text);

   bAdd = view.findViewById(R.id.b_add);
   bAdd.setOnClickListener(this);


}

@Override
```

```
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.tv_date:
                showDate();
                break;
            case R.id.b_add:
                if (listener != null)
                    listener.add(dateAndTime.getTimeInMillis(), etDesc.getText().toString());

                dismiss();
                break;
        }
    }

    private void showDate() {
        DatePickerDialog dialog = new DatePickerDialog(getActivity(), this,
                dateAndTime.get(Calendar.YEAR),
                dateAndTime.get(Calendar.MONTH),
                dateAndTime.get(Calendar.DAY_OF_MONTH));
        dialog.getDatePicker().setMaxDate(new Date().getTime());
        dialog.show();
    }

    private void showTime() {
        new TimePickerDialog(getActivity(), this,
                dateAndTime.get(Calendar.HOUR_OF_DAY),
                dateAndTime.get(Calendar.MINUTE), true)
                .show();
    }

    @Override
    public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
        dateAndTime.set(Calendar.YEAR, year);
        dateAndTime.set(Calendar.MONTH, month);
        dateAndTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);
        showTime();
    }

    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        dateAndTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
        dateAndTime.set(Calendar.MINUTE, minute);
        setInitialDateTime();
    }

    private void setInitialDateTime() {
        tvDate.setText(DateUtils.formatDateTime(getActivity(),
                dateAndTime.getTimeInMillis(),
                DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_YEAR
                        | DateUtils.FORMAT_SHOW_TIME));
    }

}
```

JournalInfoDialog.java

```
package com.smokebreak.ui.dialog;

import android.content.Context;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
```

```
import android.text.TextUtils;
import android.text.format.DateUtils;
import android.util.DisplayMetrics;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.Button;
import android.widget.TextView;

import androidx.fragment.app.DialogFragment;

import com.smokebreak.R;
import com.smokebreak.db.JournalData;

import io.realm.Realm;

public class JournalInfoDialog extends DialogFragment implements View.OnClickListener {

    private  View view;
    private Button bOk;
    private TextView tvDate, tvDesc;

    private JournalData data;

    public JournalInfoDialog(){}

    public static JournalInfoDialog newInstance(long id) {
        JournalInfoDialog f = new JournalInfoDialog();
        // Supply num input as an argument.
        Bundle args = new Bundle();
        args.putLong("id", id);
        f.setArguments(args);

        return f;
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        // Verify that the host activity implements the callback interface
        try {
            // Instantiate the EditNameDialogListener so we can send events to the host
//            listener = (OnDeleteListener) getGoalFragment();
        } catch (ClassCastException e) {
            // The activity doesn't implement the interface, throw exception
            throw new ClassCastException(context.toString()
                    + " must implement EditNameDialogListener");
        }
    }

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

         view = inflater.inflate(R.layout.journal_info_dialog, container);

        if (getDialog() != null && getDialog().getWindow() != null) {
            getDialog().getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
            getDialog().getWindow().requestFeature(Window.FEATURE_NO_TITLE);
        }
```

```
        initView(view);

        return view;
    }

    @Override
    public void onResume() {
        super.onResume();
        DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
        getDialog().getWindow().setLayout((int) (displayMetrics.widthPixels*0.8),
                view.getLayoutParams().WRAP_CONTENT);
    }

    private void initView(View view){
        bOk = view.findViewById(R.id.b_ok);
        bOk.setOnClickListener(this);


        tvDate = view.findViewById(R.id.tv_date);
        tvDesc = view.findViewById(R.id.tv_desc);

        setData();
    }

    private void setData(){
         data = Realm.getDefaultInstance().where(JournalData.class).
                equalTo("date", getArguments().getLong("id")).findFirst();

        if (data != null) {
            tvDate.setText(DateUtils.formatDateTime(getActivity(),
                    data.getDate(),
                    DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_YEAR
                            | DateUtils.FORMAT_SHOW_TIME));
            if (TextUtils.isEmpty(data.getDescription()))
                tvDesc.setVisibility(View.GONE);

            tvDesc.setText(data.getDescription());
        }
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.b_ok:
                dismiss();
                break;
        }
    }

}
```

InfoMainFragment.java

```
package com.smokebreak.ui.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ProgressBar;
```

```
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.smokebreak.App;
import com.smokebreak.R;
import com.smokebreak.api.SmokeApi;
import com.smokebreak.db.HealthData;
import com.smokebreak.db.InfoData;
import com.smokebreak.db.InfoDetailData;
import com.smokebreak.ui.activity.InfoDetailActivity;
import com.smokebreak.ui.adapters.HealthAdapter;
import com.smokebreak.ui.adapters.InfoMainAdapter;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import io.realm.Realm;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class InfoMainFragment extends Fragment implements View.OnClickListener,
InfoMainAdapter.OnInfoMainClickListener {

    private Realm realm;

    private FloatingActionButton fab;

    private RecyclerView recyclerView;
    private InfoMainAdapter infoMainAdapter;
    private ProgressBar progressBar;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ViewGroup rootView = (ViewGroup) inflater.inflate(
                R.layout.fragment_info_main, container, false);

        initView(rootView);

        loadJSONFromAsset();
        showAdapter();

        return rootView;
    }

    private void initView(View rootView) {
        realm = Realm.getDefaultInstance();

        recyclerView = rootView.findViewById(R.id.rv);
        recyclerView.setHasFixedSize(true);
        recyclerView.setVisibility(View.INVISIBLE);
        LinearLayoutManager llm = new LinearLayoutManager(getActivity());
        recyclerView.setLayoutManager(llm);

        fab = rootView.findViewById(R.id.fab);
```

```
        fab.setOnClickListener(this);

        progressBar = rootView.findViewById(R.id.progressBar);

    }

    private void loadJSONFromAsset() {
        try {
            InputStream is1 = getActivity().getAssets().open("info_all.json");
            realm.beginTransaction();
            realm.createOrUpdateAllFromJson(InfoData.class, is1);
            realm.commitTransaction();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    private void showAdapter() {
        ArrayList<InfoData> dataList = new ArrayList<>();
        dataList.addAll(realm.where(InfoData.class).findAll());
        infoMainAdapter = new InfoMainAdapter(dataList, this);
        recyclerView.setAdapter(infoMainAdapter);
        recyclerView.setVisibility(View.VISIBLE);
        progressBar.setVisibility(View.GONE);
    }

    @Override
    public void onClick(View v) {
        getInfoData();
    }

    @Override
    public void onInfoClick(int id) {
        Intent toDetail = new Intent(getActivity(), InfoDetailActivity.class);
        toDetail.putExtra("id", id);
        getActivity().startActivity(toDetail);
    }

    private void getInfoData() {
        recyclerView.setVisibility(View.INVISIBLE);
        progressBar.setVisibility(View.VISIBLE);
        App.getApp().getRetrofit().create(SmokeApi.class).getInfos().enqueue(new Callback<List<InfoData>>() {
            @Override
            public void onResponse(Call<List<InfoData>> call, Response<List<InfoData>> response) {
                infoMainAdapter = new InfoMainAdapter(response.body(), InfoMainFragment.this);
                realm.beginTransaction();
                realm.insertOrUpdate(response.body());
                realm.commitTransaction();
                recyclerView.setAdapter(infoMainAdapter);
                recyclerView.setVisibility(View.VISIBLE);
                progressBar.setVisibility(View.GONE);
            }

            @Override
            public void onFailure(Call<List<InfoData>> call, Throwable t) {

