Time Tracker User Manual (Developers)

Time Tracker App

Introduction

The Time Tracker app is a Flutter application that utilizes Firebase for data storage. It allows users to record and query their time entries based on date, task, and tag.

Architecture

The app follows a basic architecture with three main components:

1. **Data Model (`time_entry.dart`):**
    - Defines the structure of a time entry using the `TimeEntry` class.
    - Provides methods for converting data to and from Firestore.
2. **Database Operations (`database.dart`):**
    - Handles interactions with Firebase Firestore.
    - Implements methods to add time entries (`addTimeEntry`) and query time entries (`getTimeEntries`).
3. **User Interface (`add_entry_screen.dart`, `query_screen.dart`):**
    - `AddEntryScreen`: Allows users to input and save time entries.
    - `QueryScreen`: Enables users to query and view time entries based on specified criteria.

Usage Guidelines

1. **Firebase Setup:**
    - Ensure that the Firebase project is set up correctly, and the necessary configuration files (`google-services.json` and `GoogleService-Info.plist`) are added to the Android and iOS directories.

2. **Dependencies:**
    - Confirm that the required dependencies (`firebase_core` and `cloud_firestore`) are correctly added to the `pubspec.yaml` file.
3. **User Interface:**
    - Enhance the UI for a better user experience.
    - Implement additional features as needed based on user requirements.
4. **Testing:**
    - Perform thorough testing of the app to ensure proper functionality.
    - Consider edge cases and validate user inputs.
5. **Documentation:**
    - Maintain clear and concise documentation for future reference.
    - Include comments in the code for better readability and understanding.

Customization

- Customize the UI and add features to meet specific client requirements.
- Implement user authentication for a more personalized experience.

Conclusion

The Time Tracker app is a foundation for time management. As a software engineer, you have the flexibility to extend its features and enhance its capabilities based on evolving user needs. Always prioritize code quality, documentation, and testing for a robust application.