МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ANDROID ДОДАТОК ДЛЯ ОРГАНІЗАЦІЇ ТА СУПРОВОДЖЕННЯ ОНЛАЙН ЗУСТРІЧЕЙ

Здобувач освіти гр. ІН – 83 Владислав КАРПЕНКО

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп'ютерних наук Ольга ШУТИЛЄВА

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ
до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-83 спеціальності «122 – Комп'ютерні науки» денної форми навчання Карпенка Владислава Миколайовича.

Тема: «ANDROID ДОДАТОК ДЛЯ ОРГАНІЗАЦІЇ ТА СУПРОВОДЖЕННЯ ОНЛАЙН ЗУСТРІЧЕЙ»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) літературний огляд за обраною тематикою роботи; 2) постановка завдання для розробки; 3) вибір оптимальних інструментів для розробки мобільного додатку; 4) практична реалізація.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____Ольга ШУТИЛЄВА
Завдання прийняв до виконання _____ Владислав КАРПЕНКО

РЕФЕРАТ

Записка: 66 стор., 31 рис., 1 додаток, 16 джерел.

Об’єкт дослідження – процес проектування та реалізації додатку для проведення онлайн зустрічей під платформу Android.

Мета роботи – розробка мобільного додатку, який дозволяє користувачам організовувати онлайн зустрічі, при цьому на відмінну від більшості популярних рішень, дозволяє зберігати історію відвідувань, що значно спростить навігацію користувача для майбутніх повторних підключень.

Методи дослідження – метод розробки додатків для операційної системи Android.

Результати – розроблено мобільний додаток для операційної системи Android з використанням сервісу Firebase, комплекту для розробки програмного забезпечення Jitsi та з використанням мови програмування Java. Дизайн був розроблений у відповідності до сучасних тенденцій та на основі мануальних тестувань, що дозволило впевнитися у простоті інтерфейсу. Для розробки програмного забезпечення використано середовище Android Studio.

ANDROID STUDIO, ANDROID ДОДАТОК, МОБІЛЬНИЙ ДОДАТОК, ВІДЕОЗВ’ЯЗОК, JAVA, SDK.

ЗМІСТ

ВСТУП

У період пандемії зросла роль віддаленої роботи, й у забезпечення плідної взаємодії своїх співробітників, компаніям потрібно було залучити додаткові засоби комунікації. Користувачам були потрібні ефективні інструменти для спільної роботи, що призвело до стрімкого зростання ринку уніфікованих комунікацій.

Завдяки розвитку сучасних софтверних та кросплатформових рішень, а також своєчасному заміщенню застарілих технологій зв'язку та побудові відеоконференцзв'язку фінансові показники компаній, які перейшли на віддалений формат роботи, залишилися на колишньому рівні .Найбільшого поширення на ринку уніфікованих комунікацій отримали рішення відеозв'язку. У міру пом'якшення карантинних заходів сеанси такого роду зв’язку залишаються так само актуальними, їх продовжують використовувати як для вирішення питань, пов'язаних з бізнес-процесами, так і для особистого спілкування.

Переваги відеозв'язку очевидні: вони прості у використанні, економлять час, а також дозволяють працювати у закритих мережах, забезпечуючи при цьому максимальну безпеку.

Відеоконференцзв'язок передбачає як спілкування між двома людьми, і групові конференції з великою кількістю учасників. Серед сервісів відеозв'язку, що часто використовуються, можна виділити наступні: обмін миттєвими повідомленнями, планування заходів, обмін файлами, проведення презентацій. За допомогою цих інструментів можна записувати відеоконференції, додавати учасників, відстежувати заплановані заходи через календар та надсилати запрошення. Під час дзвінка можна обмінюватися інформацією через повідомлення або передавати документи простим перетягуванням у вікно файлу [1].

За забезпечення конфіденційності під час проведення онлайн-переговорів відповідає шифрування. Воно значно підвищує безпеку відеозв'язку без шкоди продуктивності.

Міркувати про плюси та мінуси конкретних сервісів відеоконференції не зовсім коректно, тому що їх вибір залежить від сфери діяльності, масштабу компанії та завдань, які планується вирішити за допомогою цих інструментів. При виборі програми для конференц-зв'язку головним критерієм залишається якість відео під час передачі даних та кількість учасників. Необхідно віддавати перевагу перевіреним продуктам, оскільки через постійні зависання зображення та інших технічних збоїв спілкуватися буде вкрай складно.

Відеоконференцзв'язок – це дійсно робочий інструмент, який часто використовується кілька разів на день для вирішення найрізноманітніших завдань. Він допомагає економити кошти компанії та робочий час співробітників, тому випробувавши його, багато компаній не хочуть повертатися до колишньої моделі ведення бізнесу [1].

У зв’язку з цим актуальність кваліфікаційної роботи бакалавра з розробки додатку сфери відеозв'язку є на дуже високому рівні.

Метою роботи є проектування та розробка мобільного додатку для відеоконференцій. Ключовою відмінністю цього додатку від інших засобів – це можливість зберігати та редагувати історію підключень до конференцій.

За допомогою відкритої SDK отримаємо можливість підключатись до конференцій через веб-браузер та мобільний додаток одночасно.

1. ЛІТЕРАТУРНИЙ ОГЛЯД ЗА ОБРАНОЮ ТЕМАТИКОЮ РОБОТИ

1.1 Аналіз методів розробки мобільних додатків

У світі мобільних пристроїв як відомо переважають 3 основні операційні системи, а саме – Android, IOS та новачок HarmonyOS. У зв'язку з різноманіттям операційних систем, компанії та замовники мають вибір у створенні програмного продукту.

Отже існує 2 методи розробки мобільних програм:

- Нативна розробка;
- Кросплатформова технологія.

Нативна технологія передбачає створення програми для мобільного пристрою конкретною мовою під конкретну платформу. Нативні програми досить відмовостійкі та легко маштабовані, через що не мають обмежень у розробці. До плюсів такої розробки можна віднести досить швидку реакцію на дії користувача, можливість мати прямий доступ до апаратної частини та розробити найбільш звичний для користувача конкретної платформи інтерфейс. До недоліків можна віднести досить високу вартість розробки та підтримки і тривалий час, необхідний на розробку. Розробляти один і той же додаток під різні платформи (iOS/Android) довго і дорого, до того ж їх тестування також займе досить тривалий час (наприклад, компіляція оновлень для Android займає близько 30 сек), тому якщо потрібно створити простий додаток відразу для двох платформ, вдаються до гібридного способу розробки[2].

Кросплатформова розробка проводиться за допомогою веб-технологій – HTML, CSS і JavaScript – які дозволяють розробити програму відразу на кілька платформ. Але для того, щоб додаток працював відповідно до своєї платформи, його потрібно «перекласти» на зрозумілу платформу мову, або додати проміжну ланку-перекладач. До переваг можна віднести низьку

вартість розробки, адже для цього іноді достатньо буде задіяти лише одного фахівця. А ось до недоліків можна віднести можливі труднощі в роботі всіх функцій та затримки в реакції на дію користувача (додаток може бути повільним), також інтерфейс буде досить простим і потрібно буде додатково доопрацьовувати [2].

1.2. Інструменти для розробки мобільних додатків

Найперше – для розробки мобільних додатків потрібно вивчити мови програмування. Для різних платформ підходять різні мови, тому спочатку потрібно визначитися з платформою, а далі – з мовою. Розробка мобільних додатків для Android найчастіше виконується на Java (рис.1.1), на якій написано більше 90% всіх додатків під Android.

Рисунок 1.1 – Приклад мови програмування Java [3]

За останні півроку більшої популярності набирає нова мова Kotlin (рис. 1.2). Поки близько 5% додатків в Google Play написані мовою Kotlin, але з кожним роком кількість цих додатків зростає.

```kotlin
// file.kt
fun main(args: Array<String>) {
    println("Hello, World!")
}
```

Рисунок 1.2 – Приклад мови програмування Kotlin [3]

Якщо говорити про iOS платформу, то тут також використовуються дві основних мови – Objective C (рис. 1.3), вона ж перша мова, яка була розроблена компанією Apple для створення програмного забезпечення під iOS.

```objective-c
#import
#import

int main(void)
{
    NSLog(@"Hello, world!");
    return 0;
}
```

Рисунок 1.3 – Приклад мови програмування Objective C [3]

А друга мова – це більш просунутий і сучасніший Swift (рис 1.4). Якщо говорити про підтримку старих пропозицій, які були написані раніше, то тут однозначно потрібно знати Objective C, бо нові додатки все частіше пишуться саме на Swift.

```swift
// file.swift
import Swift
print("Hello, World!")
```

Рисунок 1.4 – Приклад мови програмування Swift [3]

Варто згадати, що як для однієї, так і іншої платформи іноді використовується мова C ++. Вона використовується у тих випадках, коли потрібно досягти максимальної продуктивності від додатку.

Далі необхідно обрати інтегроване середовище розробки. Це важливо тому що кожна платформа (iOS/Android) та кожна мова програмування має свої унікальні речі, такі як наприклад синтаксис, запити до системи, або навіть обмеження. Інтегровані середовища розробки враховують цю специфіку та допомагають розробнику, тим самим пришвидшує та спрощує розробку продукту. Серед найпопулярніших таких середовищ можна виділити Android Studio (рис. 1.5), Xcode та Visual Studio Code.

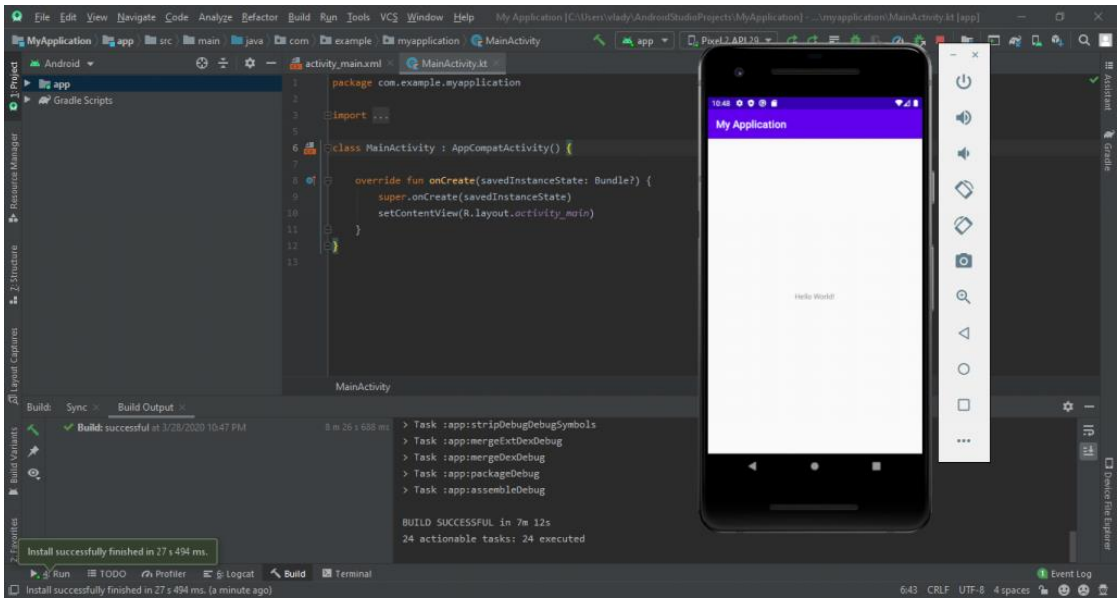


Рисунок 1.5 – Огляд IDE Android Studio

Android Studio– інтегроване середовище розробки виробництва Google, за допомогою якого розробникам стають доступні інструменти для створення програм на платформі Android OS. Android Studio можна встановити на Windows, Mac та Linux. Android Studio створене з урахуванням IntelliJ IDEA.

Android Studio містить інструменти для розробки рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

Середовище Android Studio призначене як для невеликих команд розробників мобільних додатків (навіть однієї людини), так і для великих міжнародних організацій з GIT або іншими подібними системами управління версіями. Досвідчені розробники зможуть вибрати інструменти, які найбільше підходять для масштабних проектів. Рішення для Android розробляються в Android Studio за допомогою Java або C++.

У основі робочого процесу Android Studio закладено концепт безперервної інтеграції, що дозволяє відразу виявляти наявні проблеми. Тривала перевірка коду забезпечує можливість ефективного зворотного зв'язку з розробниками. Така опція дозволяє швидше опублікувати версію мобільного додатка в Google Play App Store. Для цього є також підтримка інструментів LINT, Pro-Guard і App Signing.

За допомогою засобів оцінки продуктивності визначається стан файлу з пакетом прикладних програм. Візуалізація графіки дає змогу дізнатися, чи програма відповідає орієнтиру Google в 16 мілісекунд.

За допомогою інструмента для візуалізації пам'яті розробник дізнається, коли його додаток буде використовувати занадто багато оперативної пам'яті і коли відбудеться складання сміття. Інструменти для аналізу батареї показують, яке навантаження лягає на пристрій [4].

Android Studio сумісна з платформою Google App Engine для швидкої інтеграції у хмари нових API та функцій. У розробці є різні API, такі як Google Play, Android Pay і Health. Є підтримка всіх платформ Android, починаючи з Android 1.6. Нові функції відображаються з кожною новою версією Android Studio. На даний момент доступні такі функції:

- Розширений редактор макетів: WYSIWYG, здатність працювати з UI компонентами за допомогою Drag-and-Drop, функція попереднього перегляду макета на декількох конфігураціях екрана;
- Складання додатків, засноване на Gradle. Gradle – це система збірки проекту, така сама як Maven;
- Різні види збірок та генерація декількох .apk файлів;

- Рефакторинг коду;
- Статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій та інше;
- Вбудований ProGuard та утиліта для підписування додатків. ProGuard – утиліта, яка видаляє з готового коду фрагменти, що не використовуються, і змінює імена змінних і методів для ускладнення реверс-інжинірингу програми. Також дозволяє зменшити розмір файлів, що завантажуються на пристрій;
- Шаблони основних макетів та компонентів Android;
- Підтримка розробки додатків для Android Wear та Android TV;
- Вбудована підтримка Google Cloud Platform, яка включає інтеграцію з сервісами Google Cloud Messaging і App Engine.

Android Studio 2.1 підтримує Android Preview SDK, а це означає, що розробники зможуть розпочати роботу зі створення програми для нової програмної платформи.

Нова версія Android Studio 2.1 здатна працювати з оновленим компілятором Jack, а також має покращену підтримку Java 8 та вдосконалену функцію Instant Run. Починаючи з Platform-tools 23.1.0 для Linux винятково 64-розрядна. У Android Studio 3.0 будуть за стандартом включені інструменти мови Kotlin, засновані на JetBrains IDE [4].

1.3 Постановка задачі

Для успішної реалізації мобільного додатку у ході роботи необхідно реалізувати наступні задачі:

1. Виконати огляд інструментів для розробки та обрати оптимальні;
2. Розробити інтуїтивно зрозумілий для майбутніх користувачів інтерфейс мобільно додатку;
3. Виконати проектування архітектури мобільного додатку;
4. Розробити алгоритм роботи додатку;

5. Виконати мануальне тестування мобільного додатку та зробити відповідні висновки по роботі.

