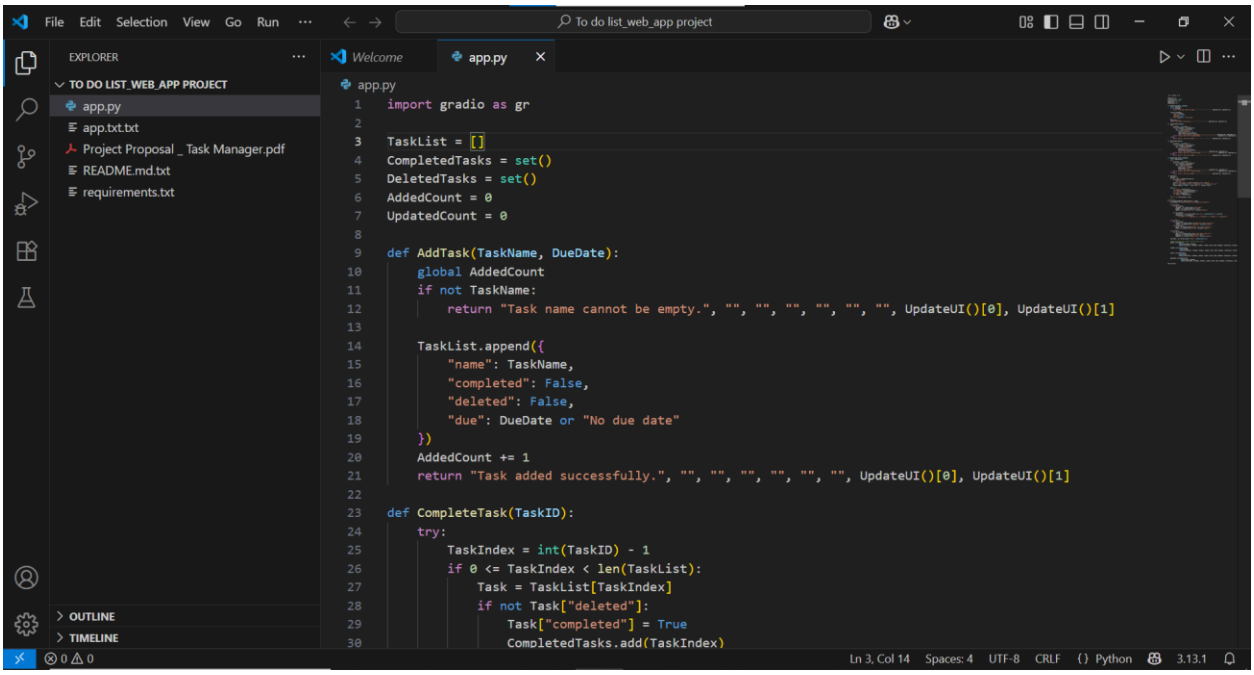
📄 To-Do List Manager – Project Documentation

