**1. Project Overview**

**Debter** is a lightweight mobile application designed to work offline, built with Flutter. The main objective of the app is to let users create, edit, and manage their personal notes without needing an internet connection. It’s a straightforward yet effective tool for students, professionals, and anyone who wants to quickly jot down thoughts, tasks, or key information whenever they need to.

This app tackles the common problem of wanting a fast, distraction-free note-taking tool that operates fully offline. By using Hive for local storage and Flutter Quill for rich text editing, Debter makes sure users have complete control over their data, even in areas with poor connectivity.

**2. User Requirements**

As the intended users and stakeholders of the Debter App, we have identified the following specific requirements based on our expectations, daily note-taking habits, and usability needs:

1. **Simple and Intuitive User Interface**

* **Requirement:** The app must have a clean and easy-to-use interface that allows quick access to create, read, and manage notes.
* **Design Influence:** We prioritized a minimal UI with a focus on usability. The note editor uses a full-screen layout for immersive writing, and all unnecessary UI.

1. **Rich Text Editing**

* **Requirement:** Users want to format their notes with bold, italic, lists, and headings.
* **Design Influence:** We integrated the **Flutter Quill** rich text editor to provide advanced text formatting capabilities.

1. **Offline Access**

* **Requirement:** The app should work without an internet connection.
* **Design Influence:** We used **Hive** as the local storage solution, allowing users to create, edit, and view notes offline.

1. **Notes Organization**

* **Requirement:** Users want to categorize their notes (e.g., "Personal", "Work", "All").
* **Design Influence:** Each note includes a category field, and the UI allows filtering notes based on categories.

1. **Synchronization (Optional/Future)**

* **Requirement:** Users want to sync their notes across devices.
* **Design Influence:** Although we’ve currently focused on local storage, the architecture is designed to later support **Firebase Firestore** for real-time syncing and cloud backup.

1. **Performance on Low-End Devices**

* **Requirement:** The app must be lightweight and run smoothly on devices with limited memory (e.g., 4GB RAM).
* **Design Influence:** We chose Hive for its speed and low memory usage, and avoided memory-intensive animations or unnecessary background processes.

**Functional Requirements:**

1. Create new text-based notes.
2. View a list of saved notes.
3. Edit or delete existing notes.
4. Organize notes by category (like Work, Personal, Ideas).
5. Store notes locally with the Hive database.
6. Switching between Light Mode and Dark Mode

**Non-Functional Requirements:**

1. Full offline support for all main features.
2. A fast and responsive user interface.
3. Reliable data storage.
4. A clean, minimalist design for ease of use.

We came up with these requirements by thinking from the perspective of the primary users, focusing on simplicity, performance, and offline access in our design choices.

**3. Design Concepts**

**UI/UX Principles:**

* A minimalist design to minimize distractions.
* A full-screen writing interface for maximum focus.
* A consistent layout with proper padding and spacing.
* A Floating Action Button (FAB) for easy note creation.
* Following Material Design principles for accessibility and familiarity.

**Navigation Flow:**

*`Splash Screen → Home (Note List) → Add/Edit Note Screen → Save → Return to Home`*

**Visual Elements:**

* **Font**: Roboto
* **Primary Theme**: Light mode with soft accent colors
* **Icons**: Material Design Icons
* **Note Editor**: Quill rich text editor widget

**4. Development Approach**

We adopted an **Agile** development methodology, breaking down the process into small, manageable iterations that focused on implementing and testing specific features. This approach helped us gather regular feedback, address bugs early on, and enhance the UI based on testing insights.

**Challenges Faced:**

* **Editor Layout**: It was tricky to get the Quill editor to occupy the full screen while hiding the toolbar.
* **Local Data Management:** We needed to ensure that rich text formats could be saved and retrieved reliably from Hive, which required converting content to and from JSON.

We tackled these challenges by customizing the editor’s UI and using JSON serialization for storing Quill documents.

**5. Technological Stack**

|  |  |  |
| --- | --- | --- |
| **Component** | **Technology Used** | **Reason for Selection** |
| Frontend | Flutter (Dart) | Cross-platform and very performant |
| Local Storage | Hive | Lightweight, fast, and optimized for Flutter |
| Rich Text Editing | Flutter Quill | Easy to integrate and supports rich text |
| State Management | setState, ValueNotifier | Simple and sufficient for our app’s scope |
| Prototyping Tool | Figma | For initial design mockups and flow visualization |

**6. Implementation Details**

**Core Features:**

1. **Create Note**: Users can input a title, write content, and select a category.
2. **View Notes**: Displays saved notes in a list format, showcasing the title and date.
3. **Edit/Delete Notes**: Tapping a note opens it in the editor with options to update or delete.
4. **Local Persistence**: Notes are stored using Hive boxes, with data being serialized from Quill documents.

**7. Testing and Quality Assurance**

**Testing Strategies:**

* **Manual Testing**: All features were tested on Android devices to confirm performance and usability.
* **Unit Testing**: The logic for saving and loading notes from Hive was tested in isolation.
* **Integration Testing**: The flow of creating notes → saving → displaying in the list was tested as a whole.

**Results:**

* No crashes were encountered during offline use.
* Notes saved correctly after restarting the app.
* The UI was responsive and consistent across various screen sizes.

**8. Future Enhancements**

**Planned Features:**

* **Cloud Sync via Firebase**: Allow cloud backups and access from multiple devices.
* **User Authentication**: Email/password login for separating user notes.
* **Google Drive Backup**: Manual or automatic note export/import.

These enhancements will improve usability and data security, aligning with the long-term needs of our users.