У разі успішної реалізації мобільного додатку, для користувачів буде можливим виконувати наступне:

1. Приєднуватись до онлайн конференцій;

2. Зберігати і використовувати список історії відвіданих конференцій;

3. Можливість підключатись до раніше відвіданих конференцій за допомогою сформованого списку, та редагувати його, даючи назви відвіданим кімнатам, що у майбутньому спросить пошук та розуміння.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

Для виконання поставленого до роботи завдання необхідно вирішити ряд питань пов’язаних зі стеком технологій, завдяки яким буде реалізовано мобільний додаток. У першу чергу треба обрати платформу, для якої буде розроблятись мобільний додаток.

Для кросплатформової розробки необхідно більше часу на вивчена особливостей тої чи іншої платформи, в нашому випадку IOS та Android, тому цей варіант для нашої розробки не підходить.

Для нативної розробки під платформу IOS потреба у часі зменшиться, оскільки доведеться вивчати особливості лише однієї платформи, також завдяки цьому зменшиться кількість багів та інтерфейсних прогалин, які міг би допустити розробник кросплатформових рішень.

Для нативної розробки під Android зберігаються ті самі плюси що й для розробки під IOS, але на відмінну від програмного забезпечення компанії Appe, платформа компанії Google має суттєву перевагу у кількості активних пристроїв, а тим самим у кількості користувачів. Смартфонів на операційній системі Android банально більше, що розширює кількість потенційних користувачів [5].

Тому у висновку було обрано операційну систему Android для побудови мобільного додатку.

Далі треба обрати мову програмування, що підтримується операційною системою. У випадку з Android існує два варіанти:

1. Java.
2. Kotlin.

Коли в травні 2019 року компанія Google оголосила Kotlin кращою мовою для Android-розробки, суперечки на вибір мови розгорілися з новою силою. З одного боку, все просто: писати потрібно тією мовою, якою зручно особисто вам. Але не можна заперечувати і ряд факторів, які здатні схилити шальки терезів на користь Java або Kotlin, а саме:

1) Вік мови

Java має велике ком'юніті, багато бібліотек, рішень та готових модулів. Грубо кажучи, якщо розробник стикається з якоюсь проблемою, він може швидко отримати відповідь на своє запитання.

А ось у випадку з Kotlin все навпаки. Молодість мови та пов'язаних бібліотек, таких як Kotlinx.Serialization або Exposed, змушує витратити чимало часу на пошук рішень. До того ж, документація мови нерідко зводиться до пояснень на кшталт «це зроблено як у Java, лише тут треба трохи змінити».

2) Кількість коду

Так, синтаксис Java передбачає, що код буде більш громіздким, ніж у Kotlin, тому написання займе більше часу.

3) Що вибрати на старті Android-розробки

Можна почати вивчення Kotlin без знання Java. Але Kotlin все ще використовує JVM і не є повноцінною альтернативою, хоч і займає певну нішу в Android-розробці. Якщо ж все-таки вибрати Kotlin, краще почати вивчення з огляду на Java або взагалі вивчати обидві мови одночасно.

4) Для яких цілей вибирається мова

Більшість існуючих Android-програм написані на Java, і сподіватися, що їх перепишуть на Kotlin, не варто. А оскільки UI операційної системи Android розроблений на Java, у цієї мови є SDK і багато готових бібліотек, так що ряд компаній все так само віддає перевагу Java.

5) Перспективи

Сучасні компанії все частіше розробляють мобільні програми на Kotlin. Але також можна зустріти специфічну бібліотеку, яка працює тільки з Java, і якщо не знати цієї мови – знайти відповідь буде нелегко.

Крім того, Kotlin – молода мова, і невідомо, що буде далі, в той час як Java характеризується кросплатформовістю: на ній не тільки мобільна технологія тримається, але і бекенд з десктопом [5].

Зараз Kotlin розвивається досить передбачувано та орієнтований саме на Android-розробку. Його все частіше вибирають молоді компанії, які беруть

старт у створенні мобільних додатків, і зараз спостерігається нестача Kotlin-розробників, а тому мова дуже затребувана. Навряд чи в найближчому майбутньому він замінить Java, але цілком може співіснувати, поступово набираючи популярності.

“Мов мобільної розробки існує безліч, але Java та Kotlin – найпоширеніші. Для розробки Android-додатків багато хто радить починати з вивчення Java. Це безпрограшний варіант, тому що якщо ви освоїте Java, наступний перехід на Kotlin не складе труднощів. Я вважаю, що потрібно знати обидві мови – неможливо знати Kotlin, не знаючи Java.” – такими словами прокоментувала обрання мови програмування під платформу Android Валерія Мерясєва (Android-розробник IT-компанії MediaSoft) [5].

2.1 Мова програмування Java

У результаті проведеного аналізу було обрано мову програмування Java. Плюси її використання перед мовою Kotlin були озвучені у попередньому розділі. У цьому розділі мова піде про саму мову програмування Java.

Java – мова програмування загального призначення. Належить до об'єктно-орієнтованих мов програмування, до мов із строгою типізацією.

Творці реалізували принцип WORA: write once, run anywhere або «пиши один раз, запускай скрізь». Це означає, що написаний на Java програму можна запустити на будь-якій платформі, якщо на ній встановлено середовище виконання Java (JRE, Java Runtime Environment).

Це завдання вирішується завдяки компіляції написаного Java коду в байт-код. Цей формат виконує JVM або віртуальна машина Java. JVM – частина середовища виконання Java (JRE). Віртуальна машина залежить від платформи [6].

У Java реалізований механізм управління пам'яттю, який називається збирачем сміття або garbage collector. Розробник створює об'єкти, а JRE за допомогою збирача сміття очищає пам'ять, коли об'єкти перестають

використовуватися. Пояснює експерт Микита Липський: «Є таке поняття – циклічне сміття. Всередині циклу на всі об'єкти є посилання, однак garbage collector Java видалить його, якщо об'єкти не можуть використовуватися з програми» [6].

Як зазначалося вище, синтаксис мови Java схожий на синтаксис інших сі-подібних мов. Ось його деякі особливості:

- чутливість до регістру – ідентифікатори User і user в Java є різними сутностями;

- Для іменування методів використовується lowerCamelCase. Якщо назва методу складається з одного слова, воно має починатися з малої літери. Приклад: firstMethodName();

- Для найменування класів використовується UpperCamelCase. Якщо назва складається з одного слова, вона повинна починатися з великої літери. Приклад: FirstClassName.

- Назва файлів програми має точно співпадати з назвою класу з урахуванням чутливості до регістру. Наприклад, якщо клас називається FirstClassName, файл має називатися FirstClassName.java;

- ідентифікатори завжди починаються з літери (A-Z, a-z), символу $ або нижнього підкреслення _;

На Java написано багато веб-програм. Популярні фреймворки, у тому числі Spring, Stuts, JSP, використовуються для створення різних програм у Інтернеті: від ecommerce-проектів до великих порталів, від освітніх платформ до урядових ресурсів.

Мобільна технологія – ще одна область використання Java. Цією мовою пишуть програми для пристроїв, що працюють під керуванням Android.

На Java створюють клієнтські програми. Простий і близький до розробників приклад: IDE NetBeans написано на «джаві».

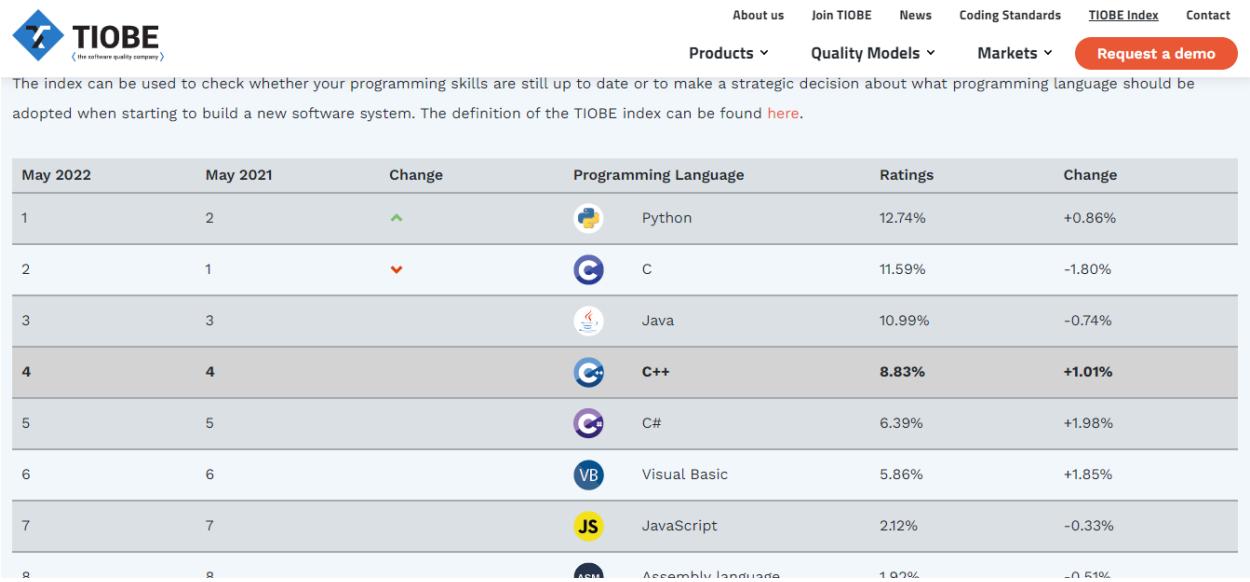
Також Java застосовується для роботи з Big Data, розробки програм для наукових цілей, наприклад обробки природних мов, програмування приладів – від побутових девайсів до промислових установок [6].

Тобто на Java можна писати різні типи додатків: веб, мобільний та десктопний софт, ігри тощо. Традиційно у цієї мови сильні позиції у промисловому програмуванні, у сегменті великих компаній (т.зв. ентерпрайз).

Мова Java вважається дружньою для початківців. Java не такий лаконічний, як Python. Однак творці Java прагнули зробити мову простою та легкою для вивчення, як і творці Python [6].

Загалом вивчення Java схоже вивчення інших мов програмування. Важливо зрозуміти, що програмування первинне, а мова вторинна. Тому важливо з перших днів навчання наголошувати на базових знаннях і розуміння принципів. Тоді буде простіше вивчати "джаву" та інші мови.

В індексі TIOBE (рис. 2.1) на травень 2022 Java займає третє місце, поступившись першим місцем Python, та другим місцем C (сі). Індекс TIOBE – індикатор популярності мов програмування, який розраховується за складною методикою з урахуванням кількості пошукових запитів, що належать до тієї чи іншої мови. У цьому перше місце цьому індексі займає мову програмування C.



May 2022	May 2021	Change	Programming Language	Ratings	Change
1	2	↑	Python	12.74%	+0.86%
2	1	↓	C	11.59%	-1.80%
3	3		Java	10.99%	-0.74%
4	**4**		**C++**	**8.83%**	**+1.01%**
5	5		C#	6.39%	+1.98%
6	6		Visual Basic	5.86%	+1.85%
7	7		JavaScript	2.12%	-0.33%
8	8		Assembly language	1.92%	-0.51%

Рисунок 2.1 – Огляд рейтингу TIOBE [7]

У рейтингу RedMonk (рис. 2.2) за червень 2022 Java займає третє місце, поступаючись JavaScript і Python. Цей рейтинг розраховується на основі

кількості репозиторіїв на GitHub, як і State of Octoverse, проте методика ранжування тут відрізняється. Наприклад, RedMonk не враховує розрахунків форки репозиторіїв.

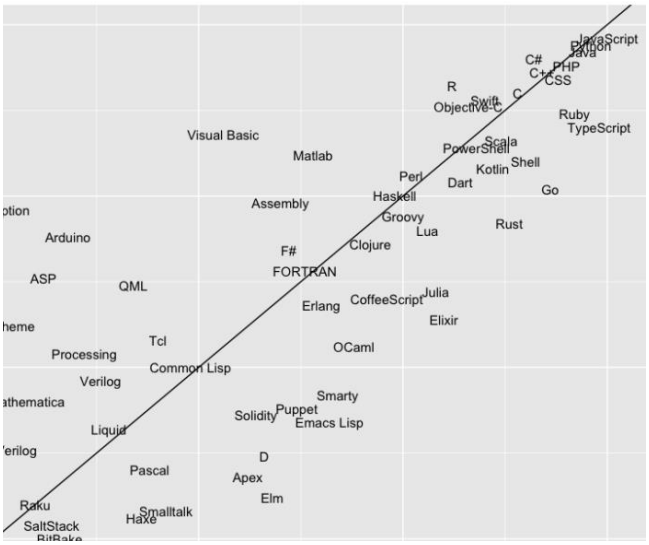


Рисунок 2.2 – Огляд рейтингу RedMonk [8]

“Новачки вибирають мову для швидкого входу в професію, і Java відповідає цьому критерію. До того ж ця мова не прив'язує людину до конкретного вузького напрямку, наприклад, фронтенд-або бекенд-розробці. На Java пишуть і фронтенд, і бекенд, та інші речі Наприклад, можна писати програми для Android. Java - мова ентерпрайзу або великих компаній, це треба враховувати при виборі мови та фінансових перспектив розробника. Немає компаній, які не використовують Java. Людство не зможе відмовитися від цієї мови, тому що нею написано дуже багато.” – В'ячеслав Ковалевський, інженер-дослідник з досвідом у світових IT гігантах. Займається розробками у сфері штучного інтелекту та та навчанням програмування [6].

2.2 Android SDK

Android SDK (рис. 2.3) – це набір засобів розробки, що використовуються для розробки програм для платформи Android, яка стала найбільшим конкурентом Apple на ринку смартфонів.

