Manager of Tasks

A Group Two Presentation

Planning (High req analysis)

- Functional requirements
- Non-functional requirements

3. Requirements

This section specifies the software product's requirements. Specify all of the software requirements to a level of detail sufficient to enable designers to design a software system to satisfy those requirements and to enable testers to test that the software system satisfies those requirements.

The specific requirements should:

Be uniquely identifiable.

State the subject of the requirement (e.g., system, software, etc.) and what shall be done.

Optionally state the conditions and constraints, if any.

Describe every input (stimulus) into the software system, every output (response) from the software system, and all functions performed by the software system in response to an input or in support of an output.

Be verifiable (e.g., the requirement realization can be proven to the customer's satisfaction).

Conform to agreed-upon syntax, keywords, and terms.

CLI

cli.py | task_manager.py

src > cli > task_manager.py > ...

```python
from taskstorage import TaskStorage
import json

class TaskManager:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super(TaskManager, cls).__new__(cls)
            cls._instance.tasks = {}  # {task_id: Task}
        return cls._instance

    def add_task(self, task: Task):
        if task.task_id in self.tasks:
            raise ValueError(f"Task with ID {task.task_id} already exists.")
        self.tasks[task.task_id] = task

    def remove_task(self, task_id: str):
        if task_id not in self.tasks:
            raise ValueError(f"Task with ID {task_id} does not exist.")
        del self.tasks[task_id]

    def save_tasks(self):
        TaskStorage().save_tasks(self.tasks)

    def load_tasks(self):
        self.tasks = TaskStorage().load_tasks(self)

    def edit_task(self, task_id: str):
        if task_id not in self.tasks:
            raise ValueError(f"Task with ID {task_id} does not exist.")

        task = self.tasks[task_id]

```

cli.py | task.py

src > cli > task.py > Task

```python
import uuid # module to help create unique IDs

# Task class to be instantiated to represent individual tasks
class Task:
    def __init__(
        self, # refrence to the instance being created
        title: str,
        description: str = "", # "" makes the description optional, sets default to empty string
        due_date: str = None, # optional dude date
        priority: str = "Medium", # default priority set to Medium, but can be changed by user
        tags: list[str] = None, # Optional list of tag
        visible: bool = True, # Visibility status for filtering
    ):
        # initializing instance attributes
        self.title = title
        self.description = description
        self.due_date = due_date
        self.priority = priority
        self.task_id = str(uuid.uuid4())  # Will always generate a custom ID
        self.tags = tags if tags else []
        self.visible = visible

    # just for console output debugging
    def __repr__(self):
        # Returns a string like "Task(title='...', ...) so we can see it in a readable format"
        return (
            f"Task(title='{self.title}', "
            f"description='{self.description}', "
            f"due_date='{self.due_date}', "
            f"priority='{self.priority}', "
            f"task_id='{self.task_id}')"
        )

    # Equality check for tasks
    def __eq__(self, other):
        # Checks equality based on task_id
        if not isinstance(other, Task):
            return False
        return self.task_id == other.task_id
```

CLI tests (TaskManager and task)

