

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

БАКАЛАВРСЬКА РОБОТА

на тему:

**«МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ОБМІНУ РЕЧАМИ,
ЩО НЕ ВИКОРИСТОВУЮТЬСЯ»**

Виконав: студент 4 курсу

Рівень бакалавр

Галузь знань 12 «Інформаційні технології»,
спеціальність 122 «Комп'ютерні науки»

Дацько Кирил Сергійович

Керівник: к.т.н., доцент

Манаков Сергій Юрійович

Рецензент: к.пед.н., доцент

Чикунів Павло Олександрович

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет **кібербезпеки та інформаційних технологій**
Кафедра **інформаційних технологій**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **122 Комп'ютерні науки**
Назва освітньої програми: **Комп'ютерні науки**

ЗАТВЕРДЖУЮ

Завідувач кафедри інформаційних технологій

доцент Н.І. Логінова

_____ « ____ » _____ 20__ року

З А В Д А Н Н Я

на кваліфікаційну (бакалаврську) роботу

здобувачу **Дацько Кирилу Сергійовичу**

1. Тема роботи: «Веб-сервіс контролю успішності учнів»

Керівник роботи: к.т.н., доцент Манаков Сергій Юрійович

затверджені наказом Ректора НУ «ОЮА» від «02» квітня 2024 року № 809-06.

2. Строк подання здобувачем роботи «20» травня 2024 року

3. Дата видачі завдання «15» вересня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка задачі	01.12.23	
2	Опис розроблених алгоритмів та програмного забезпечення	26.01.24	
3	Результати експериментів та аналіз отриманих даних	15.03.24	
4	Оформлення пояснювальної записки	26.04.24	

Здобувач _____

(підпис)

(прізвище та ініціали)

Керівник роботи _____

(підпис)

(прізвище та ініціали)

АНОТАЦІЯ

Дацько К.С. Мобільний застосунок для обміну речами, що не використовуються. – Кваліфікаційна робота бакалавра за спеціальністю 122 «Комп'ютерні науки».

Кваліфікаційна робота викладена на 42 сторінках, містить 3 розділи, 32 ілюстрації, 2 таблиці та 9 використаних джерел.

Об'єктом роботи є мобільний застосунок для обміну речами, що не використовуються.

Предмет роботи – процес проектування та розробки мобільного застосунку для обміну зайвими речами.

Метою роботи є проектування та розробка нативного мобільного застосунку для розміщення та публікації оголошень про обмін речами, які не використовуються.

Метод дослідження – прикладний.

В першому розділі виконаний аналіз предметної області та постановка задачі. Другий розділ присвячено опису розроблених алгоритмів та програмного забезпечення. Третій розділ містить результати експериментів та аналіз отриманих даних.

За результатами роботи зроблено висновки про отримані результати та потенціал її подальшого розвитку.

Ключові слова: Мобільний застосунок, Android, Java, Activity, NoSQL, Firebase.

ABSTRACT

Dats'ko K.S. A mobile app for sharing unused items. – Bachelor's qualifying work in specialty 122 "Computer Science".

The qualification work is laid out on 42 pages, contains 3 chapters, 32 illustrations, 2 tables and 9 used sources.

The object of the work is a mobile application for exchanging things that are not used.

The subject of the work is the process of designing and developing a mobile application for exchanging unnecessary things.

The purpose of the work is to design and develop a native mobile application for placing and publishing ads for the exchange of unused things.

The research method is applied.

In the first section, the analysis of the subject area and the formulation of the problem are performed. The second section is devoted to the description of the developed algorithms and software. The third section contains the results of the experiments and the analysis of the obtained data.

Based on the results of the work, conclusions were made about the obtained results and the potential for its further development.

Keywords: Mobile application, Android, Java, Activity, NoSQL, Firebase.

ЗМІСТ

ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА	
ЗАДАЧІ	8
1.1 Актуальність розробки	8
1.2 Огляд та аналіз існуючих аналогів	8
1.3 Постановка задачі	13
РОЗДІЛ 2. ОПИС РОЗРОБЛЕНИХ АЛГОРИТМІВ ТА ПРОГРАМНОГО	
ЗАБЕЗПЕЧЕННЯ	14
2.1 Огляд використовуваних технологій та інструментів	14
2.2 Проєктування бази даних	17
2.3 Структура мобільного застосунку	21
2.4 Алгоритм роботи одного з класів	28
2.5 Підключені модулі і бібліотеки	30
2.6 Опис функціоналу користувача	32
РОЗДІЛ 3. РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ ТА АНАЛІЗ ОТРИМАНИХ	
ДАНИХ	34
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	48
ДОДАТКИ	49
Додаток А. Програмний код	49

ВСТУП

Турбота про здоров'я має важливе значення для кожної людини. У період хвороби важливо мати можливість придбати ліки без необхідності купувати його в аптеці за високою ціною. До того ж і у людини, який вже перехворів залишається ліки, але які ще можна використовувати. І вони зберігаються там до тих пір, поки людина знову не захворіє або поки не закінчиться термін придатності.

Дуже складним завданням в пошуку людей, зацікавлених в обміні ліками, є відсутність можливості їх пошуку в реальному світі, а також відсутність інтернет групи або вебплатформи, яка могла б дати таку можливість. Звісно існує безліч вебзастосунків і сайтів, по розповсюдженню оголошень про обмін або навіть безкоштовний продаж будь-яких товарів і речей, але не всі мають вузьку спрямованість на ліки і медичні засоби. Для більшості людей бажано користуватися простий і не складною в освоєнні програмним забезпеченням, яке до того ж завжди перебуває під рукою.

Задачею дипломної роботи є створення нативного клієнтського мобільного застосунку мовою Java, який буде використовуватися для більш зручного пошуку людей з пропозиціями для обміну медикаментами, а також дозволяти створювати свої оголошення про можливість передати медикаменти.

Об'єктом роботи є мобільний застосунок для обміну речами, що не використовуються.

Предметом роботи є процес проєктування та розробки мобільного застосунку для обміну зайвими речами.

Метою роботи є проєктування та розробка нативного мобільного застосунку для розміщення та публікації оголошень про обмін речами, які не використовуються.

Методика дослідження – прикладна.

Практична значущість результатів дослідження: Мобільний застосунок готовий до публікації у Google Play Store.

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність розробки

Немає сумнівів що в сучасному світі звичайна людина проводить більшу частину часу за екраном смартфона. Кожен володар смартфона користується ним для перегляду новин, спілкування і викладанням фотографій. До того ж, це персональний пристрій для доступу в Internet.

У зв'язку з тим, що мобільні пристрої ставати все більш доступними, на мобільні додатки існує активний попит. Так само досить велика частина інтернет-трафіку припадатиме на мобільні девайси.

Основна причина актуальності теми застосунок – це можливість позбутися від непотрібних ліків. Майже у кожної людини або сім'ї в будинку або квартирі є аптечка, заповнена безліччю ліків, куплених в період, коли людина або член сім'ї був хворий. Однак найчастіше після одужання, залишається залишок від медичного препарату, в якому більше хворий не потребує. Залишок ліків може залишатися на зберіганні до закінчення терміну придатності. Набагато доцільніше передавати не використані ліки людям, які їх потребують, ніж просто викидати.

1.2. Огляд та аналіз існуючих аналогів

В результаті аналізу ринку інформаційних систем дошки оголошень було виявлено кілька схожих платформ серед яких виділив дві відомі веб-сторінки мають і аналог мобільного застосунка. Під час аналізу основним завданням було виявити спосіб відображення оголошень і функціональні можливості.

Першим розглянуто веб-сервіс OLX. Інтерфейс мобільного застосунку представлений на рис. 1.1.

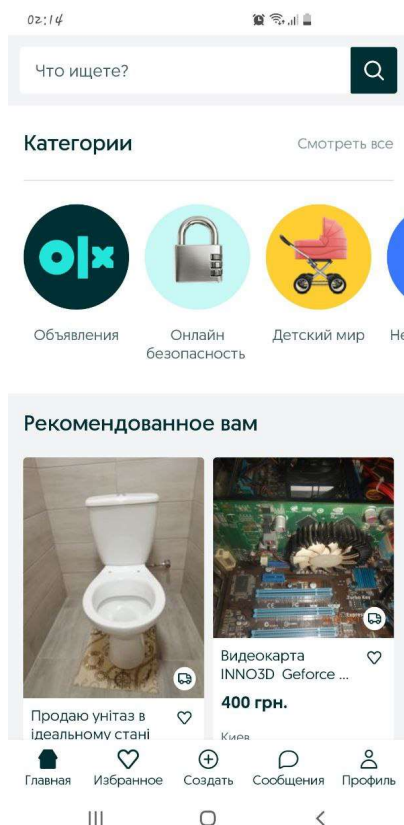


Рисунок 1.1 – Головний інтерфейс OLX

Не дивлячись на поширену думку про те, що OLX є проектом українським насправді він був створений за кордоном, а вже після злиття з сайтом Avito, став відомий для громадян України. До цього компанія мала багатий досвід в розробці сервісів з розміщення оголошень. На сьогоднішній день він налічує понад 1,5 млн користувачів в Україні та 330 млн користувачів в усьому світі.

Одне з головних переваг програми велике охоплення категорій. Опублікувати можна оголошення про продаж будь-якого виду товарів. Так само відзначив спосіб виведення списку оголошень, у вигляді списку з карток з назвою і висновком зображень товару, і наявністю зручного пошукова з фільтром по містах, по категоріях оголошень і за ціною.

В оголошеннях інформація має упорядкований вигляд панелі з галереї фотографій товару і крім назви має текстове поле, в якому публікує описує товар більш докладно. Функціонал платформи, дозволяє звернутися до публікує через

поле введення, або вивести на екран його номер телефону, що теж дає можливість зв'язатися.

Головним функціоналом цього застосунка це можливість створювати свої оголошення за допомогою форми з полем введення заголовка, вибору категорії, опису та контактних даних. У оголошення про товар застосунок дозволяє додавати фотографії з пам'яті пристрою або зробити фотографії використовуючи камеру смартфона. Після заповнення необхідних полів, перед публікацією користувач може побачити, як буде виглядати оголошення для інших користувачів.

Э кілька недоліків, пов'язаних з обов'язковими полями введення, сформованими незалежно від обмеження на максимальну кількість символів має і мінімальне. Це створює труднощі для користувача, так як насильно доводиться придумувати додатково опис. Так само додається SWOT-аналіз на рис. 1.2.



Рисунок 1.2 – SWOT-аналіз OLX

Таку програму для аналізу – це «ВіддамДаром». Це також застосунок для створення і публікації оголошень, схожий по функціональності з попереднім прикладом. Застосунок позиціонує себе як середовище барахолка, тобто для публікації оголошень про безоплатну передачу речей. Головний інтерфейс програми – це такий же список карток з фотографією і заголовком. Одночасно на

головному екрані є можливість фільтрувати оголошення по містам і регіонам, а також сортувати за датою публікації і розташуванню. Підтримка геолокації дозволяє виводити найближчі оголошення. Інтерфейс мобільного застосунку представлений на рис. 1.3.

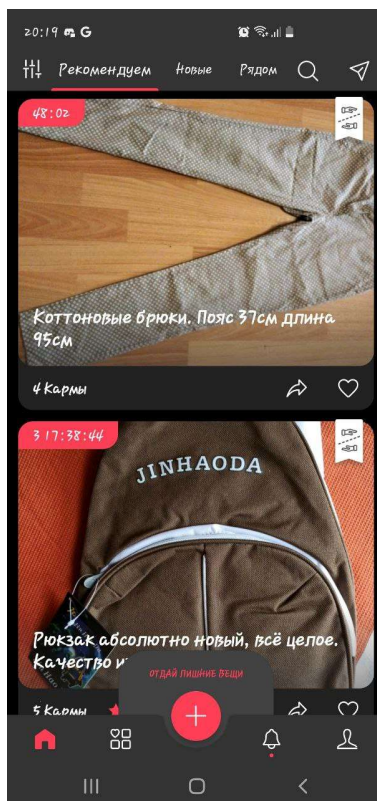


Рисунок 1.3 – Головный интерфейс ВіддамДаром

Але крім основного функціоналу застосунк має особливість в принципі обміну. Відкриваючи будь-яке оголошення, користувач не може зробити обмін «безкоштовно». Для здійснення операції з обміну, необхідно мати "карму" – чисельний параметр, який грає роль віртуальної валюти. При натисканні на кнопку "Хочу забрати" користувачеві необхідно заплатити певне значення валюти, яке визначає публікує. Вони нараховується автору, після здійснення угоди.



Рисунок 1.4 – SWOT-аналіз ВіддамДаром

Перш ніж сформувані очікувані від мобільного застосунку результати виконаємо порівняльний аналіз функцій і можливостей розглянутих рішень у табл. 1.1. Цей аналіз дозволить відмітити слабкі сторони застосунків і після цього враховувати їх при створенні вимог до дипломного проекту.

Таблиця 1.1 – Аналіз функціоналу наявних аналогів

Функції застосунків	OLX	ВіддамДаром
Створення оголошень про продаж за гроші	—	—
Створення оголошень про обміні товарами	+	—
Додавання до списку обраних	+	—
Реєстрація з використанням мейла і пароля	+	—
Можливість залишати коментарі / відгук	+	—
Редагування і видалення оголошень	+	+
Наявність категорій для декількох типів товарів	+	+
Створення оголошень про безкоштовної продажу	+	+
Спілкуватися з користувачами за допомогою чату	+	+
Реєстрація через Facebook	+	+
Реєстрація через Google	—	+

1.3 Постановка задачі

Метою проекту буде створення клієнтського мобільного застосунку «MedSharing» для розміщення та пошуку оголошень про обмін або безоплатної передачі медичних засобів та / або ліків, з наданням докладної інформації про засіб та контактних даних.

Вимоги при розробці мобільного застосунку:

- інтерфейс програми повинен бути інтуїтивно зрозумілим і зручним для користувачів;
- застосунок повинен повністю відповідати до цільової аудиторії;
- в застосунку повинна бути можливість створення оголошень та їх перегляд;
- застосунок повинен надавати повну інформацію про оголошення.

Після аналізу конкурентних рішень, представлених на ринку сформуємо завдання, які допоможуть створити мобільний застосунок.

Цілі які буду стоять на реалізації застосунку:

- сформулювати очікувані результати;
- спроектувати базу даних;
- обґрунтувати вибір технологічних рішень для розробки;
- розробити сервер застосунку;
- розробити інтерфейс мобільного застосунку;
- розробити програмне забезпечення;
- перевірити функціонування програмного забезпечення.

Розділ 2. ОПИС РОЗРОБЛЕНИХ АЛГОРИТМІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Огляд використовуваних технологій та інструментів

Android Studio – інтегроване середовище для розробки мобільних застосунків на базі Android. Була розроблена компанією JetBrains на основі іншого програмного забезпечення IntelliJ, що перейняла від можливості рефакторинга коду, інакше кажучи, структурного поділу складових програми, що значно полегшує розробку. IDE доступно безкоштовно для установки на персональний комп'ютер з операційними системами Windows, Mac і Linux.

Середовище розробки є операційною системою для більшості відомих електронних пристроїв, починаючи від смартфонів і закінчуючи телевізорами. Android дозволяє створювати Java-Застосунки з підтримкою сервісів і бібліотек Google.

На сьогоднішній день в усьому світі більша частина смартфонів на операційній системі Android і більш 2 млрд пристроїв працюють на ній.

Одне з основних властивостей фреймворка, це її повторювана безперервна інтеграція на етапі розробки. Дана властивість дозволяє виявляти помилки відразу, як тільки вони з'явилися. Це дозволяє більш ефективно розробляти код програми.

Додаткові особливості:

- вбудована робота з Github, з безліччю готових програм, команд і бібліотек, які полегшують роботу;
- підтримка інструмента Lint для відстеження сумісності з іншими версіями Android Studio;
- підтримка мов Java, C++, Kotlin;
- підтримка сервісів Google, що дозволяє поєднуватися проект із Firebase, Google Cloud Messaging;

- складання застосунка з використанням системи Gragle;
- можливість розробляти застосунок під будь-які пристрої Android;
- швидкий збір проекту в застосунок і його компіляція, як на фізичні пристрої, так і на вбудований емулятор.

Android-застосунок при розробці застосовує так звані вікна Activity, що працює за відведеним циклом (рис. 2.1, за матеріалами [2]). Увесь життєвий цикл в активності відбувається між першим викликом onCreate() до одного кінцевого виклику onDestroy(). Дія виконає все налаштування «глобального» стану в onCreate() і звільнить усі ресурси, що залишилися, в onDestroy(). Далі виклик onStart() відображає Activity, але вона може бути на передньому плані, поки не буде викликаний onStop(). В onResume() активність уже видна й з нею взаємодіє користувач. Для закриття вікна використовують метод onDestroy().