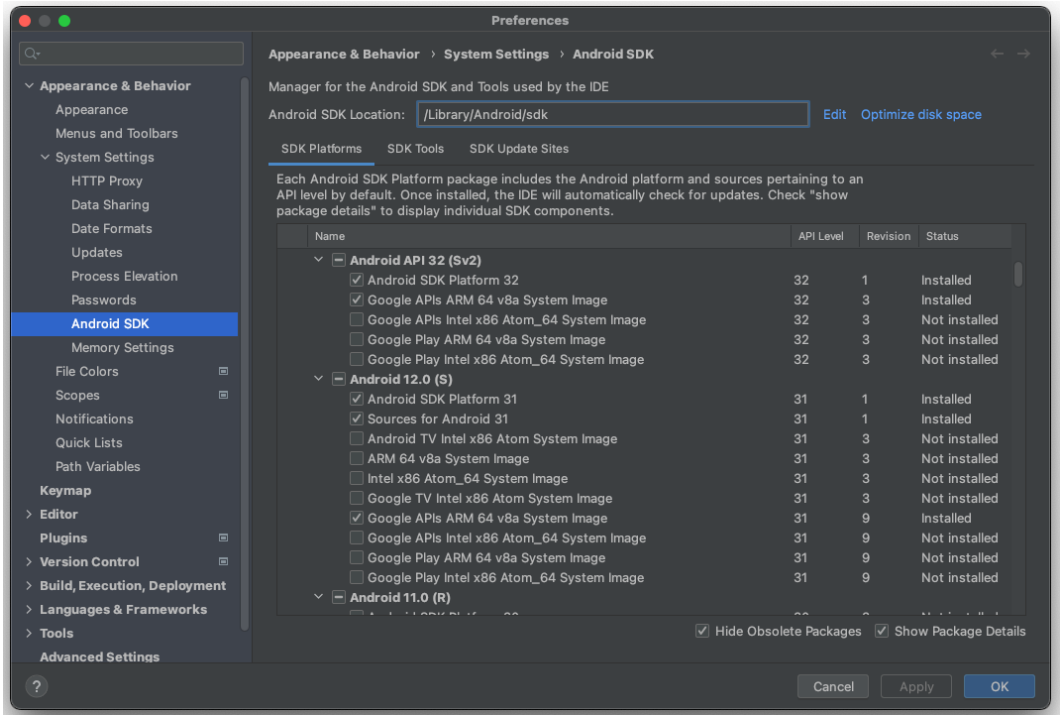


Рисунок 2.3 – Огляд Andriod SDK

Android SDK включає всі інструменти, необхідні для створення програм з нуля і навіть для їх тестування. Ці інструменти забезпечують плавний хід процесу розробки від розробки та налагодження до пакування.

Android SDK включає наступне:

- Необхідні бібліотеки.
- Відладчик.
- Емулятор.
- Відповідна документація програмних інтерфейсів додатків Android.
- Приклад вихідного коду.
- Гайди для Android.

Щоразу, коли Google випускає нову версію Android, також випускається відповідний SDK. Щоб мати змогу писати програми з найновішими функціями, розробники повинні завантажити та встановити SDK кожної версії для конкретного телефону. SDK, по суті, є наданий Android набір інструментів для конкретної версії та технології його операційних систем.

Android SDK сумісний із Windows, macOS та Linux [9].

2.3 Jitsi Meet SDK

Jitsi Meet – безкоштовне програмне забезпечення для проведення захищених шифруванням відеоконференцій. Jitsi Meet це програма з відкритим початковим кодом, що використовує транзитне шифрування даних під час передачі на сервер, за допомогою чого повідомлення шифруються перед тим, як залишити пристрій, потім розшифровуються на сервері, обробляються і знову зашифровуються перед відправкою одержувачеві або одержувачам. При використанні на довіреному сервері цей інструмент допоможе знизити ризик прослуховування, підключення сторонніх до дзвінка.

2.4 Серверна частина мобільного додатку

У якості так названого бекенду було обрано сервіс Firebase.

Розробка надійних і високоякісних додатків для мобільних пристроїв передбачає величезну самовіддачу, але що ще більш важливо, вимагає потужну і багатофункціональну платформу для розробки.

Firebase, що надається компанією Google, є однією з таких платформ, яка виборола міцні позиції серед розробників по всьому світу.

Firebase надає розробникам безліч можливостей для створення високоефективних та універсальних веб-додатків, а також програм для платформ Android та iOS.

У той час, як існує безліч конкуруючих між собою середовищ для розробки додатків, Firebase завжди використовує найкращі з доступних на даний момент платформ.

Firebase – це платформа для розробки мобільних додатків від компанії Google, в якій є найсучасніші функції для розробки, перекомпонування та покращення додатків [10] .

Firebase – це, по суті, набір інструментів, які розробники можуть використовувати, створюючи та змінюючи програми залежно від своєї потреби.

Мета Firebase полягає у вирішенні трьох основних проблем розробників:

- Швидко створити програму
- Випустити та забезпечити надійний моніторинг працездатності
- Залучити користувачів,

Розробники, що використовують цю платформу, отримують доступ до сервісів, за допомогою яких вони можуть розробляти свої продукти, і це дозволяє їм зосередитися безпосередньо на наданні якісного продукту.

Деякі з найпопулярніших функцій платформи Google Firebase включають бази даних, автентифікацію, push-сповіщення, аналітику, зберігання файлів і багато іншого.

Оскільки послуги знаходяться у хмарі, розробники можуть поетапно виконувати масштабування своїх продуктів, не відчуваючи жодних проблем. Firebase на даний момент входить до кращих платформ для розробки додатків, яким довіряють розробники по всьому світу.

Firebase зберігає текстові дані в JSON форматі та надає зручні методи для читання, оновлення та вилучення даних.

Звичайно, Firebase не може бути абсолютно безкоштовною. Частина найкрутішого функціоналу залишається за кадром для тих, хто не бажає платити. Але основні і гаряче затребувані функції реєстрації, авторизації та зберігання тексту доступні всім після реєстрації в системі [10] .

Історія Firebase досить цікава, і, як у багатьох стартапів, має за власною спиною багато злетів і падінь. Firebase бере свій початок від компанії Envolve, створеної в 2011 році Ендрю Лі та Джеймсом Тэмпліном.

Компанія запропонувала розробникам API для полегшення інтеграції чатів онлайн для сайтів. Після цього засновники Envolve виявили, що їхній чат-сервіс використовується для передачі повідомлень, не пов'язаних із чатами.

Розробники використовували платформу для синхронізації даних програм у режимі реального часу. Лі та Тэмплін вирішили відокремити архітектуру, що використовується для синхронізації в реальному часі, від системи чатів, що призвело до створення Firebase у 2011 році. Сама платформа була публічно запущена у квітні 2012 року.

Першим запущеним продуктом Firebase стала база даних Firebase Realtime. Це API для синхронізації даних програм, що працюють на пристроях Android, Web та iOS. Розробники програм можуть використовувати цю платформу для створення сумісних із різними платформами програм у реальному часі [11].

Firebase спочатку отримала фінансування у розмірі понад 1 мільйон доларів у 2012 році від спонсорів, включаючи такі компанії як Greylock Partners, New EnterpriseAssociates, Flybridge Capital Partners та Founder Collective. Компанія також отримала фінансування серії А у розмірі 5,6 мільйона доларів у червні 2013 року від компаній Flybridge Capital Partners та Union Square Ventures.

Firebase Authentication і Firebase Hosting були запущені в 2014 році компанією Firebase, ставши провідною компанією, що надають сервіс розробки для мобільних пристроїв (MBaaS).

Firebase стала частиною компанії Google у жовтні 2014 року. Потім технологічний гігант придбав компанію Divshot, платформу для веб-хостингу, після чого була об'єднана з Firebase.

Ось десять основних переваг використання Firebase:

1) Безкоштовний початковий план

2) Швидкість розробки

3) Наскрізна платформа для розробки програм

4) Працює на платформі Google

5) Розробники можуть зосередитися на фронтенді

6) Не потрібно використовувати сервер

7) Закладено можливості машинного навчання

8) Генерація трафіку для вашої програми

9) Моніторинг помилок

10) Безпека

Для того, щоб почати користуватися Firebase, не треба нічого платити, вона дозволяє користувачам входити в систему, використовуючи свій обліковий запис Google.

Firebase пропонує безкоштовний початковий план, який називається Spark, він має багато функцій, яких часто вистачає, щоб почати працювати. При збільшенні вимог до розробки, можна вибрати план Blaze [11].

Почати роботу без будь-яких первинних витрат, це, безумовно, чудова можливість, яку надає Firebase, і це одна з причин такої популярності платформи. План Spark дійсно надає хороший безкоштовний ліміт на читання та запис до бази даних. Іншою перевагою Firebase є те, що план Blaze враховує безкоштовні ліміти, які надає план Spark. Наприклад, припустимо, що хто небудь орендуєте сховище обсягом 20 ГБ пам'яті на плані Blaze. Безкоштовний тариф надає 10 ГБ, що означає, що Firebase тарифікуватиме вам лише 10 ГБ.

Firebase це відмінний варіант для розробки додатків, який може дозволити заощадити час на розробку і скоротити час виходу додатків на ринок. Зазвичай кожен розробник повинен мати доступ до сервера та хосту для створення та обслуговування баз даних та серверних служб.

Отже, для створення додатків необхідно мати бекенд та фронтенд розробника. І це так і є навіть для зовсім невеликих додатків, де бекенд та фронтенд мають взаємодіяти один з одним на різних етапах [11].

Наявність фронтенд і бекенд розробників часто призводить до помилок і проблем, які в кінцевому рахунку позначаються на якості додатків, що розробляються, і збільшують вартість і складність розробки.

Однак використання сервісів Firebase та Firestore дозволяє фронтенд розробникам самим керувати всією роботою та скорочувати час, необхідний для її завершення.

Крім того, Firebase надає величезну кількість готових до використання сервісів, які дозволяють розробникам уникнути створення шаблонного коду та написання бекенду з нуля. Firebase також пропонує розробникам весь спектр продуктів, які можуть знадобитися їм у процесі розробки.

По-перше, існує два варіанти бази даних, це Firestore та Firebase Realtime Database. Firebase дозволяє легко та просто виконувати хмарне сховище мультимедіа та забезпечує розробку програм без використання сервера за допомогою інтегрованих хмарних рішень.

Firebase повністю охоплює всі етапи розробки програм, а платформа містить усі необхідні інструменти для створення, випуску та здійснення моніторингу програм. Крім того, як завершальний етап у розробці додатків, платформа надає інструменти для залучення та утримання.

Розробники в усьому світі зупиняють свій вибір на Firebase, оскільки вона дає можливість зосередити увагу на створенні коду для фронтенду мобільних додатків. Firebase знижує обсяг розробки шаблонного коду для бекенда, прискорюючи терміни завершення розробки програм. Firebase робить розробку додатків зручною та допомагає зменшити витрати [11].

Використання Firebase також дозволяє розробникам та компаніям стандартизувати середовище розробки бекенду в єдину та просту в освоєнні технологію. Використання шаблонів для бекенда знижують обсяг навчання, необхідного для його підтримки, і дозволяє розробникам, які займаються фронтендом, виконувати більшість дій.

Незважаючи на те, що це фантастична платформа, Firebase має деякі недоліки, наприклад:

1) Не має відкритого вихідного коду
2) Firebase немає у багатьох країнах
3) Доступні лише бази даних NoSQL
4) Повільні запити
5) Не всі служби працюють безкоштовно на базовому тарифі
6) Це не дешева платформа зі складною прогнозованою ціною [11].

2.3 UI/UX дизайн для мобільного додатку

Дизайн мобільних додатків — це створення мобільної версії сайту з додатковими можливостями. При цьому головне завдання розробників полягає у створенні зручної екосистеми із досконалим UX. Приклад зображено на рисунку 2.4

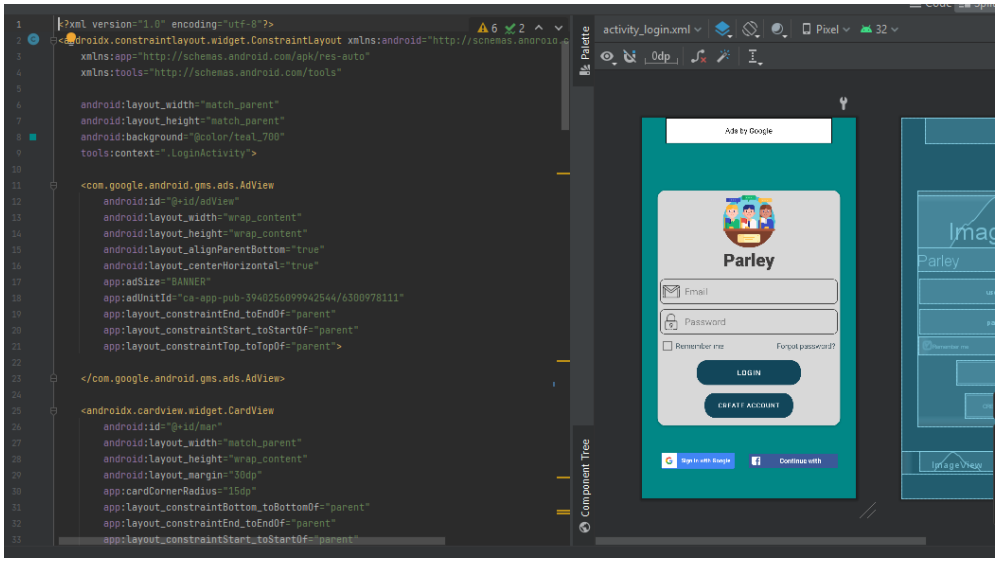


Рисунок 2.4 – Приклад створення дизайну

Завантажуючи будь-який додаток, користувач за умовчанням лояльний: він вже вчинив цільову дію, і якщо він зможе вирішити свою проблему за допомогою сервісу – то буде скористатися ним на регулярній основі. Однак якщо хоча б один розділ на шляху клієнта не працює або незручний – людина просто видалить вашу програму – і не повернеться до неї ніколи.

У цьому полягає принципова відмінність UX-дизайну сайтів і додатків: як правило, користувач оцінює зручність кількох веб-продуктів, і навіть якщо при першому візиті людина не здійснила цільову дію, завжди є ймовірність, що через деякий час вона повернеться на ваш сайт. У цьому повторне скачування додатків — це виняток, ніж поширена практика.

На етапі UX-розробки дизайнер повинен продумати весь шлях користувача від початкових екранів знайомства - і до виконання цільових дій,

яких може бути кілька. Як правило, User Journey є гіллястим деревом можливостей з різним функціоналом: підпискою на додаток, зверненням у службу підтримки, читанням тексту, оплатою товару тощо. І кожна "гілка" має бути добре продумана вже на етапі прототипу.

У проектуванні інтерфейсу є кілька важливих деталей, про які дизайнер та замовник повинні знати:

1) основні елементи управління програмою повинні бути внизу; верхній лівий кут в App дизайні задіюється мінімально і тільки для певних цілей, наприклад, кнопки "Назад", тому що до неї важко тягнутися;

2) керування необов'язково виконується лише за допомогою кнопок; у додатку користувач може скидати або утримувати певні елементи для керування;

3) програми можна скролити як знизу вгору, і справа наліво; якщо ви використовуєте бічний скролл, слід дати користувачеві підказку;

4) логотип не слід дублювати на всіх екранах програми; достатньо того, що ваш бренд буде на іконці сервісу та на екрані завантаження;

5) шрифти не повинні бути необґрунтовано маленькими;

6) кольори повинні бути диверсифіковані і нативно підказувати користувачеві, які елементи клікабельні, а які – ні; також кольором розмежовуються різні за змістом елементи;

7) підказки – це добре; якщо є ймовірність, що користувач кілька разів натискатиме на неклікабельний елемент або присутні нестандартні елементи управління – необхідно додати tooltip.

