MKSC Flutter Project - Development Approach

Introduction

This document provides additional insight into the development approach used for the MKSC Flutter project. It complements the README.md file and dives into architectural and design decisions that shaped the development process.

Project Structure

The MKSC project is organized in a modular way to separate concerns and improve maintainability. Key components include:-

- lib/: Contains the main application code.
- assets/: Contains static resources like images and fonts.
- test/: Contains unit and widget tests.

State Management

The application uses the Provider package for state management. This is a simple yet powerful pattern that allows for efficient separation of the business logic from the UI code. Providers are placed at the top of the widget tree to manage global states like user authentication and data persistence.

Database Storage

For local storage, the app uses the Sqflite package for database management and SharedPreferences for lightweight, key-value data storage. Sensitive information, like authentication tokens, is stored using the flutter_secure_storage package. This ensures that data is protected on a per-device basis.

API Integration

The app uses the HTTP package to make API requests. Data from the server is fetched and processed asynchronously using Dart's Future and async/await mechanisms. Error handling is implemented at the network and parsing levels to ensure graceful failures.

UI Design

The user interface follows Material Design guidelines with customizations using Cupertino widgets where necessary to provide a native feel on iOS. Flutter's responsive design tools ensure that the UI adapts to different screen sizes and platforms (mobile, web, desktop).

Charting and Visualization

For data visualization, the project uses the fl_chart and pie_chart packages. These libraries allow the creation of custom charts like pie charts and line graphs, which are used in the app to display analytical data.

Authentication & Security

Password hashing is handled using the bcrypt package, ensuring that user credentials are stored securely. The app also uses flutter_secure_storage to securely store sensitive data on the device.

Conclusion

The MKSC project was developed with scalability, security, and maintainability in mind. By using Flutter's cross-platform capabilities, it provides a consistent experience across mobile, web, and desktop platforms. The use of Provider, Sqflite, and secure storage solutions ensures that the app is both efficient and secure.