

---

# [TravIns]

---

Laporan Project Matakuliah  
Pengembangan Aplikasi Perangkat Bergerak  
Program Studi Sistem Informasi

diampu oleh: Aryo Pinandito, Ph.D.

## Kelompok

Marsa Fatin Halimah	- 215150407111007
Fatikah Izza Maulidina S	- 215150401111020
Muhammad Althaaf Fadhiilah	- 215150400111053
Ananda Marsekalyana Rahmihadi	- 215150407111016
Briegitta Charmita	- 215150400111025

# Daftar Isi

Daftar Isi	i
Deskripsi Aplikasi	3
Use Case 1: Create, Read & Delete Favorites List	9
Marsa Fatin Halimah	9
Deskripsi Fungsional	9
Rancangan Screenflow	11
Class Diagram	12
Android Manifest	14
Implementasi UI	15
Layout 1: [fragment_favorite.xml]	16
Layout 2: [place_item_layout.xml]	18
Implementasi Kode Program Aplikasi	24
1. Implementasi Adapter	24
2. Implementasi ListView/AdapterView/RecyclerView	38
3. Implementasi Fragment	50
4. Implementasi Komunikasi Data	55
Use Case 2: Create, Update, & Read User Profile	57
Fatikah Izza Maulidina S.	57
Deskripsi Fungsional	57
Rancangan Screenflow	59
Class Diagram	60
Android Manifest	62
Implementasi UI	64
Layout 1: [fragment_profile]	64
Layout 2: [activity_edit_profile]	71
Implementasi Kode Program Aplikasi	1
Implementasi Komunikasi Data	19
Use Case 3: Register & Search Places	21
Muhammad Althaaf Fadhiilah	21
Deskripsi Fungsional	21
Rancangan Screenflow Register	23
Class Diagram Register	25
Android Manifest	30

Implementasi UI	31
Layout 1: [activity_login]	31
Layout 2: [activity_register]	37
Layout 3: [fragment_search]	44
Implementasi Kode Program Aplikasi	49
Implementasi ListView/AdapterView/RecyclerView	70
Implementasi Fragment	86
Implementasi Komunikasi Data	90
<b>Use Case 4: Read Destination Details &amp; Recommendation Destination</b>	<b>92</b>
Ananda Marsekalyana Rahmihadi	92
Deskripsi Fungsional	92
Rancangan Screenflow	93
Class Diagram	95
Android Manifest	97
Implementasi UI	98
Layout 1: [activity_list_trip]	99
Layout 2: [activity_place_detail]	101
Implementasi Kode Program Aplikasi	106
4. Implementasi Adapter	106
5. Implementasi ListView/AdapterView/RecyclerView	115
6. Implementasi Fragment	121
4. Implementasi Thread	4
5. Implementasi Komunikasi Data	4
<b>Use Case 5: Create &amp; Read Destination Reviews</b>	<b>6</b>
Briegitta Charmita	6
Deskripsi Fungsional	6
Rancangan Screenflow	8
Class Diagram	9
Android Manifest	11
Implementasi UI	13
Layout 1: [activity_place_detail.xml]	14
Layout 2: [fragment_review.xml]	17
Implementasi Kode Program Aplikasi	20
Implementasi ListView/AdapterView/RecyclerView	27
Implementasi Fragment	31
Implementasi Thread	35
Implementasi Komunikasi Data	39

## Deskripsi Aplikasi

Indonesia memiliki beragam kekayaan alam dan budaya yang melimpah dan banyak dimanfaatkan untuk berbagai aspek, termasuk dalam aspek wisata. Sebagai negara kepulauan yang memiliki banyak keanekaragaman zona wisata, di Indonesia hampir setiap kabupaten atau kota memiliki tempat-tempat wisata yang mulai dikembangkan dengan mengelola aset alam maupun budayanya. Pada kondisi pasca pandemic Covid-19 ini, masyarakat Indonesia mulai beranjak kembali melakukan kegiatannya dengan normal. Sektor pariwisata yang sebelumnya sempat lumpuh karena terdapat banyak tempat wisata yang ditutup pada masa pandemi, kini kembali dibuka seiring dengan masyarakat yang kembali melakukan aktivitas dan pekerjaannya seperti sedia kala. Berwisata menjadi salah satu kegiatan yang banyak dipilih oleh masyarakat untuk dilakukan setelah selama kurang lebih dua tahun akses pariwisata mengalami pembatasan.

Pariwisata merupakan salah satu faktor penting dalam pertumbuhan ekonomi Indonesia. Sebagai contoh pada tahun 2019 sektor pariwisata memberikan (kontribusi/sumbangsih) sebesar 4.7% terhadap total pendapatan gross domestic product (GDP) Indonesia atau secara numerik sebesar 52.593 triliun USD dari total GDP Indonesia. (Azzahra, 2022). Pada tahun 2020, kontribusi dari sektor pariwisata mengalami penurunan sebesar 4.05% dari jumlah total GDP. Maka dari itu, diperlukan suatu upaya untuk memulihkan sektor pariwisata agar dapat mengangkat kontribusinya terhadap GDP dan perekonomian Indonesia.

Namun di sisi lain, berbagai macam masalah masih menghantui sektor pariwisata Indonesia. Salah satu masalah yang dihadapi pariwisata adalah komunikasi dan publikasi yang masih dikatakan kurang (Nugroho, 2020). Pariwisata tidak mendapat kesempatan optimum untuk berkembang tanpa publikasi yang menyatakan keberadaan pariwisata tersebut. Kurangnya publikasi ini membuat banyak tempat - tempat wisata yang potensial kurang diketahui oleh para turis. Padahal di era sekarang ini, penerapan teknologi banyak dilakukan untuk mendukung komunikasi, publikasi, dan penyebaran informasi secara lebih luas.

Meskipun beberapa daerah wisata di Indonesia dikenal hingga mancanegara, banyak diantaranya juga masih kurang diketahui dan dijamah wisatawan karena jarang terekspos serta kurangnya branding untuk memasarkan tempat wisata tersebut. Oleh karena itu, “TravIns” hadir sebagai solusi untuk meningkatkan komunikasi dan publikasi yang ada di Indonesia untuk mendukung pengembangan potensi tempat wisata tersebut serta membantu wisatawan untuk bisa mendapatkan akses informasi secara praktis, mudah, dan cepat.

Aplikasi “TravIns” selain memiliki fitur dasar sebuah aplikasi seperti Registrasi, Login, Logout, dan User Profile, “TravIns” ini memiliki beberapa fitur utama, yaitu fitur menampilkan daftar informasi tempat destinasi wisata, menandai tempat destinasi favorit, dan menambahkan review untuk destinasi wisata. Saat user berada

dalam homepage aplikasi, user dapat melihat berbagai daftar dan rekomendasi tempat destinasi wisata. Saat user memilih salah satu destinasi, maka aplikasi akan menampilkan detail lengkap termasuk deskripsi, peta lokasi, serta review dan rating untuk destinasi tersebut.

User dapat membuat ulasan untuk destinasi yang pernah dikunjungi sebelumnya dengan scrolling ke halaman sampai paling bawah. Sesudah mengirimkan review, user dapat scroll sedikit ke atas untuk melihat review yang baru saja dikirim. Atau jika user ingin tertarik untuk mengunjungi destinasi tersebut, user dapat memasukkan destinasi ke dalam favorite lists.

Fitur selanjutnya adalah Search, sebuah fitur yang penting dan dibutuhkan dalam aplikasi ini dengan beberapa alasan utama. Mulai dari memungkinkan user dengan cepat menemukan informasi yang mereka butuhkan, meningkatkan user experience, dan efisiensi penggunaan aplikasi. Tanpa fitur search, user mungkin kesulitan navigasi melalui konten atau data yang besar, mengurangi produktivitas dan kepuasan user.

“TravIns” juga memiliki fitur kelola user profil yang memungkinkan user untuk memperbarui dan mengelola informasi pribadi mereka dalam profil user. Dengan adanya fitur ini, user dapat memperbarui informasi seperti nama lengkap, username, gender, tanggal lahir, negara, dan alamat. Dengan adanya fitur kelola profil, user memiliki kebebasan untuk mempersonalisasikan aplikasi “TravIns” sesuai dengan keinginan dan kebutuhan user.

Aplikasi “TravIns” dirancang untuk memberikan pengguna pengalaman pencarian dan penemuan destinasi wisata yang memikat dan sesuai dengan preferensi mereka. Berikut adalah cara kerja umum aplikasi ini:

### 1. Home Screen:

- Pengguna memulai dengan membuka aplikasi dan dihadapkan dengan Home Screen yang menampilkan opsi-opsi navigasi utama.
  - Terdapat fitur "Inspiration for your trip" yang menampilkan beberapa destinasi populer atau menarik untuk memberikan inspirasi awal kepada pengguna.

### 2. Search Features:

- Aplikasi TravIns juga menyediakan fitur pencarian yang memungkinkan pengguna mencari destinasi berdasarkan kriteria tertentu, seperti nama, lokasi, atau penilaian pengguna.

### 3. Search Results Destination:

- Pengguna dapat menjelajahi lebih banyak destinasi dengan mengklik button “Search” pada navbar yang membuka tampilan untuk mencari

daftar destinasi wisata terkait.

- Tampilan daftar ini dapat mencakup informasi singkat, gambar, dan penilaian dari pengguna lain untuk membantu pengguna membuat keputusan.

#### 4. Add to Favorites:

- Setelah melihat destinasi yang menarik, pengguna dapat menambahkannya ke dalam daftar favorit mereka dengan menekan "Heart Icon" yang terletak di sebelah kanan atas gambar destinasi.
- Gambar "Heart Icon" yang berubah menjadi terisi penuh memberikan umpan balik visual bahwa destinasi tersebut telah ditambahkan ke daftar favorit.

#### 5. Favorites List:

- Pengguna dapat melihat daftar destinasi favorit mereka dalam tampilan "Favorites List."
- Apabila pengguna mengklik salah satu destinasi, maka akan ternavigasi kepada screen "Place Details" mengenai informasi lebih rinci terkait destinasi wisata yang ada, termasuk deskripsi, peta, dan opsi untuk mengelola atau mengedit daftar favorit.

#### 6. Delete from Favorites:

- Untuk menghapus destinasi dari daftar favorit, pengguna hanya perlu menekan kembali "Heart Icon" pada gambar destinasi di tampilan "Favorites List."
- Visual "Heart Icon" yang kembali kosong menandakan bahwa destinasi tersebut telah dihapus dari daftar favorit.

#### 7. Place Details

- Setelah pengguna memilih destinasi tertentu, mereka dapat mengakses "Place Details" untuk mendapatkan informasi lebih lanjut.
- Fitur ini menyajikan gallery foto terkait destinasi tersebut dan juga deskripsi menyeluruh tentang destinasi tersebut.
- "Place Details" juga menyediakan peta interaktif yang menunjukkan lokasi destinasi wisata. Pengguna dapat menjelajahi peta untuk mendapatkan pemahaman yang lebih baik tentang letak geografis dan lingkungan sekitar destinasi tersebut.
- Sebagai tambahan, fitur ini memberikan rekomendasi destinasi serupa atau berkaitan berdasarkan preferensi pengguna dan sejarah penelusuran mereka. Ini membantu pengguna menemukan tempat-tempat lain yang mungkin sesuai dengan minat mereka.

#### 8. Profile Menu & Edit Profile

- Pengguna dapat mengakses profil mereka dengan menavigasi ke "Profile

Menu." Di sini, mereka dapat melihat informasi dasar seperti nama pengguna, lokasi, foto profil, edit profile, dsb.

- Terdapat opsi "Edit Profile" yang memungkinkan pengguna mengubah atau melengkapi data diri mereka. Pengguna dapat memasukkan informasi seperti nama lengkap, username, gender, tanggal lahir, negara, dan juga alamat.

Teknologi dan Fitur Android yang digunakan pada aplikasi TravIns adalah :

1. Teknologi Android Studio:

- Bahasa Pemrograman: Java, sebagai bahasa pemrograman utama untuk pengembangan aplikasi Android.
- Android SDK (Software Development Kit): Berisi alat-alat yang diperlukan untuk mengembangkan aplikasi Android, seperti Android Emulator, ADB (Android Debug Bridge), dsb.
- Gradle Build System: Untuk mengelola dependensi dan membangun proyek.

2. Fitur-fitur Aplikasi "TravIns":

a. Registrasi, Login, dan Logout:

- Pengguna dapat membuat akun baru (Registrasi).
- Login menggunakan kredensial yang telah terdaftar.
- Logout untuk mengakhiri sesi pengguna.

b. Create, Update, dan Read User Profile:

- Pengguna dapat membuat profil pribadi.
- Mengedit profil (Update) seperti gambar profil, nama, atau informasi lainnya.
- Membaca profil pengguna.

c. Search:

- Fitur pencarian memungkinkan pengguna mencari destinasi wisata berdasarkan kriteria tertentu, seperti nama, lokasi, atau kategori.

d. Read Destination Details dan Recommendation Details:

- Melihat detail destinasi wisata termasuk deskripsi, gambar, fasilitas, dan informasi penting lainnya.
- Menerima rekomendasi destinasi berdasarkan preferensi pengguna.

e. Create, Read, dan Delete Favorite Lists:

- Menambahkan destinasi ke daftar favorit (Create).
- Melihat daftar destinasi favorit (Read).
- Menghapus destinasi dari daftar favorit (Delete).

f. Create dan Read Destination Reviews:

- Pengguna dapat menulis review tentang destinasi wisata (Create).

- Membaca ulasan-ulasan pengguna tentang destinasi (Read).

**3. UI/UX:**

Material Design: Menerapkan prinsip-prinsip desain Material Design untuk memberikan antarmuka yang bersih, responsif, dan mudah digunakan.

**4. Database dan Backend:**

SQLite atau Firebase: Sebagai penyimpanan lokal atau cloud untuk data pengguna, profil, ulasan, dan informasi lainnya.

**5. Networking:**

Retrofit atau Volley: Untuk mengelola permintaan jaringan ke server, seperti mendapatkan data destinasi atau menyimpan ulasan pengguna.

**6. Firebase Authentication:**

Mengelola sistem otentikasi pengguna untuk fitur Registrasi, Login, dan Logout.

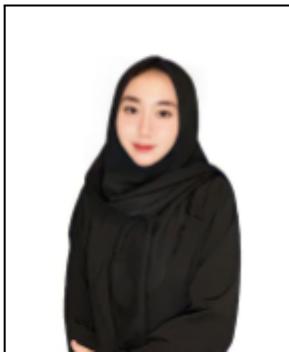
**7. Push Notifications:**

Menggunakan Firebase Cloud Messaging (FCM) atau layanan push notification lainnya untuk memberitahu pengguna tentang pembaruan atau informasi penting.

# Use Case 1: Create, Read & Delete Favorites List

Marsa Fatin Halimah

NIM 215150407111007



## Deskripsi Fungsional

Fungsi "Favorites List" bertujuan memberikan pengguna kemampuan untuk menyusun dan mengelola daftar destinasi wisata favorit mereka dengan mudah. Ini memungkinkan pengguna untuk menyimpan, meninjau, dan menghapus destinasi yang paling mereka sukai, menciptakan pengalaman perjalanan yang dipersonalisasi. Berikut adalah cara kerja:

### 1. Create (Tambahkan ke Daftar Favorit):

- Langkah 1: Pengguna menjelajahi destinasi wisata pada tampilan utama atau "See all" untuk melihat daftar lengkap.
- Langkah 2: Ketika menemukan destinasi yang diinginkan, pengguna menekan "Heart Icon" yang terletak di sebelah gambar destinasi.
- Langkah 3: Sistem mencatat destinasi tersebut sebagai favorit pengguna dan memperbarui visual "Heart Icon" menjadi terisi penuh.
- Langkah 4: Informasi tentang destinasi ini disimpan dalam "Favorites List" pengguna.

## 2. Read (Baca Daftar Favorit):

- Langkah 1: Pengguna membuka menu profil atau navigasi ke "Favorites List" pada tampilan utama.
- Langkah 2: Aplikasi menampilkan daftar destinasi yang telah ditandai sebagai favorit oleh pengguna.
- Langkah 3: Pengguna dapat melihat detail dari masing-masing destinasi, termasuk deskripsi, peta lokasi, dan rekomendasi destinasi serupa.

## 3. Delete (Hapus dari Daftar Favorit):

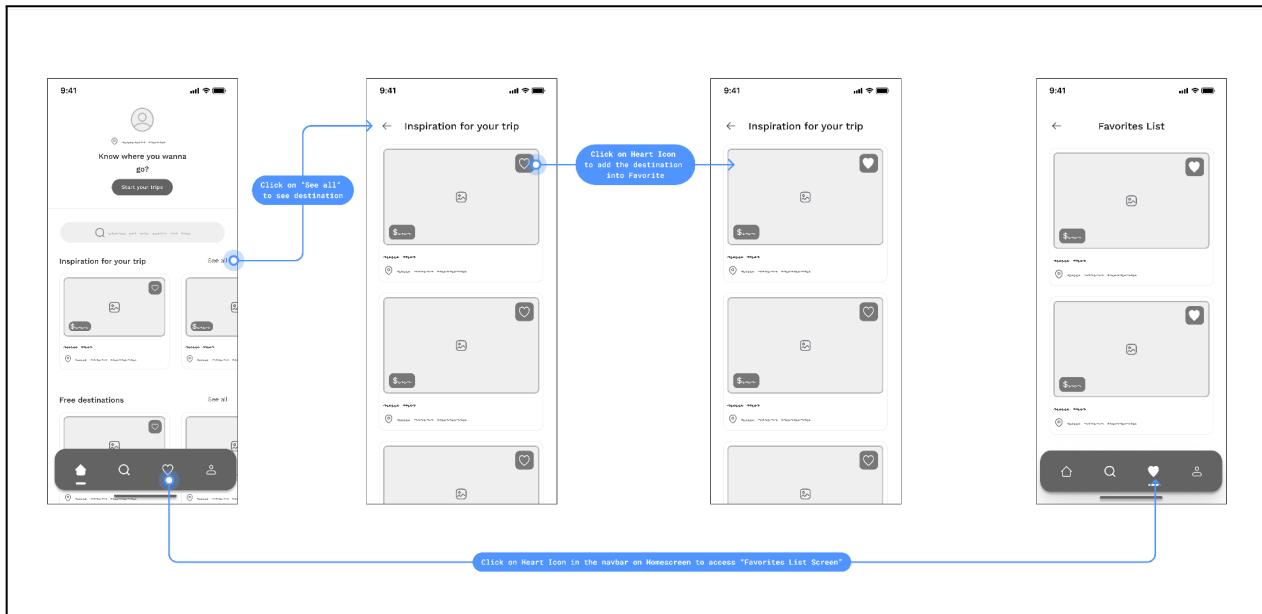
- Langkah 1: Pengguna membuka "Favorites List" atau menu profil.
- Langkah 2: Di daftar destinasi favorit, pengguna menemukan destinasi yang ingin dihapus.
- Langkah 3: Pengguna menekan "Heart Icon" kembali pada gambar destinasi yang ingin dihapus.
- Langkah 4: Sistem menghapus destinasi dari "Favorites List" dan memperbarui visual "Heart Icon" menjadi kosong.

### Penjelasan Tambahan:

- Setiap perubahan dalam "Favorites List" diupdate secara instan dan tersinkron dengan akun pengguna.
- Sistem dapat memberikan konfirmasi visual atau notifikasi ketika pengguna berhasil menambahkan atau menghapus destinasi dari daftar favorit.
- Proses Create, Read, dan Delete dirancang untuk menjadi intuitif sehingga pengguna dapat dengan mudah mengelola preferensi perjalanan mereka tanpa kesulitan.

Dengan demikian, fungsi "Favorites List" pada aplikasi "TravIns" membantu menciptakan pengalaman perjalanan yang disesuaikan dengan keinginan pengguna, memungkinkan mereka mengelola dan menikmati daftar destinasi favorit mereka dengan efisien.

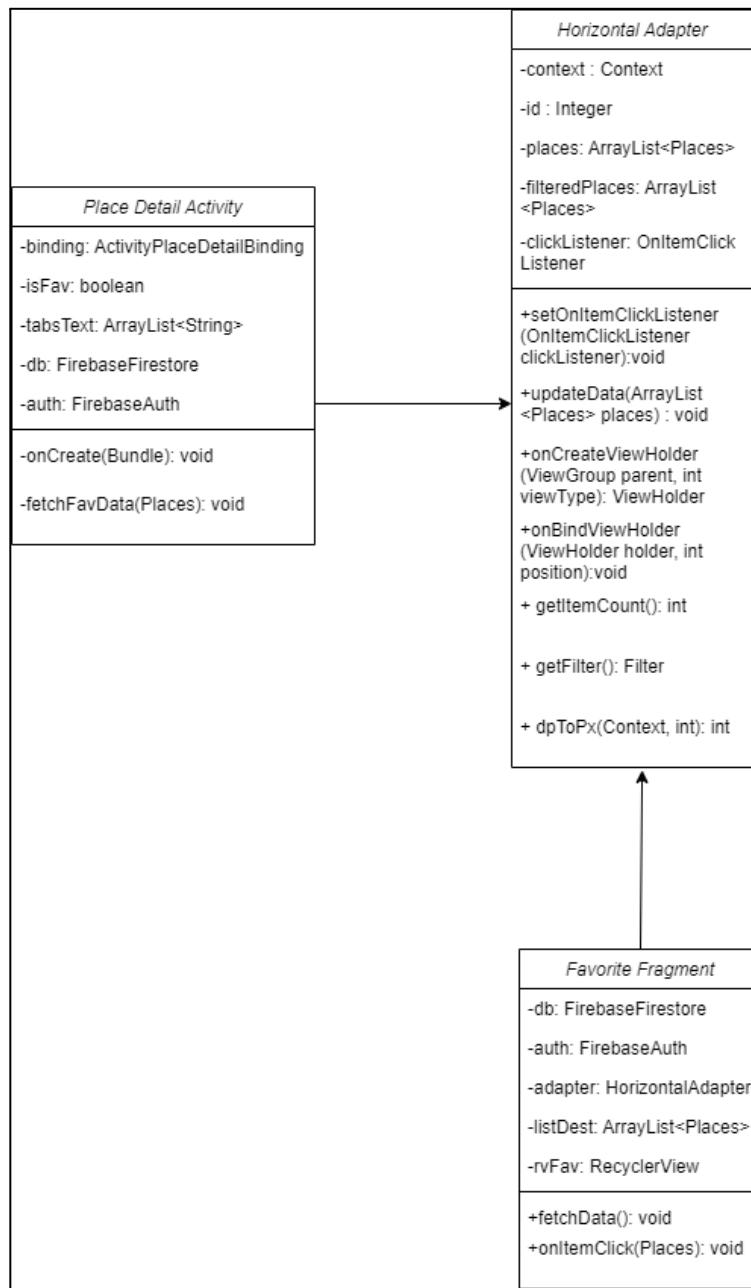
# Rancangan Screenflow



Fitur Create, Read, dan Delete (CRD) pada bagian “Favorites List” ini memberikan pengalaman yang intuitif dan personal bagi pengguna. Pertama-tama, ketika pengguna mengakses Home Screen, pengguna disajikan dengan opsi “Inspiration for your trip.” Untuk mendapatkan gambaran menyeluruh, pengguna dapat mengklik tulisan “See all,” yang mengarahkannya ke tampilan screen berupa daftar destinasi wisata. Dalam tampilan “Inspiration for your trip” yang menampilkan daftar destinasi, pengguna memiliki kemampuan untuk membuat list destinasi favorit mereka. Dengan menekan “Heart Icon” yang terletak di kanan atas gambar destinasi, pengguna dapat dengan mudah menambahkan destinasi tersebut ke dalam daftar favorit mereka. Setelah berhasil menambahkan, “Heart Icon” akan berubah menjadi bentuk terisi penuh, memberikan umpan balik visual yang jelas kepada pengguna.

Pengguna dapat melihat semua destinasi yang telah mereka tambahkan ke daftar favorit dalam tampilan screen “Favorites List.” Ini memungkinkan mereka dengan cepat mengakses dan merencanakan perjalanan berdasarkan pilihan pribadi mereka. Namun, kebebasan untuk membuat juga diimbangi dengan kemudahan untuk menghapus. Jika pengguna berubah pikiran atau ingin menyusun ulang daftar favorit mereka, mereka dapat dengan mudah menghapus destinasi dari daftar tersebut. Caranya sangat sederhana, cukup dengan menekan kembali “Heart Icon” pada gambar destinasi yang ingin dihapus. Visual “Heart Icon” yang kembali kosong memberi petunjuk bahwa destinasi tersebut telah dihapus dari daftar favorit.

# Class Diagram



## 1. Place Detail Activity:

- Class ini berfungsi sebagai tampilan detail tempat.
- Variabel isFav digunakan untuk mengetahui apakah tempat tersebut telah menjadi favorit atau belum.
- tabs Text merupakan list text yang akan digunakan sebagai judul pada tab view.
- db adalah FirebaseFirestore yang digunakan untuk mengakses dan

menyimpan data tempat di dalam database.

- auth adalah FirebaseAuth yang digunakan untuk autentikasi pengguna.
- Fungsi onCreate(Bundle) berfungsi untuk menjalankan aplikasi ketika Activity dibuat.
- Fungsi fetchFavData(Places) berfungsi untuk mengambil data favorit dari database dan menampilkannya.

## 2. Horizontal Adapter:

- Class ini berfungsi sebagai adapter untuk menampilkan data tempat secara horizontal.
- Variabel context, id, places, dan filteredPlaces merupakan variabel yang digunakan untuk menampilkan data.
- Fungsi setOnItemClickListener(OnItemClickListener clickListener) berfungsi untuk menjalankan tindakan saat item diklik.
- Fungsi updateData(ArrayList<Places> places) berfungsi untuk memperbarui data yang ditampilkan.
- Fungsi onCreateViewHolder(ViewGroup parent, int viewType) dan onBindViewHolder(ViewHolder holder, int position) merupakan fungsi yang digunakan untuk menampilkan data tempat secara horizontal.
- Fungsi getItemCount() berfungsi untuk mengembalikan jumlah item yang ada.
- Fungsi getFilter() berfungsi untuk mengambil data filter.
- Fungsi dpToPx(Context, int) berfungsi untuk mengonversi satuan ukuran dp menjadi px.

## 3. Favorite Fragment:

- Class ini berfungsi sebagai tampilan fragment favorit.
- Variabel db, auth, adapter, dan listDest merupakan variabel yang digunakan untuk mengakses dan menyimpan data favorit di dalam database.
- Variabel rvFav adalah RecyclerView yang digunakan untuk menampilkan data favorit.
- Fungsi fetchData() berfungsi untuk mengambil data favorit dari database dan menampilkannya.
- Fungsi onItemClick(Places) berfungsi untuk menjalankan tindakan saat item favorit diklik.

# Android Manifest

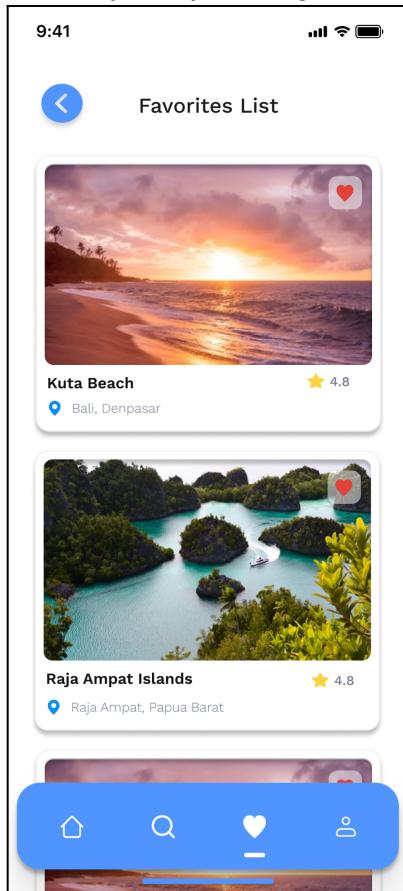
```
<activity  
    android:name=".ui.activities.PlaceDetailActivity"  
    android:exported="false"  
    android:windowSoftInputMode="adjustPan"/>
```

Ada satu aktivitas yang didefinisikan: PlaceDetailActivity.

- PlaceDetailActivity: Aktivitas ini mewakili layar detail tempat yang terdapat fitur fungsional “Favorites List” atau Visualisasi dari “Icon Heart”. Aktivitas ini tidak diizinkan untuk diekspor kepada aplikasi lain (android:exported="false"), dan tata letaknya disesuaikan (android:windowSoftInputMode="adjustPan")

# Implementasi UI

Tampilan pada Figma:



Dalam fungsional "Favorites List," pengguna dapat mengelola dan menyusun daftar destinasi wisata favorit mereka. Pengguna dapat menambahkan destinasi ke dalam daftar favorit dengan mudah melalui penekanan "Heart Icon" pada gambar destinasi yang menarik hati mereka. Setelah ditambahkan, pengguna dapat membaca detail lengkap dari setiap destinasi yang terdapat dalam daftar favorit, termasuk deskripsi, peta lokasi, dan rekomendasi destinasi serupa. Selain itu, fungsional ini memberikan fleksibilitas kepada pengguna untuk menghapus destinasi dari daftar favorit mereka dengan menekan kembali "Heart Icon." Dengan demikian, fungsional "Favorites List" memberikan pengalaman yang interaktif, memungkinkan pengguna untuk secara efektif mengelola dan mengeksplorasi destinasi yang mereka sukai.

## Layout 1: [fragment\_favorite.xml]

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.fragments.FavoriteFragment"
    android:paddingHorizontal="24dp"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingVertical="16dp">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_back"
            android:paddingStart="0dp"/>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Favorites List" />
    
```

```
        android:textColor="@color/black"
        android:textSize="22sp"
        android:textAlignment="center"
        android:paddingStart="-25dp"
        android:fontFamily="@font/worksans_regular"/>
    
```

```
</LinearLayout>

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvFav"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>

```

```
</LinearLayout>
```

Layout yang digunakan dalam fragment\_favorite.xml ini adalah `LinearLayout`, yang dipilih mungkin karena sifat linier dan berorientasi vertikalnya. `LinearLayout` cocok digunakan ketika elemen-elemen UI perlu diatur secara sejajar atau bertumpuk satu di bawah yang lain. Berikut adalah elemen-elemen yang terdapat pada layout:

1. LinearLayout Pertama:

- Atribut Layout dan Tampilan: Mewakili bagian atas layout dengan orientasi vertikal. Memiliki dua elemen anak: `ImageView` untuk tombol kembali dan `TextView` untuk judul "Favorites List"

2. ImageView:

- Atribut Layout dan Tampilan:\*\* Menunjukkan tombol kembali dengan gambar dari sumber daya drawable. Memiliki padding start yang disetel ke 0dp untuk penyesuaian tata letak.

3. TextView:

- Atribut Layout dan Tampilan:Menampilkan teks "Favorites List" dengan

ukuran font, warna, dan tata letak yang ditentukan. Menggunakan `paddingStart` yang disesuaikan untuk penyesuaian posisi teks.

#### 4. LinearLayout Kedua:

- Atribut Layout dan Tampilan: Merepresentasikan layout kedua dengan orientasi vertikal. Menyertakan `RecyclerView` untuk menampilkan daftar favorit.

#### 5. RecyclerView:

- Atribut Layout dan Tampilan: Menampilkan daftar favorit dalam bentuk daftar gulir. `@+id/rvFav` adalah ID yang dapat diakses untuk mengakses RecyclerView dalam kode Java/Kotlin.

Berikut adalah penggunaan Atribut:

- `android:layout\_width` dan `android:layout\_height`: Menentukan lebar dan tinggi elemen-elemen layout.
- `tools:context`: Memberikan konteks aktivitas atau fragmen yang digunakan untuk tujuan desain dan tata letak.
- `android:paddingHorizontal` dan `android:paddingVertical`: Menambahkan ruang padding horizontal dan vertical pada layout utama.
- `android:orientation`: Menentukan orientasi linear layout (vertical).
- Berbagai atribut lainnya, seperti ukuran teks, warna teks, dan sumber daya gambar, digunakan untuk menentukan penampilan dan tata letak elemen-elemen UI.

## Layout 2: [place\_item\_layout.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/cvPlace"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:cardCornerRadius="12dp"
        app:cardElevation="12dp"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        xmlns:app="http://schemas.android.com/apk/res-auto">

<androidx.constraintlayout.widget.ConstraintLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="8dp">

    <ImageView
        android:id="@+id/ivPlaceImg"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:background="@drawable/bg_img_rounded"
        android:clipToOutline="true"
        android:scaleType="centerCrop"
        android:src="@drawable/img_raja_ampat"
        app:layout_constraintDimensionRatio="49:30"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/btnFavPlace"
        android:layout_width="36dp"
        android:layout_height="36dp"
        android:background="@drawable/bg_rounded_8"
        android:padding="6dp"
        android:layout_margin="8dp"
        android:backgroundTint="#B3FFFFFF"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:id="@+id/fav"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:src="@drawable/ic_favourite" />

    </LinearLayout>

    <TextView
        android:id="@+id/tvPlaceName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:fontFamily="@font/worksans_semiBold"
        android:text="Raja Ampat Islands"
        android:textColor="@color/textPrimary"
        android:layout_marginEnd="8dp"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintEnd_toStartOf="@+id/tvRating"
        app:layout_constraintTop_toBottomOf="@+id/ivPlaceImg"
    />

    <TextView
        android:id="@+id/tvRating"
        android:layout_width="wrap_content"
        android:layout_height="15dp"
        android:drawableStart="@drawable/ic_star"
        android:drawablePadding="4dp"
        android:text="4.8"
        android:textSize="12sp"
        android:textColor="@color/textSecondary"
        android:fontFamily="@font/worksans_medium"
        android:layout_marginTop="8dp"
        app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintBottom_toBottomOf="@+id/tvPlaceName"
    app:layout_constraintTop_toBottomOf="@+id/ivPlaceImg"/>

    <TextView
        android:id="@+id/tvLocation"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp"
        android:text="Raja Ampat, Papua Barat"
        android:textSize="12sp"
        android:textColor="@color/textSecondary"
        android:fontFamily="@font/worksans_regular"
        android:drawableStart="@drawable/ic_location"
        android:drawablePadding="4dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@+id/tvPlaceName"/>

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```

Kode place\_item\_layout.xml menggunakan 'CardView' sebagai wadah utama untuk menampilkan setiap item destinasi wisata. Penggunaan 'CardView' memberikan efek

shadow dan corner radius yang estetis, meningkatkan tampilan visual aplikasi. Di dalam `CardView`, digunakan `ConstraintLayout` sebagai layout utama untuk menyusun elemen-elemen UI dengan tata letak yang fleksibel dan responsif.

#### 1. CardView:

- Tujuan: Memberikan efek visual menarik dengan bayangan dan sudut yang dibulatkan, menambahkan layer kedua sebagai kontainer untuk elemen-elemen UI.
- Atribut yang Digunakan:
  1. `app:cardCornerRadius`: Menentukan sudut corner dari CardView.
  2. `app:cardElevation`: Menentukan tinggi bayangan yang dihasilkan oleh CardView.
  3. `android:layout\_gravity`: Menengahkan CardView secara horizontal dan vertical.
  4. `android:layout\_margin`: Menyediakan margin sekitar CardView.
  - 5.

#### 2. ConstraintLayout:

- Tujuan: Memberikan tata letak yang fleksibel dengan menggunakan constraints, memudahkan penempatan elemen-elemen UI.
- Atribut yang Digunakan:
  - `android:layout\_width` dan `android:layout\_height`: Menentukan ukuran layout.
  - `android:padding`: Menambahkan padding pada seluruh layout.

#### 3. ImageView (ivPlaceImg):

- Tujuan: Menampilkan gambar destinasi wisata dengan proporsi yang diatur dan outline yang dibulatkan.
- Atribut yang Digunakan:
  1. `app:layout\_constraintDimensionRatio`: Menyimpan rasio aspek gambar.
  2. `app:layout\_constraintEnd\_toEndOf`,  
`app:layout\_constraintStart\_toStartOf`,  
`app:layout\_constraintTop\_toTopOf`: Menentukan constraints posisi gambar.

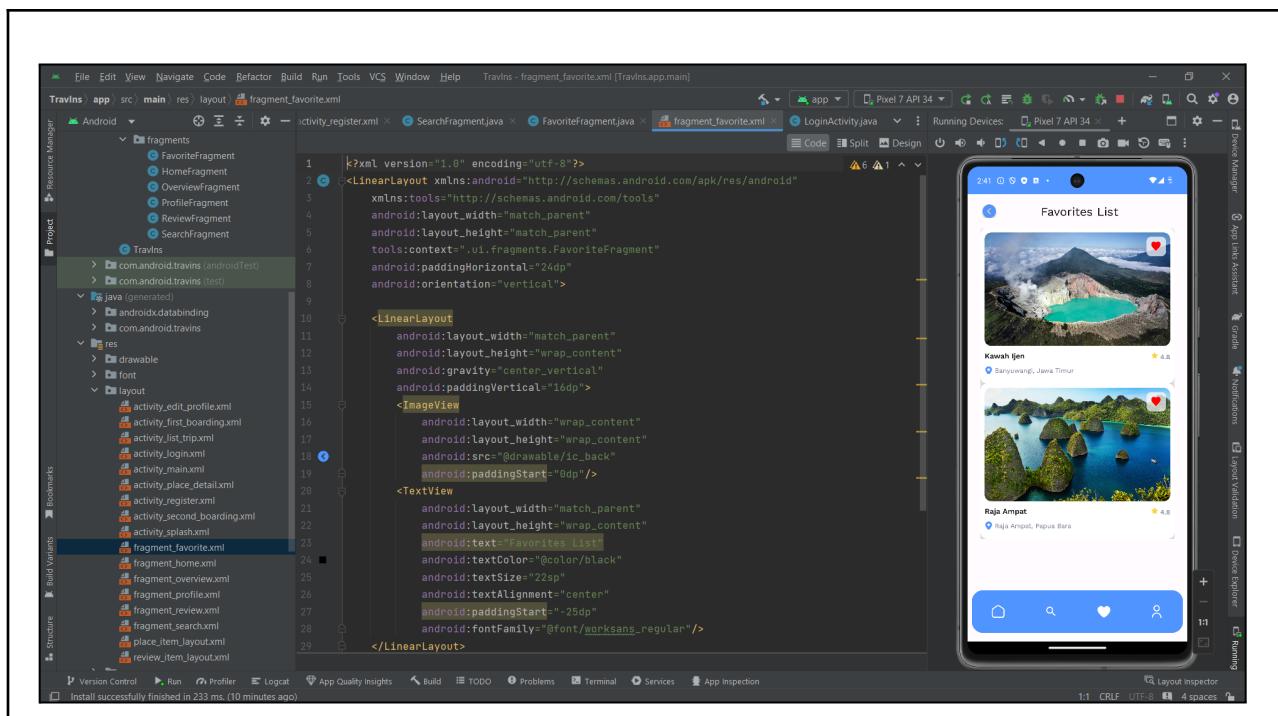
#### 4. LinearLayout (btnFavPlace):

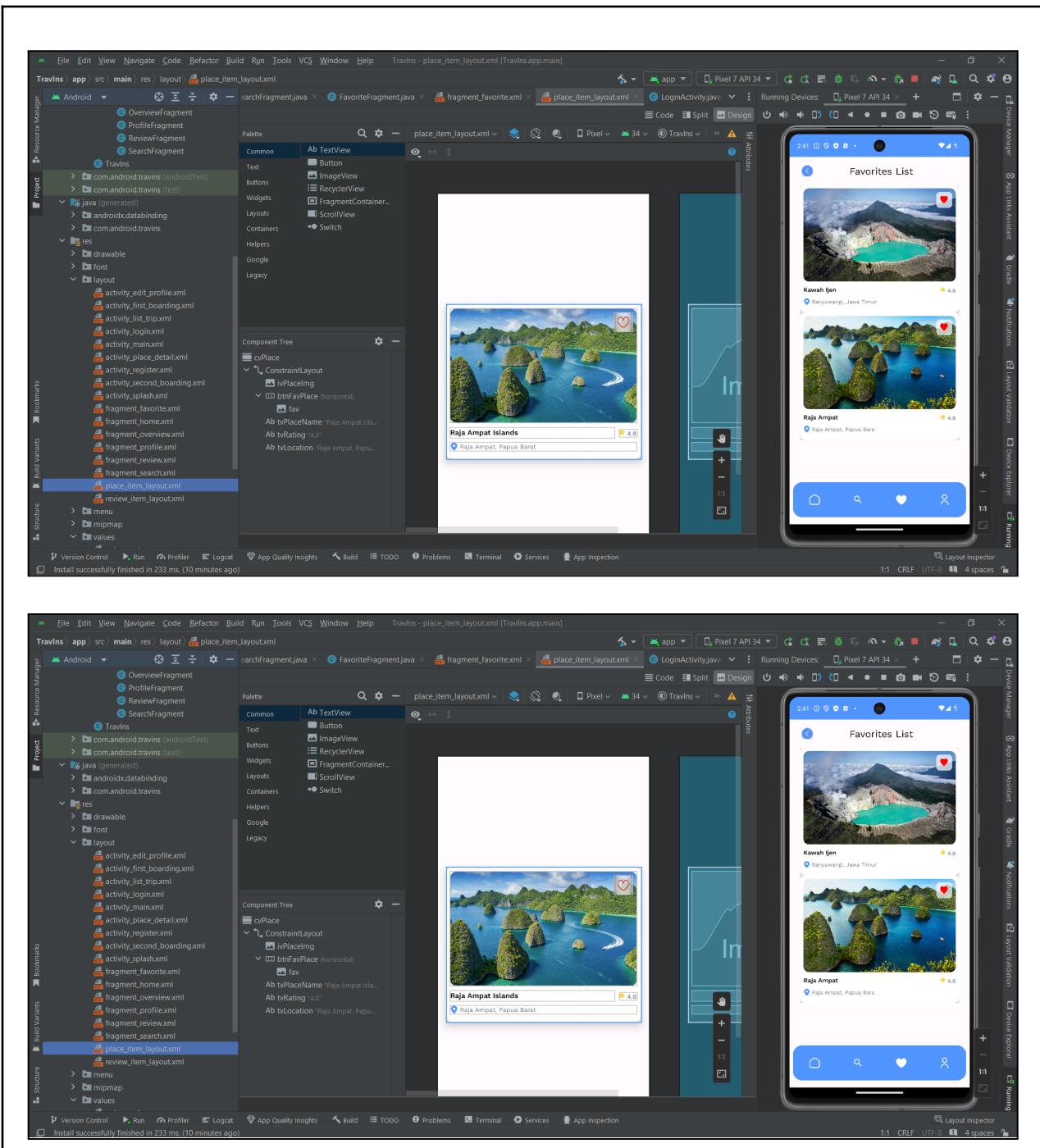
- Tujuan: Menyediakan tombol favorit untuk setiap destinasi.
- Atribut yang Digunakan:
  - `android:layout\_width` dan `android:layout\_height`: Menentukan ukuran tombol.
  - `android:background`, `android:backgroundTint`: Menentukan tampilan latar dan warna latar tombol.
  - `app:layout\_constraintEnd\_toEndOf` dan  
`app:layout\_constraintTop\_toTopOf`: Menentukan constraints posisi tombol.

## 5. TextView (tvPlaceName, tvRating, tvLocation):

- Tujuan: Menampilkan informasi seperti nama destinasi, rating, dan lokasi.
- Atribut yang Digunakan:
  1. Berbagai atribut seperti `android:layout\_width`, `android:layout\_height`, `app:layout\_constraintStart\_toStartOf`, `app:layout\_constraintEnd\_toEndOf`, `app:layout\_constraintTop\_toBottomOf`, dll., digunakan untuk menentukan tata letak dan penampilan teks.

Tampilan Screenshot layout:





# Implementasi Kode Program Aplikasi

## 1. Implementasi Adapter

Kode Program adapter : HorizontalAdapter