            }
        });
    }
}
```

InfoDetailActivity.java

```
package com.smokebreak.ui.activity;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

import com.smokebreak.R;
import com.smokebreak.db.InfoData;
import com.smokebreak.db.InfoDetailData;
import com.smokebreak.ui.adapters.InfoAdapter;
import com.smokebreak.ui.dialog.InfoDetailDialog;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;

import io.realm.Realm;

public class InfoDetailActivity extends AppCompatActivity implements
        InfoAdapter.OnInfoClickListener, View.OnClickListener {

    private RecyclerView recyclerView;
    private InfoAdapter infoAdapter;

    private Realm realm = Realm.getDefaultInstance();
    ;

    private ImageView ivBack;
    private Button bBack;

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

        initView();
        showAdapter();
    }

    private void initView() {
        recyclerView = findViewById(R.id.rv);
        recyclerView.setHasFixedSize(true);
        LinearLayoutManager llm = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(llm);

        ivBack = findViewById(R.id.iv_back);
        ivBack.setOnClickListener(this);
        bBack = findViewById(R.id.b_back);
        bBack.setOnClickListener(this);


        showAdapter();
```

```
    }

    private void showAdapter() {
        ArrayList<InfoDetailData> dataList = new ArrayList<>();
        dataList.addAll(realm.where(InfoData.class).equalTo("id", getIntent().getIntExtra("id", 0))
                .findFirst().getInfoDetailList());
        infoAdapter = new InfoAdapter(dataList, this, checkType());
        recyclerView.setAdapter(infoAdapter);
        recyclerView.setVisibility(View.VISIBLE);
    }

    private boolean checkType() {
        if (getIntent().getIntExtra("id", 0) == 1)
            return true;
        return false;
    }

    @Override
    public void onInfoClick(String title, String desc) {
        FragmentManager fm = getSupportFragmentManager();
        InfoDetailDialog newFragment = InfoDetailDialog.newInstance(title, desc);
        newFragment.show(fm, "dialog");
    }

    @Override
    public void onClick(View v) {
        onBackPressed();
        finish();
    }
}
```

InfoDetailDialog.java

```
package com.smokebreak.ui.dialog;

import android.content.Context;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.TextUtils;
import android.text.format.DateUtils;
import android.text.method.ScrollingMovementMethod;
import android.util.DisplayMetrics;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.Button;
import android.widget.TextView;

import androidx.fragment.app.DialogFragment;

import com.smokebreak.R;
import com.smokebreak.db.JournalData;

import io.realm.Realm;

public class InfoDetailDialog extends DialogFragment implements View.OnClickListener {

    private  View view;
    private Button bOk;
    private TextView tvTitle, tvDesc;
```

```
public InfoDetailDialog(){}

public static InfoDetailDialog newInstance(String title, String desc) {
   InfoDetailDialog f = new InfoDetailDialog();
   // Supply num input as an argument.
   Bundle args = new Bundle();
   args.putString("title", title);
   args.putString("desc", desc);
   f.setArguments(args);

   return f;
}


public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    view = inflater.inflate(R.layout.info_detail_dialog, container);

   if (getDialog() != null && getDialog().getWindow() != null) {
      getDialog().getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
      getDialog().getWindow().requestFeature(Window.FEATURE_NO_TITLE);
   }

   initView(view);

   return view;
}

@Override
public void onResume() {
   super.onResume();
   DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
   getDialog().getWindow().setLayout((int) (displayMetrics.widthPixels*0.8),
         view.getLayoutParams().WRAP_CONTENT);
}

private void initView(View view){
   bOk = view.findViewById(R.id.b_ok);
   bOk.setOnClickListener(this);


   tvTitle = view.findViewById(R.id.tv_title);
   tvDesc = view.findViewById(R.id.tv_desc);
   tvDesc.setMovementMethod(new ScrollingMovementMethod());

   setData();
}

private void setData(){

  String title =  getArguments().getString("title");
  String desc =  getArguments().getString("desc");

  if (!TextUtils.isEmpty(title))
     tvTitle.setText(title);

   if (!TextUtils.isEmpty(desc))
      tvDesc.setText(desc);
}
```

```
    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.b_ok:
                dismiss();
                break;
        }
    }

}
```

TestActivity.java

```
package com.smokebreak.ui.activity;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.smokebreak.R;
import com.smokebreak.db.HealthData;
import com.smokebreak.db.InfoData;
import com.smokebreak.db.TestData;
import com.smokebreak.ui.fragments.TestDetailFragment;

import java.io.IOException;
import java.io.InputStream;

import io.realm.Realm;

public class TestActivity extends AppCompatActivity implements
View.OnClickListener,TestDetailFragment.OnNextClick {

    private ImageView ivBack;
    private TextView tvTitle, tvBottom;

    private Realm realm = Realm.getDefaultInstance();
    private int totalPoint = 0;
    private int questionId = 1;
    private int questionSize = 0;

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

        initView();
    }

    private void initView(){
        ivBack = findViewById(R.id.iv_back);
        ivBack.setOnClickListener(this);
```

```
        tvTitle = findViewById(R.id.tv_title_test);
        tvBottom = findViewById(R.id.tv_bottom);
        setData();
        showQuestion();
    }

//    private void loadJSONFromAsset() {
//        try {
//            InputStream is1 = getAssets().open("test_1.json");
//            InputStream is2 = getAssets().open("test_2.json");
//            InputStream is3 = getAssets().open("test_3.json");
//            realm.beginTransaction();
//            realm.where(TestData.class).findAll().deleteAllFromRealm();
//            realm.createOrUpdateObjectFromJson(TestData.class,is1);
//            realm.createOrUpdateObjectFromJson(TestData.class,is2);
//            realm.createOrUpdateObjectFromJson(TestData.class,is3);
//
//            realm.commitTransaction();
//            realm.where(TestData.class).findAll();
//        } catch (IOException ex) {
//            ex.printStackTrace();
//        }
//    }

    private void showQuestion(){
        FragmentTransaction fragmentTransaction = getSupportFragmentManager()
                .beginTransaction();
        // добавляем в контейнер при помощи метода add()
        fragmentTransaction.replace(R.id.frame,
                TestDetailFragment.newInstance(getIntent().getIntExtra("id",0),questionId,totalPoint), "");
        fragmentTransaction.commit();

    }

    private void setData(){
        TestData testData = realm.where(TestData.class).equalTo("id",getIntent().getIntExtra("id",0)).findFirst();

        tvTitle.setText(testData.getName());
        questionSize = testData.getQuestionList().size();
        tvBottom.setText(questionId+"/"+questionSize);

    }

    @Override
    public void onNext(int point,boolean isEnd) {
        if (questionId < questionSize){
            questionId++;
            totalPoint += point;
            tvBottom.setText(questionId+"/"+questionSize);
        }else {
            questionId++;
            totalPoint += point;
            tvBottom.setText("Резултати");
        }

        if (!isEnd)
            showQuestion();
        else
            finish();
```

```
    }

    @Override
    public void onClick(View v) {
        finish();
    }
}
```

TestFragment.java

```
package com.smokebreak.ui.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ProgressBar;

import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.smokebreak.App;
import com.smokebreak.R;
import com.smokebreak.api.SmokeApi;
import com.smokebreak.db.InfoData;
import com.smokebreak.db.TestData;
import com.smokebreak.ui.activity.InfoDetailActivity;
import com.smokebreak.ui.activity.TestActivity;
import com.smokebreak.ui.adapters.InfoMainAdapter;
import com.smokebreak.ui.adapters.TestMainAdapter;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import org.json.JSONArray;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import io.realm.Realm;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class TestFragment extends Fragment implements View.OnClickListener,
TestMainAdapter.OnTestMainClickListener {

    private Realm realm;

    private FloatingActionButton fab;

    private RecyclerView recyclerView;
    private TestMainAdapter testMainAdapter;
    private ProgressBar progressBar;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ViewGroup rootView = (ViewGroup) inflater.inflate(
```

