Android Developer Assignment

Objective:

Develop an Android application that allows users to view a paginated list of users fetched from an API, add new users with offline support, and navigate to a movie listing screen. The movie list should also support pagination and navigation to a movie detail screen. The app should adhere to best practices, including MVVM or MVI architecture, dependency injection, and reactive programming with Flow or LiveData.

Assignment Requirements:

1. User List Screen:

- Fetch and display a paginated list of users from the API:
    - **GET** `https://reqres.in/api/users?page={page}`
- Each item should display the **first name**, **last name**, and **avatar image**.
- Implement pagination to load more users as the user scrolls.
- Clicking on a user should navigate to the **Movie List Screen**.

2. Add User Functionality:

- Provide an option (e.g., a floating action button) to navigate to a screen where users can create a new user.
- The creation screen should prompt the user to input a **name** and **job**.
- If the device is online, the new user should be immediately posted to the API:
    - **POST** `https://reqres.in/api/users`

Request JSON:

```
 {
  "name": "morpheus",
  "job": "leader"
}
```

- If the device is offline, the user data should be stored in the Room **Database**.
- When the device regains internet connectivity, the app should automatically sync the offline data using **WorkManager**.
- After successful syncing, update the **ID** of the user in the local database.

3. Movie List Screen:

- When a user clicks on any item from the **User List Screen**, navigate to the **Movie List Screen**.
- Fetch a paginated list of trending movies from the API:

    - **GET** `https://api.themoviedb.org/3/trending/movie/day?language=en-US&page={page}&api_key=YOUR_API_KEY`
- Display each movie's **poster image, title, and release date**.
- Implement pagination to load more movies as the user scrolls.
- Clicking on a movie should navigate to the **Movie Detail Screen**.

4. Movie Detail Screen:

- When a user clicks on a movie from the **Movie List Screen**, navigate to the **Movie Detail Screen**.
- Fetch the movie details from the API:
    - **GET** `https://api.themoviedb.org/3/movie/{movie_id}?api_key=YOUR_API_KEY`
- Display **movie title, description, release date, and poster image**.
- Use `http://image.tmdb.org/t/p/w185/{poster_path}` to load movie images.
-

Generating TMDB API Key:

To fetch movie details, you need an API key from The Movie Database (TMDB). Follow these steps to generate one:

1. Go to TMDB API and sign in or create an account.
2. Navigate to the API section and request an API key.
3. Use the generated API key in the required API requests by replacing `YOUR_API_KEY` in the assignment.

Technical Requirements:

- **Programming Language:** Kotlin
- **Architecture:** MVVM or MVI
- **Dependency Injection:** Hilt/Dagger or Koin
- **Networking:** Retrofit with best practices
- **Local Storage:** Room Database
- **Offline Handling:** WorkManager for syncing data when the device regains internet connectivity
- **Pagination:** Use Paging 3 for efficient data loading
- **Asynchronous Data Handling:** Use Kotlin Flow or LiveData for reactive UI updates
- **Image Loading:** Glide or Coil

Submission Guidelines:

- Provide a link to a public Git repository containing your project code and APK.
- Include a `README.md` file with:

    - A brief description of the app.
    - Any relevant assumptions or considerations made during development.

Evaluation Criteria:

- Correctness and completeness of implemented features.
- Adherence to best practices and architectural patterns.
- Code quality, readability, and documentation.
- Handling of edge cases and error scenarios.
- User experience and UI design considerations.

We look forward to reviewing your implementation. If you have any questions or need further clarifications, please feel free to reach out.