Command line tests to test CLI

Unittest for TaskManager and Task class

cli.py

src > cli > cli.py > main

```python
# CLI Prototype for Milestone 2

import json
from task import Task
from task_manager import TaskManager

def main():
    task_man = TaskManager()
    running = True
    print("Welcome to the task manager! Please select an option: ")
    while running:
        print("Option 1: Add Task")
        print("Option 2: Remove Task")
        print("Option 3: View Tasks")
        print("Option 4: Save Tasks")
        print("Option 5: Load Tasks")
        print("Default Option: Quit Task Manager")
        option = input("Select a number, or press anything else to quit: ")
        match option:
            case "1":
                task_name = input("What's the task name? ")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mitch\OneDrive\Documents\GitHub\CS2450---G2-Project> & C:/Users/mitch/AppData/Local/Microsoft/Windows
Apps/python3.11.exe c:/Users/mitch/OneDrive/Documents/GitHub/CS2450---G2-Project/src/cli/cli.py
Welcome to the task manager! Please select an option:
Option 1: Add Task
Option 2: Remove Task
Option 3: View Tasks
Option 4: Save Tasks
Option 5: Load Tasks
Default Option: Quit Task Manager
Select a number, or press anything else to quit:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Welcome to the task manager! Please select an option:
Option 1: Add Task
Option 2: Remove Task
Option 3: View Tasks
Option 4: Save Tasks
Option 5: Load Tasks
Default Option: Quit Task Manager
Select a number, or press anything else to quit: 1
What's the task name? Test task
Enter a description, or press 'enter' without a description to leave blank: Testing add task functionality
Enter a due date, or press 'enter' without a due date to leave blank: 04/21/25
If you wish to edit the priority, enter 'high' or 'low', or leave blank for 'medium': high
```