Після того, як спроектовано інтерфейс, можна переходити до візуальної складової та "вичищати" програму. Основне завдання на цьому етапі полягає в тому, щоб привести дизайн до єдиного Style Guide.

На всю програму бажано використовувати не більше 5 – 6 типів одного шрифту (різного розміру, кольорів та товщини) та 4 – 5 кольорів. Цього достатньо для розставлення акцентів та створення акуратного інтерфейсу.

Також варто продумати анімацію переходу з одного екрану на інший: зникнення, зміщення, прелоадер та ін. квітів.

На фінальному етапі відбувається розробка адаптивних версій. Якщо програма буде доступна тільки для iOS – пощастило, адже на етапі створення адаптивів достатньо відмалювати лише дві версії: для Iphone 6 та Iphone X. Однак якщо завдання – це сервіс для Android, то доведеться відібрати 4 – 5 найпопулярніших дозволів у конкретному сегменті цільової аудиторії.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Інформаційна модель додатку

Враховуючи при розробці особливості структури системи та кожного елементу цієї системи, було обрано створення системи структури наступного типу (рис 3.1):

1. Мобільний додаток на пристрої клієнта.
2. Інтерфейс взаємодії між клієнтом та сервісом Firebase.
3. Взаємодія в середині сервісу Firebase з Google Analytics.
4. Безпосередньо доступ до баз даних
5. Взаємодія додатку з модулем AdMob

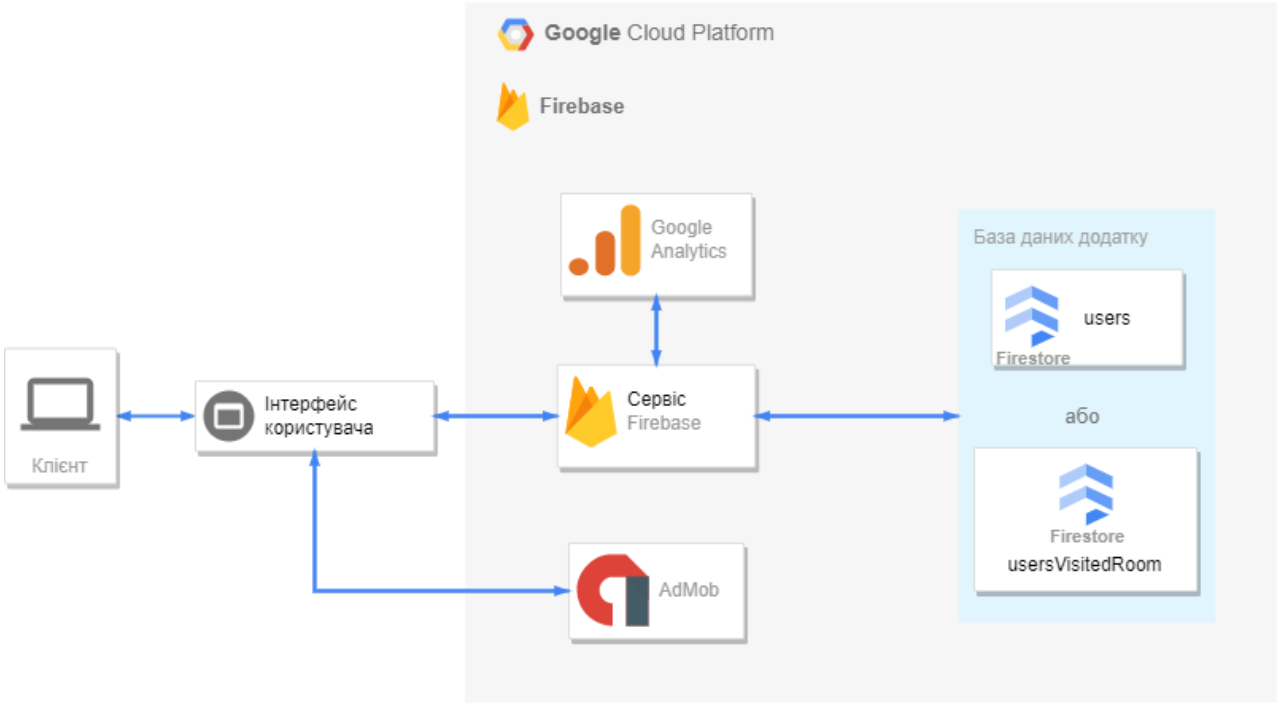


Рисунок 3.1 – Схема взаємодії компонентів додатку

Далі необхідно було вирішити питання зберігання інформації про користувача та його історію відвідувань онлайн конференцій. Оскільки бази даних у сервісу Firebase мають у основі NoSQL структуру, а саме не реляційну структуру збереження інформації у форматі JSON (рис 3.2), було вирішено

створити окремо множину користувачів та окремо множину історії користувача як зображено на рисунках 3.3 та 3.4.


```
{
 "user_base" : {
   "342343" : {
     "email" : "kaushal.xxxxx@gmail.com",
     "authToken" : "some string",
     "name" : "Kaushal",
     "phone" : "+919916xxxxxx",
     "serviceProviderId" : "firebase",
     "signInServiceType" : "google",
   },
   "354895" : {
     "email" : "xxxxx.devil@gmail.com",
     "authToken" : "some string",
     "name" : "devil",
     "phone" : "+919685xxxxxx",
     "serviceProviderId" : "firebase",
     "signInServiceType" : "github"
   },
   "371298" : {
     "email" : "bruce.wayne@wayneinc.com",
     "authToken" : "I am batman",
     "name" : "Bruce Wayne",
     "phone" : "+14085xxxxxx",
     "serviceProviderId" : "firebase",
     "signInServiceType" : "shield"
   }
 },
 "user_prefs": {
   "key1":{
     "data": "for key one"
   },
   "key2":{
     "data": "for key two"
   },
   "key3":{
     "data": "for key three"
   }
 },
 //other structures
```


Рисунок 3.2 – Структура збереження даних у базах даних NoSQL типу у форматі JSON

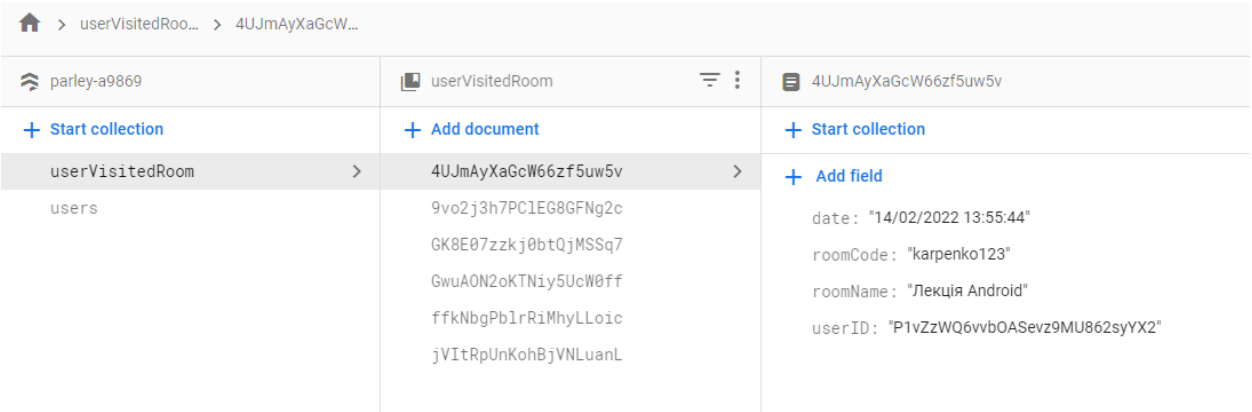


Рисунок 3. 3 – Структура збереження інформації про відвідані зустрічі користувачів

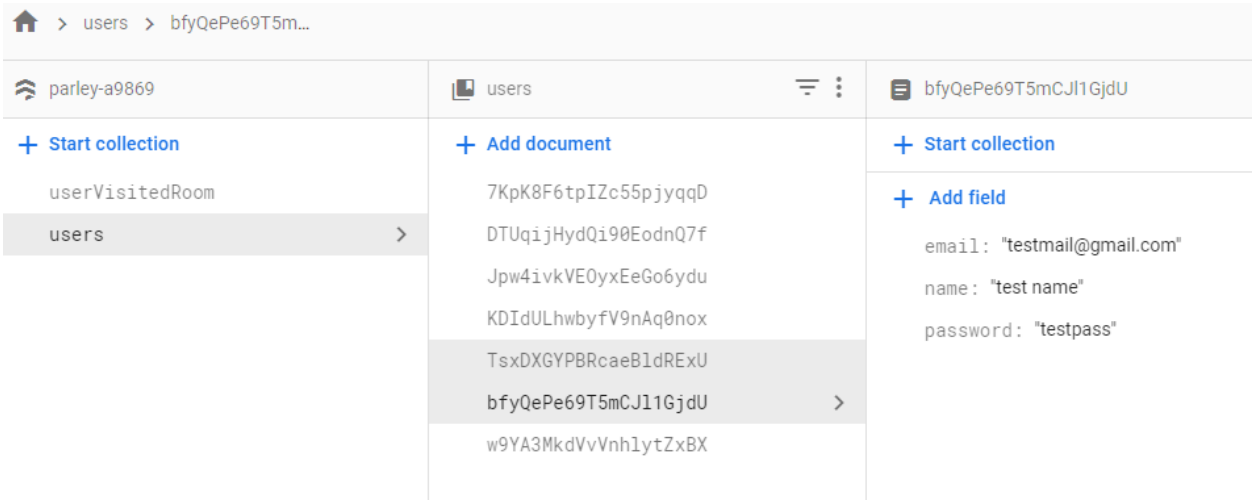


Рисунок 3.4 – Структура збереження інформації про відвідані зустрічі користувачів

3.2 Програмна реалізація

На початку треба створити початковий інтерфейс для реєстрації користувача. Для цього створюємо новий Java файл та розмітку XML для нього так як показано на рисунку 3.5.

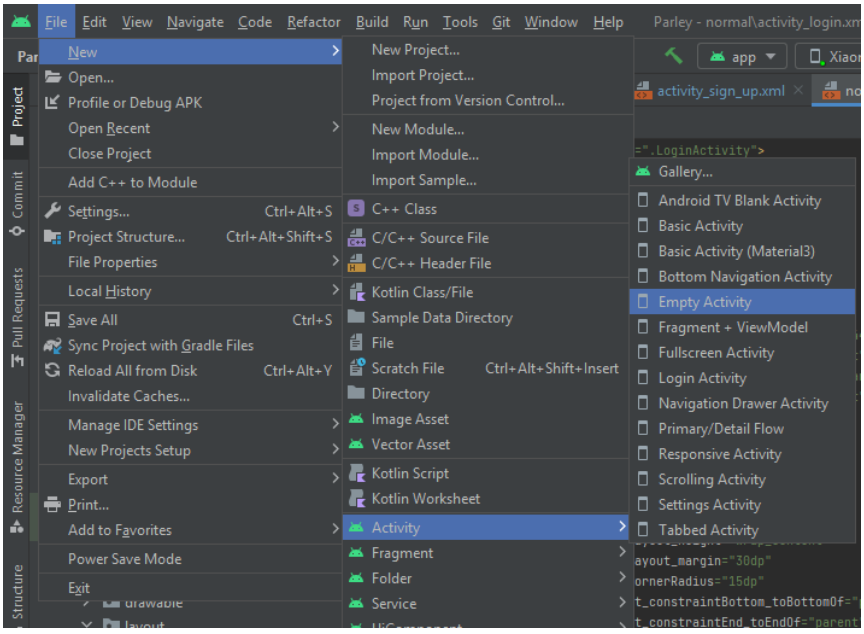


Рисунок 3.5 – Створення нового класу та розмітки для нього

Для побудови екрану авторизації необхідні такі елементи як: Button , TextView, CheckBox, ImageView та EditText.

Button – це кнопка, на головному екрані їх буде дві:

1) Для переходу на головний екран додатку;
2) для переходу на екран відновлення паролю.

EditText – це поле для вводу різноманітної інформації, на головному екрані їх буде два:

1) Для вводу пошти користувача;
2) для вводу пароля користувача.

CheckBox – це прапорець, який дозволяє користувачу вибрати один із стнаів true або false, він буде один для підтвердження швидкої авторизації користувача після закриття додатку.

Загалом екран авторизації має вигляд, як зображено на рисунку 3.6.

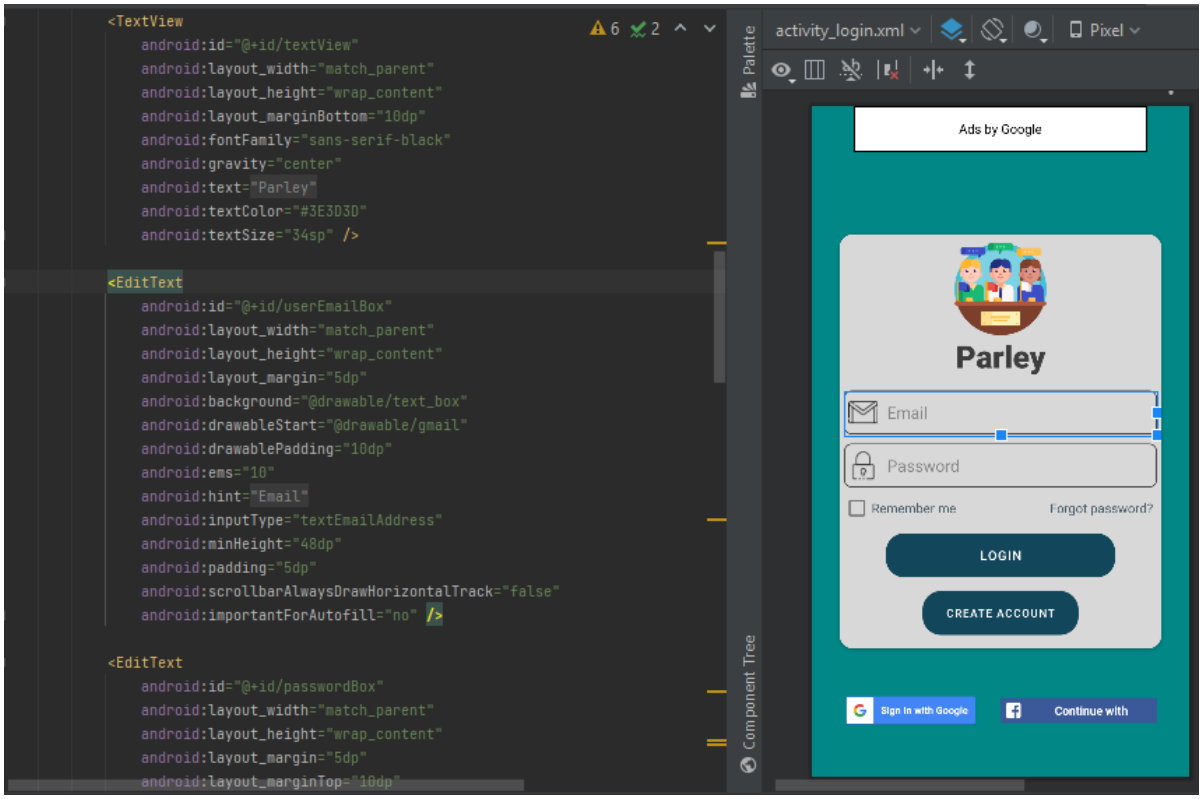


Рисунок 3.6 – Екран авторизації

Далі потрібно розробити екран реєстрації аналогічно екрану авторизації, як зображено на рисунку 3.7.

