Objective:

Develop a Flutter-based Advanced Currency Converter application that allows users to input amounts in multiple currencies, calculate their total value, and provide a normalised sum in a base currency using real-time exchange rates.

Requirements:

1. **User Interface:**
    - **Main Screen:**
        - An "Add Currency" button to add multiple currency fields.
        - An input field for entering amounts in different currencies.
        - A dropdown menu or search functionality to select the currency for each input amount.
        - A "Calculate Total" button to perform the conversion and sum up the values as per set base currency.
        - A text area to display the normalised total value in the base currency.
    - **Settings Screen:**
        - An option to select the base currency for normalisation.
    - **Currencies List Screen:**
        - A list of available currencies with their corresponding codes and names.
2. **Functionality:**
    - Fetch real-time exchange rates from a reliable API (e.g., Open Exchange Rates, ExchangeRate-API, or a similar service). API given below.
    - Allow users to input multiple amounts in different currencies.
    - Convert each input amount to the selected base currency and calculate the total value.
    - Display the normalised total value in the base currency when the "Calculate Total" button is pressed.
    - Implement basic error handling for issues such as network failures, invalid inputs, and unsupported currencies.
3. **Technical Specifications:**
    - Use the Flutter framework for the application.
    - The app should be compatible with both Android and iOS platforms.
    - The user interface should be responsive and adapt to different screen sizes and orientations.
    - Support for at least 20 different currencies.
    - The practical must follow the MVVM architecture.
    - Please use Riverpod state management.

    - The practical should be unit tested with at least code coverage of 50% (Mainly ViewModels & Repositories).
4. **Performance and Optimization:**
    - Ensure the app performs efficiently, with minimal load times for fetching exchange rates.
    - Cache exchange rates locally to minimise API calls and provide offline functionality for recently fetched rates.
5. **Testing:**
    - Include unit tests for key functionalities such as fetching exchange rates, converting amounts, and calculating totals.
    - Ensure the app is free of major bugs and provides a smooth user experience.
6. **Additional functionality: (Optional task)**
    - After the basic functionality gets done, app should work in offline mode if data exists locally.
    - Floor can be used for local caching.

Constraints:

- The app should be developed using the latest stable version of Flutter and Dart.
- Use state management solutions appropriate for the app's complexity (e.g., Provider, Riverpod).
- The app should follow best practices for code organisation, readability, and maintainability.

Submission Requirements:

- Provide the complete source code of the application in a Git repository.
- Include a README file with:
    - Instructions on how to build and run the app.
    - Any assumptions made during development.
    - Brief documentation of the code structure and design decisions.
- Provide screenshots or a short video demonstrating the app's functionality.
- Please add "surendra@webol.co.uk" & "dev@webol.co.uk" emails as collaborators on the github codebase.
- Expected development time: 10-12 hrs

API:

Consider the following API endpoints, which should be sufficient to complete the practical:
https://apilayer.com/marketplace/exchangerates_data-api#\
1. /symbols
2. /latest