UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

# UNIVERSITY OF WESTMINSTER

**Module:** Software Development I

**Module Leader:** Mr. Guhanathan Poravi

**Type of Assignment:** Individual Coursework

**Submission Date:** 21st April 2025

**Student Name:** Zubair Murshid

**Level: L4**

# Contents

# 1. Stage 01: List-Based Design and Basic CRUD Operations

## 1.1 Problem:

Design a task management application using lists to store tasks. Each task should include:

- Task Name
- Description
- Priority (High/Low)
- Due Date

The application must support the following CRUD operations:

1. Create – Add a new task to the list.
2. Read – View all tasks.
3. Update – Modify an existing task.
4. Delete – Remove a task from the list.

## 1.2 Problem understanding:

The problem requires building a task management application using lists to store tasks, each containing a name, description, priority, and due date. The application must support CRUD operations (Create, Read, Update, Delete) while ensuring proper indexing, input validation, and user-friendly interaction. Since no file storage is allowed, tasks exist only during program execution. The main challenge is efficiently managing task data within a list while handling user input errors gracefully.

## 1.3 Program Pseudocode:

```
BEGIN
  INITIALIZE tasklist as an empty list

  FUNCTION create_task(name, description, priority, end_date)
    CREATE task as a dictionary with keys: name, description, priority, end_date
    APPEND task to tasklist
    PRINT "Task created successfully!"
  END FUNCTION

  FUNCTION view_tasks()
```

```
    IF tasklist is empty
        PRINT "No tasks available!"
    ELSE
        PRINT "Your tasks:"
        FOR each task in tasklist
            PRINT task details (name, description, priority, due date)
END FUNCTION

FUNCTION update_task(index, name, description, priority, end_date)
    IF index is within valid range
        UPDATE task at index with new values
        PRINT "Task updated successfully!"
    ELSE
        PRINT "Invalid index"
END FUNCTION

FUNCTION delete_task(index)
    IF index is within valid range
        REMOVE task at index
        PRINT "Task deleted successfully!"
    ELSE
        PRINT "Invalid index"
END FUNCTION

WHILE True
    PRINT menu options (1. Create, 2. View, 3. Update, 4. Delete, 5. Quit)
    INPUT choice from user

    IF choice is "1"
        INPUT name, description, priority, and due date from user
```

```
    CALL create_task(name, description, priority, end_date)

ELSE IF choice is "2"
    CALL view_tasks()

ELSE IF choice is "3"
    CALL view_tasks()
    IF tasklist is not empty
        LOOP until valid index is entered
            INPUT index from user (convert to zero-based index)
            IF index is valid
                INPUT new name, description, priority, and due date
                CALL update_task(index, name, description, priority, end_date)
                BREAK LOOP
            ELSE
                PRINT "Invalid index. Try again"

ELSE IF choice is "4"
    CALL view_tasks()
    IF tasklist is not empty
        LOOP until valid index is entered
            INPUT index from user (convert to zero-based index)
            IF index is valid
                CALL delete_task(index)
                BREAK LOOP
            ELSE
                PRINT "Invalid index. Try again"

ELSE IF choice is "5"
    PRINT "Quitting Task Manager..."
```

```
        EXIT LOOP

    ELSE
        PRINT "Invalid choice. Enter a valid function"

END
```

## 1.4 Python Code

```
tasklist = []

def create_task(name,description,priority,end_date):
    task =
{"name":name,"description":description,"priority":priority,"end_date":end_date}
    tasklist.append(task)
    print('Task created successfully!\n')

def view_tasks():
    if len(tasklist)==0:
        print('No Tasks available!\n')
        return
    else:
        print('\nYour tasks:')
        for i in range(len(tasklist)):
            task=tasklist[i]
            print(f"\nTask {i+1}: {task['name']}\nDescription:
{task['description']}\nPriority: {task['priority']}\nDue date:
{task['end_date']}\n")
            #print(f"\nTask {i+1} --> {task['name']}
\nDescription:{task['description']} \nPriority: {task['priority']} \nTarget
Date:{task['end_date']}")

def update_task(index,name,description,priority,end_date):
    if 0<= index <len(tasklist):
        tasklist[index] =
{"name":name,"description":description,"priority":priority,"end_date":end_date}
        print('Task updated successfully!\n')
    else:
        print('Invalid index')
        return

def delete_task(index):
    if 0<=index<len(tasklist):
        del tasklist[index]
        print('Task deleted successfully!\n')
    else:
        print('Invalid index')
        return

if __name__=="__main__":
    while True:
        print('Personal Task Manager')
        print('1.Create Task\n2.View Tasks\n3.Update Task\n4.Delete Task\n5.Quit
program')

        choice=input('\nEnter your choice: ')
        if choice =="1":
            name=input('\nEnter name of task: ')
            description=input('Enter description: ')
```

```
            priority=input('Enter priority (High/Low): ')
            end_date=input('Enter due date: ')
            create_task(name,description,priority,end_date)

        elif choice=="2":
            view_tasks()

        elif choice=="3":
            view_tasks()
            if len(tasklist)!=0:
                while True:
                    try:
                        index=int(input('Enter index of task you wish to update:
'))-1
                        if 0<=index<len(tasklist):
                            name=input('Enter new name of task: ')
                            description=input('Enter new description: ')
                            priority=input('Enter new priority (High/Low): ')
                            end_date=input('Enter new due date: ')
                            update_task(name,description,priority,end_date)
                            break
                        else:
                            print("Index doesnt exist. Try Again\n")
                    except ValueError:
                        print("Invalid input. Enter valid index\n")

        elif choice=="4":
            view_tasks()
            if len(tasklist)!=0:
                while True:
                    try:
                        index=int(input('Enter index of task you wish to delete:
'))-1
                        if 0<=index<len(tasklist):
                            delete_task(index)
                            break
                        else:
                            print('Index doesnt exist. Try again.\n')
                    except ValueError:
                        print('Invalid input. Enter valid index\n')
        elif choice =='5':
            print('Quitting Task Manager...')
            break
        else:
            print('\nInvalid choice. Enter valid function\n')
```

