**Android App Project: Login-Dashboard-Details Application**

**Overview**

This project is an Android application that demonstrates user authentication and data retrieval from an API. It features three primary screens:

1. **Login Screen**: Allows users to log in using their username and password.
2. **Dashboard Screen**: Displays a list of entities fetched from the API after successful login.
3. **Details Screen**: Displays detailed information about an entity when selected from the dashboard.

The app integrates with the vu-nit3213-api API for user authentication and data retrieval.

**Features**

* **User Authentication**: Allows users to log in with a username (first name) and password (student ID).
* **RecyclerView**: The dashboard displays a list of entities using RecyclerView.
* **ViewPager in RecyclerView**: Displays entities within a RecyclerView using ViewPager.
* **API Integration**: Interacts with a remote API to fetch data after user login.
* **Hilt for Dependency Injection**: Uses Hilt to manage dependencies.
* **Retrofit for Networking**: Handles HTTP requests and responses using Retrofit.
* **Coroutines**: Manages asynchronous tasks with Kotlin Coroutines.
* **Unit Tests**: Implements basic unit testing with JUnit.

**API Overview**

**1. Login Endpoint**

* **URL**: /footscray/auth, /sydney/auth, or /ort/auth (based on location).
* **Method**: POST
* **Request Body**:

json

Copy code

{

"username": "YourFirstName",

"password": "sYourStudentID"

}

* **Response**:

json

Copy code

{

"keypass": "topicName"

}

**2. Dashboard Endpoint**

* **URL**: /dashboard/{keypass}
* **Method**: GET
* **Response**:

json

Copy code

{

"entities": [

{

"property1": "value1",

"property2": "value2",

"description": "Detailed description"

},

// More objects...

],

"entityTotal": 7

}

**Screens**

**1. Login Screen**

* Allows the user to enter a username and password.
* Makes a POST request to the API for user authentication.
* If successful, navigates to the dashboard screen.

**2. Dashboard Screen**

* Displays a list of entities retrieved from the API.
* Uses a RecyclerView to display property1 and property2 for each entity.
* Allows users to tap on an entity to view its detailed information.

**3. Details Screen**

* Displays all available information about a selected entity.
* Shows the description field and other details of the entity.

**Project Setup**

**Prerequisites**

* **Android Studio**: The project is built using Android Studio.
* **JDK**: Java Development Kit version 8 or higher.
* **Internet Connection**: Required to interact with the remote API.

**Dependencies**

The project uses the following dependencies:

* **Hilt** for dependency injection.
* **Retrofit** for networking.
* **Coroutines** for background tasks.
* **JUnit** and **Espresso** for testing.

**How to Run**

1. **Clone the repository**:

bash

Copy code

git clone https://github.com/your-repo/android-app.git

1. **Open the project** in Android Studio.
2. **Build the project**:
   * Go to Build > Rebuild Project.
3. **Run the project**:
   * Choose a device or emulator and click Run.
4. **Configure API URL**:
   * Ensure the correct base URL is set in the NetworkModule.kt:

kotlin

Copy code

.baseUrl("https://vu-nit3213-api.onrender.com")

1. **Ensure Internet Permission**:
   * Verify that the following permission is included in the AndroidManifest.xml file:

xml

Copy code

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

**Folder Structure**

* app/src/main/java/com/example/: Contains the main app code, including activities, adapters, and view models.
* app/src/main/res/layout/: Contains the layout XML files for the app’s UI.
* app/src/main/res/values/: Includes strings, colors, and other app-wide resources.

**Key Files**

* **LoginActivity.kt**: Handles user login functionality.
* **DashboardActivity.kt**: Fetches and displays entities after login.
* **DetailsActivity.kt**: Shows detailed information for a selected entity.
* **EntitiesAdapter.kt**: Binds the entity data to the RecyclerView.
* **NetworkModule.kt**: Configures Retrofit for API interactions.

**Testing**

The project includes basic unit and instrumentation tests.

**Unit Testing**

* Tests for ViewModels and business logic are included.
* **JUnit** is used for testing.

**Running Tests**

* To run the tests, use Android Studio’s test runner:
  + Go to Run > Run... and select the desired test.

**Known Issues**

* The ViewPager inside the RecyclerView might require notifyDataSetChanged() to update when new data is inserted or removed.
* Ensure proper network connection to avoid timeouts during login.

**Future Improvements**

* Add more UI elements for a better user experience.
* Handle error cases more gracefully with user-friendly messages.
* Implement additional unit tests to increase code coverage.

References :

Google. (2018). *Android Developers*. Android Developers. https://developer.android.com/

‌