Windows Programming - Midterm Project Report

To Do List app with TkInter

Student Name : Radian Try Darmawan

Student ID : 113021220

Instructor : DINH-TRUNG VU

2025-10

Chapter 1 Introduction

1.1 Background and Motivation

In modern daily life, task management has become increasingly important for productivity and organization. Many people struggle to keep track of their daily tasks, priorities, and deadlines, leading to missed commitments and reduced efficiency. Traditional paper-based to-do lists are easy to lose and difficult to organize, while many digital solutions are overly complex for beginner users.

The application focuses on essential task management features: adding tasks with priority levels, marking tasks as complete or incomplete, deleting tasks, and persistent data storage. The goal is to create a beginner-friendly tool that demonstrates fundamental GUI programming concepts while providing practical utility.

1.2 Objectives

The main objectives of this project are:

- Design and implement a task management application with an intuitive graphical user interface
- Implement core CRUD (Create, Read, Update, Delete) operations for task management
- Provide visual feedback through color-coding based on task priority and completion status
- Enable persistent data storage using JSON file format
- Demonstrate understanding of event-driven programming and GUI design principles

1.3 Tools and Environment

Table 1.1. Tools and Environment

Component	Description
Programming Language	Python
Data Storage Format	JSON
Additional Library	Datetime, os, json
GUI Framework	TkInter
OS Tested	Windows 11

Chapter 2 System Design and Architecture

2.1 Overview

The To-Do List Manager is designed as a **single-threaded**, event-driven GUI application** built with Tkinter. The architecture follows a simple Model-View pattern where:

- **Data Layer (Model)**: JSON file (`tasks.json`) stores task data persistently
- **Presentation Layer (View)**: Tkinter widgets display the user interface
- **Logic Layer:** Python functions handle business logic and data operations

The application maintains synchronization between the JSON file and the displayed task list, ensuring data consistency across sessions.

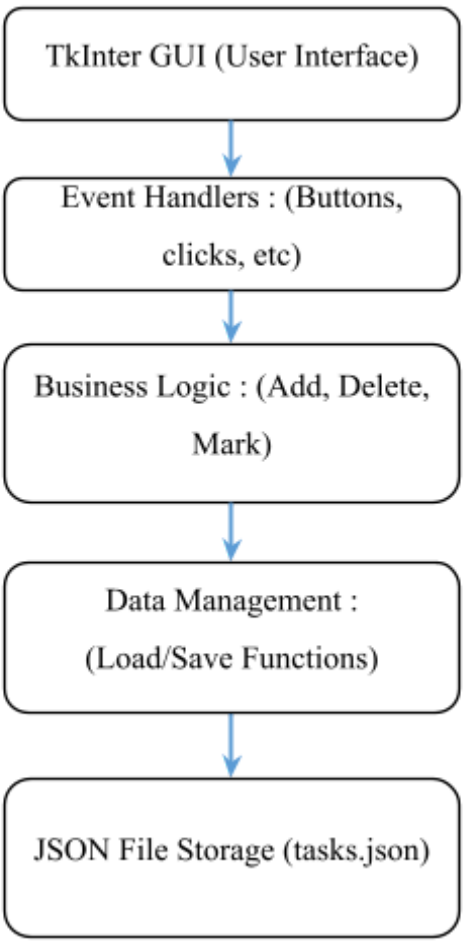

Figure 2.1. Architecture overview

2.2 Architecture Description

2.2.1 User Interface

- **Input fields**: Task Name, Priority, Due Date
- **Priority Selection**: High, Medium, Low (via Combobox)
- **Listbox**: Tasks with color-coded priorities and completion icons
- **Buttons**: Add Task, Mark Complete/Incomplete, Delete Selected, View Details, Clear All
- **Action Buttons**: Save, Load (for file operations)
- **Status Bar**: Showing helpful tips and keyboard shortcuts

2.2.2 Form Actions

- Add Task:

Inserts a new task into the list with user-provided details (name, priority, due date).

- Delete Task:

Removes the selected task after confirmation.

- Toggle Complete:

Marks selected task as complete or incomplete.

- View Details:

Displays detailed task information in a popup dialog.

- Clear All:

Deletes all tasks after user confirmation.

- Save / Load:

Writes or reads all tasks from a JSON file.