# 2. Stage 02: Text File Handling for Task Persistence

## 2.1 Problem:

Design and implement a Task Manager Application that allows users to create, view, update, and delete tasks while ensuring that the tasks persist between runs. The application should use lists to store task details during execution and text file storage to save tasks permanently.

## 2.2 Problem understanding:

The task manager should support CRUD operations (Create, Read, Update, Delete). Instead of storing tasks only in memory, tasks should be saved to a text file. When the program starts, it should load previously saved tasks from the file. When tasks are added, updated, or deleted, changes should be written back to the file. The file should be structured so that each line represents a task, storing attributes like:

- Task Name
- Description
- Priority (High/Low)
- Due Date

The system should handle errors such as invalid inputs, missing files, or empty task lists. The user should have a seamless experience where tasks persist even after restarting the program.

## 2.3 Test case

| Test Case # | Input | Expected Output | Actual Output | Remarks |
|---|---|---|---|---|
| **1: Create Task** | Enter name of task (or enter q to return to main menu): Workout<br>Enter description: Gym Session<br>Enter priority (High/Low): Low<br>Enter due date: 2025-07-06<br>Task created successfully! | Task is added successfully and saved to file | Task added successfully | Test Case Pass |
| **2: View Tasks** | User selects "View Tasks" | All tasks saved are displayed | Correct tasks are displayed | Test Case Pass |
| **3: Update Task** | Enter which task you wish to update (or enter q to return to main menu): 2<br>Enter new name for task: Workout with Jake<br>Enter new description for task: Leg day<br>Enter priority (High/Low): High<br>Enter new end date for task: 2025-06-05 | Task updated successfully! | Task updated successfully! | Test Case Pass |
| **4: Delete Tasks** | Enter which task you wish to delete (or enter q to return to main menu): 3 | Task deleted successfully! | Task deleted successfully! | Test Case Pass |
| **5: Invalid input for priority while creating task** | Enter name of task (or enter q to return to main menu): Groceris<br>Enter description: Vegetables and Fruits<br>Enter priority (High/Low): Medium<br>Invalid priority. Please enter 'High' or 'Low'. | Enter priority (High/Low): Medium<br>Invalid priority. Please enter 'High' or 'Low'.<br>Enter priority (High/Low): High | Enter priority (High/Low): Medium<br>Invalid priority. Please enter 'High' or 'Low'.<br>Enter priority (High/Low): High | Test Case Pass |

| | | | | |
|---|---|---|---|---|
| | Enter priority (High/Low): High<br>Enter due date: 2025-09-04<br>Task created successfully! | | | |
| **6.Returning to main menu if user doesn't want to continue with selected function** | Enter preferred option: 1<br><br>Enter name of task (or enter q to return to main menu): q | Enter name of task (or enter q to return to main menu): q<br><br>Personal Task Manager<br>1. Create Task<br>2. View Tasks<br>3. Update Task<br>4. Delete Task<br>5. Quit<br>Enter preferred option: | Enter name of task (or enter q to return to main menu): q<br><br>Personal Task Manager<br>1. Create Task<br>2. View Tasks<br>3. Update Task<br>4. Delete Task<br>5. Quit<br>Enter preferred option: | Pass |
| **7. Entering invalid index when updating task** | Enter which task you wish to update (or enter q to return to main menu): 50 | Invalid Index. Try again<br><br>Enter which task you wish to update (or enter q to return to main menu): | Invalid Index. Try again<br><br>Enter which task you wish to update (or enter q to return to main menu): | Pass |
| **8. File doesn't exist** | Starting program while "mytasks.txt" file doesn't exist | 1. Program initializes with an empty task list.<br>2. Task can be created and saved to the newly created file. | Program runs, handles missing file, and creates a new task successfully. | |

## 2.4 Test case Screenshots

```
Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option: 1

Enter name of task (or enter q to return to main menu): Gym Session
Enter description: Workout
Enter priority (High/Low): Low
Enter due date: 2025-08-07
Task created successfully!

Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option:
```