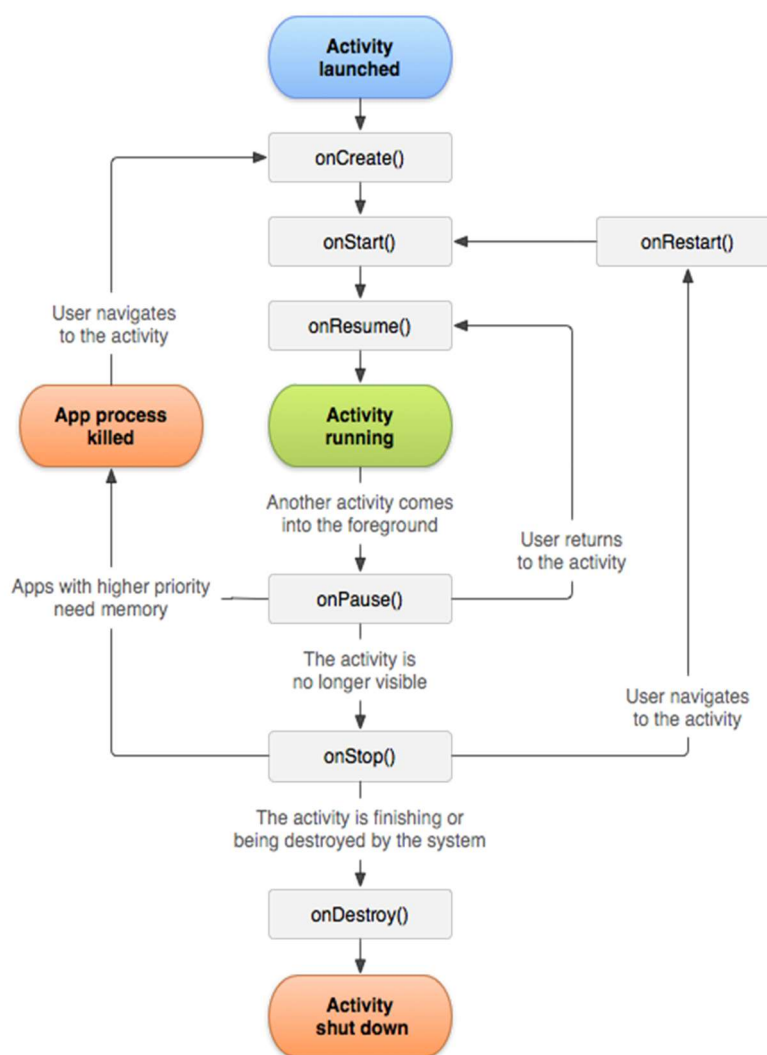


Рисунок 2.1 – Схема життєвого циклу застосунка Android

Java – мультиплатформна мова програмування, що відноситься до об'єктно–орієнтованих мов. Особливість мови полягає в можливості створювати застосунок, який буде запускатися на будь–якій платформі, при наявності на ній середовища виконавця JRE. Код просто компілюється в байт–код – незалежний від операційної системи. Отриманий код передається в інтерпретатор, який відправляє інструкції інтерпретатору.

Гнучка технологія дозволяє Java, допомагає в захисті системи, роблячи її підконтрольній системі. Зроблені операції перевищують встановлені дозволи або спроба доступу до даних без дозволу, приводять до негайного переривання коду. В Java так само збирач сміття *garbage collector* – механізм керування пам'яттю. Після того як частина коду була використана, збирач очищає пам'ять від них.

Так само Java часто використовують для розробки мобільного застосунка. Застосунок зібране в Android Studio, легко компілюється так само в байт–код у вбудовану віртуальну машину.

Система керування базою даних, інакше кажучи СКБД єднає інтерфейс із базою даних і розробленим застосунком, організовуючи доступ до сховища даних. Вона дозволяє доповнювати, змінювати або стерти дані, працює сумісно з клієнтською частиною.

Firebase – одне із із таких, але з деякими особливостями. Будучи *backend* воно дозволяє зберігати дані у віддаленому сервері. У теж час він надає великий функціонал: аутентифікація, підтримка правил читання й запису, можливість зберігати зображення, збір даних, аналіз роботи і так далі.

Головні переваги роботи з Firebase, це:

- простою робота й менш енерговитратна розробка застосунка;
- платформа має готовий набір функцій;
- можливість збирати аналітику з користувачів, що користуються застосунком;
- робота з Android Studio, що надає свої бібліотеки для роботи з функціоналом.

Firebase здатний працювати сумісно з Android Studio, надаючи зручне підключення до нового проекту. Основним функціоналом є база даних NoSQL, за допомогою якої можна зберігати, передавати дані користувачів.

Плюси NoSQL бази даних:

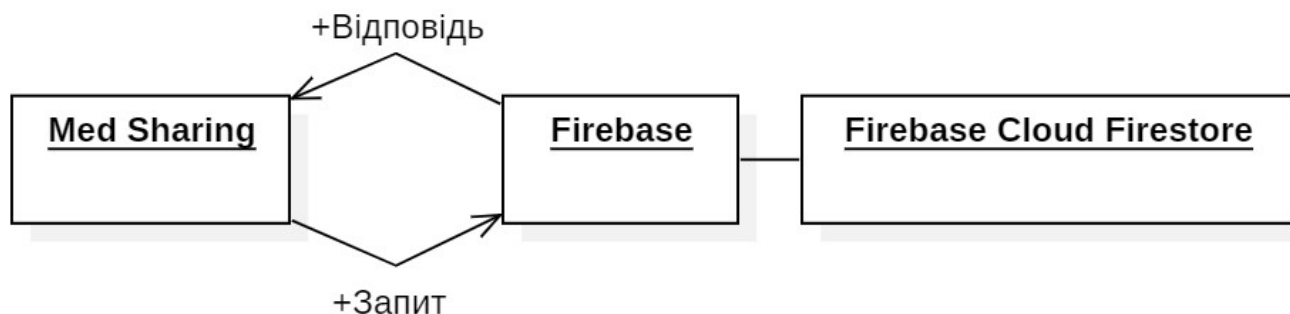
- лінійна масштабованість – додавання нових вузлів в кластер збільшує загальну продуктивність системи;
- гнучкість, що дозволяє оперувати напівструктуровані дані, реалізуючи, в. т.ч. повнотекстовий пошук по базі;
- можливість працювати з різними уявленнями інформації, в т.ч. без завдання схеми даних;
- висока доступність за рахунок реплікації даних і інших механізмів відмовостійкості;
- шарінг – автоматичного розподілу даних за різними вузлами мережі.

Мінуси NoSQL бази даних:

- складності в підтримці всіх ACID-вимог до транзакцій (атомарність, консистентність, ізоляція, довговічність) через те, що NoSQL-СКБД замість CAP-моделі (узгодженість, доступність, стійкість до поділу) скоріше відповідають моделі BASE (базова доступність, гнучке стан і підсумкова узгодженість)
- сильна прив'язка застосунку до конкретної СКБД через специфіку внутрішньої мови запитів.

2.2 Проєктування бази даних

Мобільний застосунок для розміщення оголошень буде працювати за принципом клієнт-серверної архітектури, яка складається з сервера бази даних – Firebase і клієнта – мобільного застосунку (рис. 2.2).



*Рисунок 2.2 – Архітектура клієнт-сервер мобільного застосунку
з базою даних Firebase*

Клієнтом є мобільний застосунок. Застосунок показує оголошення існуючі в базі, дані користувача, а також дані інших користувачів.

Мобільний застосунок відправляє на сервер дані аутентифікації, дані користувача, а також дані для створення оголошення.

Для зберігання й синхронізації даних мобільного застосунка буде використовуватися хмарне сховище Cloud Firebase. Це Nosql база даних, що використовує документо-орієнтовану модель, в ній немає таблиць та строк, лише колекції та документи.

Документи містять набір полей-значень, с підтримкою усіх типів даних. Колекції – це контейнери, в якому містяться документи.

База даних може зберігати дані будь-якого типу й виду, без яких або обмежень. Перед початком роботи з нею необхідно встановити ряд правил безпеки: read – можна читати, write – можна записувати.

У колекції користувачів «Users» (рис. 2.3) перебувають документи ключем. У них є ключ реєстрації унікальний для кожного користувача, та дані користувача: короткий опис про себе «bio», прізвище «last», ім'я «first». Крім того буде зберігатиметься посилання на зображення аватарки користувача «image» в окремому сховищі. Вкладену колекцію «Readers» (рис.2.4) в якій є документи з оцінкою про то ким і коли користувач відзначен, як надійний.



Рисунок 2.3 – Вузол Users



Рисунок 2.4 – Вузол Readers

У колекції «Posts» (рис. 2.5) зберігаються документи з даними про кожне оголошення його заголовку, зображення, короткому й докладному описі. У вкладених колекціях «Comments» (рис. 2.6) і «Likes» (рис. 2.7), де в першому зберігаються документи відгуки за датою, повідомлення й ключ користувача, а в другому документ похвали з датою коли вона була поставлена.

Остаточна схема бази даних наведена у схемі-діаграмі (рис. 2.8).



Рисунок 2.5 – Колекція Posts

Comments	>	BLrLgU0U1Zj6cQR97Zwm	>	+ Add field
Likes		ot2kDf3qTFdwchFYtI5a		message: "Hay"
				timestamp: June 7, 2021 at 12:06:16 AM UTC+3
				user_id: "B2l5dqbq7TelBkvrKKrTk2t4Pxxw1"
+ Add field				

Рисунок 2.6 – Колекція Comments

76MA8LI0vyctwuthVMwO	>	Comments	tQQMh6LE03Z2LuDbtD6mA4FWPY52
SIEQ2QsuhV8G1hQCfPca		Likes	>
oWtkrZ2RZWd1tpPxL1Y4			

Рисунок 2.7 – Колекція Likes

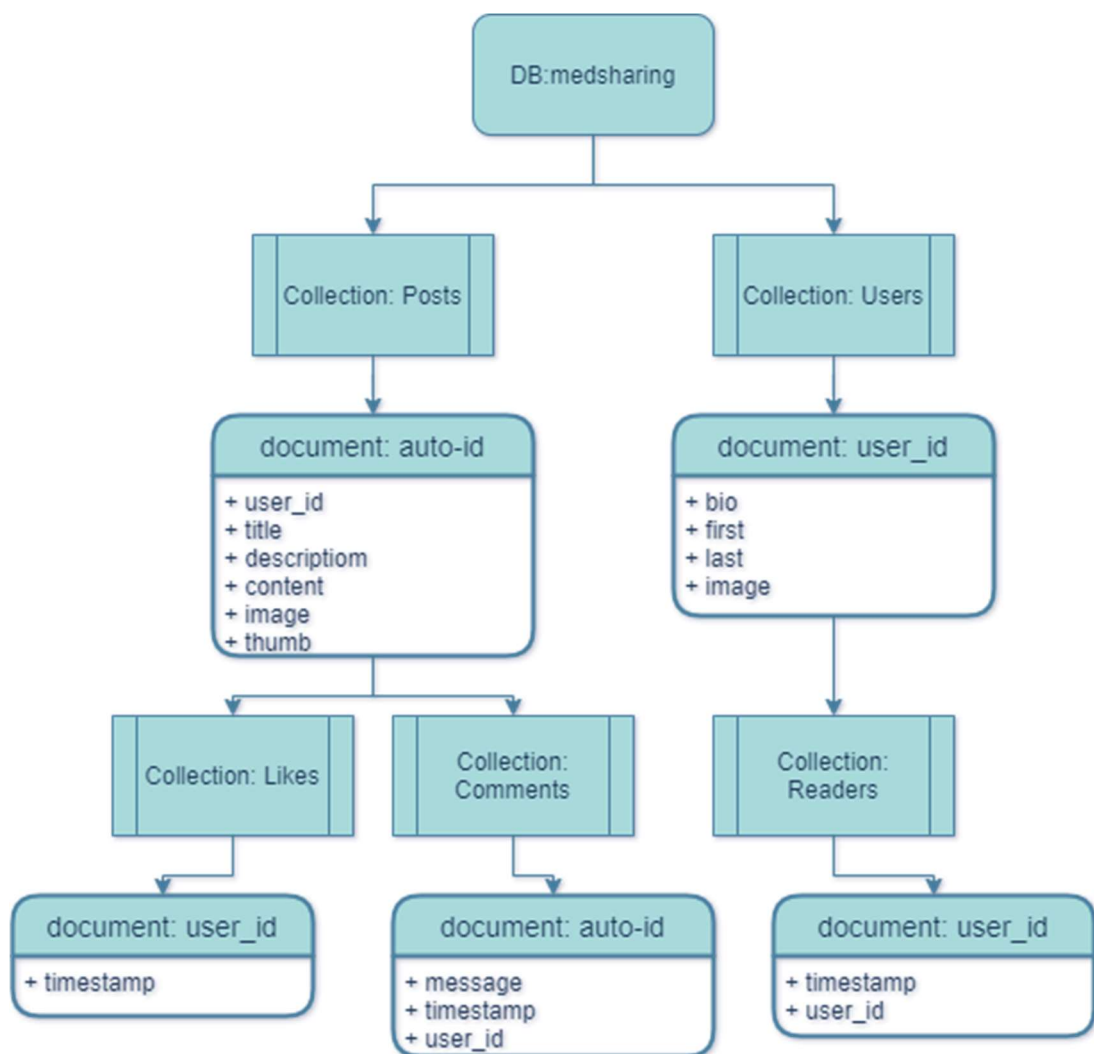


Рисунок 2.8 – Схема бази даних у CloudFirestore

2.3 Структура мобільного застосунку

Як впливає з очікуваного функціоналу розроблюваної застосунку можна поділити на окремі компоненти. У функціонально-орієнтованій діаграмі компонентів, наведено з чого буде складатися майбутній застосунок (рис. 2.9).



Рисунок 2.9 – Схема бази даних у CloudFirestore

Організація застосунку на основі декількох activity не є оптимальною. У програмі буде використовуватися фрагменти, які представляють шматочок візуального інтерфейсу програми.

Усі класи графічно представлені на діаграмі класів (рис. 2.10).

Мобільний застосунок складається з класів activity, які створюють вікна застосунку, та їх функціонал.

До класів типу діяльність належать такі класи як:

- CommentsActivity;
- LoginActivity;
- MaintActivity.
- RegisterActivity;
- NewPostActivity;
- SetupActivity.

Клас `CommentsActivity` – екран для відображення, та створення відгуків. Містить лише один метод ініціалізації, який створює зовнішній вигляд екрану, запити до бази на читання коментарів до окремого оголошення та запис коментарів.

`LoginActivity` – екран авторизації. Містить метод ініціалізації екрану, параметрів для авторизації та запитів на підтвердження авторизації. Також має методи:

- `sendToMain` – якщо з бази прийшло підтвердження, відкриває `MainActivity`.
- `clickToRegister` – відкриває екран `RegisterActivity`.

`MainActivity` – головний екран, з полем для фрагментів `HomeFragment`, `AccountFragmen`. Метод ініціалізації формує екран головного меню, меню для перемикачів між фрагментами та спливаюче вікно на шапці екрану. Після чого йде перевірка на авторизацію користувача. Також має методи:

- `onCreateOptionsMenu` – ініціалізують всплавюче меню;
- `onOptionsItemSelected` – для кожної кнопки меню додає методи `logOut`, `sendToSetting` та `sendToAdPost`;
- `sendToAdPost` – відкриває `NewPostActivity`;
- `sendToSetting` – відкриває `SetupActivity`;
- `logOut` – деавторизує користувача та запускає метод `sendToLogin`;
- `sendToLogin` – активується, якщо користувач не пройшов перевірку на авторизацію, або сам захотів. Відкриває вікно `LoginActivity`;
- `replaceFragment` – перемикач між фрагментами `HomeFragment`, `AccountFragment`;
- `initializeFragment` – додає `HomeFragment`, `AccountFragment`, до контейнеру.

`RegisterActivity` – екран реєстрації. Містить метод ініціалізації екрану, параметрів для реєстрації та запитів на створення облікового запису. Після чого йде підтвердження на, що користувач авторизован. Також має методи:

- `sendToSetup` – відкриває вікно `SetupActivity`;
- `sendToMain` – якщо з бази прийшло підтвердження, відкриває `MainActivity`;
- `clickToLogin` – відкриває екран `RegisterActivity`.

NewPostActivity – екран з полями вводу для створення оголошень. Містить метод ініціалізації екрану, параметрів для створення оголошення. Також, завдяки додатковим бібліотеками ініціалізує, завантаження зображення з пам'яті телефону. До того, як створити запит про створення оголошення, йде обов'язкова перевірка на поля вводу та зображення. Також має методи:

- `sendToMain` – якщо з бази прийшло підтвердження, відкриває MainActivity;
- `onActivityResult` – перевірка зображення.

SetupActivity – екран настройки даних профілю. Містить метод ініціалізації екрану, параметрів для редагування даних користувача. Якщо користувач, тільки зареєструвався поля, будуть пусті. Перед підведенням налаштування, йде перевірка на заповнення полів. Також, завдяки додатковим бібліотеками ініціалізує, завантаження зображення з пам'яті телефону. Також має методи:

- `onActivityResult` – перевірка зображення;
- `popUpImagePicker` – завантажує зображення з бази, якщо користувач його вже має;
- `sendToMain` – якщо з бази прийшло підтвердження, відкриває MainActivity.

Класи фрагментів, доповнюють вікна діяльності внутрішніми вікнами, або створюють список елементів (рис. 2.12). До класів типу Fragment належать такі класи як:

- AccountFragment;
- HomeFragment;
- PostFragment;
- AnotherAccountFragment.

