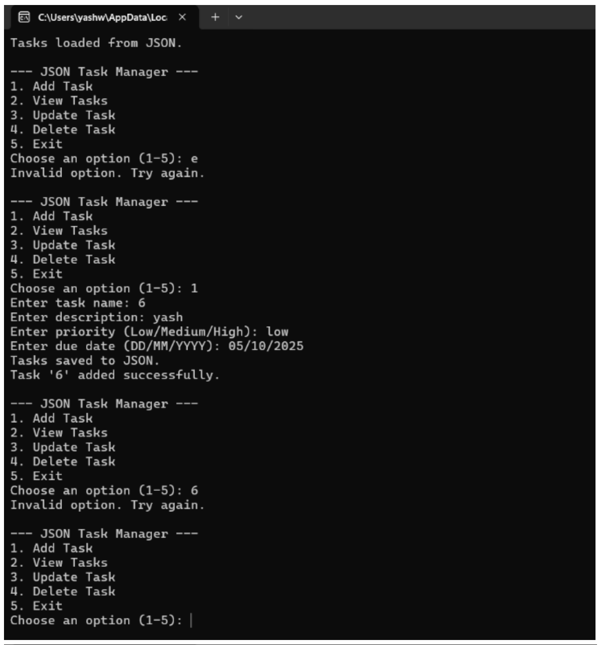
THE CODE – Task Manager

```
import json
from datetime import datetime

# JSON file path
json_file = "tasks.json"

# List of task dictionaries
tasks = []

# --------- JSON File Handling ---------
def load_tasks_from_json():
  global tasks
  try:
    with open(json_file, 'r') as file:
      tasks = json.load(file)
    print("Tasks loaded from JSON.")
  except FileNotFoundError:
    print("No existing task file found. Starting with an empty task list.")
    tasks = []
  except json.JSONDecodeError:
    print("JSON file is malformed. Starting fresh.")
    tasks = []

def save_tasks_to_json():
  try:
    with open(json_file, 'w') as file:
      json.dump(tasks, file, indent=4)
    print("Tasks saved to JSON.")
  except Exception as e:
        print(f"Error saving to JSON: {e}")


# --------- CRUD Operations ---------
def add_task():
    try:
        name = input("Enter task name: ")
        description = input("Enter description: ")

        # Validate priority
        while True:
            priority = input("Enter priority (Low/Medium/High): ").capitalize()
            if priority in ["Low", "Medium", "High"]:
                break
            else:
                print("Invalid priority. Please enter Low, Medium, or High.")

        # Validate date input
        while True:
            due_date = input("Enter due date (DD/MM/YYYY): ")
            try:
                due_date_obj = datetime.strptime(due_date, "%d/%m/%Y").date()
                if due_date_obj < datetime.today().date():
                    print("Due date cannot be in the past.")
                else:
                    break
            except ValueError:
                print("Invalid date format. Use DD/MM/YYYY.")

        task = {
            "name": name,
            "description": description,
            "priority": priority,
            "due_date": due_date
        }

        tasks.append(task)
        save_tasks_to_json()
        print(f"Task '{name}' added successfully.")
    except Exception as e:
        print(f"Error adding task: {e}")

def view_tasks():
    if not tasks:
        print("No tasks to display.")
        return
    for idx, task in enumerate(tasks, start=1):
        print(f"\nTask #{idx}")
        for key, value in task.items():
            print(f"{key.capitalize()}: {value}")
        print("-" * 20)

def update_task():
    view_tasks()
    if not tasks:
        return
    try:
        task_number = int(input("Enter task number to update: "))
        if 1 <= task_number <= len(tasks):
            task = tasks[task_number - 1]
            print(f"Updating task: {task['name']}")

            new_name = input("Enter new name: ")
```

