# Android Application with API, SQLite & UI Management

**Name:** Musayab Raza
**Reg No:** 23-ARID-579

## Application Overview

This Android application demonstrates fetching data from a public REST API, storing it locally using SQLite for offline access, and presenting it through multiple UI components. The app supports theme switching, menu-based navigation, persistent user state, WebView integration, and proper activity lifecycle handling.

## Section 1: Application Setup & Theme Management

Multiple themes (Light, Dark, Custom) are implemented. Users can switch themes from the Options Menu. The selected theme is saved using SharedPreferences and applied on app restart using lifecycle callbacks.

```
SharedPreferences prefs = getSharedPreferences("settings", MODE_PRIVATE);
prefs.edit().putString("theme", "dark").apply();
```

## Section 2: User State & Authentication Flag

A basic login screen is implemented. The isLoggedIn flag is stored in SharedPreferences. On app launch, the flag is checked and the user is redirected using Intent navigation.

## Section 3: Web Services / API Integration

Data is fetched from a public REST API (JSONPlaceholder Users API). Retrofit is used to handle HTTP requests and JSON parsing. Network errors and empty responses are handled gracefully.

```
@GET("users")
Call<List<User>> getUsers();
```

## Section 4: Local Data Persistence (SQLite)

An SQLite database is designed to store API response data. CRUD operations are implemented. When offline, data is loaded from SQLite.

## Section 5: Adapter Implementation

RecyclerView with a custom adapter is used to display stored data. Item click events are handled via adapter interfaces.

## Section 6: Menu Implementation & Navigation

Options Menu is used for global actions like theme switching and logout. Context Menu allows item-specific actions such as delete. Popup Menu is used for quick actions within list items.

## Section 7: WebView Integration

WebView is integrated to load external URLs related to fetched data. JavaScript is enabled and page loading states are handled to provide in-app browsing experience.

## Section 8: Input Controls & UI Interaction

Standard UI controls such as EditText, Button, Spinner, and Switch are used. Input validation is applied before processing user actions.

## Section 9: Activity Lifecycle & State Management

Configuration changes are handled using onSaveInstanceState. ViewModel prevents unnecessary API re-fetching and avoids memory leaks.

## Database Schema & API Details

Database: UsersDB
Table: users(id, name, email, phone)
API: https://jsonplaceholder.typicode.com/users

## Submission Requirements

• Complete Android Studio source code
• GitHub repository with README
• Screenshots of code, emulator output, and GitHub repo
• This PDF documentation