2.2.3 Events

1. Double-click: View task details.
2. Keyboard shortcuts:

- `ESC` Exit application
- -`F1` / `Enter` Add task
- `Delete` Delete selected task

2.3 Widget Plan

2.3.1 Widget Using Purpose

Widget	Purpose
Label	Field titles for task input and descriptions
Entry	Input for task name and due date
Combobox	Select task priority (High, Medium, Low)
Listbox + Scrollbar	Display tasks with icons and color-coded status
Button	Perform actions (Add, Delete, Mark, Clear, Save, Load)
MessageBox	Show error, warning, or success messages
LabelFrame	Group related widgets together

2.3.2 Variable Type and Description

Table 2.2. Variable Type and Descriptions

Field Name	Variable Type Description	Widget
Task Name	StringVar (not empty)	Entry
Priority	StringVar ("High", "Medium", "Low")	Combobox
Due Date	StringVar (YYYY-MM-DD format)	Entry
Completed	Boolean flag (True/False)	Listbox
Tasks	List of dictionaries (task objects)	Listbox
MessageBox	Display success or error messages	messagebox

2.4 Geometry Management

- Layout manager used: `grid()` for input form alignment and `pack()` for overall frame layout.
- Reason: The design uses a top input frame and a bottom scrollable list frame, ensuring adaptive resizing while keeping a clean and readable interface.

Chapter 3 Results and Analysis

3.1 Implemented Features

✔ Add, delete, and mark tasks as complete.

✔ Save and load from JSON file.

✔ Keyboard shortcuts for efficiency.

✔ Auto-updated task display with color-coded priorities.

✔ Completion indicators with checkmarks and color changes.

✔ Input validation for task name and priority.