```
package com.android.travins.ui.adapters;

import static
com.google.android.material.internal.ViewUtils.dpToPx;

import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Filter;
import android.widget.Filterable;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.bumptech.glide.Glide;

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

public class HorizontalAdapter extends RecyclerView.Adapter<HorizontalAdapter.ViewHolder> implements Filterable {  
    private Context context;  
    private Integer id;  
    private ArrayList<Places> places = new ArrayList<>();  
    private ArrayList<Places> filteredPlaces = new ArrayList<>();  
}
```

```
private OnItemClickListener clickListener;

    public void setOnItemClickListener(OnItemClickListener clickListener) {
        this.clickListener = clickListener;
    }

    // Constructor to receive the context
    public HorizontalAdapter(Context context, Integer id) {
        this.context = context;
        this.id = id;
    }

    public void updateData(ArrayList<Places> places) {
        this.places = places;
        this.filteredPlaces = new ArrayList<>(places);
        notifyDataSetChanged();
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.place_item_layout, parent,
false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        // Bind data or perform operations for each item
        Places p = filteredPlaces.get(position);

        if (id == 1){
            holder.btnFavPlace.setVisibility(View.GONE);
        }else{
            holder.btnFavPlace.setVisibility(View.VISIBLE);
            if (p.isFav()){

                holder.fav.setImageResource(R.drawable.ic_heart_red);
            }else{

                holder.fav.setImageResource(R.drawable.ic_heart);
            }
        }
        holder.name.setText(p.getName());
    }
}
```

```
holder.location.setText(p.getLocation());
holder.rate.setText(p.getRate());
Glide.with(context)
    .load(p.getImage())
    .into(holder.imgPlace);

holder.itemView.setOnClickListener(v -> {
    clickListener.onItemClick(p);
});
}

@Override
public int getItemCount() {
    // Return the size of your data list
    // For example: return dataList.size();
    return filteredPlaces.size();
}

public class ViewHolder extends RecyclerView.ViewHolder {

    TextView name;
    TextView location;
    TextView rate;
    ImageView imgPlace;
    LinearLayout btnFavPlace;
    CardView cardView;
    ImageView fav;

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

        btnFavPlace = itemView.findViewById(R.id.btnFavPlace);
        name = itemView.findViewById(R.id.tvPlaceName);
        location = itemView.findViewById(R.id.tvLocation);
        rate = itemView.findViewById(R.id.tvRating);
        imgPlace = itemView.findViewById(R.id.ivPlaceImg);
        cardView = itemView.findViewById(R.id.cvPlace);
        fav = itemView.findViewById(R.id.fav);

        LinearLayout.LayoutParams layoutParams;
        if (id==1){
            layoutParams = new LinearLayout.LayoutParams(
                dpToPx(context, 250),
                LinearLayout.LayoutParams.WRAP_CONTENT);
        }
    }
}
```

```
        }else{
            layoutParams = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        }
        cardView.setLayoutParams(layoutParams);

    }
}

@Override
public Filter getFilter() {
    return new Filter() {
        @Override
                    protected FilterResults
performFiltering(CharSequence constraint) {
                    String query =
constraint.toString().toLowerCase().trim();
                    List<Places> filtered = new ArrayList<>();

                    if (query.isEmpty()) {
                        filtered.addAll(places);
                    } else {
                        for (Places item : places) {
                            if
(item.getName().toLowerCase().contains(query)) {
                                filtered.add(item);
                            }
                        }
                    }
                    FilterResults filterResults = new
FilterResults();
                    filterResults.values = filtered;
                    return filterResults;
    }

    @Override
                    protected void publishResults(CharSequence
constraint, FilterResults results) {
                    filteredPlaces.clear();
                    filteredPlaces.addAll((List<Places>)
results.values);
                    notifyDataSetChanged();
    }
}
```

```
    };
}

public interface OnItemClickListener {
    void onItemClick(Places place);
}

public static int dpToPx(Context context, int dp) {
    float density = context.getResources().getDisplayMetrics().density;
    return Math.round(dp * density);
}
}
```

#### 1. Deklarasi dan Inisialisasi:

- Kode program mendeklarasikan dan menginisialisasi elemen-elemen yang diperlukan, seperti `Context`, `id`, `places`, dan `filteredPlaces`. Juga, ada deklarasi `OnItemClickListener` untuk menangani klik pada item.

#### 2. Konstruktor dan Method `updateData`:

- Konstruktor menerima `Context` dan `id` sebagai parameter, kemudian terdapat method `updateData` untuk memperbarui data set destinasi wisata yang ditampilkan.

#### 3. onCreateViewHolder:

- Ketika adapter memerlukan ViewHolder baru, method ini menginflate layout item (`place\_item\_layout`) sebagai tampilan setiap item dan mengembalikan objek `ViewHolder`.

#### 4. onBindViewHolder:

- Method ini mengikat data ke ViewHolder. Penggunaan kondisi `if (id == 1)` digunakan untuk menentukan apakah adapter digunakan di konteks tertentu yang memerlukan penanganan khusus. Jika ya, maka tombol favorit (`btnFabPlace`) disembunyikan. Jika tidak, tombol favorit ditampilkan, dan gambar hati (`fav`) diatur berdasarkan status "favorit" destinasi tersebut.

#### 5. ViewHolder:

- ViewHolder berfungsi untuk merepresentasikan setiap item di RecyclerView. Di dalamnya, dilakukan inisialisasi elemen-elemen UI seperti `btnFabPlace`, `name`, `location`, dll. Setelahnya, terdapat penyesuaian layout (`layoutParams`) berdasarkan kondisi `id`.

6. Filter dan Search (`getFilter` dan `performFiltering`):

- Implementasi filter digunakan untuk menyaring daftar destinasi berdasarkan input teks dari pengguna. Method `performFiltering` memproses data dan mengembalikan hasilnya, sedangkan method `publishResults` memperbarui daftar destinasi yang ditampilkan setelah penyaringan.

7. dpToPx Method:

- Method ini mengubah nilai dalam dp menjadi piksel sesuai dengan kepadatan layar perangkat, membantu menyesuaikan ukuran elemen dengan baik.

8. OnItemClickListener:

- Antarmuka ini mendefinisikan method `onItemClick` yang akan dipanggil ketika suatu item di-klik. Ini memungkinkan untuk menanggapi interaksi pengguna dengan setiap item destinasi.

9. onClickListener untuk ItemView:

- Penanganan klik pada setiap item dilakukan dengan menggunakan `clickListener.onItemClick(p)` , di mana `p` adalah objek `Places` yang terkait dengan item yang diklik.

Kode Program activity : PlaceDetailActivity

```
package com.android.travins.ui.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

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

import com.android.travins.R;
import com.android.travins.data.models.Places;
import
com.android.travins.databinding.ActivityPlaceDetailBinding;
import com.android.travins.ui.adapters.ViewPagerAdapter;
import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.tabs.TabLayoutMediator;
```

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

import java.util.ArrayList;
import java.util.Map;

public class PlaceDetailActivity extends AppCompatActivity {

    private ActivityPlaceDetailBinding binding;
    private boolean isFav;
    private ArrayList<String> tabsText = new ArrayList<>();

    private FirebaseFirestore db;
    private FirebaseAuth auth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityPlaceDetailBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        Places places = (Places) getIntent().getSerializableExtra("place");
        if (places!=null){
            binding.tvNamePlace.setText(places.getName());

            binding.tvLocationPlace.setText(places.getLocation());
            binding.tvRate.setText(places.getRate());
            Glide.with(this)
                .load(places.getImage())
                .into(binding.ivDetailPlace);
            fetchFavData(places);
        }

        tabsText.add("Overview");
        tabsText.add("Review");

        ViewPagerAdapter adapter = new
        ViewPagerAdapter(getSupportFragmentManager(),getLifecycle());
        adapter.setPlaces(places);
    }
}
```

```
        binding.pager.setAdapter(adapter);
        TabLayoutMediator tabLayoutMediator = new
TabLayoutMediator(binding.tabLayout,binding.pager, (tab,
position) ->
        tab.setText(tabsText.get(position))
    );
    tabLayoutMediator.attach();

    binding.btnBack.setOnClickListener(v -> {
        finish();
    });

    binding.btnFav.setOnClickListener(v -> {
        if (isFav) {

db.collection("users").document(auth.getCurrentUser().getUid())
.collection("favorites").document(places.getId()).delete().addOnCompleteListener(task -> {
            if (task.isSuccessful()){
                Toast.makeText(this, "Removed from
favorite.", Toast.LENGTH_SHORT).show();
                isFav = false;

binding.btnFav.setImageResource(R.drawable.ic_heart);
            }
        });
    }else{

db.collection("users").document(auth.getCurrentUser().getUid())
.collection("favorites").document(places.getId()).set(places).addOnCompleteListener(task -> {
            if (task.isSuccessful()){
                Toast.makeText(this, "Added to
favorite.", Toast.LENGTH_SHORT).show();
                isFav = true;

binding.btnFav.setImageResource(R.drawable.ic_heart_red);
            }
        });
    }
}

private void fetchFavData(Places places) {

db.collection("users").document(auth.getCurrentUser().getUid())

```

```

.collection("favorites")
    .get()
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document :
                task.getResult()) {
                Map<String, Object> data =
                document.getData();

                if
                (document.getId() .equals(places.getId())){
                    isFav = true;

                    binding.btnAddFav.setImageResource(R.drawable.ic_heart_red);
                }
            } else {
                }
            });
        }
    }
}

```

## 1. Menangani Tombol Favorit:

```

binding.btnAddFav.setOnClickListener(v -> {

    // Logika untuk menambah atau menghapus destinasi dari daftar favorit
    // pengguna

});

```

Tujuan: Memberikan logika untuk menambah atau menghapus destinasi dari daftar favorit pengguna berdasarkan status sebelumnya.

## 2. Mengambil Data Favorit Pengguna:

```

private void fetchFavData(Places places) {

    // Logika untuk mengambil data favorit pengguna dari Firebase Firestore
}

```

```
}
```

Tujuan: Mengecek apakah destinasi saat ini sudah ada dalam daftar favorit pengguna dan mengubah tampilan tombol favorit sesuai dengan hasil pengecekan.

### 3. Menampilkan Toast untuk Notifikasi:

```
Toast.makeText(this, "Removed from favorite.",  
Toast.LENGTH_SHORT).show();  
  
// atau  
  
Toast.makeText(this, "Added to favorite.", Toast.LENGTH_SHORT).show();
```

Tujuan: Memberikan notifikasi ke pengguna setelah berhasil menambah atau menghapus destinasi dari daftar favorit.

#### Kode fragment : FavoriteFragment

```
package com.android.travins.ui.fragments;  
  
import android.content.Intent;  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.Fragment;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Toast;  
  
import com.android.travins.R;  
import com.android.travins.data.models.Places;  
import com.android.travins.ui.activities.PlaceDetailActivity;  
import com.android.travins.ui.adapters.HorizontalAdapter;  
import com.google.android.gms.tasks.OnCompleteListener;
```

```
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firestore.QueryDocumentSnapshot;
import com.google.firebaseio.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class FavoriteFragment extends Fragment implements
HorizontalAdapter.OnItemClickListener{

    private FirebaseFirestore db;
    private FirebaseAuth auth;
    private HorizontalAdapter adapter;
    private ArrayList<Places> listDest = new ArrayList<>();

    private RecyclerView rvFav;
    public FavoriteFragment() {
        // Required empty public constructor
    }

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

    }

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

    @Override
    public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();
```

```

        rvFav = view.findViewById(R.id.rvFav);
        adapter = new HorizontalAdapter(this.getContext(), 2);
        adapter.setOnItemClickListener(this);

        fetchData();
        rvFav.setLayoutManager(new
LinearLayoutManager(this.getContext(),
LinearLayoutManager.VERTICAL, false));
        rvFav.setAdapter(adapter);
    }

    private void fetchData() {

        db.collection("users").document(auth.getCurrentUser().getUid())
            .collection("favorites")
                .get()
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document :
task.getResult()) {
                            Map<String, Object> data =
document.getData();
                            Places places = new
Places(document.getId(),
                                (String) data.get("name"),
                                (String) data.get("image"),
                                (String)
data.get("description"), (String) data.get("price"),
                                (String)
data.get("location"), (String) data.get("rate"),
                                (String)
data.get("total_review"));
                            places.setFav(true);
                            listDest.add(places);
                        }
                        adapter.updateData(listDest);
                    } else {
                        Toast.makeText(getActivity(), "Fetch
data failed", Toast.LENGTH_SHORT).show();
                    }
                });
}

public void onItemClick(Places place) {
    Intent intent = new Intent(this.getActivity(),

```

```
PlaceDetailActivity.class);
        intent.putExtra("place", place);
        startActivity(intent);
    }
}
```

1. Inisialisasi dan Setup: Menyiapkan objek-objek yang diperlukan seperti Firebase, adapter, dan RecyclerView.

Cara Kerja:

```
db = FirebaseFirestore.getInstance();

auth = FirebaseAuth.getInstance();

rvFav = view.findViewById(R.id.rvFav);

adapter = new HorizontalAdapter(this.getContext(),2);

adapter.setOnItemClickListener(this);

rvFav.setLayoutManager(new LinearLayoutManager(this.getContext(),
    LinearLayoutManager.VERTICAL, false));

rvFav.setAdapter(adapter);
```

Objek Firebase Firestore dan FirebaseAuth diinisialisasi. Selain itu, objek RecyclerView dan adapter `HorizontalAdapter` untuk menampilkan daftar destinasi wisata favorit diatur dan dihubungkan.

2. Fetch Data dari Firebase: Mengambil data destinasi favorit pengguna dari Firestore.

Cara Kerja:

```
db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites")

.get()

.addOnCompleteListener(task -> {

if (task.isSuccessful()) {
```

```
for (QueryDocumentSnapshot document : task.getResult()) {  
    // Mendapatkan data dari Firestore  
    Map<String, Object> data = document.getData();  
    // Membuat objek Places dari data  
    Places places = new Places(document.getId(),  
        (String) data.get("name"), (String) data.get("image"),  
        (String) data.get("description"), (String) data.get("price"),  
        (String) data.get("location"), (String) data.get("rate"),  
        (String) data.get("total_review"));  
    // Menandai destinasi sebagai favorit  
    places.setFav(true);  
    // Menambahkan objek Places ke dalam listDest  
    listDest.add(places);  
}  
// Memperbarui data dalam adapter  
adapter.updateData(listDest);  
} else {  
    // Menampilkan pesan kesalahan jika fetch data gagal  
    Toast.makeText(getActivity(),"Fetch failed",Toast.LENGTH_SHORT).show();  
}  
});
```

data

Data destinasi favorit pengguna diambil dari Firestore dengan mengakses koleksi "favorites" pada dokumen pengguna yang sedang login. Setiap dokumen diubah

menjadi objek `Places` dan ditandai sebagai favorit. List destinasi kemudian diperbarui, dan adapter diinformasikan untuk memperbarui tampilan.

### 3. Handle Item Click: Menanggapi klik pada item destinasi wisata favorit.

Cara Kerja:

```
public void onItemClick(Places place) {  
    Intent intent = new Intent(this.getActivity(),  
        PlaceDetailActivity.class);  
    intent.putExtra("place", place);  
    startActivity(intent);  
}
```

Implementasi metode `onItemClick` dari antarmuka `HorizontalAdapter.OnItemClickListener`. Ketika item di-klik, aktivitas `PlaceDetailActivity` dipanggil dan informasi tentang destinasi wisata dikirim sebagai ekstra melalui intent.

Dengan cara kerja tersebut, fragment "FavoriteFragment" mengelola dan menampilkan daftar destinasi wisata favorit pengguna dengan mengambil data dari Firebase, menggunakan adapter untuk menampilkan data tersebut di dalam RecyclerView, dan merespons klik pada setiap item untuk menampilkan detail destinasi.

## 2. Implementasi ListView/AdapterView/RecyclerView

Kode program dalam "PlaceDetailActivity" yang melibatkan elemen ListView/AdapterView/RecyclerView

PlaceDetailActivity

```
package com.android.travins.ui.activities;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.widget.Toast;  
  
import com.android.travins.R;  
import com.android.travins.data.models.Places;
```

```
import com.android.travins.databinding.ActivityPlaceDetailBinding;
import com.android.travins.ui.adapters.ViewPagerAdapter;
import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.tabs.TabLayoutMediator;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class PlaceDetailActivity extends AppCompatActivity {

    private ActivityPlaceDetailBinding binding;
    private boolean isFav;
    private ArrayList<String> tabsText = new ArrayList<>();

    private FirebaseFirestore db;
    private FirebaseAuth auth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityPlaceDetailBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        Places places = (Places) getIntent().getSerializableExtra("place");
        if (places!=null){
            binding.tvNamePlace.setText(places.getName());
            binding.tvLocationPlace.setText(places.getLocation());
            binding.tvRate.setText(places.getRate());
            Glide.with(this)
                .load(places.getImage())
                .into(binding.ivDetailPlace);
            fetchFavData(places);
        }

        tabsText.add("Overview");
        tabsText.add("Review");

        ViewPagerAdapter adapter = new ViewPagerAdapter(getSupportFragmentManager(),getLifecycle());
        adapter.setPlaces(places);

        binding.pager.setAdapter(adapter);
        TabLayoutMediator tabLayoutMediator = new
```

```

TabLayoutMediator(binding.tabLayout,binding.pager,(tab, position) ->
    tab.setText(tabsText.get(position))
);
tabLayoutMediator.attach();

binding.btnExit.setOnClickListener(v -> {
    finish();
});

binding.btnAdd.setOnClickListener(v -> {
    if (isAdd) {

        db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites").document(places.getId()).set(places).addOnCompleteListener(task -> {
            if (task.isSuccessful()){
                Toast.makeText(this, "Added to favorite.", Toast.LENGTH_SHORT).show();
                isAdd = false;
            }
        });
    } else{

        db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites").document(places.getId()).delete().addOnCompleteListener(task -> {
            if (task.isSuccessful()){
                Toast.makeText(this, "Removed from favorite.", Toast.LENGTH_SHORT).show();
                isAdd = true;
            }
        });
    }
}

private void fetchAddData(Places places) {

db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites")
    .get()
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document : task.getResult()) {
                Map<String, Object> data =
                    document.getData();
                if (data.get("name") != null)
                    addList.add(data);
            }
        }
    })
}

```

```
(document.getId().equals(places.getId())) {
    isFav = true;

binding.btnAddFav.setImageResource(R.drawable.ic_heart_red);
}
} else {
}
});
```

## 1. Inisialisasi dan Setup:

```
binding = ActivityPlaceDetailBinding.inflate(LayoutInflater());  
  
setContentView(binding.getRoot());  
  
db = FirebaseFirestore.getInstance();  
  
auth = FirebaseAuth.getInstance();
```

- Membuat objek `binding` menggunakan `ActivityPlaceDetailBinding` yang dihasilkan dari layout XML dengan menggunakan data binding.
  - Menginisialisasi objek Firebase Firestore dan FirebaseAuth.

## 2. Mengambil Data Destinasi:

```
Places places = (Places) getIntent().getSerializableExtra("place");

if (places != null) {
    // Set data destinasi ke dalam elemen tampilan
    //
    // ...
    // Memanggil metode fetchFavData untuk mengecek apakah destinasi
    // sudah ditandai sebagai favorit.
    fetchFavData(places);
}
```

- Mengambil objek `Places` yang telah dikirim melalui intent dari activity sebelumnya.
- Mengisi elemen tampilan seperti nama, lokasi, rating, dan gambar destinasi menggunakan data dari objek `places`.
- Memanggil metode `fetchFavData` untuk mengecek apakah destinasi sudah ditandai sebagai favorit.

### 3. Pengaturan ViewPager dan TabLayout:

```
ViewPagerAdapter adapter = new
ViewPagerAdapter(getSupportFragmentManager(), getLifecycle());
adapter.setPlaces(places);

binding.pager.setAdapter(adapter);

TabLayoutMediator tabLayoutMediator = new
TabLayoutMediator(binding.tabLayout, binding.pager, (tab, position) ->
    tab.setText(tabsText.get(position))
);
tabLayoutMediator.attach();
```

- Membuat objek `ViewPagerAdapter` dan mengatur datanya dengan objek `places`.
- Menetapkan adapter ke ViewPager (`binding.pager`).
- Mengaitkan TabLayout dengan ViewPager menggunakan `TabLayoutMediator` untuk menampilkan judul tab yang sesuai dengan posisi.

### 4. Menanggapi Tombol Kembali:

```
binding.btnExit.setOnClickListener(v -> {
    finish();
});
```

```
});
```

- Menambahkan listener pada tombol kembali untuk menutup aktivitas saat tombol tersebut di klik.

## 5. Menanggapi Tombol Favorit:

```
binding.btnFav.setOnClickListener(v -> {
    // ...
});
```

- Menambahkan listener pada tombol favorit untuk menambah atau menghapus destinasi dari daftar favorit pengguna.

## 6. Tambah/Hapus Destinasi dari Favorit:

```
if (isFav) {
    // Hapus destinasi dari daftar favorit

    db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites")
        .document(places.getId()).delete().addOnCompleteListener(task -> {
            // ...
        });

} else {
    // Tambahkan destinasi ke daftar favorit

    db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites")
        .document(places.getId()).set(places).addOnCompleteListener(task -> {
```

```
// ...  
});  
}
```

- Mengecek apakah destinasi sudah ditandai sebagai favorit (`isFav`).
- Jika sudah favorit, maka destinasi dihapus dari daftar favorit. Jika belum, maka destinasi ditambahkan ke daftar favorit.
- Memberikan umpan balik berupa pesan singkat menggunakan `Toast` setelah operasi selesai.

## 7. Mengecek Apakah Destinasi Sudah Favorit:

```
private void fetchFavData(Places places) {  
  
    db.collection("users").document(auth.getCurrentUser().getUid()).collection("fa  
vorites")  
  
        .get()  
        .addOnCompleteListener(task -> {  
            // ...  
        });  
}
```

- Mengambil data dari koleksi "favorites" pada Firestore untuk pengguna yang sedang login.
- Memeriksa apakah destinasi (`places`) sudah ada di dalam daftar favorit. Jika ya, maka tombol favorit ditandai sebagai favorit (`isFav` diatur menjadi `true`).

## Penjelasan Kode Adapter “HorizontalAdapter”

```
package com.android.travins.ui.adapters;  
  
import static com.google.android.material.internal.ViewUtils.dpToPx;
```

```
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Filter;
import android.widget.Filterable;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.bumptech.glide.Glide;

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

public class HorizontalAdapter extends RecyclerView.Adapter<HorizontalAdapter.ViewHolder> implements Filterable {
    private Context context;
    private Integer id;
    private ArrayList<Places> places = new ArrayList<>();
    private ArrayList<Places> filteredPlaces = new ArrayList<>();

    private OnItemClickListener clickListener;

    public void setOnItemClickListener(OnItemClickListener clickListener) {
        this.clickListener = clickListener;
    }

    // Constructor to receive the context
    public HorizontalAdapter(Context context, Integer id) {
        this.context = context;
        this.id = id;
    }

    public void updateData(ArrayList<Places> places) {
        this.places = places;
        this.filteredPlaces = new ArrayList<>(places);
        notifyDataSetChanged();
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
    {
```

```
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.place_item_layout, parent, false);
        return new ViewHolder(view);
    }

@Override
public void onBindViewHolder(ViewHolder holder, int position) {
    // Bind data or perform operations for each item
    Places p = filteredPlaces.get(position);

    if (id == 1){
        holder.btnFavPlace.setVisibility(View.GONE);
    }else{
        holder.btnFavPlace.setVisibility(View.VISIBLE);
        if (p.isFav()){
            holder.fav.setImageResource(R.drawable.ic_heart_red);
        }else{
            holder.fav.setImageResource(R.drawable.ic_heart);
        }
    }
    holder.name.setText(p.getName());
    holder.location.setText(p.getLocation());
    holder.rate.setText(p.getRate());
    Glide.with(context)
        .load(p.getImage())
        .into(holder.imgPlace);

    holder.itemView.setOnClickListener(v -> {
        clickListener.onItemClick(p);
    });
}

@Override
public int getItemCount() {
    // Return the size of your data list
    // For example: return dataList.size();
    return filteredPlaces.size();
}

public class ViewHolder extends RecyclerView.ViewHolder {

    TextView name;
    TextView location;
    TextView rate;
    ImageView imgPlace;
    LinearLayout btnFavPlace;
    CardView cardView;
    ImageView fav;

    public ViewHolder(View itemView) {
        super(itemView);
```

```

        btnFavPlace = itemView.findViewById(R.id.btnFavPlace);
        name = itemView.findViewById(R.id.tvPlaceName);
        location = itemView.findViewById(R.id.tvLocation);
        rate = itemView.findViewById(R.id.tvRating);
        imgPlace = itemView.findViewById(R.id.ivPlaceImg);
        cardView = itemView.findViewById(R.id.cvPlace);
        fav = itemView.findViewById(R.id.fav);

        LinearLayout.LayoutParams layoutParams;
        if (id==1){
            layoutParams = new LinearLayout.LayoutParams(
                dpToPx(context, 250),
                LinearLayout.LayoutParams.WRAP_CONTENT);
        }else{
            layoutParams = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        }
        cardView.setLayoutParams(layoutParams);

    }

}

@Override
public Filter getFilter() {
    return new Filter() {
        @Override
            protected FilterResults performFiltering(CharSequence constraint) {
                String query =
constraint.toString().toLowerCase().trim();
                List<Places> filtered = new ArrayList<>();

                if (query.isEmpty()) {
                    filtered.addAll(places);
                } else {
                    for (Places item : places) {
if
                        if (item.getName().toLowerCase().contains(query)) {
                            filtered.add(item);
                        }
                    }
                }

                FilterResults filterResults = new FilterResults();
                filterResults.values = filtered;
                return filterResults;
            }

        @Override
            protected void publishResults(CharSequence constraint,
FilterResults results) {

```

```

        filteredPlaces.clear();
        filteredPlaces.addAll((List<Places>) results.values);
        notifyDataSetChanged();
    }
}

public interface OnItemClickListener {
    void onItemClick(Places place);
}

public static int dpToPx(Context context, int dp) {
    float density = context.getResources().getDisplayMetrics().density;
    return Math.round(dp * density);
}
}

```

Kode program ini adalah implementasi dari adapter `HorizontalAdapter` untuk RecyclerView yang digunakan dalam konteks tampilan horizontal. Berikut penjelasannya:

#### 1. Variabel dan ArrayList:

- Tujuan: Menyimpan data dan konfigurasi adapter.
- Cara Kerja: Variabel dan ArrayList digunakan untuk menyimpan konteks (`context`), identifikasi adapter (`id`), data destinasi wisata (`places`), serta data destinasi yang sudah difilter (`filteredPlaces`).

#### 2. Konstruktor dan Inisialisasi:

- Tujuan: Menginisialisasi objek adapter dengan konteks dan ID.
- Cara Kerja: Konstruktor menerima `Context` dan `id` sebagai parameter, dan `updateData` digunakan untuk menginisialisasi data awal adapter.

#### 3. updateData:

- Tujuan: Mengupdate data dalam adapter.
- Cara Kerja: Method ini mengupdate ArrayList `places` dan `filteredPlaces`, lalu memberi tahu adapter untuk merender ulang tampilan.

#### 4. onCreateViewHolder:

- Tujuan: Membuat dan menginisialisasi tampilan holder untuk setiap item dalam RecyclerView.
- Cara Kerja: Inflasi layout item dari XML (`R.layout.place\_item\_layout`) untuk menciptakan dan mengembalikan instance dari `ViewHolder`.

#### 5. onBindViewHolder:

- Tujuan: Mengisi data ke dalam tampilan holder untuk setiap item dalam

### RecyclerView.

- Cara Kerja: Method ini mengisi tampilan holder dengan data dari `filteredPlaces`. Jika `id` sama dengan 1, tombol favorit disembunyikan; jika tidak, tombol favorit ditampilkan dan diatur berdasarkan status favorit dari destinasi. Data seperti nama, lokasi, rating, dan gambar destinasi juga diisikan ke dalam tampilan.

### 6. getItemCount:

- Tujuan: Mendapatkan jumlah item dalam RecyclerView.
- Cara Kerja: Mengembalikan ukuran dari `filteredPlaces` .

### 7. ViewHolder:

- Tujuan: Merepresentasikan tampilan setiap item dalam RecyclerView.
- Cara Kerja: Membuat dan menginisialisasi tampilan holder, termasuk beberapa elemen seperti teks, gambar, dan tombol favorit. Layout parameter untuk `CardView` diatur berdasarkan kondisi `id` .

### 8. getFilter:

- Tujuan: Menyediakan filter untuk pencarian.
- Cara Kerja: Menerapkan filter ke data destinasi wisata berdasarkan nama. Data yang difilter diperbarui dengan hasil pencarian dan adapter memberitahu untuk merender ulang tampilan.

### 9. OnItemClickListener:

- Tujuan: Menanggapi klik pada item dalam RecyclerView.
- Cara Kerja: Interface yang memberikan callback kepada kelas pemanggil ketika item di-klik.

### 10. dpToPx:

- Tujuan: Mengkonversi nilai dalam dp menjadi piksel.
- Cara Kerja: Method statis untuk mengkonversi nilai dalam dp menjadi piksel berdasarkan density layar perangkat.

Kode program ini menyediakan adapter yang dapat menangani tampilan horizontal RecyclerView dengan baik, menyediakan fitur-fitur seperti menampilkan atau menyembunyikan tombol favorit berdasarkan kondisi tertentu, dan mendukung filter pencarian berdasarkan nama destinasi.

### Penjelasan Kode class Objek

```
public class Places {  
    // ...  
}
```

Class Objek (Places): Merepresentasikan objek tempat wisata dengan atribut seperti nama, gambar, deskripsi, harga, lokasi, rating, dan jumlah ulasan. Beserta Fungsional Fitur dari Tombol Favorit yang divisualisasikan menggunakan “Icon Heart”

Atribut: Atribut-atribut yang mencerminkan karakteristik tempat wisata.

### 3. Implementasi Fragment

FavoritFragment

```
package com.android.travins.ui.fragments;

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

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

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.android.travins.ui.activities.PlaceDetailActivity;
import com.android.travins.ui.adapters.HorizontalAdapter;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class FavoriteFragment extends Fragment implements HorizontalAdapter.OnItemClickListerner{
```

```
private FirebaseFirestore db;
private FirebaseAuth auth;
private HorizontalAdapter adapter;
private ArrayList<Places> listDest = new ArrayList<>();

private RecyclerView rvFav;
public FavoriteFragment() {
    // Required empty public constructor
}

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

}

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

@Override
public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    db = FirebaseFirestore.getInstance();
    auth = FirebaseAuth.getInstance();

    rvFav = view.findViewById(R.id.rvFav);
    adapter = new HorizontalAdapter(this.getContext(), 2);
    adapter.setOnItemClickListener(this);

    fetchData();
    rvFav.setLayoutManager(new LinearLayoutManager(this.getContext(),
                                               LinearLayoutManager.VERTICAL, false));
    rvFav.setAdapter(adapter);
}

private void fetchData() {
```

```

db.collection("users").document(auth.getCurrentUser().getUid())
    .collection("favorites")
        .get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document :
                    task.getResult()) {
                    Map<String, Object> data =
                    document.getData();
                    Places places = new
                    Places(document.getId(),
                        (String) data.get("name"),
                        (String) data.get("image"),
                        (String) data.get("description"),
                        (String) data.get("price"),
                        (String) data.get("location"),
                        (String) data.get("rate"),
                        (String) data.get("total_review"));
                    places.setFav(true);
                    listDest.add(places);
                }
                adapter.updateData(listDest);
            } else {
                Toast.makeText(getActivity(), "Fetch
data failed", Toast.LENGTH_SHORT).show();
            }
        });
    }

    public void onItemClick(Places place) {
        Intent intent = new Intent(this.getActivity(),
        PlaceDetailActivity.class);
        intent.putExtra("place", place);
        startActivity(intent);
    }
}

```

Kode program di atas adalah implementasi dari sebuah Fragment dalam aplikasi Android yang memuat dan menampilkan daftar destinasi wisata favorit pengguna. Berikut adalah penjelasan singkatnya:

## 1. Inisialisasi dan Setup:

- Tujuan: Menyiapkan objek-objek yang dibutuhkan seperti Firebase, adapter, dan RecyclerView.
- Cara Kerja: Dalam metode `onViewCreated`, objek Firebase Firestore (`db`) dan Firebase Authentication (`auth`) diinisialisasi. Selanjutnya, RecyclerView (`rvFav`) dan adapter (`HorizontalAdapter`) disetup.

## 2. Fetch Data dari Firebase:

- Tujuan: Mengambil data destinasi wisata favorit pengguna dari Firestore.
- Cara Kerja: Dalam metode `fetchData`, dilakukan pengambilan data dari Firestore dengan query ke koleksi "favorites" pada dokumen pengguna yang sedang login. Data tersebut kemudian diubah menjadi objek `Places` dan dimasukkan ke dalam ArrayList `listDest`. Adapter kemudian di-update dengan data yang baru.

## 3. Handle Item Click:

- Tujuan: Menanggapi klik pada item dalam RecyclerView.
- Cara Kerja: Implementasi metode `onItemClick` dari antarmuka `HorizontalAdapter.OnItemClickListener`. Ketika item di-klik, intent dibuat untuk memulai aktivitas `PlaceDetailActivity` dengan menyertakan objek `Places` sebagai ekstra.

## 4. Layout Inflation:

- Tujuan: Menentukan layout tampilan Fragment.
- Cara Kerja: Dalam metode `onCreateView`, layout fragment di-inflate dari XML menggunakan `inflater.inflate`.

## 5. Menyusun RecyclerView:

- Tujuan: Menetapkan layout manager dan adapter untuk RecyclerView.
- Cara Kerja: Dalam metode `onViewCreated`, layout manager (LinearLayoutManager) dan adapter (`HorizontalAdapter`) diatur untuk RecyclerView. Metode `fetchData` dipanggil untuk mengambil data dari Firebase dan memuatnya ke dalam adapter.

## 6. Pesan Toast pada Gagal Fetch Data:

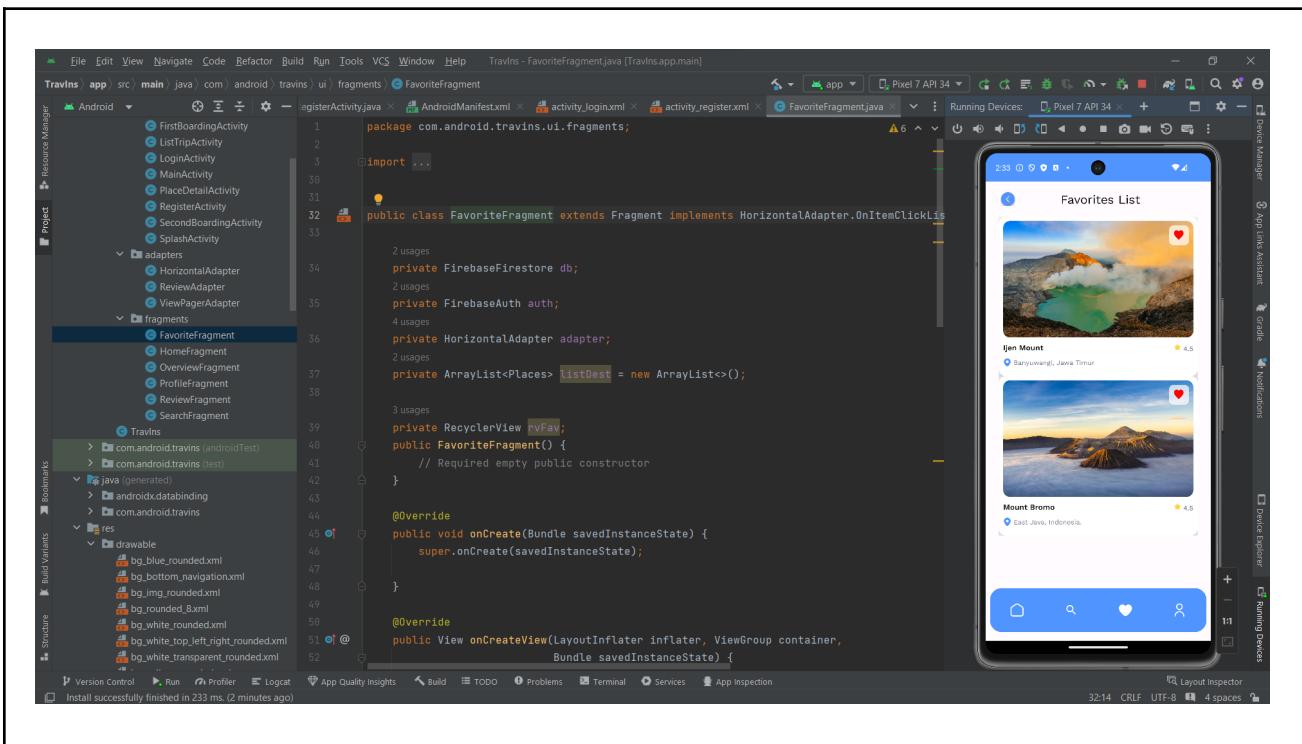
- Tujuan: Memberikan informasi kepada pengguna jika pengambilan data gagal.
  - Cara Kerja: Dalam metode `fetchData`, jika pengambilan data gagal, pesan Toast ditampilkan untuk memberi tahu pengguna bahwa proses pengambilan data tidak berhasil.

## 7. Navigasi ke Detail Destinasi:

- Tujuan: Mengarahkan pengguna ke halaman detail destinasi ketika salah satu item diklik.
  - Cara Kerja: Dalam metode `onItemClick`, intent dibuat untuk memulai `PlaceDetailActivity` dengan menyertakan objek `Places` sebagai ekstra. Intent ini kemudian dijalankan untuk membuka halaman detail destinasi.

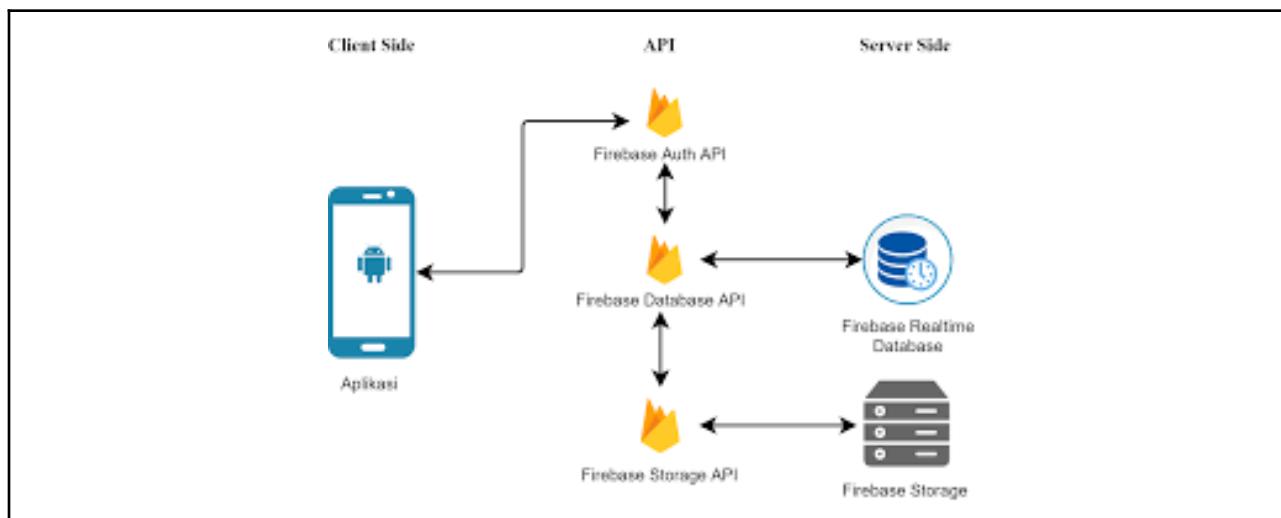
Kode tersebut menciptakan tampilan Fragment yang menampilkan daftar destinasi wisata favorit pengguna dalam RecyclerView. Setiap item dalam daftar dapat di-klik untuk membuka halaman detail destinasi.

