Project Approach Justification

Due to time constraints and a busy weekend, I have focused on implementing two key screens for the application: **Signup/Login** and **Home**. Below is a summary of the architecture, UI creation, model structuring, and API integration for these screens.

1. Signup/Login Screen

- **UI**: The screen includes essential form elements like `TextFormField` and `Elevated Button`.
- **Justification**: This screen serves as the entry point for user authentication.

2. Home Screen

- **UI**: Built with `Scaffold`, `AppBar` and `ListView`, this screen will display user data fetched from an API (Mock Data with future delay for fake response).
- **Justification**: The **Home** screen serves as the user's dashboard, displaying dynamic content. This layout is flexible and can easily be extended to include additional features.

3. MVVM Architecture

- **Approach**: The **Model-View-ViewModel (MVVM)** pattern separates the UI (View) from business logic (ViewModel and Model).
    - **Model**: Handles data and logic.
    - **View**: Displays UI.
    - **ViewModel**: Fetches and prepares data for the view.
- **Justification**: MVVM ensures scalability and maintainability, allowing easy testing and extending functionality without impacting UI logic.

4. UI Creation

- **Approach**: Flutter widgets like `Column`, `Row`, `Container`, and custom widgets are used for flexible, responsive design across devices.
- **Justification**: Flutter's cross-platform capabilities ensure the app performs consistently on both Android and iOS. Custom widgets enhance code reuse and consistency.

5. Model Creation

- **Approach**: Dart classes represent data models (e.g., User, Authentication), with methods for parsing API responses.
- **Justification**: Structured models ensure consistent data handling and easy API integration.

6. Reusable Components

- **Approach**: UI is created with the help of bunch of reusable components suc as UIScaffold, buttons, textfields and much more
- **Justification**:This approach ensures consistency across screens, reduces redundant code, and speeds up development. By reusing components, we maintain a clean and modular codebase, making future updates and additions easier and more efficient.

Conclusion

Due to limited time, I focused on delivering the core functionalities of user authentication and the home screen layout. The architecture (MVVM), UI design, model structure, and API integration are set up for easy expansion, allowing additional screens and features to be added seamlessly in the future.