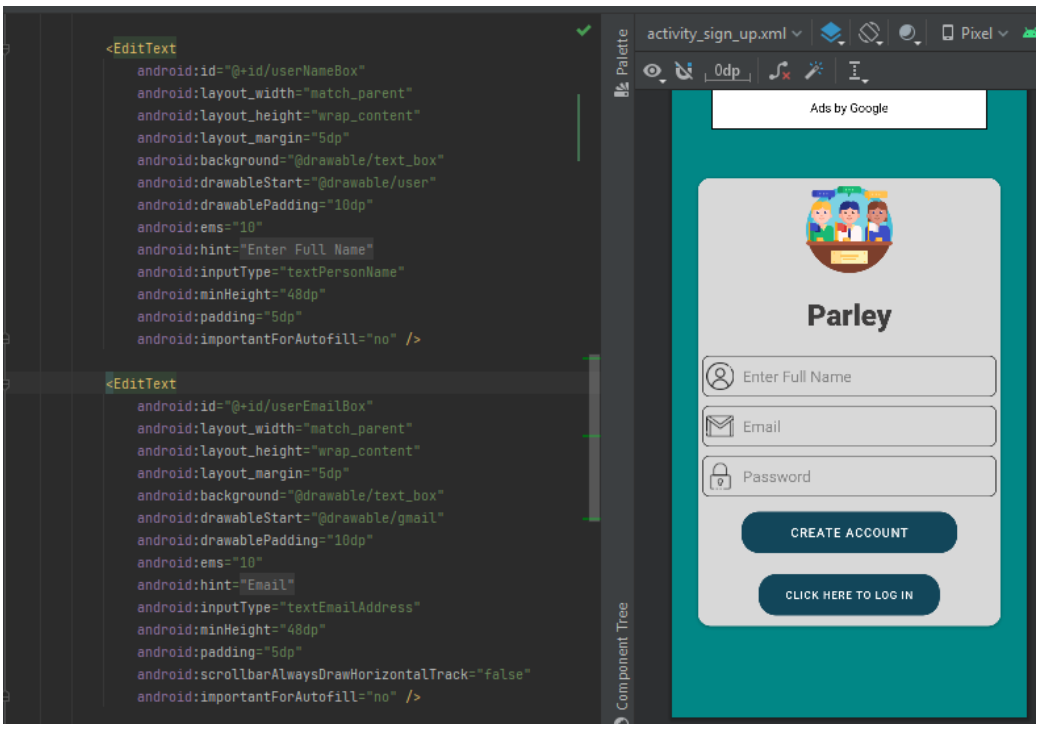


Рисунок 3.7 – Екран реєстрації користувача

Коли користувач зареєструвався або авторизувався, необхідно перенаправити його на головний екран додатку, у якому користувач може під’єднатись до онлайн зустрічі (рис 3.8):

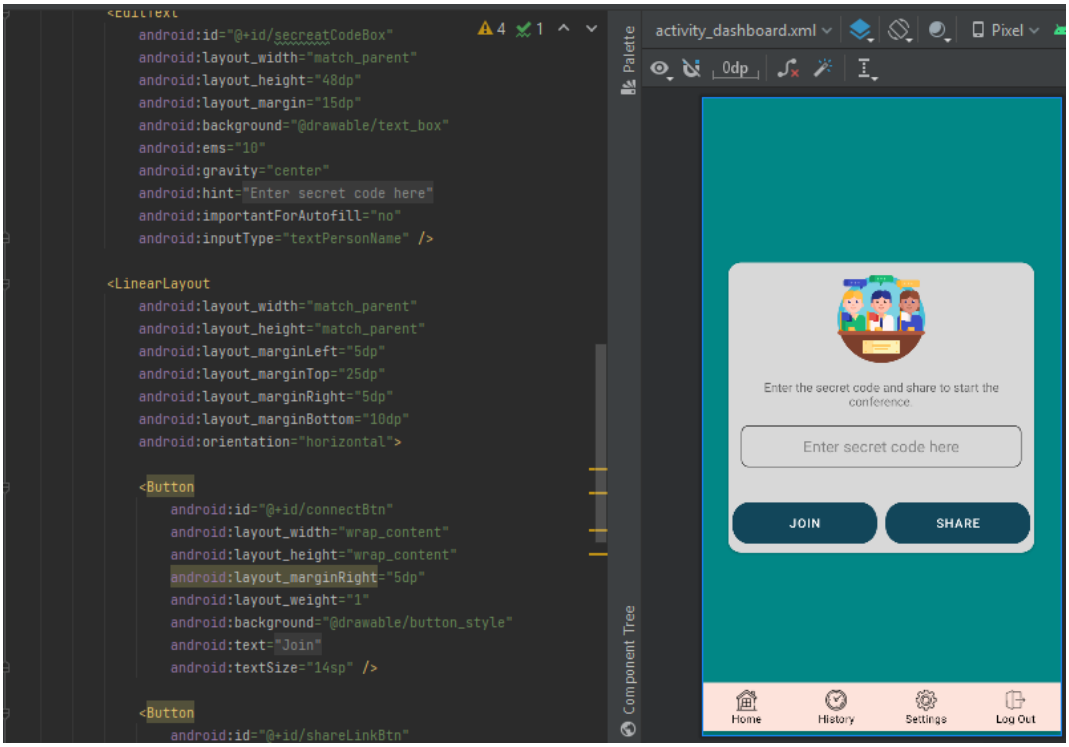


Рисунок 3.8 – Головний екран додатку

Для навігації по додатку було створено меню, як показано на рисунку 3.9.

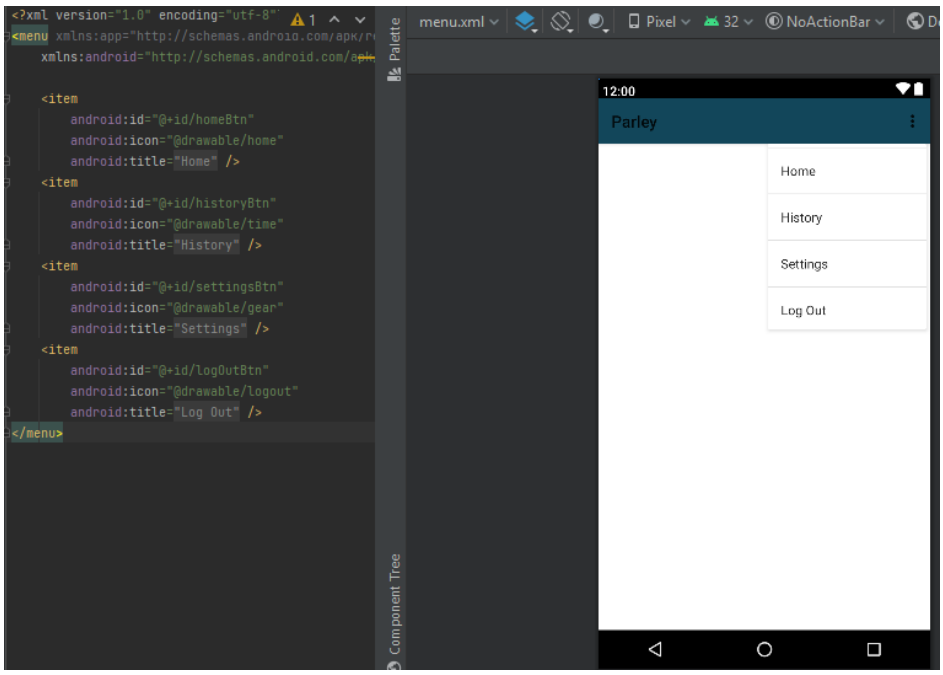


Рисунок 3.9 – Створення меню для навігації по додатку

Для збереження інформації про відвідані онлайн зустрічі було створено екран історії, як показано на рисунку 3.10.

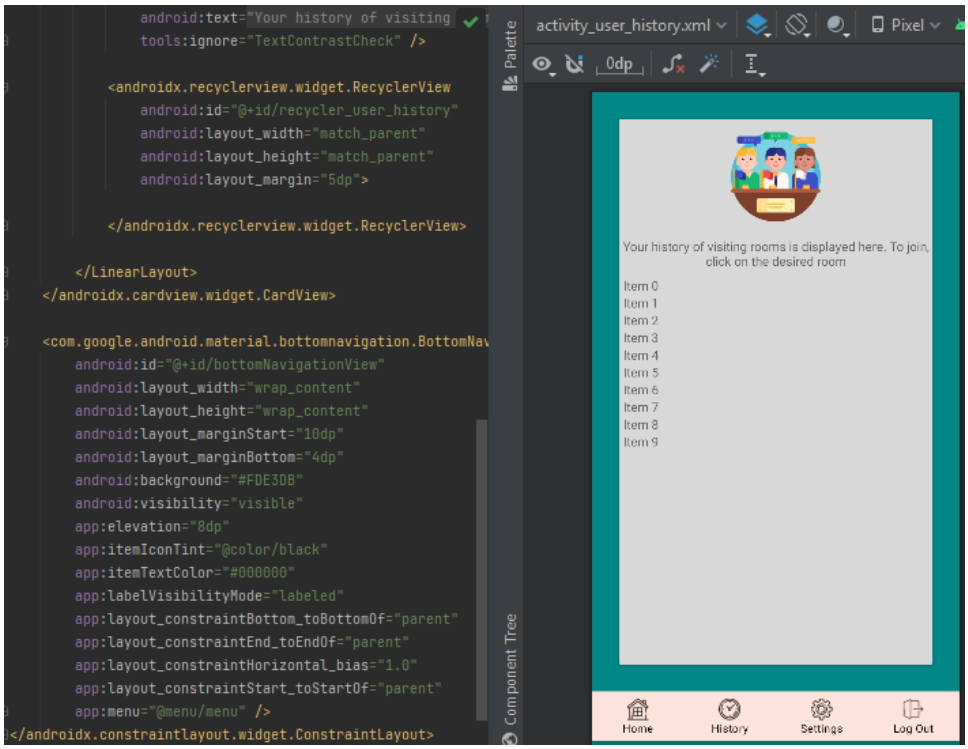


Рисунок 3.10 – Екран з історією відвідувань

Ну і звичайно було створено екран з налаштуваннями користувача з можливістю зміни мови та паролю (рис 3.11).

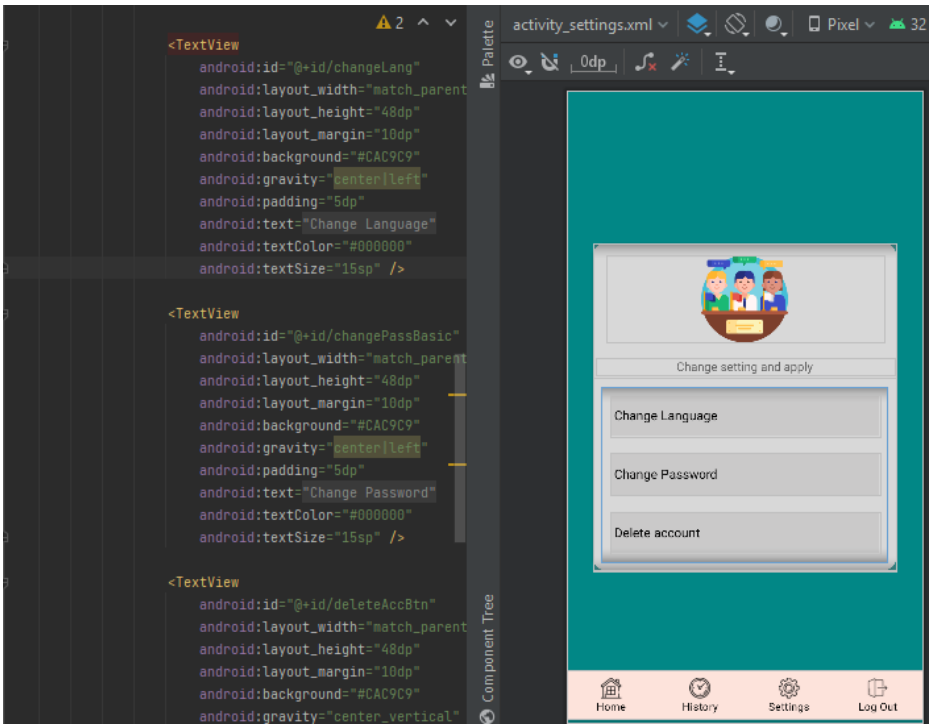


Рисунок 3.11 – Екран налаштувань

Після завершення створення інтерфейсу користувача слід переходити до написання логіки додатку. Для початку треба імпортувати всі необхідні бібліотеки, для цього необхідно відкрити файл Gradle проекту та написати всі залежності як це показано на рисунку 3.12.

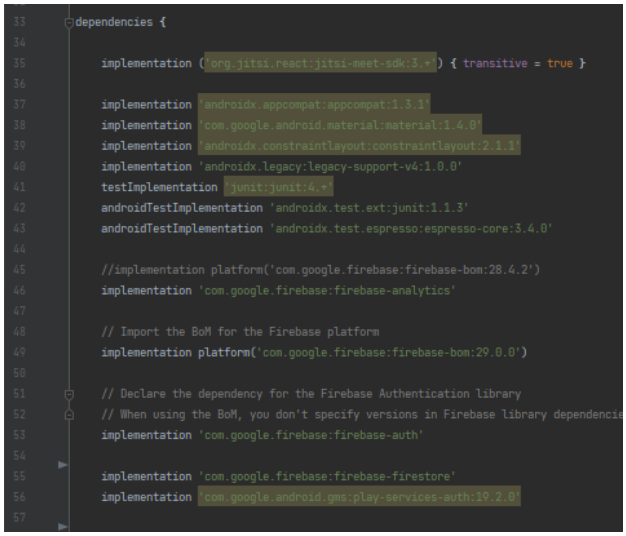


Рисунок 3.12 – Додавання залежностей проєкту

Почнемо писати логіку програми зі сторінки авторизації, для цього нам необхідно зв’язати змінні з розміткою XML, тобто ініціалізувати, для подальшої взаємодії з елементами на екрані (див. дод. А).

