Project Description

Objective

The objective of this project is to improve your skills in Flutter development, focusing on state management and API integration. Instead of complex architectures like MVVM, this task encourages you to use simple state management solutions such as setState or Provider. This task is designed to be straightforward and should be completed within approximately 4-5 days.

Requirements

- **Framework:** Flutter
- **Architecture:** Simple State Management (**setState** or **Provider**)
- **API:** JSONPlaceholder (https://jsonplaceholder.typicode.com/)
- **UI Framework:** Flutter's standard widget system (**Material Design**)

Project Description

Create a simple Flutter application with the following features:

Screen 1: User List

- Fetch a list of users from the JSONPlaceholder API.
- Display the list of users using a ListView.
- Each list item should display at least the user's name and email.

Screen 2: User Detail

- When a user taps on a list item, navigate to a detail screen.
- Display the user's name, email, phone, and website on the detail screen.

Architecture & Development Guidelines

- **State Management:** Use a simple approach, either setState() or Provider (no need for complex solutions like Bloc or Riverpod).

- **Networking Layer**: Fetch data using Dio or http package.
- **Repository Pattern (Optional):** If you want to separate concerns, you can use a repository pattern, but it's not mandatory.
- **Navigation**: Implement navigation between screens using Navigator.push().
    - UI Framework: Use Flutter’s Material Design components for a clean and simple UI.

Nice to Have

- **Unit Tests**: Adding unit tests for your networking layer and state management logic (if using Provider).
- **Inline Comments**: Use comments to explain your thought process and decision-making.

What Are We Expecting?

- **State Management**: Proper usage of setState() or Provider for managing UI state.
- **Networking**: Efficient handling of API calls and data parsing using http package or Dio.
- **Data Passing**: How data is passed between screens using Navigator.
- **Code Quality**: Clean, readable, and maintainable code following Flutter best practices.
- **Error Handling**: Proper handling of errors (e.g., showing a loading indicator or error message when the API fails).
- **Comments**: If unit tests are not included, add inline comments to explain your thought process and decisions.

Submission

Please submit your project via a GitHub repository. Ensure your code is well-documented and includes instructions on how to run the project.