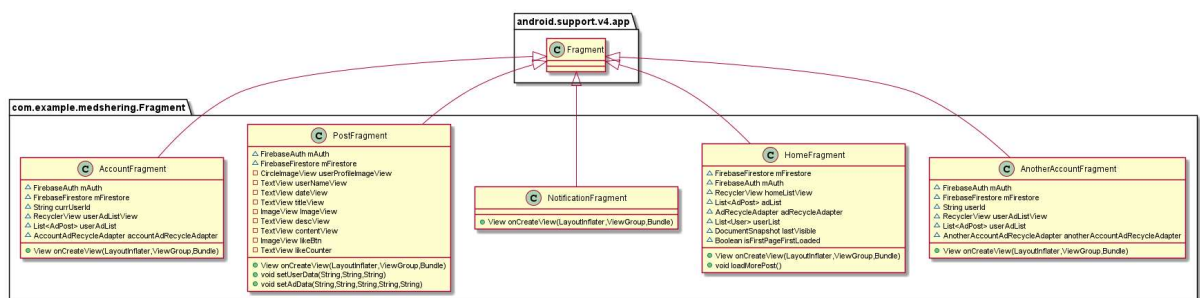


Рисунок 2.12 – Класи фрагментів

AccountFragment – внутрішній екран MainActivity, список оголошень у профілі користувача. onCreateView створює масив оголошень користувача, за відображення яких відповідає AccountAdRecycleAdapter. У цьому ж методі створюється запит до бази на читання оголошень користувача.

HomeFragment – внутрішній екран MainActivity, список оголошень. onCreateView створює масив оголошень у базі, за відображення яких відповідає AdRecycleAdapter. loadMorePost метод завантаження додаткових оголошень, коли користувач перегортає список до кінця.

PostFragment – внутрішній екран MainActivity, детальна інформація про оголошення. onCreateView створює екран зовнішнього вигляду екрану, на якому запитом до бази додається текстова та візуальна інформація оголошення та автора оголошення. Також має методи:

- setUserData – ініціалізує мале зображення автора;
- setUserData – ініціалізує зображення оголошення.

AnotherAccountFragment – внутрішній екран MainActivity, список оголошень іншого користувача. onCreateView створює масив оголошень користувача, за відображення яких відповідає AnotherAccountAdRecycleAdapter. У цьому ж методі створюється запит до бази на читання оголошень користувача. Адаптери в свою чергу беруть дані, та відображають їх на кожному з елементів фрагменту (рис. 3.12).

До класів типу фрагмент належать такі класи як:

- AccountAdRecycleAdapter;
- AdRecycleAdapter;
- AnotherAccountAdRecycleAdapter;
- CommentsRecyclerAdapter.

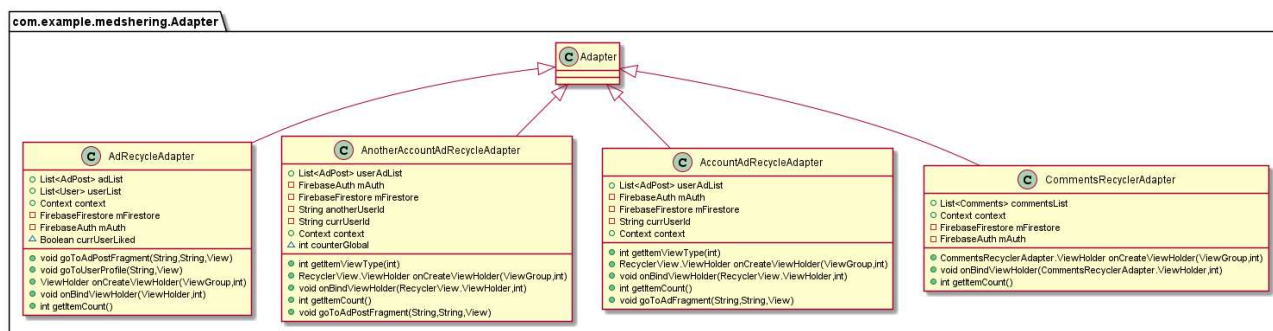


Рисунок 2.13 – Класи адаптери

AccountAdRecycleAdapter – відображає дані профіля та оголошення у AccountFragment.

AdRecycleAdapter – відображає дані оголошень у PostFragment;

AnotherAccountAdRecycleAdapter – відображає дані профілю та оголошень іншого користувача у AnotherAccountFragment;

CommentsRecyclerAdapter – відображає дані відгуків Comments.

Щоб одержувати дані з бази були структуровані, створено додаткові класи моделі. Їх мета одержувати дані з бази формувати у окремі екземпляри класу. Кожен такий клас містить методи для вилучення окремих параметрів з бази (рис. 2.14).

До класів типу модель належать такі класи як:

- AdPost;
- AdPostId;
- Comments;
- User.

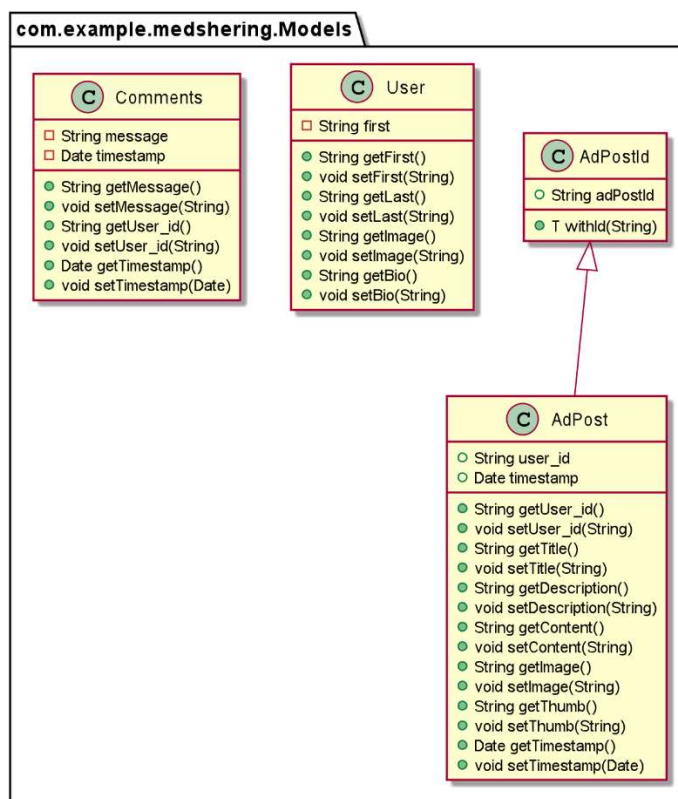


Рисунок 2.14 – Класи моделі

AdPost – модель для даних оголошення. Містить параметри даних оголошень і методи для виведення даних у вигляді строк. Наслідує клас **AdPostId**. **Comments** – модель для даних відгуків. Містить параметри даних відгуків методи для виведення даних користувача у вигляді строк.

User – модель для даних відгуків. Містить параметри даних відгуків методи для виведення даних користувача у вигляді строк.

2.4 Алгоритм роботи одного з класів

Для демонстрації роботи класу `LoginActivity` була побудована, діаграма послідовності (рис. 2.15).

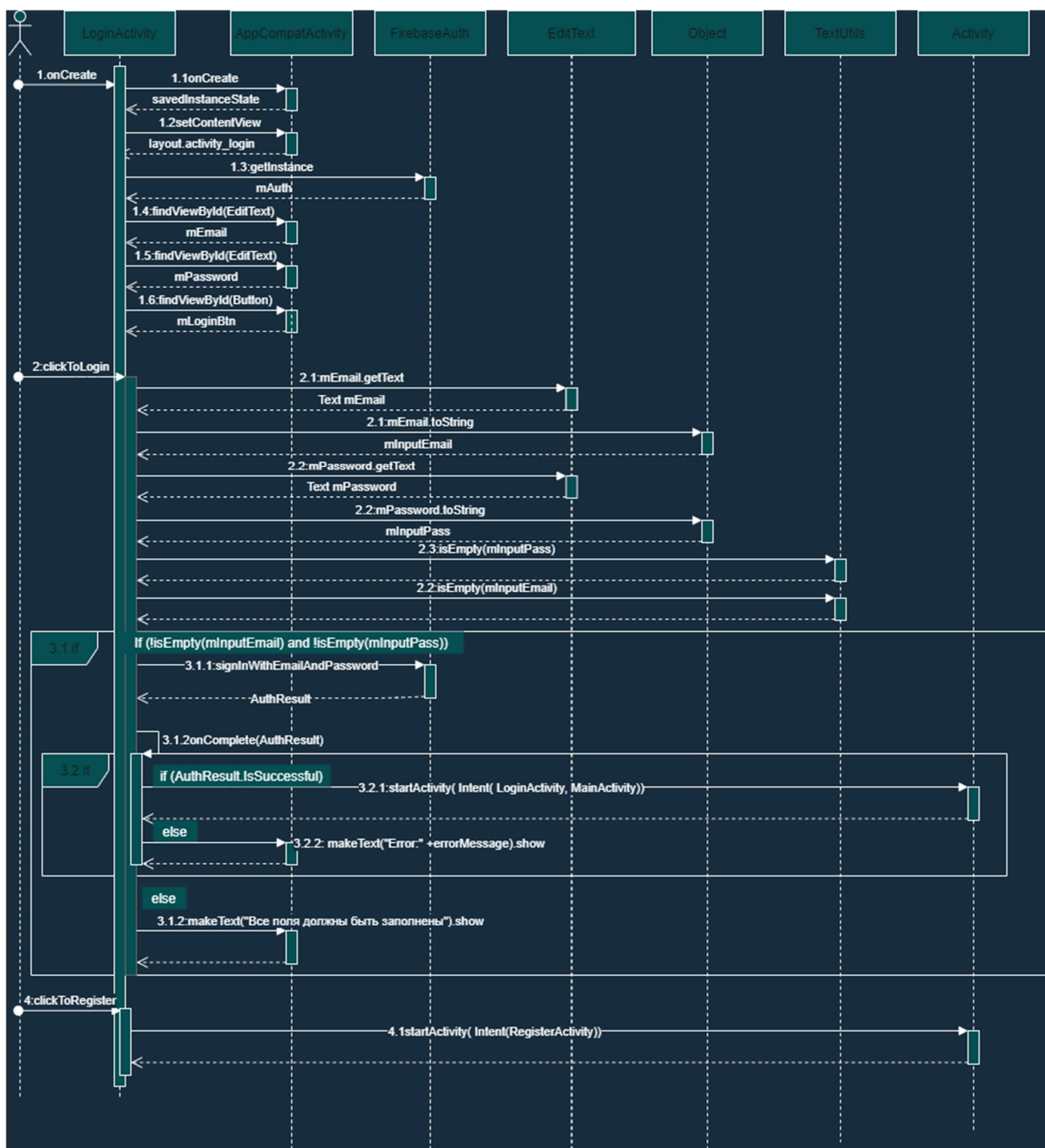


Рисунок 2.15 – Діаграма послідовності LoginActivity

У LoginActivity використовуються додаткові шість класів. Основний клас успадковує AppCompatActivity для використання основних функцій платформи. FirebaseAuth клас бібліотеки Firebase для авторизації користувача. EditText - для роботи з елементом інтерфейсу введення й змінювання тексту. Object суперклас для роботи з іншими класами. TextUtils містить корисні методи для роботи з

текстом. Клас Activity для роботи з вікнами інтерфейсу. Послідовність LoginActivity наступна. Спочатку метод onCreate ініціалізує активність. Потім setContentView передає ресурс розмітки інтерфейсу activity_login. Отримуємо екземпляр класу FirebaseAuth. Знаходимо поля введення та кнопку входу, за допомогою findViewById, та переходимо до режиму слухача. Коли користувач нажимає кнопку «Войти», запускається метод clickToLogin. getText читає поля введення та повертає Text, потім toString надає строчний тип. isEmpty повертає істину, якщо строки пусті. Далі йде перевірка введених даних. Після перевірки на істину, що поля були заповнені, в іншому випадку виведе на екран повідомлення про це, строки mEmail та mPass методом signWithEmailAndPassword відправляються у бази для підтвердження авторизації. Якщо все пройшло успішно startActivity закриває вікно авторизації та пересилає користувача до головного екрану MainActivity. Якщо мейл не був коректним, або введені дані не співпадають з жодним обліковим записом в базі, makeText виведе повідомлення про помилку. Додатково користувач може натиснути на кнопку «Впервые здесь?» і запустити вікно RegisterActivity.

2.5 Підключені модулі і бібліотеки

Для підключення до бази даних Firebase використовуються наступні модулі:

- Firebase core – ініціалізації Firebase в проєкті;
- Firebase auth – авторизації і реєстрація користувачів;
- Firebase database – призначений для підключення до бази даних;
- Firebase storage – для підключення до сховища.

Крім звичайних бібліотек в проєкті застосовувалася бібліотека CircleImageView – аналог imageView з круглою рамкою (рис. 2.16).

```
tools:context=".Activity.SetupActivity" >

<de.hdodenhof.circleimageview.CircleImageView
    android:id="@+id/profile_image"
    android:layout_width="128dp"
    android:layout_height="128dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="96dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"

    android:src="@drawable/profileimage"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Рисунок 2.16 – Приклад реалізації в екрані аутентифікації

Бібліотека Glide одна з найпопулярніших, та дозволяє завантажувати зображення, як з пристрою так і з сервера. Разом із нею використовувалася бібліотека Cropper.Js для обрізки завантажуваних фотографій. Застосовувалися при реалізації вікна налаштувань призначених для користувача даних, де тепер можна завантажити аватарки для профілю (рис. 2.17).

```
currUserId = mAuth.getCurrentUser().getUid();

mChoosePhoto.setOnClickListener((view) -> {
    CropImage.activity()
        .setGuidelines(CropImageView.Guidelines.ON)
        //.setMinCropResultSize(256, 128)
        //.setAspectRatio(2, 1)
        .start( activity: NewPostActivity.this);
});
```

Рисунок 2.17 – Завантаження фотографій на екрані автентифікації

Для того, щоб не виникало проблем з зображеннями які завантажуються, допоможе Compressor, що дозволяє стискати фотографії не псуючи їх якість. При

завантаженні зображення для створення оголошень це дозволило зберегти розмірність оголошень (рис. 2.18).

