Prioritized To-Do List App – Revised Technical Specification

Project Overview

The **Prioritized To-Do List App** is designed to help users efficiently manage their tasks by organizing them based on priority and category. This application will enhance productivity by enabling users to create, modify, and manage tasks with deadlines. Developed in Python, the app will use CSV files for data storage. This version focuses on delivering a functional task manager with filtering options, while more advanced features (such as a calendar view and time zone handling) will be considered for future versions.

Architecture

-Frontend: The interface will be built using either Tkinter (for a desktop GUI) or Bootstrap(for a web-based UI), depending on the final decision.
- Backend: Pure Python logic will handle all task management functionalities.
- Data Storage: Task data will be persisted in CSV files, with no plans to implement SQLite in this version.

Data Structure

Tasks will be stored in a pandas DataFrame, which provides a flexible and efficient way to manage task data. Each task will include the following attributes:
- Title: (String) The name of the task.
- Category: (String) The category of the task (e.g., Work, Personal, School).
- Priority: (String) The priority level (High, Medium, Low).
- Deadline: (Datetime) The due date for the task.
- Status: (Boolean) Indicates whether the task is completed.

Functional Requirements

1. **Task Creation:** Users can create new tasks by providing a title, category, priority, and deadline.
2. **Task Modification**: Users can edit task details, including title, category, priority, and deadline.
3. **Task Deletion:** Users can delete tasks from the list.
4. **Mark as Done/Not Done:** Users can toggle the completion status of tasks.
5. **Task Filtering and Sorting:** Users can filter tasks by category, priority, and deadline.
6. **CSV Data Storage:** Tasks will be saved in CSV files to maintain persistence between sessions.

Nice-to-Have Features

1. **Calendar View**:

   Description: A visual calendar that allows users to view tasks with deadlines on specific dates.

   Purpose: Enhances user experience by providing an overview of tasks in a calendar format.

   Status: Optional; implementation will depend on available time after core functionalities are complete.

2. **Time Zone Handling**:

   Description: Functionality to manage time zones based on the user's local time.

   Purpose: Enables users in different time zones to see task deadlines in their local time.

   Status: Optional; the current system operates in UTC.

User Interface Components

- **Main Window**: Displays the current list of tasks, along with options to create, modify, or delete tasks.
- **Input Fields**: Fields for entering task details, including title, category, priority, and deadline.
- **Buttons**:
  - Add Task
  - Modify Task
  - Delete Task
  - Mark as Done/Not Done
  - Filter Tasks
- **Calendar View** (Optional): A graphical representation of tasks with deadlines.

Technical Implementation Details

- **CSV File Handling**: Utilize Python's built-in `csv` module to store tasks and manage data persistence.
- **Datetime Handling**: The `datetime` module will be used for managing and validating task deadlines in UTC, with future iterations considering user time zone management.
- **Frontend Options**:
  - **Tkinter**: For the desktop implementation, using Python’s Tkinter module to create a basic graphical interface.
  - **Bootstrap**: For a web-based version, utilizing Flask for the backend and Bootstrap for a responsive front-end interface.

Data Flow

1. **Input**: Users enter task details through the interface.
2. **Processing**: The app processes requests for task creation, modification, or deletion.
3. **Output**: Updated task lists and statuses are displayed to the user.

Final Assessment

- **Key Change**: The focus has shifted more towards core task management functionality, leaving out the SQLite database and notification system for this version.
- **Confidence Level**: High confidence in successfully implementing the outlined core features.
- **Potential Challenges**: Choosing between Tkinter and Bootstrap for the front end may slightly impact development time.
- **Nice-to-Have Features**: The calendar view and time zone handling are optional features that may be integrated if time allows.