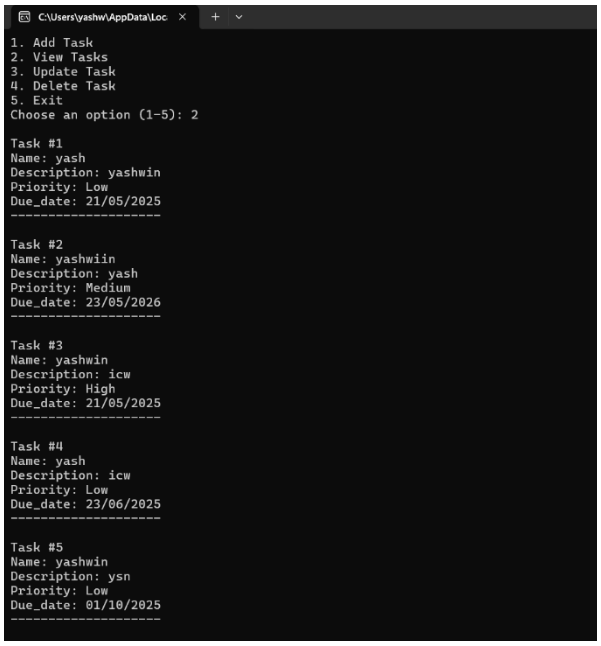
```
if new_name:
    task['name'] = new_name

new_description = input("Enter new description: ")
if new_description:
    task['description'] = new_description

# Validate priority input
while True:
    new_priority = input("Enter new priority (Low/Medium/High): ")
    if new_priority == "":
        break  # Keep existing
    if new_priority.capitalize() in ["Low", "Medium", "High"]:
        task['priority'] = new_priority.capitalize()
        break
    else:
        print("Invalid priority. Please enter Low, Medium, or High.")

# Validate due date
while True:
    new_due_date = input("Enter new due date (DD/MM/YYYY): ")
    if new_due_date == "":
        break  # Keep existing
    try:
        due_date_obj = datetime.strptime(new_due_date, "%d/%m/%Y").date()
        if due_date_obj < datetime.today().date():
            print("Due date cannot be in the past.")
        else:
            task['due_date'] = new_due_date
            break
    except ValueError:
```

```
                    print("Invalid date format. Use DD/MM/YYYY.")

            save_tasks_to_json()
            print("Task updated successfully.")
        else:
            print("Invalid task number.")
    except ValueError:
        print("Please enter a valid number.")

def delete_task():
    view_tasks()
    if not tasks:
        return
    try:
        task_number = int(input("Enter task number to delete: "))
        if 1 <= task_number <= len(tasks):
            removed_task = tasks.pop(task_number - 1)
            save_tasks_to_json()
            print(f"Task '{removed_task['name']}' deleted.")
        else:
            print("Invalid task number.")
    except ValueError:
        print("Please enter a valid number.")

# --------- Main Program ---------
if __name__ == "__main__":
    load_tasks_from_json()

    while True:
        print("\n--- JSON Task Manager ---")
        print("1. Add Task")
print("2. View Tasks")
print("3. Update Task")
print("4. Delete Task")
print("5. Exit")

choice = input("Choose an option (1-5): ")

if choice == '1':
  add_task()
elif choice == '2':
  view_tasks()
elif choice == '3':
  update_task()
elif choice == '4':
  delete_task()
elif choice == '5':
  print("Exiting... Goodbye!")
  break
else:
  print("Invalid option. Try again.")
```

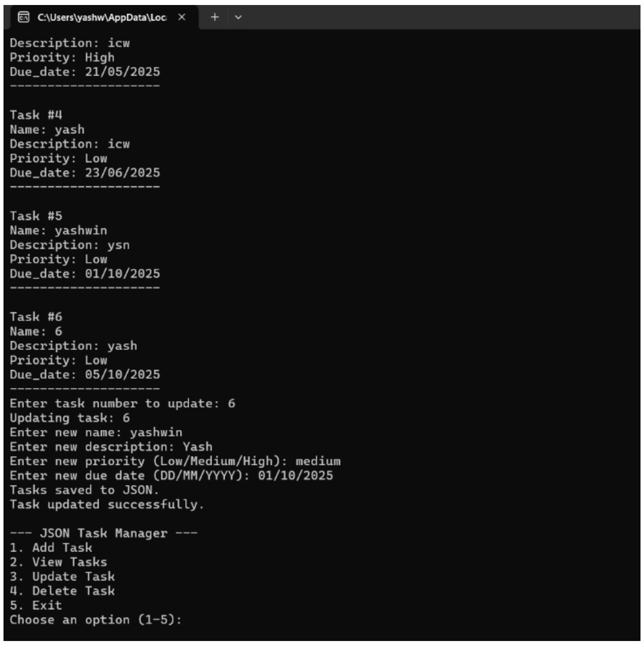
Screenshots of testing



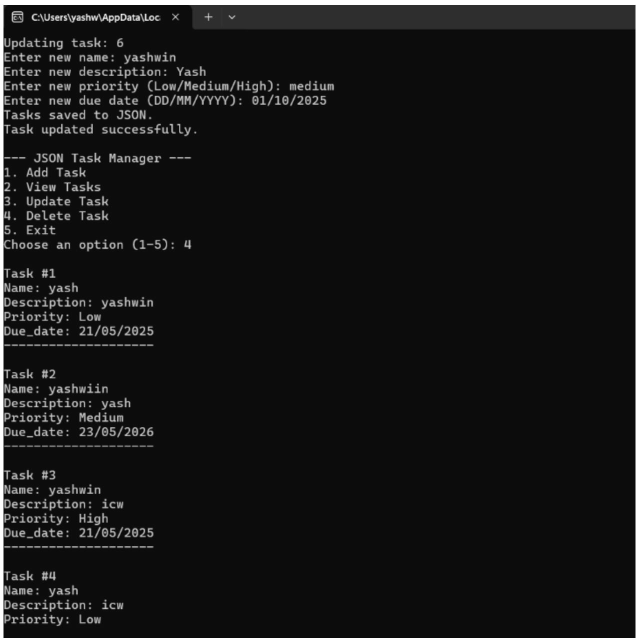
```
Tasks loaded from JSON.

--- JSON Task Manager ---
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5): e
Invalid option. Try again.

--- JSON Task Manager ---
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5): 1
Enter task name: 6
Enter description: yash
Enter priority (Low/Medium/High): low
Enter due date (DD/MM/YYYY): 05/10/2025
Tasks saved to JSON.
Task '6' added successfully.

--- JSON Task Manager ---
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5): 6
Invalid option. Try again.

--- JSON Task Manager ---
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5):
```