```
try {
    compressedImageFile = new Compressor( context: NewPostActivity.this)
        .setMaxHeight(100)
        .setMaxWidth(100)
        .setQuality(1)
        .compressToBitmap(newThumbFile);
} catch (IOException e) {
    e.printStackTrace();
}
```

Рисунок 2.18 – Стиснення в вікні створення оголошення

2.6 Опис функціоналу користувача

Для визначення чітких вимог до проєктованої програмної системи варто визначити, що повинна програма буде надавати користувачеві. У діаграмі варіантів (рис. 2.19) описуються можливості для користувача. У застосунку передбачена дві можливі ролі «Користувач не авторизований» – людини яка вперше завантажила застосунок, та «Користувач авторизований» – у якого є обліковий запис у системі.

Суцільні лінії вказують на відносини між асоціаціями, які показують варіант використання. Користувач, запустивши застосунок має можливість перемикатися між вкладками.

Пунктирними лініями вказують зв'язок розширення (extend) або включення (include), які виникають між варіантами використання.

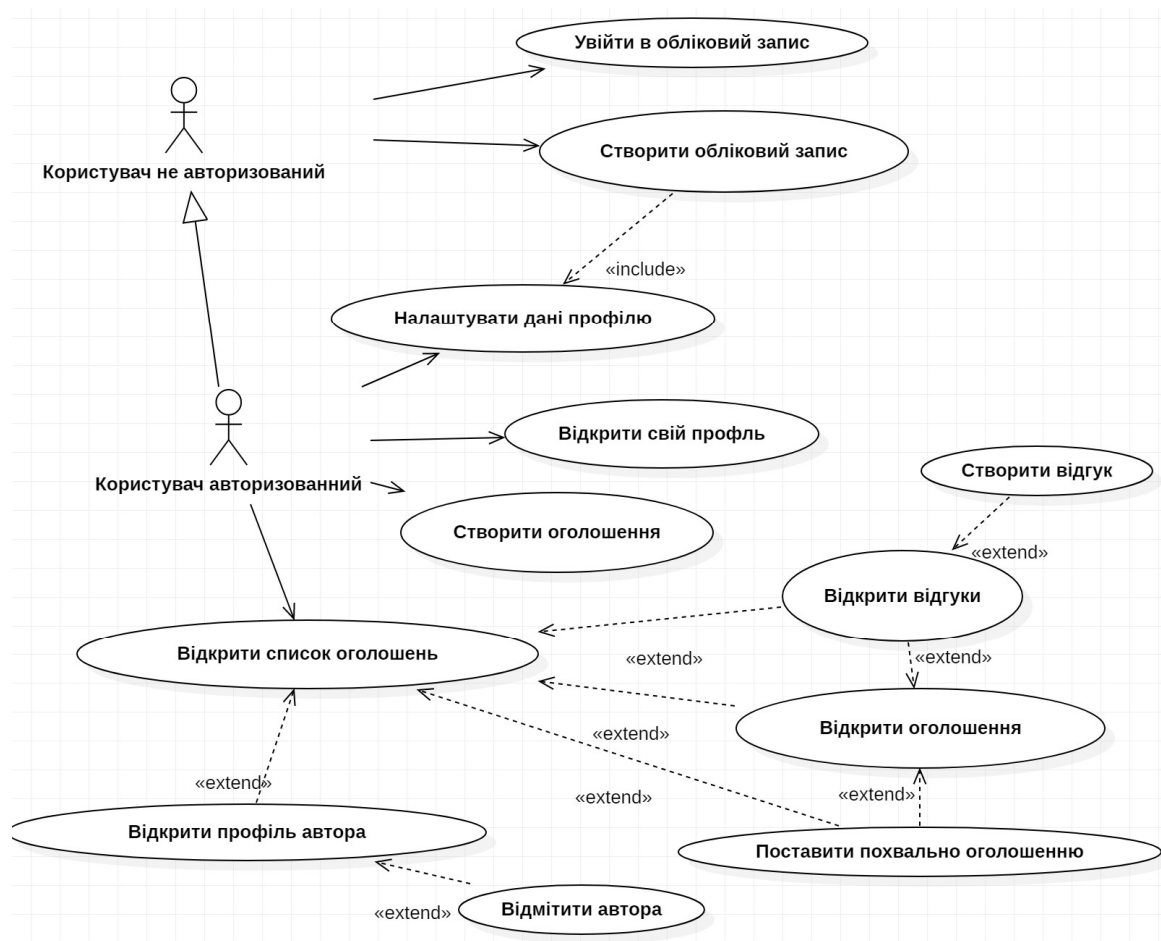


Рисунок 2.19 – Діаграма варіантів використання

Відношення включення (include) з'являється, коли один варіант використання включається в інший крапці й варіанти стають компонентами. Відношення розширення (extend) відображає два варіанти використання у застосунку. Тільки за певних умов використовується варіант розширення.

Користувач тільки після авторизації може використовувати весь функціонал. Йому пропонується або створити обліковий запис, або увійти в існуючу. При першому варіанті, він може настроїти дані профілю або зробити це в будь-який зручний для нього момент. Для зареєстрованих користувачів, відкривається головна сторінка з якої можна переглянути існуючі оголошення в базі, створити оголошення або переглянути свій профіль. У списку оголошень користувач може подивитися більш детальну інформацію, відкривши оголошення. Після цього, або потім можливо буде відправити відгук, який буде бачити усі користувачі.

Розділ 3. РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ ТА АНАЛІЗ ОТРИМАНИХ ДАНИХ

При вході в застосунок користувача зустрічає екран авторизації (рис. 3.1). Якщо користувач вперше встановив застосунок, йому необхідно пройти реєстрацію. Після цього застосунок готовий до використання.

The image displays two side-by-side screenshots of the MedSharing application interface. Both screens feature a light gray background and a central icon of a clipboard with a medical cross. Below the icon, the text 'MedSharing' is displayed in a light gray font.

The left screenshot shows the login screen. It includes two input fields: 'Електронна пошта' (Email) and 'Пароль' (Password). Below these fields is a dark blue button with the text 'Увійти' (Login) in green. At the bottom of the screen, there is a link that says 'Вперше тут?' (First time here?).

The right screenshot shows the registration screen. It includes three input fields: 'Електронна пошта' (Email), 'Пароль' (Password), and 'Підтвердіть пароль' (Confirm password). Below these fields is a dark blue button with the text 'Реєстрація' (Registration) in green. At the bottom of the screen, there is a link that says 'Вже є акаунт?' (Already have an account?).

Рисунок 3.1 – Екрани авторизації

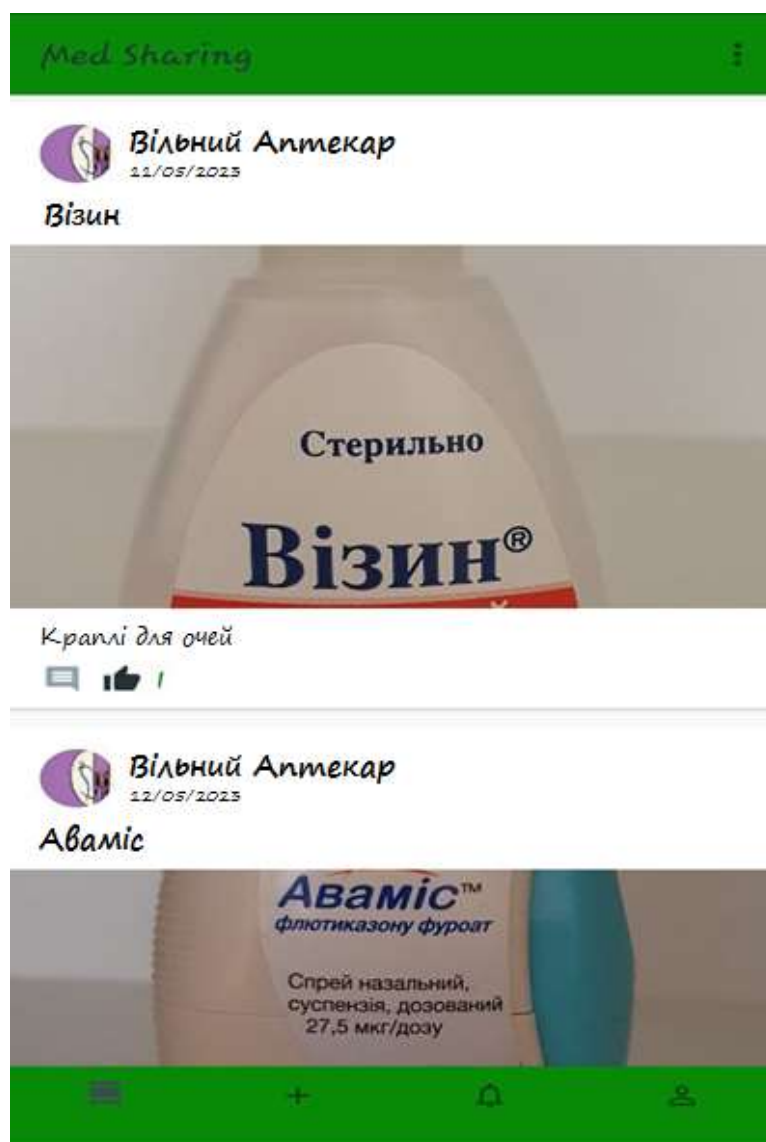


Рисунок 3.2 – Головна сторінка застосунка з меню вкладками

Перед нами головне вікно програми, меню з вкладками, кожне з яких відкриває певний вікно (рис. 3.2). Тут відразу ви зможете побачити завантажені оголошення.

При натисканні на оголошення, а саме – по зображенню, відкриється вікно з більш детальною інформацією (рис. 3.3).



Рисунок 3.3 – Вікно оголошення

Додати нову обяву

Оберіть фото

Заголовок

Короткий опис

Детальний опис

Додати оголошення

Додати нову обяву

Фото обрано

Хепілоп

Спрей для роту (продаж)

Користувався місяць і потім залишив у коробці на рік із інструкцією. Зберігається у належних умовах. Віддам за 100 грн. Звертайтеся за номером (XXX)-XXX-XX-XX

Додати оголошення

Рисунок 3.4– Екрани форми створення об'яви

Якщо ж ви хочете створити своє оголошення натисніть на значок плюса «+» і перед вами відкривається вікно «форма додавання». При натисканні на напис «Виберіть фото», ви можете додати фотографію з ліками з галереї або сфотографувати прямо зараз. Після цього, заповніть поля: «Тема», «Короткий опис» і «Детальний опис». Після чого натисніть на кнопку «Додати оголошення» і почекайте поки оголошення завантажується (рис. 3.4).

Звернімо увагу, що на головній сторінці, на кожному оголошенні показано лише заголовок і короткий опис (рис. 3.5). Тут також можна поставити лайк, в знак того, що оголошення здалося вам корисним.

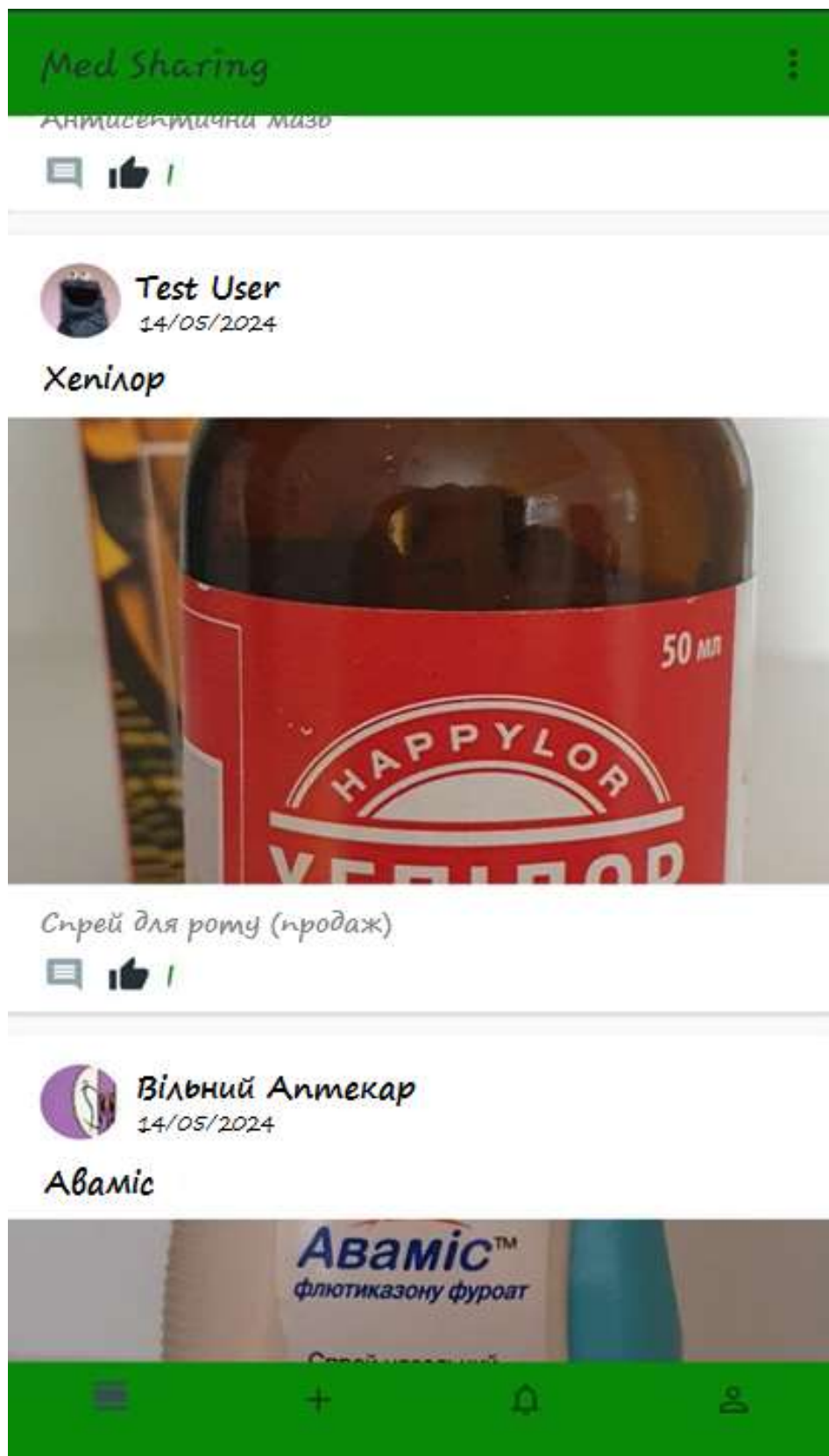


Рисунок 3.5 – Нове оголошення

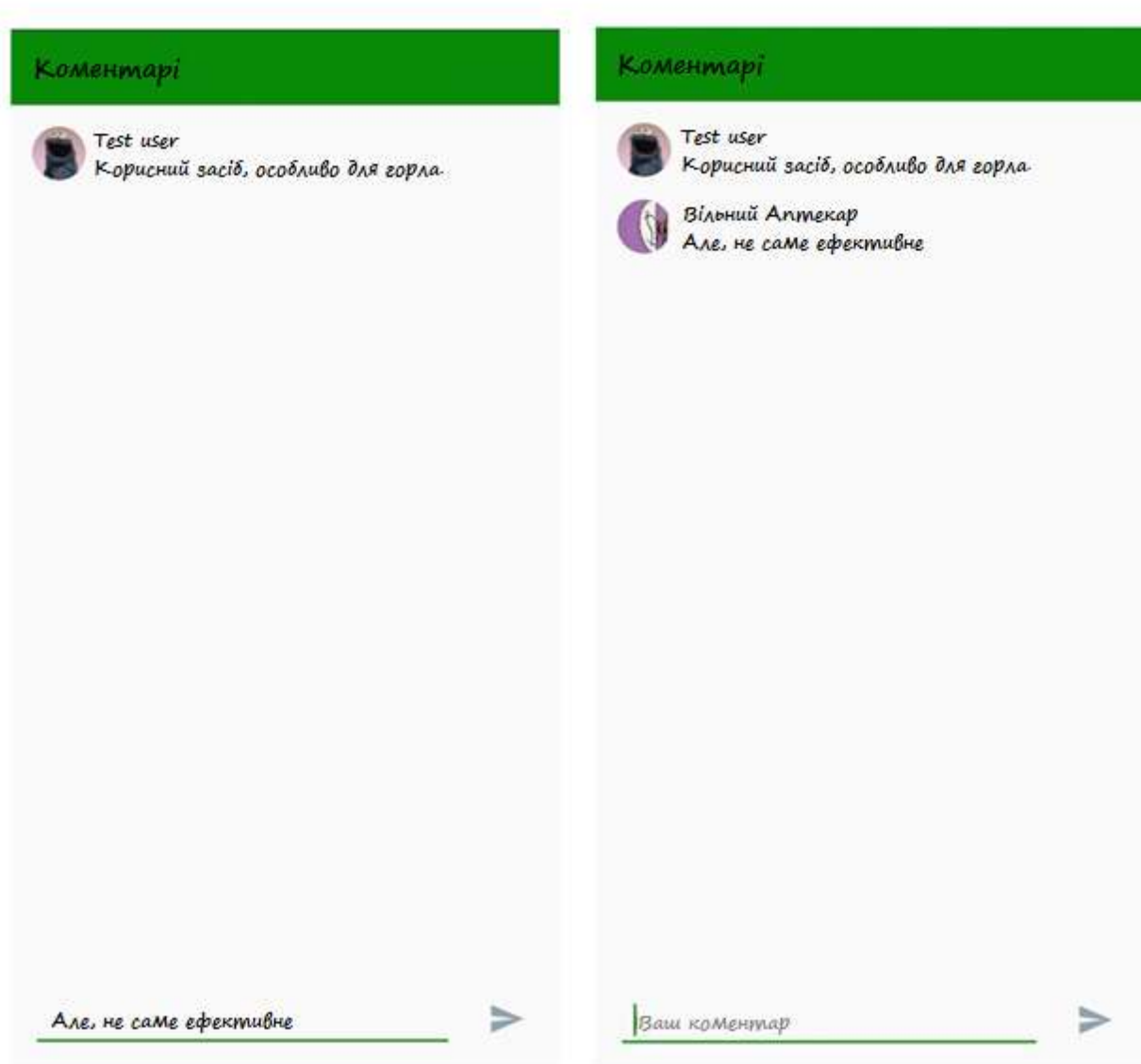


Рисунок 3.6 – Вікна коментарів

При натисканні значка повідомлення під коротким описом ви відкриєте коментарі, де будь-який користувач може залишити відгук про оголошення або думка за ціною. Сторонній користувач або навіть ви самі можете ввести повідомлення та натиснути стрілку, коментар буде додано і всі бачать (рис. 3.6).

При натисканні на кнопку «профіль» відкриється вікно з інформацією про користувача, його ім'я, прізвище та коротка інформація про нього самого (рис. 3.7).



Рисунок 3.7 – Вікно профілю

Зверху також є спливаюче вікно з трьома кнопками. Яка надає додатковий функціонал, з вибору трьох вкладок (рис. 3.8):

- Налаштування акаунта;
- Додати нову об'яву;
- Вийти.



Рисунок 3.8 – Спливаюче меню

При виборі «Налаштування облікового запису» з'явиться форма заповнення з фотографією і полями введення (рис 4.9). При натисканні на округлу форму зображення ви можете завантажити собі нове і після внесених змін до поля введення, після натискання кнопки «Зберегти настройки», ви повернетеся на головний екран.

Для виходу з програми необхідно натиснути на три точки у верхній частині програми та натиснути «Вийти». Користувач повернеться до вікна автентифікації.

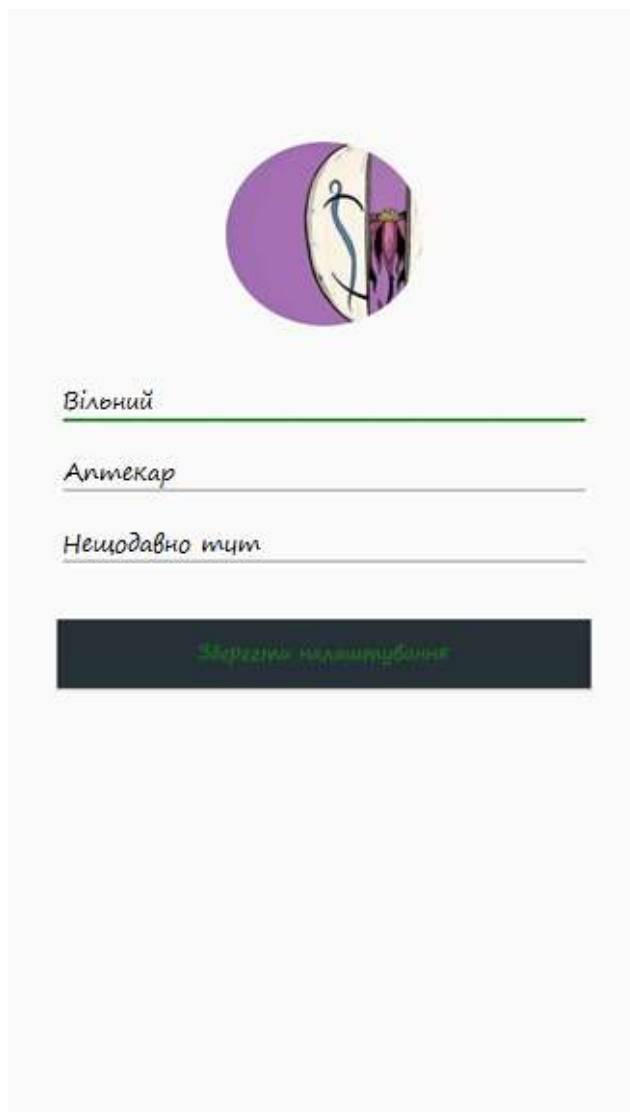


Рисунок 3.9 – Налаштування профілю

Протягом розробки застосунків неодноразово тестувалася з метою виявлення неполадок і невідповідностей задуманому функціоналу. В Android Studio було проведено серію тестів на смартфонах з різними версіями ОС Android. При виявленні помилки в застосунку, після аналізу поведінки, вносилися зміни в код програми. Активності були піддані перевірці на всіх етапах їх життєвого циклу.

Після перевірки на емуляторах застосунків було перевірено і на реальних фізичних пристроях. При цьому було проведено функціональне тестування, що дозволяють оцінити працездатність застосунку очима користувача (табл. 3.1).

Таблиця 3.1 – TestCase мобільного застосунку

Функціонал	Test-case	Кроки	Очікуваний результат	Результат тестування
Увійти в обліковий запис	Перевірка авторизації неіснуючого користувача	1) Ввести в поле електронну пошту не зареєстровану в систему, і будь-який пароль 2) Натиснути «Увійти»	Застосунок видасть помилку про неіснуючу або віддаленої облікового запису	✓
	Перевірка авторизації існуючого користувача	1) Ввести в поле електронну пошту зареєстровану в систему і будь-який пароль 2) Натиснути «Увійти»	Застосунок відкриє головний екран користувача	✓
	Перевірка авторизації з порожніми полями	1) Залишити всі поля введення порожнім 2) Натиснути «Увійти»	Застосунок видасть помилку про необхідність заповнити поля	✓
	Перевірка авторизації з неправильно введеним мейлом	1) Ввести в поле електронної пошти мейл без собачки і пароль 2) Натиснути «Увійти»	Застосунок видасть помилку про некоректне введення мейла	✓
Створити обліковий запис	Перевірка реєстрації користувача з некоректним мейлом	1) Ввести в поле введення мейл без собачки і пароль 2) Натиснути «Реєстрація»	Застосунок видасть помилку про некоректне введення мейла	✓
	Перевірка реєстрації користувача з однаковими паролями	1) Ввести в поле мейл і випадкову рядки в поле «Пароль» і «Підтвердження пароля»	Застосунок видасть помилку про не збігаються паролі	✓

Функціонал	Test-case	Кроки	Очікуваний результат	Результат тестування
		2) Натиснути «Реєстрація»		
	Перевірка реєстрації користувача з існуючим файлом	1) Ввести в поле введення зареєстрований мейл і та пароль 2) Натиснути «Реєстрація»	Застосунок видасть помилку про існуючий обліковий запис	✓
	Перевірка реєстрації з порожніми полями	1) Залишити поля введення порожніми 2) Натиснути «Реєстрація»	Застосунок видасть помилку про порожні полях	✓
Налаштувати профіль користувача	Перевірка настройки профілю при незавантаженої аватарке	1) Заповнити поля введення, залишити без зображення 2) Натиснути «Зберегти настройки»	Застосунок видасть помилку про необхідність додати фото	✓
	Перевірка настройки профілю з порожніми полями	1) Завантажити фото, залишити будь-яке поле введення порожнім 2) Натиснути «Зберегти настройки»	Застосунок видасть помилку про необхідність заповнити поля	✓
Відкрити свій профіль	Перевірка відкриття профілю	На меню навігація нажати профіль	Відкриється вікно	✓
Відкрити список оголошень	Перевірити кнопку «список оголошень»	Нажати на кнопку в меню-навігації	Відкриється список оголошень	✓
	Перевірка настройки прокрутки вікна	Провести пальцем по знизу-вгору по головному екрану	Список перекинеться у низ	✓

Функціонал	Test-case	Кроки	Очікуваний результат	Результат тестування
Відкрити профіль автора	Перевірка відкрити вікно профілю	1) На будь-якому елементі списку оголошення, натиснути на зображення автора	Відкриється профіль автора	✓
Відмітити автора	Перевірка кнопки додати відміток	Натиснути на кнопку «Молодець», у відкритому профілі іншого користувача	Кнопка змінить свій зовнішній вигляд, а значення справа зміниться на +1	✓
	Перевірка кнопки видалити відмітку	Натиснути на кнопку повторно на кнопку «Молодець», у відкритому профілі іншого користувача	Кнопка змінить свій зовнішній вигляд, а значення справа зміниться на -1	✓
Поставити похвально оголошенню	Перевірка кнопки «Лайк»	Натиснути на кнопку «Лайк», у списку оголошень	Кнопка змінить свій зовнішній вигляд, а значення справа зміниться на +1	✓
Створити оголошення	Перевірка кнопки створити оголошення	Натиснути кнопку «Створити	Відкриється екран створення оголошення	✓
	Перевірка створення оголошення без фотографії	Заповнити поля введення	Застосунок видасть помилку про необхідність додати зображення	✓
	Перевірка створення оголошення з порожніми полями	1) Натиснути на «Виберіть фото» і завантажити зображення 2) Будь-яке поле введення залишити порожнім	Застосунок видасть помилку про необхідність заповнити всі поля	✓
Відкрити відгуки	Перевірка кнопки відкрити відгук	Натиснути кнопку вилучення на будь-якому елементі списку оголошення	Відкриється вікно відгуків	✓

Функціонал	Test-case	Кроки	Очікуваний результат	Результат тестування
	Перевірка кнопка відкрити відгук	Натиснути кнопку вугуку у відкритому оголошені	Відкриється вікно відгуків	✓
Створити відгук	Перевірка введення коментаря без з порожнім полем введення	1) Залишити поле введення порожнім 2) Натиснути на стрілку праворуч від поля введення	Застосунок видасть помилку про необхідність ввести текст в поле введення	✓
	Перевірка введення коментаря з числовими значення	1) Ввести текст у поле введення 2) Натиснути на стрілку праворуч від поля введення	Поле введення стане порожнім, а на екрані з'явиться коментар із зображенням і ім'ям автора	✓
Відкрити оголошення	Перевірка функціоналу	У списку оголошень натиснути на зображення	Відкриється екран оголошення	✓

Як видно з табл. 3.1, результати роботи спроектованого та розробленого мобільного застосунку співпадають з очікуваними.

ВИСНОВКИ

Метою даної ВКР було створення мобільного застосунку орієнтованого на поширення оголошень про обмін та продаж медикаментів.

У роботі був проведений аналіз існуючих конкурентів і предметної області, порівняльний аналіз і створення вимог до проекту. У пункті реалізації були встановлені вимоги до мобільного застосунку, аргументовано вибір середовища розробки. Застосунок було прийнято реалізувати в відомій середовищі розробки Android Studio мовою програмування Java, під операційну систему Android. На обраній нереляційній базі даних Firebase було створене сховище для даних, що надходять від користувачів.

Була описана структура і функціоналу доступного користувачу. Описано використовувані в проекті додаткові бібліотеки і їх призначення в застосунку. У застосунку була реалізована реєстрація, можливість додавати, коментувати оголошення і налаштувати дані профілю.

Описано принцип роботи з застосунком і перевірено його функціонування, яке результати якого співпадають з очікуваними.

Надалі можливий розвиток застосунку, для збільшення його функціоналу:

- реалізація категорій і підрозділів ліків і медичних засобів, для більш зручного пошуку;
- пошукова система, для пошуку оголошень за ключовими словами в заголовках і поділі області пошуку на регіони і міста;
- чат для спілкування між продавцем і покупцем.

Розроблений мобільний застосунок може в майбутньому стати зручним середовищем для оголошень не тільки ліків, але і як повноцінна дошка оголошень, для продажу і обміну різними речами і продуктами не тільки медичного призначення.

ДОДАТКИ

Додаток А. Програмний код