*Figure 1: Creating Task*

```
Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option: 2

Your Tasks:

Task 1: Gym Session
Description: Workout
Priority: low
Due date: 2025-08-07

Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option:
```

*Figure 2: Viewing Tasks*

```
Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option: 3

Your Tasks:

Task 1: Gym Session
Description: Workout
Priority: low
Due date: 2025-08-07

Enter which task you wish to update (or enter q to return to main menu): 1
Enter new name for task: Workout with Jake
Enter new description for task: Leg day Sessions
Enter priority (High/Low): High
Enter new end date for task: 2025-09-05
Task updated successfully!

Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option:
```

*Figure 3: Updating Tasks*

```
Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option: 4

Your Tasks:

Task 1: Workout with Jake
Description: Leg day Sessions
Priority: high
Due date: 2025-09-05

Enter which task you wish to delete (or enter q to return to main menu): 1
Task deleted successfully!

Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option:
```

*Figure 4: Deleting Tasks*

```
Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option: 1

Enter name of task (or enter q to return to main menu): Groceries
Enter description: Fruits and Vegetable
Enter priority (High/Low): Medium
Invalid priority. Please enter 'High' or 'Low'.
Enter priority (High/Low):
```

*Figure 5: Error message when invalid priority is input*

```
Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option: 1

Enter name of task (or enter q to return to main menu): q

Personal Task Manager
1. Create Task
2. View Tasks
3. Update Task
4. Delete Task
5. Quit
Enter preferred option:
```

*Figure 6: Returning to main menu when user wishes to cancel operation*

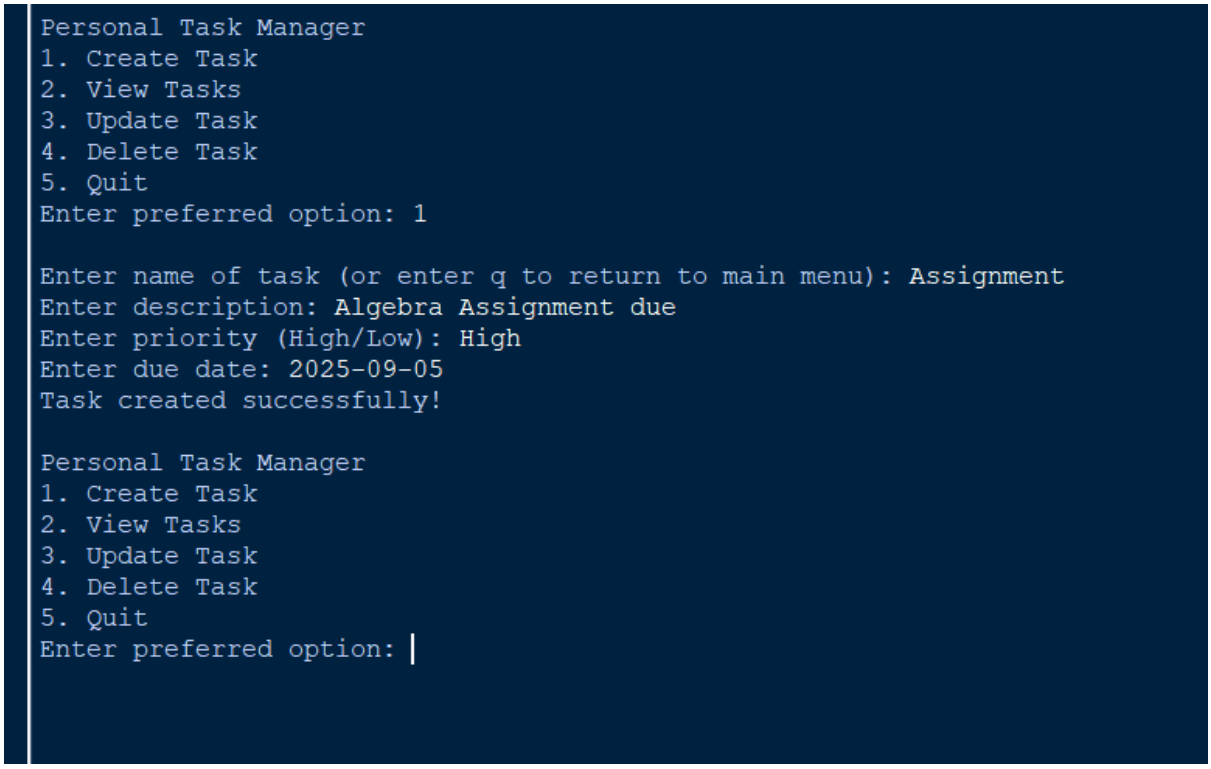



*Figure 7: Program not crashing when FileNotFoundError is passed*

# 3. Stage 3: Using Dictionaries and JSON File Handling

## 3.1 Objective:

Transition from list-based storage to dictionaries, using JSON for structured data persistence. Implement loading and saving tasks to a JSON file.