## Screenshot tampilan fragment:



## 4. Implementasi Komunikasi Data

Gambarkan arsitektur komunikasi data client-server yang terlibat dalam fungsional yang diuraikan. Uraikan cara kerjanya, baik itu terkait format data yang digunakan mekanisme pengiriman, penerimaan, dan proses datanya sehingga fungsional yang diuraikan bisa terpenuhi/terselesaikan



Gambar tersebut memberikan gambaran umum tentang cara kerja API. API adalah alat yang penting yang memungkinkan aplikasi berkomunikasi satu sama lain dan mengakses data dan layanan dari server. Dalam diagram tersebut, ada dua sisi utama API: sisi klien dan sisi server. Sisi klien adalah aplikasi yang menggunakan API. Aplikasi ini dapat berupa aplikasi web, aplikasi seluler, atau aplikasi desktop. Aplikasi klien biasanya menggunakan API untuk mengakses data atau layanan yang disediakan oleh server. Dalam diagram tersebut, sisi klien diwakili oleh ikon telepon Android. Telepon Android dapat menggunakan API Firebase untuk mengakses data dan layanan dari server Firebase.

Penjelasan tentang masing-masing API :

### 1. Firebase Auth API

Firebase Auth API menyediakan berbagai fitur untuk mengelola autentikasi dan otorisasi pengguna, termasuk:

- Registrasi pengguna
- Masuk dengan akun Google atau email

- Verifikasi email
- Kelola akun pengguna

## 2. Firebase Database API

Firebase Database API adalah database real-time yang memungkinkan aplikasi klien untuk menyimpan dan mengambil data secara real-time. Data disimpan dalam format JSON dan dapat diakses dari aplikasi klien menggunakan API.

## 3. Firebase Storage API

Firebase Storage API adalah layanan penyimpanan file yang memungkinkan aplikasi klien untuk menyimpan file di cloud. File dapat diakses dari aplikasi klien menggunakan API.

# Use Case 2: Create, Update, & Read User Profile

Fatikah Izza Maulidina S.

NIM 215150401111020



## Deskripsi Fungsional

Use case Create, Update, & Read User Profile adalah bagian penting dalam aplikasi "TravIns" yang memungkinkan pengguna untuk membuat, memperbarui, dan membaca profil pengguna mereka di dalam aplikasi. Berikut adalah deskripsi fungsional dari masing-masing fungsi ini:

### 1. Create User Profile:

- Deskripsi Fungsional: Fitur ini memungkinkan pengguna baru untuk membuat profil pribadi mereka di dalam aplikasi "TravIns".

- Tujuan: Tujuan dari pembuatan profil pengguna adalah untuk mengumpulkan informasi yang relevan tentang pengguna, seperti full name, username, gender, birth of date, country, and address.

- Cara Kerja: Ketika pengguna baru mendaftar atau membuat akun di aplikasi, mereka akan diminta untuk mengisi formulir atau menyediakan informasi seperti full name, alamat email/phone number, password, and confirm password. Informasi ini akan disimpan dalam basis data aplikasi. Selanjutnya, pengguna dapat mengklik icon profil pada navigasi bar di halaman home/beranda yang akan diarahkan pada halaman profil. Setelah itu, pengguna dapat mengklik edit profile information and

menambahkan informasi profil yang belum terisi pada halaman edit profil. Ketika pengguna telah membuat identitas profil yang lengkap pada aplikasi tersebut, pengguna perlu mengklik “Save changes” untuk menyimpan data profil yang telah dibuat.

## 2. Update User Profile:

- Deskripsi Fungsional: Fitur ini memungkinkan pengguna untuk memperbarui informasi yang ada dalam profil pengguna mereka.

- Tujuan: Tujuan dari pembaruan profil pengguna adalah untuk memungkinkan pengguna untuk memperbarui informasi pribadi mereka sesuai dengan perubahan yang mungkin terjadi, seperti alamat baru, perubahan nama, perubahan username, atau perubahan negara.

- Cara Kerja: Pengguna dapat mengakses bagian ikon profil pada halaman budi aplikasi "TravIns" dan melakukan perubahan yang diperlukan, seperti mengubah full name, username, gender, birth of date, country, atau address yang dimiliki. Pengguna lalu menyimpan data dengan meng klik Save Changes dan data akan berubah.

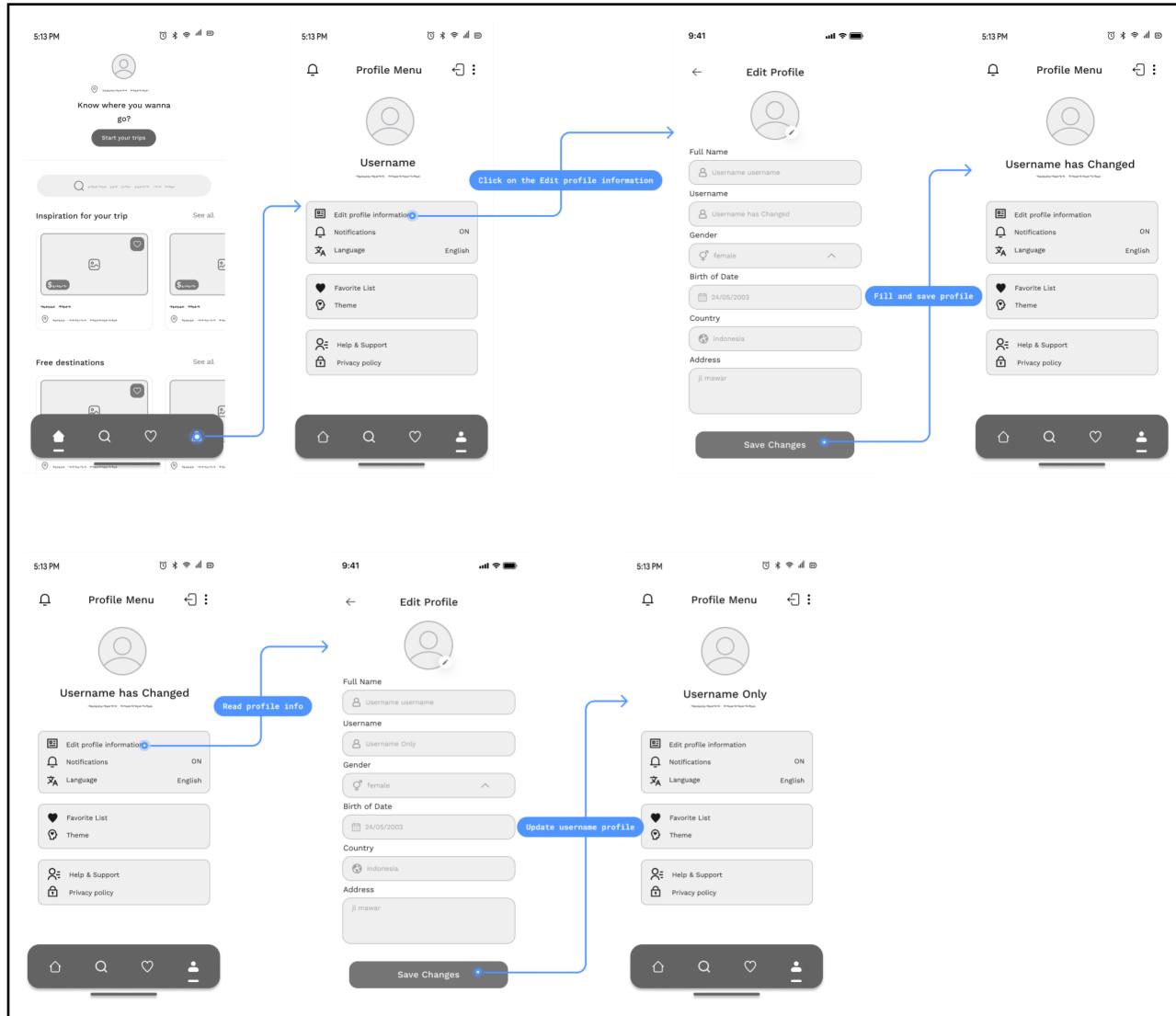
## 3. Read User Profile:

- Deskripsi Fungsional: Fitur ini memungkinkan pengguna dan sistem untuk membaca atau melihat profil pengguna yang ada.

- Tujuan: Tujuan dari membaca profil pengguna adalah untuk memberikan akses kepada pengguna untuk melihat informasi yang mereka miliki dalam profil mereka, dan memungkinkan sistem untuk mengakses informasi tersebut untuk memberikan pengalaman yang disesuaikan.

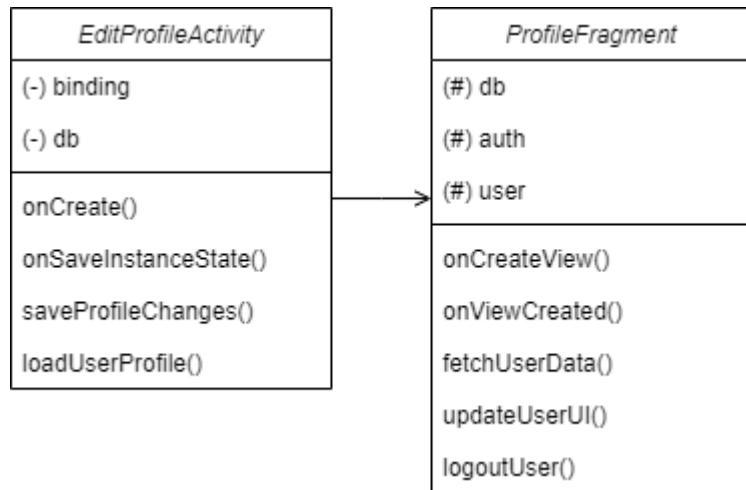
- Cara Kerja: Pengguna dapat mengakses profil mereka melalui aplikasi dan melihat informasi yang telah mereka berikan sebelumnya. Sistem juga dapat menggunakan informasi ini untuk menyesuaikan rekomendasi destinasi wisata, menawarkan promo khusus berdasarkan preferensi pengguna, atau menyajikan konten yang relevan.

# Rancangan Screenflow



Pada rancangan screenflow tersebut dimulai dari halaman beranda yang memiliki icon profil untuk menuju ke halaman profil menu. Pada halaman tersebut terlihat username dan negara pengguna sebagai informasi data yang ditampilkan. Pengguna perlu mengklik tulisan “Edit profile information” untuk melihat dan membuat inputan kebutuhan profile. Selanjutnya, ketika pengguna mengklik “Save changes” maka profile pengguna telah dibuat dan dapat dibaca kembali pada halaman edit profile tersebut. Pengguna juga dapat mengupdate profilnya seperti username atau country dan akan ditampilkan username baru yang telah terupdate pada halaman profile menu.

# Class Diagram



## 1. EditProfileActivity:

Attributes:

1. `binding`: Private attribute bertipe `ActivityEditProfileBinding` untuk menyimpan referensi ke layout activity.
2. `db`: Private attribute bertipe `FirebaseFirestore` untuk berinteraksi dengan database Firestore.

Methods:

1. `onCreate()`: Metode yang dijalankan saat activity dibuat.
2. `onSaveInstanceState()`: Metode untuk menyimpan status activity saat perubahan konfigurasi.
3. `saveProfileChanges()`: Private method untuk menyimpan perubahan profil pengguna ke database Firestore.
4. `loadUserProfile()`: Private method untuk memuat profil pengguna dari database Firestore.

## 2. ProfileFragment:

Attributes :

1. `db`: Protected attribute bertipe `FirebaseFirestore` untuk berinteraksi dengan database Firestore.
2. `auth`: Protected attribute bertipe `FirebaseAuth` untuk otentikasi pengguna.
3. `user`: Protected attribute bertipe `User` untuk menyimpan data pengguna.

Methods:

1. `onCreateView()`: Metode untuk membuat tampilan (View) dari fragmen.
2. `onViewCreated()`: Metode untuk menginisialisasi komponen UI dan melakukan pengaturan tampilan.
3. `fetchUserData()`: Private method untuk mengambil data pengguna dari database Firestore berdasarkan email pengguna yang sedang login.
4. `updateUserUI()`: Private method untuk memperbarui antarmuka pengguna (UI) dengan informasi dari objek user.
5. `logoutUser()`: Private method untuk proses logout pengguna dari aplikasi.

Dalam class diagram tersebut, `EditProfileActivity` dan `ProfileFragment` merupakan dua kelas yang memiliki atribut dan method yang berbeda untuk mengatur interaksi antara pengguna, komponen antarmuka pengguna, dan database Firestore. Class diagram membantu dalam visualisasi struktur kelas beserta hubungan dan fungsionalitasnya dalam suatu aplikasi.

# Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:name=".TravIns"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.TravIns"
        tools:targetApi="31">
        <meta-data
            android:name="com..sdk.ApplicationId"
            android:value="@string/_app_id"/>

        <activity
            android:name=".ui.activities.PlaceDetailActivity"
            android:exported="false"
            android:windowSoftInputMode="adjustPan"/>

        <activity
            android:name=".ui.activities.SplashActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".ui.activities.EditProfileActivity"
            android:exported="false" />
        <activity
            android:name=".ui.activities.ListTripActivity"
            android:exported="false" />
        <activity
            android:name=".ui.activities.RegisterActivity"
            android:exported="false" />
        <activity
            android:name=".ui.activities.LoginActivity"
```

```
        android:exported="false" />
    <activity
        android:name=".ui.activities.SecondBoardingActivity"
        android:exported="false" />
    <activity
        android:name=".ui.activities.FirstBoardingActivity"
        android:exported="false" />
    <activity
        android:name=".ui.activities.MainActivity"
        android:exported="false"></activity>
</application>

</manifest>
```

Fungsionalitas aplikasi memerlukan permission com.example.project.DEBIT\_ACCT yang digunakan untuk melakukan koneksi jaringan untuk berkomunikasi dengan server atau layanan online lainnya. Permission tersebut dideklarasikan dalam file manifest berikut:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Fungsionalitas ini menggunakan sebuah Activity dengan nama EditProfileActivity yang dideklarasikan dengan elemen berikut dalam file AndroidManifest.xml:

```
<activity
    android:name=".ui.activities.EditProfileActivity"
    android:exported="false" />
```

# Implementasi UI

## Layout 1: [fragment\_profile]

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ui.fragments.ProfileFragment">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:paddingHorizontal="8dp"
        android:gravity="center_horizontal">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingVertical="16dp">

            <ImageView
                android:id="@+id/btnNotif"
                android:layout_width="34dp"
                android:layout_height="34dp"
                app:layout_constraintLeft_toLeftOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintBottom_toBottomOf="parent"
                android:src="@drawable/ic_notification"
                app:tint="@color/blueSecondary" />

            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                app:layout_constraintLeft_toRightOf="@+id/btnNotif"
                android:text="Profile Menu"
                android:fontFamily="@font/worksans_semiBold"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintRight_toLeftOf="@+id/btnLogout"
                android:textAlignment="center"
                android:paddingLeft="8dp"
                android:textSize="24sp"/>

            <ImageView
                android:id="@+id/btnLogout"
                android:layout_width="34dp"
                android:layout_height="34dp" />
        
```

```
        android:layout_height="34dp"
        app:layout_constraintRight_toLeftOf="@+id/btnMore"
        app:layout_constraintTop_toTopOf="parent"
        app:tint="@color/blueSecondary"
        android:src="@drawable/ic_logout"/>

    <ImageView
        android:id="@+id/btnMore"
        android:layout_width="34dp"
        android:layout_height="34dp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:tint="@color/blueSecondary"
        android:src="@drawable/ic_more"/>

</androidx.constraintlayout.widget.ConstraintLayout>

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_profile"/>

<TextView
    android:id="@+id/profileFullName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="User"
    android:fontFamily="@font/worksans_semibold"
    android:textSize="18sp"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableStart="@drawable/ic_location"
    android:text=" Indonesia"
    android:fontFamily="@font/worksans_regular"/>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    android:layout_marginTop="16dp"
    android:layout_marginHorizontal="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="12dp">

        <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"

        <TextView
            android:id="@+id(btnEditProfile"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Edit profile information"
            android:fontFamily="@font/worksans_regular"
            android:drawablePadding="16dp"
            android:paddingVertical="4dp"

        android:drawableStart="@drawable/ic_profile_info"/>

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingVertical="4dp">
            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="Notifications"
                android:fontFamily="@font/worksans_regular"
                android:drawablePadding="16dp"

        android:drawableStart="@drawable/ic_notification"
                android:drawableTint="@color/blueSecondary"
                app:layout_constraintTop_toTopOf="parent"

            app:layout_constraintLeft_toLeftOf="parent"/>

            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="ON"

            app:layout_constraintRight_toRightOf="parent"
                android:fontFamily="@font/worksans_regular"
                app:layout_constraintTop_toTopOf="parent"
                android:textColor="@color/blueSecondary"/>
        </androidx.constraintlayout.widget.ConstraintLayout>

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingVertical="4dp">
            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="Language"
                android:fontFamily="@font/worksans_regular"
```

```
        android:drawablePadding="16dp"

    android:drawableStart="@drawable/ic_language"
        android:drawableTint="@color/blueSecondary"
        app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintLeft_toLeftOf="parent"/>

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="English"

    app:layout_constraintRight_toRightOf="parent"
        android:fontFamily="@font/worksans_regular"
        app:layout_constraintTop_toTopOf="parent"
        android:textColor="@color/blueSecondary"/>
    </androidx.constraintlayout.widget.ConstraintLayout>
</LinearLayout>

</LinearLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    android:layout_marginTop="16dp"
    android:layout_marginHorizontal="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="12dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Favorite List"
                android:fontFamily="@font/worksans_regular"
                android:drawablePadding="16dp"
                android:paddingVertical="4dp"
                android:drawableStart="@drawable/ic_heart_red"/>

```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Theme"
    android:fontFamily="@font/worksans_regular"
    android:drawablePadding="16dp"
    android:drawableStart="@drawable/ic_theme"
    android:drawableTint="@color/blueSecondary"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"/>
</LinearLayout>
</LinearLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    android:layout_marginTop="16dp"
    android:layout_marginHorizontal="16dp"
    android:layout_marginBottom="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="12dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Help & Support"
                android:fontFamily="@font/worksans_regular"
                android:drawablePadding="16dp"
                android:paddingVertical="4dp"
                android:drawableStart="@drawable/ic_help"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Privacy policy"
                android:fontFamily="@font/worksans_regular"
                android:drawablePadding="16dp"
                android:drawableStart="@drawable/ic_lock"
                android:drawableTint="@color/blueSecondary"
                app:layout_constraintTop_toTopOf="parent"
```

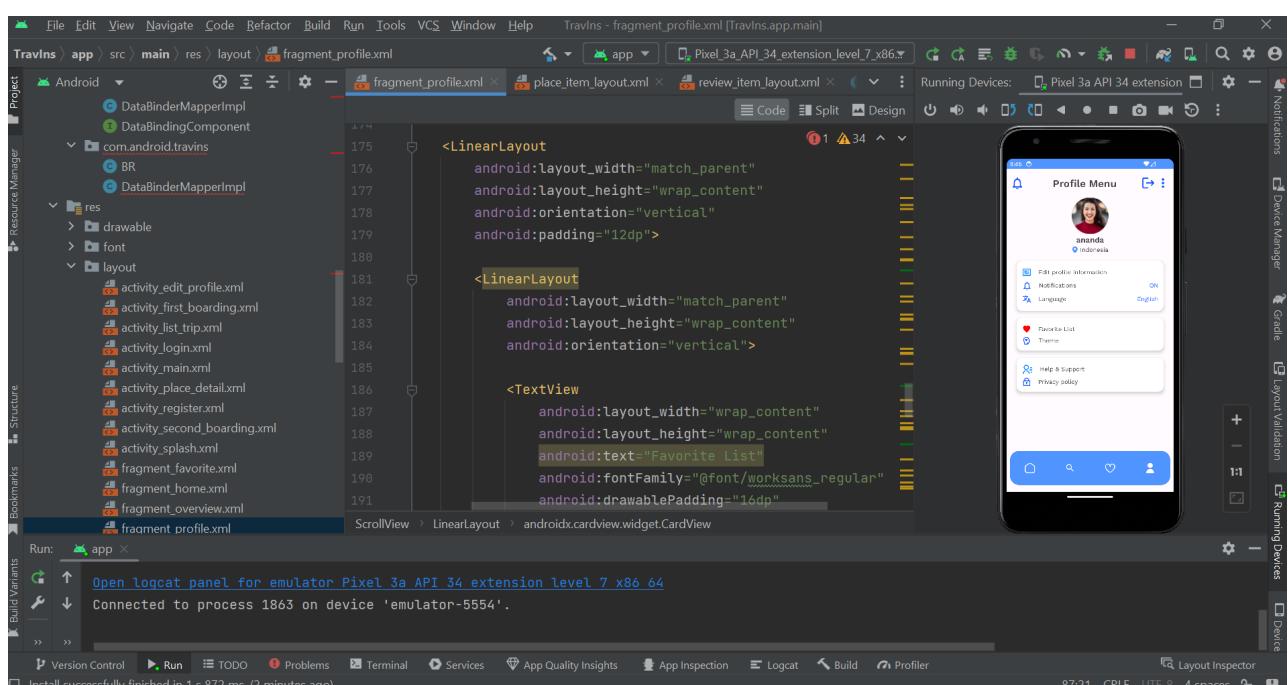
```

        app:layout_constraintLeft_toLeftOf="parent"/>
    </LinearLayout>
</LinearLayout>
</androidx.cardview.widget.CardView>

</LinearLayout>

</ScrollView>

```



Layout yang digunakan adalah sebuah `ScrollView` yang berisi beberapa elemen-elemen tata letak (layout) seperti `LinearLayout`, `ImageView`, `TextView`, dan `CardView`. Berikut adalah penjelasan mengenai penggunaan layout, elemen-elemen, dan atribut-atribut yang diterapkan dalam layout tersebut:

### Layout Utama:

- **ScrollView:** Digunakan untuk memungkinkan konten yang lebih besar dari layar dapat di-scroll.

### Elemen-elemen di dalam `ScrollView`:

#### 1. `LinearLayout`:

- Digunakan untuk mengatur tata letak vertikal untuk elemen-elemen di dalamnya.
- Atribut `orientation="vertical"`: Mengatur orientasi tata letak menjadi vertikal.

#### 2. `ConstraintLayout`:

- Digunakan untuk mengatur tata letak elemen-elemen dengan konstrain (batasan) tertentu.
- Memiliki `ImageView` dan `TextView` yang diatur menggunakan batasan-batasan tertentu.

### 3. ImageView:

- Digunakan untuk menampilkan gambar.
- Menggunakan atribut `src` untuk mengatur gambar yang ditampilkan.
  - Menggunakan atribut `layout\_width`, `layout\_height`, dan `app:layout\_constraint` untuk menentukan ukuran dan penempatan relatif terhadap elemen lain di dalam `ConstraintLayout`.

### 4. TextView:

- Digunakan untuk menampilkan teks.
- Digunakan untuk menunjukkan informasi seperti nama pengguna, lokasi, menu, dan opsi lainnya.
  - Menggunakan atribut untuk mengatur teks, gaya teks, ukuran teks, dan penempatan teks.

### 5. CardView:

- Mengapit beberapa elemen ke dalam kartu untuk memberikan efek visual kartu.
- Menggunakan atribut untuk mengatur sudut lengkungan, elevasi (bayangan), dan margin kartu.

Atribut-atribut yang Diterapkan dalam Elemen-elemen Layout:

- `android:layout\_width` dan `android:layout\_height`: Digunakan untuk menentukan lebar dan tinggi elemen.
- `android:orientation`: Mengatur orientasi tata letak (vertikal/horizontal) pada `LinearLayout`.
- `android:padding` dan `android:paddingHorizontal`: Memberikan ruang pola di sekitar elemen.
- `app:layout\_constraintLeft\_toLeftOf`, `app:layout\_constraintRight\_toRightOf`, `app:layout\_constraintTop\_toTopOf`, dst.: Menentukan batasan-batasan posisi elemen di dalam `ConstraintLayout`.
- `android:src`: Mengatur sumber gambar yang akan ditampilkan di `ImageView`.
- `android:text`: Mengatur teks yang akan ditampilkan di `TextView`.
- `android:fontFamily`: Mengatur jenis font teks yang akan digunakan.
- `android:drawableStart` dan `android:drawableTint`: Menambahkan gambar sebagai ikon sebelum teks dan mengatur warna gambar pada `TextView`.

Layout tersebut digunakan untuk menampilkan profil pengguna dan beberapa menu opsi terkait. Menggunakan `ScrollView` memungkinkan pengguna untuk menelusuri konten lebih besar secara vertikal. Elemen-elemen seperti `ImageView`, `TextView`, dan `CardView` digunakan untuk menampilkan gambar, teks, dan opsi menu dalam tata letak yang teratur dan dapat di-scroll. Atribut-atribut yang diterapkan membantu

dalam penataan dan penampilan visual dari elemen-elemen tersebut.

## Layout 2: [activity\_edit\_profile]

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.activities.EditProfileActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingHorizontal="24dp"
        android:orientation="vertical"
        android:gravity="center_horizontal">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:paddingVertical="16dp">
            <ImageView
                android:id="@+id/btnBack"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/ic_back"
                android:paddingStart="0dp"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Edit Profile"
                android:textColor="@color/black"
                android:textSize="22sp"
                android:textAlignment="center"
                android:paddingStart="0dp"
                android:fontFamily="@font/worksans_regular"/>
        </LinearLayout>

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_profile"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingBottom="8dp">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Full Name"
    android:paddingVertical="8dp"/>
<EditText
    android:id="@+id/edtTxtFullName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Type here"
    android:elevation="8dp"
    android:background="@drawable/bg_white_rounded"
    android:padding="10dp"/>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Username"
        android:paddingVertical="8dp"/>
    <EditText
        android:id="@+id/edtTxtUsername"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Type here"
        android:elevation="8dp"
        android:background="@drawable/bg_white_rounded"
        android:padding="10dp"/>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gender"
        android:paddingVertical="8dp"/>
    <EditText
        android:id="@+id/edtTxtGender"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Select here"
        android:elevation="8dp"
        android:background="@drawable/bg_white_rounded"
```

```
        android:padding="10dp"/>
    
```

```
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Birth of Date"
        android:paddingVertical="8dp"/>
    <EditText
        android:id="@+id/edtTxtBod"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="dd/mm/yyyy"
        android:elevation="8dp"
        android:background="@drawable/bg_white_rounded"
        android:padding="10dp"/>

```

```
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Country"
        android:paddingVertical="8dp"/>
    <EditText
        android:id="@+id/edtTxtCountry"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Type here"
        android:elevation="8dp"
        android:background="@drawable/bg_white_rounded"
        android:padding="10dp"/>

```

```
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Address"
```

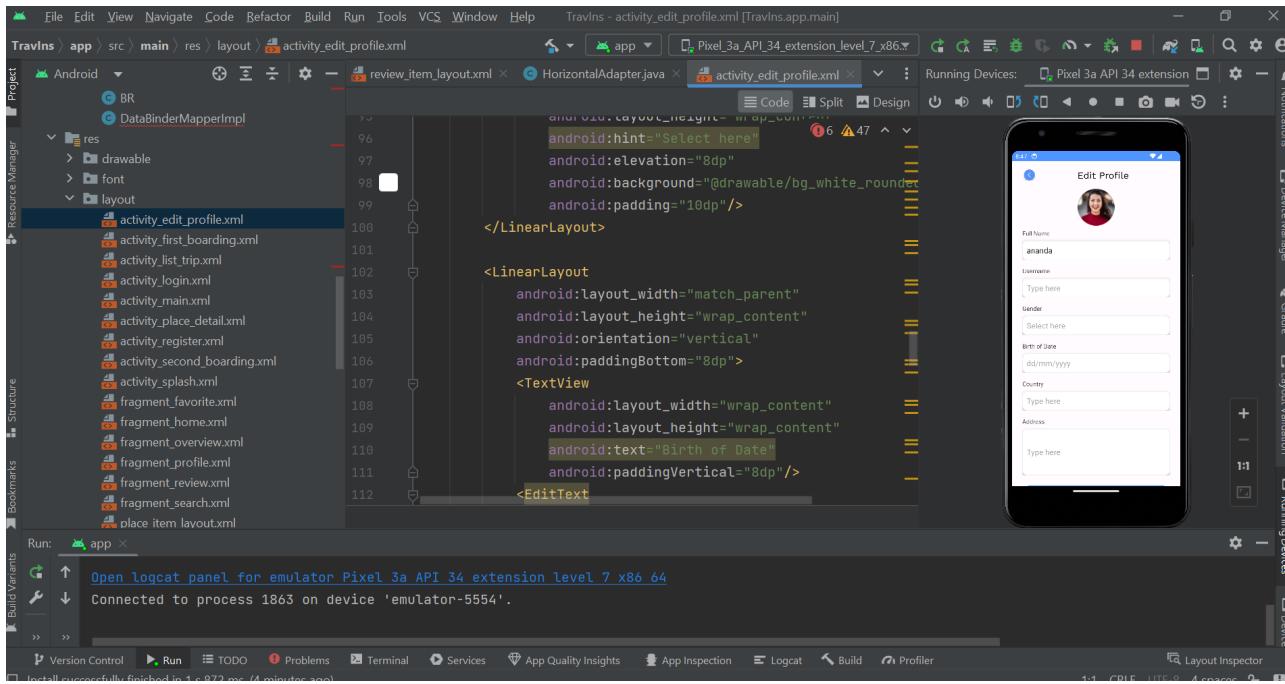
```

        android:paddingVertical="8dp"/>
<EditText
    android:id="@+id/edtTxtAddress"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Type here"
    android:elevation="8dp"
    android:minLines="4"
    android:background="@drawable/bg_white_rounded"
    android:padding="10dp"/>
</LinearLayout>

<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/btnSave"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/bg_blue_rounded"
    android:textAllCaps="false"
    android:text="Save Changes"
    android:textColor="@color/white"
    android:textSize="16sp"
    android:layout_marginVertical="16sp"
/>
</LinearLayout>

</ScrollView>

```



Layout yang disajikan merupakan tata letak untuk halaman pengeditan profil

pengguna. Berikut adalah penjelasan mengenai penggunaan layout, elemen-elemen yang ada di dalamnya, serta atribut-atribut yang diterapkan:

#### Layout Utama:

- ScrollView: Digunakan untuk memungkinkan konten yang lebih besar dari layar dapat di-scroll secara vertikal.

#### Elemen-elemen di dalam `ScrollView`:

##### 1. LinearLayout:

- Digunakan untuk mengatur tata letak vertikal untuk elemen-elemen di dalamnya.
- Atribut `orientation="vertical"`: Mengatur orientasi tata letak menjadi vertikal.
- Menggunakan `paddingHorizontal` untuk memberikan ruang di sisi kiri dan kanan layout.

##### 2. ImageView:

- Menampilkan gambar profil pengguna.

##### 3. TextView & EditText:

- Digunakan untuk menampilkan label dan input untuk informasi profil seperti Nama Lengkap, Username, Gender, Tanggal Lahir, Negara, dan Alamat.
- `TextView` menampilkan label informasi.
- `EditText` sebagai input untuk mengubah atau menambahkan informasi baru.

##### 4. AppCompatButton:

- Menampilkan tombol untuk menyimpan perubahan yang telah dilakukan pada profil.
- Menggunakan `background` untuk memberikan warna latar belakang tombol.
- Menggunakan `textColor` untuk mengatur warna teks pada tombol.

#### Atribut-atribut yang Diterapkan dalam Elemen Layout:

- `android:layout\_width` dan `android:layout\_height`: Mengatur lebar dan tinggi elemen.
- `android:paddingHorizontal` dan `android:paddingVertical`: Menentukan ruang di sekitar elemen.
- `android:src`: Menentukan sumber gambar yang akan ditampilkan pada `ImageView`.
- `android:text`, `android:textColor`, `android:textSize`, `android:textAlignment`, `android:fontFamily`: Mengatur teks yang akan ditampilkan pada `TextView` sesuai dengan kebutuhan desain dan informasi yang disajikan.
- `android:hint`, `android:background`, `android:elevation`, `android:minLines`: Mengatur teks petunjuk (hint), latar belakang elemen, elevasi, dan jumlah baris minimum pada `EditText`.
- `android:background`: Memberikan latar belakang berbentuk drawable (di sini,

mungkin diterapkan bentuk bulat untuk kotak input).

- `android:textAllCaps`: Mengontrol apakah teks di tombol akan ditampilkan dalam huruf kapital atau tidak.

Layout tersebut disusun untuk memberikan antarmuka pengguna yang memungkinkan pengguna untuk mengedit dan menyimpan informasi profil mereka. Penggunaan `ScrollView` memungkinkan pengguna untuk melakukan scrolling jika konten melebihi ukuran layar. Elemen-elemen seperti `TextView`, `EditText`, `ImageView`, dan `Button` digunakan untuk menampilkan informasi profil dan tombol untuk menyimpan perubahan. Atribut-atribut yang diterapkan membantu dalam penataan dan penampilan visual dari elemen-elemen tersebut.

# Implementasi Kode Program Aplikasi

## Kode Program Activity : EditProfileActivity

```
package com.android.travins.ui.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

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

import com.android.travins.R;
import com.android.travins.databinding.ActivityEditProfileBinding;
import com.android.travins.databinding.ActivityLoginBinding;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

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

public class EditProfileActivity extends AppCompatActivity {

    private ActivityEditProfileBinding binding;

    FirebaseFirestore db;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityEditProfileBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        db = FirebaseFirestore.getInstance();

        String id = getIntent().getStringExtra("id");
        String name = getIntent().getStringExtra("name");
        String username = getIntent().getStringExtra("username");
        String gender = getIntent().getStringExtra("gender");
        String email = getIntent().getStringExtra("email");
        String country = getIntent().getStringExtra("country");
        String bod = getIntent().getStringExtra("bod");
        String address = getIntent().getStringExtra("address");

        binding.edtTxtFullName.setText(name);
        binding.edtTxtAddress.setText(address);
        binding.edtTxtBod.setText(bod);
        binding.edtTxtCountry.setText(country);
    }
}
```

```

binding.edtTxtGender.setText(gender);
binding.edtTxtUsername.setText(username);
binding.edtTxtFullName.setText(name);

binding.btnBack.setOnClickListener(v -> {
    finish();
});

binding.btnSave.setOnClickListener(v -> {
    CollectionReference usersCollection =
db.collection("users");
    assert id != null;
    DocumentReference userDocument =
usersCollection.document(id);
    Map<String, Object> updates = new HashMap<>();
    updates.put("fullName",
binding.edtTxtFullName.getText().toString());
    updates.put("username",
binding.edtTxtUsername.getText().toString());
    updates.put("address",
binding.edtTxtAddress.getText().toString());
    updates.put("country",
binding.edtTxtCountry.getText().toString());
    updates.put("gender",
binding.edtTxtGender.getText().toString());
    updates.put("bod", binding.edtTxtBod.getText().toString());
    userDocument.update(updates)
        .addOnSuccessListener(aVoid -> {
            Toast.makeText(EditProfileActivity.this,"Update
successful!",Toast.LENGTH_SHORT).show();
            finish();
        })
        .addOnFailureListener(e -> {
            Toast.makeText(EditProfileActivity.this,"Update
failed!",Toast.LENGTH_SHORT).show();
            // Update failed
            // Handle the error
        });
    });
}

}

```

[Letakkan kode program Activity/Fragment Keseluruhan di sini]

### Kode Program Fragment: ProfileFragment

```
package com.android.travins.ui.fragments;
```

```
import static android.content.ContentValues.TAG;

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

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

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.android.travins.R;
import com.android.travins.data.models.User;
import com.android.travins.ui.activities.EditProfileActivity;
import com.android.travins.ui.activities.ListTripActivity;
import com.android.travins.ui.activities.LoginActivity;
import com.android.travins.ui.activities.RegisterActivity;
import com.google.android.gms.tasks.OnCompleteListener;
```

```
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firebaseio.QueryDocumentSnapshot;
import com.google.firebaseio.firebaseio.QuerySnapshot;

import java.util.Map;
import java.util.Objects;

public class ProfileFragment extends Fragment {

    protected FirebaseFirestore db;
    protected FirebaseAuth auth;
    protected User user = new User();

    TextView fullName;
    TextView goToEditProfile;

    ImageView btnLogout;;

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

```
@Override

    public View onCreateView(LayoutInflater inflater, ViewGroup
container,

                           Bundle savedInstanceState) {

    // Inflate the layout for this fragment

    return inflater.inflate(R.layout.fragment_profile,
container, false);

}

@Override

    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {

    super.onViewCreated(view, savedInstanceState);

    db = FirebaseFirestore.getInstance();

    auth = FirebaseAuth.getInstance();

    fullName = view.findViewById(R.id.profileFullName);

    goToEditProfile = view.findViewById(R.id.btnEditProfile);

    btnLogout = view.findViewById(R.id.btnExit);

    btnLogout.setOnClickListener(v -> {

        auth.signOut();

        startActivity(new Intent(view.getContext(),
LoginActivity.class));

        this.getActivity().finish();

    });

}
```

```
        goToEditProfile.setOnClickListener(v -> {

            Intent mainIntent = new Intent(this.getActivity(),
EditProfileActivity.class);

            mainIntent.putExtra("id",user.getId());

            mainIntent.putExtra("name",user.getFullName());

            mainIntent.putExtra("username",user.getUsername());

            mainIntent.putExtra("email",user.getEmail());

            mainIntent.putExtra("gender",user.getGender());

            mainIntent.putExtra("country",user.getCountry());

            mainIntent.putExtra("bod",user.getBod());

            mainIntent.putExtra("address",user.getAddress());

            startActivity(mainIntent);

        });

        db.collection("users").whereEqualTo("email",
Objects.requireNonNull(auth.getCurrentUser()).getEmail())

        .get()

        .addOnCompleteListener(new

OnCompleteListener<QuerySnapshot>() {

    @Override

        public void onComplete(@NonNull

Task<QuerySnapshot> task) {

            if (task.isSuccessful()) {

                for (QueryDocumentSnapshot document :

task.getResult()) {

                    Map<String, Object> data =
document.getData();

```

```

fullName.setText(data.get("fullName").toString());

        user = new
User(document.getId(), (String) data.get("fullName"),
                (String)
data.get("username"), (String) data.get("email"),
                (String)
data.get("password"), (String) data.get("gender"),
                (String)
data.get("country"), (String) data.get("bod"),
                (String)
data.get("address"));

    }

} else {
        Log.w("ERROR", "Error getting
documents.", task.getException());
}

}

});;

}

}

```

Kode program yang diberikan mengilustrasikan dua bagian utama dari aplikasi Android: `EditProfileActivity` dan `ProfileFragment`.

#### EditProfileActivity:

- `EditProfileActivity` merupakan sebuah `Activity` yang bertanggung jawab untuk menampilkan layar pengeditan profil pengguna.
- Pada `onCreate()`:

- Mengambil data profil yang diterima dari intent. Data tersebut akan ditampilkan ke dalam elemen-elemen UI seperti `EditText`.
- Mendefinisikan perilaku tombol "Save" untuk menyimpan perubahan pada profil pengguna ke database Firebase. Ketika tombol tersebut ditekan, data yang dimasukkan oleh pengguna akan diambil dari elemen-elemen UI, diperbarui dalam koleksi pengguna Firebase, dan pesan toast akan ditampilkan sesuai dengan keberhasilan atau kegagalan penyimpanan data.

ProfileFragment:

- `ProfileFragment` adalah `Fragment` yang menampilkan profil pengguna.
- Pada `onViewCreated()`:
  - Mendapatkan data pengguna dari Firebase berdasarkan email yang sedang masuk.
  - Menampilkan data pengguna di dalam elemen UI, seperti `TextView`.
  - Mengatur perilaku tombol "Edit Profile" yang akan membuka `EditProfileActivity` untuk memungkinkan pengguna mengubah data profil mereka.

Kedua bagian kode ini berinteraksi dengan database Firebase Firestore untuk mengambil dan menyimpan data pengguna. Mereka juga bertanggung jawab untuk menampilkan dan mengelola UI sesuai dengan data yang diterima dari database.

Kedua komponen ini bekerja sama untuk memberikan pengalaman yang mulus bagi pengguna dalam mengelola dan mengubah informasi profil mereka melalui antarmuka yang diberikan oleh `EditProfileActivity` dan `ProfileFragment` .

Untuk lebih jelasnya terdapat rincian penjelasan pada setiap kodennya :

EditProfileActivity :

```
// Inisialisasi binding untuk layout activity
binding = ActivityEditProfileBinding.inflate(getApplicationContext());
setContentView(binding.getRoot());
```

```
// Mendapatkan instance dari Firebase Firestore
```

```
db = FirebaseFirestore.getInstance();
```

```
// Mendapatkan data dari intent
```

```
String id = getIntent().getStringExtra("id");
String name = getIntent().getStringExtra("name");
// ... (mengambil data lainnya)

// Mengisi elemen-elemen UI dengan data yang diterima
binding.edtTxtFullName.setText(name);
// ... (mengisi elemen lainnya)

// Ketika tombol 'Save' ditekan, menyimpan perubahan ke Firestore
binding.btnSave.setOnClickListener(v -> {
    // Mendapatkan referensi ke koleksi 'users' di Firestore
    CollectionReference usersCollection = db.collection("users");
    // Mendapatkan dokumen pengguna yang sesuai dengan 'id'
    DocumentReference userDocument = usersCollection.document(id);

    // Mengupdate data pengguna sesuai dengan yang diinput oleh pengguna di UI
    Map<String, Object> updates = new HashMap<>();
    updates.put("fullName", binding.edtTxtFullName.getText().toString());
    // ... (update data lainnya)

    // Melakukan update di Firestore
    userDocument.update(updates)
        .addOnSuccessListener(aVoid -> {
            Toast.makeText(EditProfileActivity.this, "Update successful!",
                Toast.LENGTH_SHORT).show();
        });
});
```

```
        finish();

    })

    .addOnFailureListener(e -> {
        Toast.makeText(EditProfileActivity.this, "Update failed!", Toast.LENGTH_SHORT).show();
        // Update gagal, handle error
    });
});
```

Profile Fragment :

```
// Mendapatkan instance Firebase Firestore dan Auth
db = FirebaseFirestore.getInstance();
auth = FirebaseAuth.getInstance();
```

```
// Ketika tombol 'Logout' ditekan, keluar dari akun
btnLogout.setOnClickListener(v -> {
    auth.signOut();
    startActivity(new Intent(view.getContext(), LoginActivity.class));
    this.getActivity().finish();
});
```

