**Fluxpense - Design and Development Report**

**Design Decisions**

Fluxpense was designed with a focus on simplicity and user-friendliness. The minimalistic approach was taken to streamline the user experience, providing only essential features for expense tracking. The clean architecture pattern was adopted, utilizing Flutter's Clean Architecture to ensure maintainability and scalability.

**Key Design Choices:**

**1. Clean Architecture**:

- Separation of concerns into layers for a clean and organized codebase.

- Independent business logic, presentation, and data layers for easier testing and maintainability.

**2. Flutter Riverpod for State Management:**

- Chose Flutter Riverpod for its simplicity and effectiveness in managing the app's state.

- Facilitates seamless communication between widgets and ensures a smooth user experience.

**Why Clean Architecture?**

**1. Separation of Concerns:**

- Clear Separation: Clean Architecture divides the codebase into independent layers, promoting maintainability.

**2. Testability:**

- Isolated Testing: Enables easy unit testing by keeping business logic independent of external frameworks, ensuring robust and reliable tests.

**3. Independence from Frameworks:**

- Framework Agnostic: Allows for changes in frameworks without impacting core business logic, enhancing adaptability.

**4. Adaptability to Change:**

- Localized Changes: Facilitates changes like database switches or UI framework updates without affecting the entire system.

**5. Scalability:**

- Scalable Structure: Provides a scalable structure supporting the growth of the application.

**Why Riverpod for State Management?**

**1. Declarative Syntax:**

- Readability: Utilizes a declarative syntax for expressive and readable code.

**2. Provider-Based:**

- Simplified State Management: Built on a provider system, simplifying state and dependency management.

**3. Reactive Programming:**

- Automatic Updates: Supports a reactive model for automatic UI updates with state changes.

**4. Scoped Providers:**

- Scoped State: Offers scoped providers for managing state at different levels of the widget tree.

**5. Flutter Integration:**

- Flutter-Friendly: Integrates seamlessly with Flutter, aligning well with Flutter's widget lifecycle.

**6. Null Safety Support:**

- Dart Null Safety: Developed with Dart's null safety, ensuring code robustness and reducing null reference errors.

**Dependencies used:**

**1. SQLite (sqflite):**

- For local database storage, ensuring persistent and reliable data storage.

**2. Local Notifications (flutter\_local\_notifications):**

- Integrated for reminding users to add their daily expenses, enhancing user engagement. (incomplete/ongoing)

**3. Timezone and Internationalization (timezone, intl):**

- For accurate handling of time-related data and supporting international users.

**4. Mocking Library (mockito):**

- Utilized for effective testing, ensuring robust and bug-free code.(ongoing)

**5. Charting Library (fl\_chart):**

- Integrated fl\_chart to visualize expense summaries using line and pie charts.

- Provides a visually appealing representation of expense data for better user insights.

**6. Shared Preferences (shared\_preferences):**

- Used for storing lightweight app settings, such as theme preferences and notification settings.

**Testing Approach (ongoing)**

Testing is a critical aspect of the development process to ensure the reliability and stability of Fluxpense. The focus was on unit testing business logic and integration testing key features.

**Testing Strategies:**

**1. Unit Testing:**

- Prioritized unit tests for isolated testing of individual functions and methods.

- Ensured that each unit of code works as expected independently.

**2. Integration Testing:**

- Integration tests to verify the interaction and collaboration between different components of the app.

- Checked the seamless functioning of key features, including expense management and chart visualization.

**3.Widget Testing:**

- Widget tests to validate the UI components, ensuring correct rendering and behavior.

- Covered various scenarios to guarantee the responsiveness and correctness of the user interface.

**Personal Note**

I would like to express my gratitude for the opportunity to take on this test. It has been an enriching experience, and I had a great time working on it. I appreciate the chance to showcase my skills in Flutter development.

Despite the time constraints, I am committed to further improving Fluxpense by addressing ongoing works and pending features. I look forward to a positive response .

Thank you for this opportunity.