**TASK MANAGEMENT APPLICATION**

**Objectives and purpose.**

**Application Objective**

Task management system aims to provide users with a simple, efficient, and user-friendly way to store, retrieve, update, and delete tasks.

Following is the primary objective of the application:

* **Task Management: To** allow users to add, update, delete, and retrieve tasks.
* **Categorization**: To enable users to categorise tasks according to its priority.
* **Goal setting**: To enable users to set goals and track them.

### **Use Case: Managing Tasks**

**Primary Actors**: User/Administrators/Team member

**Pre-condition: Users** must have an account in the task management application. User must login into the system.

**Main Flow**

* **Task creation:** Users create a new task by entering the details such as task title,description,due date at primary level.
* **Display tasks:** Display created task. Filter and sort task by name, due date, category, tags etc.
* **Task Updation:** Change the name of the task,category,due date etc. Update progress of the task.
* **Delete Task:** The user can delete the task if it is no longer needed.
* **Task completion:** Once the task is marked as completed it moves to the completed section.

**Post Condition: Tasks** are updated, completed or deleted. Users have a clear understanding of the responsibility and deadline, and user goals are met.

**Data structure and Algorithm**

1. **Data structure**

**Task objective diagram**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Task ID  Title  Description  Priority  Deadline  Status | Unique identifier (Integer)  Name (string)  Task Information(string)  Task priority(enum)  Deadline(datetime)  Task status -pending/completed(string) |

**Algorithm**

1.Insertion

2.Deletion/modification (Remove task)

3. Sorting (Sorting based on the priority, deadline)

4.Searching (Search task by ID)

**Pseudocode and flowchart for Insertion Algorithm**

FUNCTION addTask(title, description, priority,deadline)

taskID - unique ID

newTask - Create Task object with input values

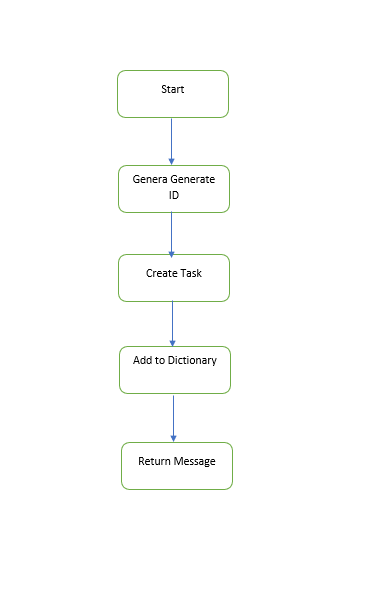
taskStorage[taskID] - newTask

RETURN “Task added successfully”

**Source code**

from datetime import datetime  
class TaskManager:  
 def \_\_init\_\_(self):  
  
 self.tasks = {}  
  
 def add\_task(self, title, description, priority, deadline):  
 try:  
 task\_id = len(self.tasks) + 1  
 deadline\_date = datetime.strptime(deadline, "%Y-%m-%d")  
 if deadline\_date < datetime.now():  
 return "Error: Deadline cannot be in the past."  
 new\_task = Task(task\_id, title, description, priority, deadline)  
 self.tasks[task\_id] = new\_task  
 return f"Task {task\_id} added successfully."  
 except ValueError:  
 return "Error: Invalid deadline format. Use YYYY-MM-DD."

**Flow chart:**



**Pseudocode and flowchart for Deletion Algorithm**

FUNCTION deleteTask(taskID):

IF taskID IN task storage:

REMOVE taskStorage[taskID]

RETURN “Task deleted successfully”

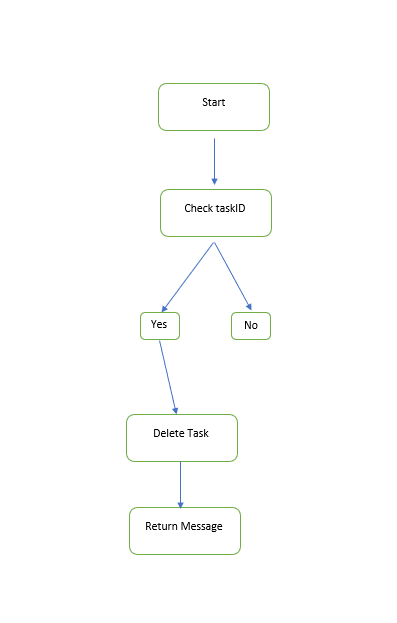
ELSE:

RETURN”Task not found”

**Source code**

def delete\_task(self, task\_id):  
  
 if task\_id in self.tasks:  
 del self.tasks[task\_id]  
 return f"Task {task\_id} deleted successfully."  
 return "Error: Task not found."

