Calling APIs in Flutter

Understanding APIs

Application Programming Interfaces (API) facilitates communication between the software systems. They provide your application access to real-time data on a distant server.

Asynchronous programming:

Asynchronous Programming allows you to perform time-consuming actions (*such as File I/O, making network requests, database operations*) without blocking the main UI thread.

API interactions are asynchronous network requests .

The response after calling an API is unpredictable. It may cause the app to freeze or become *unresponsive*.

Future, Async Await Keywords:

They prevent your App from *freezing*.

Future: It shows the potential value or error that isn't available yet.

Await/Async: They pass the execution until the *future* value is returned.

Making an API call:

1- Identify the web address or endpoint from where you want to fetch the data.
2- Use HTTP to communicate with the server.
3- Manage and interpret the response (which be in the JSON format).
4- Use JSON.decode to transform the JSON string into the Dart Object.
5- Create the Dart Model Class matching with JSON structure.
6- Map the model class within the decoded JSON data.
7- At last Update the UI accordingly.

Error Handling Techniques:

1- *Try-Catch Block:* Handles exceptions and prevents your app from crashing
2- *Using Timeouts:* Prevents your app from freezing.
3- *User Messages*: Use meaningful message to inform user so that he can try again later.

Best Practices while calling APIs:

- *Number 1* – Minimize number of API calls by implementing catching.
- *Number 2* – Enhance security using HTTP protocols and data validation techniques
- *Number 3* – Use modular code to improve code maintainability and testability (separate api call from UI logic).
- *Number 4* – Implement Error Handling Techniques to prevent crashes.

Some Considerations for API calls:

- *Number 1* – Get familiar with keywords like *Future, Async* and *Await*.
- *Number 2* – Create Try Catch Blocks to prevent crashes and handle issues.
- *Number 3* – Get familiar with *HTTP methods* such as *get, post, delete* and *put*. Use 'dart:convert' to parse JSON string into Dart Objects such as list or map.
- *Number 4* – Become familiar with provider Riverpod setState and block to handle state changes. Take note of the API rate limits to avoid getting blocked by the Server.
- *Number 5* – Use additional Headers and Authorized tokens in your Headers.