```
// Ketika tombol 'Edit Profile' ditekan, membuka EditProfileActivity
goToEditProfile.setOnClickListener(v -> {
    Intent mainIntent = new Intent(this.getActivity(), EditProfileActivity.class);
    // Menambahkan data pengguna yang diperlukan untuk diedit
    mainIntent.putExtra("id", user.getId());
```

```
mainIntent.putExtra("name", user.getFullName());

// ... (menambahkan data lainnya)

startActivity(mainIntent);

});

// Mengambil data pengguna dari Firestore berdasarkan email saat ini

db.collection("users").whereEqualTo("email",
Objects.requireNonNull(auth.getCurrentUser()).getEmail())

.get()

.addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {

@Override

public void onComplete(@NonNull Task<QuerySnapshot> task) {

if (task.isSuccessful()) {

for (QueryDocumentSnapshot document : task.getResult()) {

// Menyimpan data pengguna ke objek User

user = new User(

document.getId(),

(String) document.get("fullName"),

// ... (menyimpan data lainnya)

);

}

} else {

Log.w("ERROR", "Error getting documents.", task.getException());

}

}

});
```

# Implementasi Fragment

## Kode Program Fragment: ProfileFragment

```
package com.android.travins.ui.fragments;

import static android.content.ContentValues.TAG;

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

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

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.android.travins.R;
import com.android.travins.data.models.User;
import com.android.travins.ui.activities.EditProfileActivity;
import com.android.travins.ui.activities.ListTripActivity;
```

```
import com.android.travins.ui.activities.LoginActivity;
import com.android.travins.ui.activities.RegisterActivity;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.Map;
import java.util.Objects;

public class ProfileFragment extends Fragment {

    protected FirebaseFirestore db;
    protected FirebaseAuth auth;
    protected User user = new User();

    TextView fullName;
    TextView goToEditProfile;

    ImageView btnLogout;;
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);

    }

@Override

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

    // Inflate the layout for this fragment

    return inflater.inflate(R.layout.fragment_profile,
                           container, false);
}

@Override

    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {

    super.onViewCreated(view, savedInstanceState);

    db = FirebaseFirestore.getInstance();

    auth = FirebaseAuth.getInstance();

    fullName = view.findViewById(R.id.profileFullName);

    goToEditProfile = view.findViewById(R.id.btnEditProfile);

    btnLogout = view.findViewById(R.id.btnExit);

    btnLogout.setOnClickListener(v -> {

        auth.signOut();

        startActivity(new Intent(view.getContext(),
                               LoginActivity.class));

        this.getActivity().finish();
    });
}
```

```
});

goToEditProfile.setOnClickListener(v -> {

    Intent mainIntent = new Intent(this.getActivity(),
EditProfileActivity.class);

    mainIntent.putExtra("id",user.getId());

    mainIntent.putExtra("name",user.getFullName());

    mainIntent.putExtra("username",user.getUsername());

    mainIntent.putExtra("email",user.getEmail());

    mainIntent.putExtra("gender",user.getGender());

    mainIntent.putExtra("country",user.getCountry());

    mainIntent.putExtra("bod",user.getBod());

    mainIntent.putExtra("address",user.getAddress());

    startActivity(mainIntent);

});

db.collection("users").whereEqualTo("email",
Objects.requireNonNull(auth.getCurrentUser()).getEmail())

.get()

.addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {

    @Override

        public void onComplete(@NonNull
Task<QuerySnapshot> task) {

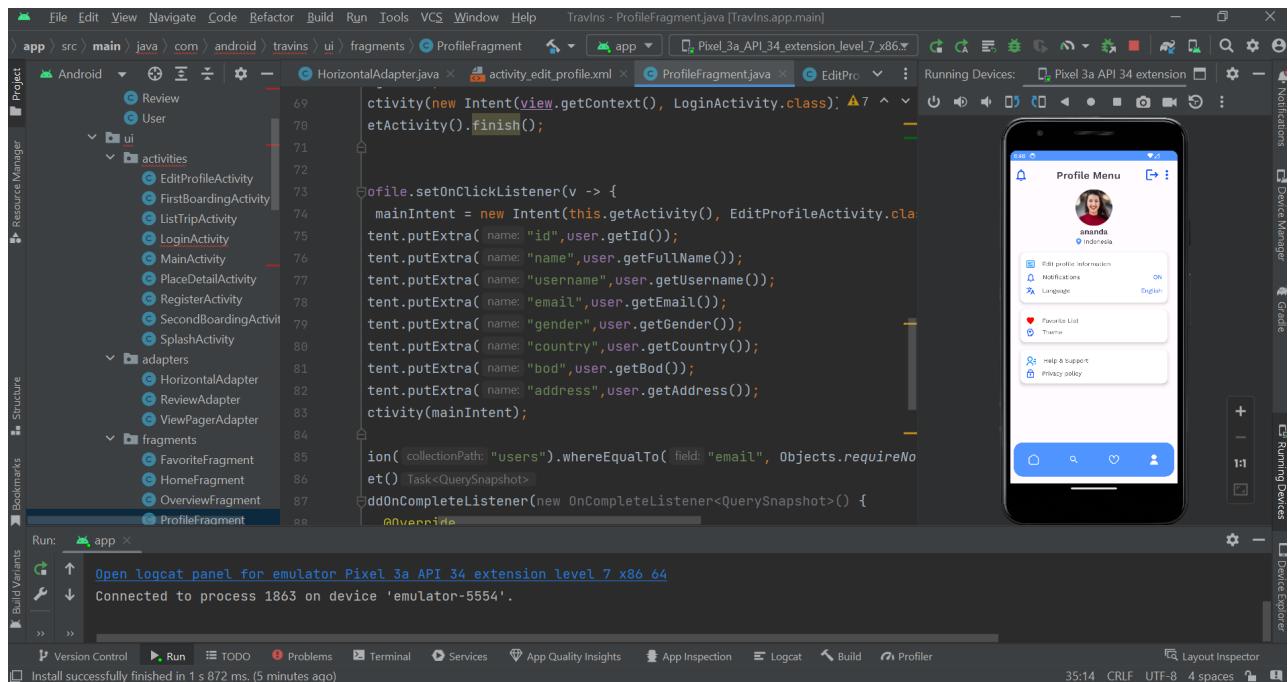
            if (task.isSuccessful()) {

                for (QueryDocumentSnapshot document :
task.getResult()) {

                    Map<String, Object> data =
document.getData();

```

```
fullName.setText(data.get("fullName").toString());  
  
        user = new  
User(document.getId(), (String) data.get("fullName"),  
                           (String)  
data.get("username"), (String) data.get("email"),  
                           (String)  
data.get("password"), (String) data.get("gender"),  
                           (String)  
data.get("country"), (String) data.get("bod"),  
                           (String)  
data.get("address"));  
  
    }  
  
} else {  
  
    Log.w("ERROR", "Error getting  
documents.", task.getException());  
  
}  
  
}  
  
});  
  
}  
  
}
```



Kode ini adalah sebuah `Fragment` yang menampilkan informasi profil pengguna. Di sini, saat `ProfileFragment` ditampilkan, data profil pengguna diambil dari Firestore dan ditampilkan di UI Fragment ini.

Kode untuk Membuka EditProfileActivity dari ProfileFragment:

```
```java
goToEditProfile.setOnClickListener(v -> {

    Intent mainIntent = new Intent(this.getActivity(), EditProfileActivity.class);

    // Menambahkan data pengguna yang diperlukan untuk diedit
    mainIntent.putExtra("id", user.getId());
    mainIntent.putExtra("name", user.getFullName());
    // ... (menambahkan data lainnya)
    startActivity(mainIntent);
});```

```

Ketika `btnEditProfile` di klik, ia akan membuat `Intent` baru untuk membuka `EditProfileActivity`. Data pengguna yang diperlukan untuk diedit diambil dari `user` yang telah diisi dengan informasi pengguna yang diambil dari Firestore sebelumnya. Intent ini akan membuka `EditProfileActivity` dengan membawa informasi pengguna yang diperlukan untuk diedit.

Activity/XML yang Memuat/Menjalankan Fragment Ini:

Untuk menampilkan `ProfileFragment` di dalam sebuah Activity, Anda dapat menggunakan kode seperti ini di Activity Anda:

```
```java
// Di dalam Activity

FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
transaction.replace(R.id.fragment_container, new ProfileFragment());
transaction.commit();

````
```

Di sini, `R.id.fragment\_container` adalah ID dari container di layout Activity yang akan menampung Fragment ini.

Komunikasi Antar Activity dan Fragment:

Untuk membuka `ProfileFragment` dari sebuah `Activity`, kita bisa menggunakan contoh berikut:

```
```java
// Membuat FragmentManager

FragmentManager fragmentManager = getSupportFragmentManager();
// Memulai transaksi fragment

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
// Mengganti konten container dengan ProfileFragment

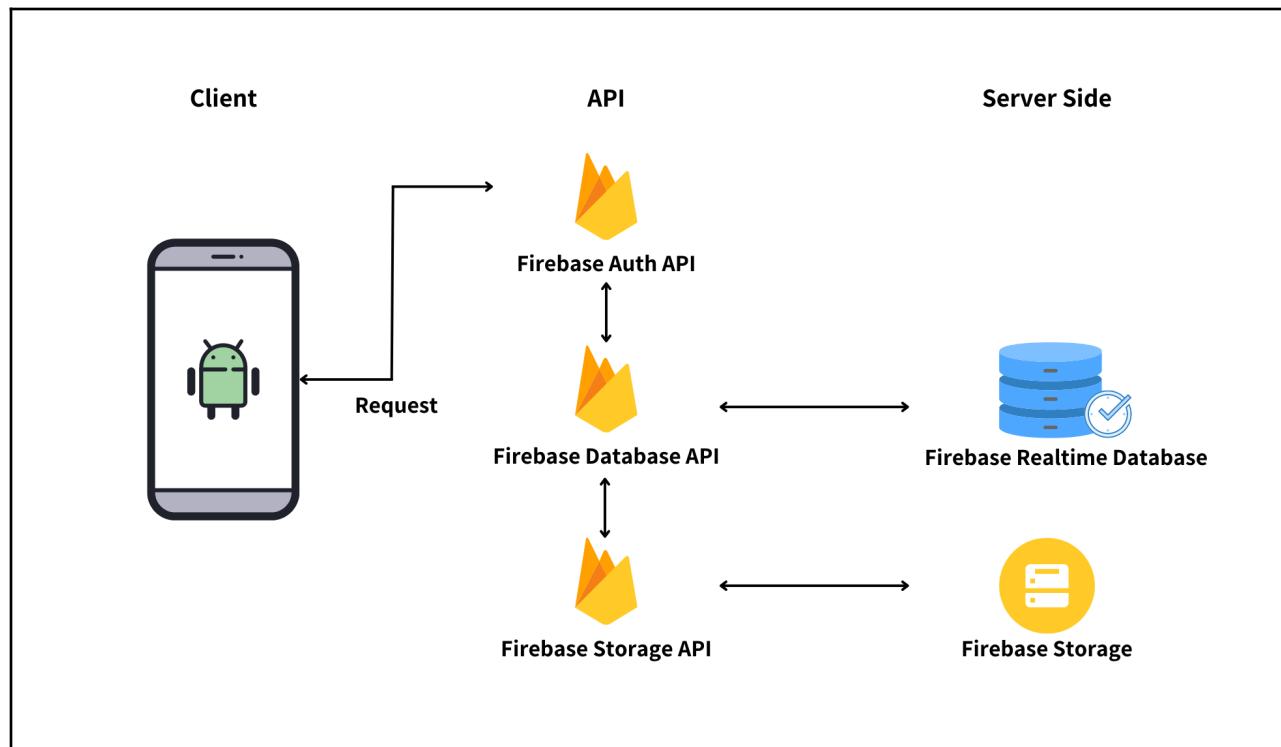
ProfileFragment profileFragment = new ProfileFragment();
fragmentTransaction.replace(R.id.fragment_container, profileFragment);
fragmentTransaction.commit();

````
```

Di sini, `ProfileFragment` dimasukkan ke dalam layout menggunakan `FragmentManager`. Proses ini dapat dilakukan di dalam sebuah `Activity`. Yang perlu diperhatikan adalah penggunaan `fragment\_container`, yang merupakan sebuah container di dalam layout Activity yang akan menampung Fragment.

Itulah bagaimana komunikasi antara Activity-Fragment dilakukan di Android menggunakan `FragmentManager` dan `Intent`.

## Implementasi Komunikasi Data



Gambar tersebut memberikan gambaran umum tentang cara kerja API. API adalah alat yang penting yang memungkinkan aplikasi berkomunikasi satu sama lain dan mengakses data dan layanan dari server. Dalam diagram tersebut, ada dua sisi utama API: sisi klien dan sisi server. Sisi klien adalah aplikasi yang menggunakan API. Aplikasi ini dapat berupa aplikasi web, aplikasi seluler, atau aplikasi desktop. Aplikasi klien biasanya menggunakan API untuk mengakses data atau layanan yang disediakan oleh server. Dalam diagram tersebut, sisi klien diwakili oleh ikon telepon Android. Telepon Android dapat menggunakan API Firebase untuk mengakses data dan layanan dari server Firebase.

Penjelasan tentang masing-masing API :

**Firebase Auth API** : Firebase Auth API menyediakan berbagai fitur untuk mengelola autentikasi dan otorisasi pengguna, termasuk:

1. Registrasi pengguna
2. Masuk dengan akun Google atau email
3. Verifikasi email
4. Kelola akun pengguna
5. Firebase Database API

Firebase Database API adalah database real-time yang memungkinkan aplikasi klien untuk menyimpan dan mengambil data secara real-time. Data disimpan dalam format JSON dan dapat diakses dari aplikasi klien menggunakan API.

#### **Firebase Storage API**

Firebase Storage API adalah layanan penyimpanan file yang memungkinkan aplikasi klien untuk menyimpan file di cloud. File dapat diakses dari aplikasi klien menggunakan API.

# **Use Case 3: Register & Search Places**

**Muhammad Althaaf Fadhiilah**

NIM 215150400111053



## **Deskripsi Fungsional**

### **Login dan Register**

Register dilakukan agar pengguna dapat menggunakan fitur yang ada dalam aplikasi. User melakukan registrasi dengan mengisi form seperti nama lengkap, email, password, dan konfirmasi password. Setelah pengguna mengisi form dan menekan tombol “Register” maka program akan mengautentikasi metode masuk aplikasi menggunakan Firebase Authentication dan data user akan tersimpan pada Firebase Database. Selain menggunakan email, user juga dapat melakukan registrasi/login menggunakan akun Google dengan menekan tombol “Google” pada halaman Login maupun Register.

Ketika user melakukan login maka proses autentikasi akan dilakukan, ketika autentikasi berhasil user akan diarahkan menuju halaman home. Tetapi ketika user menginputkan email yang belum terdaftar atau menginput password yang kurang sesuai maka proses autentikasi akan gagal dan menerima pesan

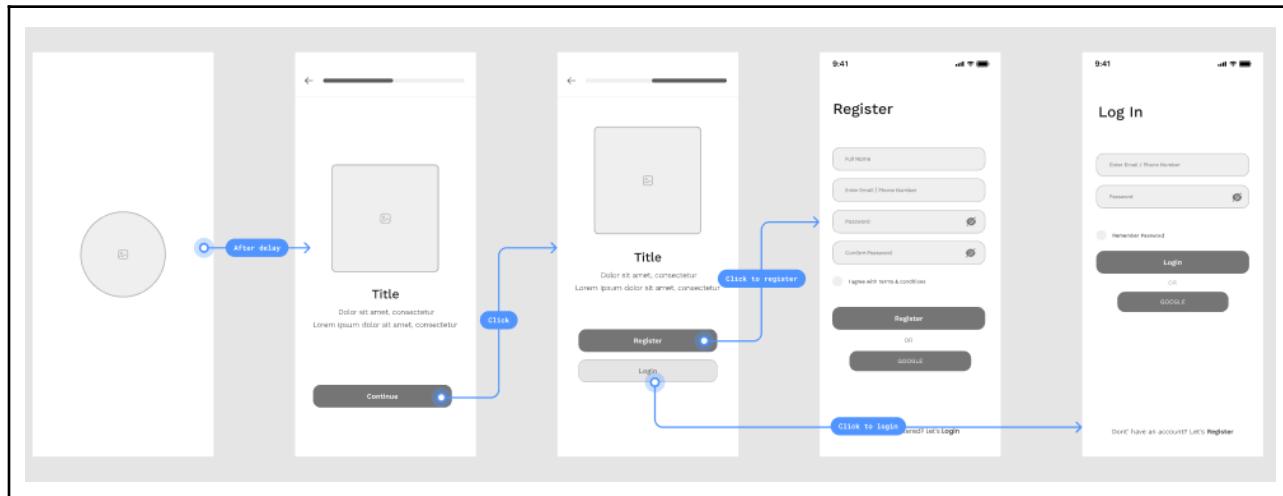
### **Search Destination**

Search Destination merupakan sebuah fitur yang dapat diakses user untuk mencari

destinasi yang diinginkan. Fitur ini terhubung pada Firestore Database yang menyimpan semua informasi terkait destinasi yang ada. User dapat mencari destinasi dengan menginputkan nama destinasi pada search bar yang berada pada atas halaman.

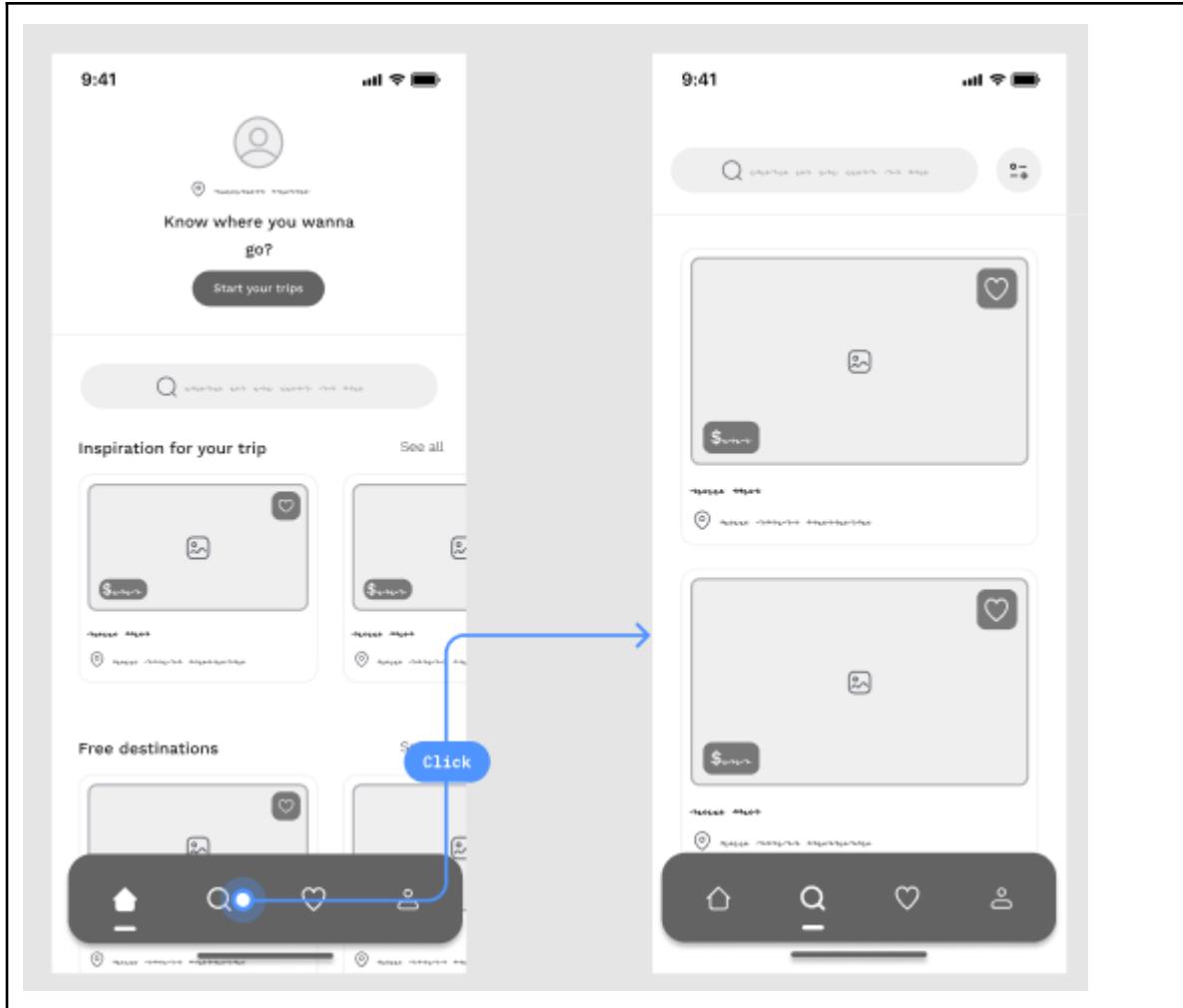
# Rancangan Screenflow Register

## 1. Login dan Register



Ketika user pertama kali menggunakan aplikasi TravIns, user akan mendapatkan tampilan halaman Splash Screen dan Onboarding. Pada halaman Onboarding, user akan memilih “Login” ketika sudah memiliki akun dan “Register” ketika user belum memiliki akun. Pada halaman Register user diminta mengisi form seperti nama lengkap, email, password, dan konfirmasi password. Kemudian ketika semua form telah terisi, user menekan tombol “Register” dan akan langsung diarahkan ke halaman Login. Setelah itu user menginput email dan password yang telah didaftarkan pada halaman Register sebelumnya dan menekan tombol “Login” ketika email dan password yang diberikan telah dirasa benar. Setelah itu user diarahkan menuju halaman home pada aplikasi TravIns

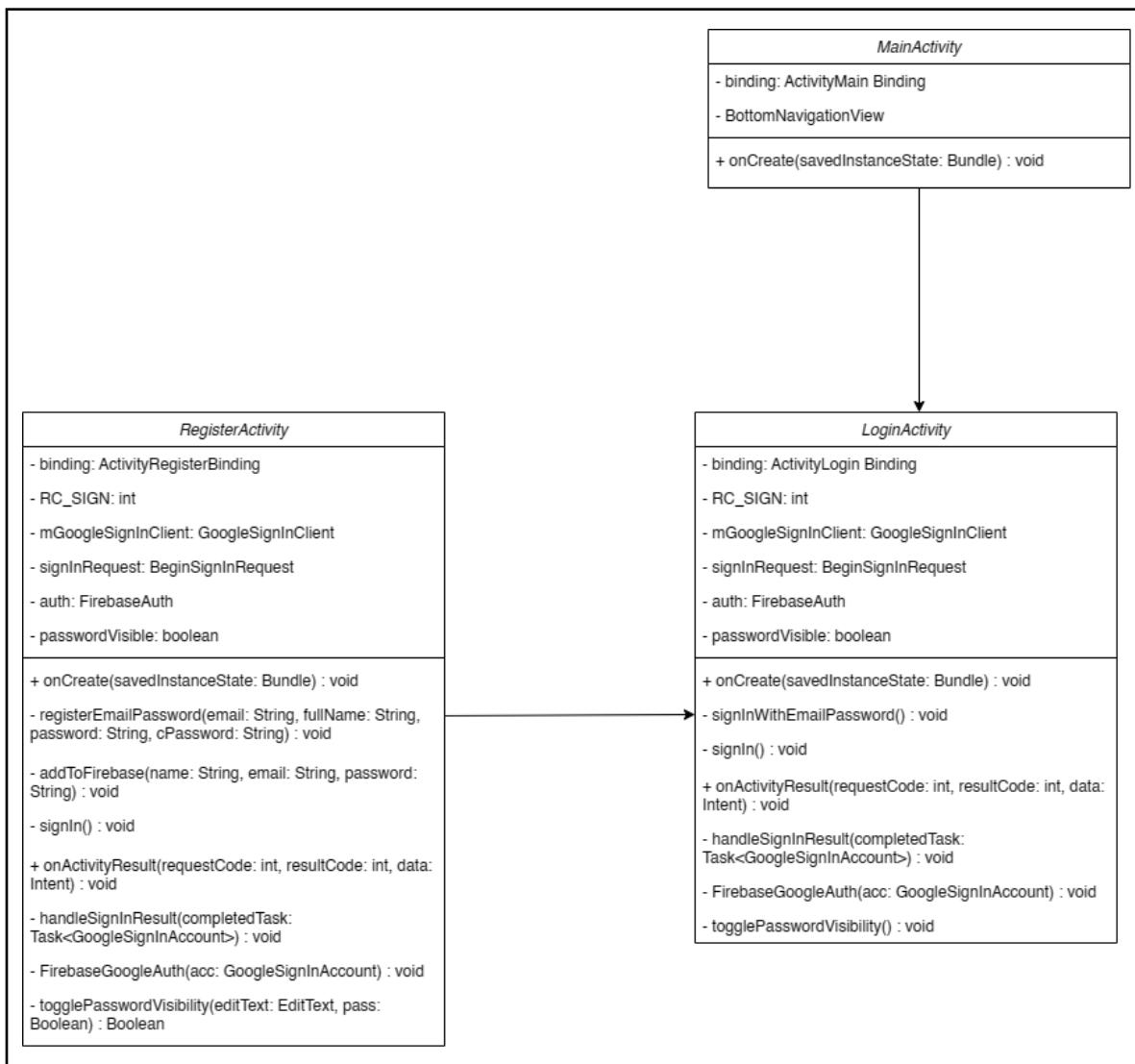
## 2. Search Destination



Fitur Search Destination dapat diakses melalui navigation bar yang terletak pada bagian bawah halaman. Dengan menekan icon “Loop” maka akan langsung mengarah ke halaman Search Destination. Pada halaman Search Destination pengguna dapat melakukan pencarian destinasi dengan menginputkan kata kunci pada search bar yang berada pada bagian atas halaman

# Class Diagram Register

## 1. Class Diagram Login dan Register



### 1. MainActivity

Pada class ini terdapat 2 objek yaitu AcitivtyMain Binding bernama binding dan BottomNavigationView untuk menampilkan menu navigasi pada bagian bawah halaman. Kemudia method yang digunakan yaitu onCreate.

### 2. RegisterActivity

### **Atribut:**

- binding: Bertipe ActivityRegisterBinding, digunakan untuk mengakses elemen UI yang terhubung dengan layout aktivitas registrasi.
- RC\_SIGN\_IN: Konstanta untuk kode permintaan aktivitas Google Sign-In.
- mGoogleSignInClient: Bertipe GoogleSignInClient, digunakan untuk mengelola proses Google Sign-In.
- signInRequest: Bertipe BeginSignInRequest, berisi konfigurasi untuk memulai proses sign-in.
- auth: Bertipe FirebaseAuth, digunakan untuk otentikasi pengguna menggunakan Firebase.
- db: Bertipe FirebaseFirestore, digunakan untuk berinteraksi dengan Firebase Firestore, basis data cloud Firebase.
- passwordVisibleC: Menyimpan status visibilitas password untuk konfirmasi password, digunakan untuk menentukan apakah konfirmasi password saat ini terlihat atau tidak.
- passwordVisible: Menyimpan status visibilitas password, digunakan untuk menentukan apakah password saat ini terlihat atau tidak.

### **Metode:**

- onCreate(savedInstanceState: Bundle): Metode utama yang dipanggil saat aktivitas dibuat. Inisialisasi variabel, objek, dan menangani logika saat aktivitas dibuat.
- registerEmailPassword(email: String, fullName: String, password: String, cPassword: String): Metode untuk mendaftarkan pengguna menggunakan email dan password.
- addToFirebase(name: String, email: String, password: String): Metode untuk menambahkan data pengguna ke Firebase Firestore setelah pendaftaran berhasil.
- signIn(): Metode untuk memulai proses Google Sign-In.
- onActivityResult(requestCode: int, resultCode: int, data: Intent): Metode yang dipanggil setelah aktivitas lain selesai, digunakan untuk menangani hasil dari aktivitas Google Sign-In.
- handleSignInResult(completedTask: Task<GoogleSignInAccount>): Metode untuk menangani hasil dari proses Google Sign-In.
- FirebaseGoogleAuth(acc: GoogleSignInAccount): Metode untuk mengotentikasi pengguna Firebase menggunakan informasi akun Google.
- togglePasswordVisibility(editText: EditText, pass: Boolean): Metode untuk mengubah visibilitas password pada input teks dan mengembalikan status visibilitas yang baru.

### 3. LoginActivity

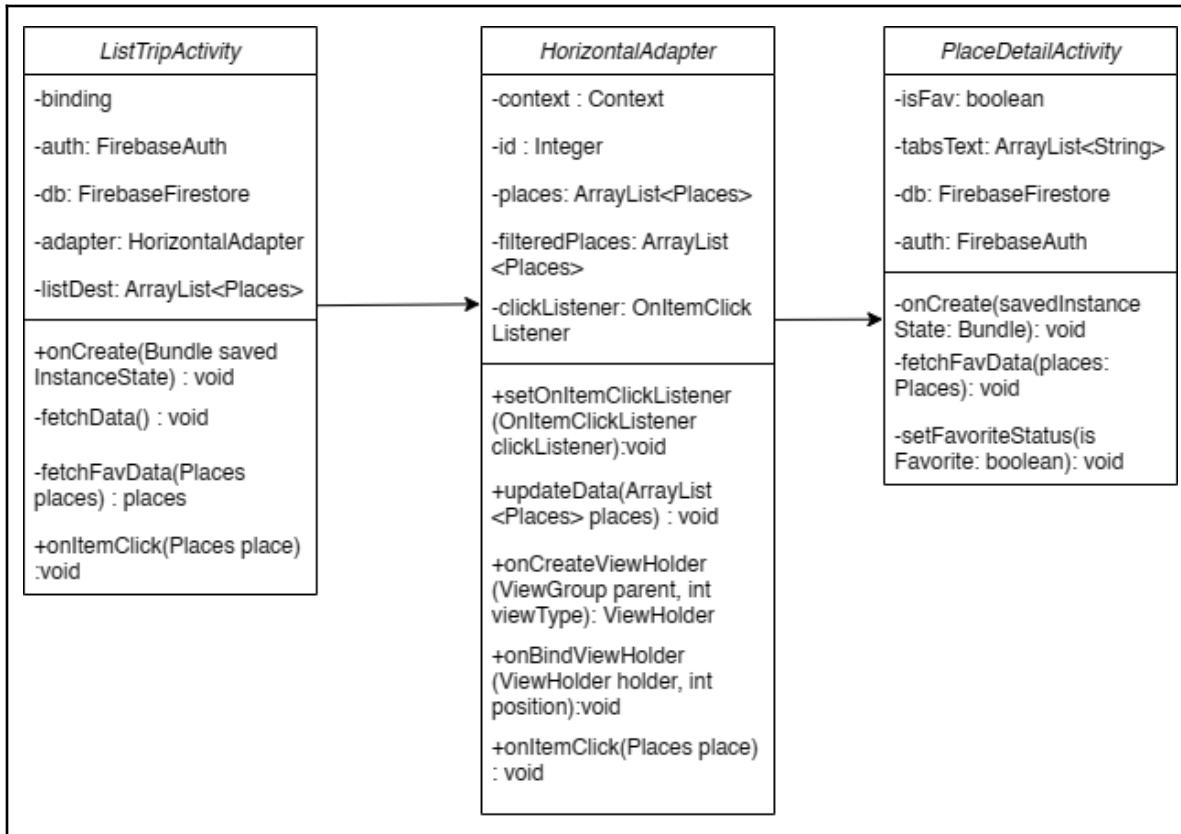
#### Atribut:

- binding: Bertipe ActivityLoginBinding, digunakan untuk mengakses elemen UI yang terhubung dengan layout aktivitas login.
- RC\_SIGN\_IN: Konstanta untuk kode permintaan aktivitas Google Sign-In.
- mGoogleSignInClient: Bertipe GoogleSignInClient, digunakan untuk mengelola proses Google Sign-In.
- signInRequest: Bertipe BeginSignInRequest, berisi konfigurasi untuk memulai proses sign-in.
- auth: Bertipe FirebaseAuth, digunakan untuk otentikasi pengguna menggunakan Firebase.
- passwordVisible: Menyimpan status visibilitas password, digunakan untuk menentukan apakah password saat ini terlihat atau tidak.

#### Metode:

- onCreate(savedInstanceState: Bundle): Metode utama yang dipanggil saat aktivitas dibuat. Inisialisasi variabel, objek, dan menangani logika saat aktivitas dibuat.
- signInWithEmailAndPassword(): Metode untuk melakukan otentikasi menggunakan email dan password.
- signIn(): Metode untuk memulai proses Google Sign-In.
- onActivityResult(requestCode: int, resultCode: int, data: Intent): Metode yang dipanggil setelah aktivitas lain selesai, digunakan untuk menangani hasil dari aktivitas Google Sign-In.
- handleSignInResult(completedTask: Task<GoogleSignInAccount>): Metode untuk menangani hasil dari proses Google Sign-In.
- FirebaseAuth.signInWithGoogle(acc: GoogleSignInAccount): Metode untuk mengotentikasi pengguna Firebase menggunakan informasi akun Google.
- togglePasswordVisibility(): Metode untuk mengubah visibilitas password pada input teks.

## 2. Class Diagram Search Destination



### 1. ListTripActivity

Memiliki beberapa atribut seperti binding, auth, db, adapter, listDest. Dan juga beberapa method yang digunakan yaitu OnCreate, fetchData() untuk mengakses data, onItemClick untuk menjalankan activity baru dengan Intent.

### 2. HorizontalAdapter

Pada adapter ini ada beberapa atribut seperti context, id dengan tipe integer, ArrayList<Places> dengan nama places, kemudian ada ArrayList<Places> lagi namun yang ini digunakan untuk list yang sudah dilakukan filter terhadap itemnya, dan yang terakhir ada clickListener untuk menjalankan activity baru.

### 3. PlaceDetailActivity

Activity ini bertujuan untuk mengambil data apakah destinasi tersebut

favorit atau bukan. Adapun beberapa atributnya seperti isFav bertipe boolean, ArrayList<String> bernama tabsText, db untuk mengakses database. Kemudian ada method yang digunakan yaitu OnCreate, fetchFavData untuk mengakses data destinasi favorit, dan setFavoriteStatus untuk mengubah status favorit dari destinasi.

# Android Manifest

Fungsionalitas Login dan Register memerlukan permission android.permission.INTERNET yang digunakan untuk mengakses internet agar dapat terintegrasi dengan firebase. Permission tersebut dideklarasikan dalam file manifest berikut:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Fungsionalitas ini menggunakan sebuah Activity dengan nama LoginActivity.java yang dideklarasikan dengan elemen berikut dalam file AndroidManifest.xml:

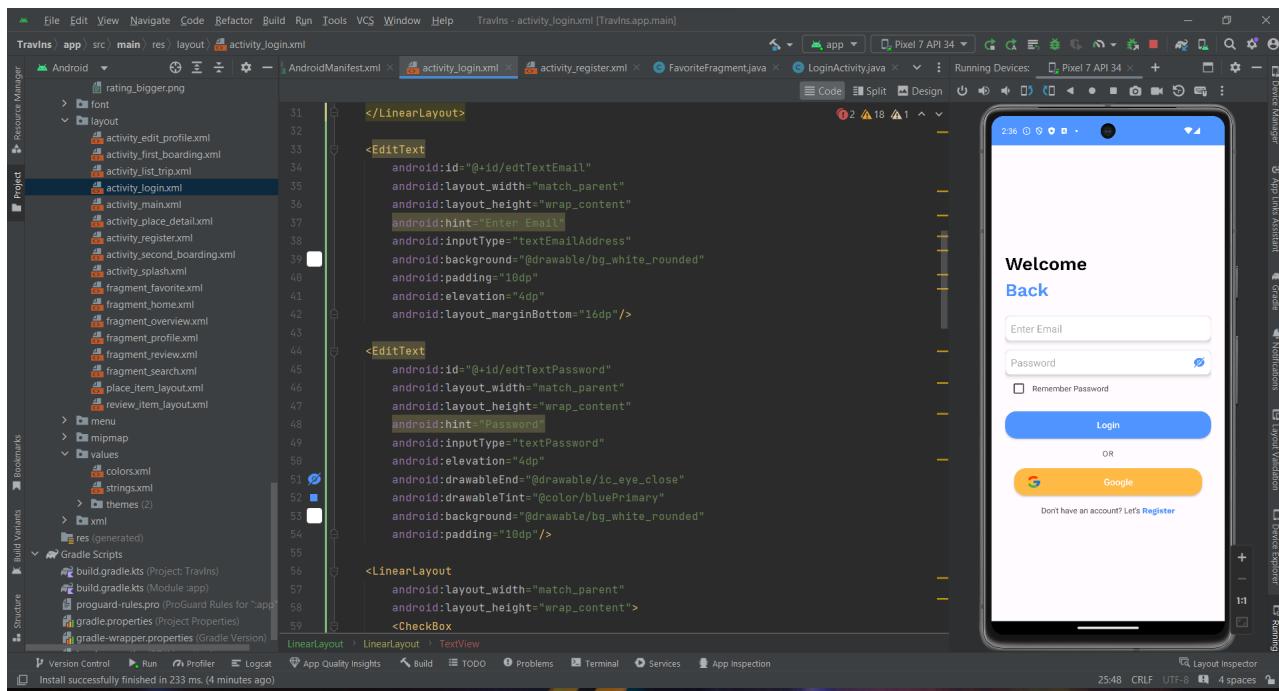
```
<application . . .>
    <activity
        android:name=".ui.activities.LoginActivity"
        android:exported="false" />
    </activity>
</application>
```

Fungsionalitas ini menggunakan sebuah Activity dengan nama LoginActivity.java yang dideklarasikan dengan elemen berikut dalam file AndroidManifest.xml:

```
<application . . .>
    <activity
        android:name=".ui.activities.RegisterActivity"
        android:exported="false" />
    </activity>
</application>
```

# Implementasi UI

## Layout 1: [activity\_login]



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.activities.LoginActivity"
    android:paddingHorizontal="24dp"
    android:orientation="vertical"
    android:gravity="center">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="24dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Welcome" />
    </LinearLayout>
</LinearLayout>
```

```
        android:fontFamily="@font/worksans_semiBold"
        android:textColor="@color/black"
        android:textSize="32sp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Back"
        android:fontFamily="@font/worksans_semiBold"
        android:textColor="@color/bluePrimary"
        android:textSize="32sp"/>
    </LinearLayout>

    <EditText
        android:id="@+id/editTextEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Email"
        android:inputType="textEmailAddress"
        android:background="@drawable/bg_white_rounded"
        android:padding="10dp"
        android:elevation="4dp"
        android:layout_marginBottom="16dp"/>

    <EditText
        android:id="@+id/editTextPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword"
        android:elevation="4dp"
        android:drawableEnd="@drawable/ic_eye_close"
        android:drawableTint="@color/bluePrimary"
        android:background="@drawable/bg_white_rounded"
        android:padding="10dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <CheckBox
            android:id="@+id/checkBox"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Remember Password"/>
    </LinearLayout>

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btnLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_blue_rounded"
```