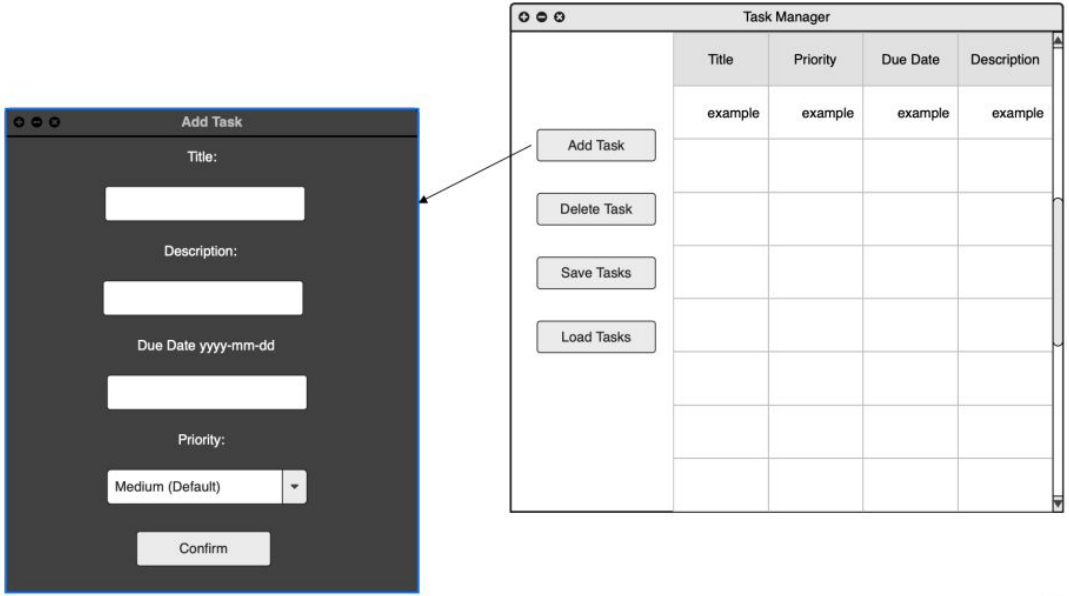
GUI Wireframe

Task Manager Window

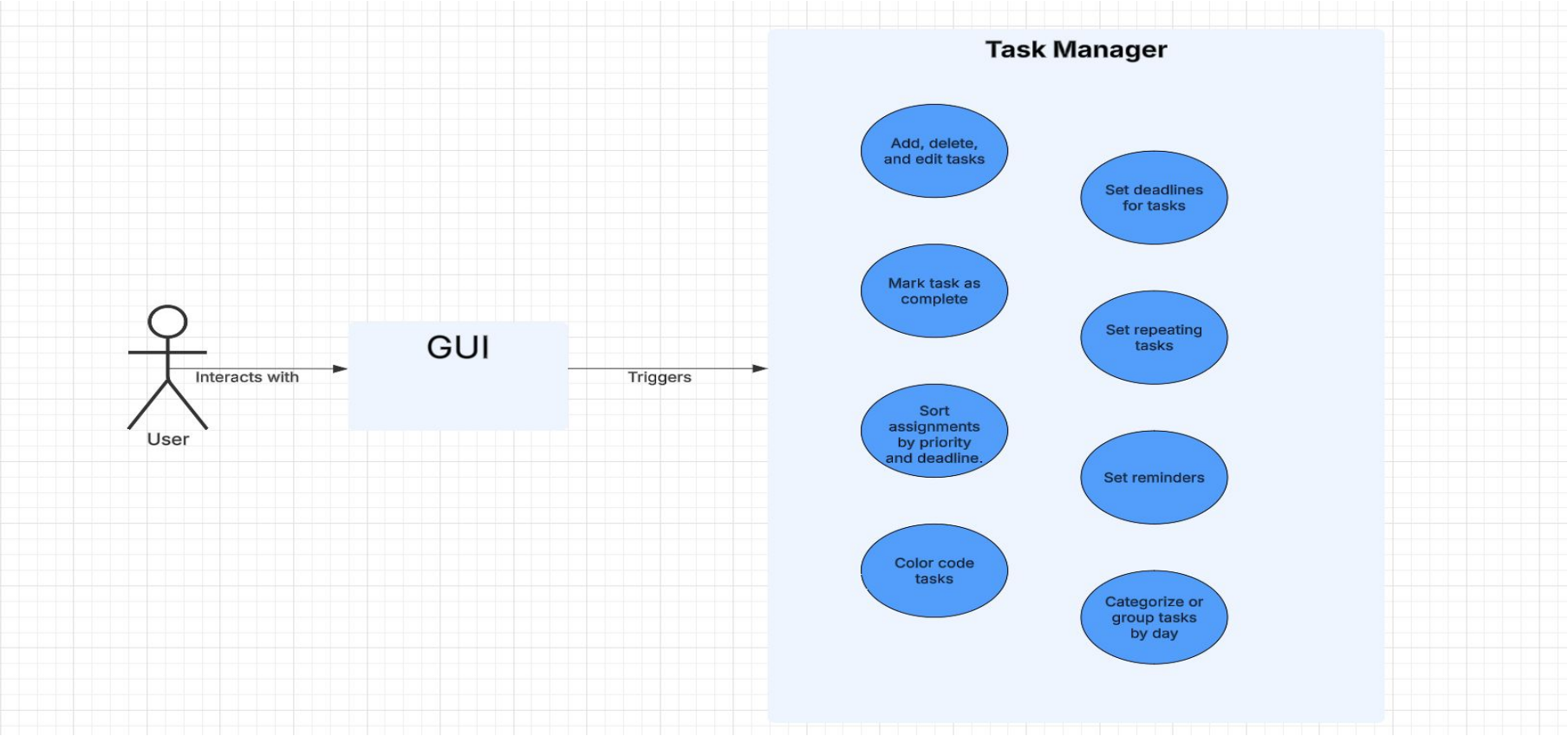
- Simple, user-friendly layout
- Sidebar with buttons (add, delete, etc.)
- Tabular format

Add Task Window

- Simple input fields: Title, Description, Due Date
- Priority dropdown (low, medium, high)
- Confirm button to add task