**Flow chart**



**Pseudocode and flowchart for Sorting Algorithm**

FUNCTION SortTasks(by):

taskList <= Convert taskStorage

IF by = “priority”:

SORT taskList BY task.priority DEC

ELSE IF by="deadline”:

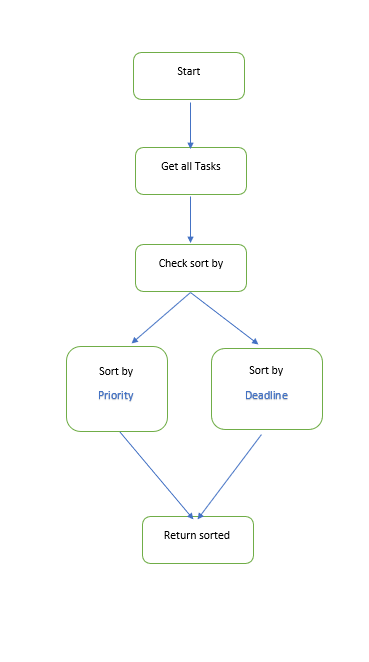
SORT taskList BY task.deadline ASC

RETURN taskList

**Source code**

def view\_tasks(self, sort\_by=None):  
  
 if not self.tasks:  
 return "No tasks available."  
 task\_list = list(self.tasks.values())  
 if sort\_by == "priority":  
 task\_list.sort(key=lambda task: task.priority, reverse=True)  
 elif sort\_by == "deadline":  
 task\_list.sort(key=lambda task: datetime.strptime(task.deadline, "%Y-%m-%d"))  
 return "\n".join(str(task) for task in task\_list)

Flow chart

****

**Pseudocode and flowchart for Search Algorithm**

FUNCTION searchTask(taskID):

IF taskID IN taskStorage[taskID]

RETURN taskStorage[taskID]

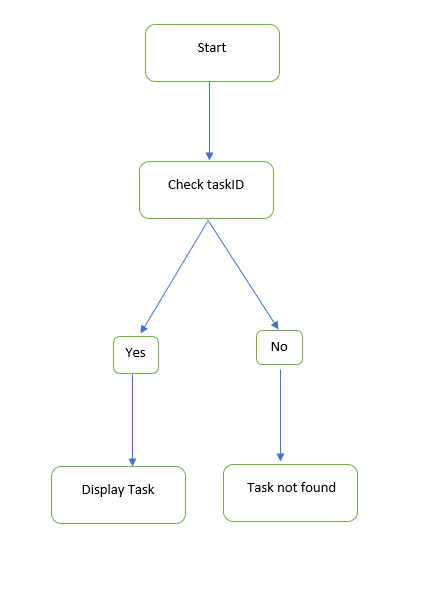
ELSE:

RETURN”Task not found”

**Source code**

def search\_task(self, task\_id):  
  
 if task\_id in self.tasks:  
 return str(self.tasks[task\_id])  
 return "Error: Task not found."

**Flow chart**



**Testing**

**Test objective**

1. Verify that all functions of task management system working as expected
2. Test the correctness of core functionalities such as insertion, deletion, sorting and searching.
3. Identify the potential bugs and errors

**Test scope**

* Task creation
* Task deletion
* Sorting task
* Searching task

**Test Cases-Insertion**

**Test case 1**: Add new task

**Test Input:** Input valid task information

Click “Add task” or run insertion function

**Expected output:** Task added successfully with unique taskID and confirmation message displayed.

**Test case 2:** Add task with no title

**Test Input:** leave the task title column empty.

Click “add task” or run insertion function

**Expected output:** Error message “Title required” displayed

**Test case 3:** Add task with past deadline

**Test input:** Add task with past deadline

Click “add task” or run insertion function

**Expected output:** Error message "Deadline cannot be in the past." displayed

**Source code**

Test adding a new task with valid information.  
  
def test\_add\_task(self):  
 response = self.manager.add\_task("Test Task","This is a test task", 3, self.future\_date)  
 self.assertEqual(response, "Task 1 added successfully.")  
 self.assertEqual(len(self.manager.tasks), 1)  
   
 # Test adding a task with no title.  
def test\_add\_task\_with\_no\_title(self):  
 response = self.manager.add\_task("","This is a test task", 3, self.future\_date)  
 self.assertEqual(response, "Error: Title required.")  
 self.assertEqual(len(self.manager.tasks), 0)  
   
 # Test adding a task with a past deadline.  
def test\_add\_task\_with\_past\_deadline(self):  
 response = self.manager.add\_task("Test Task","This is a test task", 3, self.past\_date)  
 self.assertEqual(response, "Error: Deadline cannot be in the past.")  
 self.assertEqual(len(self.manager.tasks), 0)

