# User Profile App

## 1. Overview

This is an Android application that displays a list of users and allows users to view detailed information along with weather forecasting for the selected user's location.

### Features

- Fetch user data from RandomUser API
- Fetch weather data based on user location using OpenWeatherMap API
- Display paginated list of users
- Display user details with weather information
- Offline caching for API requests

## 2. Tech Stack & Libraries

### Programming Language & Architecture

- **Language:** Kotlin
- **Architecture:** MVVM + Repository Pattern

### Libraries Used

| Feature | Library | Purpose |
|---|---|---|
| Networking | Retrofit | API calls |
| JSON Parsing | Gson | Deserialize API responses |
| Image Loading | Glide | Load user profile images & weather icons |
| Database | Room | Local storage for users |
| Pagination | Paging 3 | Load users efficiently |
| Coroutines | Kotlin Coroutines | Asynchronous processing |

# 3. Features

## User List Screen

- Displays a **paginated list** of users fetched from the **RandomUser API**.
- Uses a **RecyclerView with Paging 3** for efficient data loading and smooth scrolling.
- Supports **offline caching** to reduce API calls and enhance performance.
- Clicking on a user item navigates to the **User Details Screen**.

## User Details Screen

- Displays selected user's **profile picture, name, email, phone number, and location**.
- Fetches and displays **real-time weather details** based on the user's latitude and longitude.
- Shows **temperature, weather description, humidity, and wind speed**.

# 4. Data Flow

## User Data Flow

1. The app requests user data from the **RandomUser API**.
2. The API response is processed, and the user list is **stored in Room** for offline access.
3. The data is displayed in a **RecyclerView with Paging 3** for efficient scrolling.

## Weather Data Flow

1. When a user is selected, their **latitude and longitude** are extracted.
2. A request is sent to the **OpenWeatherMap API** to fetch weather details for that location.
3. The **temperature, humidity, wind speed, and weather description** are extracted from the API response and displayed on the screen.

# 5. API Integration

## RandomUser API (User Data)

The app fetches user details from the **RandomUser API**, which provides a list of randomly generated user profiles. Each profile includes personal details such as name, email, phone number, and location (latitude and longitude).

### OpenWeatherMap API (Weather Data)

The **OpenWeatherMap API** is used to fetch weather information based on the user's coordinates. The weather data includes **current temperature, humidity, wind speed, and a short description** of the weather conditions.

To access weather data, an **API key** is required from OpenWeatherMap.

## 6. Offline Capabilities

- The app **caches user data** in a local Room database, enabling users to view previously fetched profiles even when offline.
- Weather data is **fetched in real-time**, but previously viewed user data remains accessible.
- An **offline caching mechanism** is implemented using an HTTP interceptor to serve cached responses when the network is unavailable.

## 7. User Experience & UI Design

### User List Screen (Main Screen)

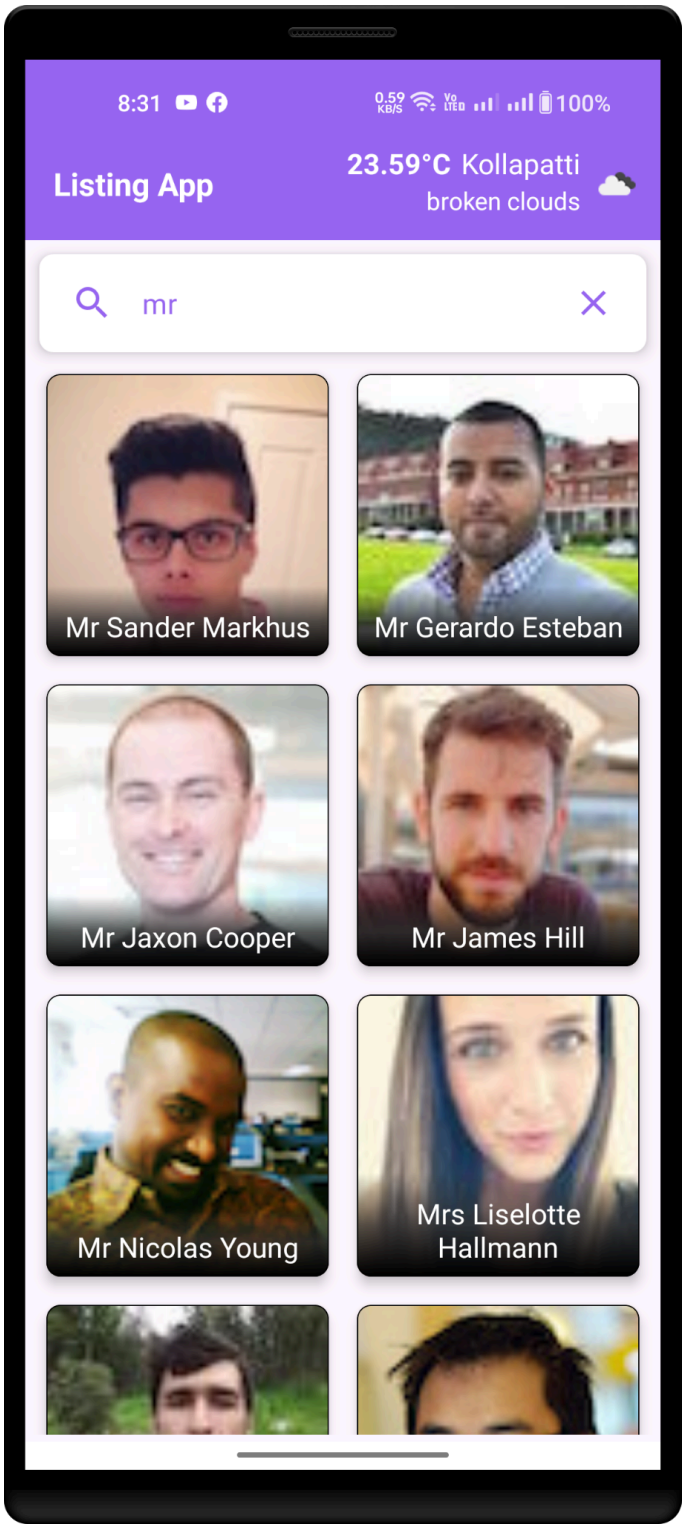
- Displays a **list of user profiles** fetched from the API.
- Uses **cards** to present user details in a clean and structured manner.
- Supports **infinite scrolling** with Pagination.
- Clicking on a user **navigates to the details screen**.

### User Details Screen

- Shows user **profile picture, name, email, and location**.
- Displays **real-time weather information** with an easy-to-read format.
- Uses a **loading indicator** while weather data is being fetched.

# 8. Screens

**User List Screen**



## User Details Screen