## 3.2 Objective understanding:

The task involves transitioning from list-based storage to using dictionaries for managing tasks, where each task is represented by key-value pairs (e.g., task name, description, priority, and due date). This change will allow for more structured data management. Additionally, tasks will be saved and loaded using JSON for data persistence, enabling tasks to be stored in a file and retrieved between program executions. The main goal is to implement functionality to load tasks from a JSON file when the program starts and save tasks to the JSON file when changes occur, ensuring task data persists across sessions.

### 3.3 Sample JSON File

```
[
    {
        "name": "Shopping",
        "description": "Going Grocery Shopping on Monday",
        "priority": "low",
        "due_date": "2025-04-21"
    },
    {
        "name": "Breakfast",
        "description": "Going out for breakfast",
        "priority": "high",
        "due_date": "2025-04-20"
    },
    {
        "name": "Finish CS Assignment",
        "description": "Complete remaining pseudocode of project",
        "priority": "high",
        "due_date": "2025-04-22"
    },
    {
        "name": "Dentist Appointment",
        "description": "Call to book a check up for next week",
        "priority": "low",
        "due_date": "2025-04-25"
    },
    {
        "name": "Update Resume and Linkedin profile",
        "description": "Add recent sd projects",
        "priority": "high",
        "due_date": "2025-04-20"
    }
]
```

*Figure 8: Screenshot Sample of JSON File*

## 3.4 Test case

| Test Case # | Input | Expected Output | Actual Output | Remarks |
|---|---|---|---|---|
| **1: Loading Task from JSON (Valid Data)** | User selects "View Tasks" | All tasks saved are displayed | Correct tasks are displayed | Test Case Pass |
| **2: Loading Task from JSON (Invalid Data in JSON File)** | User selects "View Tasks" | No Tasks available! (As data in json file is invalid) | No Tasks available! | Test Case Pass |
| **3: Save to JSON (Create task)** | Enter name of task (or enter exit to return to menu): Breakfast<br>Enter description: Going out for breakfast<br>Enter priority (High/Low): High<br>Enter due date: 2025-04-20<br>Task created successfully! | Task created successfully! | Task created successfully! | Test Case Pass |
| **4: Save to JSON (Update Task)** | Enter index of task you wish to update (or enter exit to return to menu): 2<br><br>Enter new name of task: Shopping<br>Enter description: Going Grocery Shopping on Monday<br>Enter priority (High/Low): Low | Task updated successfully! | Task updated successfully! | Test Case Pass |

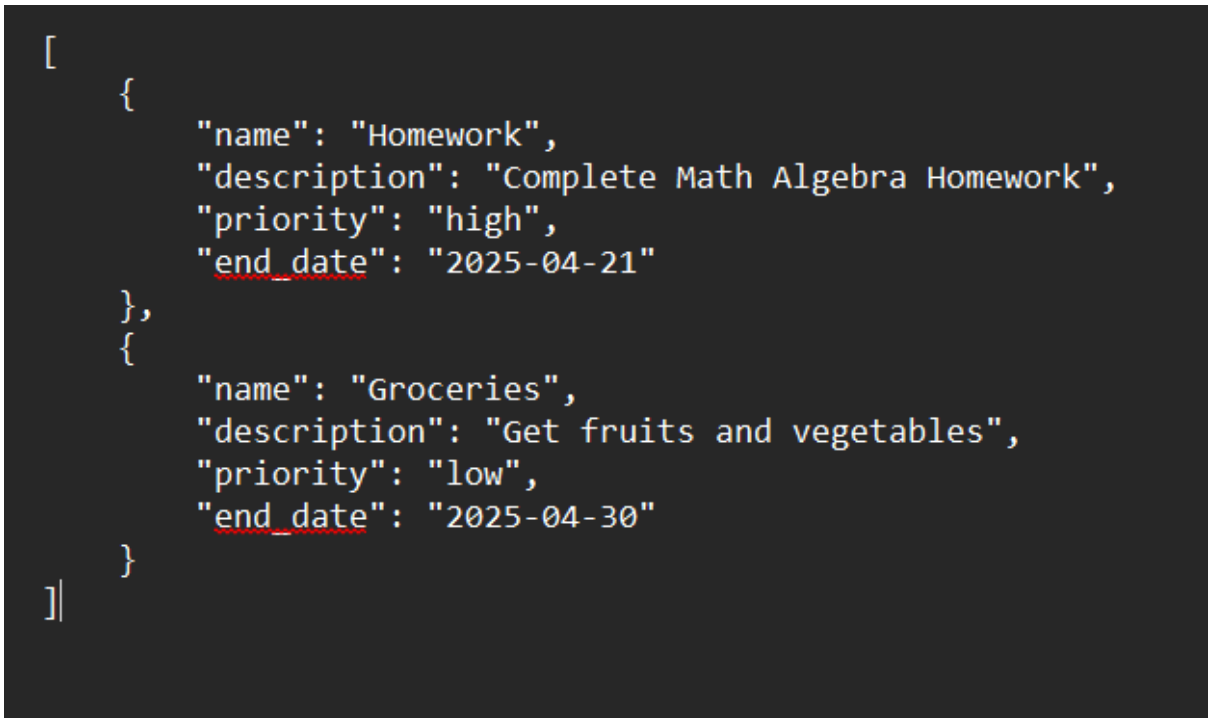
| | | | | |
|---|---|---|---|---|
| | Enter due date: 2025-04-21<br>Task updated successfully! | | | |