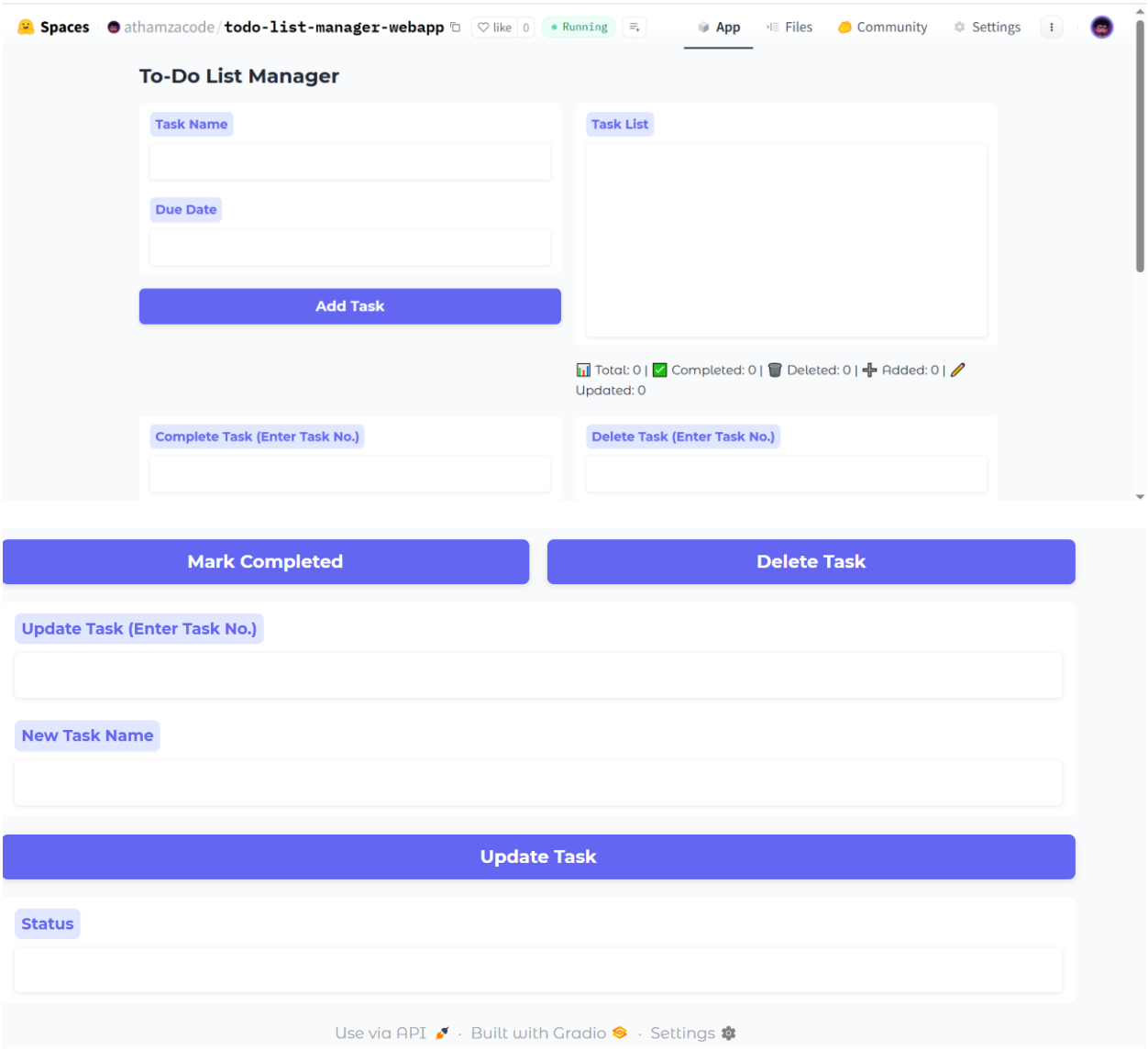
Author: Hamza Badshah

Project: To-Do List Manager Web App (Python + Gradio)

Introduction

This project is a web-based **To-Do List Manager** application built with **Python and Gradio**. It allows users to add, update, complete, and delete tasks using a simple graphical interface. The application also tracks task statistics such as total tasks, completed, deleted, added, and updated tasks.





- Web interface of the To-Do List Manager
- Task added successfully
- Task marked completed
- Task deleted or updated

Task-Wise Explanation

Task 1: Add Task

Function: AddTask(TaskName, DueDate)

```
def AddTask(TaskName, DueDate):
    global AddedCount
    if not TaskName:
        return "Task name cannot be empty.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]


    TaskList.append({
        "name": TaskName,
        "completed": False,
        "deleted": False,
        "due": DueDate or "No due date"
    })
    AddedCount += 1
    return "Task added successfully.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
```

Explanation:

- Takes a task name and due date as input.
- Adds the task to the TaskList with completed and deleted flags set to False.
- Increments the added counter.
- Returns an updated display.