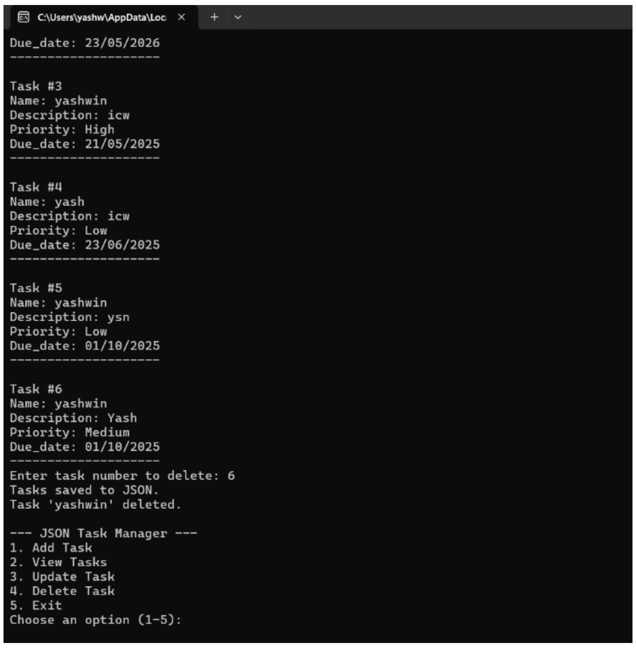
```
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5): 2

Task #1
Name: yash
Description: yashwin
Priority: Low
Due_date: 21/05/2025
--------------------

Task #2
Name: yashwiin
Description: yash
Priority: Medium
Due_date: 23/05/2026
--------------------

Task #3
Name: yashwin
Description: icw
Priority: High
Due_date: 21/05/2025
--------------------

Task #4
Name: yash
Description: icw
Priority: Low
Due_date: 23/06/2025
--------------------

Task #5
Name: yashwin
Description: ysn
Priority: Low
Due_date: 01/10/2025
--------------------
```



```
Description: icw
Priority: High
Due_date: 21/05/2025
--------------------

Task #4
Name: yash
Description: icw
Priority: Low
Due_date: 23/06/2025
--------------------

Task #5
Name: yashwin
Description: ysn
Priority: Low
Due_date: 01/10/2025
--------------------

Task #6
Name: 6
Description: yash
Priority: Low
Due_date: 05/10/2025
--------------------
Enter task number to update: 6
Updating task: 6
Enter new name: yashwin
Enter new description: Yash
Enter new priority (Low/Medium/High): medium
Enter new due date (DD/MM/YYYY): 01/10/2025
Tasks saved to JSON.
Task updated successfully.

--- JSON Task Manager ---
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5):
```



```
Updating task: 6
Enter new name: yashwin
Enter new description: Yash
Enter new priority (Low/Medium/High): medium
Enter new due date (DD/MM/YYYY): 01/10/2025
Tasks saved to JSON.
Task updated successfully.

--- JSON Task Manager ---
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5): 4

Task #1
Name: yash
Description: yashwin
Priority: Low
Due_date: 21/05/2025
--------------------

Task #2
Name: yashwiin
Description: yash
Priority: Medium
Due_date: 23/05/2026
--------------------

Task #3
Name: yashwin
Description: icw
Priority: High
Due_date: 21/05/2025
--------------------

Task #4
Name: yash
Description: icw
Priority: Low
```



```
Due_date: 23/05/2026
--------------------

Task #3
Name: yashwin
Description: icw
Priority: High
Due_date: 21/05/2025
--------------------

Task #4
Name: yash
Description: icw
Priority: Low
Due_date: 23/06/2025
--------------------

Task #5
Name: yashwin
Description: ysn
Priority: Low
Due_date: 01/10/2025
--------------------

Task #6
Name: yashwin
Description: Yash
Priority: Medium
Due_date: 01/10/2025
--------------------
Enter task number to delete: 6
Tasks saved to JSON.
Task 'yashwin' deleted.

--- JSON Task Manager ---
1. Add Task
2. View Tasks
3. Update Task
4. Delete Task
5. Exit
Choose an option (1-5):
```