## 3.5 Test case Screenshots

### 3.5.1 Test case 01

```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: G:\My Drive\Sem1_SD1\CW\Stage 03\Stage03.py ============
Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice: 2

Your tasks:

Task 1: Homework
Description: Complete Math Algebra Homework
Priority: high
Due date: 2025-04-21

Task 2: Groceries
Description: Get fruits and vegetables
Priority: low
Due date: 2025-04-30

Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice:
```

*Figure 9:Loading Task from JSON (Valid*



```
[
    {
        "name": "Homework",
        "description": "Complete Math Algebra Homework",
        "priority": "high",
        "end_date": "2025-04-21"
    },
    {
        "name": "Groceries",
        "description": "Get fruits and vegetables",
        "priority": "low",
        "end_date": "2025-04-30"
    }
]
```

*Figure 10: JSON File Screenshot*

### 3.5.2 Test Case 02

```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: G:\My Drive\Sem1_SD1\CW\Stage 03 - Copy\Stage03.py =========
Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice: 2
No Tasks available!

Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice:
```

*Figure 11: Loading Task from JSON (Invalid)*

```
[
    {
        "name": "Homework",
        "description": "Complete Math Algebra Homework",
    }
]
```

*Figure 12: JSON File Screenshot*

### 3.5.3 Test Case 03

```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

============= RESTART: G:\My Drive\Sem1_SD1\CW\Stage 03\Stage03.py =============
Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice: 1

Enter name of task (or enter exit to return to menu): Breakfast
Enter description: Going out for breakfast
Enter priority (High/Low): High
Enter due date: 2025-04-20
Task created successfully!

Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice:
```

*Figure 13: Save to JSON (Create task)*

```
[
    {
        "name": "Homework",
        "description": "Complete Math Algebra Homework",
        "priority": "high",
        "end_date": "2025-04-21"
    },
    {
        "name": "Groceries",
        "description": "Get fruits and vegetables",
        "priority": "low",
        "end_date": "2025-04-30"
    },
    {
        "name": "Breakfast",
        "description": "Going out for breakfast",
        "priority": "high",
        "end_date": "2025-04-20"
    }
]
```


*Figure 14: JSON File Screenshot*

### 3.5.4 Test Case 04

```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: G:\My Drive\Sem1_SD1\CW\Stage 03\Stage03.py ==========
Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice: 3

Your tasks:

Task 1: Homework
Description: Complete Math Algebra Homework
Priority: high
Due date: 2025-04-21

Task 2: Groceries
Description: Get fruits and vegetables
Priority: low
Due date: 2025-04-30

Task 3: Breakfast
Description: Going out for breakfast
Priority: high
Due date: 2025-04-20

Enter index of task you wish to update (or enter exit to return to menu): 2

Enter new name of task: Shopping
Enter description: Going Grocery Shopping on Monday
Enter priority (High/Low): Low
Enter due date: 2025-04-21
Task updated successfully!

Personal Task Manager
1.Create Task
2.View Tasks
3.Update Task
4.Delete Task
5.Quit program

Enter your choice:
```

*Figure 15: Save to JSON (Update Task)*

```
[
    {
        "name": "Homework",
        "description": "Complete Math Algebra Homework",
        "priority": "high",
        "end_date": "2025-04-21"
    },
    {
        "name": "Shopping",
        "description": "Going Grocery Shopping on Monday",
        "priority": "low",
        "end_date": "2025-04-21"
    },
    {
        "name": "Breakfast",
        "description": "Going out for breakfast",
        "priority": "high",
        "end_date": "2025-04-20"
    }
]
```


*Figure 16: JSON File Screenshot*

# 4. Stage 4: Tkinter GUI for Viewing, Searching, and Sorting Tasks

## 4.1 Objective

Implement a Tkinter GUI to display all tasks in a tabular format. Enable users to filter tasks by criteria such as name, priority, or due date. Allow users to sort tasks by clicking on column headers (e.g., by name, priority, or due date).

## 4.2 Objective Understanding

The problem involves developing a Tkinter-based graphical user interface that allows users to view tasks in a structured, table-like format. The application should support filtering tasks based on specific criteria such as name, priority, or due date, enhancing searchability and usability. Additionally, users should be able to sort tasks dynamically by clicking on column headers, enabling better organization and task management. The key challenge is integrating these interactive features within the Tkinter framework while maintaining smooth and responsive UI behavior.