```
        android:textAllCaps="false"
        android:text="Login"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:layout_marginVertical="16sp"
    />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OR"
        android:fontFamily="@font/worksans_regular"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center">
        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/btnLoginGoogle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/bg_yellow_rounded"
            android:textAllCaps="false"
            android:text="Google"
            android:textColor="@color/white"
            android:textSize="16sp"
            android:drawableStart="@drawable/ic_google"
            android:paddingHorizontal="20dp"
            android:drawablePadding="8dp"
            android:layout_marginHorizontal="16sp"
            android:layout_marginVertical="16sp"
        />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Don't have an account? Let's "/>
        <TextView
            android:id="@+id/moveToRegister"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Register"
            android:textColor="@color/bluePrimary"
            android:fontFamily="@font/worksans_bold"/>
    </LinearLayout>
</LinearLayout>
```

## 1. LinearLayout (Utama)

Attributes:

- xmlns:android, xmlns:app, xmlns:tools: Mendefinisikan namespace untuk atribut Android, AppCompat, dan tools.
- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- tools:context: Menentukan konteks untuk tujuan desain di Android Studio.
- android:paddingHorizontal: Memberikan padding horizontal pada layout.
- android:orientation: Menentukan orientasi layout (vertical).

## 2. LinearLayout (Judul "Welcome Back")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- android:orientation: Menentukan orientasi layout (vertical).
- android:paddingBottom: Memberikan padding di bagian bawah layout.

## 3. TextView (Judul "Welcome")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi TextView.
- android:text: Menetapkan teks pada TextView.
- android:fontFamily: Menentukan jenis huruf.

## 4. TextView (Judul "Back")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi TextView.
- android:text: Menetapkan teks pada TextView.
- android:fontFamily: Menentukan jenis huruf.

## 5. EditText (Input Email)

Attributes:

- `android:id`: Memberikan identifikasi unik pada `EditText`.
- `android:layout_width`, `android:layout_height`: Menentukan lebar dan tinggi `EditText`.
- `android:hint`: Menetapkan teks panduan untuk input.
- `android:inputType`: Menentukan jenis input (alamat email).
- `android:background`: Menetapkan latar belakang menggunakan drawable.
- `android:padding`: Memberikan padding pada `EditText`.
- `android:elevation`: Menetapkan elevasi tampilan.
- `android:layout_marginBottom`: Memberikan margin di bagian bawah.

## 6. `EditText` (Input Password)

Attributes:

Sama seperti `EditText` untuk input email, tetapi dengan tambahan atribut:

- `android:drawableEnd`, `android:drawableTint`: Menetapkan gambar (eye icon) dan pewarnaan pada ujung drawable.
- `android:inputType`: Menetapkan jenis input (kata sandi).

## 7. `LinearLayout` (Checkbox "Remember Password")

Attributes:

- `android:layout_width`, `android:layout_height`: Menentukan lebar dan tinggi layout.
- `android:gravity`: Menentukan gravitasi (pemusatan).

Child Views:

- `CheckBox`: Kotak centang untuk mengingat kata sandi.
- `TextView`: Teks "Remember Password".

## 8. `AppCompatButton` (Button "Login")

Attributes:

- `android:id`: Memberikan identifikasi unik pada `Button`.

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi Button.
- android:background: Menetapkan latar belakang menggunakan drawable.
- android:textAllCaps: Menetapkan apakah seluruh teks di Button harus huruf kapital.
- android:textColor: Menetapkan warna teks.
- android:textSize: Menetapkan ukuran teks.
- android:layout\_marginVertical: Memberikan margin vertikal.

## 9. TextView (Teks "OR")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi TextView.
- android:fontFamily: Menetapkan jenis huruf.

## 10. LinearLayout (Google Login Button)

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- android:gravity: Menentukan gravitasi (pemusatan).

Child Views:

- AppCompatButton: Button untuk login dengan Google.
- Berbagai atribut untuk styling dan penempatan.

## 11. LinearLayout (Teks "Don't have an account? Let's Register")

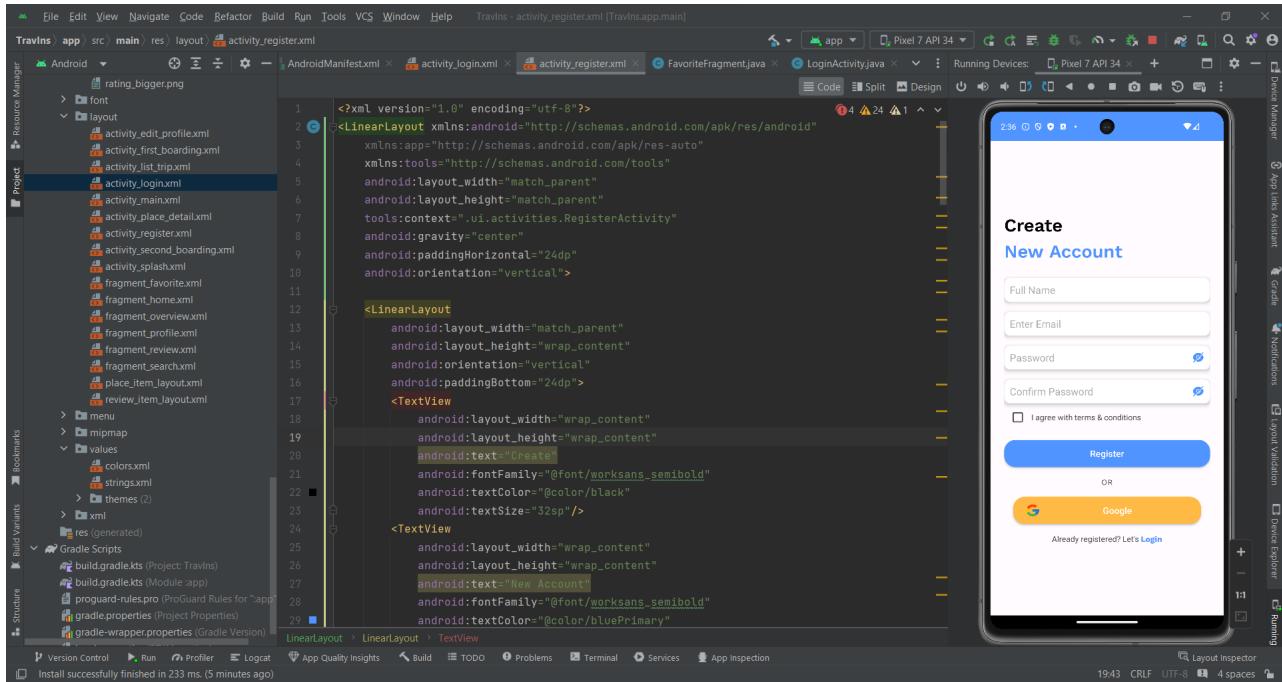
Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- android:gravity: Menentukan gravitasi (pemusatan).

Child Views:

- TextView: Teks "Don't have an account? Let's ".
- TextView: Teks "Register" dengan pewarnaan berbeda dan jenis huruf berbeda.

## Layout 2: [activity\_register]



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.activities.RegisterActivity"
    android:gravity="center"
    android:paddingHorizontal="24dp"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="24dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Create"
            android:fontFamily="@font/worksans_semiBold"
            android:textColor="@color/black"
            android:textSize="32sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Account"
            android:fontFamily="@font/worksans_semiBold"
            android:textColor="@color/bluePrimary"/>
    </LinearLayout>

```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Create"
        android:fontFamily="@font/worksans_semiBold"
        android:textColor="@color/black"
        android:textSize="32sp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Account"
        android:fontFamily="@font/worksans_semiBold"
        android:textColor="@color/bluePrimary"
        android:textSize="32sp"/>
</LinearLayout>

<EditText
    android:id="@+id/editTextFullName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Full Name"
    android:elevation="4dp"
    android:inputType="text"
    android:background="@drawable/bg_white_rounded"
    android:padding="10dp"
    android:layout_marginBottom="16dp"/>

<EditText
    android:id="@+id/editTextEmail"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter Email"
    android:elevation="4dp"
    android:inputType="textEmailAddress"
    android:background="@drawable/bg_white_rounded"
    android:padding="10dp"
    android:layout_marginBottom="16dp"/>

<EditText
    android:id="@+id/editTextPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Password"
    android:elevation="4dp"
    android:inputType="textPassword"
    android:layout_marginBottom="16dp"
    android:drawableEnd="@drawable/ic_eye_close"
    android:drawableTint="@color/bluePrimary"
    android:background="@drawable/bg_white_rounded"
    android:padding="10dp"/>

<EditText
    android:id="@+id/editTextCPassword"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:hint="Confirm Password"
        android:elevation="4dp"
        android:inputType="textPassword"
        android:drawableEnd="@drawable/ic_eye_close"
        android:drawableTint="@color/bluePrimary"
        android:background="@drawable/bg_white_rounded"
        android:padding="10dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <CheckBox
            android:id="@+id/checkBox"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I agree with terms & conditions"/>
    </LinearLayout>

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id	btnRegister"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_blue_rounded"
        android:textAllCaps="false"
        android:text="Register"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:layout_marginVertical="16sp"
    />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OR"
        android:fontFamily="@font/worksans_regular"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center">
        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id	btnRegisterGoogle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/bg_yellow_rounded"
            android:textAllCaps="false"
            android:text="Google"
            android:textColor="@color/white"
            android:textSize="16sp"
        />
    </LinearLayout>

```

```
        android:drawableStart="@drawable/ic_google"
        android:paddingHorizontal="20dp"
        android:drawablePadding="8dp"
        android:layout_marginHorizontal="16sp"
        android:layout_marginVertical="16sp"
    />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Already registered? Let's "/>
    <TextView
        android:id="@+id/moveToLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        android:textColor="@color/bluePrimary"
        android:fontFamily="@font/worksans_bold"/>
</LinearLayout>

</LinearLayout>
```

## 1. LinearLayout (Utama)

### Attributes:

- `xmlns:android`, `xmlns:app`, `xmlns:tools`: Mendefinisikan namespace untuk atribut Android, AppCompat, dan tools.
- `android:layout_width`, `android:layout_height`: Menentukan lebar dan tinggi layout.
- `tools:context`: Menentukan konteks untuk tujuan desain di Android Studio.
- `android:gravity`: Menentukan gravitasi (pemusatan) kontennya.
- `android:paddingHorizontal`: Memberikan padding horizontal pada layout.
- `android:orientation`: Menentukan orientasi layout (vertical).
- 

## 2. LinearLayout (Judul "Create New Account")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- android:orientation: Menentukan orientasi layout (vertical).
- android:paddingBottom: Memberikan padding di bagian bawah layout.

### 3. TextView (Judul "Create")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi TextView.
- android:text: Menetapkan teks pada TextView.
- android:fontFamily: Menetapkan jenis huruf.

### 4. TextView (Judul "New Account")

Attributes:

Sama seperti TextView untuk judul "Create", tetapi dengan perbedaan pada teks dan warna.

### 5. EditText (Input Full Name)

Attributes:

- android:id: Memberikan identifikasi unik pada EditText.
- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi EditText.
- android:hint: Menetapkan teks panduan untuk input.
- android:inputType: Menentukan jenis input (teks).
- android:background: Menetapkan latar belakang menggunakan drawable.
- android:padding: Memberikan padding pada EditText.
- android:elevation: Menetapkan elevasi tampilan.
- android:layout\_marginBottom: Memberikan margin di bagian bawah.
- 

### 6. EditText (Input Email)

Attributes:

Sama seperti EditText untuk input Full Name, tetapi dengan perbedaan pada id, hint, dan jenis input (alamat email).

#### 7. EditText (Input Password)

Attributes:

Sama seperti EditText untuk input Full Name, tetapi dengan perbedaan pada id, hint, jenis input (kata sandi), dan drawable untuk mata yang tertutup (ic\_eye\_close).

#### 8. EditText (Konfirmasi Password)

Attributes:

Sama seperti EditText untuk input Password, tetapi dengan perbedaan pada id, hint, dan jenis input.

#### 9. LinearLayout (Checkbox "I agree with terms & conditions")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- android:gravity: Menentukan gravitasi (pemusatan).

Child Views:

- CheckBox: Kotak centang untuk menyetujui syarat dan ketentuan.
- TextView: Teks "I agree with terms & conditions".

#### 10. AppCompatButton (Button "Register")

Attributes:

- android:id: Memberikan identifikasi unik pada Button.
- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi Button.
- android:background: Menetapkan latar belakang menggunakan drawable.
- android:textAllCaps: Menetapkan apakah seluruh teks di Button harus huruf kapital.
- android:textColor: Menetapkan warna teks.

- android:textSize: Menetapkan ukuran teks.
- android:layout\_marginVertical: Memberikan margin vertikal.

## 11. TextView (Teks "OR")

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi TextView.
- android:fontFamily: Menetapkan jenis huruf.

## 12. LinearLayout (Google Register Button)

Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- android:gravity: Menentukan gravitasi (pemusatan).

Child Views:

- AppCompatButton: Button untuk register dengan Google.
- Berbagai atribut untuk styling dan penempatan.

## 13. LinearLayout (Teks "Already registered? Let's Login")

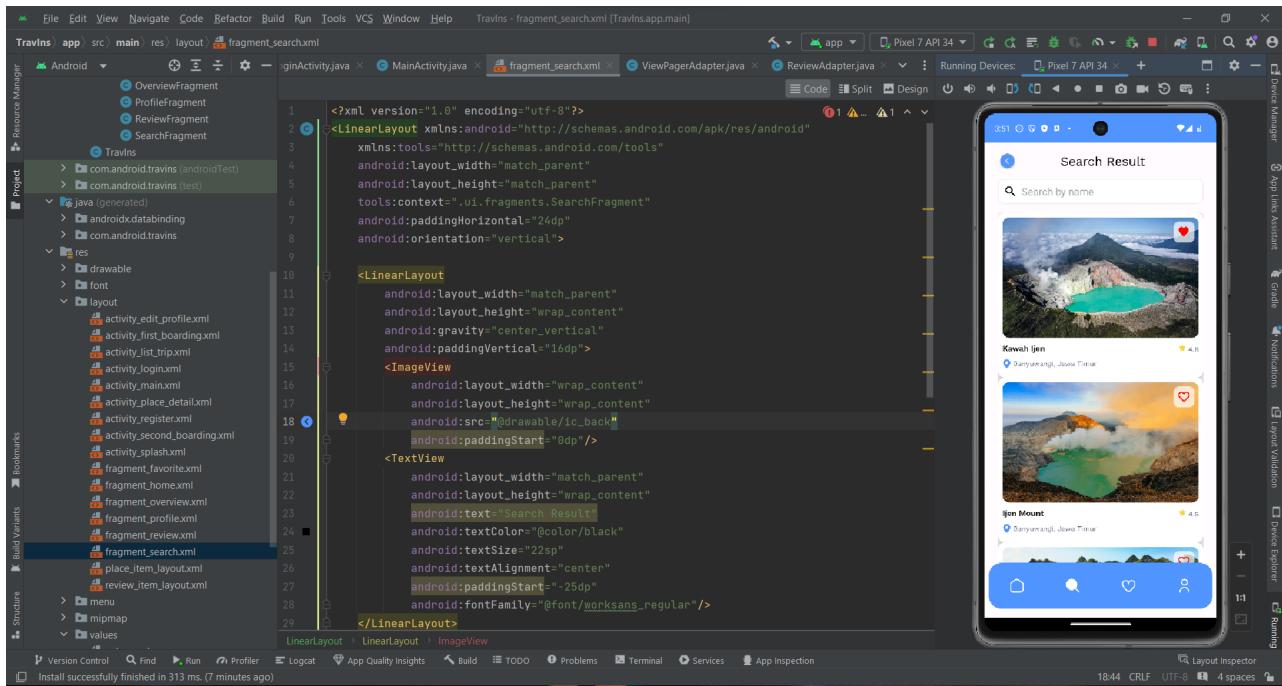
Attributes:

- android:layout\_width, android:layout\_height: Menentukan lebar dan tinggi layout.
- android:gravity: Menentukan gravitasi (pemusatan).

Child Views:

- TextView: Teks "

## Layout 3: [fragment\_search]



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.fragments.SearchFragment"
    android:paddingHorizontal="24dp"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingVertical="16dp">
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_back"
            android:paddingStart="0dp"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Search Result"
            android:textColor="@color/black"
            android:textSize="22sp"
            android:textAlignment="center"
            android:paddingStart="-25dp"
            android:fontFamily="@font/worksans_regular"/>
    </LinearLayout>
</LinearLayout>
```

```
        android:textSize="22sp"
        android:textAlignment="center"
        android:paddingStart="-25dp"
        android:fontFamily="@font/worksans_regular"/>
    
```

```
</LinearLayout>

<EditText
    android:id="@+id edtTxtSearch"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Search by name"
    android:drawableTint="@color/black"
    android:drawablePadding="8dp"
    android:drawableStart="@drawable/ic_search"
    android:background="@drawable/bg_white_rounded"
    android:padding="10dp"
    android:layout_marginBottom="16dp"/>

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvListTrip"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>

</LinearLayout>
```

## 1. LinearLayout Utama:

- `android:orientation="vertical"`: Menetapkan orientasi layout ke vertikal.
- `android:paddingHorizontal="24dp"`: Menambahkan padding horizontal sebesar 24dp.

## 2. LinearLayout (Header):

- `android:layout_width="match_parent"`: Mengatur lebar layout ke seluruh lebar parent.
- `android:layout_height="wrap_content"`: Mengatur tinggi layout sesuai dengan kontennya.
- `android:gravity="center_vertical"`: Mendatar pusat elemen anak secara vertikal.
- `android:paddingVertical="16dp"`: Menambahkan padding vertikal sebesar 16dp.

## Elemen dalam LinearLayout Header:

- `ImageView`: Gambar panah kembali.
- `android:layout_width="wrap_content"`: Mengatur lebar gambar sesuai dengan kontennya.

- android:layout\_height="wrap\_content": Mengatur tinggi gambar sesuai dengan kontennya.
- android:src="@drawable/ic\_back": Menetapkan sumber gambar dari resource dengan nama "ic\_back".
- android:paddingStart="0dp": Menetapkan padding di sisi kiri gambar.
- TextView: Judul "Search Result".
- android:layout\_width="match\_parent": Mengatur lebar teks agar mengisi seluruh lebar parent.
- android:layout\_height="wrap\_content": Mengatur tinggi teks sesuai dengan kontennya
- android:text="Search Result": Menetapkan teks "Search Result".
- android:textColor="@color/black": Menetapkan warna teks.
- android:textSize="22sp": Menetapkan ukuran teks.
- android:textAlignment="center": Menyelaraskan teks ke tengah secara horizontal.
- android:paddingStart="-25dp": Memberikan padding negatif untuk menyesuaikan posisi teks.
- android:fontFamily="@font/worksans\_regular": Menetapkan jenis huruf.

### 3. EditText (Search Box):

- android:id="@+id/edtTxtSearch": Memberikan ID pada EditText untuk referensi di dalam kode Java.
- android:layout\_width="match\_parent": Mengatur lebar EditText agar mengisi seluruh lebar parent.
- android:layout\_height="wrap\_content": Mengatur tinggi EditText sesuai dengan kontennya.
- android:hint="Search by name": Menetapkan hint (teks panduan) yang muncul ketika EditText kosong.
- android:drawableTint="@color/black": Menetapkan warna ikon Drawable.
- android:drawablePadding="8dp": Menambahkan padding antara ikon dan teks di dalam EditText.
- android:drawableStart="@drawable/ic\_search": Menetapkan ikon pencarian di sisi kiri EditText.
- android:background="@drawable/bg\_white\_rounded": Menetapkan latar belakang berbentuk bulat dan berwarna putih.
- android:padding="10dp": Menambahkan padding internal sebesar 10dp.
- android:layout\_marginBottom="16dp": Menambahkan margin di bagian bawah sebesar 16dp.

### 4. RecyclerView:

- android:id="@+id/rvListTrip": Memberikan ID pada RecyclerView untuk referensi di dalam kode Java.
- android:layout\_width="match\_parent": Mengatur lebar RecyclerView agar mengisi seluruh lebar parent.
- android:layout\_height="wrap\_content": Mengatur tinggi RecyclerView sesuai

dengan kontennya.

#### Penggunaan Atribut:

- `xmlns:android="http://schemas.android.com/apk/res/android"`: Menentukan namespace default yang digunakan dalam file XML.
- `xmlns:tools="http://schemas.android.com/tools"`: Namespace yang digunakan untuk akses alat bantu pengembangan Android.
- `tools:context=".ui.fragments.SearchFragment"`: Menetapkan konteks aktivitas yang digunakan oleh alat bantu pengembangan.
- `android:paddingHorizontal="24dp"`: Menetapkan padding horizontal sebesar 24dp pada layout utama.
- `android:orientation="vertical"`: Menetapkan orientasi layout utama ke vertikal.
- `android:gravity="center_vertical"`: Mendatar pusat elemen anak secara vertikal pada LinearLayout header.
- `android:paddingVertical="16dp"`: Menambahkan padding vertikal sebesar 16dp pada LinearLayout header.
- `android:layout_width="match_parent"`: Menetapkan lebar elemen anak agar sesuai dengan lebar parent.
- `android:layout_height="wrap_content"`: Menetapkan tinggi elemen anak agar sesuai dengan kontennya.
- `android:src="@drawable/ic_back"`: Menetapkan sumber gambar untuk ImageView dari resource.
- `android:text="Search Result"`: Menetapkan teks pada TextView.
- `android:textColor="@color/black"`: Menetapkan warna teks pada TextView.
- `android:textSize="22sp"`: Menetapkan ukuran teks pada TextView.
- `android:textAlignment="center"`: Menyelaraskan teks ke tengah secara horizontal pada TextView.
- `android:fontFamily="@font/worksans_regular"`: Menetapkan jenis huruf pada TextView.
- `android:id="@+id edtTxtSearch"`: Memberikan ID pada EditText untuk referensi di dalam kode Java.
- `android:hint="Search by name"`: Menetapkan hint pada EditText.
- `android:drawableTint="@color/black"`: Menetapkan warna ikon Drawable pada EditText.
- `android:drawablePadding="8dp"`: Menambahkan padding antara ikon dan teks di dalam EditText.
- `android:drawableStart="@drawable/ic_search"`: Menetapkan ikon pencarian di sisi kiri EditText.
- `android:background="@drawable/bg_white_rounded"`: Menetapkan latar belakang berbentuk bulat dan berwarna putih pada EditText.
- `android:padding="10dp"`: Menambahkan padding internal sebesar 10dp pada EditText.
- `android:layout_marginBottom="16dp"`: Menambahkan margin di bagian bawah sebesar 16dp pada EditText.
- `android:id="@+id/rvListTrip"`: Memberikan ID



# Implementasi Kode Program Aplikasi

Kode program berikut digunakan untuk menjalankan Activity LoginActivity dengan tambahan Toast untuk pop up kecil feedback program.

## Kode Program activity: LoginActivity

```
package com.android.travins.ui.activities;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.text.method.PasswordTransformationMethod;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Toast;

import com.android.travins.R;
import com.android.travins.databinding.ActivityLoginBinding;
import
com.android.travins.databinding.ActivitySecondBoardingBinding;
import com..CallbackManager;
import com..login.widget.LoginButton;
import com.google.android.gms.auth.api.Auth;
import
com.google.android.gms.auth.api.identity.BeginSignInRequest;
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.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
```

```
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.GoogleAuthProvider;
//import com..login.LoginManager;

public class LoginActivity extends AppCompatActivity {
    private ActivityLoginBinding binding;
    private static final int RC_SIGN_IN = 123; // Any integer
constant for the request code
    private GoogleSignInClient mGoogleSignInClient;
    private BeginSignInRequest signInRequest;
    private FirebaseAuth auth;
    private boolean passwordVisible = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityLoginBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        //google
        auth = FirebaseAuth.getInstance();

        signInRequest = BeginSignInRequest.builder()

.setGoogleIdTokenRequestOptions(BeginSignInRequest.GoogleIdToke
nRequestOptions.builder()
            .setSupported(true)

.setServerClientId(getString(R.string.default_web_client_id))
            .setFilterByAuthorizedAccounts(true)
            .build())
        .build();

        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN
        )

.requestIdToken(signInRequest.getGoogleIdTokenRequestOptions().getServerClientId())
            .requestEmail()
            .build();

        mGoogleSignInClient = GoogleSignIn.getClient(this,
gso);
```

```
binding.moveToRegister.setOnClickListener(v -> {
    Intent mainIntent = new Intent(LoginActivity.this,
RegisterActivity.class);
    startActivity(mainIntent);
});

binding.btnLoginGoogle.setOnClickListener(v -> {
    signIn();
});

binding.btnLogin.setOnClickListener(v -> {
    if
(binding.edtTextEmail.getText().toString().isEmpty() ||
binding.edtTextPassword.getText().toString().isEmpty()){
        Toast.makeText(LoginActivity.this,"Email or
password is required!",Toast.LENGTH_SHORT).show();
    }else{
        signInWithEmailPassword();
    }
});

binding.edtTextPassword.setOnTouchListener((v, event)
-> {
    if (event.getAction() == MotionEvent.ACTION_UP) {
        // Check if touch is within the bounds of the
drawable
        if (event.getRawX() >=
(binding.edtTextPassword.getRight() -
binding.edtTextPassword.getCompoundDrawables()[2].getBounds().w
idth())) {
            // Toggle password visibility
            togglePasswordVisibility();
            return true;
        }
    }
    return false;
});
}

private void signInWithEmailPassword(){
String email =
binding.edtTextEmail.getText().toString();
String password =
binding.edtTextPassword.getText().toString();
auth.signInWithEmailAndPassword(email, password)
```

```

        .addOnCompleteListener(this, task -> {
            if (task.isSuccessful()) {
                Toast.makeText(this, "Login
successful.", Toast.LENGTH_SHORT).show();
                Intent mainIntent = new
Intent(LoginActivity.this, MainActivity.class);
                startActivity(mainIntent);
            } else {
                Toast.makeText(this, "Login failed.",
Toast.LENGTH_SHORT).show();
            }
        });
    }
    private void signIn() {
        Intent signInIntent =
mGoogleSignInClient.getSignInIntent();
        startActivityForResult(signInIntent, RC_SIGN_IN);
    }

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

        if (requestCode == RC_SIGN_IN) {
            Task<GoogleSignInAccount> task =
GoogleSignIn.getSignedInAccountFromIntent(data);
            handleSignInResult(task);
        }
    }

    private void handleSignInResult(Task<GoogleSignInAccount>
completedTask) {
        try {
            GoogleSignInAccount account =
completedTask.getResult(ApiException.class);
            FirebaseGoogleAuth(account);

        } catch (ApiException e) {
            Log.w("GoogleSignIn", "signInResult:failed code=" +
e.getStatusMessage());
            Toast.makeText(this, "Google Sign In Failed",
Toast.LENGTH_SHORT).show();
        }
    }
}

```

```
private void FirebaseAuth(GoogleSignInAccount acc) {
    AuthCredential authCredential =
GoogleAuthProvider.getCredential(acc.getIdToken(),null);

auth.signInWithCredential(authCredential).addOnCompleteListener
(this, new OnCompleteListener<AuthResult>() {
    @Override
        public void onComplete(@NonNull Task<AuthResult>
task) {
            if (task.isSuccessful()){
                Toast.makeText(LoginActivity.this, "Login
Success", Toast.LENGTH_SHORT).show();
                Intent mainIntent = new
Intent(LoginActivity.this, MainActivity.class);
                startActivity(mainIntent);
            }
            else {
                Toast.makeText(LoginActivity.this,"Login
Error",Toast.LENGTH_SHORT).show();

            }
        }
    });
}

private void togglePasswordVisibility() {
    if (passwordVisible) {
        // Hide password
        Drawable eyeIcon = ContextCompat.getDrawable(this,
R.drawable.ic_eye_close);
        binding.edtTextPassword.setTransformationMethod(new
PasswordTransformationMethod());

binding.edtTextPassword.setCompoundDrawablesRelativeWithIntrinsic
Bounds(null,null,eyeIcon,null);
    } else {
        // Show password
        Drawable eyeIcon = ContextCompat.getDrawable(this,
R.drawable.ic_eye);

binding.edtTextPassword.setTransformationMethod(null);

binding.edtTextPassword.setCompoundDrawablesRelativeWithIntrinsic
Bounds(null,null,eyeIcon,null);
    }
    passwordVisible = !passwordVisible;
}
```

```
// Move cursor to the end of the text

binding.edtTextPassword.setSelection(binding.edtTextPassword.get
tText().length());
}

}
```

Kode program berikut digunakan untuk menjalankan Activity LoginActivity dengan menggunakan autentikasi untuk mengambil Instance dari Firebase Authentication dengan:

```
private FirebaseAuth auth; FirebaseAuth.getInstance();

private void signInWithEmailAndPassword() {

    String email = binding.edtTextEmail.getText().toString();

                String      password      =
binding.edtTextPassword.getText().toString();

    auth.signInWithEmailAndPassword(email, password)

        .addOnCompleteListener(this, task -> {

            if (task.isSuccessful()) {

                Toast.makeText(this, "Login successful.",

Toast.LENGTH_SHORT).show();

                    Intent mainIntent = new
Intent(LoginActivity.this, MainActivity.class);

                startActivity(mainIntent);

            } else {

                Toast.makeText(this, "Login failed.",

Toast.LENGTH_SHORT).show();
            }
        }
    );
}
```

```
        }

    });

}
```

Method tersebut digunakan untuk mengautentikasi email dan password yang digunakan pengguna dengan Firebase Authentication. Kemudian ada method “Toast.makeText()” untuk memberikan feedback kepada pengguna apakan password dan email yang diinputkan sudah benar atau belum. Ketika berhasil maka akan menampilkan pop up “Login Success” namun jika gagal akan memunculkan pop up “Login Error”.

#### Kode Program activity: RegisterActivity

```
package com.android.travins.ui.activities;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.text.method.PasswordTransformationMethod;
import android.util.Log;
import android.view.MotionEvent;
import android.widget.EditText;
import android.widget.Toast;

import com.android.travins.R;
import com.android.travins.databinding.ActivityLoginBinding;
import com.android.travins.databinding.ActivityRegisterBinding;
import
com.google.android.gms.auth.api.identity.BeginSignInRequest;
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.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.GoogleAuthProvider;
import com.google.firebaseio.firebaseio.DocumentReference;
import com.google.firebaseio.firebaseio.FirebaseFirestore;

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

public class RegisterActivity extends AppCompatActivity {

    private ActivityRegisterBinding binding;
    private static final int RC_SIGN_IN = 123; // Any integer
constant for the request code
    private GoogleSignInClient mGoogleSignInClient;
    private BeginSignInRequest signInRequest;
    private FirebaseAuth auth;
    private FirebaseFirestore db;
    private boolean passwordVisibleC = false;
    private boolean passwordVisible = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityRegisterBinding.inflate(LayoutInflater());
        setContentView(binding.getRoot());

        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        signInRequest = BeginSignInRequest.builder()

.setGoogleIdTokenRequestOptions(BeginSignInRequest.GoogleIdToke
nRequestOptions.builder()
            .setSupported(true)

.setServerClientId(getString(R.string.default_web_client_id))
```

```
        .setFilterByAuthorizedAccounts(true)
        .build())
    .build();

        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
)

.requestIdToken(signInRequest.getGoogleIdTokenRequestOptions() .
getServerClientId())
        .requestEmail()
        .build();

mGoogleSignInClient = GoogleSignIn.getClient(this,
gso);

binding.editTextPassword.setOnTouchListener((v, event)
-> {
    if (event.getAction() == MotionEvent.ACTION_UP) {
        // Check if touch is within the bounds of the
drawable
                if (event.getRawX() >=
(binding.editTextPassword.getRight() -
binding.editTextPassword.getCompoundDrawables() [2].getBounds() .
width())) {
                    // Toggle password visibility
                        passwordVisible =
togglePasswordVisibility(binding.editTextPassword,passwordVisib
le);
                    return true;
                }
            }
        return false;
});

binding.editTextCPassword.setOnTouchListener((v, event)
-> {
    if (event.getAction() == MotionEvent.ACTION_UP) {
        // Check if touch is within the bounds of the
drawable
                if (event.getRawX() >=
(binding.editTextCPassword.getRight() -
binding.editTextCPassword.getCompoundDrawables() [2].getBounds() .
width())) {
                    // Toggle password visibility
                        passwordVisibleC =

```

```
        togglePasswordVisibility(binding.editTextCPassword,passwordVisibleC);
                return true;
            }
        }
        return false;
    });

binding.moveToLogin.setOnClickListener(v -> {
    Intent mainIntent = new Intent(RegisterActivity.this, LoginActivity.class);
    startActivity(mainIntent);
});

binding.btnRegisterGoogle.setOnClickListener(v -> {
    signIn();
});

binding.btnRegister.setOnClickListener(v -> {
    String email = binding.editTextEmail.getText().toString();
    String fullName = binding.editTextFullName.getText().toString();
    String password = binding.editTextPassword.getText().toString();
    String cPassword = binding.editTextCPassword.getText().toString();

    if (email.isEmpty() || fullName.isEmpty() || password.isEmpty() || cPassword.isEmpty()){
        Toast.makeText(RegisterActivity.this,"Name, email or password is required!",Toast.LENGTH_SHORT).show();
    }else{
        if (cPassword.matches(password)){
            registerEmailPassword(email,fullName,password,cPassword);
        }
        else{
            Toast.makeText(RegisterActivity.this,"Password not match!",Toast.LENGTH_SHORT).show();
        }
    }
});
```

```

        private void registerEmailPassword(String email, String
fullName, String password, String cPassword) {
    auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, task -> {
            if (task.isSuccessful()) {
                addToFirebase(fullName, email, password);
            } else {
                // If registration fails, display a
                message to the user.
                Toast.makeText(this, "Registration
failed.", Toast.LENGTH_SHORT).show();
            }
        });
}

private void addToFirebase(String name, String email,
String password) {
    Map<String, Object> user = new HashMap<>();
    user.put("fullName", name);
    user.put("email", email);
    user.put("password", password);
    user.put("bod", "");
    user.put("username", "");
    user.put("gender", "");
    user.put("country", "");
    user.put("address", "");
}

db.collection("users").document(auth.getCurrentUser().getUid())
.set(user)
    .addOnSuccessListener(documentReference -> {
        Toast.makeText(RegisterActivity.this,
"Registration successful, please login.",
Toast.LENGTH_SHORT).show();
        Intent mainIntent = new
Intent(RegisterActivity.this, LoginActivity.class);
        startActivity(mainIntent);
        finish();
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e)
{
        Toast.makeText(RegisterActivity.this,
"Eror adding document", Toast.LENGTH_SHORT).show();
    }
}

```

```
        }
    });

    private void signIn() {
        Intent signInIntent = mGoogleSignInClient.getSignInIntent();
        startActivityForResult(signInIntent, RC_SIGN_IN);
    }

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

        if (requestCode == RC_SIGN_IN) {
            Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
            handleSignInResult(task);
        }
    }

    private void handleSignInResult(Task<GoogleSignInAccount> completedTask) {
        try {
            GoogleSignInAccount account = completedTask.getResult(ApiException.class);
            FirebaseGoogleAuth(account);

        } catch (ApiException e) {
            Log.w("GoogleSignIn", "signInResult:failed code=" + e.getStatusMessage());
            Toast.makeText(this, "Google Sign In Failed", Toast.LENGTH_SHORT).show();
        }
    }

    private void FirebaseGoogleAuth(GoogleSignInAccount acc) {
        AuthCredential authCredential = GoogleAuthProvider.getCredential(acc.getIdToken(), null);

        auth.signInWithCredential(authCredential).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult>
```

```
task) {
            if (task.isSuccessful()) {
                Toast.makeText(RegisterActivity.this,
"Login Success", Toast.LENGTH_SHORT).show();
                Intent mainIntent = new
Intent(RegisterActivity.this, MainActivity.class);
                startActivity(mainIntent);
            } else {
                Toast.makeText(RegisterActivity.this,"Login
Error",Toast.LENGTH_SHORT).show();
            }
        });
    }

    private Boolean togglePasswordVisibility(EditText
editText,Boolean pass) {
    if (pass) {
        // Hide password
        Drawable eyeIcon = ContextCompat.getDrawable(this,
R.drawable.ic_eye_close);
        editText.setTransformationMethod(new
PasswordTransformationMethod());

        editText.setCompoundDrawablesRelativeWithIntrinsicBounds(null,n
ull,eyeIcon,null);
    } else {
        // Show password
        Drawable eyeIcon = ContextCompat.getDrawable(this,
R.drawable.ic_eye);
        editText.setTransformationMethod(null);

        editText.setCompoundDrawablesRelativeWithIntrinsicBounds(null,n
ull,eyeIcon,null);
    }
    pass = !pass;

    // Move cursor to the end of the text
    editText.setSelection(editText.getText().length());

    return pass;
}
}
```

Kode di atas adalah implementasi dari kelas RegisterActivity dalam aplikasi Android yang digunakan untuk mengelola proses registrasi pengguna. Berikut adalah atribut dan metode yang dapat digunakan:

Instansiasi Atribut:

1. binding: Objek dari kelas ActivityRegisterBinding yang digunakan untuk mengikat dan mengakses elemen antarmuka pengguna dari layout XML.
2. RC\_SIGN\_IN: Konstanta integer yang digunakan sebagai kode permintaan (request code) untuk proses sign-in Google.
3. mGoogleSignInClient: Objek dari kelas GoogleSignInClient untuk mengelola proses sign-in dengan akun Google.
4. signInRequest: Objek dari kelas BeginSignInRequest untuk konfigurasi permintaan sign-in Google.
5. auth: Objek dari kelas FirebaseAuth untuk mengelola otentikasi Firebase.
6. db: Objek dari kelas FirebaseFirestore untuk mengakses dan menyimpan data di Firebase Firestore.
7. passwordVisibleC dan passwordVisible: Variabel boolean untuk menentukan apakah kata sandi terlihat atau tidak.

Metode yang digunakan:

1. onCreate(Bundle savedInstanceState): Metode untuk menginisialisasi aktivitas saat dibuat, termasuk mengatur tampilan, mengonfigurasi objek GoogleSignInClient, dan menangani interaksi pengguna.
2. togglePasswordVisibility(EditText editText, Boolean pass): Metode untuk menampilkan atau menyembunyikan teks kata sandi pada EditText. Mengganti ikon mata terbuka/tutup dan memindahkan kursor ke akhir teks.
3. registerEmailPassword(String email, String fullName, String password, String cPassword): Metode untuk mendaftarkan pengguna dengan email dan kata sandi. Juga menambahkan informasi pengguna ke Firestore setelah registrasi berhasil.
4. addToFirebase(String name, String email, String password): Metode untuk

menambahkan informasi pengguna ke Firestore setelah registrasi berhasil.

5. `signIn()`: Metode untuk memulai proses sign-in dengan akun Google.
6. `onActivityResult(int requestCode, int resultCode, Intent data)`: Metode yang dipanggil saat hasil dari aktivitas yang diluncurkan kembali ke RegisterActivity, digunakan untuk menangani hasil sign-in Google.
7. `handleSignInResult(Task<GoogleSignInAccount> completedTask)`: Metode untuk menangani hasil sign-in Google.
8. `FirebaseGoogleAuth(GoogleSignInAccount acc)`: Metode untuk mengotentikasi pengguna Firebase dengan kredensial Google.

#### Kode Program activity: ListTripActivity

```
package com.android.travins.ui.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.Toast;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.android.travins.databinding.ActivityListTripBinding;
import com.android.travins.ui.adapters.HorizontalAdapter;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class ListTripActivity extends AppCompatActivity
implements HorizontalAdapter.OnItemClickListener {

    private ActivityListTripBinding binding;
    private FirebaseAuth auth;
```

```
private FirebaseFirestore db;
private HorizontalAdapter adapter;
private ArrayList<Places> listDest = new ArrayList<>();
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityListTripBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());
    listDest = new ArrayList<>();
    db = FirebaseFirestore.getInstance();
    auth = FirebaseAuth.getInstance();

    adapter = new HorizontalAdapter(this,2);
    adapter.setOnItemClickListener(this);
    fetchData();

    binding.btnExit.setOnClickListener(v -> finish());

    binding.rvListInspiration.setLayoutManager(new
LinearLayoutManager(this, LinearLayout.VERTICAL,
false));
    binding.rvListInspiration.setAdapter(adapter);

    binding.edtTxtSearch.addTextChangedListener(new
TextWatcher() {
    @Override
        public void beforeTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
    }

    @Override
        public void onTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
            adapter.getFilter().filter(charSequence);
    }

    @Override
    public void afterTextChanged(Editable editable) {
    }
});

private void fetchData() {
    db.collection("places")
        .get()
```

```
        .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
    @Override
        public void onComplete(@NonNull
Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document
: task.getResult()) {
                    Map<String, Object> data =
document.getData();
                    Places places = new
Places(document.getId(),
data.get("name"), (String) data.get("image"),
data.get("description"), (String) data.get("price"),
data.get("location"), (String) data.get("rate"),
data.get("total_review"));

                    fetchFavData(places);
                }
            } else {
                Toast.makeText(ListTripActivity.this,"Fetch
failed",Toast.LENGTH_SHORT).show();
            }
        });
}
private Places fetchFavData(Places places) {

db.collection("users").document(auth.getCurrentUser().getUid())
.collection("favorites")
.get()
.addOnCompleteListener(task -> {
    if (task.isSuccessful()) {
        for (QueryDocumentSnapshot document :
task.getResult()) {
            Map<String, Object> data =
document.getData();

            if
(document.getId().equals(places.getId())))
{
```

```
        places.setFav(true);
    }
    listDest.add(places);
}
adapter.updateData(listDest);
} else {
}
);
}

return places;
}

@Override
public void onItemClick(Places place) {
    Intent intent = new Intent(ListTripActivity.this,
PlaceDetailActivity.class);
    intent.putExtra("place",place);
    startActivity(intent);
}
}
```

Kode program berikut digunakan untuk menjalankan Activity ListTripActivity dengan menggunakan FirebaseFirestore untuk mengakses data dari Firestore Database dengan:

```
private FirebaseFirestore db; db = FirebaseFirestore.getInstance();
```

Melakukan instansiasi pada class onCreate. Disini dilakukan juga setLayoutManager dari semua card destinasi yaitu menggunakan LinearLayoutManager

```
super.onCreate(savedInstanceState);

        binding = ActivityListTripBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        listDest = new ArrayList<>();

        db = FirebaseFirestore.getInstance();

        auth = FirebaseAuth.getInstance();

        adapter = new HorizontalAdapter(this,2);

        adapter.setOnItemClickListener(this);

        fetchData();

        binding.btnExit.setOnClickListener(v -> finish());

        binding.rvListInspiration.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false));

        binding.rvListInspiration.setAdapter(adapter);
```

Method fetchData() digunakan untuk mengakses data yang berada pada Firestore Database. db.collection("places") digunakan untuk mengakses data dengan collection path places, kemudian diambil sesuai variablenya. jika gagal maka akan menampilkan pop up "Fetch data failed" menggunakan Toast.makeText.

```
private void fetchData() {

    db.collection("places")

        .get()
```

```
        .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
    @Override
        public void onComplete(@NonNull
Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document :
task.getResult()) {
                    Map<String, Object> data =
document.getData();
                    Places places = new
Places(document.getId(),
                    (String) data.get("name"),
                    (String) data.get("image"),
                    (String)
data.get("description"), (String) data.get("price"),
                    (String)
data.get("location"), (String) data.get("rate"),
                    (String)
data.get("total_review"));
                    fetchFavData(places);
                }
            } else {
                Toast.makeText(ListTripActivity.this,"Fetch
failed",Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

```
        }

    });

}
```

Kode dibawah dilakukan untuk mengakses fitur search dengan menggunakan .addTextChangedListener pada edtTxtSearch. Metode TextChanged dipanggil untuk memberi tahu bahwa, di dalam s, karakter hitungan yang dimulai dari awal akan diganti dengan teks baru dengan panjang setelahnya. Setelah terdeteksi adanya perubahan maka akan dipanggil adapter dan mengimplementasikan method getFilter() untuk memfilter RecyclerView.