Для авторизації користувача за допомогою пошти та паролю було зроблено метод, який приймає на вхід дані введені в поля, та передає ці значення у функцію об’єкта FirebaseAuth auth (рис. 3.13).

```
private void authWithMailAndPass() {
    String email, password;
    email = emailBox.getText().toString();
    password = passwordBox.getText().toString();

    auth.signInWithEmailAndPassword(email, password).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            if (auth.getCurrentUser().isEmailVerified()) {
                startActivity(new Intent( packageContext: LoginActivity.this, DashboardActivity.class));
                Toast.makeText( context: LoginActivity.this, "Success", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: LoginActivity.this, "Check email and verify account", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText( context: LoginActivity.this, task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
```

Рисунок 3.13 – Авторизація користувача через пошту та пароль

Додаток також має можливість авторизувати користувача за допомогою його пошти. Впровадження цього способу важча ніж звичайна реєстрація, але за допомогою документації все вдалось вирішити (див. дод. А)

З впровадженням авторизації через токен Facebook були найбільші труднощі. Сервіс не дозволяє використовувати своє програмне забезпечення без двух речей:

1) Встановленого додатку Facebook на смартфоні;

2) Без завантаженої на сайт Facebook for Developers політики конфіденційності.

Якщо спробувати авторизуватись за допомогою Facebook, але у вас не буде встановлено додатку на смартфоні, або у програми яку ви використовуєте не буде відповідної до правил компанії Facebook завантаженої на сайт для

розробників політики конфіденційності, ви отримаєте повідомлення про помилку яке зображено на рисунку 3.14.

Рисунок 3.14 – Проблеми з авторизацією за допомогою Facebook

Сама ж реалізація авторизації через Facebook досить проста та зображена на рисунках 3.15 та 3.16.

```
private void authWithFacebook() {
    LoginManager.getInstance().logInWithReadPermissions( activity: LoginActivity.this, Arrays.asList("public_profile"));
    LoginManager.getInstance().registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) {
            Toast.makeText( context: LoginActivity.this, "Success", Toast.LENGTH_SHORT).show();
            handleFacebookToken(loginResult.getAccessToken());
        }

        @Override
        public void onCancel() {

        }

        @Override
        public void onError(FacebookException error) {

        }
    });
}
```

Рисунок 3.15 – Авторизація за допомогою Facebook

```
private void handleFacebookToken(AccessToken accessToken) {
    AuthCredential credential = FacebookAuthProvider.getCredential(accessToken.getToken());
    auth.signInWithCredential(credential).addOnCompleteListener( activity: LoginActivity.this, task -> {
        if (task.isSuccessful()) {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            updateUI(user);
        } else {
            Toast.makeText( context: LoginActivity.this, task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
```

Рисунок 3.16 – Авторизація за допомогою Facebook

Для того щоб користувач не авторизувався по декілька разів було створено CheckBox, натискаючи на який користувач може або дозволити зберігати авторизованним аккаунт, або ні. Для цього було використано– SharedPreferences.

SharedPreferences — це постійне сховище на платформі Android, яке використовується додатками, наприклад, для зберігання своїх налаштувань. Цей репозиторій є відносно постійним, користувач може зайти в налаштування програми та очистити дані програми, таким чином очистивши всі дані в сховищі.

Коли користувач натискає на значок, у SharedPreferences записується значення чекбоксу true, і навпаки, коли чекбокс знятий, записується значення false. У методі onStart() при запуску додатку перевіряється стан у SharedPreferences, і відповідно до результату перенаправляє користувача на головну, або ні (див. дод. А).

Навігація у додатку між реєстрацією та авторизацією виконується за допомогою лямбда виклику методу onClickListener(), який відслідковує натиск на кнопку (рис 3.17).

```
logInBtn.setOnClickListener(v -> {
    startActivity(new Intent( packageContext: SignUpActivity.this, LoginActivity.class));
    Bungee.slideRight( context: SignUpActivity.this);
});
```

Рисунок 3.17 – Реалізація навігації між окремими Activity

Для створення нового користувача було створено метод, де у об’єкта FirebaseAuth auth було викликано функцію для створення користувача (див. дод. А).

Також у методі використовується запит до бази даних Firebase, під назвою FirebaseFirestore. Цим запитом інформація про користувача записується до відповідної множини, завдяки тому що база даних FirebaseFirestore набагато гнучкіша за звичайні реляційні бази, у майбутньому

не виникне труднощів у розширені збору інформації про користувача, наприклад про його день народження, для привітання.

На екрані відновлення паролю користувач може ввести адрес електронної пошти, після чого отримати повідомлення з проханням перейти за посиланням та встановити новий пароль, після чого повернутись у додаток та спробувати авторизуватись знову.

Налаштування всередині додатку досить скромні, але достатні для такого рівня, оскільки мають все необхідне, а саме видалення аккаунту, зміну паролю та зміну мови інтерфейсу. Для реалізації зміни паролю було використано функцію об’єкта FirebaseAuth, для цього ми дізнавались поточного користувача за допомогою методу getCurrentUser(), та викликали у нього метод updatePassword(), як це показано на рисунку 3.18.

```
    changePassBtn.setOnClickListener(v -> {

        String newPass;
        newPass = passwordNew.getText().toString();
        user.updatePassword(newPass)
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        startActivity(new Intent( packageContext: ChangePassActivity.this, LoginActivity.class));
                        Bungee.slideRight( context: ChangePassActivity.this);
                        finishAffinity();
                    } else {
                        Toast.makeText( context: ChangePassActivity.this, task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
    });
}
```

Рисунок 3.18 – Зміна паролю користувача

Для того щоб користувач міг змінити мову інтерфейсу було використано рані згадуване локальне сховище SharedPreferences (рис 3.19).

```
private void setLocate(String language) {

    Locale locale = new Locale(language);
    Locale.setDefault(locale);
    Configuration configuration = new Configuration();
    configuration.locale = locale;
    getBaseContext().getResources().updateConfiguration(configuration,getBaseContext().getResources().getDisplayMetrics());

    SharedPreferences.Editor editor = getSharedPreferences( name: "Settings", MODE_PRIVATE).edit();
    editor.putString( s: "My_lang", language);
    editor.apply();

}
```

Рисунок 3.19 – Встановлення локалізації у додатку

Також для того щоб зміна мови відбулась, усі строкові змінні які фігурують на екрані користувача, треба занести до змінних у папці strings.xml (див. дод. А), та створити копію цього файлу, але вже на інших мовах. Таким чином ми отримуємо список зафіксованих змінних з відповідним текстом.

При натисканні на кнопку ChangeLanguage на екрані користувача з’являється діалогове вікно, у якому вказані можливі мови інтерфейсу, це можна реалізувати наступним способом, зображеному на рисунку 3.20.

```
private void showChangeLanguageDialog() {
    final String[] listItems = {"English", "Українська"};
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Chose Language");
    builder.setSingleChoiceItems(listItems,  checkedItem: -1, (dialog, which) -> {
        if (which == 0) {
            setLocate("en");
            recreate();
        } else if (which == 1) {
            setLocate("uk");
            recreate();
        }
        dialog.dismiss();

    });
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

Рисунок 3.20 – Виклик діалогу зміни мови інтерфейсу

Видалення аккаунту користувача виконано за допомогою об’єкта FirebaseAuth та виклику методу delete() у поточного користувача (рис 3.21).

```
deleteAcc.setOnClickListener(v -> {
    AlertDialog.Builder dialog = new AlertDialog.Builder( context: SettingsActivity.this);
    dialog.setTitle("Are you sure?");
    dialog.setMessage("Remove your account from the system?");
    dialog.setPositiveButton("Yes", (dialog1, which) -> user.delete().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            startActivity(new Intent( packageContext: SettingsActivity.this, LoginActivity.class));
            Bungee.slideRight( context: SettingsActivity.this);
            finishAffinity();
        } else {
            Toast.makeText( context: SettingsActivity.this, task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
        }
    }));
    dialog.setNegativeButton("No", (dialog12, which) -> dialog12.dismiss());
    AlertDialog alertDialog = dialog.create();
    alertDialog.show();
});
```

Риснунок 3.21 – Видалення акканута користувача

На головному екрані додатка ми маємо поле вводу ключа кімнати та дві кнопки : поділитись кодом та під’єднатись до зустрічі, реалізація зображена у Додатку А – клас HistoryAdapter.java.

Найголовніше, що було виконано при створенні додатку – це реалізація виведення інформації про відвідані зустрічі користувачем у спеціальний список RecyclerView. Це продвинута версія звичайного списку ListView. Величезний плюс у використанні списку RecyclerView у тому що він не забиває пам’ять пристрою попередньо вкладеною інформацією як ListView, оскільки видаляє з пам’яті об’єкти які зникли з екрану (рис 3.22).

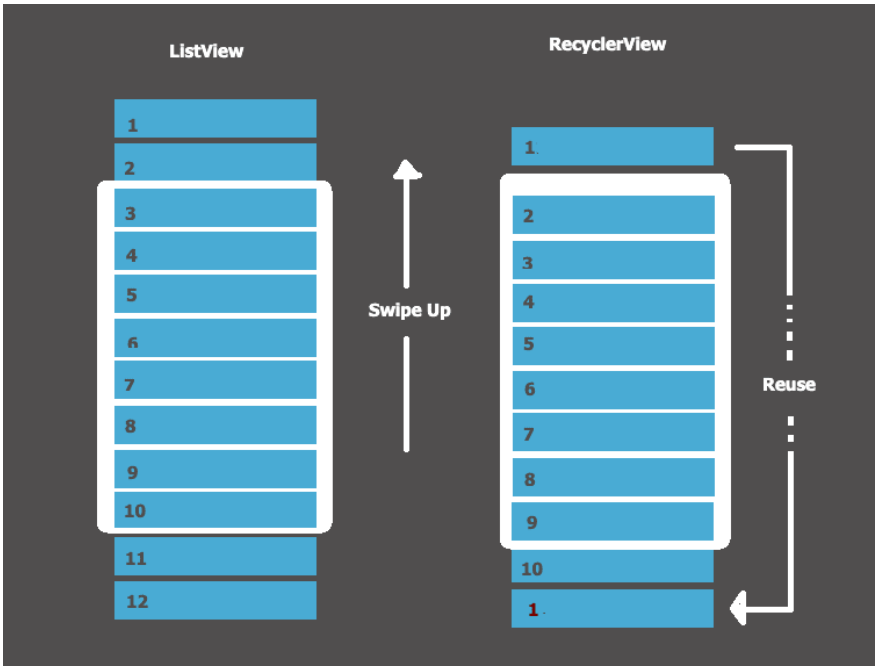


Рисунок 3.22 – Наглядна різниця роботи ListView та RecyclerView

До мінусів використання такого списку є його величезна складність у реалізації. Демонстрація одного з методів відображення представлена у Додатку А – клас HistoryAdapter.java.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було виконано:

1) літературний огляд сучасних джерел за тематикою розробки мобільних додатків на основі чого було сформовано задачі для виконання.

2) було розглянуто доступні і обрано оптимальні інструменти для розробки мобільних додатків;

3) було сформовано вимоги до мобільного додатку.

4) окремо ознайомлено з основними інструментами та вдосконаленням їх використання;

5) інтегровано та виконано всі задачі яки стояли перед готовим продуктом;

6) проведено мануальне тестування та тестування у реальних умовах для перевірки готового функцілоналу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Зростання актуальності відеоконференцзв'язку в пандемію [Електронний ресурс] - Режим доступу до ресурсу: https://www.itweek.com/infrastructure/article/detail.php?ID=221286&utm_source=theme-articles-right&utm_medium=link

2. Розробка мобільних програм від А до Я: повний гайд [Електронний ресурс] - Режим доступу до ресурсу : https://dan-it.com.ua/blog/razrabotka-mobilnyh-prilozhenij-ot-a-do-ja-polnyj-gajd/

3. “Hello World” in 20 Programming Languages [Електронний ресурс] - Режим доступу до ресурсу : https://www.hongkiat.com/blog/hello-world-different-programming-languages/

4. Android Studio: переваги та особливості [Електронний ресурс] - Режим доступу до ресурсу : https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/

5. Java vs Kotlin для Android-розробки: відповіді «за» та «проти» [Електронний ресурс] - Режим доступу до ресурсу :

https://tproger.com/articles/java-vs-kotlin/

6. Мова програмування Java: особливості, популярність, ситуація на ринку праці [Електронний ресурс] - Режим доступу до ресурсу :

https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda#zarplata-java-programmistov

7. TIOBE Index for May 2022 [Електронний ресурс] - Режим доступу до ресурсу : https://www.tiobe.com/tiobe-index/

8. The RedMonk Programming Language Rankings: January 2022 [Електронний ресурс] - Режим доступу до ресурсу :

https://redmonk.com/sogrady/2022/03/28/language-rankings-1-22/

9. Android SDK [Електронний ресурс] - Режим доступу до ресурсу : https://www.techopedia.com/definition/4220/android-sdk

10. Що таке Firebase і чому варто з цим познайомитись [Електронний ресурс] - Режим доступу до ресурсу : https://kolmogorov.pro/what-is-firebase-chto-takoe

11. Що таке Firebase? Розкриваємо всі таємниці [Електронний ресурс] - Режим доступу до ресурсу :

https://blog.back4app.com/ua/%D1%87%D1%82%D0%BE%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-firebase/

12. Документація Firebase [Електронний ресурс] - Режим доступу до ресурсу : https://firebase.google.com/docs?authuser=0&hl=en

13. Android База даних Firebase Realtime [Електронний ресурс] - Режим доступу до ресурсу: https://learntutorials.net/ua/android/topic/5511/

14. Draw.io [Електронний ресурс] - Режим доступу до ресурсу: https://app.diagrams.net/

15. Firebase [Електронний ресурс] - Режим доступу до ресурсу: https://console.firebase.google.com/u/0/

16. Android Developer [Електронний ресурс] - Режим доступу до ресурсу: https://developer.android.com/

ДОДАТКИ

Додаток А. Лістинг коду програми

```
package karpenko.test.parley;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import spencerstudios.com.bungeelib.Bungee;

public class ChangePassActivity extends AppCompatActivity {

    private EditText passwordNew;
    private Button changePassBtn;
    private FirebaseUser user;

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

        passwordNew = findViewById(R.id.newPassBox);
        changePassBtn = findViewById(R.id.changePassBtn);
        user = FirebaseAuth.getInstance().getCurrentUser();

        changePassBtn.setOnClickListener(v -> {

            String newPass;
            newPass = passwordNew.getText().toString();
            user.updatePassword(newPass)
                    .addOnCompleteListener(task -> {
                        if (task.isSuccessful()) {
                            startActivity(new Intent(ChangePassActivity.this,
LoginActivity.class));
                            Bungee.slideRight(ChangePassActivity.this);
                            finishAffinity();
                        } else {
                            Toast.makeText(ChangePassActivity.this,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
                        }
                    });
        });
    }

    @Override
    public void onBackPressed() {
        startActivity(new
Intent(ChangePassActivity.this,SettingsActivity.class));
        Bungee.slideRight(ChangePassActivity.this);
    }

}
```

```
package karpenko.test.parley;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import spencerstudios.com.bungeelib.Bungee;

public class ChangePassByEmail extends AppCompatActivity {

    EditText userEmail;
    Button sentOnEmail;
    FirebaseAuth auth;