## 4.3GUI Interface Explanation

The Task Manager app uses a GUI (Graphical User Interface) built with “tkinter” and “ttk” modules to view tasks. Below is a breakdown of what each component does:

### 4.3.1 GUI Layout Map

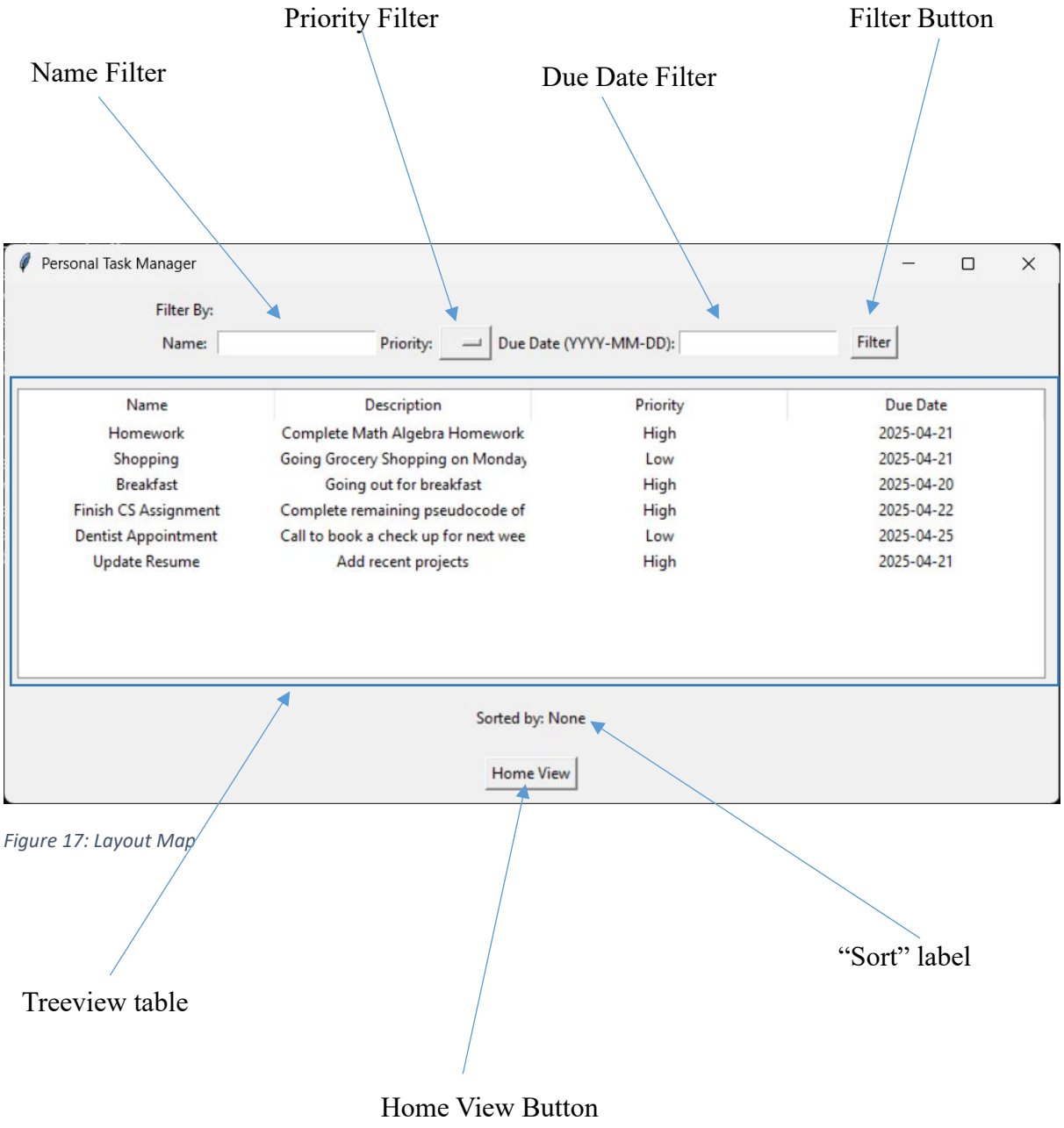



*Figure 17: Layout Map*

### 4.3.2 Filter and search Section

- Name Filter (Entry box): User can type keywords to filter tasks by name.
- Priority Filter (Dropdown menu): User can select High or Low to show only tasks of that priority. This could be left blank also.
- Due Date Filter (Entry box): Filter by a specific date. This should match the format used when creating the task as indicated along with the title.
- Filter Button: Applies all filters to show user tasks only of their preference.

### 4.3.3 Task Table

- The Treeview table shows the name, description, priority, and due date of tasks.
- Column headers are clickable to sort tasks between ascending and descending order.

### 4.3.4 Sort Information:

- “Sort” label placed below the task table displays the current sorting criteria (e.g., "Sorted by: Name").

### 4.3.5 Home View Button:

- A button labeled "Home View" that resets all filter and sort settings to their default values, reloading all tasks from the JSON file.

### 4.3.6 Other features

- Table rows are auto centered and expandable.
- Treeview layout automatically adjusts to window resizing.