**Test Cases-Deletion**

**Test case 1**: Deleting existing task

**Test Input:** Input valid taskID

Click “Delete” or run delete function

**Expected output:** Amessage” Task deleted Successfully” displayed.

Source code

def test\_delete\_existing\_task(self):  
   
 # Test deleting an existing task.  
 self.manager.add\_task("Test Task", "This is a test task", 3, self.future\_date)  
 response = self.manager.delete\_task(1)  
 self.assertEqual(response, "Task 1 deleted successfully.")  
 self.assertEqual(len(self.manager.tasks), 0)

**Test Cases-Sorting**

**Test case 1:** “Sort task by priority”

**Test Input:** Add multiple tasks with different priorities.

Select “sort by priority” ‘

**Expect output:** Task are displayed descending order of priority

**Test case 2:** “Sort task by deadline”

**Test Input:** Add multiple tasks with different deadlines.

Select “sort by deadline”

**Expect output:** Tasks are displayed ascending order of deadline.

**Source code**

# Test sorting tasks by priority.  
def test\_sort\_tasks\_by\_priority(self):  
   
 self.manager.add\_task("Low Priority Task", "Description", 1, self.future\_date)  
 self.manager.add\_task("High Priority Task", "Description", 5, self.future\_date)  
 sorted\_tasks = self.manager.view\_tasks(sort\_by="priority")  
 self.assertTrue("High Priority Task" in sorted\_tasks.splitlines()[0])  
  
# Test sorting tasks by deadline.  
def test\_sort\_tasks\_by\_deadline(self):  
  
 future\_date\_1 = (datetime.now() + timedelta(days=10)).strftime("%Y-%m-%d")  
 future\_date\_2 = (datetime.now() + timedelta(days=5)).strftime("%Y-%m-%d")  
 self.manager.add\_task("Task 1", "Description", 3, future\_date\_1)  
 self.manager.add\_task("Task 2", "Description", 3, future\_date\_2)  
 sorted\_tasks = self.manager.view\_tasks(sort\_by="deadline")  
 self.assertTrue("Task 2" in sorted\_tasks.splitlines()[0])

**Test Cases-Searching**

**Test case 1:** “searching an existing task”

**Test Input:** Input valid taskID

Click “search task” or run search function

**Expect output:** Task display with unique taskID

**Test case 2:** “searching a non- existing task”

**Test Input:** Input invalid taskID

Click “search task” or run search function.

**Expect output:** Error message "Task not found" displayed

**Source code**

#Test searching for an existing task.  
def test\_search\_existing\_task(self):  
 self.manager.add\_task("Test Task", "This is a test task", 3, self.future\_date)  
 response = self.manager.search\_task(1)  
 self.assertIn("Test Task", response)  
  
#Test searching for a task that does not exist.  
def test\_search\_nonexistent\_task(self):  
 response = self.manager.search\_task(99)  
 self.assertEqual(response, "Error: Task not found.")

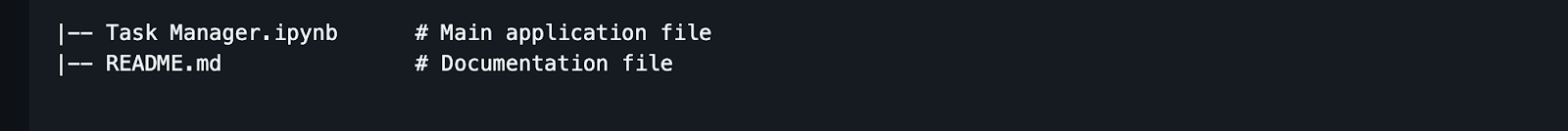
**Task Management Application- README**

The Task Manager is a simple command-line application for managing tasks. It allows users to create, delete, view, search, and mark tasks as completed. This lightweight application is ideal for organizing daily activities efficiently.

**Objective**

Task management system aims to provide users with a simple, efficient, and user-friendly way to store, retrieve, update, and delete tasks.

**Project structure**



**Features**

- **Add Tasks:** Create a new task with a title, description, priority, and deadline.

- **Delete Tasks**: Remove tasks by their unique ID.

- **View Tasks**: Display all tasks, with options to sort by priority or deadline.

- **Search Tasks**: Search for a specific task by its unique ID.

- **Complete Tasks**: Mark tasks as completed.