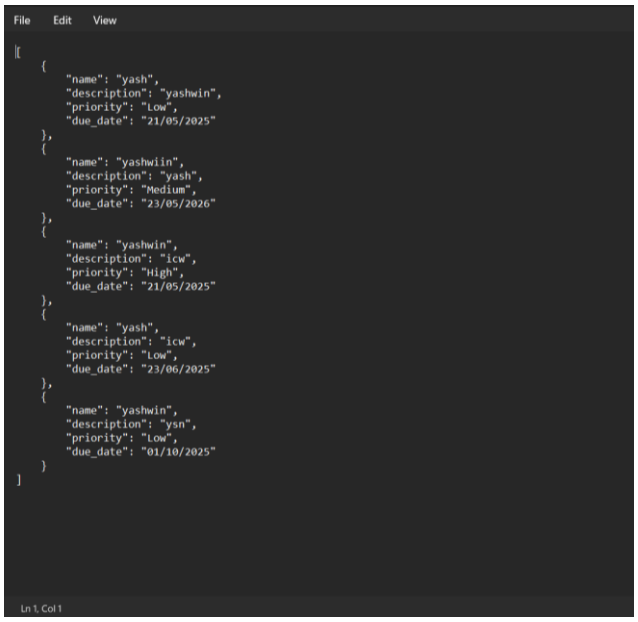
The Code – Tkinter GUI

```
import tkinter as tk

from tkinter import ttk

import json

from datetime import datetime


# JSON file path

json_file = "tasks.json"


def load_tasks():

    try:

        with open(json_file, "r") as f:

            return json.load(f)

    except (FileNotFoundError, json.JSONDecodeError):

        return []


def filter_tasks(name_query, priority_filter, due_date_filter, task_list):

    name_query = name_query.lower().strip()
    filtered = []

    for task in task_list:
        match = True
        if name_query and name_query not in task['name'].lower():
            match = False
        if priority_filter != "All" and task['priority'] != priority_filter:
            match = False
        if due_date_filter:
            try:
                filter_date = datetime.strptime(due_date_filter, "%d/%m/%Y").date()
                task_date = datetime.strptime(task["due_date"], "%d/%m/%Y").date()
                if task_date != filter_date:
                    match = False
            except ValueError:
                continue
        if match:
            filtered.append(task)

    return filtered

def sort_by_name(task_list):
    return sorted(task_list, key=lambda t: t['name'].lower())

def sort_by_priority(task_list):
    priority_order = {"High": 1, "Medium": 2, "Low": 3}
    return sorted(task_list, key=lambda t: priority_order.get(t["priority"], 99))

def sort_by_due_date(task_list):
    return sorted(task_list, key=lambda t: datetime.strptime(t["due_date"], "%d/%m/%Y"))
```

```
class TaskManagerApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Personal Task Manager")
        self.root.geometry("900x600")
        self.tasks = load_tasks()
        self.filtered_tasks = self.tasks.copy()
        self.setup_gui()
        self.populate_tree(self.filtered_tasks)

    def setup_gui(self):
        filter_frame = tk.LabelFrame(self.root, text="Filter Tasks")
        filter_frame.pack(pady=10, fill=tk.X, padx=10)

        tk.Label(filter_frame, text="Name:").grid(row=0, column=0, padx=5, pady=5, sticky="e")
        self.name_entry = tk.Entry(filter_frame)
        self.name_entry.grid(row=0, column=1, padx=5, pady=5)

        tk.Label(filter_frame, text="Priority:").grid(row=0, column=2, padx=5, pady=5, sticky="e")
        self.priority_filter = ttk.Combobox(filter_frame, values=["All", "High", "Medium", "Low"],
state="readonly")
        self.priority_filter.current(0)
        self.priority_filter.grid(row=0, column=3, padx=5, pady=5)

        tk.Label(filter_frame, text="Due Date (DD/MM/YYYY):").grid(row=0, column=4, padx=5,
pady=5, sticky="e")
        self.due_date_entry = tk.Entry(filter_frame)
        self.due_date_entry.grid(row=0, column=5, padx=5, pady=5)

        self.filter_button = tk.Button(filter_frame, text="Filter", command=self.perform_filter,
bg="#add8e6")
        self.filter_button.grid(row=0, column=6, padx=10)
        sort_frame = tk.Frame(self.root)

        sort_frame.pack(pady=5)


        self.sort_buttons = {}


        self.sort_buttons["name"] = tk.Button(sort_frame, text="Sort by Name",
command=self.sort_name)

        self.sort_buttons["name"].pack(side=tk.LEFT, padx=10)


        self.sort_buttons["priority"] = tk.Button(sort_frame, text="Sort by Priority",
command=self.sort_priority)

        self.sort_buttons["priority"].pack(side=tk.LEFT, padx=10)


        self.sort_buttons["due"] = tk.Button(sort_frame, text="Sort by Due Date",
command=self.sort_due_date)

        self.sort_buttons["due"].pack(side=tk.LEFT, padx=10)


        tree_frame = tk.Frame(self.root)

        tree_frame.pack(pady=10, fill=tk.BOTH, expand=True)


        tree_scroll_y = tk.Scrollbar(tree_frame, orient=tk.VERTICAL)

        tree_scroll_y.pack(side=tk.RIGHT, fill=tk.Y)


        self.tree = ttk.Treeview(

            tree_frame,

            columns=("Name", "Description", "Priority", "Due Date"),

            show="headings",

            yscrollcommand=tree_scroll_y.set

        )

        tree_scroll_y.config(command=self.tree.yview)
```