```
            R.layout.fragment_test, container, false);

    initView(rootView);
    loadJSONFromAsset();

    showAdapter();

    return rootView;
}

private void initView(View rootView){
    realm = Realm.getDefaultInstance();

    recyclerView = rootView.findViewById(R.id.rv);
    recyclerView.setHasFixedSize(true);
    recyclerView.setVisibility(View.INVISIBLE);
    LinearLayoutManager llm = new LinearLayoutManager(getActivity());
    recyclerView.setLayoutManager(llm);

    fab = rootView.findViewById(R.id.fab);
    fab.setOnClickListener(this);

    progressBar = rootView.findViewById(R.id.progressBar);
}

@Override
public void onClick(View v) {
    getTestData();
}

private void showAdapter() {
    ArrayList<TestData> dataList = new ArrayList<>();
    dataList.addAll(realm.where(TestData.class).findAll());
    testMainAdapter = new TestMainAdapter(dataList,  this);
    recyclerView.setAdapter(testMainAdapter);
    recyclerView.setVisibility(View.VISIBLE);
    progressBar.setVisibility(View.GONE);
}

private void loadJSONFromAsset() {
    try {
        InputStream is1 = getActivity().getAssets().open("test_all.json");
        realm.beginTransaction();
        realm.createOrUpdateAllFromJson(TestData.class, is1);
        realm.commitTransaction();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

@Override
public void onInfoClick(int id) {
    Intent toTest = new Intent(getActivity(), TestActivity.class);
    toTest.putExtra("id",id);
    getActivity().startActivity(toTest);
}

private void getTestData() {
    recyclerView.setVisibility(View.INVISIBLE);
    progressBar.setVisibility(View.VISIBLE);
    App.getApp().getRetrofit().create(SmokeApi.class).getTests().enqueue(new Callback<List<TestData>>() {
```

```
        @Override
        public void onResponse(Call<List<TestData>> call, Response<List<TestData>> response) {
            testMainAdapter = new TestMainAdapter(response.body(), TestFragment.this);
            realm.beginTransaction();
            realm.insertOrUpdate(response.body());
            realm.commitTransaction();
            recyclerView.setAdapter(testMainAdapter);
            recyclerView.setVisibility(View.VISIBLE);
            progressBar.setVisibility(View.GONE);
        }

        @Override
        public void onFailure(Call<List<TestData>> call, Throwable t) {

        }
      });
    }
}
```

TestDetailFragment.java

```
package com.smokebreak.ui.fragments;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.smokebreak.App;
import com.smokebreak.R;
import com.smokebreak.db.AnswerData;
import com.smokebreak.db.QuestionData;
import com.smokebreak.db.ResultData;
import com.smokebreak.db.TestData;
import com.smokebreak.ui.dialog.AddJournalDialog;
import com.smokebreak.ui.dialog.InfoDetailDialog;
import com.smokebreak.ui.view.AnswersView;

import org.jetbrains.annotations.NotNull;

import java.util.List;

import io.realm.Realm;

public class TestDetailFragment extends Fragment implements View.OnClickListener {

    private Realm realm = Realm.getDefaultInstance();
    private TextView tvTitle, tvQuestion;
    private AnswersView answersView;
    private Button bNext;

    private int totalPoint = 0;

    private OnNextClick onNextClick;
```

```
private boolean isEnd = false;

public interface OnNextClick{
   void onNext(int point, boolean isEnd);
}

public static TestDetailFragment newInstance(int testId, int questionId, int totalPoint) {
   TestDetailFragment f = new TestDetailFragment();
   // Supply num input as an argument.
   Bundle args = new Bundle();
   args.putInt("testId", testId);
   args.putInt("questionId", questionId);
   args.putInt("totalPoint", totalPoint);
   f.setArguments(args);

   return f;
}

@Override
public void onAttach(@NonNull @NotNull Context context) {
   super.onAttach(context);
   try {
      // Instantiate the EditNameDialogListener so we can send events to the host
      onNextClick = (OnNextClick) getActivity();
   } catch (ClassCastException e) {
      // The activity doesn't implement the interface, throw exception
      throw new ClassCastException(context.toString()
            + " must implement EditNameDialogListener");
   }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
   ViewGroup rootView = (ViewGroup) inflater.inflate(
         R.layout.fragment_test_detail, container, false);

   initView(rootView);

   return rootView;
}

private void initView(View rootView) {
   tvTitle = rootView.findViewById(R.id.tv_title);
   tvQuestion = rootView.findViewById(R.id.tv_question);
   answersView = rootView.findViewById(R.id.answer_view);
   bNext = rootView.findViewById(R.id.b_next);
   bNext.setOnClickListener(this);

   totalPoint = getArguments().getInt("totalPoint");

   if (getArguments().getInt("questionId") > realm.where(TestData.class).
         equalTo("id", getArguments().getInt("testId")).findFirst().getQuestionList().size()){
      showResult();
   }else
      setData();

}

private void setData() {
   TestData testData = realm.where(TestData.class).equalTo("id", getArguments().getInt("testId")).findFirst();
```

```
        for (QuestionData questionData : testData.getQuestionList()){
            if (questionData.getId() == getArguments().getInt("questionId")) {
                questionData.getAnswerData();
                tvTitle.setText("Питання №" + questionData.getId() + ":");
                tvQuestion.setText(questionData.getQuestion());
                addAnswer(questionData.getAnswerData());
            }
        }
    }

    private void showResult(){
        TestData testData = realm.where(TestData.class).equalTo("id", getArguments().getInt("testId")).findFirst();
        ResultData userResult = null;
        int max = 0;
        for (ResultData data: testData.getResultList()) {
            if(data.getMax() > max)
                max = data.getMax();
            if (data.getMin() <= totalPoint  && totalPoint <= data.getMax())
                userResult = data;
        }

        tvTitle.setText("Ви набрали "+totalPoint+" балів з "+max);
        tvQuestion.setText("Ваш результат :\n" + userResult.getResult());
        bNext.setText("Закрити");
        isEnd = true;
    }

    private void addAnswer(List<AnswerData> answerDataList){
        answersView.setData(answerDataList);
    }

    @Override
    public void onClick(View v) {
        if(answersView.getPoint() == -1 && !isEnd){
            showError();
            return;
        }
        if (onNextClick != null)
            onNextClick.onNext(answersView.getPoint(),isEnd);
    }

    private void showError(){
        App.getApp().showToastError("Оберіть хоча б одну відповідь",getActivity());
    }

}
```

SmokeApi.java

```
package com.smokebreak.api;

import com.smokebreak.db.HealthData;
import com.smokebreak.db.InfoData;
import com.smokebreak.db.TestData;

import java.util.List;

import retrofit2.Call;
import retrofit2.http.GET;

public interface SmokeApi {
```

```
    @GET("/health-achievements")
    Call<List<HealthData>> getHealths();

    @GET("/infos")
    Call<List<InfoData>> getInfos();

    @GET("/tests")
    Call<List<TestData>> getTests();
}
```

DailyReceiver.java

```
package com.smokebreak.notifycation;

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.util.Log;

import androidx.core.app.NotificationCompat;

import com.smokebreak.R;

public class DailyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        Log.d("DailyReceiver","DailyReceiver");

        NotificationManager mNotificationManager;

        NotificationCompat.Builder mBuilder =
                new NotificationCompat.Builder(context.getApplicationContext(), "notify_001");


        NotificationCompat.BigTextStyle bigText = new NotificationCompat.BigTextStyle();
        bigText.setBigContentTitle(intent.getStringExtra("title"));
//        bigText.setBigContentTitle("Today's Bible Verse");
        bigText.setSummaryText("Ви молодець!");

//        mBuilder.setContentIntent(pendingIntent);
        mBuilder.setSmallIcon(intent.getIntExtra("icon",R.drawable.ic_health_true));
        mBuilder.setAutoCancel(true);
//        mBuilder.setContentTitle("Your Title");
//        mBuilder.setContentText("Your text");
        mBuilder.setPriority(Notification.PRIORITY_MAX);
        mBuilder.setStyle(bigText);

        mNotificationManager =
                (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
        {
            String channelId = "Your_channel_id";
            NotificationChannel channel = new NotificationChannel(
                    channelId,
```