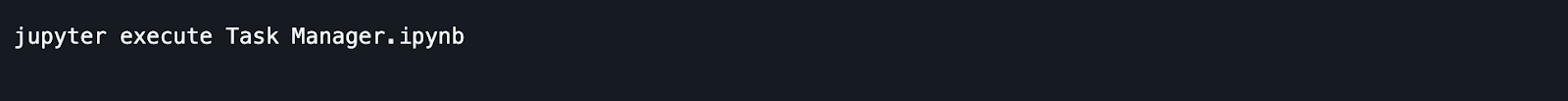
- **Simple Menu Navigation**: Intuitive menu-driven interface for easy task management.

**Prerequisites**

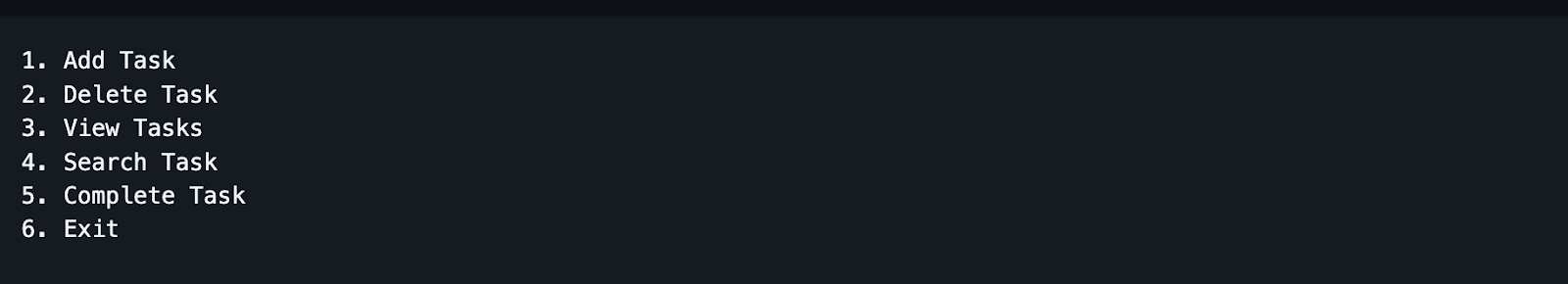
- Python 3.7 or higher.

**How to run the application**

Run the application using the following command:



After running the application, you will be prompted with following options:



**Add a new Task.**

- Select 1 for adding a new task from the main menu.

- Enter the task title

- Enter the task description.

- Enter task priority level (Low,Medium,High).

- Enter task deadline (YYYY-MM-DD).

**Delete Task.**

- Select 2 for adding a new task from the main menu.

- Enter task ID to delete.

**View Tasks.**

- Select 3 for the view task from the main menu.

- From the following menu select

1. View All Tasks.

2. View Tasks Sorted by Priority.

3. View Tasks Sorted by Deadline.

**Search Task.**

- Select 4 for the view task from the main menu.

- Enter task ID to search.

**Complete Task.**

- Select 5 for Complete Task from the main menu.

- Enter task ID to mark as complete

**Reference**

* Ashishps1(2023). awesome-low-level-design/problems/task-management-system.md at main · ashishps1/awesome-low-level-design. [online] GitHub. Available at:<https://github.com/ashishps1/awesome-low-level-design/blob/main/problems/task-management-system.md> [Accessed 16 Dec. 2024].
* BBC Bitesize. (2020). Designing an algorithm - Revision 2 - KS3 Computer Science - BBC Bitesize.:<https://www.bbc.co.uk/bitesize/guides/z3bq7ty/revision/2>.
* data, W. (2023). What data structure would be the most efficient for a task scheduler? [online] Game Development Stack Exchange. Available at:<https://gamedev.stackexchange.com/questions/208359/what-data-structure-would-be-the-most-efficient-for-a-task-scheduler> [Accessed 16 Dec. 2024].
* GrandQ (2023). Which data structure is suitable for task-based system. Stack Overflow.:<https://stackoverflow.com/questions/77211239/which-data-structure-is-suitable-for-task-based-system>.
* NocoBase (2024). [Tutorial] Chapter 2: Designing a Task Management System. [online] Medium. Available at:<https://medium.com/@nocobase/tutorial-chapter-2-designing-a-task-management-system-0e08896df49f> [Accessed 16 Dec. 2024].
* FionaG26 (2024). *GitHub - FionaG26/Task\_Manager*. [online] GitHub. Available at: <https://github.com/FionaG26/Task_Manager> [Accessed 13 Jan. 2025].
* usman010803 (2023). *GitHub - usman010803/Task\_Management\_System: This code repository contains a simple task management system implemented using Python and the PyQt5 library. The task management system provides a graphical user interface (GUI) for adding, viewing, and completing tasks.* [online] GitHub. Available at: <https://github.com/usman010803/Task_Management_System>.
* Rajath. (2024) ' Assignment -DSA -Task Management Application 1.1,