```
AccountAdRecycleAdapter.java
package com.example.medshering;
import android.content.Context;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.RecyclerView;
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.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;
import java.util.List;
import de.hdodenhof.circleimageview.CircleImageView;
import static android.app.PendingIntent.getActivity;
import static com.example.medshering.MainActivity.adPostFragment;
public class AccountAdRecycleAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public List<AdPost> userAdList;
    private FirebaseAuth mAuth;
    private FirebaseFirestore mFirestore;
    private String currUserId;
    public Context context;
    public AccountAdRecycleAdapter(List<AdPost> userAdList) {
        this.userAdList = userAdList;
    }
    @Override
    public int getItemViewType(int position) {
        if (position == 0)
            return 0;
        return 1;
    }
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        context = parent.getContext();
        mFirestore = FirebaseFirestore.getInstance();
        mAuth = FirebaseAuth.getInstance();
        currUserId = mAuth.getCurrentUser().getUid();
```

```

        View viewAd =
LayoutInflater.from(parent.getContext()).inflate(R.layout.account_ad_list_item,
parent, false);
        View viewUser =
LayoutInflater.from(parent.getContext()).inflate(R.layout.account_user_data_item,
parent, false);
        switch (viewType) {
            case 0: return new ViewHolderUserData(viewUser);
            case 1: return new ViewHolderAd(viewAd);
            default: return new ViewHolderAd(viewAd);
        }

    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int
position) {
        holder.setIsRecyclable(false);

        switch (holder.getItemViewType()) {
            case 0:
                final ViewHolderUserData viewHolderUserData = (ViewHolderUserData)
holder;
                DocumentReference userDocRef =
mFirestore.collection("Users").document(currUserId);
                userDocRef.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
                    @Override
                    public void onSuccess(DocumentSnapshot documentSnapshot) {
                        User user = documentSnapshot.toObject(User.class);
                        String firstName = user.getFirst();
                        String lastName = user.getLast();
                        String profileImageUrl = user.getImage();
                        String bio = user.getBio();
                        viewHolderUserData.setUserData(firstName, lastName,
profileImageUrl, bio);
                    }
                });
                viewHolderUserData.setUserReadersCounter(Integer.toString(0));
                Query readerQuery = mFirestore.collection("Users/" + currUserId +
"/Readers");
                readerQuery.addSnapshotListener(new EventListener<QuerySnapshot>()
{
                    @Override
                    public void onEvent(QuerySnapshot documentSnapshots,
FirebaseFirestoreException e) {
                        if (!documentSnapshots.isEmpty()) {
                            int counter = 0;
                            counter = documentSnapshots.size();
                        }
                        viewHolderUserData.setUserReadersCounter(Integer.toString(counter));
                    }
                });
                break;
            case 1:
                ViewHolderAd viewHolder = (ViewHolderAd) holder;

                String titleString = userAdList.get(position - 1).getTitle();
                String descString = userAdList.get(position - 1).getDescription();
                String imageUrl = userAdList.get(position - 1).getImage();
                viewHolder.setaddata(titleString, descString, imageUrl);
                final String adPostId = userAdList.get(position).adPostId;
                viewHolder.adTitleView.setOnClickListener(new

```

```

View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast.makeText(context, "Нажал на заголовок",
Toast.LENGTH_SHORT).show();

    }
});
break;
default:
    throw new IllegalStateException("Неожиданное значение: " +
holder.getItemViewType());
}
}
@Override
public int getItemCount() {
    return userAdList.size();
}
public class ViewHolderAd extends RecyclerView.ViewHolder {
    private View mView;
    private TextView adTitleView;
    private ImageView adImageView;
    private TextView adDescView;
    public ViewHolderAd(View itemView) {
        super(itemView);
        mView = itemView;
    }

    public void setaddata(String titleString, String descString, String
imageUrl) {
        adTitleView = mView.findViewById(R.id.acc_ad_title);
        adTitleView.setText(titleString);
        adDescView = mView.findViewById(R.id.acc_ad_desc);
        adDescView.setText(descString);
        adImageView = mView.findViewById(R.id.acc_ad_image);
        RequestOptions placeholderOption = new RequestOptions();
        placeholderOption.placeholder(R.drawable.rectangle);

        Glide.with(context).applyDefaultRequestOptions(placeholderOption).load(imageUrl).i
nto(adImageView);
    }

}
public class ViewHolderUserData extends RecyclerView.ViewHolder {
    private View mView;
    private CircleImageView userImageView;
    private TextView userNameView;
    private TextView userBioView;
    private TextView userReadersCounter;
    public ViewHolderUserData(View itemView) {
        super(itemView);
        mView = itemView;
    }

    public void setUserData(String firstName, String lastName, String
profileImageUrl, String bio) {
        userNameView = mView.findViewById(R.id.account_user_name);
        userNameView.setText(firstName + " " + lastName);
        userImageView = mView.findViewById(R.id.account_user_image);
        RequestOptions placeholderOption = new RequestOptions();
        placeholderOption.placeholder(R.drawable.ellipse);

        Glide.with(context).applyDefaultRequestOptions(placeholderOption).load(profileImag
eUrl).into(userImageView);
    }
}

```

```

        userBioView = mView.findViewById(R.id.account_user_bio);
        userBioView.setText(bio);
    }
    public void setUserReadersCounter(String counterString) {
        userReadersCounter =
mView.findViewById(R.id.account_user_readers_counter);
        userReadersCounter.setText(counterString);
    }
}
    public void goToAdFragment(String adPostId, String user_id, View view) {
        Bundle args = new Bundle();
        args.putString("adPostId", adPostId);
        args.putString("userId", user_id);
        AppCompatActivity activity = (AppCompatActivity) view.getContext();
        adPostFragment.setArguments(args);

activity.getSupportFragmentManager().beginTransaction().replace(R.id.main_containe
r, adPostFragment).addToBackStack(null).commit();
    }

}

```

AccountFragment.java

```

package com.example.medshering;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;
import java.util.ArrayList;
import java.util.List;
import javax.annotation.Nullable;
public class AccountFragment extends Fragment {
    FirebaseAuth mAuth;
    FirebaseFirestore mFirestore;
    String currUserId;
    // recycler adapter
    RecyclerView userAdListView;
    List<AdPost> userAdList;
    AccountAdRecycleAdapter accountAdRecycleAdapter;
    public AccountFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_account, container, false);
        mAuth = FirebaseAuth.getInstance();
        mFirestore = FirebaseFirestore.getInstance();
        currUserId = mAuth.getCurrentUser().getUid();
        userAdListView = view.findViewById(R.id.user_ad_list);
        userAdList = new ArrayList<>();
        accountAdRecycleAdapter = new AccountAdRecycleAdapter(userAdList);
        userAdListView.setLayoutManager(new

```

```

LinearLayoutManager(container.getContext()));
        userAdListView.setAdapter(accountAdRecycleAdapter);
        Query accountQuery =
mFirestore.collection("Posts").whereEqualTo("user_id", currUserId);
        //.orderBy("timestamp", Query.Direction.DESENDING);
        accountQuery.addSnapshotListener(getActivity(), new
EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots,
@Nullable FirebaseFirestoreException e) {
                for (DocumentChange doc :
queryDocumentSnapshots.getDocumentChanges()) {
                    if (doc.getType() == DocumentChange.Type.ADDED) {
                        AdPost adPost = doc.getDocument().toObject(AdPost.class);
                        userAdList.add(adPost);
                        accountAdRecycleAdapter.notifyDataSetChanged();
                    }
                }
            }
        });
        return view;
    }
}

```

AdPost.java

```

package com.example.medshering;
import java.util.Date;
public class AdPost extends AdPostId {
    public String
        user_id
        , title
        , description
        , content
        , image
        , thumb;

    public Date timestamp;
    public AdPost() {}
    public AdPost(String user_id, String title, String description, String
content, String image, String thumb, Date timestamp) {
        this.user_id = user_id;
        this.title = title;
        this.description = description;
        this.content = content;
        this.image = image;
        this.thumb = thumb;
        this.timestamp = timestamp;
    }
    public String getUser_id() {
        return user_id;
    }
    public void setUser_id(String user_id) {
        this.user_id = user_id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDescription() {
        return description;
    }
}

```



```

    public void setDescription(String description) {
        this.description = description;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getImage() {
        return image;
    }
    public void setImage(String image) {
        this.image = image;
    }
    public String getThumb() {
        return thumb;
    }
    public void setThumb(String thumb) {
        this.thumb = thumb;
    }
    public Date getTimestamp() {
        return timestamp;
    }
    public void setTimestamp(Date timestamp) {
        this.timestamp = timestamp;
    }
}

AdPostId.java
package com.example.medshering;

import java.util.Date;

public class AdPost extends AdPostId {
    public String
        user_id
        , title
        , description
        , content
        , image
        , thumb;

    public Date timestamp;

    public AdPost() {}

    public AdPost(String user_id, String title, String description, String
content, String image, String thumb, Date timestamp) {
        this.user_id = user_id;
        this.title = title;
        this.description = description;
        this.content = content;
        this.image = image;
        this.thumb = thumb;
        this.timestamp = timestamp;
    }
    public String getUser_id() {
        return user_id;
    }
    public void setUser_id(String user_id) {

```

```

        this.user_id = user_id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }

    public String getImage() {
        return image;
    }
    public void setImage(String image) {
        this.image = image;
    }
    public String getThumb() {
        return thumb;
    }
    public void setThumb(String thumb) {
        this.thumb = thumb;
    }
    public Date getTimestamp() {
        return timestamp;
    }
    public void setTimestamp(Date timestamp) {
        this.timestamp = timestamp;
    }
}

}

AdRecycleAdapter.java
package com.example.medshering;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.RecyclerView;
import android.text.format.DateFormat;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import com.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.google.android.gms.tasks.OnCompleteListener;

```