```
                "Channel human readable title",
                NotificationManager.IMPORTANCE_HIGH);
            mNotificationManager.createNotificationChannel(channel);
            mBuilder.setChannelId(channelId);
        }

        mNotificationManager.notify(0, mBuilder.build());
    }

}
```

ДОДАТОК Б. КЕРІВНИЦТВО КОРИСТУВАЧА

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

«SmokeBreak» - це мобільне застосування для індивідуальної боротьби з тютюнопалінням. Він робить процес відмови користувача від вживання тютюнових засобів більш ефективним. Його робота заснована на щоденному відслідковуванню прогресу боротьби з палінням, веденню списку цілей та щоденнику паління, перегляду довідкових матеріалів, тестуванню залежності.

Користувач має можливість заповнити дані про свої тютюнові звички до профілю курця, переглянути показники прогресу відмови від паління, статистику та досягнення для здоров’я, створити цілі за обраною категорією, скинути лічильник відмови та додати зрив з описом до щоденника паління, переглянути дописи довідки за обраною категорією, обрати тестування, відповісти на запитання та переглянути результат.

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні вимоги

Для коректної роботи даного мобільного застосування необхідне дотримання наступних умов :

- наявність мобільного пристрою з операційною системою на базі платформи Android з версією ядра 4.1 або вище.
- наявність вільного дискового простору на мобільному пристрої об’ємом від 300 мегабайт.