```
binding.edtTxtSearch.addTextChangedListener(new TextWatcher() {

    @Override

        public void beforeTextChanged(CharSequence charSequence,
        int i, int i1, int i2) {

    }

    @Override

        public void onTextChanged(CharSequence charSequence, int
        i, int i1, int i2) {

            adapter.getFilter().filter(charSequence);

    }

    @Override

        public void afterTextChanged(Editable editable) {

    }

});
```

Pada ListTripActivity juga digunakan sebuah intent untuk memulai activity PlaceDetailActivity dengan membawa data tambahan berupa objek “places”

```
public void onItemClick(Places place) {  
  
    Intent intent = new Intent(ListTripActivity.this,  
PlaceDetailActivity.class);  
  
    intent.putExtra("place",place);  
  
    startActivity(intent);  
  
}
```

## Implementasi ListView/AdapterView/RecyclerView

Penggunaan ListView/AdapterView/RecyclerView da.

Kode Program adapter : HorizontalAdapter

```
package com.android.travins.ui.adapters;  
  
import static com.google.android.material.internal.ViewUtils.dpToPx;  
  
import android.content.Context;  
  
import android.content.Intent;  
  
import android.util.Log;  
  
import android.view.LayoutInflater;  
  
import android.view.View;  
  
import android.view.ViewGroup;  
  
import android.widget.Filter;  
  
import android.widget.Filterable;
```

```
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.bumptech.glide.Glide;

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

public class HorizontalAdapter extends RecyclerView.Adapter<HorizontalAdapter.ViewHolder> implements Filterable {
    private Context context;
    private Integer id;
    private ArrayList<Places> places = new ArrayList<>();
    private ArrayList<Places> filteredPlaces = new ArrayList<>();

    private OnItemClickListener clickListener;
```

```
    public void setOnItemClickListener(OnItemClickListener clickListener) {
        this.clickListener = clickListener;
    }

    // Constructor to receive the context
    public HorizontalAdapter(Context context, Integer id) {
        this.context = context;
        this.id = id;
    }

    public void updateData(ArrayList<Places> places) {
        this.places = places;
        this.filteredPlaces = new ArrayList<>(places);
        notifyDataSetChanged();
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.place_item_layout, parent, false);
        return new ViewHolder(view);
    }

    @Override
```

```
public void onBindViewHolder(ViewHolder holder, int position) {

    // Bind data or perform operations for each item

    Places p = filteredPlaces.get(position);

    if (id == 1){

        holder.btnFavPlace.setVisibility(View.GONE);

    }else{

        holder.btnFavPlace.setVisibility(View.VISIBLE);

        if (p.isFav()){

            holder.fav.setImageResource(R.drawable.ic_heart_red);

        }else{

            holder.fav.setImageResource(R.drawable.ic_heart);

        }

    }

    holder.name.setText(p.getName());

    holder.location.setText(p.getLocation());

    holder.rate.setText(p.getRate());

    Glide.with(context)

        .load(p.getImage())

        .into(holder.imgPlace);

}

holder.itemView.setOnClickListener(v -> {

    clickListener.onItemClick(p);

})
```

```
    });

}

@Override
public int getItemCount() {
    // Return the size of your data list
    // For example: return dataList.size();
    return filteredPlaces.size();
}

public class ViewHolder extends RecyclerView.ViewHolder {

    TextView name;
    TextView location;
    TextView rate;
    ImageView imgPlace;
    LinearLayout btnFavPlace;
    CardView cardView;
    ImageView fav;

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

        btnFavPlace = itemView.findViewById(R.id.btnFavPlace);
```

```
        name = itemView.findViewById(R.id.tvPlaceName);

        location = itemView.findViewById(R.id.tvLocation);

        rate = itemView.findViewById(R.id.tvRating);

        imgPlace = itemView.findViewById(R.id.ivPlaceImg);

        cardView = itemView.findViewById(R.id.cvPlace);

        fav = itemView.findViewById(R.id.fav);

        LinearLayout.LayoutParams layoutParams;

        if (id==1){

            layoutParams = new LinearLayout.LayoutParams (

                dpToPx(context, 250),

                LinearLayout.LayoutParams.WRAP_CONTENT);

        }else{

            layoutParams = new LinearLayout.LayoutParams (

                LinearLayout.LayoutParams.MATCH_PARENT,

                LinearLayout.LayoutParams.WRAP_CONTENT);

        }

        cardView.setLayoutParams(layoutParams);

    }

}

@Override

public Filter getFilter() {
```

```
    return new Filter() {

        @Override

            protected FilterResults performFiltering(CharSequence constraint) {

                String query = constraint.toString().toLowerCase().trim();

                List<Places> filtered = new ArrayList<>();

                if (query.isEmpty()) {

                    filtered.addAll(places);

                } else {

                    for (Places item : places) {

                        if (item.getName().toLowerCase().contains(query)) {

                            filtered.add(item);

                        }

                    }

                }

                FilterResults filterResults = new FilterResults();

                filterResults.values = filtered;

                return filterResults;

            }

        }

    }

    @Override
```

```
        protected void publishResults(CharSequence constraint,
FilterResults results) {
            filteredPlaces.clear();

            filteredPlaces.addAll((List<Places>) results.values);

            notifyDataSetChanged();
        }

    };

}

public interface OnItemClickListener {
    void onItemClick(Places place);
}

public static int dpToPx(Context context, int dp) {
    float density = context.getResources().getDisplayMetrics().density;
    return Math.round(dp * density);
}
```

Kode di atas merupakan implementasi dari RecyclerView Adapter untuk menampilkan daftar tempat wisata secara horizontal. Berikut adalah penjelasan kegunaan beberapa bagian utama dari kode tersebut:

#### Variabel dan Konstruktor:

- context: Menyimpan konteks dari Activity atau Fragment yang menggunakan adapter
- id: Menyimpan nilai integer yang digunakan untuk mengatur tata letak item di RecyclerView
- places: ArrayList yang berisi data tempat wisata
- filteredPlaces: ArrayList yang menyimpan data tempat wisata yang telah difilter
- clickListener: Interface untuk menangani klik pada item RecyclerView.

#### **Metode onCreateViewHolder:**

Membuat dan menginisialisasi ViewHolder untuk setiap item dalam RecyclerView. Menggunakan layout place\_item\_layout.xml sebagai tampilan item.

#### **Metode onBindViewHolder:**

Menghubungkan data dari objek Places ke ViewHolder. Menyembunyikan atau menampilkan tombol favorit berdasarkan nilai id. Menggunakan Glide untuk memuat gambar tempat wisata. Mengatur tindakan klik pada item menggunakan antarmuka OnItemClickListener.

#### **Metode getItemCount:**

Mengembalikan jumlah item dalam daftar, yaitu ukuran dari filteredPlaces.

#### **Kelas ViewHolder:**

Menyimpan referensi ke elemen UI dalam setiap item. Mengatur tata letak item berdasarkan nilai id.

#### **Metode getFilter:**

Mengimplementasikan metode Filterable untuk memungkinkan filtering data. Menggunakan performFiltering untuk memfilter data berdasarkan nama tempat. Menggunakan publishResults untuk memperbarui daftar yang ditampilkan setelah proses filter selesai.

#### **Antarmuka OnItemClickListener:**

Digunakan untuk menangani klik pada item RecyclerView.

### **Metode dpToPx:**

Mengonversi nilai dalam DP (density-independent pixel) ke piksel berdasarkan density perangkat.

Dengan menggunakan adapter ini, Anda dapat menampilkan daftar tempat wisata secara horizontal dengan kemampuan filter berdasarkan nama tempat. Adapter ini juga mendukung tata letak yang berbeda berdasarkan nilai id yang diteruskan pada konstruktornya.

### Kode Program class objek : Places

```
package com.android.travins.data.models;

import java.io.Serializable;

public class Places implements Serializable {

    protected String id;
    protected String name;
    protected String image;
    protected String description;
    protected String price;
    protected String location;
    protected String rate;
    protected String total_review;
    protected boolean isFav;

    public Places(String id, String name, String image, String description, String price, String location, String rate, String total_review) {
        this.id = id;
    }
}
```

```
    this.name = name;

    this.image = image;

    this.description = description;

    this.price = price;

    this.location = location;

    this.rate = rate;

    this.total_review = total_review;

    this.isFav = false;

}

public boolean isFav() {

    return isFav;

}

public void setFav(boolean fav) {

    isFav = fav;

}

public String getId() {

    return id;

}

public void setId(String id) {

    this.id = id;
```

```
}

public String getName() {
    return name;
}

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

public String getImage() {
    return image;
}

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

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}
```

```
}

public String getPrice() {
    return price;
}

public void setPrice(String price) {
    this.price = price;
}

public String getLocation() {
    return location;
}

public void setLocation(String location) {
    this.location = location;
}

public String getRate() {
    return rate;
}

public void setRate(String rate) {
    this.rate = rate;
}
```

```
}

public String getTotal_review() {
    return total_review;
}

public void setTotal_review(String total_review) {
    this.total_review = total_review;
}

}
```

Pada class ini di definisikan atribut dari kelas Places yaitu ada id, name, image, description, price, location, rate, total\_review, isFav. Atribut ini menggunakan method setter getter yang nantinya akan dihubungkan dengan server database.

#### Kode Program itemRow: SearchFragment

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/cvPlace"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardCornerRadius="12dp"
    app:cardElevation="12dp"
    android:layout_gravity="center"
    android:layout_margin="16dp"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="8dp">
```

```
<ImageView
    android:id="@+id/ivPlaceImg"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@drawable/bg_img_rounded"
    android:clipToOutline="true"
    android:scaleType="centerCrop"
    android:src="@drawable/img_raja_ampat"
    app:layout_constraintDimensionRatio="49:30"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<LinearLayout
    android:id="@+id/btnFavPlace"
    android:layout_width="36dp"
    android:layout_height="36dp"
    android:background="@drawable/bg_rounded_8"
    android:padding="6dp"
    android:layout_margin="8dp"
    android:backgroundTint="#B3FFFFFF"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <ImageView
        android:id="@+id/fav"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/ic_favourite" />

</LinearLayout>

<TextView
    android:id="@+id/tvPlaceName"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:fontFamily="@font/worksans_semiBold"
    android:text="Raja Ampat Islands"
    android:textColor="@color/textPrimary"
    android:layout_marginEnd="8dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/tvRating"

    app:layout_constraintTop_toBottomOf="@+id/ivPlaceImg" />
```

```
<TextView
    android:id="@+id/tvRating"
    android:layout_width="wrap_content"
    android:layout_height="15dp"
    android:drawableStart="@drawable/ic_star"
    android:drawablePadding="4dp"
    android:text="4.8"
    android:textSize="12sp"
    android:textColor="@color/textSecondary"
    android:fontFamily="@font/worksans_medium"
    android:layout_marginTop="8dp"
    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintBottom_toBottomOf="@+id/tvPlaceName"
    app:layout_constraintTop_toBottomOf="@+id/ivPlaceImg"/>

<TextView
    android:id="@+id/tvLocation"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp"
    android:text="Raja Ampat, Papua Barat"
    android:textSize="12sp"
    android:textColor="@color/textSecondary"
    android:fontFamily="@font/worksans_regular"
    android:drawableStart="@drawable/ic_location"
    android:drawablePadding="4dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@+id/tvPlaceName"/>

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```

Kode di atas merupakan item yang ditampilkan dari setiap rowView dari RecyclerView pada halaman search. Yaitu pada recyclerview destinasi. Seperti tertera di atas terdapat beberapa atribut seperti ImageView untuk gambar destinasi, TextView untuk informasi terkait destinasi dan rating, dan ada LinearLayout untuk menyusun atribut.

## Implementasi Fragment

Kode Program fragment : SearchFragment

```
package com.android.travins.ui.fragments;

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

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

import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.Toast;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.android.travins.ui.activities.PlaceDetailActivity;
import com.android.travins.ui.adapters.HorizontalAdapter;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class SearchFragment extends Fragment implements
HorizontalAdapter.OnItemClickListener {

    private FirebaseFirestore db;
```

```
private RecyclerView rvSearch;
private HorizontalAdapter adapter;
private ArrayList<Places> listDest = new ArrayList<>();
private EditText editTextSearch;
private FirebaseAuth auth;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
}

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

@Override
public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    db = FirebaseFirestore.getInstance();
    rvSearch = view.findViewById(R.id.rvListTrip);
    editTextSearch = view.findViewById(R.id.edtTxtSearch);
    listDest = new ArrayList<>();
    auth = FirebaseAuth.getInstance();
    fetchData();
    adapter = new HorizontalAdapter(this.getContext(), 2);
    adapter.setOnItemClickListener(this);

    rvSearch.setLayoutManager(new
        LinearLayoutManager(this.getContext(),
        LinearLayoutManager.VERTICAL, false));
    rvSearch.setAdapter(adapter);

    editTextSearch.addTextChangedListener(new TextWatcher()
    {
        @Override
        public void beforeTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
        }

        @Override
        public void onTextChanged(CharSequence
```

```

charSequence, int i, int i1, int i2) {
        adapter.getFilter().filter(charSequence);
    }

    @Override
    public void afterTextChanged(Editable editable) {
    }
} );

}

private void fetchData() {
    db.collection("places")
        .get()
            .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
        @Override
            public void onComplete(@NonNull
Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document
: task.getResult()) {
                        Map<String, Object> data =
document.getData();
                        Places places = new
Places(document.getId(),
                            (String)
data.get("name"), (String) data.get("image"),
                            (String)
data.get("description"), (String) data.get("price"),
                            (String)
data.get("location"), (String) data.get("rate"),
                            (String)
data.get("total_review"));

                        fetchFavData(places);
                }
            }
        } else {
            Toast.makeText(getApplicationContext(), "Fetch
data failed", Toast.LENGTH_SHORT).show();
        }
    });
}

```

```

private Places fetchFavData(Places places) {

    db.collection("users").document(auth.getCurrentUser().getUid())
        .collection("favorites")
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document :
                        task.getResult()) {
                        Map<String, Object> data =
                            document.getData();

                        if
                            (document.getId().equals(places.getId())){
                                places.setFav(true);
                            }
                            listDest.add(places);
                        }
                        adapter.updateData(listDest);
                    } else {
                    }
                });
            }

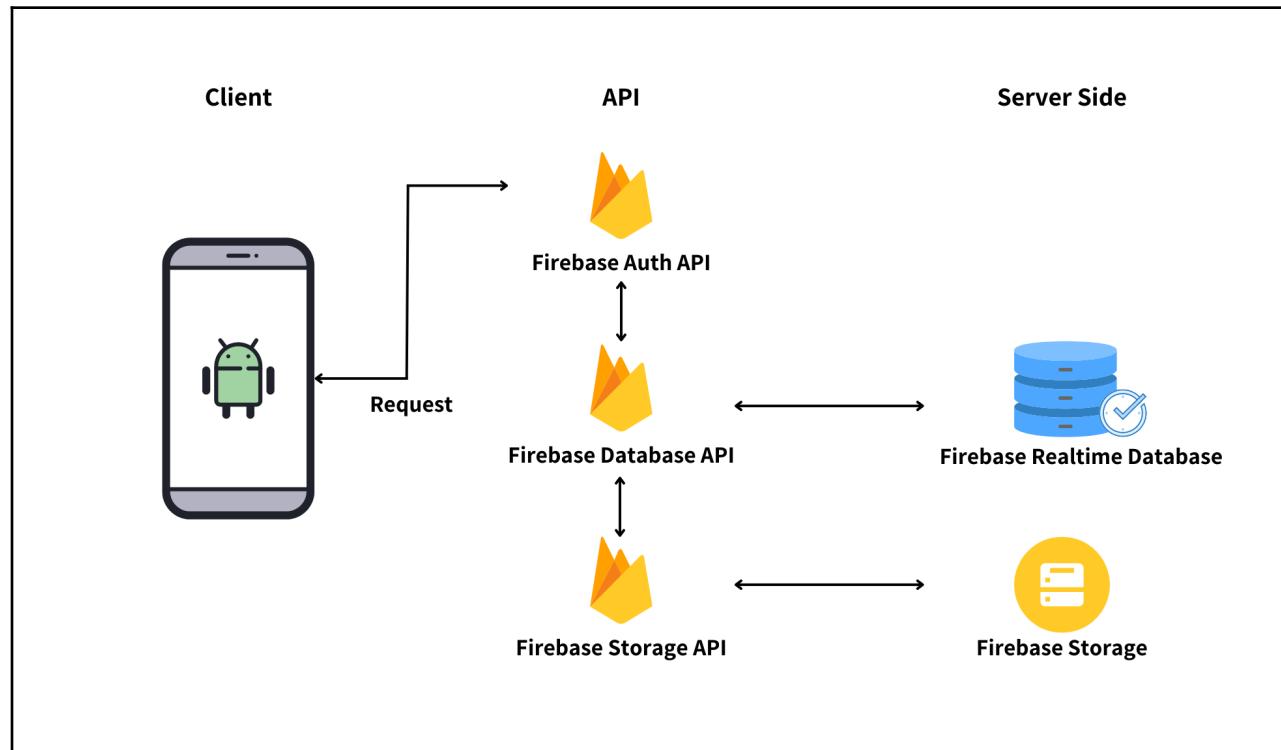
        return places;
    }

    @Override
    public void onItemClick(Places place) {
        Intent intent = new Intent(this.getActivity(),
        PlaceDetailActivity.class);
        intent.putExtra("place",place);
        startActivity(intent);
    }
}

```

Fungsi dari fragment “SearchFragment” sendiri kurang lebih sama seperti ListTripActivity di atas, hanya saja terdapat perbedaan di bagian extends public class yaitu pada SearchFragment menggunakan extends Fragment. Layout yang digunakan pada fragment ini yaitu fragment\_search.XML, penggunaan layout ini sendiri sudah dijelaskan pada bagian Layout 3:[fragment\_search] yang kurang lebih fungsinya yaitu untuk melakukan search/filter destinasi yang ada dalam RecyclerView.

## Implementasi Komunikasi Data



Gambar tersebut memberikan gambaran umum tentang cara kerja API. API adalah alat yang penting yang memungkinkan aplikasi berkomunikasi satu sama lain dan mengakses data dan layanan dari server. Dalam diagram tersebut, ada dua sisi utama API: sisi klien dan sisi server. Sisi klien adalah aplikasi yang menggunakan API. Aplikasi ini dapat berupa aplikasi web, aplikasi seluler, atau aplikasi desktop. Aplikasi klien biasanya menggunakan API untuk mengakses data atau layanan yang disediakan oleh server. Dalam diagram tersebut, sisi klien diwakili oleh ikon telepon Android. Telepon Android dapat menggunakan API Firebase untuk mengakses data dan layanan dari server Firebase.

Penjelasan:

Firestore Authentication: menyediakan berbagai fitur untuk mengelola autentikasi dan otorisasi pengguna, termasuk:

1. Registrasi pengguna
2. Masuk dengan akun Google atau email
3. Verifikasi email

4. Kelola akun pengguna
5. Firestore Database

Firestore Database adalah database real-time yang memungkinkan aplikasi klien untuk menyimpan dan mengambil data secara real-time. Data disimpan dalam format JSON dan dapat diakses dari aplikasi klien menggunakan API.

#### Firebase Storage API

Firebase Storage API adalah layanan penyimpanan file yang memungkinkan aplikasi klien untuk menyimpan file di cloud. File dapat diakses dari aplikasi klien menggunakan API.

# **Use Case 4: Read Destination Details & Recommendation Destination**

**Ananda Marsekalyana Rahmihadi**

NIM 215150407111016



## **Deskripsi Fungsional**

**Read Destination Details :**

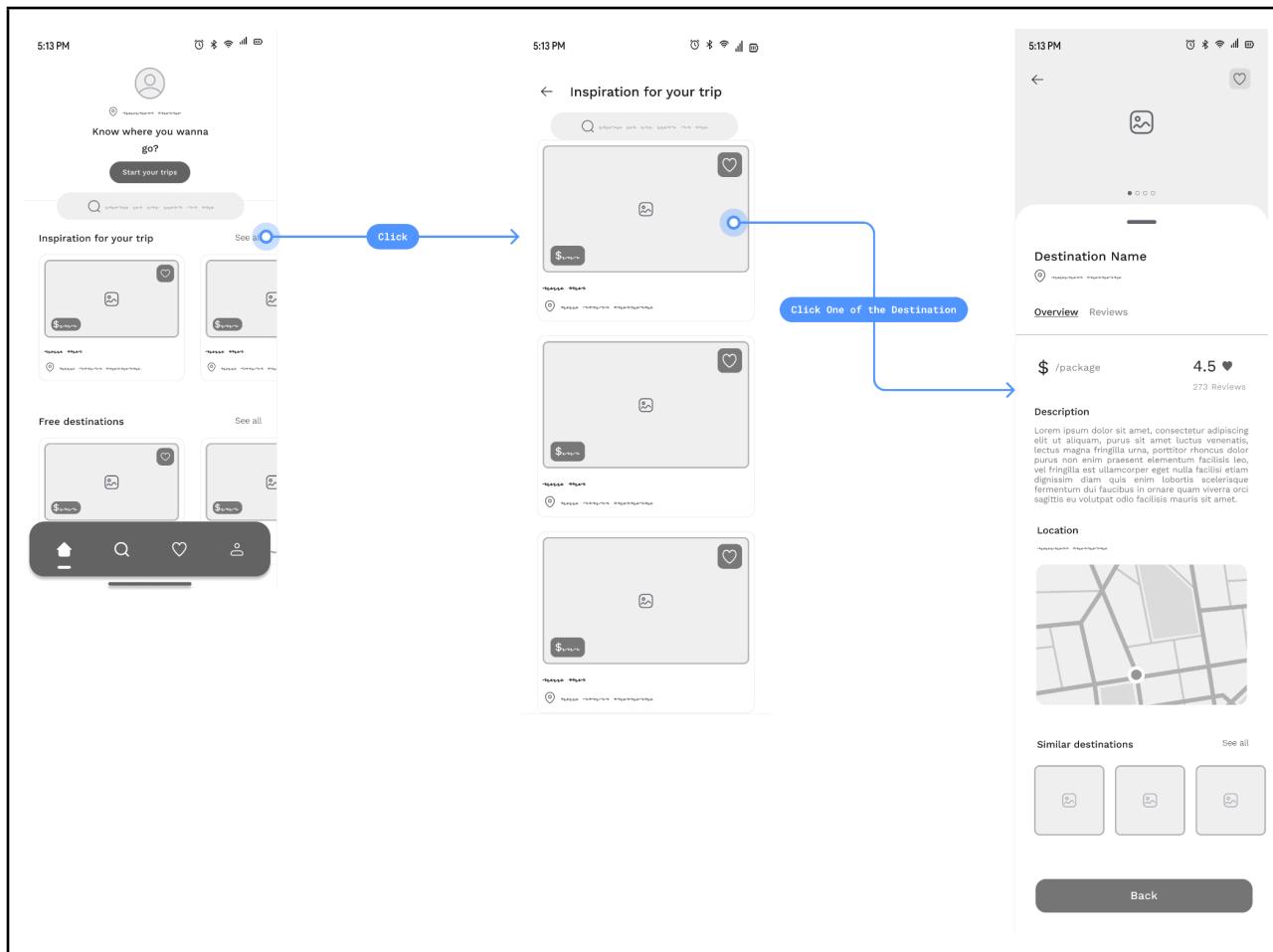
Pengguna dapat membaca informasi rinci atau detail terkait suatu tujuan wisata atau lokasi tertentu. Fitur ini memberikan pengguna akses untuk berbagai informasi yang dapat membantu mereka merencanakan perjalanan mereka dengan lebih baik.

**Recommended Destination :**

Pengguna dapat mendapat informasi mengenai jenis perjalanan yang diinginkan. Fitur ini memberikan rekomendasi destinasi wisata kepada pengguna berdasarkan preferensi, profil, dan minat mereka dalam aplikasi. Aplikasi menggunakan sistem cerdas dengan penggunaan algoritma untuk menganalisis data pengguna dan memberikan rekomendasi destinasi berdasarkan pola dan tren yang teridentifikasi.

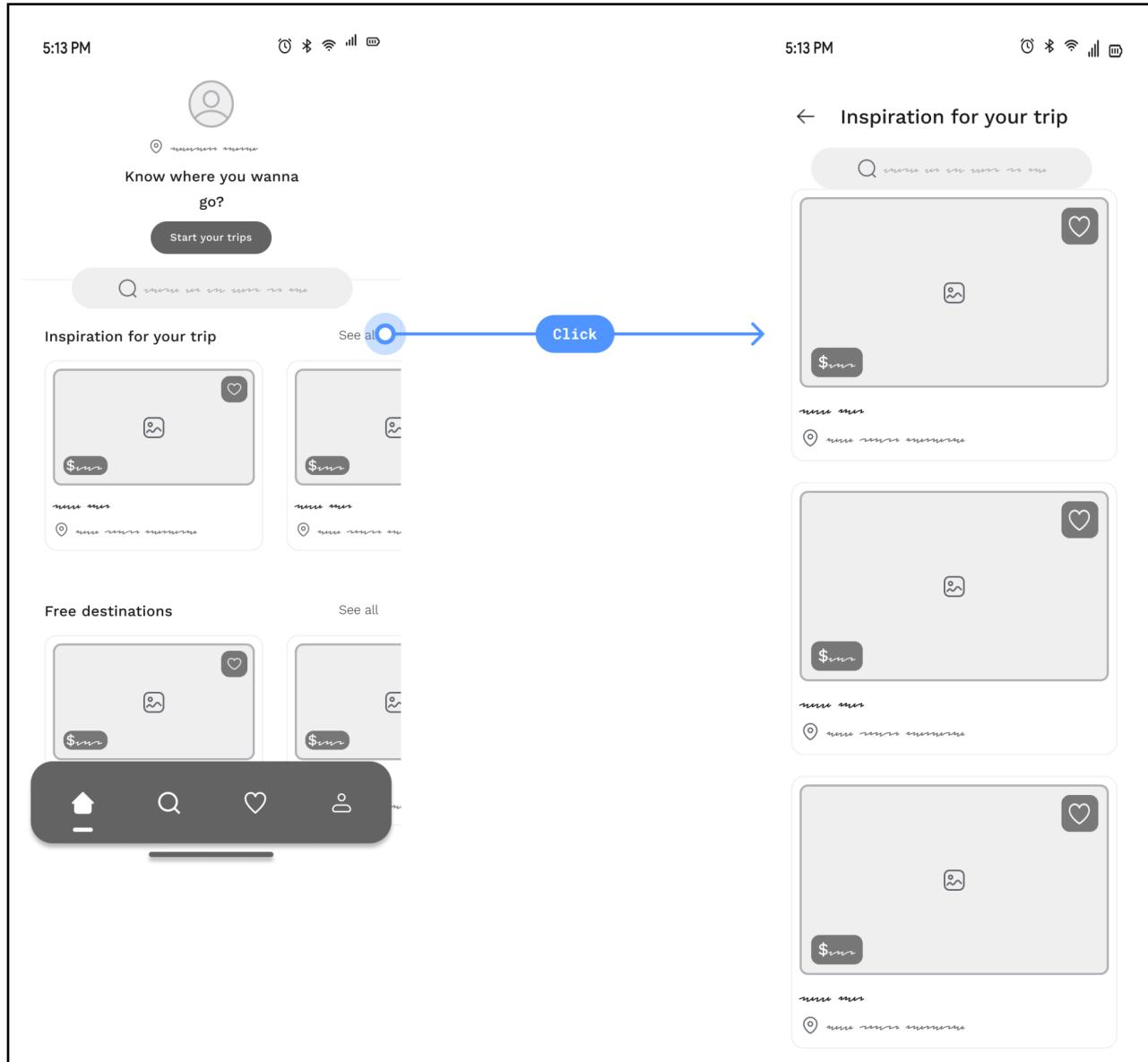
# Rancangan Screenflow

## 1. Read Destination Details



Pengguna membuka aplikasi Travellns dan masuk ke halaman utama atau homepage. Pengguna memilih opsi "Inspiration for your trip" yang dapat membawa mereka ke halaman dengan destinasi yang direkomendasikan. Selanjutnya pengguna memilih destinasi tertentu yang menarik perhatian mereka. Pengguna diarahkan ke halaman detail destinasi yang telah dipilih. Halaman ini dapat mencakup, deskripsi singkat destinasi, gambar-gambar destinasi, informasi mengenai destinasi lainnya yang mirip destinasi tersebut.

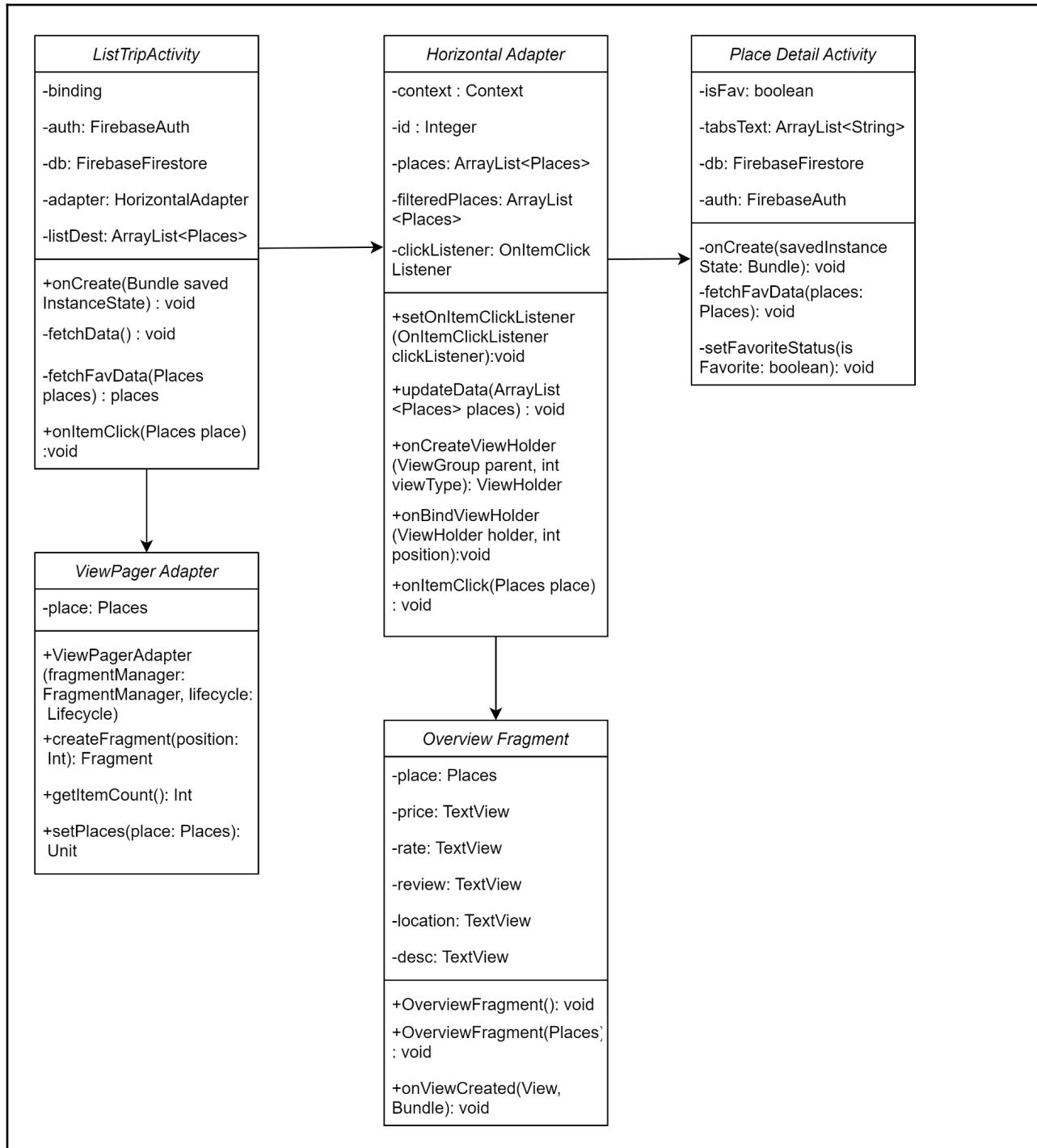
## 2. Read Recommendation Destination



Pengguna membuka aplikasi TravellIns dan masuk ke halaman utama atau homepage. Pengguna melihat berbagai fitur dan opsi yang tersedia, dan memilih navigasi fitur "Inspiration for your trip" atau "Recommended Destinations". Sistem menampilkan daftar destinasi atau trip yang direkomendasikan berdasarkan preferensi, riwayat perjalanan, atau algoritma rekomendasi. Pengguna melihat daftar destinasi yang direkomendasikan beserta rating.

# Class Diagram

Gambarkan HANYA Class Diagram yang terlibat dan digunakan untuk menyelesaikan fungsionalitas 1 di sini.



Jelaskan fungsi dari setiap class yang tergambar dalam Class Diagram

ListTripActivity memiliki beberapa atribut dan metod seperti:

- `onCreate(Bundle)`: Menginisialisasi aktivitas, mengatur adapter, dan mengambil data tempat perjalanan.
- `fetchData()`: Mengambil data tempat perjalanan dari Firebase Firestore.
- `fetchFavData(Places)`: Mengambil data favorit pengguna dari Firebase Firestore berdasarkan tempat perjalanan.
- `onItemClick(Places)`: Menangani klik pada item dalam RecyclerView.

Penjelasan singkat beberapa method dari Kelas HorizontalAdapter

- `onCreateViewHolder(ViewGroup, int)`: Membuat ViewHolder untuk setiap item dalam RecyclerView.
- `onBindViewHolder(ViewHolder, int)`: Mengikat data ke ViewHolder.
- `getItemCount()`: Mendapatkan jumlah item dalam RecyclerView.
- `getFilter()`: Mendapatkan objek Filter untuk melakukan pencarian.
- `updateData(ArrayList<Places>)`: Memperbarui data tempat perjalanan dan RecyclerView.

Penjelasan singkat beberapa method dari Kelas ViewPagerAdapter:

- `createFragment(int)`: Membuat fragmen sesuai dengan posisi dalam ViewPager.
- `getCount()`: Mendapatkan jumlah fragmen.
- `setPlaces(Places)`: Mengatur tempat perjalanan yang dipilih.

Penjelasan singkat beberapa method dari Kelas PlaceDetailActivity:

- `onCreate(Bundle)`: Menginisialisasi aktivitas, mengatur adapter ViewPager, dan mengambil data tempat perjalanan.
- `fetchFavData(Places)`: Mengambil status favorit tempat perjalanan dari Firebase Firestore.

Penjelasan singkat beberapa method dari Kelas OverviewFragment:

- `onCreateView(LayoutInflater, ViewGroup, Bundle)`: Membuat tampilan fragment.
- `onViewCreated(View, Bundle)`: Dipanggil setelah tampilan fragment telah dibuat untuk mengatur tampilan dengan data tempat perjalanan.

# Android Manifest

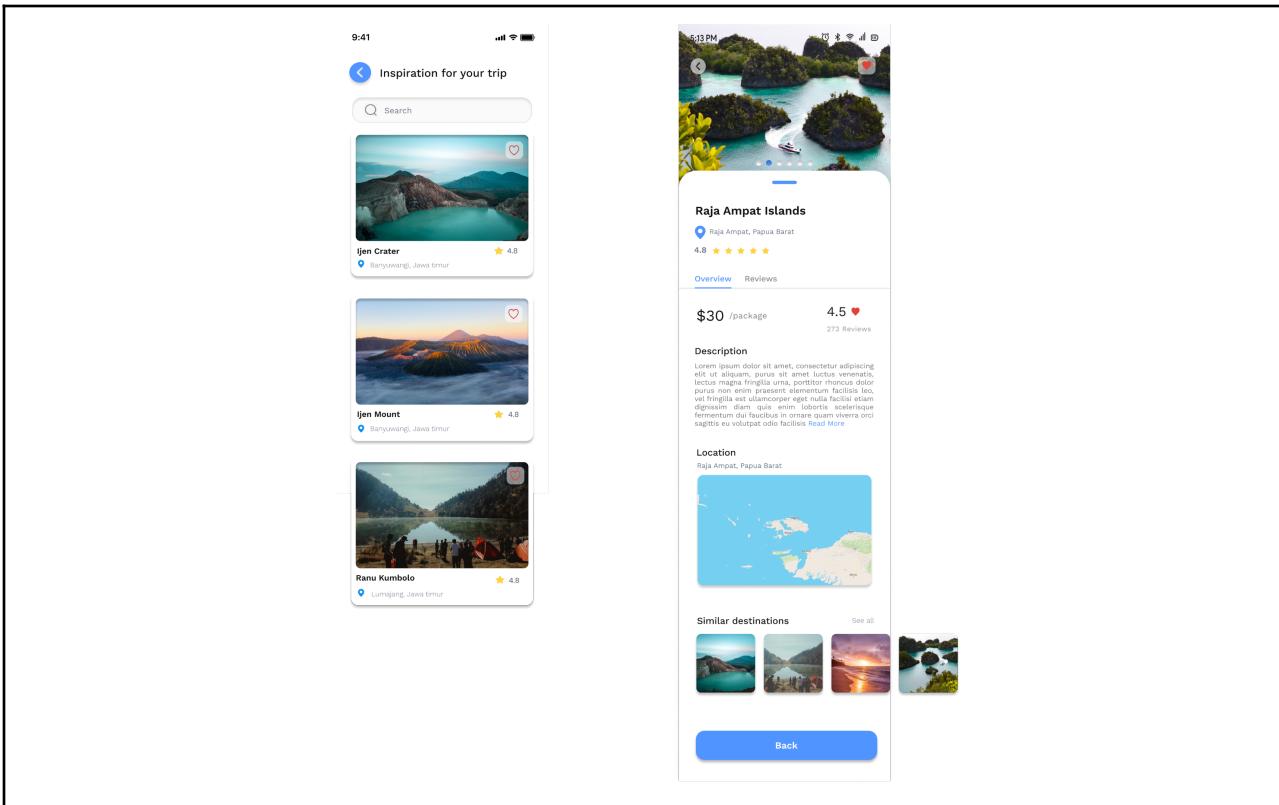
```
<application
    <activity
        android:name=".ui.activities.PlaceDetailActivity"
        android:exported="false"
        android:windowSoftInputMode="adjustPan"/>
    ...
    <activity
        android:name=".ui.activities.ListTripActivity"
        android:exported="false" />
</application>
```

Tuliskan isi dari file AndroidManifest.xml di sini.

Ada dua aktivitas yang didefinisikan: PlaceDetailActivity dan ListTripActivity.

- PlaceDetailActivity: Aktivitas ini mewakili layar detail tempat. Aktivitas ini tidak diizinkan untuk diekspor kepada aplikasi lain (android:exported="false"), dan tata letaknya disesuaikan (android:windowSoftInputMode="adjustPan")
- ListTripActivity: Aktivitas ini mewakili layar daftar perjalanan. Aktivitas ini juga tidak diizinkan diekspor dan tidak memiliki konfigurasi tata letak khusus.

# Implementasi UI



Pada fungsional read recommendation destination pengguna dapat melihat daftar destinasi yang direkomendasikan setelah sistem menampilkan daftar destinasi atau trip yang direkomendasikan berdasarkan preferensi pengguna.

Pada fungsional detail place destination pengguna dapat melihat deskripsi singkat destinasi dan gambar destinasi yang telah dipilih, serta terdapat daftar tampilan destinasi lainnya yang mirip dengan destinasi tersebut. Deskripsi singkat menjelaskan informasi mengenai lokasi, informasi-informasi menarik yang ada pada destinasi.

## Layout 1: [activity\_list\_trip]

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.activities.ListTripActivity"
    android:paddingHorizontal="24dp"
    android:orientation="vertical">

    <LinearLayout
        android:id="@+id/btnBack"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingVertical="16dp">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_back"
            android:paddingStart="0dp"/>

        <TextView
            android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Inspiration for your trip"
        android:textColor="@color/black"
        android:textSize="22sp"
        android:textAlignment="textStart"
        android:paddingStart="16dp"
        android:fontFamily="@font/worksans_regular"/>
    
```

```
</LinearLayout>

<EditText
    android:id="@+id edtTxtSearch"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Search"
    android:drawableTint="@color/black"
    android:drawablePadding="8dp"
    android:drawableStart="@drawable/ic_search"
    android:background="@drawable/bg_white_rounded"
    android:padding="10dp"
    android:layout_marginBottom="16dp"/>

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvListInspiration"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"/>

    </LinearLayout>
```

Layout ini digunakan untuk membuat antarmuka pengguna dari aktivitas ListTripActivity dengan elemen-elemen seperti judul, kotak pencarian, dan daftar inspirasi perjalanan. RecyclerView (rvListInspiration) untuk menampilkan daftar inspirasi perjalanan dalam bentuk RecyclerView.

## Layout 2: [activity\_place\_detail]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.activities.PlaceDetailActivity"
    android:orientation="vertical">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageView
            android:id="@+id/ivDetailPlace"
            android:layout_width="match_parent"
            android:layout_height="280dp"
            android:scaleType="center"
            android:src="@drawable/img_raja_ampat"/>

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="16dp">
            <ImageView
                android:id="@+id/btnBack"
```

```
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:src="@drawable/ic_back_white"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"/>
    <ImageView
        android:id="@+id	btnFav"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:src="@drawable/ic_heart"
        android:padding="8dp"

    android:background="@drawable/bg_white_transparent_rounded"/>
        </androidx.constraintlayout.widget.ConstraintLayout>
    </FrameLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="-50dp"
    android:orientation="vertical"

    android:background="@drawable/bg_white_top_left_right_rounded">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="12dp">
        <TextView
            android:id="@+id/tvNamePlace"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Raja Ampat Islands"
            android:fontFamily="@font/worksans_semiBold"
            android:textColor="@color/black"
            android:textSize="18sp"/>
        <TextView
            android:id="@+id/tvLocationPlace"
            android:paddingTop="8dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Raja Ampat, Papua Barat"
            android:drawablePadding="8dp"
```

```
        android:textColor="@color/grey"
        android:fontFamily="@font/worksans_regular"
        android:drawableStart="@drawable/ic_location"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingTop="8dp"
        android:layout_gravity="center_vertical">
        <TextView
            android:id="@+id/tvRate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="4.8"
            android:textSize="14sp"
            android:textColor="@color/grey"

        android:fontFamily="@font/worksans_semiibold"/>
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:paddingStart="10dp">
            <ImageView
                android:layout_width="16dp"
                android:layout_height="16dp"
                android:src="@drawable/ic_star"/>
            <ImageView
                android:layout_width="16dp"
                android:layout_height="16dp"
                android:src="@drawable/ic_star"/>
            <ImageView
                android:layout_width="16dp"
                android:layout_height="16dp"
                android:src="@drawable/ic_star"/>
            <ImageView
                android:layout_width="16dp"
                android:layout_height="16dp"
                android:src="@drawable/ic_star"/>
            <ImageView
                android:layout_width="16dp"
                android:layout_height="16dp"
                android:src="@drawable/ic_star"/>
        </LinearLayout>
    </LinearLayout>
```

```
</LinearLayout>

<com.google.android.material.tabs.TabLayout
    android:id="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:tabGravity="fill"
    app:tabMode="fixed"/>

<androidx.viewpager2.widget.ViewPager2
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

</LinearLayout>

</LinearLayout>
```

Layout ini menciptakan antarmuka pengguna yang terdiri dari header dengan gambar latar belakang dan ikon kembali serta ikon favorit di bagian atas, diikuti oleh informasi detail tempat dan bagian tab untuk menampilkan konten gambar destinasi yang berbeda-beda sesuai dengan tujuan destinasi.

Tampilan Screenshot layout

TravIns - activity\_list\_trip.xml [TravIns.app.main]

```
<LinearLayout
    android:id="@+id/btnBack"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:paddingVertical="16dp">
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_back"
    android:paddingStart="0dp"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Inspiration for your trip"
    android:textColor="@color/black"
    android:padding="16dp"/>
```

Running Devices: Pixel 3a API 34 extension



Notifications Device Manager Grade Layout Validation Running Devices

TravIns - activity\_place\_detail.xml [TravIns.app.main]

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
<ImageView
    android:id="@+id/ivDetailPlace"
    android:layout_width="match_parent"
    android:layout_height="280dp"
    android:scaleType="center"
    android:src="@drawable/img_raja_ampat"/>
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp">
<ImageView
    android:id="@+id	btnBack"
    android:layout_width="30dp"/>
```

Running Devices: Pixel 3a API 34 extension



Notifications Device Manager Grade Layout Validation Running Devices

# Implementasi Kode Program Aplikasi

Tuliskan kode program Activity/Fragment yang terlibat untuk menyelesaikan fungsional yang dijelaskan. Serta jelaskan bagaimana cara kerja dari kode program yang digunakan tersebut. Penjelasan tidak dilakukan secara baris-per-baris tapi dijelaskan dengan mencuplik kode program yang sedang dijelaskan.