## 4.4 Screenshots of Tkinter GUI

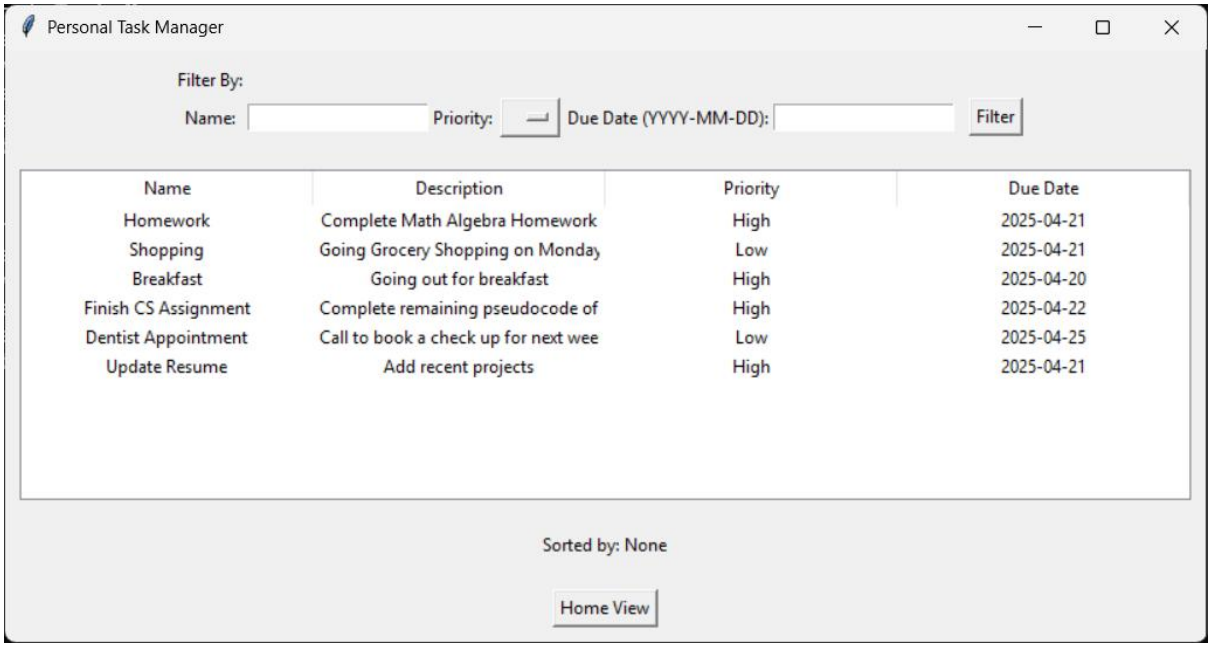



*Figure 18: Default App View (on start)*

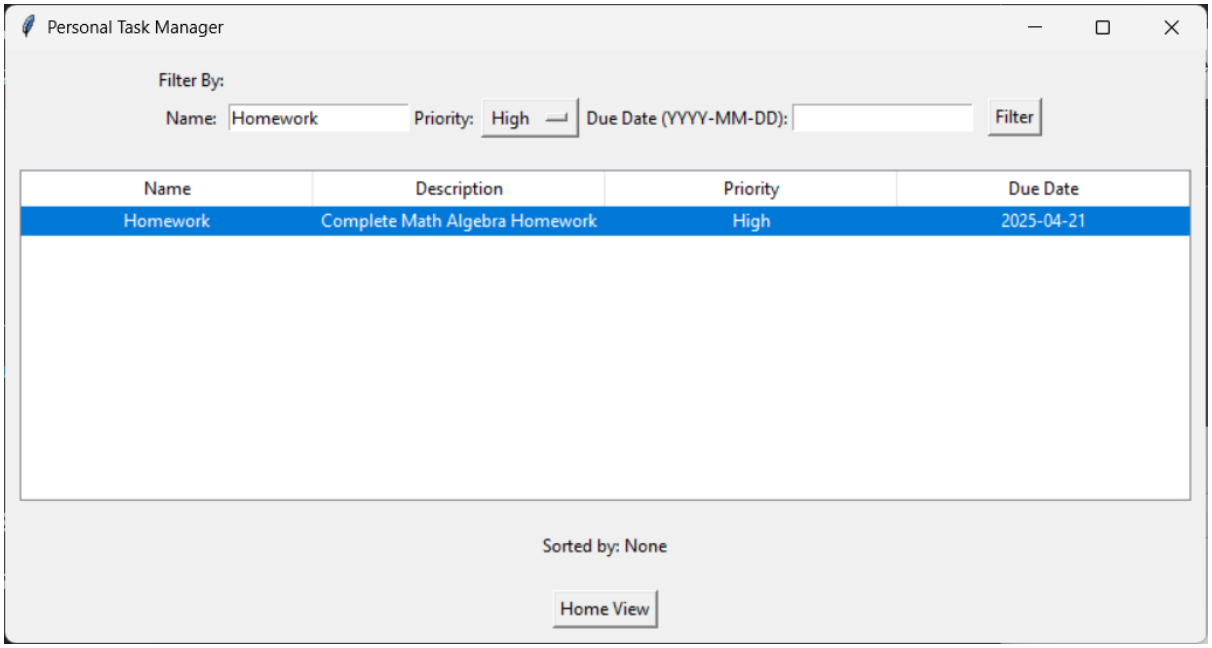



*Figure 19: Task list after applying filter.*

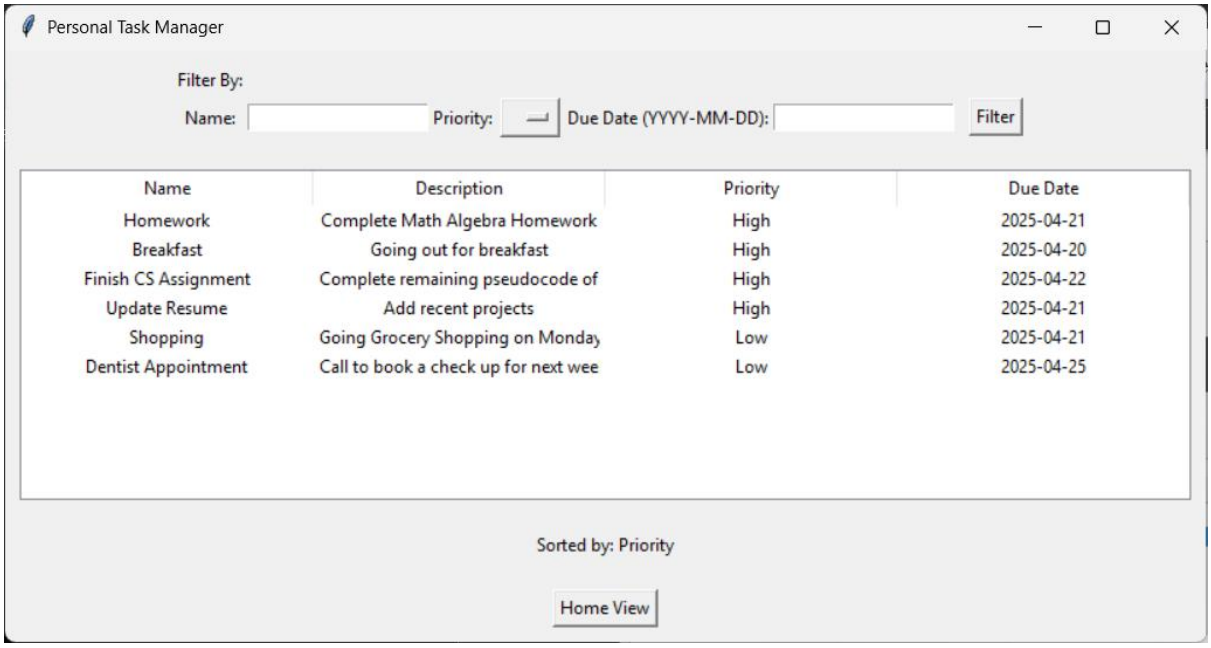



*Figure 20: Task list after sorting by priority.*

## 4.5 Instructions for Search and Sort functionalities

### 4.5.1 Filter Tasks

- Name Filter:
    - Enter a partial or full task name in the Name field.
    - Click the Filter button to view tasks that match the entered name.
- Priority Filter:
    - Select a priority from the dropdown (either "High" or "Low").
    - Click the Filter button to view tasks that match the selected priority.
- Due Date Filter:
    - Enter a due date in the format YYYY-MM-DD in the Due Date field.
    - Click the Filter button to view tasks with the specified due date.

### 4.5.2 Sort Tasks

- Click on any column header in the task table (Name, Priority, or Due Date) to sort tasks by that column.
- The first click sorts the tasks in ascending order (e.g., alphabetically for the "Name" column).
- Subsequent clicks toggle the sort order between ascending and descending.
- The current sorting criteria will be displayed at the top of the window (e.g., "Sorted by: Name").

### 4.5.3 Reset Filters and Sorting

- Click the Home View button to clear all filter inputs (Name, Priority, Due Date).
- All tasks are reloaded from the JSON file and displayed, with the sorting reset to ascending order.

## 4.6 Tips for Best User Experience

1. Filter and Sort Together: Combine filters and sorting to narrow down the task list further. For example, you can filter tasks by High Priority and then sort them by Due Date to see your high-priority tasks sorted chronologically.
2. Sorting Order: If you are viewing tasks by priority or due date, consider sorting the column to ensure tasks are displayed in a meaningful order (e.g., sorting by due date helps in planning your tasks by urgency).
3. Clear Filters: If you are unsure which filters to apply or if you want to start over, use the Home View button to reset everything and view all tasks.
4. Task Completeness: Ensure that each task is properly filled in with a name, description, priority, and due date. Empty fields or incomplete tasks can make it harder to filter or sort effectively.
5. Due Date Format: Ensure that due dates are entered correctly in the YYYY-MM-DD format when filtering. Invalid or incorrect formats may result in no tasks being displayed.