2.2 Встановлення застосування

Встановлення програмного додатку відбувається з файлу збірки APK. Для цього необхідно створити файл збірки, завантажити його на мобільний пристрій та встановити додаток. Більшість операційних систем на базі платформи Android мають інтегрований інсталятор застосувань з APK-файлів.

2.3 Встановлення застосування

Після встановлення застосування на робочому столі мобільного пристрою з’явиться значок-іконка мобільного застосування «SmokeBreak». Після натиску на іконку відбувається запуск програмного додатку та починається робота з застосуванням.

3 РОБОТА З ЗАСТОСУНКОМ

3.1 Стартовий екран та профіль курця

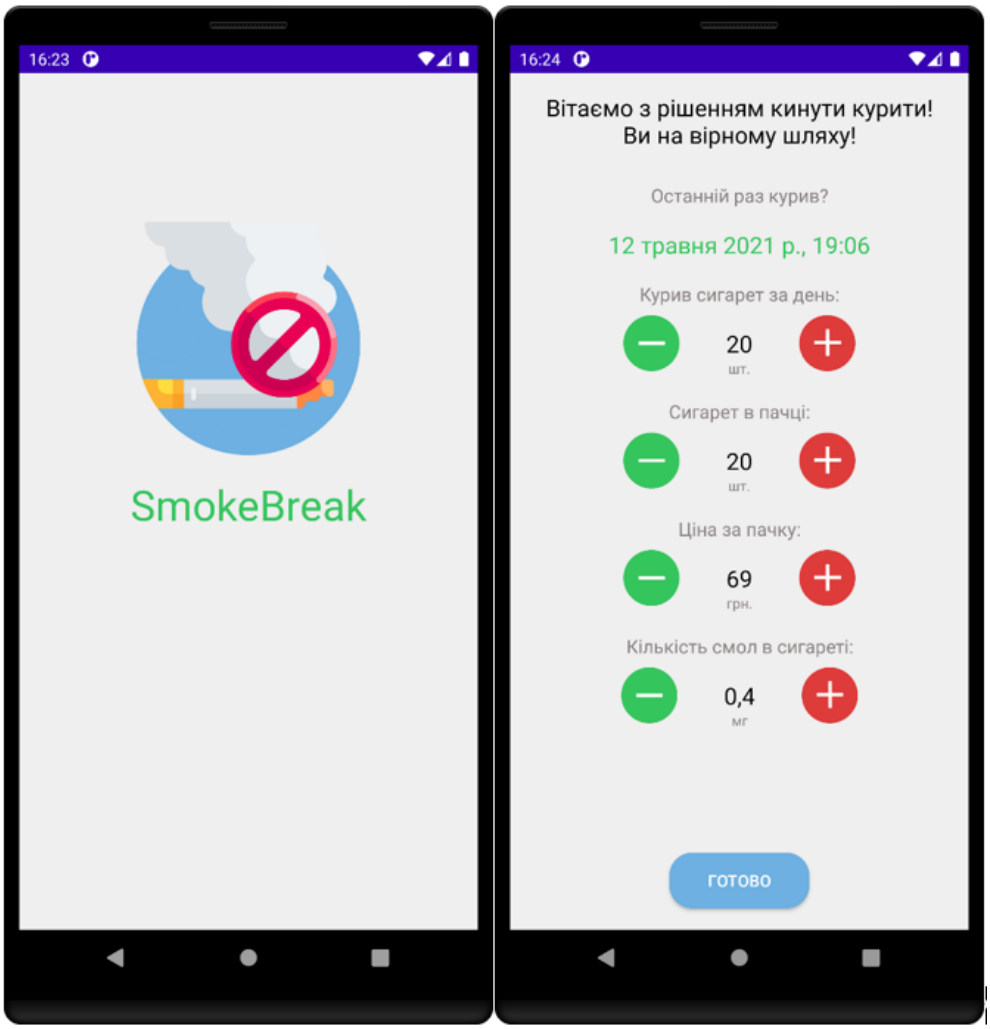

Рисунок 3.1 – Стартовий екран та екран профілю курця

Після початку роботи застосування користувачу демонструється стартовий екран з лого застосунку. Після цього демонструється екран профілю курця, де користувач може вказати показники своєї залежності натиском на клавіші «-» та «+» та обрати дату останньої випаленої цигарки. Після натиску на кнопку «Готово» користувач переходить до головної сторінки.

3.2 Головна сторінка

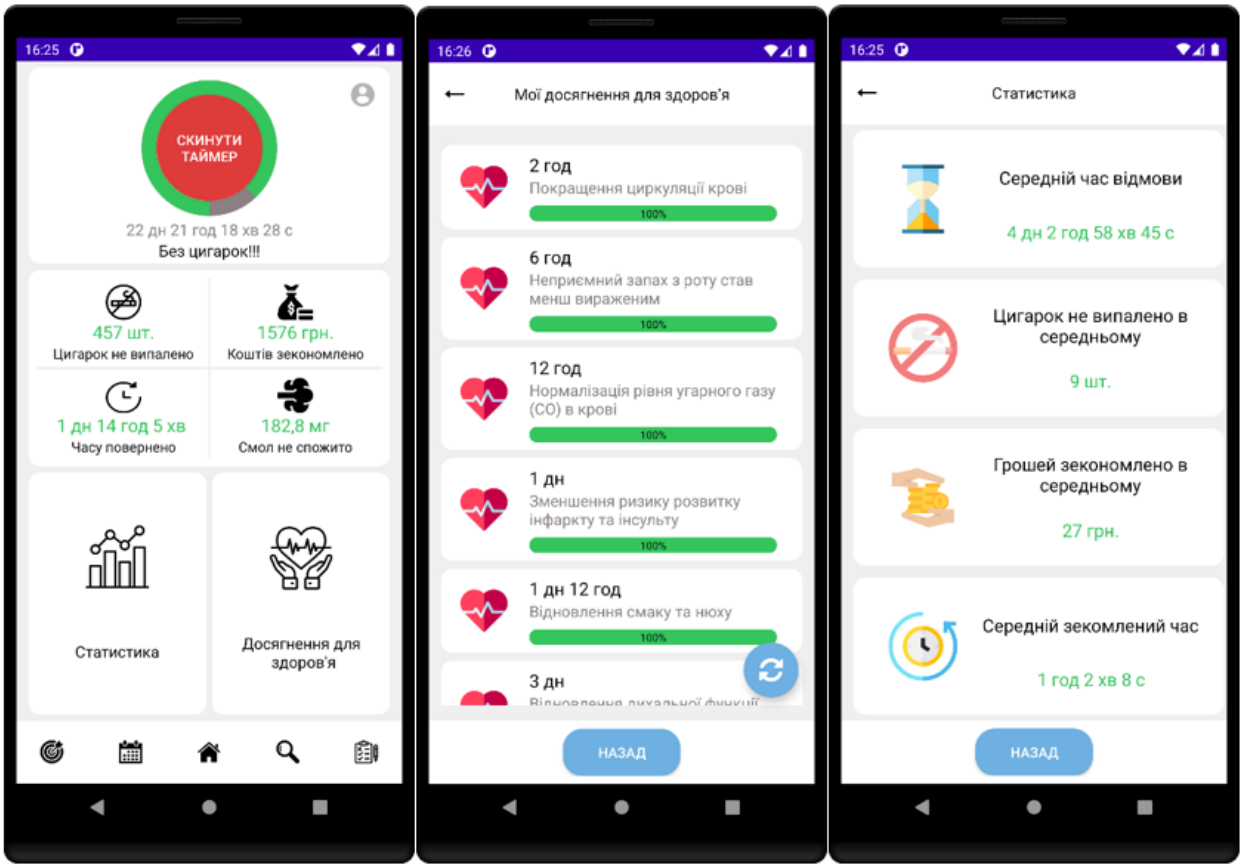

Рисунок 3.3 – Головний екран, екран досягнень для здоров’я та екран статистики