```

import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.annotation.Nullable;
import de.hdodenhof.circleimageview.CircleImageView;
import static com.example.medshearing.MainActivity.adPostFragment;
public class AdRecycleAdapter extends
RecyclerView.Adapter<AdRecycleAdapter.ViewHolder> {
    public List<AdPost> adList;
    public List<User> userList;
    public Context context;
    private FirebaseFirestore mFirestore;
    private FirebaseAuth mAuth;
    Boolean currUserLiked = false;
    public class ViewHolder extends RecyclerView.ViewHolder {
        private final View mView;
        private final TextView titleView;
        private ImageView imageView;
        private TextView userNameView;
        private CircleImageView userProfileImageView;
        private final ImageView likeBtn;
        private final ImageView commBtn;

        public ViewHolder(View itemView) {
            super(itemView);
            mView = itemView;

            commBtn = mView.findViewById(R.id.ad_comm_btn);
            likeBtn = mView.findViewById(R.id.add_like_btn);
            titleView = mView.findViewById(R.id.acc_ad_title);
        }
        public void setDescText(String descString) {
            TextView descView = mView.findViewById(R.id.ad_desc);
            descView.setText(descString);
        }
        public void setTitleText(String titleString) {
            titleView.setText(titleString);
        }
        public void setAdImage(String imageUrl, String thumbnailUrl) {
            ImageView = mView.findViewById(R.id.ad_image);
            RequestOptions placeholderOption = new RequestOptions();
            placeholderOption.placeholder(R.drawable.rectangle);
            Glide.with(context).applyDefaultRequestOptions(placeholderOption)
                .load(imageUrl)
                .thumbnail(Glide.with(context).load(thumbnailUrl))
                .into(ImageView);
        }

        public void setTime(String dateString) {
            TextView dateView = mView.findViewById(R.id.ad_date);
            dateView.setText(dateString);
        }
    }
}

```

```

    }
    @SuppressWarnings("CheckResult")
    public void setUserData(String firstName, String lastName, String
profileImageUrl) {
        userNameView = mView.findViewById(R.id.ad_user_name);
        userProfileImageView = mView.findViewById(R.id.ad_user_image);
        RequestOptions placeholderOption = new RequestOptions();
        placeholderOption.placeholder(R.drawable.ellipse);

        Glide.with(context).applyDefaultRequestOptions(placeholderOption).load(profileImage
eUrl).into(userProfileImageView);

        userNameView.setText(firstName + " " + lastName);
    }
    public void updateLikeCounter(int counter) {
        TextView likeCounter = mView.findViewById(R.id.ad_like_count);
        if (counter > 0)
            likeCounter.setText(Integer.toString(counter));
        else
            likeCounter.setText(" ");
    }
}

public void goToAdPostFragment(String adPostId, String user_id, View view) {
    Bundle args = new Bundle();
    args.putString("adPostId", adPostId);
    args.putString("userId", user_id);
    AppCompatActivity activity = (AppCompatActivity) view.getContext();
    //PostFragment blogPostFragment = new PostFragment();
    adPostFragment.setArguments(args);

    activity.getSupportFragmentManager().beginTransaction().replace(R.id.main_containe
r, adPostFragment).addToBackStack(null).commit();
}

public void goToUserProfile(String user_id, View view) {
    Bundle args = new Bundle();
    args.putString("userId", user_id);
    AppCompatActivity activity = (AppCompatActivity) view.getContext();
    AnotherAccountFragment anotherAccountFragment = new
AnotherAccountFragment();
    anotherAccountFragment.setArguments(args);

    activity.getSupportFragmentManager().beginTransaction().replace(R.id.main_containe
r, anotherAccountFragment).addToBackStack(null).commit();

}

public AdRecycleAdapter(List<AdPost> adList, List<User> userList) {
    this.adList = adList;
    this.userList = userList;
}

@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
    View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.ad_list_item, parent,
false);
    context = parent.getContext();
    mFirestore = FirebaseFirestore.getInstance();
    mAuth = FirebaseAuth.getInstance();
    return new ViewHolder(view);
}

```

```

@Override
public void onBindViewHolder(@NonNull final ViewHolder holder, int position) {
    holder.setIsRecyclable(false);
    final String currentUserId = mAuth.getCurrentUser().getUid();
    final String adPostId = adList.get(position).adPostId;
    String desc_data = adList.get(position).getDescription();
    holder.setDescText(desc_data);
    String title_data = adList.get(position).getTitle();
    holder.setTitleText(title_data);
    String image_url = adList.get(position).getImage();
    String thumbnail_url = adList.get(position).getThumb();
    holder.setAdImage(image_url, thumbnail_url);
    try {
        long millisecond = adList.get(position).getTimestamp().getTime();
        String dateString = DateFormat.format("d/MM/yyyy", new
Date(millisecond)).toString();
        holder.setTime(dateString);
    } catch (Exception e) {
        Toast.makeText(context, "Exception : " + e.getMessage(),
Toast.LENGTH_SHORT).show();
    }
    final String user_id = adList.get(position).getUser_id();
    if (userList.get(position) != null) {
        final String firstName = userList.get(position).getFirst();
        final String lastName = userList.get(position).getLast();
        String profileImageUrl = userList.get(position).getImage();

        holder.setUserData(firstName, lastName, profileImageUrl);
    }

    mFirestore.collection("Posts/" + adPostId +
"/Likes").addSnapshotListener(new EventListener<QuerySnapshot>() {
        @Override
        public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots,
@Nullable FirebaseFirestoreException e) {

            if (!queryDocumentSnapshots.isEmpty()) {
                int counter = queryDocumentSnapshots.size();
                holder.updateLikeCounter(counter);
            } else {
                holder.updateLikeCounter(0);
            }
        }
    });

    mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).addSnapshotListener(new
EventListener<DocumentSnapshot>() {
        @Override
        public void onEvent(@Nullable DocumentSnapshot documentSnapshot,
@Nullable FirebaseFirestoreException e) {
            if (documentSnapshot.exists()) {

holder.likeBtn.setImageDrawable(context.getDrawable(R.mipmap.action_like_btn_accen
t));

                } else {

holder.likeBtn.setImageDrawable(context.getDrawable(R.mipmap.action_like_btn_gray
));

                }

            }
        });
    holder.likeBtn.setOnClickListener(new OnClickListener() {
        @Override

```

```

        public void onClick(View view) {
            currUserLiked = !currUserLiked;
            mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                    if(!task.getResult().exists()){
                        Map<String, Object> likesMap = new HashMap<>();
                        likesMap.put("timestamp",
FieldValue.serverTimestamp());

                        mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).set(likesMap);
                    } else {
                        mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).delete();
                    }
                }
            });
        }
    });
    holder.commBtn.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent commentIntent = new Intent(context,
CommentsActivity.class);
            commentIntent.putExtra("adPostId", adPostId);
            context.startActivity(commentIntent);
        }
    });
    holder.titleView.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            goToAdPostFragment(adPostId, user_id, view);
        }
    });
    holder.imageView.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            goToAdPostFragment(adPostId, user_id, view);
        }
    });
    if (holder.userNameView !=null) {
        holder.userNameView.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                //Toast.makeText(context, firstName + " " + lastName,
Toast.LENGTH_SHORT).show();
                goToUserProfile(user_id, view);
            }
        });
    }
    if (holder.userProfileImageView != null) {
        holder.userProfileImageView.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                goToUserProfile(user_id, view);
            }
        });
    }
}

```

```

    }

    @Override
    public int getItemCount() {
        return adList.size();
    }
}

AnotherAccountAdRecycleAdapter.java
package com.example.medshering;
import android.content.Context;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.RecyclerView;
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.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import de.hdodenhof.circleimageview.CircleImageView;
import static android.app.PendingIntent.getActivity;
import static com.example.medshering.MainActivity.adPostFragment;
public class AnotherAccountAdRecycleAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public List<AdPost> userAdList;
    private FirebaseAuth mAuth;
    private FirebaseFirestore mFirestore;
    private String anotherUserId;
    private String currUserId;
    public Context context;
    int counterGlobal;
    public AnotherAccountAdRecycleAdapter(List<AdPost> userAdList) {
        this.userAdList = userAdList;
    }
    @Override
    public int getItemViewType(int position) {
        if (position == 0)
            return 0;
        return 1;
    }
}

@NonNull

```

```

@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
    context = parent.getContext();
    mFirestore = FirebaseFirestore.getInstance();
    mAuth = FirebaseAuth.getInstance();
    currUserId = mAuth.getCurrentUser().getUid();
    View viewAd =
LayoutInflater.from(parent.getContext()).inflate(R.layout.account_ad_list_item,
parent, false);
    View viewUser =
LayoutInflater.from(parent.getContext()).inflate(R.layout.another_account_user_dat
a_item, parent, false);
    switch (viewType) {
        case 0: return new
AnotherAccountAdRecycleAdapter.ViewHolderUserData(viewUser);
        case 1: return new ViewHolder(viewAd);
        default: return new ViewHolder(viewAd);
    }
}

@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int
position) {
    holder.setIsRecyclable(false);
    anotherUserId = userAdList.get(position).getUser_id();
    switch (holder.getItemViewType()) {
        case 0:
            final AnotherAccountAdRecycleAdapter.ViewHolderUserData
viewHolderUserData = (AnotherAccountAdRecycleAdapter.ViewHolderUserData) holder;

            final DocumentReference userDocRef =
mFirestore.collection("Users").document(anotherUserId);
            userDocRef.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
                @Override
                public void onSuccess(DocumentSnapshot documentSnapshot) {
                    User user = documentSnapshot.toObject(User.class);
                    String firstName = user.getFirst();
                    String lastName = user.getLast();
                    String profileImageUrl = user.getImage();
                    String bio = user.getBio();

                    viewHolderUserData.setUserData(firstName, lastName,
profileImageUrl, bio);
                }
            });
            viewHolderUserData.setUserReadersCounter(Integer.toString(0));
            Query readerQuery = mFirestore.collection("Users/" + anotherUserId
+ "/Readers");

            readerQuery.addSnapshotListener(new EventListener<QuerySnapshot>()
{
                @Override
                public void onEvent(QuerySnapshot documentSnapshots,
FirebaseFirestoreException e) {

                    if (!documentSnapshots.isEmpty()) {

                        int counter = 0;
                        counter = documentSnapshots.size();
                        counterGlobal = counter;
                        //Toast.makeText(context, Integer.toString(counter),

```



```

Toast.LENGTH_SHORT).show();

viewHolderUserData.setUserReadersCounter(Integer.toString(counter));
    }
    });
    mFirestore.collection("Users/" + anotherUserId +
"/Readers").document(currUserId).get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if(!task.getResult().exists()){
        } else {

viewHolderUserData.userReadBtn.setImageResource(R.mipmap.action_remove_reader_btn_
accent);

        }

    }
});

viewHolderUserData.userReadBtn.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Toast.makeText(context, "read button",
Toast.LENGTH_SHORT).show();
        mFirestore.collection("Users/" + anotherUserId +
"/Readers").document(currUserId).get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot>
task) {
                if(!task.getResult().exists()){
                    Map<String, Object> readMap = new HashMap<>();
                    readMap.put("user_id", currUserId);
                    readMap.put("timestamp",
FieldValue.serverTimestamp());
                    mFirestore.collection("Users/" + anotherUserId
+ "/Readers").document(currUserId).set(readMap);

viewHolderUserData.userReadBtn.setImageResource(R.mipmap.action_remove_reader_btn_
accent);

                } else {

                    mFirestore.collection("Users/" + anotherUserId
+ "/Readers").document(currUserId).delete();

viewHolderUserData.setUserReadersCounter(Integer.toString(counterGlobal - 1));

viewHolderUserData.userReadBtn.setImageResource(R.mipmap.action_add_reader_btn_acc
ent);

                }

            }
        });
    }
});
break;
case 1:

```

```

ViewHolder viewHolderAd = (ViewHolder) holder;
String titleString = userAdList.get(position - 1).getTitle();
String descString = userAdList.get(position - 1).getDescription();
String imageUrl = userAdList.get(position - 1).getImage();
viewHolderAd.setAdData(titleString, descString, imageUrl);
final String adPostId = userAdList.get(position).adPostId;

viewHolderAd.adTitleView.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast.makeText(context, "Am apasat pe titlu",
Toast.LENGTH_SHORT).show();
        //goToAdFragment(blogPostId, currUserId, view);
    }
});

break;
}
}
@Override
public int getItemCount() {
    return userAdList.size();
}
}
public class ViewHolder extends RecyclerView.ViewHolder {

    private View mView;
    private TextView adTitleView;
    private ImageView adImageView;
    private TextView adDescView;
    public ViewHolder(View itemView) {
        super(itemView);
        mView = itemView;
    }
    public void setAdData(String titleString, String descString, String
imageUrl) {
        adTitleView = mView.findViewById(R.id.acc_ad_title);
        adTitleView.setText(titleString);

        adDescView = mView.findViewById(R.id.acc_ad_desc);
        adDescView.setText(descString);

        adImageView = mView.findViewById(R.id.acc_ad_image);
        RequestOptions placeholderOption = new RequestOptions();
        placeholderOption.placeholder(R.drawable.rectangle);

        Glide.with(context).applyDefaultRequestOptions(placeholderOption).load(imageUrl).i
nto(adImageView);
    }
}
}
public class ViewHolderUserData extends RecyclerView.ViewHolder {
    private View mView;
    private CircleImageView userImageView;
    private TextView userNameView;
    private TextView userBioView;
    private TextView userReadersCounter;
    private TextView userReadBtn;
    public ViewHolderUserData(View itemView) {
        super(itemView);
        mView = itemView;
    }
}

```

```

        userReadBtn = mView.findViewById(R.id.account_user_read_btn);
    }
    public void setUserData(String firstName, String lastName, String
profileImageUrl, String bio) {
        userNameView = mView.findViewById(R.id.account_user_name);
        userNameView.setText(firstName + " " + lastName);
        userImageView = mView.findViewById(R.id.account_user_image);
        RequestOptions placeholderOption = new RequestOptions();
        placeholderOption.placeholder(R.drawable.ellipse);

Glide.with(context).applyDefaultRequestOptions(placeholderOption).load(profileImag
eUrl).into(userImageView);
        userBioView = mView.findViewById(R.id.account_user_bio);
        userBioView.setText(bio);

    }

    public void setUserReadersCounter(String counterString) {
        userReadersCounter =
mView.findViewById(R.id.account_user_readers_counter);
        userReadersCounter.setText(counterString);
    }
}
    public void goToAdPostFragment(String adPostId, String user_id, View view) {
        Bundle args = new Bundle();
        args.putString("adPostId", adPostId);
        args.putString("userId", user_id);

        AppCompatActivity activity = (AppCompatActivity) view.getContext();
        adPostFragment.setArguments(args);

activity.getSupportFragmentManager().beginTransaction().replace(R.id.main_containe
r, adPostFragment).addToBackStack(null).commit();
    }

}

```

```

AnotherAccountFragment.java
package com.example.medshering;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;
import java.util.ArrayList;
import java.util.List;
import javax.annotation.Nullable;

public class AnotherAccountFragment extends Fragment {

    FirebaseAuth mAuth;
    FirebaseFirestore mFirestore;
    String userId;
    RecyclerView userAdListView;

```

```

List<AdPost> userAdList;
AnotherAccountAdRecycleAdapter anotherAccountAdRecycleAdapter;
public AnotherAccountFragment() {

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

        mAuth = FirebaseAuth.getInstance();
        mFirestore = FirebaseFirestore.getInstance();
        userId = getArguments().getString("userId");

        userAdListView = view.findViewById(R.id.another_user_ad_list);
        userAdList = new ArrayList<>();
        anotherAccountAdRecycleAdapter = new
        AnotherAccountAdRecycleAdapter(userAdList);

        userAdListView.setLayoutManager(new
        LinearLayoutManager(container.getContext()));
        userAdListView.setAdapter(anotherAccountAdRecycleAdapter);

        Query accountQuery =
        mFirestore.collection("Posts").whereEqualTo("user_id", userId);
        accountQuery.addSnapshotListener(getActivity(), new
        EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots,
            @Nullable FirebaseFirestoreException e) {
                for (DocumentChange doc :
                queryDocumentSnapshots.getDocumentChanges()) {

                    if (doc.getType() == DocumentChange.Type.ADDED) {

                        AdPost adPost = doc.getDocument().toObject(AdPost.class);
                        userAdList.add(adPost);
                        anotherAccountAdRecycleAdapter.notifyDataSetChanged();

                    }
                }
            }
        });

        return view;
    }
}

Comments.java
package com.example.medshering;
import java.util.Date;
public class Comments {
    private String message, user_id;
    private Date timestamp;

    public Comments(){

    }

    public Comments(String message, String user_id, Date timestamp) {

```

```

        this.message = message;
        this.user_id = user_id;
        this.timestamp = timestamp;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getUser_id() {
        return user_id;
    }

    public void setUser_id(String user_id) {
        this.user_id = user_id;
    }

    public Date getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(Date timestamp) {
        this.timestamp = timestamp;
    }
}

CommentsActivity.java
package com.example.medshering;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CommentsActivity extends AppCompatActivity {
    private Toolbar commentToolbar;
    private EditText comment_field;
    private ImageView comment_post_btn;
    private RecyclerView comment_list;

```