Task 2: Complete Task

Function: CompleteTask(TaskID)

```
def CompleteTask(TaskID):
    try:
        TaskIndex = int(TaskID) - 1
        if 0 <= TaskIndex < len(TaskList):
            Task = TaskList[TaskIndex]
            if not Task["deleted"]:
                Task["completed"] = True
                CompletedTasks.add(TaskIndex)
                return "Task marked as completed.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
        return "Invalid Task ID or already deleted.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
    except:
        return "Enter a valid task number.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
```

Explanation:

- Accepts task number.
- Converts it to an index and marks the task as completed (if not deleted).
- Adds it to the CompletedTasks set.

Task 3: Delete Task

Function: DeleteTask(TaskID)

```
def DeleteTask(TaskID):
    try:
        TaskIndex = int(TaskID) - 1
        if 0 <= TaskIndex < len(TaskList):
            Task = TaskList[TaskIndex]
            if not Task["deleted"]:
                Task["deleted"] = True
                DeletedTasks.add(TaskIndex)
                return "Task deleted.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
        return "Invalid Task ID or already deleted.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
    except:
        return "Enter a valid task number.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
```

Explanation:

- Accepts a task ID.
- Marks it as deleted and tracks it in DeletedTasks.

Task 4: Update Task

Function: UpdateTask(TaskID, NewName)

```
def UpdateTask(TaskID, NewName):
    global UpdatedCount
    try:
        TaskIndex = int(TaskID) - 1
        if 0 <= TaskIndex < len(TaskList):
            Task = TaskList[TaskIndex]
            if not Task["deleted"]:
                Task["name"] = NewName
                UpdatedCount += 1
                return "Task updated.", "", "", "", "", "", "", UpdateUI()[0], UpdateUI()[1]
        return "Invalid Task ID or already deleted.", "", "", "", "", "", "", UpdateUI()[0],
UpdateUI()[1]
    except:
        return "Enter a valid task number.", "", "", "", "", "", "", UpdateUI()[0],
UpdateUI()[1]
```

Explanation:

- Updates the task name by task number.
- Ensures task is not deleted.
- Tracks number of updates.

<u>Task 5: UI and Stats Display</u>

Function: UpdateUI()

```
def UpdateUI():
    Display = []
    for idx, task in enumerate(TaskList):
        if task["deleted"]:
            continue
        status = "Completed" if task["completed"] else "Pending"
        due = f"(Due: {task['due']})" if task['due'] != "No due date" else ""
        Display.append(f"{idx+1}. {task['name']} — {status} {due}")
    Stats = (
        f"📊 Total: {len(TaskList)} | "
        f"✅ Completed: {len(CompletedTasks)} | "
        f"🗑️ Deleted: {len(DeletedTasks)} | "
        f"➕ Added: {AddedCount} | "
        f"✏️ Updated: {UpdatedCount}"
    )
    return "\n".join(Display), Stats
```

Explanation:

- Prepares formatted task list and status.
- Tracks task status in real-time (added, deleted, completed, etc.).

<u>Task 6: Gradio Interface</u>

Code: Gradio UI layout

```
with gr.Blocks(theme=gr.themes.Soft()) as demo:
    ...
demo.launch()
```

Explanation:

- Layout includes task name, due date, input fields, and action buttons.
- Buttons are linked to functions.
- Automatically refreshes the UI after each action.
- Simple and intuitive design for managing tasks.

Challenges Faced & Solutions

Challenge	Solution
Clearing the input fields after actions	Used multiple output values to reset the fields after task operations.
Avoiding UI crash on wrong input	Used try-except blocks for all functions that parse Task IDs.
Updating UI in real-time	Created UpdateUI() function and bound it to all button actions.
Handling deleted tasks correctly	Used flags (deleted, completed) to manage task states without removing.

Conclusion

The To-Do List Manager is a responsive and user-friendly app for tracking tasks with operations. It uses Python's logic and Gradio's simple UI blocks to deliver a clean productivity tool. The modular design and feedback system make it scalable for future features like priority levels, categories, or login-based task storage.