На головному екрані користувач може переглянути власні показники відмови від паління : час відмови, кількість невипалених цигарок, об’єм зекономлених коштів та часу, об’єм неспожитих смол. Користувач може скинути лічильник, натиснувши на кнопку «Скинути таймер», що викликає діалогове вікно додавання зриву. З головного екрану користувач може перейти до профілю курця натиснувши на іконку користувача, до списку цілей, щоденнику паління, довідки та тестування у нижній панелі, а також до екранів досягнення для здоров’я та статистики у відповідних панелях на головному екрані.

3.3 Список цілей

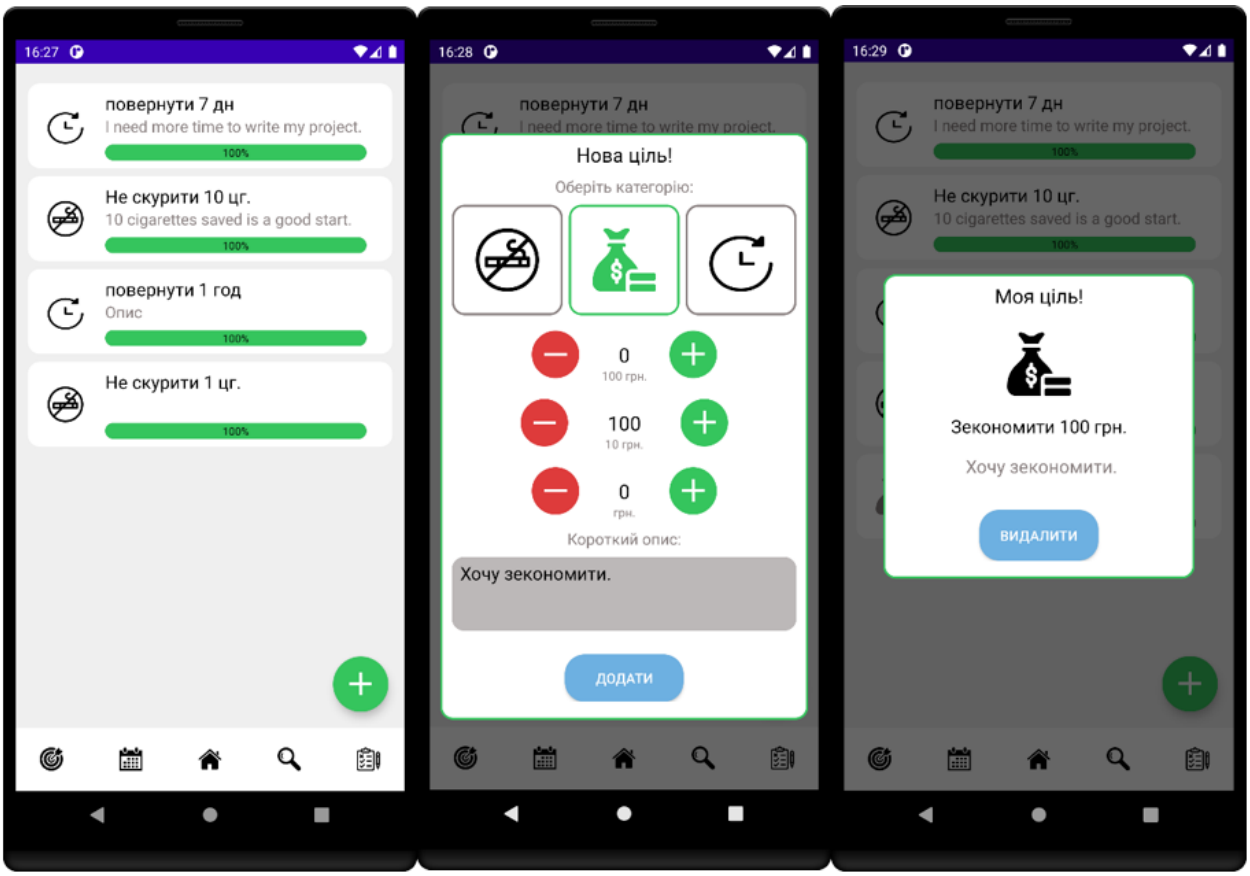

Рисунок 3.3 – Екран списку цілей та діалогові вікна створення цілі та перегляду цілі

На екрані списку цілей користувач може переглянути прогрес виконання створених цілей та створити нову ціль натиском на кнопку «+». Це викликає діалогове вікно створення нової цілі, де користувач може обрати категорію цілі, вказати цільове значення натисканням на кнопки «-» та «+», а також додати опис цілі в текстовому полі. Після натиску на кнопку «Додати» ціль буде створена, додана до списку цілей та почнеться відслідковування прогресу її виконання. Користувач може обрати ціль зі списку натисканням на неї, що викликає діалогове вікно перегляду цілі, що містить кнопку «Видалити», натиск на яку видалить ціль зі списку цілей.

3.4 Щоденник паління

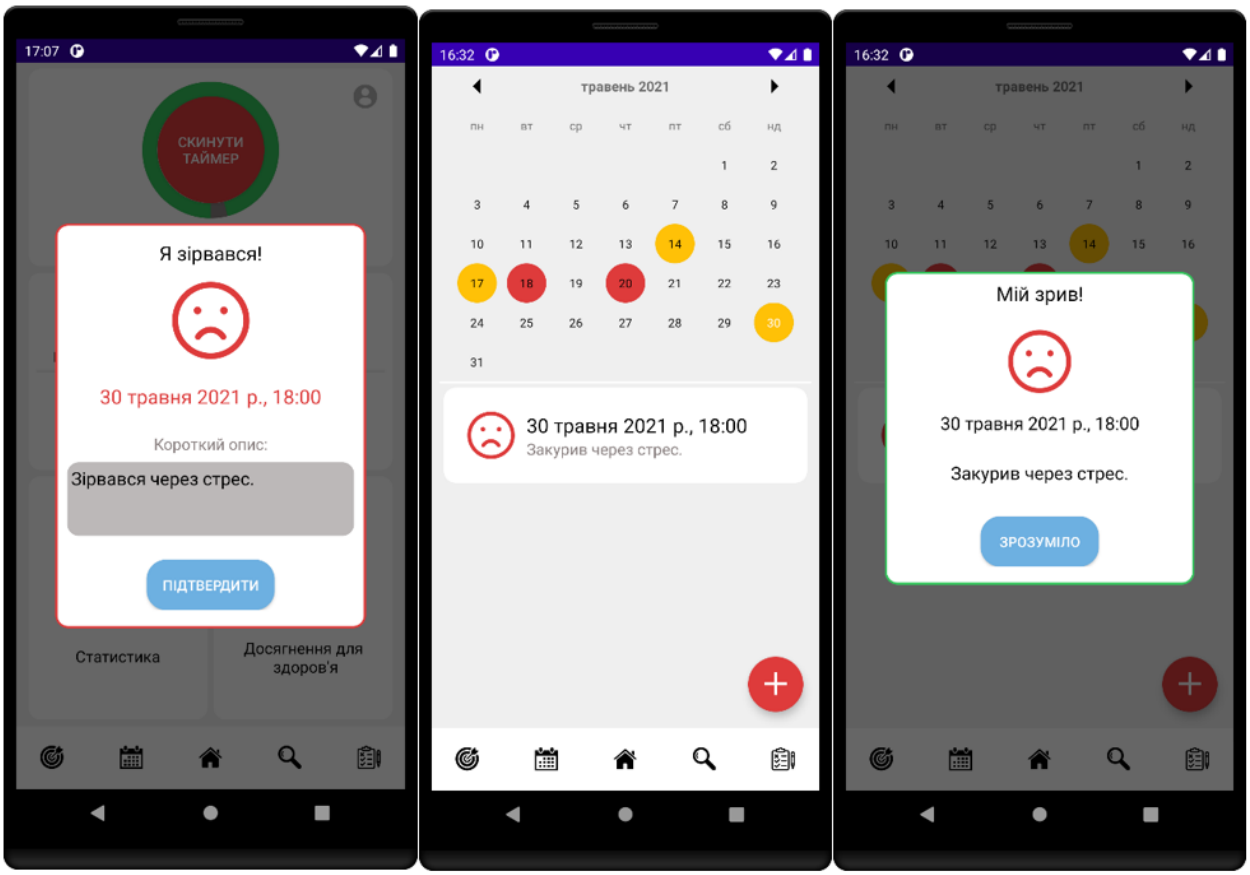

Рисунок 3.4 – Діалогове вікно додання зриву, екран щоденнику паління та діалогове вікно перегляду зриву.

Діалогове вікно додання змісту містить дату зриву, яку користувач може змінити в календарі та циферблаті натиском на неї, а також вказати причину зриву в текстовому полі. Натисканням на кнопку «Підтвердити» ціль буде створено та додано до щоденнику паління.