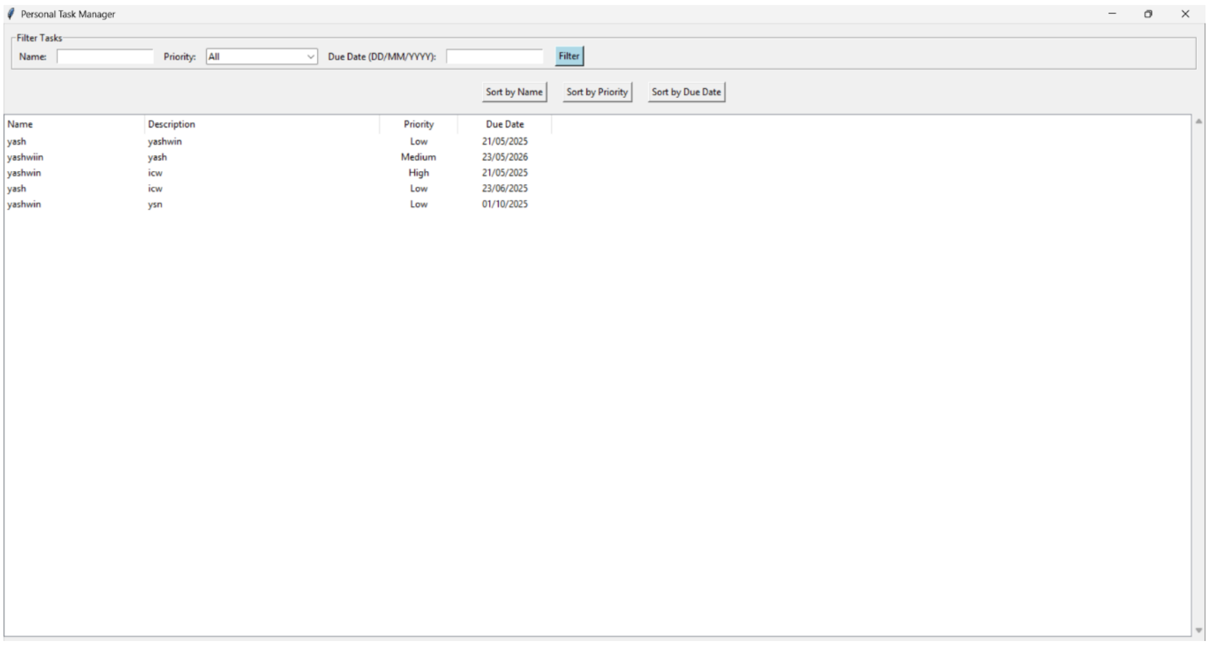
```
    self.tree.heading("Name", text="Name", anchor="w")
    self.tree.heading("Description", text="Description", anchor="w")
    self.tree.heading("Priority", text="Priority", anchor="center")
    self.tree.heading("Due Date", text="Due Date", anchor="center")

    self.tree.column("Name", anchor="w", width=180, stretch=False)
    self.tree.column("Description", anchor="w", width=300, stretch=False)
    self.tree.column("Priority", anchor="center", width=100, stretch=False)
    self.tree.column("Due Date", anchor="center", width=120, stretch=False)

    self.tree.pack(fill=tk.BOTH, expand=True)

  def populate_tree(self, task_list):
    self.tree.delete(*self.tree.get_children())
    for task in task_list:
      self.tree.insert("", tk.END, values=(
        task["name"],
        task["description"],
        task["priority"],
        task["due_date"]
      ))

  def perform_filter(self):
    self.reset_button_colors()
    self.filter_button.config(bg="#add8e6")  # Keep filter button blue
    name_query = self.name_entry.get()
    priority = self.priority_filter.get()
    due_date = self.due_date_entry.get()

    self.filtered_tasks = filter_tasks(name_query, priority, due_date, self.tasks)
    self.populate_tree(self.filtered_tasks)
    def reset_button_colors(self):
        for btn in self.sort_buttons.values():
            btn.config(bg="SystemButtonFace")
        self.filter_button.config(bg="#add8e6")  # Reset filter button to light blue

    def sort_name(self):
        self.reset_button_colors()
        self.filtered_tasks = sort_by_name(self.filtered_tasks)
        self.populate_tree(self.filtered_tasks)
        self.sort_buttons["name"].config(bg="#add8e6")

    def sort_priority(self):
        self.reset_button_colors()
        self.filtered_tasks = sort_by_priority(self.filtered_tasks)
        self.populate_tree(self.filtered_tasks)
        self.sort_buttons["priority"].config(bg="#ffa07a")

    def sort_due_date(self):
        self.reset_button_colors()
        self.filtered_tasks = sort_by_due_date(self.filtered_tasks)
        self.populate_tree(self.filtered_tasks)
        self.sort_buttons["due"].config(bg="#90ee90")

# Main loop
if __name__ == "__main__":
    root = tk.Tk()
    app = TaskManagerApp(root)
    root.mainloop()
```