3.2 GUI Screenshots and Explanations

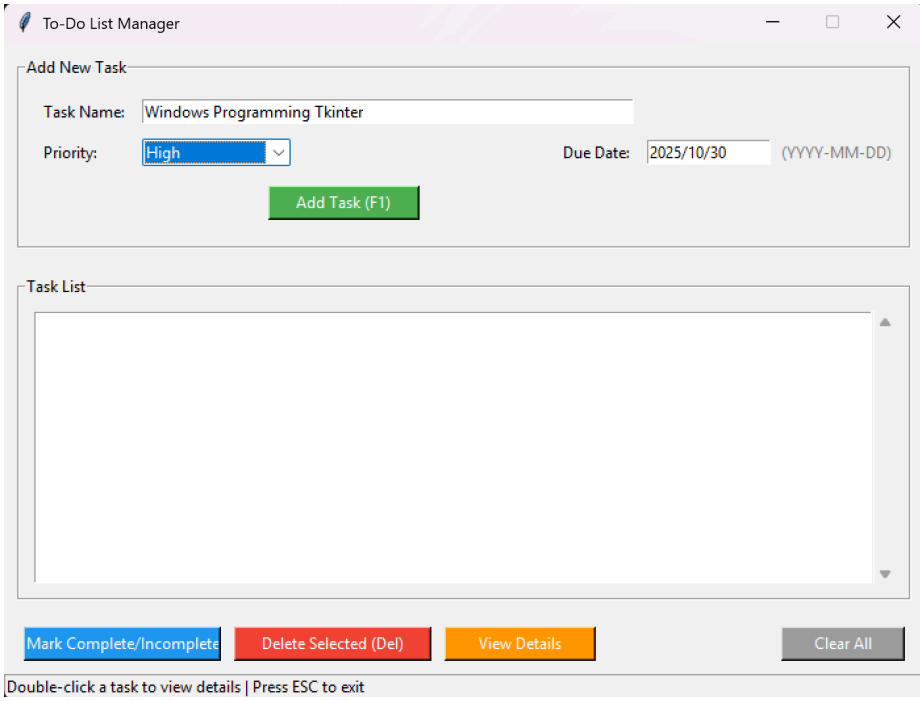

Figure 3.2.1 shows a working interface of the system on default

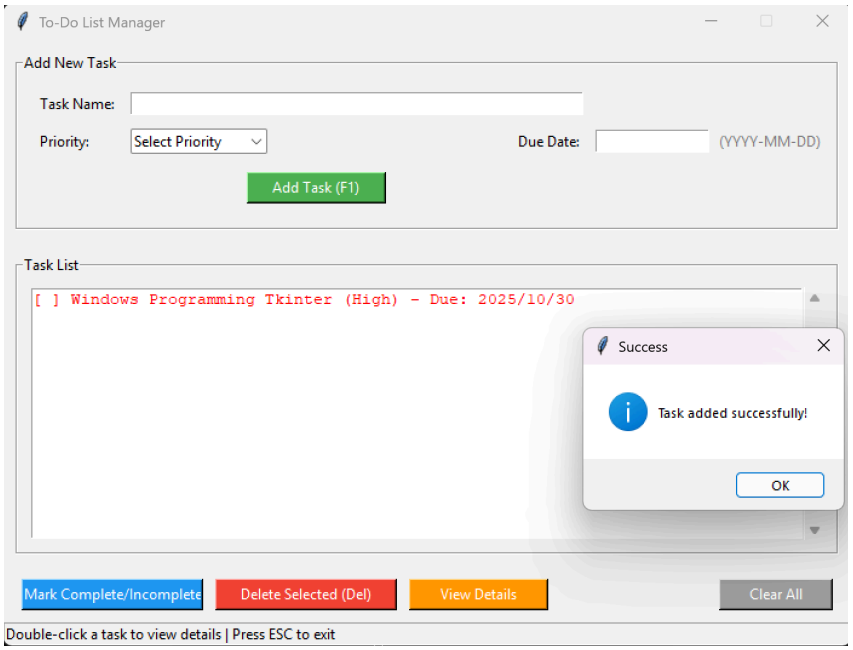

Figure 3.2.2 shows an input field and task added successfully message

After we input the Task Name, Priority, and Due Date, we can click Add Task or use F1 for shortcut. After everything has been filled this window will show up acknowledging our submission

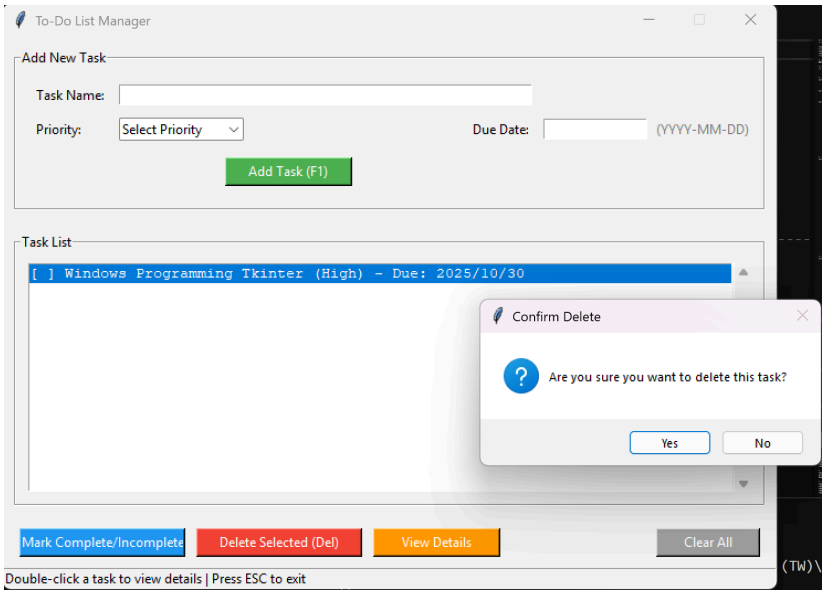

Figure 3.2.3 shows a confirm delete button to make sure the user agrees to delete