На екрані щоденника паління користувач може переглянути зриви за датою, обраною в календарі. В календарі дні, за які були зриви, позначаються жовтим, якщо кількість зривів за днів було від 0 до 2 включно, червоним – більше 2. Натиском на зрив в списку зривів користувач викликає діалогове вікно перегляду зриву.

3.5 Модуль довідки

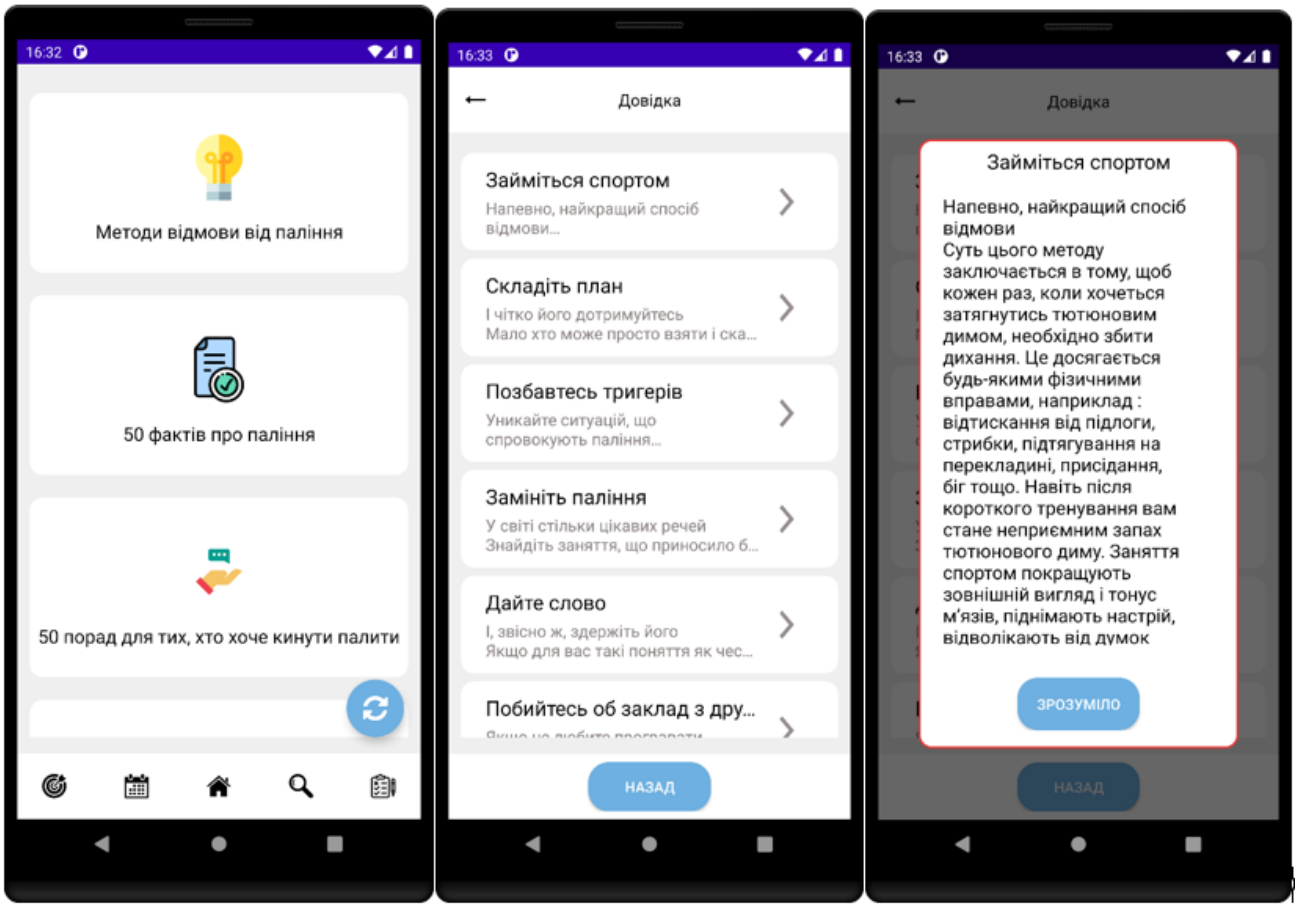

Рисунок 3.5 – Екран категорій довідки, екран списку дописів довідки, модальне вікно перегляду допису довідки

На екрані довідки користувач може переглянути список категорій довідки та обрати категорію, після чого користувач переходить до списку дописів довідки, де відображається назва допису та частина опису.

Натиском на кнопку «Назад» користувач повертається до вікна списку категорій довідки.

Обравши допис зі списку, користувач викликає модальне вікно перегляду допису, що містить повний текст допису, який користувач може пролистувати.

Натиском на кнопку «Зрозуміло» користувач закриває модальне вікно.

3.6 Модуль тестування

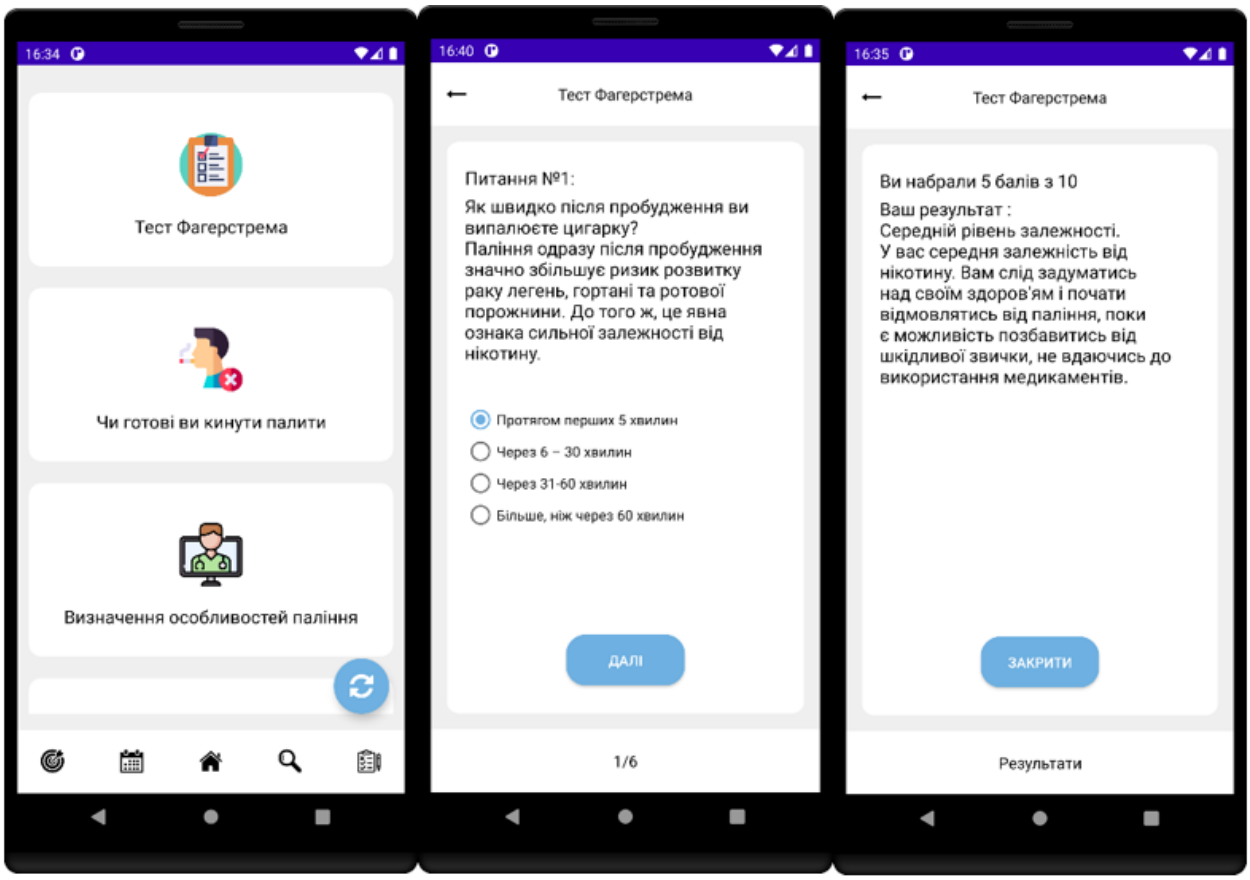

Рисунок 3.6 – Екран списку тестувань, екран проходження тестування, екран результатів тестування

На екрані тестування користувач може переглянути список тестувань для проходження та обрати тестування з переліку, що викликає перехід до екрану проходження тестування.