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

        userEmail = findViewById(R.id.userEmail);
        sentOnEmail = findViewById(R.id.changePassBtn);
        auth = FirebaseAuth.getInstance();

        sentOnEmail.setOnClickListener(v -> {

            String email;

            email = userEmail.getText().toString();

            auth.sendPasswordResetEmail(email)
                    .addOnCompleteListener(task -> {
                        if (task.isSuccessful()) {
                            Toast.makeText(ChangePassByEmail.this,
getString(R.string.check_email), Toast.LENGTH_SHORT).show();
                            startActivity(new
Intent(ChangePassByEmail.this,LoginActivity.class));
                            Bungee.slideRight(ChangePassByEmail.this);
                            finish();
                        }
                    });

        });

    }

    @Override
    public void onBackPressed() {
        startActivity(new
Intent(ChangePassByEmail.this,LoginActivity.class));
        Bungee.slideRight(ChangePassByEmail.this);
        finish();
    }
}
```

```
package karpenko.test.parley;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdSize;
import com.google.android.gms.ads.AdView;
import com.google.android.material.bottomnavigation.BottomNavigationItemView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import org.jitsi.meet.sdk.JitsiMeet;
import org.jitsi.meet.sdk.JitsiMeetActivity;
import org.jitsi.meet.sdk.JitsiMeetConferenceOptions;

import java.net.MalformedURLException;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import spencerstudios.com.bungeelib.Bungee;

public class DashboardActivity extends AppCompatActivity {

    private EditText secretCode;
    private Button connectBtn, shareBtn;
    private BottomNavigationItemView settings, history, logOut;
    private FirebaseFirestore firestore;
    private FirebaseAuth auth;
    private String userID, roomCode;
    private SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
    private Date date = new Date();
    private AdView mAdView;


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

        AdView adView = new AdView(this);
        adView.setAdSize(AdSize.BANNER);
        adView.setAdUnitId("ca-app-pub-3940256099942544/6300978111");
        mAdView = findViewById(R.id.adView);
        AdRequest adRequest = new AdRequest.Builder().build();
        mAdView.loadAd(adRequest);

        secretCode = findViewById(R.id.secreatCodeBox);
        connectBtn = findViewById(R.id.connectBtn);
        shareBtn = findViewById(R.id.shareLinkBtn);
        settings = findViewById(R.id.settingsBtn);
        history = findViewById(R.id.historyBtn);
        logOut = findViewById(R.id.logOutBtn);
```

```
        firestore = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        jitsiConferenceOptions();

        connectBtn.setOnClickListener(v -> connectUserToRoomAndSaveRoomId());

        settings.setOnClickListener(v ->{
            startActivity(new Intent(DashboardActivity.this,
SettingsActivity.class));
            Bungee.slideLeft(DashboardActivity.this);
        });

        history.setOnClickListener(v -> {
            startActivity(new Intent(DashboardActivity.this,
UserHistory.class));
            Bungee.slideLeft(DashboardActivity.this);
        });

        logOut.setOnClickListener(v -> logOutUser());

        shareBtn.setOnClickListener(v -> shareSecretCode());
    }

    private void logOutUser() {
        SharedPreferences preferences = getSharedPreferences("checkbox",
MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("remember", "false");
        editor.apply();
        startActivity(new Intent(DashboardActivity.this,
LoginActivity.class));
        Bungee.slideRight(DashboardActivity.this);
        FirebaseAuth.getInstance().signOut();
        finishAffinity();
    }

    private void shareSecretCode() {
        String code;
        code = secretCode.getText().toString();

        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_TEXT, "Hey! Join to my video chat by
this code : " + code +
                ".\n Enter it in app : https://play.google.com/ ");
        startActivity(Intent.createChooser(intent, "Share to your friends"));
    }

    private void connectUserToRoomAndSaveRoomId() {

        JitsiMeetConferenceOptions options = new
JitsiMeetConferenceOptions.Builder().setRoom(secretCode.getText().toString())
                .setWelcomePageEnabled(false).build();

        JitsiMeetActivity.launch(DashboardActivity.this, options);

        userID = auth.getCurrentUser().getUid();
        roomCode = secretCode.getText().toString();

        DocumentReference documentReference =
firestore.collection("userVisitedRoom").document();
        Map<String, Object> userHistory = new HashMap<>();
        userHistory.put("userID", userID);
```

```
        userHistory.put("roomName", "Default Name");
        userHistory.put("roomCode", roomCode);
        userHistory.put("date", formatter.format(date));
        documentReference.set(userHistory).addOnSuccessListener(unused ->
Toast.makeText(DashboardActivity.this, getString(R.string.code_saved),
Toast.LENGTH_SHORT).show());

    }

    private void jitsiConferenceOptions() {

        URL url;
        try {
            url = new URL("https://meet.jit.si");

            JitsiMeetConferenceOptions jitsiMeetConferenceOptions = new
JitsiMeetConferenceOptions.Builder()

.setServerURL(url).setWelcomePageEnabled(false).setAudioMuted(true).setVideoM
uted(true).build();

JitsiMeet.setDefaultConferenceOptions(jitsiMeetConferenceOptions);

        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
    }

}


package karpenko.test.parley;

public class History {

    String roomCode, roomName,date;


    public History(){};

    public History(String roomCode, String roomName, String date) {
        this.roomCode = roomCode;
        this.roomName = roomName;
        this.date = date;
    }

    public String getRoomCode() {
        return roomCode;
    }

    public void setRoomCode(String roomCode) {
        this.roomCode = roomCode;
    }

    public String getRoomName() {
        return roomName;
    }

    public void setRoomName(String roomName) {
        this.roomName = roomName;
    }

    public String getDate() {
        return date;
```

```
    }

    public void setDate(String date) {
        this.date = date;
    }
}


package karpenko.test.parley;

import android.content.Context;
import android.text.InputType;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.PopupMenu;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.WriteBatch;

import org.jitsi.meet.sdk.JitsiMeet;
import org.jitsi.meet.sdk.JitsiMeetActivity;
import org.jitsi.meet.sdk.JitsiMeetConferenceOptions;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class HistoryAdapter extends
RecyclerView.Adapter<HistoryAdapter.HistoryViewHolder> {

    Context context;
    ArrayList<History> HistoryArrayList;

    public HistoryAdapter(Context context, ArrayList<History>
HistoryArrayList) {
        this.context = context;
        this.HistoryArrayList = HistoryArrayList;
    }

    @NonNull
    @Override
    public HistoryAdapter.HistoryViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {

        View view =
LayoutInflater.from(context).inflate(R.layout.user_history_single_item,
parent, false);

        return new HistoryViewHolder(view);
    }

    @Override
```

```
    public void onBindViewHolder(@NonNull HistoryAdapter.HistoryViewHolder
holder, int position) {

        History history = HistoryArrayList.get(position);

        holder.roomName.setText(history.roomName);
        holder.roomCode.setText(history.roomCode);
        holder.roomDate.setText(String.valueOf(history.date));

        holder.itemView.setOnClickListener(v -> {
            jitsiConferenceOptions();
            JitsiMeetConferenceOptions options = new
JitsiMeetConferenceOptions.Builder().setRoom(holder.roomCode.getText().toStri
ng())
                    .setWelcomePageEnabled(false).build();

            JitsiMeetActivity.launch(context, options);
        });

        holder.options.setOnClickListener(v -> {
            PopupMenu popupMenu = new PopupMenu(context, holder.options);
            popupMenu.inflate(R.menu.history_recyclerview_menu);

            popupMenu.setOnMenuItemClickListener(item -> {
                if (item.getItemId() == R.id.menu_remove) {
                    AlertDialog.Builder dialog = new
AlertDialog.Builder(context);
                    dialog.setTitle(R.string.are_u_sure);

dialog.setMessage(R.string.remove_info_about_visited_room);
                    dialog.setPositiveButton(R.string.yes, (dialog13, which)
-> FirebaseFirestore.getInstance()
                            .collection("userVisitedRoom")
                            .whereEqualTo("roomCode", history.roomCode).get()
                            .addOnSuccessListener(queryDocumentSnapshots -> {
                                WriteBatch b =
FirebaseFirestore.getInstance().batch();
                                List<DocumentSnapshot> s =
queryDocumentSnapshots.getDocuments();
                                for (DocumentSnapshot snapshot : s) {
                                    b.delete(snapshot.getReference());
                                }
                                b.commit().addOnSuccessListener(unused ->
Toast.makeText(context, R.string.removed, Toast.LENGTH_SHORT).show());
                            }));
                    dialog.setNegativeButton(R.string.close, (dialog12,
which) -> dialog12.dismiss());
                    AlertDialog alertDialog = dialog.create();
                    alertDialog.show();
                } else if (item.getItemId() == R.id.menu_edit) {
                    AlertDialog.Builder dialog = new
AlertDialog.Builder(context);
                    dialog.setTitle(R.string.set_new_room_name);
                    EditText editText = new EditText(context);
                    editText.setInputType(InputType.TYPE_CLASS_TEXT);
                    dialog.setView(editText);

                    dialog.setPositiveButton(R.string.ok, (dialog1, which) ->
{
                        String newRoomName = editText.getText().toString();
                        String roomCode = history.roomCode;

                        UpdateData(roomCode, newRoomName);
                    });
```

```
                    dialog.setNegativeButton(R.string.close, (dialog12,
which) -> dialog12.dismiss());
                    AlertDialog alertDialog = dialog.create();
                    alertDialog.show();

                }
                return true;

            });

            popupMenu.show();
        });
    }

    private void UpdateData(String roomCode, String newRoomName) {

        Map<String, Object> newName = new HashMap<>();
        newName.put("roomName", newRoomName);


FirebaseFirestore.getInstance().collection("userVisitedRoom").whereEqualTo("r
oomCode", roomCode)
                .get().addOnCompleteListener(task -> {
            if (task.isSuccessful() && !task.getResult().isEmpty()) {

                DocumentSnapshot documentSnapshot =
task.getResult().getDocuments().get(0);
                String documentID = documentSnapshot.getId();
                FirebaseFirestore.getInstance().collection("userVisitedRoom")
                        .document(documentID).update(newName)
                        .addOnSuccessListener(unused ->
Toast.makeText(context, R.string.updated, Toast.LENGTH_SHORT).show())
                        .addOnFailureListener(e -> Toast.makeText(context,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show());

            } else {
                Toast.makeText(context,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
            }
        });

    }

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

    public static class HistoryViewHolder extends RecyclerView.ViewHolder {

        TextView roomName, roomCode, roomDate, options;


        public HistoryViewHolder(@NonNull View itemView) {
            super(itemView);

            roomName = itemView.findViewById(R.id.tvRoomName);
            roomCode = itemView.findViewById(R.id.tvRoomCode);
            roomDate = itemView.findViewById(R.id.tvRoomVisited);
            options = itemView.findViewById(R.id.tvOptions);

        }
    }
```

```
    private void jitsiConferenceOptions() {

        URL url;
        try {
            url = new URL("https://meet.jit.si");

            JitsiMeetConferenceOptions jitsiMeetConferenceOptions = new
JitsiMeetConferenceOptions.Builder()

.setServerURL(url).setWelcomePageEnabled(false).setAudioMuted(true).setVideoM
uted(true).build();

JitsiMeet.setDefaultConferenceOptions(jitsiMeetConferenceOptions);

        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
    }

}


package karpenko.test.parley;

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.facebook.AccessToken;
import com.facebook.CallbackManager;
import com.facebook.FacebookCallback;
import com.facebook.FacebookException;
import com.facebook.login.LoginManager;
import com.facebook.login.LoginResult;
import com.facebook.login.widget.LoginButton;
import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdSize;
import com.google.android.gms.ads.AdView;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.FacebookAuthProvider;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.GoogleAuthProvider;

import java.util.Arrays;
```

```
import spencerstudios.com.bungeelib.Bungee;

public class LoginActivity extends AppCompatActivity {

    private EditText emailBox, passwordBox;
    private TextView changePass;
    private Button loginBtn, createAccountBtn;
    private FirebaseAuth auth;
    private CheckBox rememberMe;
    private ImageView googleSignIn;
    private GoogleSignInClient mGoogleSignInClient;
    private FirebaseAuth mAuth;
    private CallbackManager callbackManager;
    private LoginButton facebookSignIn;
    private AdView mAdView;

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

        AdView adView = new AdView(this);
        adView.setAdSize(AdSize.BANNER);
        adView.setAdUnitId("ca-app-pub-3940256099942544/6300978111");
        mAdView = findViewById(R.id.adView);
        AdRequest adRequest = new AdRequest.Builder().build();
        mAdView.loadAd(adRequest);

        emailBox = findViewById(R.id.userEmailBox);
        passwordBox = findViewById(R.id.passwordBox);
        loginBtn = findViewById(R.id.loginBtn);
        changePass = findViewById(R.id.changePass);
        createAccountBtn = findViewById(R.id.createAccBtn);
        auth = FirebaseAuth.getInstance();
        rememberMe = findViewById(R.id.rememderMe);
        googleSignIn = findViewById(R.id.signInWithGoogle);
        mAuth = FirebaseAuth.getInstance();
        facebookSignIn = findViewById(R.id.signInWithFacebook);
        callbackManager = CallbackManager.Factory.create();

        rememberOrNot();

        facebookSignIn.setOnClickListener(v -> {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            if (user != null) {
                authWithFacebook();
            } else {
                AlertDialog.Builder dialog = new
AlertDialog.Builder(LoginActivity.this);
                dialog.setTitle(getString(R.string.attention));
                dialog.setMessage(R.string.facebook_login_info);
                dialog.setPositiveButton(R.string.ok, (dialog1, which) ->
authWithFacebook());
                dialog.setNegativeButton(R.string.close, (dialog12, which) ->
dialog12.dismiss());
                AlertDialog alertDialog = dialog.create();
                alertDialog.show();
            }
        });

        loginBtn.setOnClickListener(v -> authWithMailAndPass());

        createAccountBtn.setOnClickListener(v -> {
```

```
            startActivity(new Intent(LoginActivity.this,
SignUpActivity.class));
            Bungee.slideLeft(LoginActivity.this);
        });

        changePass.setOnClickListener(v -> {
            startActivity(new Intent(LoginActivity.this,
ChangePassByEmail.class));
            Bungee.slideLeft(LoginActivity.this);
        });

        rememberMe.setOnCheckedChangeListener((buttonView, isChecked) -> {
            if (buttonView.isChecked()) {
                rememberUser();
            } else if (!buttonView.isChecked()) {
                forgotUser();
            }
        });