The saved tasks in tasks. json



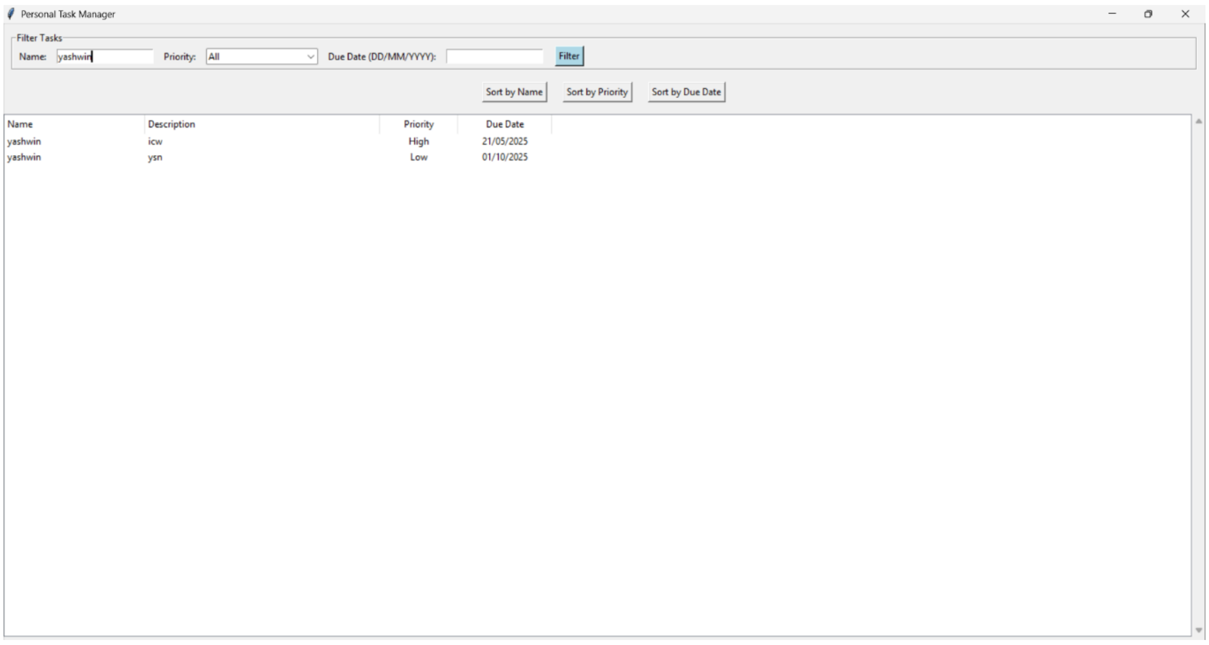
```
[
    {
        "name": "yash",
        "description": "yashwin",
        "priority": "Low",
        "due_date": "21/05/2025"
    },
    {
        "name": "yashwiin",
        "description": "yash",
        "priority": "Medium",
        "due_date": "23/05/2026"
    },
    {
        "name": "yashwin",
        "description": "icw",
        "priority": "High",
        "due_date": "21/05/2025"
    },
    {
        "name": "yash",
        "description": "icw",
        "priority": "Low",
        "due_date": "23/06/2025"
    },
    {
        "name": "yashwin",
        "description": "ysn",
        "priority": "Low",
        "due_date": "01/10/2025"
    }
]
```