На екрані проходження тестування користувач може переглянути текст запитання, обрати відповідь з переліку варіантів відповіді, натиском на кнопку «Далі» перейти до наступного запитання, або до вікна результатів, якщо запитання закінчились.

На екрані результатів користувач може переглянути свій результат проходження тесту, що базується на обраних ним варіантах відповіді.

ДОДАТОК В. ГРАФІЧНІ МАТЕРІАЛИ

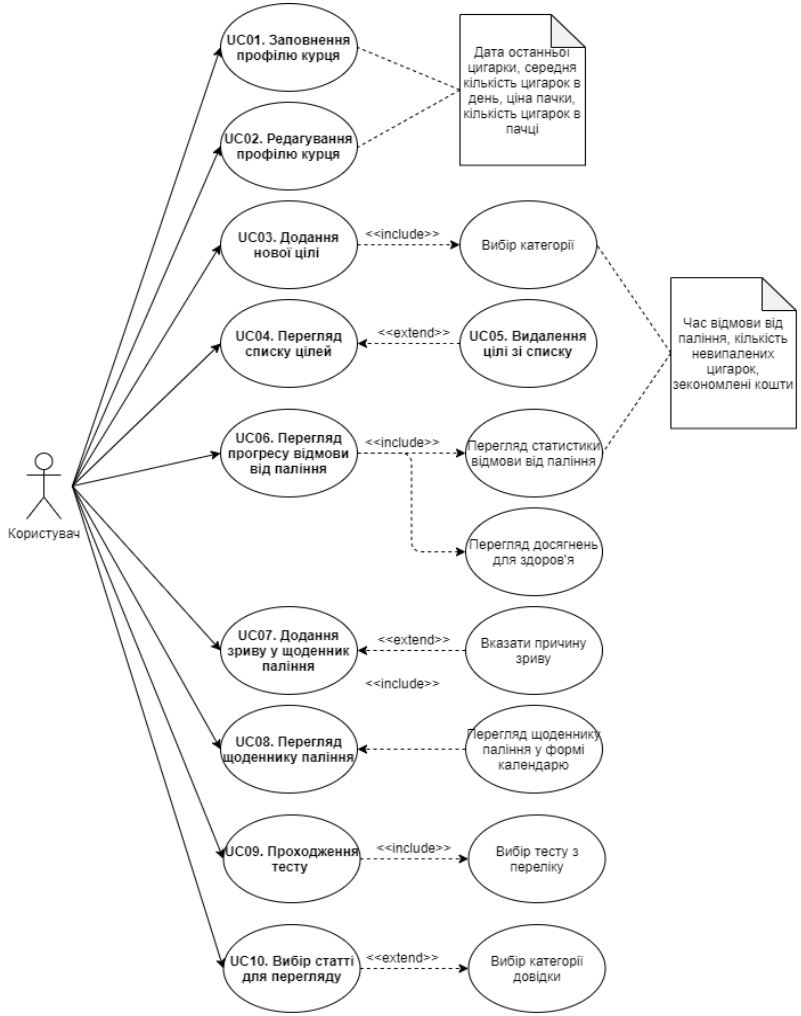

Рисунок 1 - Схема структурна варіантів використання

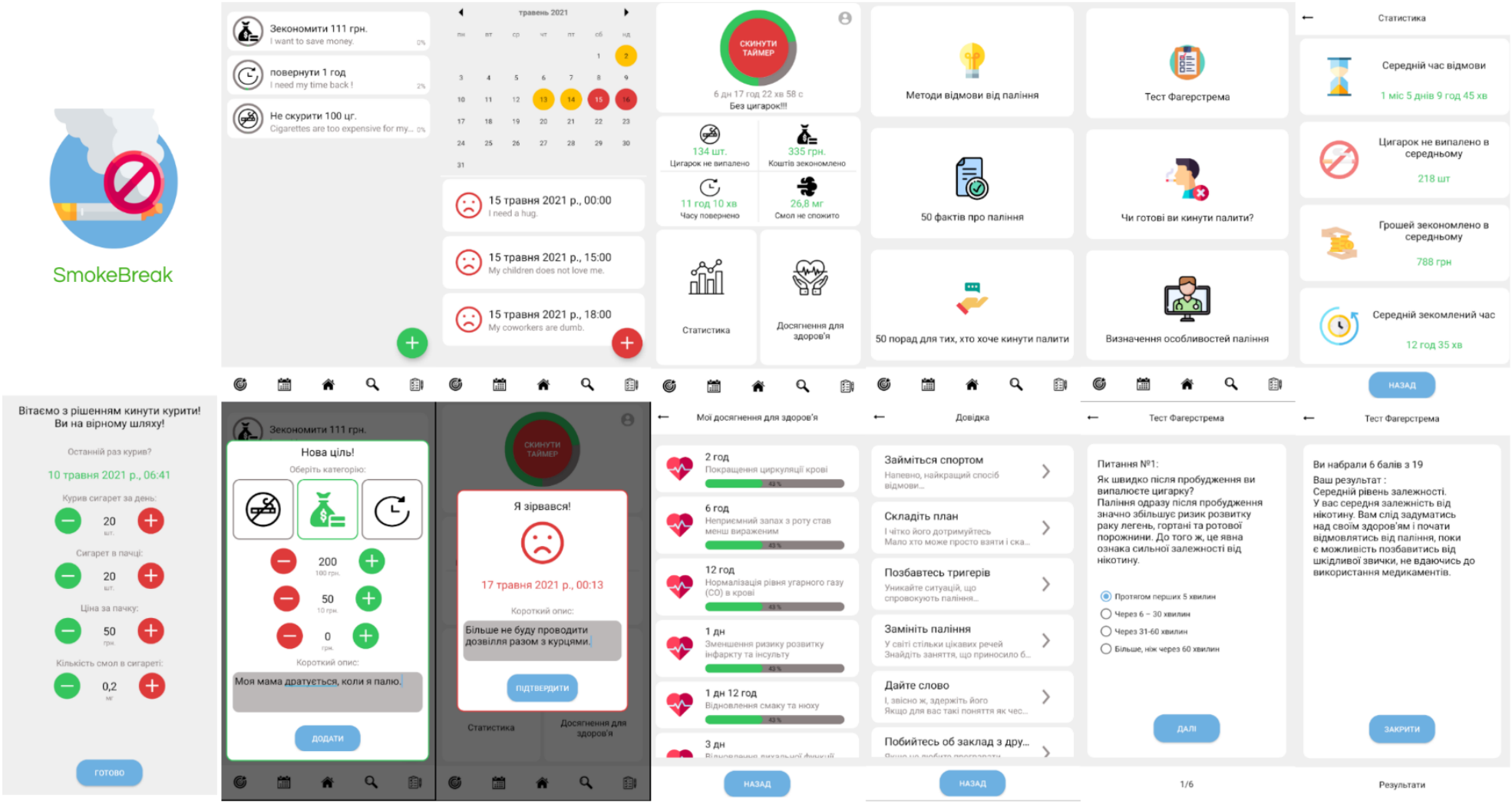

Рисунок 2 – Креслення вигляду екранних форм

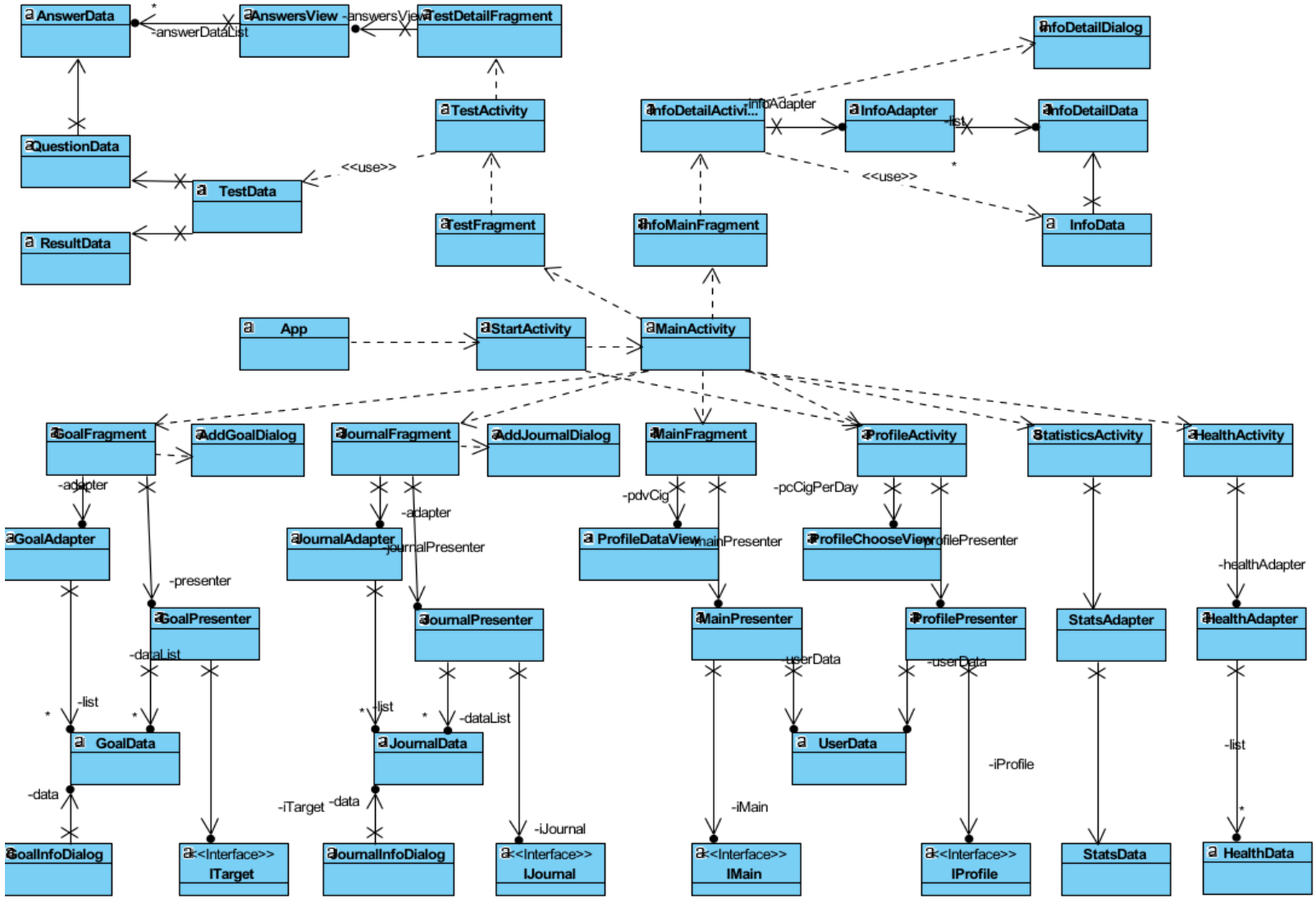
AnswerData
AnswersView
TestDetailFragment
InfoDetailDialog
TestActivity
InfoDetailActivi...
InfoAdapter
InfoDetailData
QuestionData
TestData
ResultData
TestFragment
InfoMainFragment
InfoData
App
StartActivity
MainActivity
GoalFragment
AddGoalDialog
JournalFragment
AddJournalDialog
MainFragment
ProfileActivity
StatisticsActivity
HealthActivity
GoalAdapter
JournalAdapter
ProfileDataView
ProfileChooseView
GoalPresenter
JournalPresenter
MainPresenter
ProfilePresenter
StatsAdapter
HealthAdapter
GoalData
JournalData
UserData
GoalInfoDialog
ITarget
JournalInfoDialog
IJournal
IMain
IProfile
StatsData
HealthData
<<use>>
-answerDataList
-answersView
-infoAdapter
-list
-adapter
-presenter
-dataList
-journalPresenter
-pdvCig
-pcCigPerDay
-mainPresenter
-profilePresenter
-healthAdapter
-userData
-data
-iTarget
-iJournal
-iMain
-iProfile

Рисунок 3 – Схема структурна класів програмного забезпечення