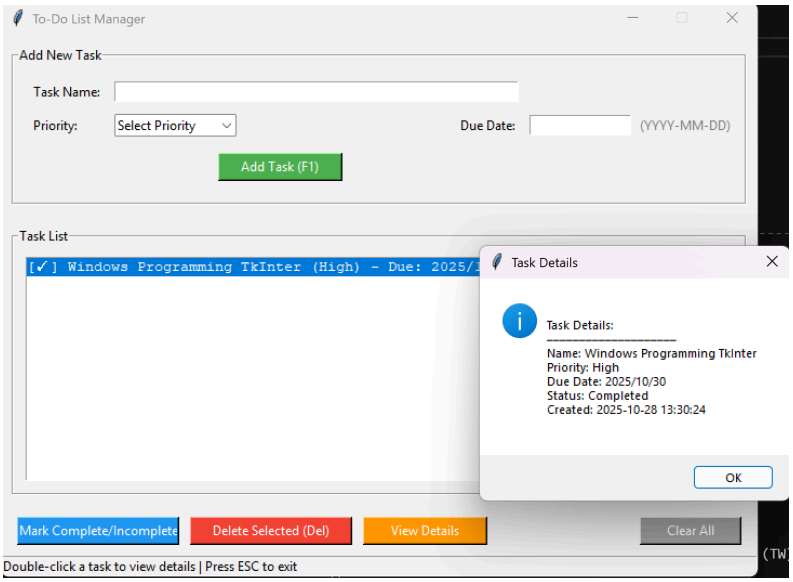

Figure 3.2.4 shows the detail of the task we put in the Task List

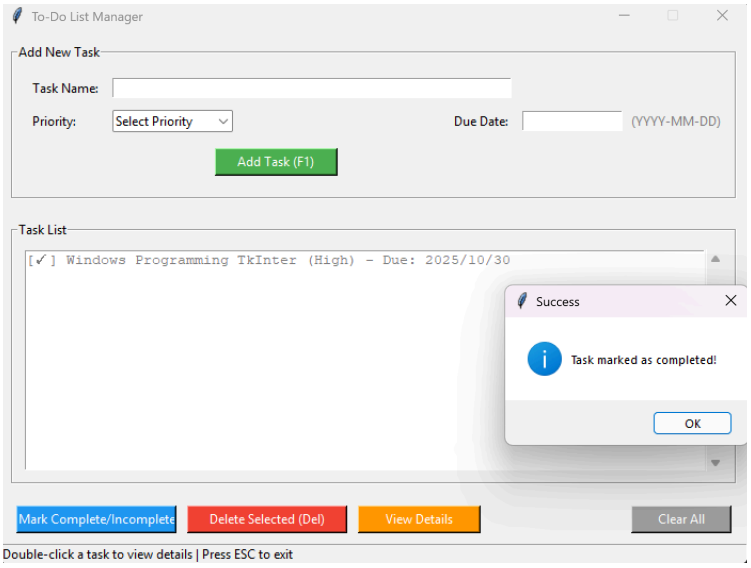

Figure 3.2.4 shows Task marked as completed message, we use this if we are sure that our task has been completed

3.2.1 Main Interface

The main window displays task input fields, a priority selector, and a list area. Tasks appear with color coding:

- Red text for High priority
- Orange text for Medium priority
- Green text for Low priority
- Gray text for completed tasks

3.2.2 Successful Save/Load

When tasks are saved successfully, a messagebox confirms the operation. The saved data is stored in `tasks.json` in JSON format, containing task name, priority, due date, and completion status.

Sample `tasks.json` structure :

```
```json
[
  {
    "name": "Complete midterm project",
    "priority": "High",
    "due_date": "2025-10-28",
    "completed": false,
    "created_at": "2025-10-20 14:30:00"
  },
  {
    "name": "Buy groceries",
    "priority": "Medium",
    "due_date": "2025-10-25",
    "completed": true,
    "created_at": "2025-10-21 09:15:00"
  }
]
```
```

Chapter 4 Challenges and Solution

Challenge	Description	Solution
Data Persistence	Handling corrupted or empty JSON files	Added exception handling for safe load/save operations
UI Responsiveness	Ensuring widgets resize properly	Combined `pack()` and `grid()` with proper configuration
Input Validation	Preventing empty or invalid task entries	Implemented validation checks before adding tasks

Color Management	Applying colors to individual listbox items	Used `itemconfig()` method with index-based styling

Table 4.1. Challenges and Solutions

Chapter 5 Conclusion and Future Works

5.1 Conclusion

The To-Do List Manager project successfully demonstrates event-driven programming with Tkinter. It integrates GUI components, file handling, and logical task operations cohesively. The program fulfills all midterm requirements—add, delete, mark complete, and save/load from file—while featuring an intuitive interface with color coding and task prioritization.

This project provided valuable hands-on experience in GUI development, data persistence, and user interface design. The application is stable, reliable, and suitable for everyday task management use.

5.2 Future Work

Planned improvements include:

- Sorting and filtering tasks by date or priority
- Search bar for quick task lookup
- Task editing functionality without deleting and re-adding
- Date picker widget for easier due date selection
- Task categories to organize tasks by type (Work, Personal, etc.)
- Notifications or reminders for upcoming due date

Chapter 6 References and Appendix

References:

- Python Tkinter official documentation:
  https://docs.python.org/3/library/tkinter.html
- Python JSON Handling:
  https://docs.python.org/3/library/json.html
- Real Python - Tkinter Tutorial:
  https://realpython.com/python-gui-tkinter/

Appendix:

- Source Code: `todo_list_manager.py`
- Dataset: `tasks.json`
- GitHub Repository: https://github.com/radiandrmwn/ToDoListWP