

*School of Mechanical & Manufacturing Engineering (SMME),*

*Department of Aerospace Engineering*

**Name: Muhaddasa Sohail**

**CMS: 455387**

**AE -01**

**OOP Project**

**Title: Time Management Assistant**

**Project: Time Management Assistant**

**Introduction**:

Time Management Assistant is a user-friendly tool which helps us effectively manage and track our tasks to improve our time management skills. With this tool we can easily add, view, edit and delete tasks, check their completion status and due dates. This program is built using python and all the task data is stored in a JSON file. This tool enhances productivity and helps us stay organized and systematic in our daily activities.

**Features:**

**Task Management**:

* **Add Tasks**: Users can create new tasks by specifying a title, description, priority, and due date.
* **Edit Tasks**: Existing tasks can be modified, allowing users to update any details as needed.
* **Delete Tasks**: Users can remove tasks from their list, helping to keep their task list organized.

T**ask Viewing**:

* **View All Tasks**: Users can see a complete list of all tasks, helping them stay aware of their commitments.
* **Filter by Completion Status**: Users can view completed or incomplete tasks separately, making it easier to track progress.

C**ompletion Tracking**:

* **Mark Tasks as Completed**: Users can easily mark tasks as finished, which helps in tracking what has been accomplished.

**Due Date Notifications**:

* **Notify Due Tasks**: The application can alert users about tasks that are due today, ensuring that important deadlines are not missed.

**Search Functionality**:

* **Search Tasks**: Users can search for tasks by title or description, making it easy to find specific tasks.

**Data Persistence**:

* **JSON Data Storage**: All tasks are stored in a JSON file, allowing for data persistence across program sessions. This means users can close the application and return later to find their tasks intact.

**User-Friendly Command-Line Interface**:

* **Interactive Menu**: The application features a simple and intuitive command-line menu that guides users through various options, enhancing usability.

**Code Structure:**  
**Modules used:**

**json**: Used for reading and writing task data to and from a JSON file. This allows for data persistence, meaning tasks remain stored even after the program is closed.

**datetime**: Utilized to handle date and time operations, such as checking due dates and validating date formats.

**Classes:**

1.**Task Class**:

It represents a single task with its attributes and methods. This class has all the necessary information about a specific class.

**Attributes:**

title: The title of the task.

description: A brief description of the task (optional).

priority: The priority level of the task (High, Normal, Low).

due\_date: The date by which the task should be completed (optional).

completed: A Boolean indicating whether the task is completed.

**Methods:**

* \_\_init\_\_: Initializes a new task with the provided attributes.
* \_\_str\_\_: Returns a string representation of the task, showing its details in a user-friendly format.

2.**TaskTracker Class:**

This class manages the collection of tasks and handles user interactions. It is like the controller of the application.it manages all the tasks, interacts with the user and makes sure that the data is saved and loaded correctly.

**Attributes**:

* filename: The name of the file where tasks are stored (defaults to tasks.json).
* tasks: A list that holds all the Task objects.

**Methods**:

* \_\_init\_\_: Initializes the TaskTracker, loading tasks from the specified file.
* load\_tasks: Reads tasks from the JSON file and converts them into Task objects. Handles exceptions if the file is missing or contains invalid data.
* save\_tasks: Writes the current list of tasks back to the JSON file.
* add\_task: Creates a new Task object, adds it to the list, and saves the updated list.
* view\_tasks: Displays tasks based on their completion status.
* delete\_task: Removes a task from the list based on its index and updates the file.
* edit\_task: Allows users to modify an existing task's details.
* mark\_task\_completed: Marks a specified task as completed.
* search\_tasks: Searches for tasks based on a query in their title or description.
* notify\_due\_tasks: Checks for tasks due today and alerts the user.
* run: The main loop that presents a menu to the user and handles their input.

**Program flow chart:**

**Initialization:**

When the program starts, it initializes the TaskTracker class. This process includes loading any existing tasks from the tasks.json file, ensuring that users can pick up where they left off in their task management.

**User Interaction:**

After initialization, the program displays a menu of options to the user. This menu includes various actions such as adding tasks, viewing tasks, editing tasks, and more. The user can choose any option by entering the corresponding number.

**Processing User Choices:**

Each menu option corresponds to a specific function within the program:

* + Add Task: The user is prompted to enter details for the new task, which is then created and added to the task list. The updated list is saved back to the JSON file.
  + View Tasks: Users can view all tasks or filter them based on their completion status. This provides a clear overview of ongoing tasks.
  + Edit Task: Users can modify existing tasks by selecting a task number and providing updated details. The changes are saved immediately.
  + Delete Task: Users can remove tasks from the list. The program validates the input and updates the task list accordingly.
  + Mark Task as Completed: This feature allows users to indicate that a task has been completed, updating its status and saving the changes.
  + Search Tasks: Users can search for specific tasks by entering keywords from the task title or description. The program filters and displays relevant results.
  + Notify Due Tasks: The program checks for any tasks that are due today and alerts the user, helping them manage deadlines effectively.
  + Exit: When the user chooses to exit, the program terminates gracefully, ensuring that all changes are saved.

**Data Persistence:**

Throughout the application, any changes made to tasks are immediately saved to the tasks.json file. This ensures that data is persistent, allowing users to close and reopen the application without losing any information.

A diagram of a company

Description automatically generated

Figure flow chart of the program