GUI Records



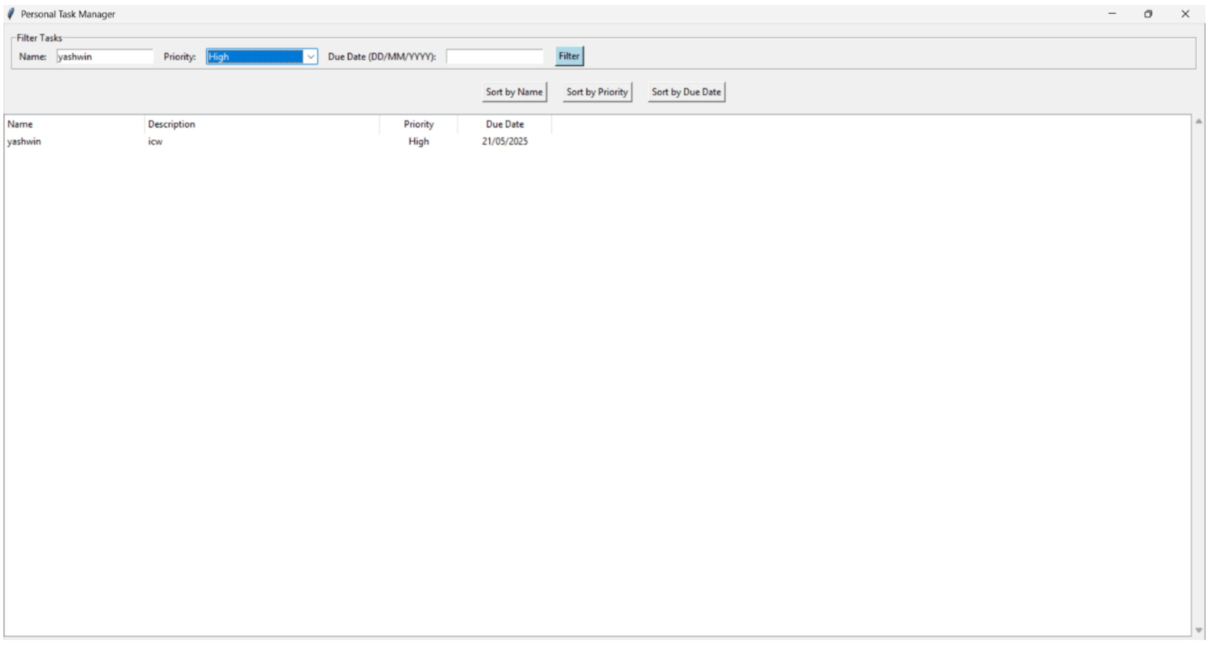
Priority: All

Name	Description	Priority	Due Date
yash	yashwin	Low	21/05/2025
yashwiin	yash	Medium	23/05/2026
yashwin	icw	High	21/05/2025
yash	icw	Low	23/06/2025
yashwin	ysn	Low	01/10/2025



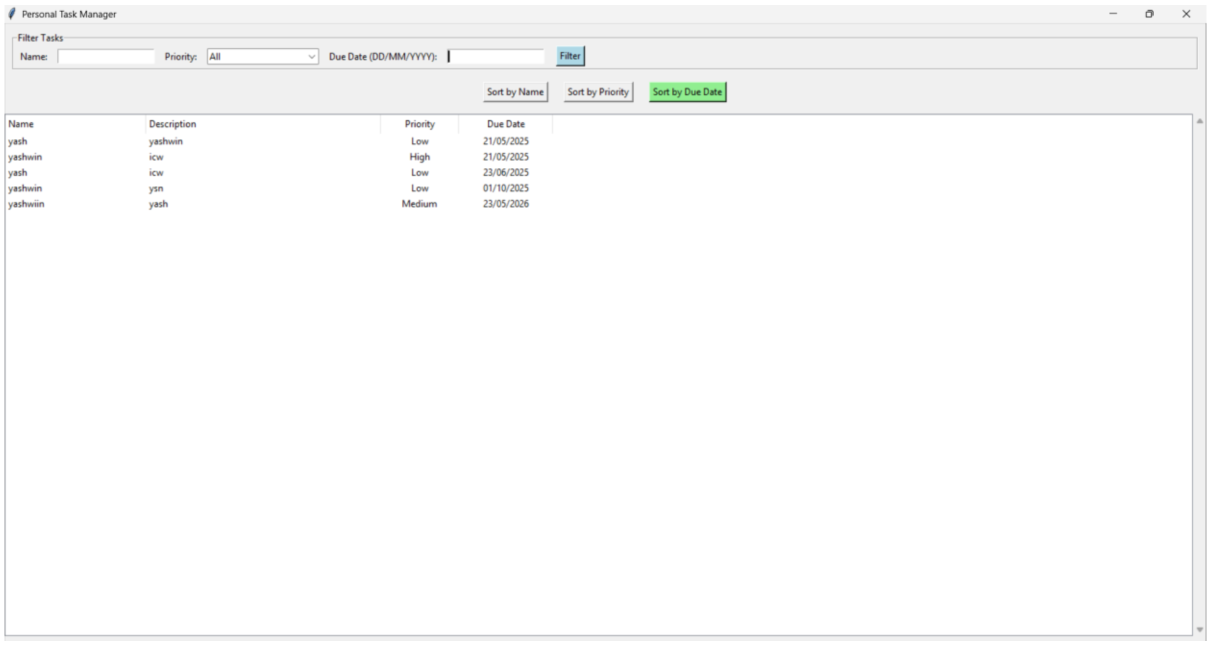
Name: yashwin Priority: All Due Date (DD/MM/YYYY):

Name	Description	Priority	Due Date
yashwin	icw	High	21/05/2025
yashwin	ysn	Low	01/10/2025