## 4. Implementasi Adapter

Kode Program adapter : ViewPagerActivity

```
package com.android.travins.ui.adapters;

import com.android.travins.data.models.Places;
import com.android.travins.ui.fragments.OverviewFragment;
import com.android.travins.ui.fragments.ReviewFragment;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.lifecycle.Lifecycle;
import androidx.viewpager2.adapter.FragmentStateAdapter;

public class ViewPagerAdapter extends FragmentStateAdapter {

    private Places place;

    public ViewPagerAdapter(@NonNull FragmentManager fragmentManager, @NonNull Lifecycle lifecycle) {
        super(fragmentManager, lifecycle);
    }

    @NonNull
    @Override
    public Fragment createFragment(int position) {
        switch (position) {
            case 0:
                return new OverviewFragment(place);
            case 1:
                return new ReviewFragment(place);
            default:
                return null;
        }
    }
}
```

```
@Override  
public int getItemCount() {  
    return 2;  
}  
  
public void setPlaces (Places place){  
    this.place = place;  
}  
}
```

Pengaturan Objek Places: Objek Places yang akan digunakan oleh fragmen-fragmen diatur menggunakan metode setPlaces.

Fungsi createFragment: Dalam metode ini, ViewPager2 meminta adapter untuk membuat fragmen berdasarkan posisi saat ini.

Fungsi getItemCount: ViewPager2 menggunakan metode ini untuk mengetahui jumlah total item (fragmen) yang akan ditampilkan.

Kelas ViewPagerAdapter ini dirancang untuk digunakan dalam pengaturan ViewPager2 yang memiliki dua fragmen, yaitu OverviewFragment dan ReviewFragment. Masing-masing fragmen akan menampilkan informasi berbeda tentang objek Places yang diatur melalui metode setPlaces.

Kode Program activity : ListTripActivity

```
package com.android.travins.ui.activities;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.widget.Toast;  
  
import com.android.travins.R;  
import com.android.travins.data.models.Places;  
import com.android.travins.databinding.ActivityListTripBinding;  
import com.android.travins.ui.adapters.HorizontalAdapter;
```

```
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firestore.QueryDocumentSnapshot;
import com.google.firebaseio.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class ListTripActivity extends AppCompatActivity
    implements HorizontalAdapter.OnItemClickListener {

    private ActivityListTripBinding binding;
    private FirebaseAuth auth;
    private FirebaseFirestore db;
    private HorizontalAdapter adapter;
    private ArrayList<Places> listDest = new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityListTripBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        listDest = new ArrayList<>();
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        adapter = new HorizontalAdapter(this,2);
        adapter.setOnItemClickListener(this);
        fetchData();

        binding.btnExit.setOnClickListener(v -> finish());

        binding.rvListInspiration.setLayoutManager(new
            LinearLayoutManager(this, LinearLayoutManager.VERTICAL,
            false));
        binding.rvListInspiration.setAdapter(adapter);

        binding.edtTxtSearch.addTextChangedListener(new
            TextWatcher() {
            @Override
                public void beforeTextChanged(CharSequence
                    charSequence, int i, int i1, int i2) {
            }
        }
    }
}
```

```
        @Override
            public void onTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
                adapter.getFilter().filter(charSequence);
            }

        @Override
        public void afterTextChanged(Editable editable) {
        }
    });

private void fetchData() {
    db.collection("places")
        .get()
        .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
        @Override
            public void onComplete(@NonNull
Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document
: task.getResult()) {
                        Map<String, Object> data =
document.getData();
                        Places places = new
Places(document.getId(),
                            data.get("name"), (String)
data.get("image"),
                            (String)
data.get("description"), (String) data.get("price"),
                            (String)
data.get("location"), (String) data.get("rate"),
                            (String)
data.get("total_review"));

                        fetchFavData(places);
                    }
                } else {
                    Toast.makeText(ListTripActivity.this, "Fetch
failed", Toast.LENGTH_SHORT).show();
                }
            }
        });
}
```

```

    }
    private Places fetchFavData(Places places) {
        db.collection("users").document(auth.getCurrentUser().getUid())
            .collection("favorites")
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document :
                        task.getResult()) {
                        Map<String, Object> data =
                            document.getData();
                        if (document.getId() .equals(places.getId())){
                            places.setFav(true);
                        }
                        listDest.add(places);
                    }
                    adapter.updateData(listDest);
                } else {
                }
            });
        return places;
    }

    @Override
    public void onItemClick(Places place) {
        Intent intent = new Intent(ListTripActivity.this,
            PlaceDetailActivity.class);
        intent.putExtra("place",place);
        startActivity(intent);
    }
}

```

Kode ini menggambarkan aktivitas yang menampilkan daftar tempat wisata dalam RecyclerView, dengan kemampuan pencarian dan menanggapi klik pada setiap item untuk menampilkan detail tempat wisata. Data tempat wisata diambil dari Firebase Firestore dan ditampilkan menggunakan HorizontalAdapter.

Kode Program activity : PlaceDetailActivity

```
package com.android.travins.ui.activities;
```

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

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

import com.android.travins.R;
import com.android.travins.data.models.Places;
import
com.android.travins.databinding.ActivityPlaceDetailBinding;
import com.android.travins.ui.adapters.ViewPagerAdapter;
import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.tabs.TabLayoutMediator;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firebaseio.QueryDocumentSnapshot;
import com.google.firebaseio.firebaseio.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class PlaceDetailActivity extends AppCompatActivity {

    private ActivityPlaceDetailBinding binding;
    private boolean isFav;
    private ArrayList<String> tabsText = new ArrayList<>();

    private FirebaseFirestore db;
    private FirebaseAuth auth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityPlaceDetailBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        Places places = (Places)
getIntent().getSerializableExtra("place");
        if (places!=null){
            binding.tvNamePlace.setText(places.getName());
            binding.tvLocationPlace.setText(places.getLocation());
        }
    }
}
```

```
        binding.tvRate.setText(places.getRate());
        Glide.with(this)
            .load(places.getImage())
            .into(binding.ivDetailPlace);
    fetchFavData(places);
}

tabsText.add("Overview");
tabsText.add("Review");

        ViewPagerAdapter adapter = new
ViewPagerAdapter(getSupportFragmentManager(),getLifecycle());
        adapter.setPlaces(places);

        binding.pager.setAdapter(adapter);
        TabLayoutMediator tabLayoutMediator = new
TabLayoutMediator(binding.tabLayout,binding.pager,(tab,
position) ->
        tab.setText(tabsText.get(position))
    );
    tabLayoutMediator.attach();

    binding.btnBack.setOnClickListener(v -> {
        finish();
    });

    binding.btnFav.setOnClickListener(v -> {
        if (isFav) {

db.collection("users").document(auth.getCurrentUser().getUid())
    .collection("favorites").document(places.getId()).delete().addOnCompleteListener(task -> {
        if (task.isSuccessful()){
            Toast.makeText(this, "Removed from
favorite.", Toast.LENGTH_SHORT).show();
            isFav = false;
        }
    });
} else{

db.collection("users").document(auth.getCurrentUser().getUid())
    .collection("favorites").document(places.getId()).set(places).addOnCompleteListener(task -> {
```

```

        if (task.isSuccessful()) {
            Toast.makeText(this, "Added to
favorite.", Toast.LENGTH_SHORT).show();
            isFav = true;
        }

        binding.btnAdd.setOnClickListener(v -> {
            if (!isFav) {
                db.collection("users").document(auth.getCurrentUser().getUid())
                    .collection("favorites")
                    .add(new DocumentSnapshot());
                isFav = true;
            } else {
                db.collection("users").document(auth.getCurrentUser().getUid())
                    .collection("favorites")
                    .document(task.getId()).delete();
                isFav = false;
            }
        });
    }

    private void fetchFavData(Places places) {
        db.collection("users").document(auth.getCurrentUser().getUid())
            .collection("favorites")
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document :
                        task.getResult()) {
                        Map<String, Object> data =
                            document.getData();

                        if (document.getId().equals(places.getId())){
                            isFav = true;
                        }

                        binding.btnAdd.setOnClickListener(v -> {
                            if (!isFav) {
                                db.collection("users").document(auth.getCurrentUser().getUid())
                                    .collection("favorites")
                                    .add(new DocumentSnapshot());
                                isFav = true;
                            } else {
                                db.collection("users").document(auth.getCurrentUser().getUid())
                                    .collection("favorites")
                                    .document(task.getId()).delete();
                                isFav = false;
                            }
                        });
                    }
                }
            });
    }
}

```

Kode ini menyusun aktivitas yang menampilkan detail tempat wisata, memanfaatkan TabLayout dan ViewPager untuk menampilkan konten yang terbagi ke dalam beberapa tab. Selain itu, disediakan fitur untuk pengguna agar dapat menandai tempat wisata sebagai favorit.

Kode fragment : OverviewFragment

```
package com.android.travins.ui.fragments;
```

```
import android.os.Bundle;

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

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.android.travins.R;
import com.android.travins.data.models.Places;

/
 * A simple {@link Fragment} subclass.
 * Use the {@link OverviewFragment#newInstance} factory method
 * to
 * create an instance of this fragment.
 */
public class OverviewFragment extends Fragment {

    private Places place;
    private TextView price;
    private TextView rate;
    private TextView review;
    private TextView location;
    private TextView desc;

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

    public OverviewFragment(Places place) {
        this.place = place;
    }

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

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
```

```

        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_overview,
container, false);
}

@Override
public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    price = view.findViewById(R.id.tvPrice);
    rate = view.findViewById(R.id.tvRate);
    review = view.findViewById(R.id.tvReviews);
    desc = view.findViewById(R.id.tvDesc);
    location = view.findViewById(R.id.tvLocation);

    if (place!=null){
        price.setText(place.getPrice());
        rate.setText(place.getRate());
        review.setText(place.getTotal_review()+" reviews");
        desc.setText(place.getDescription());
        location.setText(place.getLocation());
    }
}

```

Kode ini merupakan implementasi dari fragment yang berfungsi untuk menampilkan ringkasan atau overview dari suatu tempat wisata. Informasi tersebut diambil dari objek Places yang diterima melalui konstruktor dan ditampilkan melalui TextViews pada tampilan fragment. Fragment ini dapat digunakan dalam sebuah ViewPager atau dalam suatu aktivitas yang menampilkan detail tempat wisata.

## 5. Implementasi ListView/AdapterView/RecyclerView

Tuliskan dan jelaskan kode program dalam Activity/Fragment yang melibatkan elemen ListView/AdapterView/RecyclerView.

```

package com.android.travins.ui.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;

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

```

```
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.Toast;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.android.travins.databinding.ActivityListTripBinding;
import com.android.travins.ui.adapters.HorizontalAdapter;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firebaseio.QueryDocumentSnapshot;
import com.google.firebaseio.firebaseio.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class ListTripActivity extends AppCompatActivity implements
HorizontalAdapter.OnItemClickListener {

    private ActivityListTripBinding binding;
    private FirebaseAuth auth;
    private FirebaseFirestore db;
    private HorizontalAdapter adapter;
    private ArrayList<Places> listDest = new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityListTripBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        listDest = new ArrayList<>();
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        adapter = new HorizontalAdapter(this,2);
        adapter.setOnItemClickListener(this);
        fetchData();

        binding.btnExit.setOnClickListener(v -> finish());

        binding.rvListInspiration.setLayoutManager(new
LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false));
        binding.rvListInspiration.setAdapter(adapter);

        binding.edtTxtSearch.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i,
```

```

int i1, int i2) {
    adapter.getFilter().filter(charSequence);
}

@Override
public void afterTextChanged(Editable editable) {
}
});

}

private void fetchData() {
    db.collection("places")
        .get()
        .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
    @Override
        public void onComplete(@NonNull Task<QuerySnapshot>
task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document :
task.getResult()) {
                    Map<String, Object> data =
document.getData();
                    Places places = new
Places(document.getId(),
                            (String) data.get("name"),
                            (String)
data.get("image"),
                            (String)
data.get("description"),
                            (String) data.get("price"),
                            (String) data.get("location"),
                            (String)
data.get("rate"),
                            (String)
data.get("total_review"));

                    fetchFavData(places);
                }
            } else {
                Toast.makeText(ListTripActivity.this,"Fetch
data failed",Toast.LENGTH_SHORT).show();
            }
        }
    });
}

private Places fetchFavData(Places places) {

db.collection("users").document(auth.getCurrentUser().getUid()).collecti
on("favorites")
    .get()
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document :
task.getResult()) {

```

```
Map<String, Object> data =  
document.getData();  
  
if  
(document.getId().equals(places.getId())) {  
    places.setFav(true);  
}  
listDest.add(places);  
}  
adapter.updateData(listDest);  
} else {  
}  
});  
  
return places;  
}  
  
@Override  
public void onItemClick(Places place) {  
    Intent intent = new Intent(ListTripActivity.this,  
PlaceDetailActivity.class);  
    intent.putExtra("place", place);  
    startActivity(intent);  
}  
}
```

Pada kode program diatas menggunakan RecyclerView yang fungsinya digunakan untuk menampilkan daftar item secara efisien dalam antarmuka pengguna menggantikan penggunaan ListView atau GridView karena lebih fleksibel dan efisien untuk menangani daftar besar. Adapter adalah objek dari kelas HorizontalAdapter dikonfigurasi untuk menampilkan daftar item. Kode program di atas juga mencakup integrasi dengan Firebase (Firestore dan Authentication) untuk mengambil dan menyimpan data. Penggunaan RecyclerView dan HorizontalAdapter memberikan kontrol yang baik atas tata letak dan kinerja dalam menampilkan daftar item.

[Tuliskan penjelasan kode program Adapter yang digunakan]

```
public class ListTripActivity extends AppCompatActivity implements  
HorizontalAdapter.OnItemClickListener {  
  
    private ActivityListTripBinding binding;  
    private FirebaseAuth auth;  
    private FirebaseFirestore db;  
    private HorizontalAdapter adapter;  
    private ArrayList<Places> listDest = new ArrayList<>();  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding = ActivityListTripBinding.inflate(getLayoutInflater());  
        setContentView(binding.getRoot());
```

```
listDest = new ArrayList<>();
db = FirebaseFirestore.getInstance();
auth = FirebaseAuth.getInstance();

adapter = new HorizontalAdapter(this,2);
adapter.setOnItemClickListener(this);
fetchData();
...
});
}
```

## 1. Adapter (HorizontalAdapter):

Fungsi:

- Mengimplementasikan RecyclerView.Adapter untuk mengelola tampilan item dalam daftar horizontal.
- Menangani tampilan setiap item dalam daftar.

Atribut:

- ArrayList<Places> data: Menyimpan data tempat yang akan ditampilkan.
- ArrayList<Places> filteredData: Menyimpan data yang sesuai dengan filter pencarian.
- OnItemClickListener listener: Mendengarkan klik pada item.
- int viewType: Menyimpan jenis tampilan yang akan ditampilkan.

Metode Utama:

- onCreateViewHolder(): Membuat ViewHolder untuk setiap item.
- onBindViewHolder(): Menghubungkan data dengan tampilan setiap item.
- getItemCount(): Mengembalikan jumlah item dalam daftar.

[Tuliskan penjelasan kode program class Objek yang digunakan]

```
public class Places {
    // ...
}
```

## 2. Class Objek (Places):

Fungsi:

Merepresentasikan objek tempat wisata dengan atribut seperti nama, gambar, deskripsi, harga, lokasi, rating, dan jumlah ulasan.

Atribut:

Atribut-atribut yang mencerminkan karakteristik tempat wisata.



## **6. Implementasi Fragment**

Tuliskan dan jelaskan kode program Fragment yang digunakan untuk menyelesaikan fungsionalitas yang sedang diuraikan.

```
package com.android.travins.ui.fragments;
public class OverviewFragment extends Fragment {

    private Places place;
    private TextView price;
    private TextView rate;
    private TextView review;
    private TextView location;
    private TextView desc;

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

    public OverviewFragment(Places place) {
        this.place = place;
    }

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

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

    @Override
    public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        price = view.findViewById(R.id.tvPrice);
        rate = view.findViewById(R.id.tvRate);
        review = view.findViewById(R.id.tvReviews);
        desc = view.findViewById(R.id.tvDesc);
        location = view.findViewById(R.id.tvLocation);

        if (place!=null) {
```

```
        price.setText(place.getPrice());
        rate.setText(place.getRate());
        review.setText(place.getTotal_review()+" reviews");
        desc.setText(place.getDescription());
        location.setText(place.getLocation());
    }
}
```

Tuliskan dan jelaskan kode program dalam Activity/XML yang memuat/menjalankan Fragment ini.

1. Variabel:

Places place: Menyimpan objek Places yang akan ditampilkan.

TextView price, rate, review, desc, location: Variabel untuk menampilkan informasi harga, rating, jumlah ulasan, deskripsi, dan lokasi tempat wisata.

2. Konstruktor:

Konstruktor kosong diperlukan oleh Fragment. Konstruktor lain menerima objek Places sebagai parameter untuk ditampilkan.

3. onCreate Method:

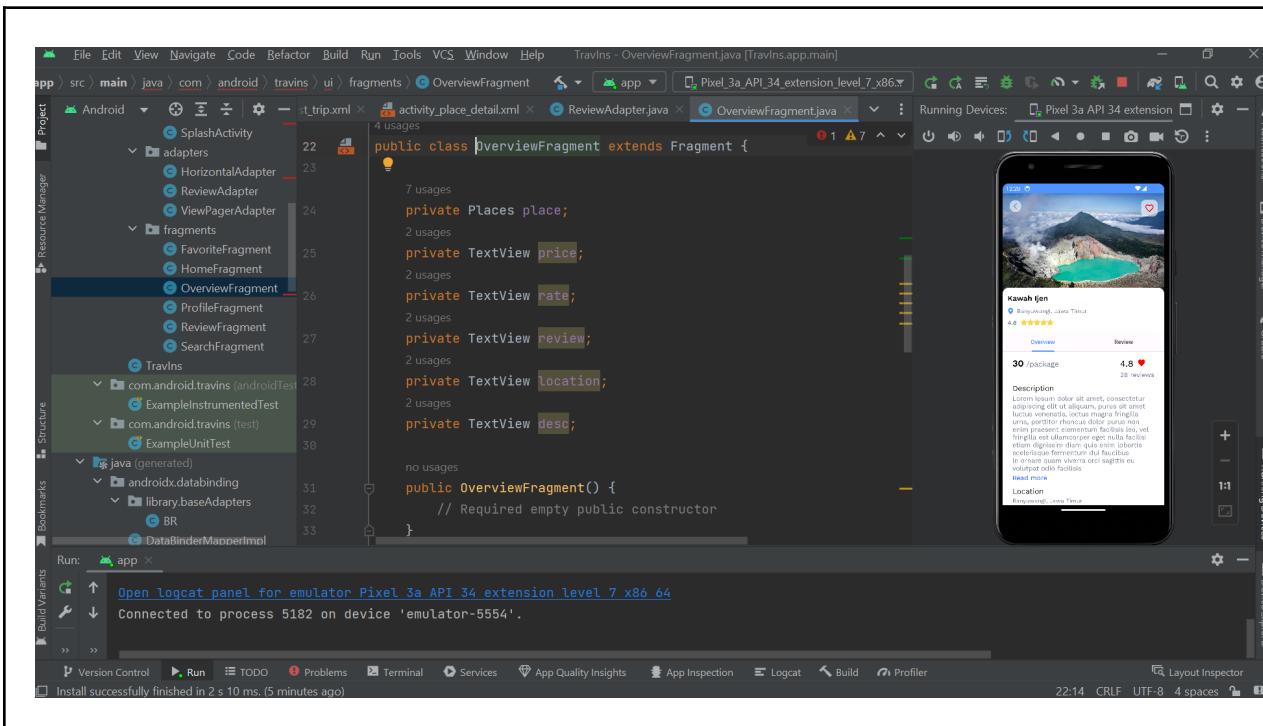
Metode ini merupakan bagian dari siklus hidup Fragment tetapi tidak mengandung implementasi tambahan dalam contoh ini.

4. onCreateView Method:

Metode ini menginflasi tata letak (layout) fragment dari file XML dengan nama fragment\_overview.xml.

5. onViewCreated Method:

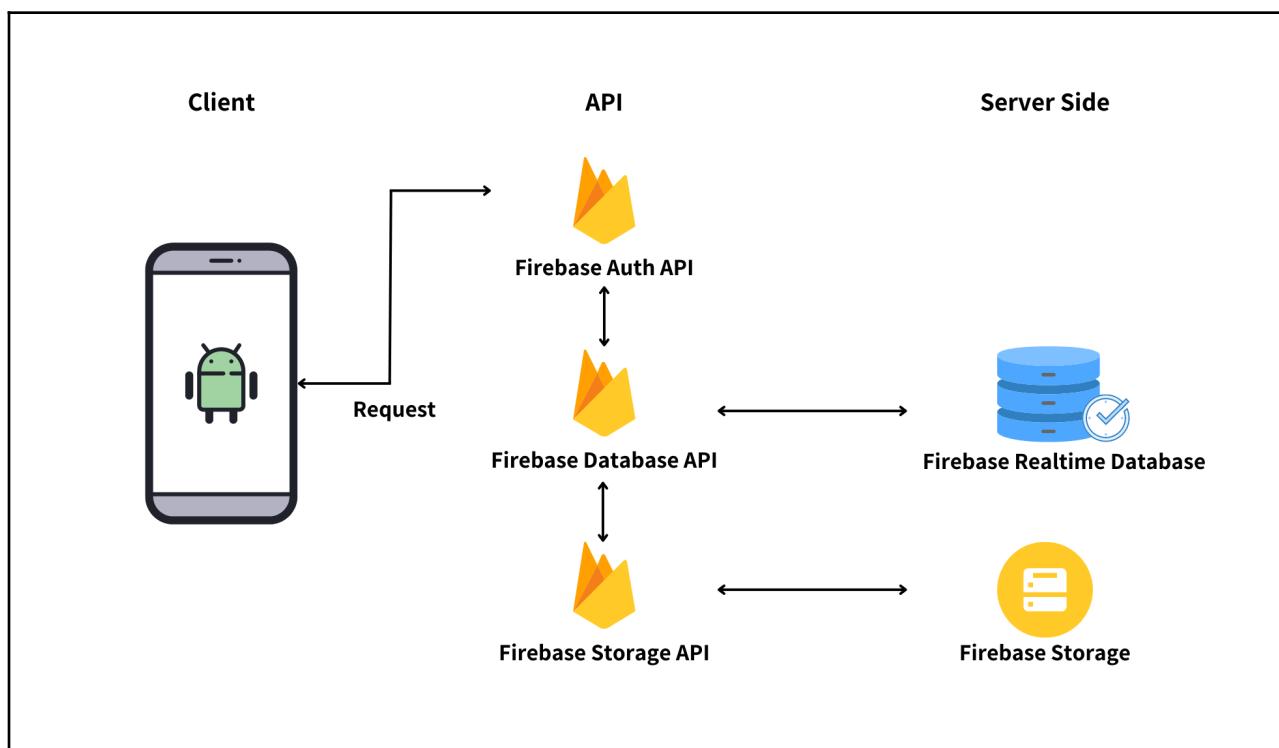
Metode ini dipanggil setelah tampilan fragment telah dibuat. Menginisialisasi TextView berdasarkan ID yang terdapat di layout. Mengisi TextView dengan informasi dari objek Places jika objek tidak null.



## 4. Implementasi Thread

Aplikasi ini tidak menggunakan thread untuk menangani operasi I/O yang dilakukan oleh Firebase dengan alasan berikut, setiap aplikasi memiliki thread utama yang digunakan untuk menjalankan kode UI. Jika aplikasi melakukan operasi I/O yang memakan waktu lama, operasi tersebut akan memblokir thread utama. Hal ini dapat menyebabkan aplikasi menjadi tidak responsif dan bahkan berhenti bekerja. Maka dari itu firebase menggunakan thread-pool internal untuk menangani operasi I/O, sehingga aplikasi tidak perlu menangani thread secara manual. Thread-pool adalah kumpulan thread yang dapat digunakan untuk menjalankan operasi I/O secara bersamaan. Dengan menggunakan thread-pool, operasi I/O tidak akan memblokir thread utama sehingga dapat membuat aplikasi lebih efisien dan responsif.

## 5. Implementasi Komunikasi Data



Gambar tersebut memberikan gambaran umum tentang cara kerja API. API adalah alat yang penting yang memungkinkan aplikasi berkomunikasi satu sama lain dan mengakses data dan layanan dari server. Dalam diagram tersebut, ada dua sisi utama API: sisi klien dan sisi server. Sisi klien adalah aplikasi yang menggunakan API. Aplikasi ini dapat berupa aplikasi web, aplikasi seluler, atau aplikasi desktop. Aplikasi klien biasanya menggunakan API untuk mengakses data atau layanan yang disediakan oleh server. Dalam diagram tersebut, sisi klien diwakili oleh ikon telepon Android. Telepon Android dapat menggunakan API Firebase untuk mengakses data dan layanan dari server Firebase.

Penjelasan tentang masing-masing API :

#### 4. Firebase Auth API

Firebase Auth API menyediakan berbagai fitur untuk mengelola autentikasi dan otorisasi pengguna, termasuk:

- Registrasi pengguna
- Masuk dengan akun Google atau email
- Verifikasi email
- Kelola akun pengguna

#### 5. Firebase Database API

Firebase Database API adalah database real-time yang memungkinkan aplikasi klien untuk menyimpan dan mengambil data secara real-time. Data disimpan dalam format JSON dan dapat diakses dari aplikasi klien menggunakan API.

#### 6. Firebase Storage API

Firebase Storage API adalah layanan penyimpanan file yang memungkinkan aplikasi klien untuk menyimpan file di cloud. File dapat diakses dari aplikasi klien menggunakan API.

# Use Case 5: Create & Read Destination Reviews

Briegitta Charmita

NIM 215150400111025



## Deskripsi Fungsional

### 1. Create Destination Reviews

- Deskripsi Fungsional

Fitur ini memungkinkan pengguna untuk membuat review dan rating mereka pada suatu tempat destinasi wisata di dalam aplikasi "TravIns".

- Tujuan

Tujuan dari pembuatan destination reviews adalah agar user dapat membagikan pengalaman mereka selama mengunjungi suatu tempat, memberikan pandangan pribadi, dan memberikan wawasan kepada pengguna lain tentang apa yang diharapkan, yang dapat meningkatkan kepuasan pribadi dan dapat menciptakan komunitas pengguna di dalam aplikasi travel, di mana orang dapat berinteraksi, bertukar informasi, dan saling membantu dalam perencanaan perjalanan.

- Cara Kerja

Pada homepage, pengguna perlu memilih salah satu destinasi wisata yang ingin dibuat reviewnya. Kemudian pada halaman detail destinasi tersebut, pengguna memilih 'Review' untuk menuju tampilan review. Pengguna yang ingin membuat ulasan dan ratingnya sendiri, maka dapat

melakukan scroll ke bawah sampai menemukan kotak 'Write your reviews'. Pengguna dapat menuliskan review dan memberikan rating dalam bentuk bintang, kemudian mengklik 'Post Review' untuk submit review yang telah dibuat.

## 2. Read Destination Reviews

- Deskripsi Fungsional

Fitur ini memungkinkan pengguna untuk melihat/membaca review yang ada dalam suatu tempat destinasi wisata.

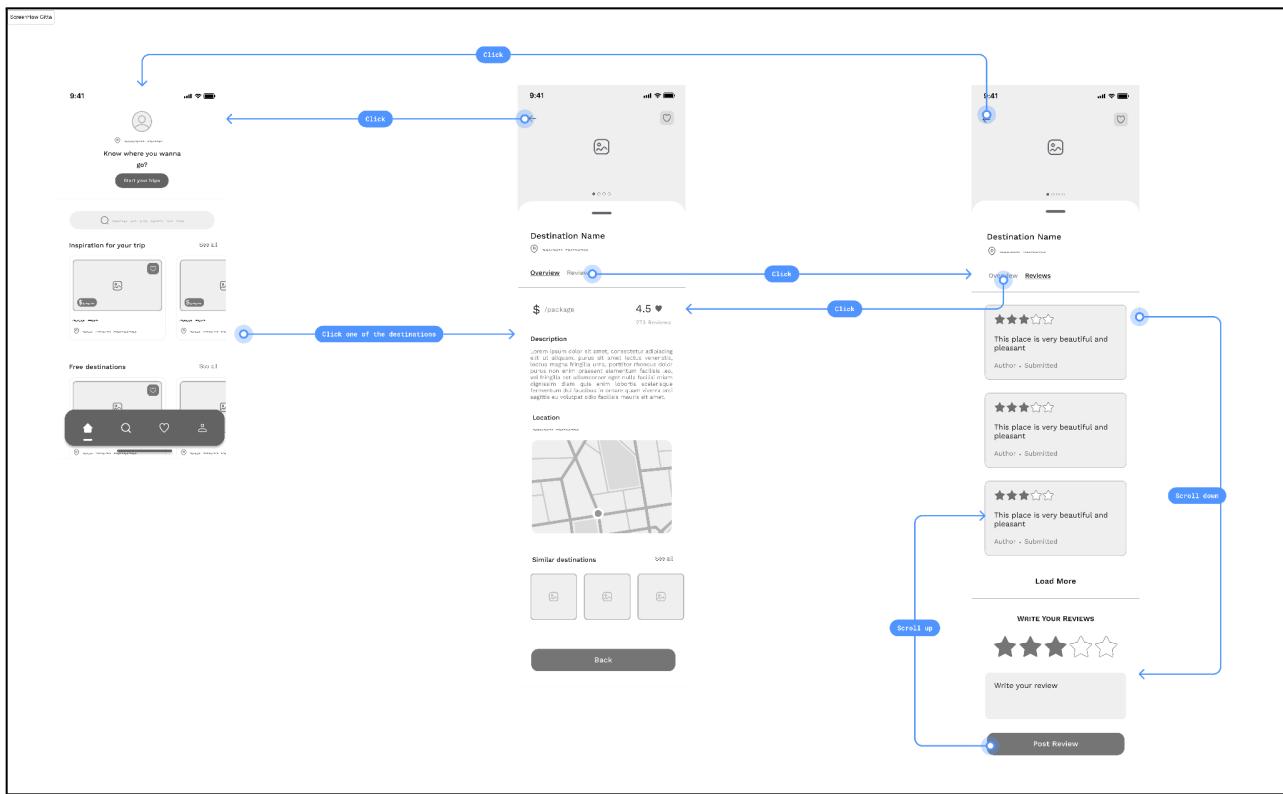
- Tujuan

Tujuan dari melihat review suatu tempat destinasi wisata adalah untuk memungkinkan user membaca berbagai ulasan yang memberikan mereka perspektif yang lebih beragam tentang suatu destinasi yang membuat keputusan perjalanan yang lebih informan melalui review yang memberikan wawasan tentang keberlanjutan dan kondisi terkini di destinasi, serta memungkinkan pengguna untuk berpartisipasi dalam komunitas pengguna aplikasi ini.

- Cara Kerja

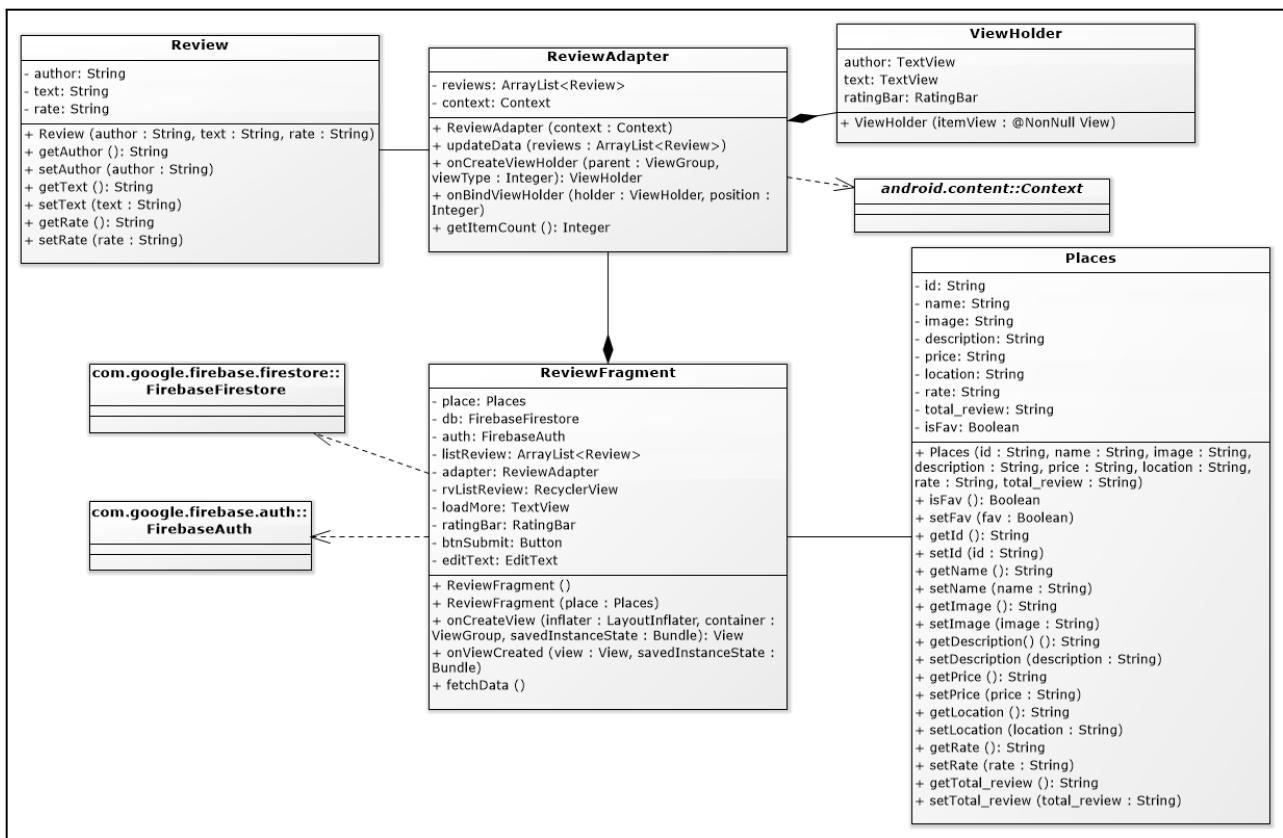
Pengguna dapat mengakses salah satu tempat destinasi wisata pada halaman aplikasi "TravIns" sehingga masuk ke detail destinasi. User membuka bagian review dengan mengklik 'Review', kemudian user dapat membaca berbagai review dan rating yang diberikan user lain ataupun melihat review yang baru mereka buat.

# Rancangan Screenflow



Gambar rancangan screenflow dalam bentuk wireframe di atas menggambarkan tampilan aktivitas-aktivitas yang dilakukan untuk menyelesaikan fungsionalitas create dan read destination reviews dari halaman utama atau homepage. Pada homepage, pengguna perlu memilih salah satu destinasi wisata yang ingin dilihat maupun dibuat reviewnya. Kemudian pada halaman detail destinasi, pengguna memilih ‘Review’ untuk menuju tampilan review. Pengguna dapat melihat langsung review yang diberikan pada destinasi wisata tersebut. Jika pengguna ingin membuat ulasan dan ratingnya sendiri, maka pengguna dapat scroll review-review tersebut sampai menemukan kotak ‘Write your reviews’. Terakhir, pengguna yang ingin melihat review yang baru saja ia tulis maka bisa scroll sedikit ke atas.

# Class Diagram



Fungsi dari setiap class yang terlibat dalam fungsionalitas create dan read destination reviews yang tergambar dalam Class Diagram di atas:

## 1. Class Review

- Tujuan: Mengelola tampilan daftar ulasan dalam RecyclerView.
- Fungsionalitas: Menyediakan adapter untuk menampilkan daftar ulasan, memperbarui data, dan mengatur tampilan setiap ulasan.

## 2. Class ReviewAdapter

- Tujuan: Mewakili sebuah ulasan dengan atribut untuk penulis, teks, dan nilai ulasan.
- Fungsionalitas: Memberikan metode untuk menetapkan dan mengambil informasi tentang sebuah ulasan.

## 3. Class ReviewFragment

- Tujuan: Menampilkan dan mengelola ulasan untuk suatu tempat.
- Fungsionalitas: Mengambil ulasan dari database, menyiapkan UI untuk menulis ulasan, dan menampilkan daftar ulasan dalam RecyclerView.

#### 4. Class Places

- Tujuan: Menampilkan dan mengelola ulasan untuk suatu tempat.
- Fungsionalitas: Mengambil ulasan dari database, menyiapkan UI untuk menulis ulasan, dan menampilkan daftar ulasan dalam RecyclerView.

# Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:name=".TravIns"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.TravIns"
        tools:targetApi="31">
        <meta-data
            android:name="com..sdk.ApplicationId"
            android:value="@string/_app_id"/>

        <activity
            android:name=".ui.activities.PlaceDetailActivity"
            android:exported="false"
            android:windowSoftInputMode="adjustPan"/>

        <activity
            android:name=".ui.activities.SplashActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>

        <activity
            android:name=".ui.activities.EditProfileActivity"
            android:exported="false" />
        <activity
            android:name=".ui.activities.ListTripActivity"
            android:exported="false" />
        <activity
            android:name=".ui.activities.RegisterActivity"
            android:exported="false" />
        <activity>
```

```
        android:name=".ui.activities.LoginActivity"
        android:exported="false" />
    <activity
        android:name=".ui.activities.SecondBoardingActivity"
        android:exported="false" />
    <activity
        android:name=".ui.activities.FirstBoardingActivity"
        android:exported="false" />
    <activity
        android:name=".ui.activities.MainActivity"
        android:exported="false"></activity>
</application>

</manifest>
```

Fungsionalitas Create dan Read Destination Reviews memerlukan permission android.permission.INTERNET yang digunakan untuk memungkinkan aplikasi mengakses internet. Izin ini diperlukan agar aplikasi dapat berkomunikasi dengan Firebase. Permission tersebut dideklarasikan dalam file manifest berikut:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Fungsionalitas ini menggunakan sebuah Activity dengan nama PlaceDetailActivity yang dideklarasikan dengan elemen berikut dalam file AndroidManifest.xml:

```
<application . . .>
    <activity
        android:name=".ui.activities.PlaceDetailActivity"
        android:exported="false"
        android:windowSoftInputMode="adjustPan"/>
</application>
```

# Implementasi UI

Tampilan UI:

The screenshot displays a mobile application interface for a travel destination. At the top, there is a header bar with icons for back, search, and other navigation. Below the header is a large image of a tropical island landscape with lush greenery and clear blue water. The title "Raja Ampat Islands" is centered above the image, along with the location "Raja Ampat, Papua Barat" and a rating of "4.8 ★★★★☆". There are two tabs at the bottom of this section: "Overview" and "Reviews", with "Reviews" being the active tab.

Below the image, three review cards are displayed, each showing a 5-star rating icon and a short positive comment. The comments read: "This place is very beautiful and pleasant", "This place is very beautiful and pleasant", and "This place is very beautiful and pleasant". Each comment includes a "Author Name • Submitted" link. A "Load More" button is located below these cards.

At the bottom of the screen, there is a "WRITE YOUR REVIEWS" section. It features a 5-star rating icon, a text input field labeled "Write your review", and a blue "Post Review" button.

Pada fungsional create destination reviews, user dapat membuat review/ulasan dan rating setelah sistem menampilkan daftar destinasi atau trip yang direkomendasikan berdasarkan preferensi user setelah user scroll ke bagian paling bawah dalam detail destinasi bagian review.

Pada fungsional read destination reviews, user dapat melihat ulasan dan rating penilaian destinasi yang telah ia buat atau ketika ingin melihat review yang telah dituliskan user lain setelah user men-click Review ketika melihat detail destinasi.

