Title: Task Scheduler Project Report

Sourov,Shahriar Hossen

19-40848-2

**Project Description:**

Task Scheduler is a command-line interface (CLI) application that helps users manage their tasks. It allows users to add, view, and manage tasks by setting due dates, priorities, and descriptions. The application provides reminders for upcoming tasks and allows users to mark tasks as complete. It utilizes data structures and algorithms to handle task scheduling and organization efficiently.

**Features:**

1. Add a Task: Users can add a new task by providing details such as the task description, due date, and priority level.

2. View Task List: Users can view their task list, which displays all tasks along with their corresponding details, including description, due date, priority, and completion status.

3. Sort Tasks: Users can sort their task list based on due date or priority to prioritize and organize their tasks effectively.

4. Reminders: The application can provide reminders for upcoming tasks, notifying users of approaching due dates.

5. Complete a Task: Users can mark a task as complete, updating the completion status and removing it from the active task list.

6. Edit task: Users can edit task, change the content, date and priority.

7. Delete task: Users can delete any task.

8. Export task: Export task to text file with all information.

9. Search task: Tasks can be searched by date, content, priority, reminder wise.

10. Exit the Application: Users can choose to exit the Task Scheduler application.

**Implemented Features:**

**1.Adding a Task:**

* Users can add tasks to the scheduler by providing a task description, due date, and priority.
* The **add\_task** function checks if a task with the same name and date already exists to avoid duplicates.

def add\_task(description, due\_date, priority):

new\_task = [description, due\_date, priority]

if task\_exists(new\_task):

print("A task with the same name and date already exists. Task not added.")

else:

tasks.append(new\_task)

save\_tasks()

print("Task added successfully!")

**2.Viewing Task List**:

* The **view\_task\_list** function displays the current list of tasks, including their descriptions, due dates, and priorities

def view\_task\_list():

for idx, task in enumerate(tasks, start=1):

print(f"{idx}. {task[0]} - Due: {task[1]} - Priority: {task[2]}")

**3.Sorting Tasks:**

* Users can choose to sort tasks by priority or date using the **sort\_tasks** function.
* The sorted tasks are displayed to the user.

def sort\_tasks(sort\_option):

if sort\_option == '1':

tasks.sort(key=lambda x: x[2])

elif sort\_option == '2':

tasks.sort(key=lambda x: x[1])

else:

print("Invalid sort option. Use '1' for priority or '2' for date.")

return

save\_tasks()

print(f"Tasks sorted successfully!")

**4.Completing a Task:**

* The user can mark a task as completed by entering the task index using the **complete\_task** function.
* The completed task is removed from the list.

def complete\_task(task\_index):

if 1 <= task\_index <= len(tasks):

completed\_task = tasks.pop(task\_index - 1)

save\_tasks()

print(f"Task '{completed\_task[0]}' completed and removed from the list.")

else:

print("Invalid task index.")

**5.Editing a Task:**

* Users can edit a task by entering the task index and providing new details using the edit\_task function.
* The updated task information is saved.

def edit\_task(task\_index, new\_description, new\_due\_date, new\_priority):

if 1 <= task\_index <= len(tasks):

tasks[task\_index - 1] = [new\_description, new\_due\_date, new\_priority]

save\_tasks()

print("Task edited successfully!")

else:

print("Invalid task index.")

**6.Deleting a Task:**

* Users can delete a task by entering the task index using the **delete\_task** function.
* The deleted task is removed from the list.

def delete\_task(task\_index):

if 1 <= task\_index <= len(tasks):

deleted\_task = tasks.pop(task\_index - 1)

save\_tasks()

print(f"Task '{deleted\_task[0]}' deleted successfully.")

else:

print("Invalid task index.")

**7.Exporting Tasks:**

* The **export\_tasks** function exports the current list of tasks to the **tasks.txt** file

def export\_tasks():

with open(filename, 'w') as file:

for task in tasks:

file.write(','.join(task) + '\n')

print(f"Tasks exported to '{filename}' successfully!")

**8.Searching for a Task:**

* Users can search for tasks containing a specific keyword using the **search\_task** function.
* Matching tasks are displayed to the user.

def search\_task(keyword):

matching\_tasks = [task for task in tasks if keyword.lower() in task[0].lower()]

if matching\_tasks:

for idx, task in enumerate(matching\_tasks, start=1):

print(f"{idx}. {task[0]} - Due: {task[1]} - Priority: {task[2]}")

else:

print(f"No tasks found matching the keyword '{keyword}'.")

**User interface:**

Task Scheduler

--------------

1. Add a Task

2. View Task List

3. Sort Tasks

4. Complete a Task

5. Edit Task

6. Delete Task

7. Export Tasks

8. Search Task

9. Exit

Enter your choice (1-9): 1

Enter task description: walking

Enter due date (YYYY-MM-DD): 2023-11-15

Enter priority (1-9): 3

Task added successfully!

Task Scheduler

--------------

1. Add a Task

2. View Task List

3. Sort Tasks

4. Complete a Task

5. Edit Task

6. Delete Task

7. Export Tasks

8. Search Task

9. Exit

Enter your choice (1-9): 2

1. home - Due: 23-11-10 - Priority: 5

2. project submit - Due: 23-11-12 - Priority: 1

3. reading - Due: 23-11-14 - Priority: 1

4. walking - Due: 2023-11-15 - Priority: 3

Task Scheduler

--------------

1. Add a Task

2. View Task List

3. Sort Tasks

4. Complete a Task

5. Edit Task

6. Delete Task

7. Export Tasks

8. Search Task

9. Exit

Enter your choice (1-9): 3

Choose sorting option:

1. Sort by Priority

2. Sort by Date

Enter option (1 or 2): 1

Tasks sorted successfully!

Sorted Task List:

1. project submit - Due: 23-11-12 - Priority: 1

2. reading - Due: 23-11-14 - Priority: 1

3. walking - Due: 2023-11-15 - Priority: 3

4. home - Due: 23-11-10 - Priority: 5

Task Scheduler

--------------

1. Add a Task

2. View Task List

3. Sort Tasks

4. Complete a Task

5. Edit Task

6. Delete Task

7. Export Tasks

8. Search Task

9. Exit

Enter your choice (1-9): 4

Enter the index of the task to complete: 2

Task 'reading' completed and removed from the list.

Task Scheduler

--------------

1. Add a Task

2. View Task List

3. Sort Tasks

4. Complete a Task

5. Edit Task

6. Delete Task

7. Export Tasks

8. Search Task

9. Exit