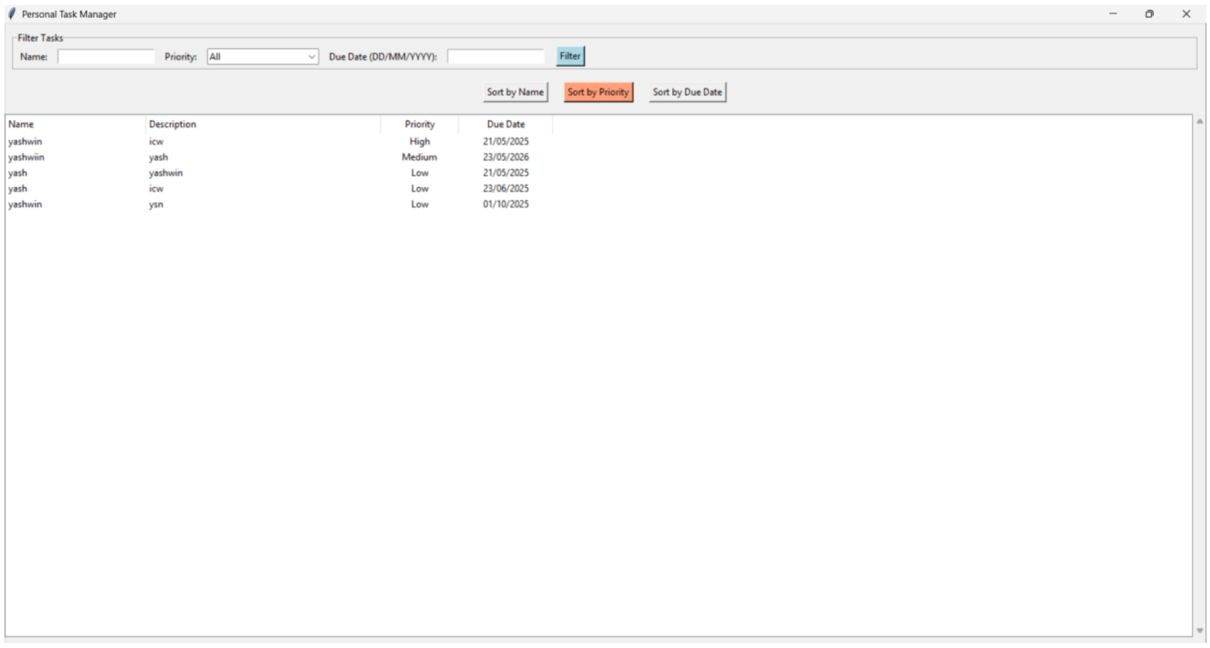


Name: yashwin Priority: High Due Date (DD/MM/YYYY):

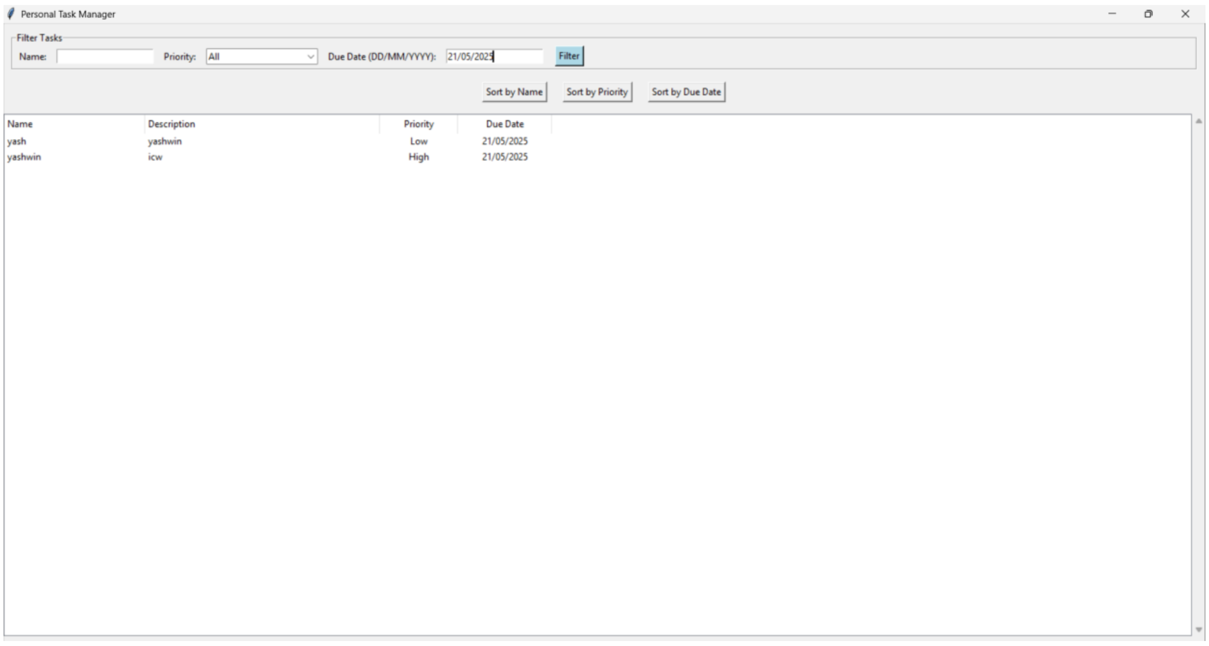
Name	Description	Priority	Due Date
yashwin	icw	High	21/05/2025



Name	Description	Priority	Due Date
yash	yashwin	Low	21/05/2025
yashwin	icw	High	21/05/2025
yash	icw	Low	23/06/2025
yashwin	ysn	Low	01/10/2025
yashwiin	yash	Medium	23/05/2026



Name	Description	Priority	Due Date
yashwin	icw	High	21/05/2025
yashwiin	yash	Medium	23/05/2026
yash	yashwin	Low	21/05/2025
yash	icw	Low	23/06/2025
yashwin	ysn	Low	01/10/2025



Personal Task Manager

Filter Tasks

Name:

Priority: All

Due Date (DD/MM/YYYY): 21/05/2025

Name	Description	Priority	Due Date
yash	yashwin	Low	21/05/2025
yashwin	icw	High	21/05/2025