## Layout 1: [activity\_place\_detail.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.activities.PlaceDetailActivity"
    android:orientation="vertical">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageView
            android:id="@+id/ivDetailPlace"
            android:layout_width="match_parent"
            android:layout_height="280dp"
            android:scaleType="center"
            android:src="@drawable/img_raja_ampat"/>

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="16dp">
            <ImageView
                android:id="@+id/btnBack"
                android:layout_width="30dp"
                android:layout_height="30dp"
                android:src="@drawable/ic_back_white"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintLeft_toLeftOf="parent"/>
            <ImageView
                android:id="@+id/btnFav"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintRight_toRightOf="parent"
                android:src="@drawable/ic_heart"
                android:padding="8dp"

            android:background="@drawable/bg_white_transparent_rounded"/>
        </androidx.constraintlayout.widget.ConstraintLayout>
    </FrameLayout>

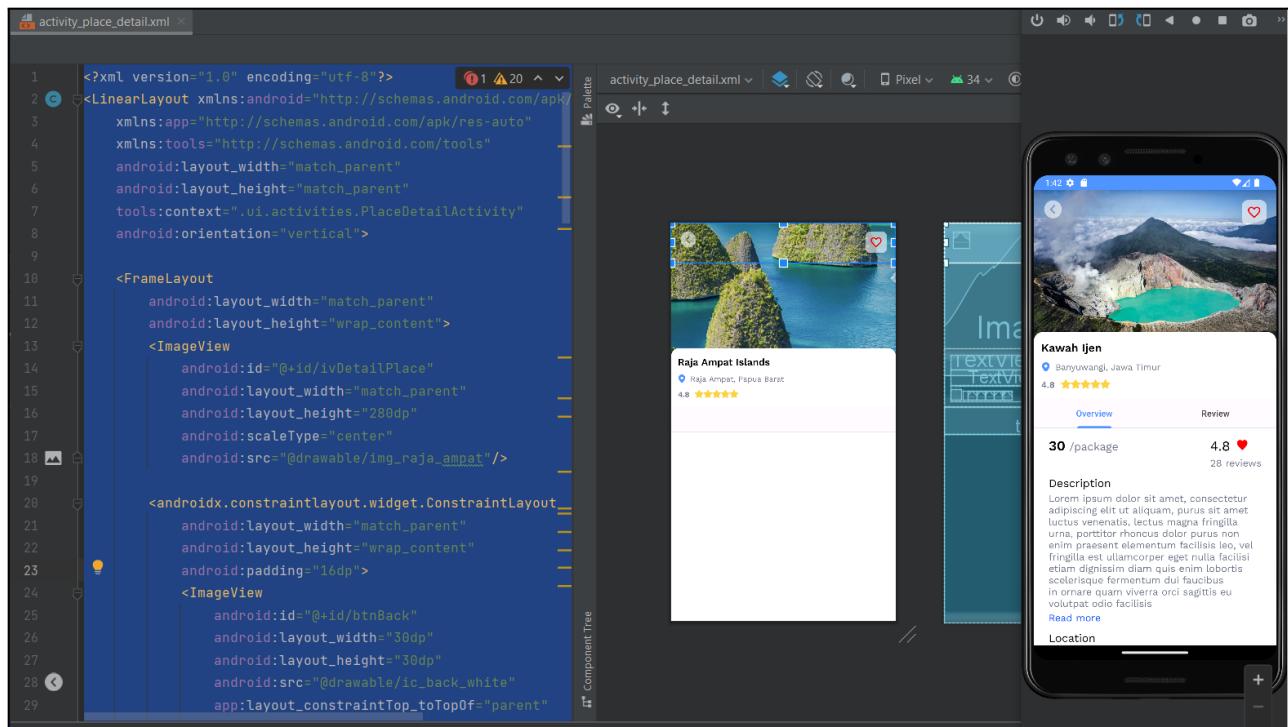
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="-50dp"
        android:orientation="vertical"
        android:background="@drawable/bg_white_top_left_right_rounded">

        <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="12dp">
    <TextView
        android:id="@+id/tvNamePlace"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Raja Ampat Islands"
        android:fontFamily="@font/worksans_semiBold"
        android:textColor="@color/black"
        android:textSize="18sp"/>
    <TextView
        android:id="@+id/tvLocationPlace"
        android:paddingTop="8dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Raja Ampat, Papua Barat"
        android:drawablePadding="8dp"
        android:textColor="@color/grey"
        android:fontFamily="@font/worksans_regular"
        android:drawableStart="@drawable/ic_location"/>
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingTop="8dp"
        android:layout_gravity="center_vertical">
    <TextView
        android:id="@+id/tvRate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="4.8"
        android:textSize="14sp"
        android:textColor="@color/grey"
        android:fontFamily="@font/worksans_semiBold"/>
<LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:paddingStart="10dp">
    <ImageView
        android:layout_width="16dp"
        android:layout_height="16dp"
        android:src="@drawable/ic_star"/>
    <ImageView
        android:layout_width="16dp"
        android:layout_height="16dp"
        android:src="@drawable/ic_star"/>
    <ImageView
        android:layout_width="16dp"
        android:layout_height="16dp"
        android:src="@drawable/ic_star"/>
```

```
<ImageView  
    android:layout_width="16dp"  
    android:layout_height="16dp"  
    android:src="@drawable/ic_star"/>  
<ImageView  
    android:layout_width="16dp"  
    android:layout_height="16dp"  
    android:src="@drawable/ic_star"/>  
    </LinearLayout>  
  </LinearLayout>  
</LinearLayout>  
  
<com.google.android.material.tabs.TabLayout  
    android:id="@+id/tab_layout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:tabGravity="fill"  
    app:tabMode="fixed"/>  
  
<androidx.viewpager2.widget.ViewPager2  
    android:id="@+id/pager"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" />  
  
</LinearLayout>  
</LinearLayout>
```

Layout ini digunakan dalam aktivitas `PlaceDetailActivity` untuk menampilkan detail suatu tempat. Bagian atas layout menggunakan `FrameLayout` dengan `ImageView` untuk menampilkan gambar header tempat. Di bagian ini, juga terdapat `ConstraintLayout` yang berisi tombol kembali dan tombol favorit. Bagian berikutnya menggunakan `LinearLayout` untuk menampilkan informasi seperti nama tempat, lokasi, dan rating. Desain tampilan umum diatur dalam `LinearLayout` terpisah dengan latar belakang yang menarik. Terdapat juga `TabLayout` dan `ViewPager2` untuk mengelola konten yang berbeda seperti Overview dan Review. Penggunaan atribut seperti `scaleType`, `drawablePadding`, `fontFamily`, dan penataan dengan `ConstraintLayout` dan `LinearLayout` untuk membuat tampilan yang estetis dan fungsional bagi user, meningkatkan pengalaman dalam menjelajahi detail tempat.



## Layout 2: [fragment\_review.xml]

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".ui.fragments.ReviewFragment">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rvListReview"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

        <TextView
            android:id="@+id/loadMore"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=""
            android:visibility="gone" />
    </LinearLayout>
</ScrollView>

```

```
        android:paddingTop="8dp"
        android:textColor="@color/bluePrimary"
        android:textSize="18sp"
        android:fontFamily="@font/worksans_semiBold"/>

    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:background="@color/grey"
        android:layout_marginVertical="18dp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Write Your Reviews"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:textAllCaps="true"
        android:fontFamily="@font/worksans_semiBold"/>

    <!--
    <ImageView-->
        android:layout_width="wrap_content"-->
        android:layout_height="80dp"-->
        android:src="@drawable/rating_bigger"-->
        android:paddingBottom="8dp"/>-->

    <RatingBar
        android:id="@+id/ratingBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:theme="@style/RatingBar"/>

    <EditText
        android:id="@+id/edtTxtReview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Write your review"
        android:minLines="4"
        android:elevation="8dp"
        android:layout_marginHorizontal="24dp"
        android:background="@drawable/bg_white_rounded"
        android:padding="10dp"/>

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btnSubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_blue_rounded"
        android:textAllCaps="false"
        android:text="Submit"
        android:textColor="@color/white"
        android:textSize="18sp"
        android:layout_marginVertical="16sp"
```

```

        android:layout_marginHorizontal="24dp"
    />
</LinearLayout>

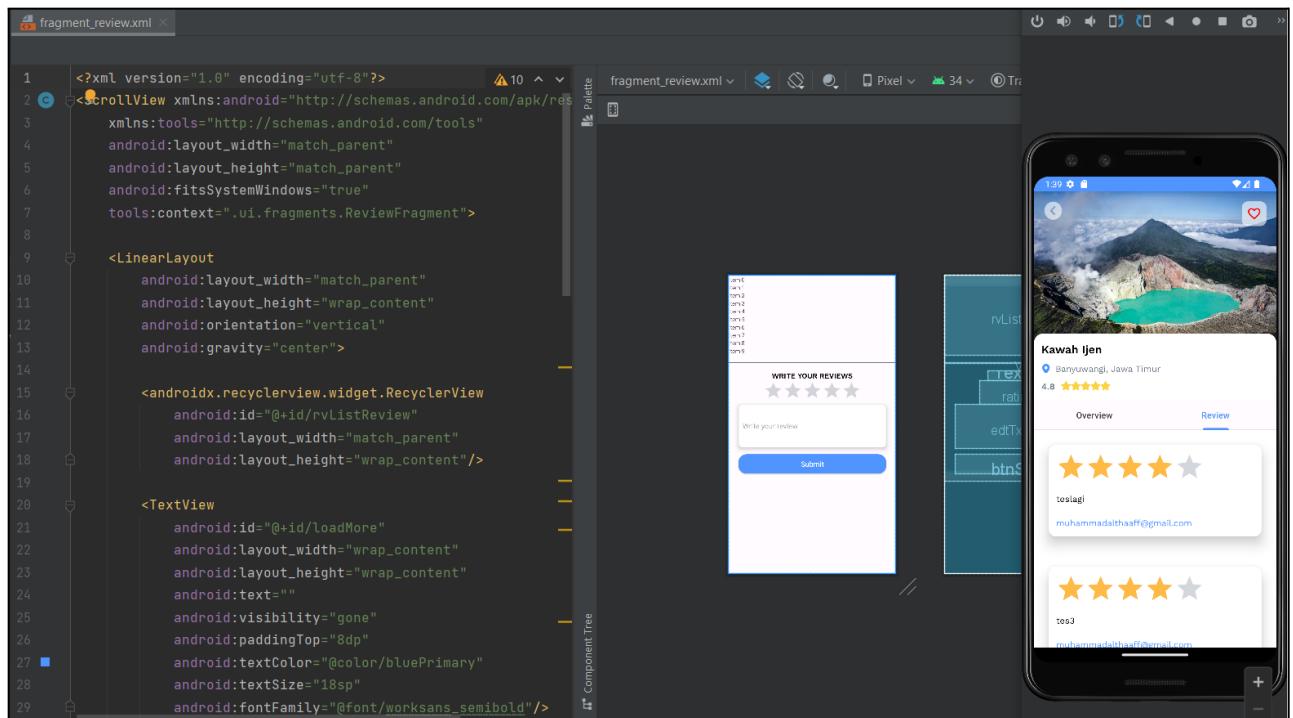
</ScrollView>

```

Layout ini terkait dengan fragment `ReviewFragment` dalam aktivitas `PlaceDetailActivity`. Dalam layout ini, terdapat `ScrollView` yang mengizinkan user untuk menggulirkan kontennya. Bagian utama dari layout ini adalah `RecyclerView` dengan ID `rvListReview`, yang bertanggung jawab untuk menampilkan daftar review dalam bentuk daftar vertikal. `TextView` dengan ID `loadMore` digunakan untuk memberikan pesan jika tidak ada review atau untuk memberikan opsi "Load More" jika diperlukan.

Selain itu, terdapat bagian untuk menulis ulasan baru dengan menggunakan `RatingBar` untuk memberikan peringkat, `EditText` untuk menulis ulasan, dan `AppCompatButton` untuk mengirim ulasan tersebut. Desain tampilan layout ini menciptakan antarmuka user untuk pengelolaan ulasan tempat dalam aktivitas `PlaceDetailActivity`.

Berikut tampilan screenshot layout-nya ketika aplikasi sedang dijalankan.



# Implementasi Kode Program Aplikasi

## Kode program Activity: PlaceDetailActivity.java

```
package com.android.travins.ui.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

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

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.android.travins.databinding.ActivityPlaceDetailBinding;
import com.android.travins.ui.adapters.ViewPagerAdapter;
import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.tabs.TabLayoutMediator;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firebaseio.QueryDocumentSnapshot;
import com.google.firebaseio.firebaseio.QuerySnapshot;

import java.util.ArrayList;
import java.util.Map;

public class PlaceDetailActivity extends AppCompatActivity {

    private ActivityPlaceDetailBinding binding;
    private boolean isFav;
    private ArrayList<String> tabsText = new ArrayList<>();

    private FirebaseFirestore db;
    private FirebaseAuth auth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityPlaceDetailBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        Places places = (Places) getIntent().getSerializableExtra("place");
        if (places!=null){
            binding.tvNamePlace.setText(places.getName());
            binding.tvLocationPlace.setText(places.getLocation());
            binding.tvRate.setText(places.getRate());
            Glide.with(this)
                .load(places.getImage())
                .into(binding.ivDetailPlace);
        }
    }
}
```

```

        fetchFavData(places);
    }

    tabsText.add("Overview");
    tabsText.add("Review");

        ViewPagerAdapter adapter = new
ViewPagerAdapter(getSupportFragmentManager(),getLifecycle());
        adapter.setPlaces(places);

        binding.pager.setAdapter(adapter);
        TabLayoutMediator tabLayoutMediator = new
TabLayoutMediator(binding.tabLayout,binding.pager,(tab, position) ->
        tab.setText(tabsText.get(position)))
    );
    tabLayoutMediator.attach();

    binding.btnExit.setOnClickListener(v -> {
        finish();
    });

    binding.btnFav.setOnClickListener(v -> {
        if (isFav){

db.collection("users").document(auth.getCurrentUser().getUid()).collection(
    "favorites").document(places.getId()).delete().addOnCompleteListener(task
-> {
            if (task.isSuccessful()){
                Toast.makeText(this, "Removed from favorite.",

Toast.LENGTH_SHORT).show();
                isFav = false;

binding.btnFav.setImageResource(R.drawable.ic_heart);
            }
        });
    }else{

db.collection("users").document(auth.getCurrentUser().getUid()).collection(
    "favorites").document(places.getId()).set(places).addOnCompleteListener(tas
k -> {
            if (task.isSuccessful()){
                Toast.makeText(this, "Added to favorite.",

Toast.LENGTH_SHORT).show();
                isFav = true;

binding.btnFav.setImageResource(R.drawable.ic_heart_red);
            }
        });
    });
}

```

```

    private void fetchFavData(Places places) {

        db.collection("users").document(auth.getCurrentUser().getUid()).collection(
        "favorites")
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document :
task.getResult()) {
                        Map<String, Object> data = document.getData();

                        if (document.getId().equals(places.getId())){
                            isFav = true;

                            binding.btnAddFav.setImageResource(R.drawable.ic_heart_red);
                        }
                    } else {
                    }
                });
            });
    }
}

```

### Kode program Fragment: ReviewFragment.java

```

package com.android.travins.ui.fragments;

import static android.content.Context.INPUT_METHOD_SERVICE;
import static androidx.core.content.ContextCompat.getSystemService;

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

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

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.android.travins.R;

```

```
import com.android.travins.data.models.Places;
import com.android.travins.data.models.Review;
import com.android.travins.ui.activities.LoginActivity;
import com.android.travins.ui.activities.RegisterActivity;
import com.android.travins.ui.adapters.HorizontalAdapter;
import com.android.travins.ui.adapters.ReviewAdapter;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firebaseio.QueryDocumentSnapshot;

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

public class ReviewFragment extends Fragment {

    private Places place;
    private FirebaseFirestore db;
    private FirebaseAuth auth;
    private ArrayList<Review> listReview = new ArrayList<>();

    private RecyclerView rvListReview;
    private ReviewAdapter adapter;

    private TextView loadMore;
    private RatingBar ratingbar;
    private Button btnSubmit;
    private EditText editText;

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

    public ReviewFragment(Places place) {
        this.place = place;
    }

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

    }

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

```

@Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        db = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        adapter = new ReviewAdapter(this.getContext());
        rvListReview = view.findViewById(R.id.rvListReview);
        loadMore = view.findViewById(R.id.loadMore);
        ratingbar= view.findViewById(R.id.ratingBar);
        btnSubmit = view.findViewById(R.id.btnSubmit);
        editText = view.findViewById(R.id.edtTxtReview);

        fetchData();

        // InputMethodManager mgr = (InputMethodManager)
        getSystemService(INPUT_METHOD_SERVICE);
        // mgr.hideSoftInputFromWindow(editText.getWindowToken(), 0);

        rvListReview.setLayoutManager(new
        LinearLayoutManager(this.getContext(),           LinearLayoutManager.VERTICAL,
        false));
        rvListReview.setAdapter(adapter);

        btnSubmit.setOnClickListener(v -> {
            String rate=String.valueOf(ratingbar.getRating());
            String text = editText.getText().toString();

            Map<String, Object> review = new HashMap<>();
            review.put("author", auth.getCurrentUser().getEmail());
            review.put("rate", rate);
            review.put("text", text);

            db.collection("places").document(place.getId()).collection("reviews").add(review)
                .addOnSuccessListener(documentReference -> {
                    Toast.makeText(view.getContext(), "Review Submitted", Toast.LENGTH_SHORT).show();
                    fetchData();
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(view.getContext(), "Error submitting review", Toast.LENGTH_SHORT).show();
                    }
                });
        });
    }
}

```

```

    }

    private void fetchData() {
        listReview = new ArrayList<>();

        db.collection("places").document(place.getId()).collection("reviews")
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document :
task.getResult()) {
                        Map<String, Object> data = document.getData();
                        Review review = new Review((String)
data.get("author"), (String) data.get("text"), (String) data.get("rate"));

                        listReview.add(review);
                    }
                    adapter.updateData(listReview);
                    if (listReview.isEmpty()){
                        loadMore.setText("No Review");
                        loadMore.setVisibility(View.VISIBLE);
                    }
                } else {
                    Toast.makeText(getApplicationContext(),"Fetch data
failed",Toast.LENGTH_SHORT).show();
                }
            });
    }
}

```

## Activity (PlaceDetailActivity)

Inisialisasi dan Pengaturan Awal: Membuat objek dari data binding untuk mengakses elemen UI.

```
binding = ActivityPlaceDetailBinding.inflate(getApplicationContext());
```

Menampilkan Detail Tempat Wisata: Mengambil objek Places dari intent untuk menampilkan detail tempat wisata seperti nama, lokasi, dan gambar.

```
Places places = (Places) getIntent().getSerializableExtra("place");
```

Menampilkan Review dan Mengelola Tabs: Menyiapkan data untuk tabs (Overview dan Review) dan membuat adapter untuk ViewPager.

```

        tabsText.add("Overview");
        tabsText.add("Review");
        ViewPagerAdapter adapter = new
ViewPagerAdapter(getSupportFragmentManager(),getLifecycle());

```

**Menangani Tombol Kembali dan Favorit:** Mengatur fungsi tombol kembali untuk menutup aktivitas dan menangani logika penambahan atau penghapusan dari daftar favorit.

```
binding.btnExit.setOnClickListener(v -> { finish(); });
binding.btnFav.setOnClickListener(v -> { // ... });
```

**Mengambil dan Menyimpan Data Favorit:** Memanggil metode `fetchFavData` untuk mendapatkan data favorit pengguna terkait tempat wisata dan mengatur tampilan tombol favorit.

```
fetchFavData(places);
```

**Menggunakan Firebase untuk Interaksi dengan Database:** Menggunakan Firebase Firestore untuk menambah atau menghapus tempat dari daftar favorit pengguna.

```
db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites").document(places.getId()).delete().addOnCompleteListener(task -> { // ... });

db.collection("users").document(auth.getCurrentUser().getUid()).collection("favorites").document(places.getId()).set(places).addOnCompleteListener(task -> { // ... });
```

### Fragment (ReviewFragment)

**Inisialisasi dan Pengaturan Awal:** Membuat objek adapter untuk menangani tampilan daftar review.

```
adapter = new ReviewAdapter(this.getContext());
```

**Mengambil Data Review dari Database:** Memanggil metode `fetchData` untuk mengambil data review dari Firebase Firestore terkait tempat wisata.

```
fetchData();
```

**Menangani Input Review Pengguna:** Mengatur fungsi tombol submit untuk mengambil input pengguna, seperti peringkat dan teks review, dan menyimpannya ke Firebase Firestore.

```
btnSubmit.setOnClickListener(v -> { // ... });
```

```
db.collection("places").document(place.getId()).collection("reviews").add(review).addOnSuccessListener(documentReference -> { // ... });
```

**Menampilkan Data Review dalam RecyclerView:** Menggunakan RecyclerView untuk menampilkan daftar review dalam tata letak vertikal.

```
rvListReview.setLayoutManager(new LinearLayoutManager(this.getContext(), LinearLayoutManager.VERTICAL, false));
rvListReview.setAdapter(adapter);
```

**Firebase Firestore untuk Interaksi dengan Database:** Menggunakan Firebase Firestore untuk mengambil data review terkini terkait tempat wisata dan menampilkan dalam RecyclerView.

```
db.collection("places").document(place.getId()).collection("reviews").get()
.addOnCompleteListener(task -> { // ... });
```

Activity `PlaceDetailActivity` bertanggung jawab untuk menampilkan detail tempat wisata dan mengelola fungsi tombol favorit. Pertama, aktivitas ini menginisialisasi elemen UI menggunakan data binding. Selanjutnya, data tempat wisata diambil dari intent, dan detailnya, seperti nama, lokasi, dan gambar, ditampilkan. Selain itu, terdapat tabs untuk "Overview" dan "Review" yang diatur dengan bantuan ViewPager. Tombol kembali ditangani untuk menutup aktivitas, dan tombol favorit diimplementasikan dengan logika penambahan atau penghapusan dari daftar favorit pengguna. Firebase Firestore digunakan untuk berinteraksi dengan database, memungkinkan penambahan atau penghapusan tempat dari daftar favorit.

Sementara itu, `ReviewFragment` berperan dalam menampilkan dan mengelola review pengguna terhadap tempat wisata. Dalam fragment ini, RecyclerView digunakan untuk menampilkan daftar review secara vertikal. Metode `fetchData` memperbarui tampilan dengan mendapatkan data review terkini dari Firebase Firestore terkait tempat wisata. Pengguna dapat memberikan review melalui input peringkat dan teks, yang kemudian disimpan di Firebase Firestore.

Kedua bagian ini saling melengkapi untuk memberikan pengguna pengalaman yang komprehensif dalam mengeksplorasi dan memberikan ulasan terhadap tempat wisata yang mereka kunjungi dalam aplikasi perjalanan ini. Firebase digunakan sebagai backend untuk menyimpan dan mengambil data, sementara tampilan UI dikelola dengan bantuan RecyclerView dan ViewPager.

## Implementasi ListView/AdapterView/RecyclerView

### Kode Program Adapter (Review Adapter)

```
public class ReviewAdapter extends RecyclerView.Adapter<ReviewAdapter.ReviewViewHolder> {
    private Context context;
    private ArrayList<Review> reviewList;

    // Konstruktor untuk inisialisasi adapter
    public ReviewAdapter(Context context) {
        this.context = context;
        this.reviewList = new ArrayList<>();
    }

    // Metode untuk mengupdate data review pada adapter
    public void updateData(ArrayList<Review> reviewList) {
        this.reviewList = reviewList;
        notifyDataSetChanged();
    }

    @Override
    public ReviewViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.item_review, parent, false);
        return new ReviewViewHolder(view);
    }

    @Override
    public void onBindViewHolder(ReviewViewHolder holder, int position) {
        Review review = reviewList.get(position);
        holder.bind(review);
    }

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

```

}

// Metode untuk membuat tampilan (row) baru
@NonNull
@Override
    public ReviewViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.row_review, parent, false);
        return new ReviewViewHolder(view);
}

// Metode untuk mengisi data ke dalam tampilan (row) yang sudah dibuat
@Override
    public void onBindViewHolder(@NonNull ReviewViewHolder holder, int position) {
        Review review = reviewList.get(position);
        holder.tvAuthor.setText(review.getAuthor());
        holder.tvText.setText(review.getText());
        holder.tvRate.setText(review.getRate());
    }

// Metode untuk mendapatkan jumlah item dalam dataset
@Override
public int getItemCount() {
    return reviewList.size();
}

// Kelas ViewHolder untuk menyimpan referensi tampilan (View)
public static class ReviewViewHolder extends RecyclerView.ViewHolder {
    TextView tvAuthor, tvText, tvRate;

    public ReviewViewHolder(@NonNull View itemView) {
        super(itemView);
        tvAuthor = itemView.findViewById(R.id.tvAuthor);
        tvText = itemView.findViewById(R.id.tvText);
        tvRate = itemView.findViewById(R.id.tvRate);
    }
}
}
}

```

1. ReviewAdapter adalah adapter khusus untuk RecyclerView yang menangani tampilan daftar review.
2. Konstruktor digunakan untuk inisialisasi adapter dan dataset (reviewList).
3. Metode updateData digunakan untuk mengupdate dataset dan memberi tahu adapter bahwa dataset telah berubah.
4. Metode onCreateViewHolder digunakan untuk membuat tampilan (row) baru.
5. Metode onBindViewHolder mengisi data ke dalam tampilan (row) yang sudah dibuat.
6. Metode getItemCount memberikan jumlah item dalam dataset.
7. Kelas ReviewViewHolder menyimpan referensi tampilan (View) dalam setiap item.

### Kode Program Class Objek (Review)

```
public class Review {  
    private String author;  
    private String text;  
    private String rate;  
  
    // Konstruktor untuk inisialisasi objek Review  
    public Review(String author, String text, String rate) {  
        this.author = author;  
        this.text = text;  
        this.rate = rate;  
    }  
  
    // Getter untuk mendapatkan nilai author  
    public String getAuthor() {  
        return author;  
    }  
  
    // Getter untuk mendapatkan nilai text  
    public String getText() {  
        return text;  
    }  
  
    // Getter untuk mendapatkan nilai rate  
    public String getRate() {  
        return rate;  
    }  
}
```

1. Review adalah kelas objek yang merepresentasikan satu review dengan atribut author, text, dan rate.
2. Konstruktor digunakan untuk inisialisasi objek Review dengan nilai atribut.
3. Getter digunakan untuk mendapatkan nilai dari setiap atribut.

### Kode Program XML RowView (row\_review.xml)

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="vertical"  
    android:padding="8dp">  
  
    <TextView  
        android:id="@+id/tvAuthor"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Author"
```

```
        android:textSize="16sp"
        android:textColor="@android:color/black"
        android:fontFamily="@font/worksans_semibold" />

    <TextView
        android:id="@+id/tvText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Review Text"
        android:textSize="14sp"
        android:textColor="@android:color/black"
        android:fontFamily="@font/worksans_regular" />

    <TextView
        android:id="@+id/tvRate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Rate"
        android:textSize="12sp"
        android:textColor="@android:color/black"
        android:fontFamily="@font/worksans_regular" />

</LinearLayout>
```

1. XML ini digunakan sebagai tata letak untuk setiap item dalam RecyclerView.
2. Mengandung tiga TextView untuk menampilkan informasi tentang review, yaitu author, text, dan rate.
3. Menggunakan properti seperti fontFamily untuk menentukan jenis font dan textSize untuk mengatur ukuran teks.

# Implementasi Fragment

## Kode Program Fragment: ReviewFragment.java

```
package com.android.travins.ui.fragments;

import static android.content.Context.INPUT_METHOD_SERVICE;
import static androidx.core.content.ContextCompat.getSystemService;

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

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

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.android.travins.data.models.Review;
import com.android.travins.ui.activities.LoginActivity;
import com.android.travins.ui.activities.RegisterActivity;
import com.android.travins.ui.adapters.HorizontalAdapter;
import com.android.travins.ui.adapters.ReviewAdapter;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.firebaseio.FirebaseFirestore;
import com.google.firebaseio.firestore.QueryDocumentSnapshot;

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

public class ReviewFragment extends Fragment {

    private Places place;
    private FirebaseFirestore db;
    private FirebaseAuth auth;
```

```
private ArrayList<Review> listReview = new ArrayList<>();

private RecyclerView rvListReview;
private ReviewAdapter adapter;

private TextView loadMore;
private RatingBar ratingbar;
private Button btnSubmit;
private EditText editText;

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

public ReviewFragment(Places place) {
    this.place = place;
}

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

}

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

@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    db = FirebaseFirestore.getInstance();
    auth = FirebaseAuth.getInstance();

    adapter = new ReviewAdapter(this.getContext());
    rvListReview = view.findViewById(R.id.rvListReview);
    loadMore = view.findViewById(R.id.loadMore);
    ratingbar= view.findViewById(R.id.ratingBar);
    btnSubmit = view.findViewById(R.id.btnSubmit);
    editText = view.findViewById(R.id.edtTxtReview);

    fetchData();

    // InputMethodManager mgr = (InputMethodManager)
getSystemService(INPUT_METHOD_SERVICE);
    // mgr.hideSoftInputFromWindow(editText.getWindowToken(), 0);
}
```

```

        rvListReview.setLayoutManager(new
LinearLayoutManager(this.getContext(),           LinearLayoutManager.VERTICAL,
false));
        rvListReview.setAdapter(adapter);

        btnSubmit.setOnClickListener(v -> {
            String rate=String.valueOf(ratingbar.getRating());
            String text = editText.getText().toString();

            Map<String, Object> review = new HashMap<>();
            review.put("author", auth.getCurrentUser().getEmail());
            review.put("rate", rate);
            review.put("text", text);

            db.collection("places").document(place.getId()).collection("reviews").add(r
evie)
                .addOnSuccessListener(documentReference -> {
                    Toast.makeText(view.getContext(), "Review
Submitted", Toast.LENGTH_SHORT).show();
                    fetchData();
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(view.getContext(), "Error
submitting review", Toast.LENGTH_SHORT).show();
                    }
                });
        });

    }

private void fetchData() {
    listReview = new ArrayList<>();

db.collection("places").document(place.getId()).collection("reviews")
    .get()
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document :
task.getResult()) {
                Map<String, Object> data = document.getData();
                Review review = new Review((String)
data.get("author"), (String) data.get("text"), (String) data.get("rate"));

                listReview.add(review);
            }
            adapter.updateData(listReview);
            if (listReview.isEmpty()){

```

```

        loadMore.setText("No Review");
        loadMore.setVisibility(View.VISIBLE);
    }

} else {
    Toast.makeText(getActivity(),"Fetch data failed",Toast.LENGTH_SHORT).show();
}
);
}
}

```

ReviewFragment.java adalah Fragment yang bertanggung jawab atas penanganan review. Menggunakan Firestore untuk penyimpanan data, FirebaseAuth untuk otentikasi, dan memiliki elemen UI seperti RecyclerView, RatingBar, Button, dan EditText. Metode fetchData() mengambil review dari Firestore, dan listener klik pada btnSubmit menambahkan review baru.

Kode program XML UI Layout yang memuat menjalankan Fragment ini: fragment\_review.xml

```

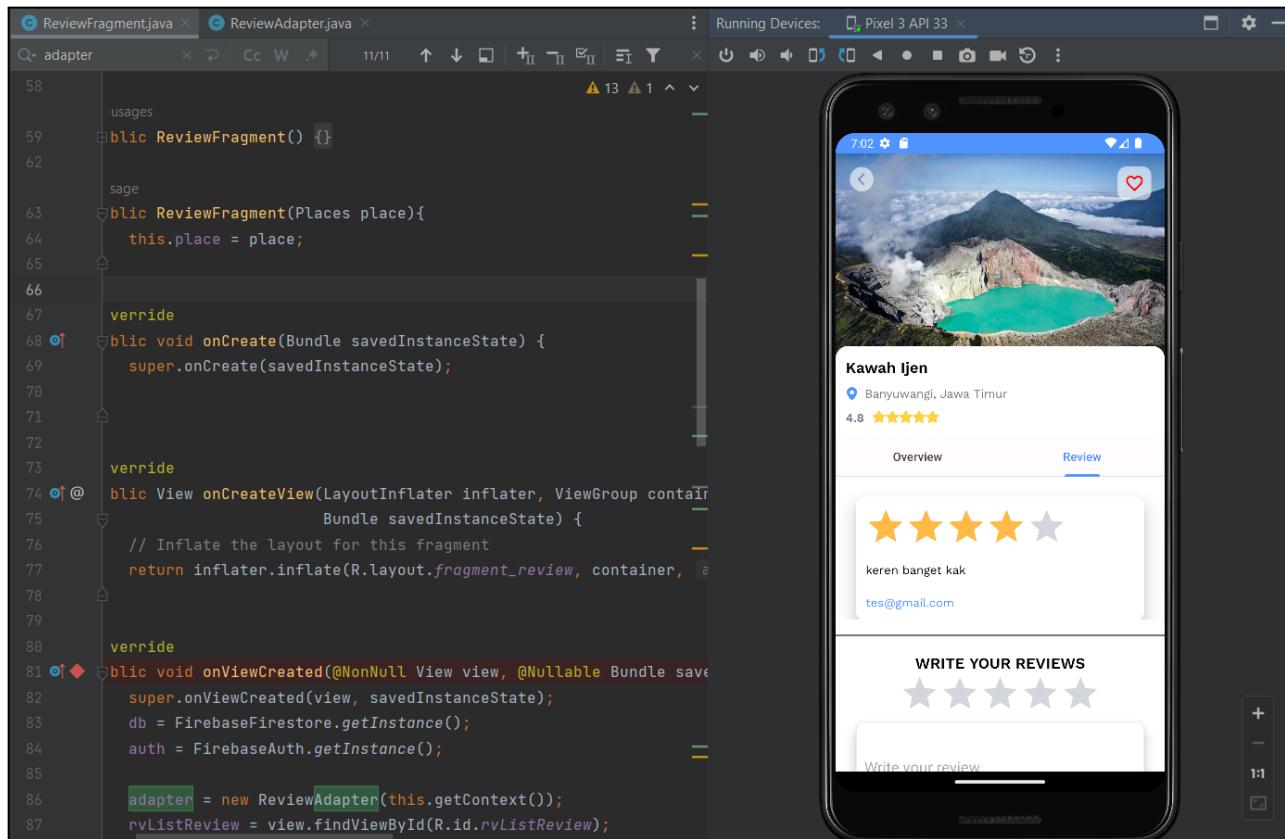
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".ui.fragments.ReviewFragment">

    ...
</ScrollView>

```

fragment\_review.xml adalah layout XML yang mendefinisikan antarmuka pengguna untuk ReviewFragment. Ini mencakup RecyclerView untuk menampilkan review, TextView "Load More", RatingBar untuk masukan pengguna, EditText untuk teks review, dan Button untuk mengirim ulasan.

Berikut screenshot tampilan fragment ketika aplikasi sedang dijalankan dan fragment sedang ditampilkan.



## Implementasi Thread

Kode program dalam Fragment yang melibatkan penggunaan Thread:

```
public class ReviewFragment extends Fragment {

    // ...

    private RecyclerView rvListReview;
    private ReviewAdapter adapter;

    // ...

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
```

```
super.onViewCreated(view, savedInstanceState);
db = FirebaseFirestore.getInstance();
auth = FirebaseAuth.getInstance();

// Inisialisasi adapter
adapter = new ReviewAdapter(this.getContext());
rvListReview = view.findViewById(R.id.rvListReview);
loadMore = view.findViewById(R.id.loadMore);
ratingbar = view.findViewById(R.id.ratingBar);
btnSubmit = view.findViewById(R.id.btnSubmit);
editText = view.findViewById(R.id.edtTxtReview);

// Mengatur adapter untuk RecyclerView
rvListReview.setLayoutManager(new
LinearLayoutManager(this.getContext(),           LinearLayoutManager.VERTICAL,
false));
rvListReview.setAdapter(adapter);

// ...

btnSubmit.setOnClickListener(v -> {
    // ...

    // Memanggil fetchData untuk memperbarui data ulasan
    fetchData();
});

}

private void fetchData() {
listReview = new ArrayList<>();

db.collection("places").document(place.getId()).collection("reviews")
    .get()
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document :
task.getResult()) {
                Map<String, Object> data = document.getData();
                Review review = new Review((String)
data.get("author"), (String) data.get("text"), (String) data.get("rate"));

```

```

        listReview.add(review);
    }

    adapter.updateData(listReview);

    if (listReview.isEmpty()) {
        loadMore.setText("No Review");
        loadMore.setVisibility(View.VISIBLE);
    }

} else {
    Toast.makeText(getApplicationContext(),"Fetch data failed",Toast.LENGTH_SHORT).show();
}
});
```

}

Berikut adalah adapter yang digunakan pada fragmen ini:

1. `fetchData()`: Pemanggilan metode yang melibatkan asynchronous operations (panggilan ke Firebase Firestore) yang berjalan di latar belakang menggunakan thread terpisah.

Pemanggilan `fetchData()` pada `onViewCreated` dan dalam metode `onSuccessListener` setelah submit review melibatkan operasi Firebase Firestore, yang berjalan di thread latar belakang. Sedangkan, Firebase SDK secara otomatis menangani manajemen thread untuk operasi tersebut.

2. `ReviewAdapter`

Digunakan untuk mengelola tampilan daftar ulasan (reviews) pada RecyclerView dan terlibat dalam pembaruan data terkait dengan ulasan yang diterima dari Firestore.

Adapter ini digunakan dalam metode `onViewCreated` dan `fetchData`. Dalam `onViewCreated`, adapter diinisialisasi dan diatur sebagai adapter dari RecyclerView. Pada saat pembaruan data (misalnya, setelah menambahkan ulasan baru), metode `updateData` dari adapter dipanggil untuk memperbarui tampilan daftar ulasan.

```

package com.android.travins.ui.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RatingBar;
import android.widget.TextView;
```

```
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.android.travins.R;
import com.android.travins.data.models.Places;
import com.android.travins.data.models.Review;

import java.util.ArrayList;

public class ReviewAdapter extends RecyclerView.Adapter<ReviewAdapter.ViewHolder>{

    private ArrayList<Review> reviews = new ArrayList<>();
    private Context context;

    public ReviewAdapter(Context context) {
        this.context = context;
    }

    public void updateData(ArrayList<Review> reviews) {
        this.reviews = reviews;
        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public ReviewAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.review_item_layout, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ReviewAdapter.ViewHolder holder, int position) {
        Review r = reviews.get(position);
        holder.text.setText(r.getText());
        holder.author.setText(r.getAuthor());
        holder.ratingBar.setRating(Float.parseFloat(r.getRate()));
    }

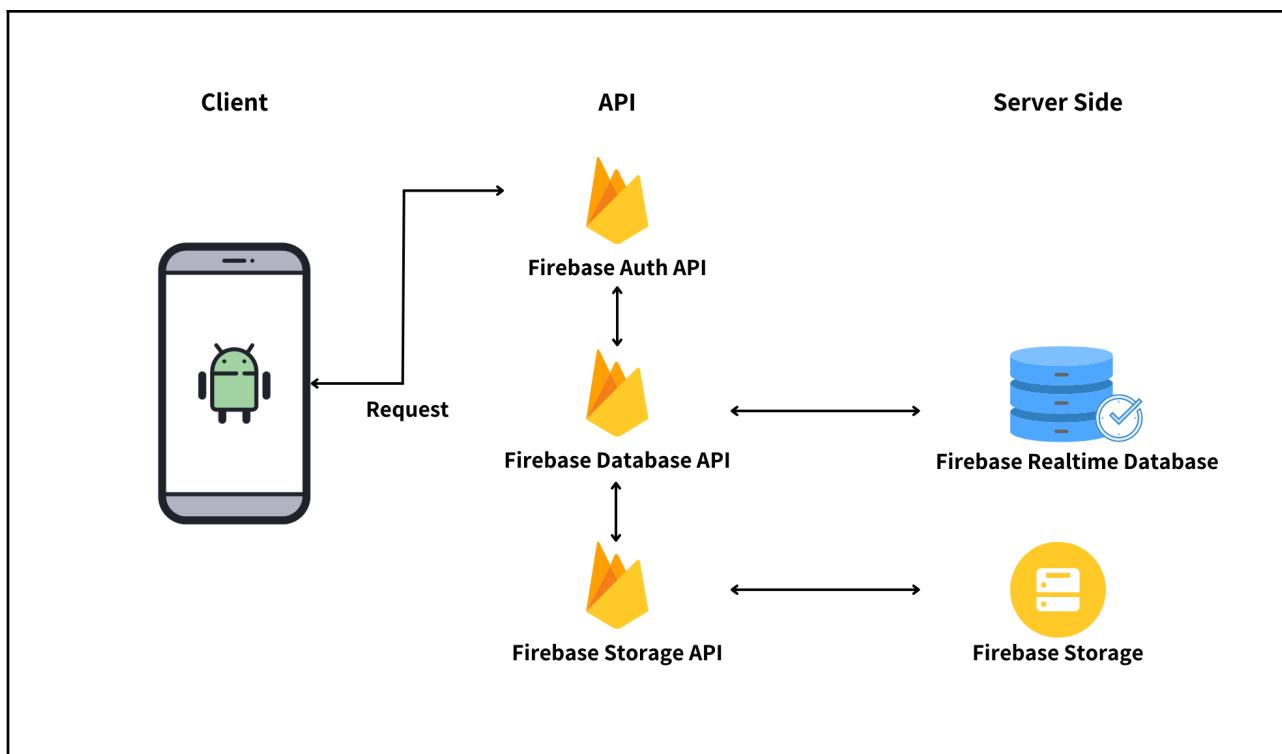
    @Override
    public int getItemCount() {
        return reviews.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {

        TextView author;
        TextView text;
        RatingBar ratingBar;
        public ViewHolder(@NonNull View itemView) {
```

```
        super(itemView);
        author = itemView.findViewById(R.id.textAuthor);
        text = itemView.findViewById(R.id.txtReview);
        ratingBar = itemView.findViewById(R.id.rating);
    }
}
```

## Implementasi Komunikasi Data



Gambar tersebut memberikan gambaran umum tentang cara kerja API. API adalah alat yang penting yang memungkinkan aplikasi berkomunikasi satu sama lain dan mengakses data dan layanan dari server. Dalam diagram tersebut, ada dua sisi utama API: sisi klien dan sisi server. Sisi klien adalah aplikasi yang menggunakan API. Aplikasi ini dapat berupa aplikasi web, aplikasi seluler, atau aplikasi desktop. Aplikasi klien biasanya menggunakan API untuk mengakses data atau layanan yang disediakan oleh server. Dalam diagram tersebut, sisi klien diwakili oleh ikon telepon Android. Telepon Android dapat menggunakan API Firebase untuk mengakses data dan layanan dari server Firebase. Berikut penjelasan tentang masing-masing API.

Firebase Auth API menyediakan berbagai fitur untuk mengelola autentikasi dan otorisasi pengguna, termasuk:

1. Registrasi pengguna
2. Masuk dengan akun Google atau email
3. Verifikasi email
4. Kelola akun pengguna

Firebase Database API adalah database real-time yang memungkinkan aplikasi klien untuk menyimpan dan mengambil data secara real-time. Data disimpan dalam format JSON dan dapat diakses dari aplikasi klien menggunakan API.

Firebase Storage API adalah layanan penyimpanan file yang memungkinkan aplikasi klien untuk menyimpan file di cloud. File dapat diakses dari aplikasi klien menggunakan API.