        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken("141783523039-
iersgh9r01kraj1qajnl063ji01essvq.apps.googleusercontent.com")
                .requestEmail()
                .build();
        mGoogleSignInClient = GoogleSignIn.getClient(LoginActivity.this,
gso);
        googleSignIn.setOnClickListener(v -> resultLauncher.launch(new
Intent(mGoogleSignInClient.getSignInIntent())));
    }

    private void handleFacebookToken(AccessToken accessToken) {
        AuthCredential credential =
FacebookAuthProvider.getCredential(accessToken.getToken());

auth.signInWithCredential(credential).addOnCompleteListener(LoginActivity.thi
s, task -> {
            if (task.isSuccessful()) {
                FirebaseUser user =
FirebaseAuth.getInstance().getCurrentUser();
                updateUI(user);
            } else {
                Toast.makeText(LoginActivity.this,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void updateUI(FirebaseUser user) {
        Intent intent = new Intent(LoginActivity.this,
DashboardActivity.class);
        startActivity(intent);
    }

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


    ActivityResultLauncher<Intent> resultLauncher =
registerForActivityResult(new
```

```
ActivityResultContracts.StartActivityForResult(), result -> {

        if (result.getResultCode() == Activity.RESULT_OK) {
            Intent intent = result.getData();
            Task<GoogleSignInAccount> task =
GoogleSignIn.getSignedInAccountFromIntent(intent);
            try {
                // Google Sign In was successful, authenticate with Firebase
                GoogleSignInAccount account =
task.getResult(ApiException.class);

                assert account != null;
                firebaseAuthWithGoogle(account.getIdToken());
                Toast.makeText(LoginActivity.this, R.string.success,
Toast.LENGTH_SHORT).show();
            } catch (ApiException e) {
                // Google Sign In failed, update UI appropriately
                Toast.makeText(LoginActivity.this,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
            }
        }

    });

    private void firebaseAuthWithGoogle(String idToken) {
        AuthCredential credential = GoogleAuthProvider.getCredential(idToken,
null);
        mAuth.signInWithCredential(credential)
                .addOnCompleteListener(this, task -> {
                    if (task.isSuccessful()) {
                        // Sign in success, update UI with the signed-in
user's information
                        startActivity(new Intent(LoginActivity.this,
DashboardActivity.class));
                        finish();
                        Toast.makeText(LoginActivity.this, R.string.success,
Toast.LENGTH_SHORT).show();

                    } else {
                        // If sign in fails, display a message to the user.
                        Toast.makeText(LoginActivity.this,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
    }

    private void authWithFacebook() {

LoginManager.getInstance().logInWithReadPermissions(LoginActivity.this,
Arrays.asList("public_profile"));
        LoginManager.getInstance().registerCallback(callbackManager, new
FacebookCallback<LoginResult>() {
            @Override
            public void onSuccess(LoginResult loginResult) {
                Toast.makeText(LoginActivity.this, R.string.success,
Toast.LENGTH_SHORT).show();
                handleFacebookToken(loginResult.getAccessToken());
            }

            @Override
            public void onCancel() {

            }
```

```
            @Override
            public void onError(FacebookException error) {

            }
        });
    }

    private void authWithMailAndPass() {
        String email, password;
        email = emailBox.getText().toString();
        password = passwordBox.getText().toString();

        auth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                if (auth.getCurrentUser().isEmailVerified()) {
                    startActivity(new Intent(LoginActivity.this,
DashboardActivity.class));
                    Toast.makeText(LoginActivity.this, R.string.success,
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(LoginActivity.this, R.string.verify_email,
Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(LoginActivity.this,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void rememberOrNot() {
        SharedPreferences preferences = getSharedPreferences("checkbox",
MODE_PRIVATE);
        String checkBox = preferences.getString("remember", "");
        if (checkBox.equals("true")) {
            startActivity(new Intent(LoginActivity.this,
DashboardActivity.class));
        }
    }

    private void rememberUser() {
        SharedPreferences preferences1 = getSharedPreferences("checkbox",
MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences1.edit();
        editor.putString("remember", "true");
        editor.apply();
    }

    private void forgotUser() {
        SharedPreferences preferences1 = getSharedPreferences("checkbox",
MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences1.edit();
        editor.putString("remember", "false");
        editor.apply();
    }

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

        FirebaseUser user = mAuth.getCurrentUser();

        SharedPreferences preferences = getSharedPreferences("checkbox",
```

```
MODE_PRIVATE);
        String checkBox = preferences.getString("remember", "");
        if (checkBox.equals("true") && user != null) {
            startActivity(new Intent(LoginActivity.this,
DashboardActivity.class));
        }
    }
}


package karpenko.test.parley;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;
import android.os.Handler;

import com.google.android.gms.ads.MobileAds;
import com.google.android.gms.ads.initialization.InitializationStatus;
import
com.google.android.gms.ads.initialization.OnInitializationCompleteListener;

import java.util.Locale;

public class MainActivity extends AppCompatActivity {

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

        MobileAds.initialize(this, new OnInitializationCompleteListener() {
            @Override
            public void onInitializationComplete(InitializationStatus
initializationStatus) {
                startActivity(new
Intent(MainActivity.this,LoginActivity.class));
                finish();
            }
        });

       /* new Handler().postDelayed(() -> {
            startActivity(new Intent(MainActivity.this,LoginActivity.class));
            finish();
        }, 2500);*/
    }

    private void setLocate(String language) {

        Locale locale = new Locale(language);
        Locale.setDefault(locale);
        Configuration configuration = new Configuration();
        configuration.locale = locale;

getBaseContext().getResources().updateConfiguration(configuration,getBaseCont
ext().getResources().getDisplayMetrics());

        SharedPreferences.Editor editor = getSharedPreferences("Settings",
MODE_PRIVATE).edit();
        editor.putString("My_lang", language);
        editor.apply();
```

```
    }

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

        SharedPreferences editor = getSharedPreferences("Settings",
MODE_PRIVATE);
        String lang = editor.getString("My_lang","");
        setLocate(lang);

    }

}


package karpenko.test.parley;


import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdSize;
import com.google.android.gms.ads.AdView;
import com.google.android.material.bottomnavigation.BottomNavigationItemView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.Locale;

import spencerstudios.com.bungeelib.Bungee;

public class SettingsActivity extends AppCompatActivity {


    private TextView passChange, deleteAcc, langChange;
    private BottomNavigationItemView home, history, logOut;
    private FirebaseUser user;
    private AdView mAdView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        loadLocate();
        AdView adView = new AdView(this);
        adView.setAdSize(AdSize.BANNER);
        adView.setAdUnitId("ca-app-pub-3940256099942544/6300978111");
        mAdView = findViewById(R.id.adView);
        AdRequest adRequest = new AdRequest.Builder().build();
        mAdView.loadAd(adRequest);

        passChange = findViewById(R.id.changePassBasic);
        deleteAcc = findViewById(R.id.deleteAccBtn);
```

```
        langChange = findViewById(R.id.changeLang);
        home = findViewById(R.id.homeBtn);
        history = findViewById(R.id.historyBtn);
        logOut = findViewById(R.id.logOutBtn);
        user = FirebaseAuth.getInstance().getCurrentUser();

        home.setOnClickListener(v -> {
            startActivity(new Intent(SettingsActivity.this,
DashboardActivity.class));
            Bungee.slideRight(SettingsActivity.this);
        });

        logOut.setOnClickListener(v -> logOutUser());

        history.setOnClickListener(v -> {
            startActivity(new Intent(SettingsActivity.this,
UserHistory.class));
            Bungee.slideRight(SettingsActivity.this);
        });

        passChange.setOnClickListener(v -> {
            startActivity(new Intent(SettingsActivity.this,
ChangePassActivity.class));
            Bungee.slideLeft(SettingsActivity.this);
        });

        deleteAcc.setOnClickListener(v -> {
            AlertDialog.Builder dialog = new
AlertDialog.Builder(SettingsActivity.this);
            dialog.setTitle(getString(R.string.are_u_sure));
            dialog.setMessage(getString(R.string.remove_acc_from_system));
            dialog.setPositiveButton(R.string.yes, (dialog1, which) ->
user.delete().addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    startActivity(new Intent(SettingsActivity.this,
LoginActivity.class));
                    Bungee.slideRight(SettingsActivity.this);
                    finishAffinity();
                } else {
                    Toast.makeText(SettingsActivity.this,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
                }
            }));
            dialog.setNegativeButton(R.string.no, (dialog12, which) ->
dialog12.dismiss());
            AlertDialog alertDialog = dialog.create();
            alertDialog.show();
        });

        langChange.setOnClickListener(v -> {
            showChangeLanguageDialog();
        });
    }

    private void showChangeLanguageDialog() {
        final String[] listItems = {"English", "Українська"};
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle(R.string.chose_lang);
        builder.setSingleChoiceItems(listItems, -1, (dialog, which) -> {
            if (which == 0) {
                setLocate("en");
                recreate();
            } else if (which == 1) {
                setLocate("uk");
```

```
                recreate();
            }
            dialog.dismiss();

        });
        AlertDialog dialog = builder.create();
        dialog.show();
    }

    private void setLocate(String language) {

        Locale locale = new Locale(language);
        Locale.setDefault(locale);
        Configuration configuration = new Configuration();
        configuration.locale = locale;

getBaseContext().getResources().updateConfiguration(configuration,getBaseCont
ext().getResources().getDisplayMetrics());

        SharedPreferences.Editor editor = getSharedPreferences("Settings",
MODE_PRIVATE).edit();
        editor.putString("My_lang", language);
        editor.apply();

    }

    private void loadLocate() {
        SharedPreferences preferences = getSharedPreferences("Settings",
Activity.MODE_PRIVATE);
        String language = preferences.getString("My_lang", "");
        setLocate(language);

    }


    private void logOutUser() {
        SharedPreferences preferences = getSharedPreferences("checkbox",
MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("remember", "false");
        editor.apply();
        startActivity(new Intent(SettingsActivity.this,
LoginActivity.class));
        Bungee.slideRight(SettingsActivity.this);
        FirebaseAuth.getInstance().signOut();
        finishAffinity();
    }

}


package karpenko.test.parley;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdSize;
import com.google.android.gms.ads.AdView;
```

```
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;

import spencerstudios.com.bungeelib.Bungee;


public class SignUpActivity extends AppCompatActivity {

    private EditText userName, userEmail, userPassword;
    private Button createAccountBtn, logInBtn;

    private FirebaseAuth auth;
    private FirebaseFirestore firebaseFirestore;
    private AdView mAdView;

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

        AdView adView = new AdView(this);
        adView.setAdSize(AdSize.BANNER);
        adView.setAdUnitId("ca-app-pub-3940256099942544/6300978111");
        mAdView = findViewById(R.id.adView);
        AdRequest adRequest = new AdRequest.Builder().build();
        mAdView.loadAd(adRequest);

        userName = findViewById(R.id.userNameBox);
        userEmail = findViewById(R.id.userEmailBox);
        userPassword = findViewById(R.id.passwordBox);
        createAccountBtn = findViewById(R.id.createAccBtn);
        logInBtn = findViewById(R.id.loginBtn);
        auth = FirebaseAuth.getInstance();
        firebaseFirestore = FirebaseFirestore.getInstance();

        logInBtn.setOnClickListener(v -> {
            startActivity(new Intent(SignUpActivity.this,
LoginActivity.class));
            Bungee.slideRight(SignUpActivity.this);
        });

        createAccountBtn.setOnClickListener(v ->
createAccountWithEmailAndPassword());

    }


    private void createAccountWithEmailAndPassword() {
        String email, name, password;
        name = userName.getText().toString();
        email = userEmail.getText().toString();
        password = userPassword.getText().toString();

        User user = new User();

        user.setName(name);
        user.setEmail(email);
        user.setPassword(password);

        auth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {

auth.getCurrentUser().sendEmailVerification().addOnCompleteListener(task1 ->
```

```
{
                    if (task1.isSuccessful()) {
                        Toast.makeText(SignUpActivity.this,
R.string.verify_email, Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(SignUpActivity.this,
task1.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
                firebaseFirestore.collection("users")
                        .document()
                        .set(user).addOnSuccessListener(unused -> {
                    startActivity(new Intent(SignUpActivity.this,
LoginActivity.class));
                    Bungee.slideRight(SignUpActivity.this);
                });
                Toast.makeText(SignUpActivity.this, R.string.success,
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(SignUpActivity.this,
task.getException().getLocalizedMessage(), Toast.LENGTH_SHORT).show();
            }

        });
    }


    @Override
    public void onBackPressed() {
        startActivity(new Intent(SignUpActivity.this, LoginActivity.class));
        Bungee.slideRight(SignUpActivity.this);
        finish();
    }
}


package karpenko.test.parley;

public class User {
    private String name, email,password;

    public User(){

    }

    public String getName() {
        return name;
    }

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

    public String getEmail() {
        return email;
    }

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

    public String getPassword() {
        return password;
    }
```

```
    public void setPassword(String password) {
        this.password = password;
    }
}


package karpenko.test.parley;

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

import android.app.ProgressDialog;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;

import com.google.android.material.bottomnavigation.BottomNavigationItemView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.ArrayList;

import spencerstudios.com.bungeelib.Bungee;
import timber.log.Timber;

public class UserHistory extends AppCompatActivity {

    private RecyclerView recyclerView;
    private ArrayList<History> histories;
    private HistoryAdapter historyAdapter;
    private FirebaseFirestore db = FirebaseFirestore.getInstance();
    private FirebaseAuth auth;
    private FirebaseUser user;
    private ProgressDialog progressDialog;
    private BottomNavigationItemView home, settings, logOut;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_history);
        home = findViewById(R.id.homeBtn);
        settings = findViewById(R.id.settingsBtn);
        logOut = findViewById(R.id.logOutBtn);
        auth = FirebaseAuth.getInstance();
        user = FirebaseAuth.getInstance().getCurrentUser();
        db = FirebaseFirestore.getInstance();
        recyclerView = findViewById(R.id.recycler_user_history);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        histories = new ArrayList<History>();
        historyAdapter = new HistoryAdapter(UserHistory.this, histories);
        recyclerView.setAdapter(historyAdapter);


        settings.setOnClickListener(v -> {
            startActivity(new Intent(UserHistory.this,
SettingsActivity.class));
            Bungee.slideLeft(UserHistory.this);
```

```
        });

        home.setOnClickListener(v -> {
            startActivity(new Intent(UserHistory.this,
DashboardActivity.class));
            Bungee.slideRight(UserHistory.this);
        });

        logOut.setOnClickListener(v -> logOutUser());

        EventChangeListener();

    }

    private void EventChangeListener() {

        db.collection("userVisitedRoom").whereEqualTo("userID",
user.getUid())
                .addSnapshotListener((value, error) -> {


                    for (DocumentChange documentChange :
value.getDocumentChanges()) {
                        if (documentChange.getType() ==
DocumentChange.Type.ADDED) {

histories.add(documentChange.getDocument().toObject(History.class));
                        }
                        historyAdapter.notifyDataSetChanged();
                    }
                });
    }

    private void logOutUser() {
        SharedPreferences preferences = getSharedPreferences("checkbox",
MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("remember", "false");
        editor.apply();
        startActivity(new Intent(UserHistory.this, LoginActivity.class));
        Bungee.slideRight(UserHistory.this);
        FirebaseAuth.getInstance().signOut();
        finishAffinity();
    }

}
```