**App Architecture**

1**. Overview of MVC in Android**

The Model-View-Controller (MVC) pattern ensures clean separation between UI, business logic, and data management.

Model (Data Layer) → Manages Firebase Firestore, Authentication, and Storage

View (UI Layer - XML) → Displays data to users & handles interactions

Controller (Logic Layer - Java) → Manages user actions and updates the Model/View

2**. MVC Architecture Breakdown**

* **Model (Data Layer - Firebase)**

Stores & manages data (Firebase Firestore, Firebase Authentication, Firebase Storage)

Handles business logic related to bookings, destinations, and users

Works with Firebase SDK to retrieve and update data

* **Components:**

Firebase Authentication (User Login/Sign-up)

Firebase Firestore (User Bookings, Destinations, Payments)

Firebase Storage (Profile Pictures, Booking Receipts)

**Example:**

/users/{userID} → Stores user info

/bookings/{bookingID} → Stores user bookings

/destinations/{destinationID} → Stores available destinations

* **View (Presentation Layer - XML)**

Defines UI elements for Login, Booking, Destination Selection, Payment

Uses RecyclerView, TextView, EditText, Buttons

Displays data fetched from Controller

* **Key Screens:**

LoginActivity.xml → User authentication UI

BookingActivity.xml → Displays booking details

ProfileActivity.xml → Shows user information

**Example:**

TextView → Displays booking status

RecyclerView → Lists available destinations

Button → Triggers booking confirmation

* **Controller (Logic Layer - Java)**

Handles UI interactions and updates Model (Firebase)

Processes user authentication, bookings, and data fetching

Uses Firebase SDK for data retrieval and updates

* **Key Functions:**

UserController → Manages login, signup, authentication

BookingController → Handles booking creation, modifications

DestinationController → Fetches available destinations from Firestore

**Example**:

When User clicks "Book Now" button, the Controller:

Validates input

Sends booking data to Firestore.

Updates View with confirmation

**3. High-Level System Flow**

User opens the app (View)

Enters login credentials (Controller calls Firebase Authentication)

Firebase validates user and retrieves user data (Model)

User selects a destination (View)

Controller processes booking request

Booking is saved in Firebase Firestore (Model)

View updates to show confirmation & booking details

**4. Firebase Services Used**

Firebase Authentication → User login (Google, Email/Password)

Firebase Firestore → Stores booking details, destinations

Firebase Storage → Uploads user images, travel documents

Firebase Cloud Functions → Handles payments & notifications

Firebase Cloud Messaging (FCM) → Sends real-time booking updates

**5. Deployment & Future Enhancements**

Deploy to Google Play Store

Implement AI-based travel recommendations

Real-time updates using Firestore snapshot listeners

Payment integration with Stripe via Firebase Cloud Functions

Push notifications via Firebase Cloud Messaging (FCM)