```

private CommentsRecyclerAdapter commentsRecyclerAdapter;
private List<Comments> commentsList;
private FirebaseFirestore firebaseFirestore;
private FirebaseAuth firebaseAuth;
private String ad_post_id;
private String current_user_id;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_comments);
    commentToolBar = findViewById(R.id.comment_toolbar);
    setSupportActionBar(commentToolBar);
    getSupportActionBar().setTitle("Комментарии");
    firebaseAuth = FirebaseAuth.getInstance();
    firebaseFirestore = FirebaseFirestore.getInstance();
    current_user_id = firebaseAuth.getCurrentUser().getUid();
    ad_post_id = getIntent().getStringExtra("adPostId");
    comment_field = findViewById(R.id.comment_field);
    comment_post_btn = findViewById(R.id.comment_post_btn);
    comment_list = findViewById(R.id.comment_list);
    commentsList = new ArrayList<>();
    commentsRecyclerAdapter = new CommentsRecyclerAdapter(commentsList);
    comment_list.setHasFixedSize(true);
    comment_list.setLayoutManager(new LinearLayoutManager(this));
    comment_list.setAdapter(commentsRecyclerAdapter);
    Query query = firebaseFirestore.collection("Posts/" + ad_post_id +
"/Comments").orderBy("timestamp", Query.Direction.ASCENDING);

    query.addSnapshotListener(CommentsActivity.this, new
EventListener<QuerySnapshot>() {
        @Override
        public void onEvent(QuerySnapshot documentSnapshots,
FirebaseFirestoreException e) {
            if (!documentSnapshots.isEmpty()) {
                for (DocumentChange doc :
documentSnapshots.getDocumentChanges()) {
                    if (doc.getType() == DocumentChange.Type.ADDED) {

                        String commentId = doc.getDocument().getId();
                        Comments comments =
doc.getDocument().toObject(Comments.class);
                        commentsList.add(comments);
                        commentsRecyclerAdapter.notifyDataSetChanged();
                    }
                }
            }
        }
    });
    comment_post_btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String comment_message = comment_field.getText().toString();
            Map<String, Object> commentsMap = new HashMap<>();
            commentsMap.put("message", comment_message);
            commentsMap.put("user_id", current_user_id);
            commentsMap.put("timestamp", FieldValue.serverTimestamp());
            firebaseFirestore.collection("Posts/" + ad_post_id +
"/Comments").add(commentsMap).addOnCompleteListener(new
OnCompleteListener<DocumentReference>() {
                @Override
                public void onComplete(@NonNull Task<DocumentReference> task)
{

```



```

        holder.setComment_message(commentMessage);
        String user_id = commentsList.get(position).getUser_id();
        mFirestore.collection("Users").document(user_id).get().addOnCompleteListener(new
        OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {

                if (task.isSuccessful()) {
                    String firstName = task.getResult().getString("first");
                    String lastName = task.getResult().getString("last");
                    String profileImageUrl = task.getResult().getString("image");

                    holder.setUserData(firstName, lastName, profileImageUrl);
                } else {
                    //
                }
            }
        });
    }
    @Override
    public int getItemCount() {

        if(commentsList != null) {

            return commentsList.size();

        } else {

            return 0;

        }

    }
}
public class ViewHolder extends RecyclerView.ViewHolder {
    private View mView;
    private TextView comment_message;
    private CircleImageView comment_image;
    private TextView comment_username;
    public ViewHolder(View itemView) {
        super(itemView);
        mView = itemView;
    }
    public void setComment_message(String message) {

        comment_message = mView.findViewById(R.id.comment_message);
        comment_message.setText(message);
    }
    public void setUserData(String firstName, String lastName, String
    profileImageUrl) {
        comment_image = mView.findViewById(R.id.comment_image);
        comment_username = mView.findViewById(R.id.comment_username);
        comment_username.setText(firstName + " " + lastName);

        RequestOptions placeholderOption = new RequestOptions();
        placeholderOption.placeholder(R.drawable.ellipse);

        Glide.with(context).applyDefaultRequestOptions(placeholderOption).load(profileImage
        eUrl).into(comment_image);

    }
}

```



```

}

HomeFragment.java
package com.example.medshering;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QuerySnapshot;
import java.util.ArrayList;
import java.util.List;
import javax.annotation.Nullable;

public class HomeFragment extends Fragment {
    FirebaseFirestore mFirestore;
    FirebaseAuth mAuth;

    RecyclerView homeListView;
    List<AdPost> adList;
    AdRecycleAdapter adRecycleAdapter;
    List<User> userList;

    DocumentSnapshot lastVisible;
    Boolean isFirstPageFirstLoaded = true;

    public HomeFragment() {
        // Required empty public constructor
    }

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

        mAuth = FirebaseAuth.getInstance();

        homeListView = view.findViewById(R.id.home_ad_list);
        adList = new ArrayList<>();
        userList = new ArrayList<>();

        adRecycleAdapter = new AdRecycleAdapter(adList, userList);

        homeListView.setLayoutManager(new
LinearLayoutManager(container.getContext()));
        homeListView.setAdapter(adRecycleAdapter);

        if (mAuth.getCurrentUser() != null) {
            mFirestore = FirebaseFirestore.getInstance();

```

```

homeListView.addOnScrollListener(new RecyclerView.OnScrollListener() {
    @Override
    public void onScrolled(RecyclerView recyclerView, int dx, int dy)
    {
        super.onScrolled(recyclerView, dx, dy);

        Boolean reachedBottom = !recyclerView.canScrollVertically(1);
        if (reachedBottom) {
            loadMorePost();
        }
    }
});

Query firstQuery = mFirestore.collection("Posts").orderBy("timestamp",
Query.Direction.DESENDING).limit(3);

firstQuery.addSnapshotListener(getActivity(), new
EventListener<QuerySnapshot>() {
    @Override
    public void onEvent(@Nullable QuerySnapshot
queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {

        if (isFirstPageFirstLoaded) {
            lastVisible =
queryDocumentSnapshots.getDocuments().get(queryDocumentSnapshots.size() - 1);
            adList.clear();
            userList.clear();
        }

        for (DocumentChange doc :
queryDocumentSnapshots.getDocumentChanges()) {
            if (doc.getType() == DocumentChange.Type.ADDED) {

                String adPostId = doc.getDocument().getId();
                final AdPost adPost =
doc.getDocument().toObject(AdPost.class).withId(adPostId);

                String adUserId =
doc.getDocument().getString("user_id");

mFirestore.collection("Users").document(adUserId).get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull
Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            User user =
task.getResult().toObject(User.class);

            if (isFirstPageFirstLoaded) {
                adList.add(adPost);
                userList.add(user);
            } else {
                adList.add(0, adPost);
                userList.add(0, user);
            }

            adRecycleAdapter.notifyDataSetChanged();
        }
    }
});
});

```

```

        }
        }
        isFirstPageFirstLoaded = false;
    }
    });
}
// Inflate the layout for this fragment
return view;
}

public void loadMorePost() {
    if (mAuth.getCurrentUser() != null) {

        Query nextQuery = mFirestore.collection("Posts")
            .orderBy("timestamp", Query.Direction.DESENDING)
            .startAfter(lastVisible)
            .limit(3);

        nextQuery.addSnapshotListener(getActivity(), new
        EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot
            queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {

                if (!queryDocumentSnapshots.isEmpty()) {

                    lastVisible =
                    queryDocumentSnapshots.getDocuments().get(queryDocumentSnapshots.size() - 1);

                    for (DocumentChange doc :
                    queryDocumentSnapshots.getDocumentChanges()) {
                        if (doc.getType() == DocumentChange.Type.ADDED) {

                            String adPostId = doc.getDocument().getId();
                            final AdPost adPost =
                            doc.getDocument().toObject(AdPost.class).withId(adPostId);

                            String adUserId =
                            doc.getDocument().getString("user_id");

                            mFirestore.collection("Users").document(adUserId).get().addOnCompleteListener(new
                            OnCompleteListener<DocumentSnapshot>() {
                                @Override
                                public void onComplete(@NonNull
                                Task<DocumentSnapshot> task) {

                                    if (task.isSuccessful()) {
                                        User user =

                                        adList.add(adPost);
                                        userList.add(user);

                                        adRecycleAdapter.notifyDataSetChanged();
                                    }
                                }
                            });
                        }
                    }
                }
            }
        });
    }
}

```

```

    }
    });
}
}

LoginActivity.java
package com.example.medshering;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class LoginActivity extends AppCompatActivity {

    EditText mEmail;
    EditText mPassword;
    Button mLoginBtn;
    FirebaseAuth mAuth;

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

        mAuth = FirebaseAuth.getInstance();
        mEmail = (EditText) findViewById(R.id.email_field);
        mPassword = (EditText) findViewById(R.id.password_field);
        mLoginBtn = (Button) findViewById(R.id.login_button);

        mLoginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String mInputEmail = mEmail.getText().toString();
                String mInputPass = mPassword.getText().toString();

                //если поля не пусты
                if (!TextUtils.isEmpty(mInputEmail) &&
!TextUtils.isEmpty(mInputPass)) {

                    mAuth.signInWithEmailAndPassword(mInputEmail,
mInputPass).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {

                            if (task.isSuccessful()) {
                                sendToMain();
                            } else {
                                String errorMessage =
task.getException().getMessage();
                                Toast.makeText(LoginActivity.this, "Error: " +
errorMessage, Toast.LENGTH_LONG).show();
                            }
                        }
                    });
                }
            }
        });
    }
}

```

```

        }
        });
    } else
        Toast.makeText(LoginActivity.this, "Все поля должны быть
заполнены", Toast.LENGTH_LONG).show();

    }
});

}

public void clickToRegister(View view) {
    //Toast.makeText(LoginActivity.this, "AI APASAT!",
Toast.LENGTH_LONG).show();
    // intent catre RegisterActivity
    startActivity(new Intent(this, RegisterActivity.class));
}

private void sendToMain() {
    Intent mainIntent = new Intent(LoginActivity.this, MainActivity.class);
    startActivity(mainIntent);
    finish();
}

}

}
MaintActivity.java
package com.example.medshering;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.internal.BottomNavigationView;
import android.support.design.internal.BottomNavigationView;
import android.support.design.widget.BottomNavigationView;
import android.support.v4.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import java.lang.reflect.Field;

class BottomNavigationViewHelper {

    @SuppressLint("RestrictedApi")
    static void removeShiftMode(BottomNavigationView view) {
        BottomNavigationView menuView = (BottomNavigationView)
view.getChildAt(0);
        try {
            Field shiftingMode =
menuView.getClass().getDeclaredField("mShiftingMode");
            shiftingMode.setAccessible(true);
            shiftingMode.setBoolean(menuView, false);
            shiftingMode.setAccessible(false);
            for (int i = 0; i < menuView.getChildCount(); i++) {
                BottomNavigationView item = (BottomNavigationView)

```

```

menuView.getChildAt(i);
            item.setShiftingMode(false);
            // set once again checked value, so view will be updated
            item.setChecked(item.getItemData().isChecked());
        }
    } catch (NoSuchFieldException e) {
        Log.e("ERROR NO SUCH FIELD", "Unable to get shift mode field");
    } catch (IllegalAccessException e) {
        Log.e("ERROR ILLEGAL ALG", "Unable to change value of shift mode");
    }
}

}

public class MainActivity extends AppCompatActivity {
    android.support.v7.widget.Toolbar mMainToolbar;
    FirebaseAuth mAuth;
    FirebaseFirestore mFirestore;
    public static String currentUserId;
    BottomNavigationView bottomNav;
    HomeFragment homeFragment;
    NotificationFragment notificationFragment;
    AccountFragment accountFragment;
    public static PostFragment adPostFragment;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mAuth = FirebaseAuth.getInstance();
        mFirestore = FirebaseFirestore.getInstance();

        FirebaseUser mCurrentUser = FirebaseAuth.getInstance().getCurrentUser();

        if (mCurrentUser != null) {
            mMainToolbar = (android.support.v7.widget.Toolbar)
findViewById(R.id.main_toolbar);
            setSupportActionBar(mMainToolbar);
            getSupportActionBar().setTitle("Med Sharing");
            getSupportActionBar().getThemedContext();

mMainToolbar.setTitleTextColor(getResources().getColor(R.color.colorPrimaryDark));
            bottomNav = findViewById(R.id.bottom_nav);
            BottomNavigationViewHelper.removeShiftMode(bottomNav);
            homeFragment = new HomeFragment();
            notificationFragment = new NotificationFragment();
            accountFragment = new AccountFragment();
            adPostFragment = new PostFragment();

            initializeFragment();

            bottomNav.setNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
                @Override
                public boolean onNavigationItemSelected(@NonNull MenuItem item) {

                    android.support.v4.app.Fragment currentFragment =
getSupportFragmentManager().findFragmentById(R.id.main_container);

                    switch (item.getItemId()) {
                        case R.id.bottom_home:
                            replaceFragment(homeFragment, currentFragment);
                            return true;
                        case R.id.bottom_notification:
                            replaceFragment(notificationFragment,
currentFragment);

```

```

        return true;
    case R.id.bottom_account:
        replaceFragment(accountFragment, currentFragment);
        return true;
    case R.id.bottom_add:
        sendToAddPost();
        return true;
    default:
        return false;
    }
    });
}

@Override
protected void onStart() {
    super.onStart();

    FirebaseUser mCurrentUser = FirebaseAuth.getInstance().getCurrentUser();
    if (mCurrentUser == null) {
        sendToLogin();
    } else {
        currentUserId = mAuth.getCurrentUser().getUid();

        mFirestore.collection("Users").document(currentUserId).get().addOnCompleteListener
        (new OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {

                if(task.isSuccessful()){
                    if(!task.getResult().exists()){
                        sendToSettings();
                        finish();
                    }
                } else {
                    String errorMessage = task.getException().getMessage();
                    Toast.makeText(MainActivity.this, "Error : " +
                    errorMessage, Toast.LENGTH_LONG).show();
                }
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_logout_btn) {
            logout();
        } else if (item.getItemId() == R.id.action_settings_btn) {
            sendToSettings();
        } else if (item.getItemId() == R.id.action_addpost_btn) {
            sendToAddPost();
        }
        return false;
    }
}

```

```

private void sendToAddPost() {
    Intent toNewPost = new Intent(MainActivity.this, NewPostActivity.class);
    startActivity(toNewPost);
}
private void sendToSettings() {
    Intent toSettings = new Intent(MainActivity.this, SetupActivity.class);
    startActivity(toSettings);
}

private void logOut() {
    mAuth.signOut();
    sendToLogin();
}

private void sendToLogin() {
    Intent toLogin = new Intent(MainActivity.this, LoginActivity.class);
    startActivity(toLogin);
    finish();
}

private void replaceFragment(android.support.v4.app.Fragment newFragment,
android.support.v4.app.Fragment currentFragment){

    FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
    if(newFragment == homeFragment){
        fragmentTransaction.hide(accountFragment);
        fragmentTransaction.hide(notificationFragment);
    }
    if(newFragment == accountFragment){
        fragmentTransaction.hide(homeFragment);
        fragmentTransaction.hide(notificationFragment);
    }
    if(newFragment == notificationFragment){
        fragmentTransaction.hide(homeFragment);
        fragmentTransaction.hide(accountFragment);
    }
    fragmentTransaction.show(newFragment);
    //fragmentTransaction.replace(R.id.main_container, fragment);
    fragmentTransaction.commit();
}
private void initializeFragment(){

    FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();

    fragmentTransaction.add(R.id.main_container, homeFragment);
    fragmentTransaction.add(R.id.main_container, notificationFragment);
    fragmentTransaction.add(R.id.main_container, accountFragment);

    fragmentTransaction.hide(notificationFragment);
    fragmentTransaction.hide(accountFragment);

    fragmentTransaction.commit();

}
}

```

```

NewPostActivity.java
package com.example.medshering;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Typeface;
import android.net.Uri;

```



```

import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.theartofdev.edmodo.cropper.CropImage;
import com.theartofdev.edmodo.cropper.CropImageView;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import id.zelory.compressor.Compressor;

public class NewPostActivity extends AppCompatActivity {

    android.support.v7.widget.Toolbar mToolbar;

    TextView mChoosePhoto;
    EditText mTitle;
    EditText mDesc;
    EditText mContent;
    Button mAddBtn;
    FirebaseAuth mAuth;
    StorageReference mStorageRef;
    FirebaseFirestore mFirestore;
    Uri postImageUri = null;
    String currUserId;
    Bitmap compressedImageFile;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_post);
        mToolbar = (android.support.v7.widget.Toolbar)
findViewById(R.id.new_post_toolbar);
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("Добавить новое объявление");
        mChoosePhoto = findViewById(R.id.choose_photo);
        mTitle = findViewById(R.id.title);
        mDesc = findViewById(R.id.description);
        mContent = findViewById(R.id.content);
        mAddBtn = findViewById(R.id.add_post_btn);
        mStorageRef = FirebaseStorage.getInstance().getReference();
        mFirestore = FirebaseFirestore.getInstance();
        mAuth = FirebaseAuth.getInstance();

        currUserId = mAuth.getCurrentUser().getUid();

```

```

mChoosePhoto.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        CropImage.activity()
            .setGuidelines(CropImageView.Guidelines.ON)
            //.setMinCropResultSize(256, 128)
            //.setAspectRatio(2, 1)
            .start(NewPostActivity.this);
    }
});
mAddBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if (postImageUri == null) {
            Toast.makeText(NewPostActivity.this, "Необходимо выбрать
фото", Toast.LENGTH_LONG).show();
        }
        final String title = mTitle.getText().toString();
        final String desc = mDesc.getText().toString();
        final String content = mContent.getText().toString();

        if (!TextUtils.isEmpty(title) && !TextUtils.isEmpty(desc) &&
!TextUtils.isEmpty(content) && postImageUri != null) {

            final String randomString = UUID.randomUUID().toString();
            final StorageReference image_path =
mStorageRef.child("post_images").child(currUserId).child(randomString + ".jpg");
            UploadTask uploadTask = image_path.putFile(postImageUri);

            uploadTask.addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {

                    final Map<String, Object> postMap = new HashMap<>();

                    mStorageRef.child("post_images/" + currUserId + "/" +
randomString + ".jpg").getDownloadUrl()
                        .addOnSuccessListener(new
OnSuccessListener<Uri>() {
                            @Override
                            public void onSuccess(Uri uri) {

                                //Map<String, Object> postMap = new
HashMap<>();

                                postMap.put("title", title);
                                postMap.put("description", desc);
                                postMap.put("content", content);
                                postMap.put("image", uri.toString());
                                postMap.put("user_id", currUserId);
                                postMap.put("timestamp",
FieldValue.serverTimestamp());
                            }
                        });

                    File newThumbFile = new File(postImageUri.getPath());
                    try {

                        compressedImageFile = new
Compressor(NewPostActivity.this)

```

```

        .setMaxHeight(100)
        .setMaxWidth(100)
        .setQuality(1)
        .compressToBitmap(newThumbFile);

    } catch (IOException e) {
        e.printStackTrace();
    }
    ByteArrayOutputStream baos = new
ByteArrayOutputStream();

compressedImageFile.compress(Bitmap.CompressFormat.JPEG, 100, baos);
byte[] thumbData = baos.toByteArray();

    UploadTask uploadTaskThumbnail =
mStorageRef.child("post_images").child(currUserId).child("thumbs")
        .child(randomString +
".jpg").putBytes(thumbData);

    uploadTaskThumbnail.addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {

            mStorageRef.child("post_images/" + currUserId
+ "/thumbs/" + randomString + ".jpg").getDownloadUrl()
                .addOnSuccessListener(new
OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        postMap.put("thumb",
uri.toString());

mFirestore.collection("Posts").add(postMap).addOnCompleteListener(new
OnCompleteListener<DocumentReference>() {
                    @Override
                    public void
onComplete(@NonNull Task<DocumentReference> task) {
                        if
(task.isSuccessful()) {
                            //post saved, go
to main

Toast.makeText(NewPostActivity.this, "Объявление успешно добавлено",
Toast.LENGTH_LONG).show();

                            sendToMain();
                        } else {
                            String err =
task.getException().getMessage();

Toast.makeText(NewPostActivity.this, err, Toast.LENGTH_LONG).show();
                        }
                    }
                });
            }
        });
    }
});

```

```

        });
    }
}

private void sendToMain() {
    Intent toMain = new Intent(NewPostActivity.this, MainActivity.class);
    startActivity(toMain);
    finish();
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
        CropImage.ActivityResult result = CropImage.getActivityResult(data);
        if (resultCode == RESULT_OK) {

            postImageUri = result.getUri();
            mChoosePhoto.setText("Photo choosed");
            mChoosePhoto.setTypeface(mChoosePhoto.getTypeface(),
Typeface.BOLD);

        } else if (resultCode ==
CropImage.CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {

            Exception error = result.getError();

        }

    }
}

```

```

NotificationFrgment.java
package com.example.medshering;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.medshering.R;

```

```

public class NotificationFragment extends Fragment {

    public NotificationFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_notification, container, false);
    }
}

```

```
}
```

```
PostFragment.java
```

```
package com.example.medshering;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.text.format.DateFormat;
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.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import javax.annotation.Nullable;

import de.hdodenhof.circleimageview.CircleImageView;
```

```
public class PostFragment extends Fragment {
```

```
    FirebaseAuth mAuth;
    FirebaseFirestore mFirestore;
```

```
    private CircleImageView userProfileImageView;
    private TextView userNameView;
    private TextView dateView;
    private TextView titleView;
    private ImageView imageView;
    private TextView descView;
    private TextView contentView;
    private ImageView likeBtn;
    private TextView likeCounter;
```

```
    @Override
```

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

```
        View view = inflater.inflate(R.layout.fragment_ad_post, container, false);
```

```
        mAuth = FirebaseAuth.getInstance();
```

```

mFirestore = FirebaseFirestore.getInstance();
final String currentUserId = mAuth.getCurrentUser().getUid();
final String adPostId = getArguments().getString("adPostId");
final String userId = getArguments().getString("userId");
userProfileImageView = view.findViewById(R.id.post_user_image);
userNameView = view.findViewById(R.id.post_user_name);
dateView = view.findViewById(R.id.post_date);
titleView = view.findViewById(R.id.post_title);
ImageView = view.findViewById(R.id.post_image);
descView = view.findViewById(R.id.post_desc);
contentView = view.findViewById(R.id.post_content);
android.widget.ImageView commBtn = view.findViewById(R.id.post_comm_btn);
likeBtn = view.findViewById(R.id.post_like_btn);
likeCounter = view.findViewById(R.id.post_like_counter);
commBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent commentIntent = new Intent(getActivity(),
CommentsActivity.class);
        commentIntent.putExtra("adPostId", adPostId);
        startActivity(commentIntent);
    }
});
mFirestore.collection("Posts/" + adPostId +
"/Likes").addSnapshotListener(new EventListener<QuerySnapshot>() {
    @Override
    public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots,
@Nullable FirebaseFirestoreException e) {

        if (!queryDocumentSnapshots.isEmpty()) {
            int counter = queryDocumentSnapshots.size();
            likeCounter.setText(Integer.toString(counter));
        } else {
            likeCounter.setText(" ");
        }
    }
});
mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).addSnapshotListener(new
EventListener<DocumentSnapshot>() {
    @Override
    public void onEvent(@Nullable DocumentSnapshot documentSnapshot,
@Nullable FirebaseFirestoreException e) {
        if (documentSnapshot.exists()) {

likeBtn.setImageDrawable(getActivity().getDrawable(R.mipmap.action_like_btn_accent
));

        } else {

likeBtn.setImageDrawable(getActivity().getDrawable(R.mipmap.action_like_btn_gray))
;

        }
    }
});

likeBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {

```

```

        if(!task.getResult().exists()){
            Map<String, Object> likesMap = new HashMap<>();
            likesMap.put("timestamp",
FieldValue.serverTimestamp());

            mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).set(likesMap);
        } else {
            mFirestore.collection("Posts/" + adPostId +
"/Likes").document(currentUserId).delete();
        }
    }
    });
}

});
DocumentReference userDocRef =
mFirestore.collection("Users").document(userId);
userDocRef.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
    @Override
    public void onSuccess(DocumentSnapshot documentSnapshot) {
        User user = documentSnapshot.toObject(User.class);
        String firstName = user.getFirst();
        String lastName = user.getLast();
        String profileImageUrl = user.getImage();
        setUserData(firstName, lastName, profileImageUrl);
    }
});
DocumentReference adDocRef =
mFirestore.collection("Posts").document(adPostId);
adDocRef.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
    @Override
    public void onSuccess(DocumentSnapshot documentSnapshot) {
        AdPost adpost = documentSnapshot.toObject(AdPost.class);
        try {
            long millisecond = adpost.getTimestamp().getTime();
            String adDate = DateFormat.format("d/MM/yyyy", new
Date(millisecond)).toString();
            dateView.setText(adDate);
        } catch (Exception e) {
            Toast.makeText(getActivity(), e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
        String adTitle = adpost.getTitle();
        String adImageUrl = adpost.getImage();
        String adImageThumbUrl = adpost.getThumb();
        String adDesc = adpost.getDescription();
        String adContent = adpost.getContent();

        setAdData(adTitle, adImageUrl, adImageThumbUrl, adDesc,
adContent);
    }
});
return view;
}

public void setUserData(String firstName, String lastName, String
profileImageUrl) {
    RequestOptions placeholderOption = new RequestOptions();
    placeholderOption.placeholder(R.drawable.ellipse);

    Glide.with(this).applyDefaultRequestOptions(placeholderOption).load(profileImageUr

```

```

1).into(userProfileImageView);
    userNameView.setText(firstName + " " + lastName);
}
public void setAdData(String adTitle, String adImageUrl, String
adImageThumbUrl, String adDesc, String adContent) {
    titleView.setText(adTitle);

    RequestOptions placeholderOption = new RequestOptions();
    placeholderOption.placeholder(R.drawable.rectangle);
    Glide.with(this).applyDefaultRequestOptions(placeholderOption)
        .load(adImageUrl)
        .thumbnail(Glide.with(this).load(adImageThumbUrl))
        .into(ImageView);

    descView.setText(adDesc);
    contentView.setText(adContent);
}
}

```

RegisterActivity.java

```

package com.example.medshering;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class RegisterActivity extends AppCompatActivity {

    EditText mEmail;
    EditText mPassword;
    EditText mConfirmPassword;
    Button mRegisterBtn;
    FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        mAuth = FirebaseAuth.getInstance();
        mEmail = (EditText) findViewById(R.id.email_field);
        mPassword = (EditText) findViewById(R.id.password_field);
        mConfirmPassword = (EditText) findViewById(R.id.confirm_password_field);
        mRegisterBtn = (Button) findViewById(R.id.register_button);
        mRegisterBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email = mEmail.getText().toString();
                String pass = mPassword.getText().toString();
                String confirmPass = mConfirmPassword.getText().toString();

                if (!TextUtils.isEmpty(email) && !TextUtils.isEmpty(pass) &&

```



```

!TextUtils.isEmpty(confirmPass)) {

    if (pass.equals(confirmPass)) {

        mAuth.createUserWithEmailAndPassword(email,
pass).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task)
{
                if (task.isSuccessful()) {
                    sendToSetup();

                } else {
                    String errorMessage =
task.getException().getMessage();
                    Toast.makeText(RegisterActivity.this, "Error :
" + errorMessage, Toast.LENGTH_LONG).show();
                }
            }
        });

    } else {
        Toast.makeText(RegisterActivity.this, "Пароли не
совпадают", Toast.LENGTH_LONG).show();
    }
    } else {
        Toast.makeText(RegisterActivity.this, "Поля не должны быть
пустыми", Toast.LENGTH_LONG).show();
    }
}

});

}

@Override
protected void onStart() {
    super.onStart();

    FirebaseUser currentUser = mAuth.getCurrentUser();
    if (currentUser != null) {
        sendToMain();
    }

}

private void sendToMain() {
    Intent toMain = new Intent(RegisterActivity.this, MainActivity.class);
    startActivity(toMain);
    finish();
}

private void sendToSetup() {
    Intent toSetup = new Intent(RegisterActivity.this, SetupActivity.class);
    startActivity(toSetup);
    finish();
}

public void clickToLogin(View view) {
    startActivity(new Intent(this, LoginActivity.class));
}

}

```

SetupActivity.java

```

package com.example.medshering;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.WriteBatch;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.theartofdev.edmodo.cropper.CropImage;
import com.theartofdev.edmodo.cropper.CropImageView;

import java.util.HashMap;
import java.util.Map;

import de.hdodenhof.circleimageview.CircleImageView;

public class SetupActivity extends AppCompatActivity {
    CircleImageView setupImage;
    Uri imageUri;
    EditText mFirstName;
    EditText mLastName;
    EditText mBio;
    Button mSaveBtn;
    FirebaseAuth mAuth;
    StorageReference mStorageRef;
    FirebaseFirestore mFirestore;
    String firstName;
    String lastName;
    String biography;
    String user_id;

    boolean isImageChanged = false;

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

        mFirstName = findViewById(R.id.firstname_field);
        mLastName = findViewById(R.id.lastname_field);
        mBio = findViewById(R.id.biography_field);
        mSaveBtn = findViewById(R.id.save_settings_btn);

```

```

        setupImage = findViewById(R.id.profile_image);

        mAuth = FirebaseAuth.getInstance();
        mStorageRef = FirebaseStorage.getInstance().getReference();
        mFirestore = FirebaseFirestore.getInstance();

        user_id = FirebaseAuth.getInstance().getCurrentUser().getUid();

        //retrieving initial data

        mFirestore.collection("Users").document(user_id).get().addOnCompleteListener(new
        OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                if (task.isSuccessful()) {
                    if (task.getResult().exists()) {
                        //data exist and fields will be completed
                        String firstN = task.getResult().getString("first");
                        String lastN = task.getResult().getString("last");
                        String bio = task.getResult().getString("bio");
                        String image = task.getResult().getString("image");

                        imageUrl = Uri.parse(image);

                        mFirstName.setText(firstN);
                        mLastName.setText(lastN);
                        mBio.setText(bio);

                        RequestOptions placeholderRequest = new RequestOptions();
                        placeholderRequest.placeholder(R.drawable.profileimage);

                        Glide.with(SetupActivity.this).setDefaultRequestOptions(placeholderRequest).load(i
                        mage).into(setupImage);

                    }

                    } else {
                        String err = task.getException().getMessage();
                        Toast.makeText(SetupActivity.this, "Firestore GET: " + err,
                        Toast.LENGTH_LONG).show();
                    }
                }
            });

        setupImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                //daca e mai mare decat Marshmello, e nevoie de request permission
                aici

                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

                    if (ContextCompat.checkSelfPermission(SetupActivity.this,
                    android.Manifest.permission.READ_EXTERNAL_STORAGE) !=
                    PackageManager.PERMISSION_GRANTED) {

                        Toast.makeText(SetupActivity.this, "В доступе отказано",
                        Toast.LENGTH_LONG).show();
                        ActivityCompat.requestPermissions(SetupActivity.this, new
                        String[]{android.Manifest.permission.READ_EXTERNAL_STORAGE}, 1);

                    } else {
                        popUpImagePicker();
                    }
                }
            }
        });
    }
}

```

```

        }
    } else {
        popUpImagePicker();
    }
}

});
mSaveBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (imageUri == null) {
            Toast.makeText(SetupActivity.this, "Необходимо выбрать фото",
                Toast.LENGTH_LONG).show();
        }

        firstName = mFirstName.getText().toString();
        lastName = mLastName.getText().toString();
        biography = mBio.getText().toString();

        if (!TextUtils.isEmpty(firstName) && !TextUtils.isEmpty(lastName)
            && !TextUtils.isEmpty(biography)) {

            if (isImageChanged) {
                final StorageReference image_path =
mStorageRef.child("profile_images").child(user_id + ".jpg");
                UploadTask uploadTask = image_path.putFile(imageUri);
                // Register observers to listen for when the download is
done or if it fails
                uploadTask.addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception exception) {
                        String errorMsg = exception.getMessage();
                        Toast.makeText(SetupActivity.this, "Error: " +
errorMsg, Toast.LENGTH_LONG).show();
                    }
                }).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {
                        // taskSnapshot.getMetadata() contains file
metadata such as size, content-type, etc.
                        // ...
                        //Toast.makeText(SetupActivity.this, "Image upload
successfully", Toast.LENGTH_LONG).show();

                        mStorageRef.child("profile_images/" + user_id +
".jpg").getDownloadUrl()
                            .addOnSuccessListener(new
OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    Map<String, Object> userMap = new
HashMap<>();

                                    userMap.put("first", firstName);
                                    userMap.put("last", lastName);
                                    userMap.put("bio", biography);
                                    userMap.put("image",
uri.toString());

                                    mFirestore.collection("Users").document(user_id).set(userMap).addOnCompleteListene
r(new OnCompleteListener<Void>() {
                                        @Override
                                        public void

```

```

onComplete(@NonNull Task<Void> task) {
    if (task.isSuccessful()) {
        Toast.makeText(SetupActivity.this, "Подключен к Firestore",
            Toast.LENGTH_LONG).show();
        sendToMain();
    } else {
        String err =
            task.getException().getMessage();
        Toast.makeText(SetupActivity.this, err, Toast.LENGTH_LONG).show();
    }
}

}).addOnFailureListener(new
OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText(SetupActivity.this,
            "Попробуйте ещё раз", Toast.LENGTH_LONG).show();
    }
});

} else {
    //daca poza nu e schimbata se vor update doar fieldurile
    text

    WriteBatch batch = mFirestore.batch();
    DocumentReference userRef =
mFirestore.collection("Users").document(user_id);
    batch.update(userRef, "first", firstName);
    batch.update(userRef, "last", lastName);
    batch.update(userRef, "bio", biography);

    // Commit the batch
    batch.commit().addOnCompleteListener(new
OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            Toast.makeText(SetupActivity.this, "Настройки
сохранены", Toast.LENGTH_LONG).show();
            sendToMain();
        }
    });
}

} else {
    Toast.makeText(SetupActivity.this, "Все поля должны быть
заполнены", Toast.LENGTH_LONG).show();
}

});

}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {

```

```

        CropImage.ActivityResult result = CropImage.getActivityResult(data);
        if (resultCode == RESULT_OK) {
            imageUri = result.getUri();
            setupImage.setImageURI(imageUri);

            isImageChanged = true;

        } else if (resultCode ==
CropImage.CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {
            Exception error = result.getError();
        }
    }

    private void popUpImagePicker() {
        CropImage.activity()
            .setGuidelines(CropImageView.Guidelines.ON)
            .setAspectRatio(1,1)
            .start(SetupActivity.this);
    }

    private void sendToMain() {
        Intent toMain = new Intent(SetupActivity.this, MainActivity.class);
        startActivity(toMain);
        finish();
    }
}

```

User.java

```
package com.example.medshering;
```

```

public class User {

    private String first, last, image, bio;

    public User() {
    }

    public User(String first, String last, String image, String bio) {
        this.first = first;
        this.last = last;
        this.image = image;
        this.bio = bio;
    }

    public String getFirst() {
        return first;
    }

    public void setFirst(String first) {
        this.first = first;
    }

    public String getLast() {
        return last;
    }

    public void setLast(String last) {
        this.last = last;
    }

    public String getImage() {

```

```
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    public String getBio() {
        return bio;
    }

    public void setBio(String bio) {
        this.bio = bio;
    }
}
```