Use Case Diagram

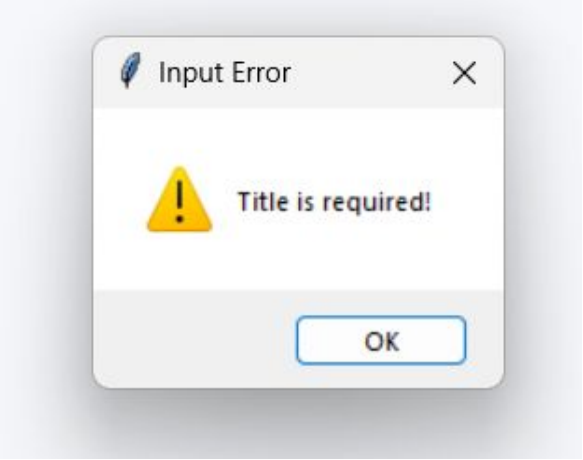


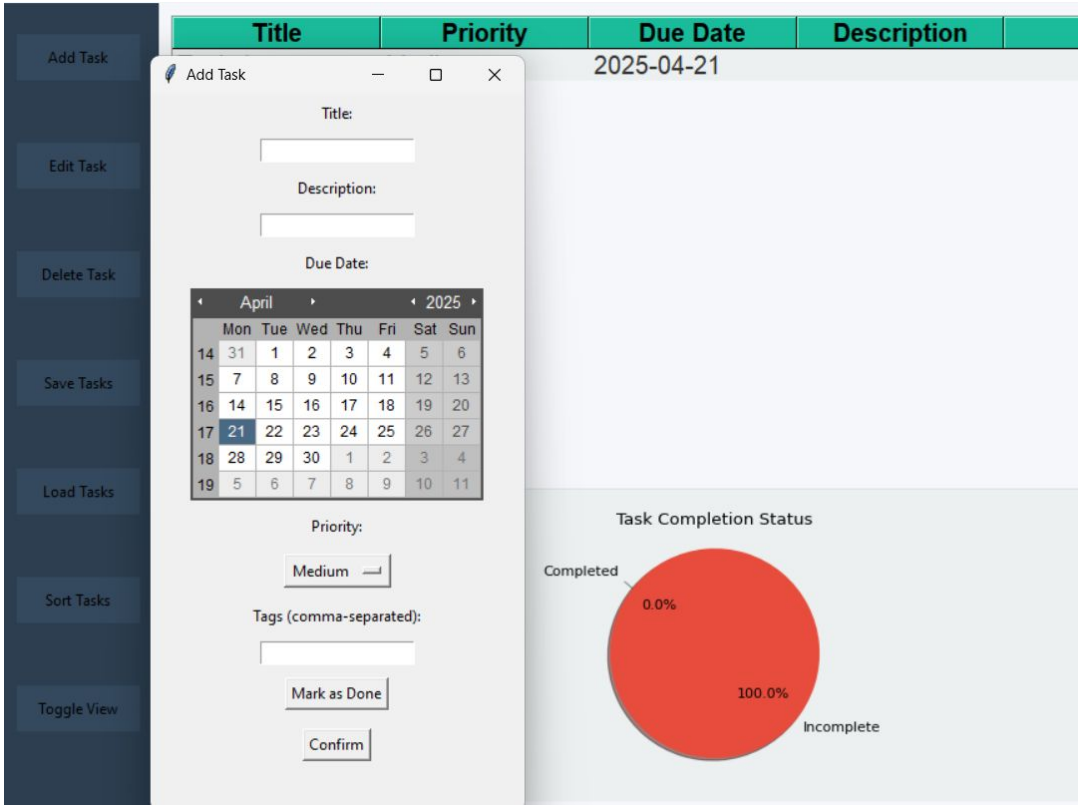
Add Task

- Input fields for Title, Description
- Tkcalendar for Due Date (ensures only valid dates)
- Dropdown for priority

Input Error

- Must enter task title



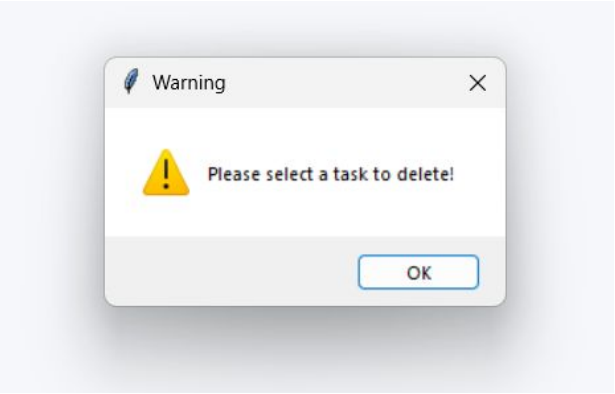


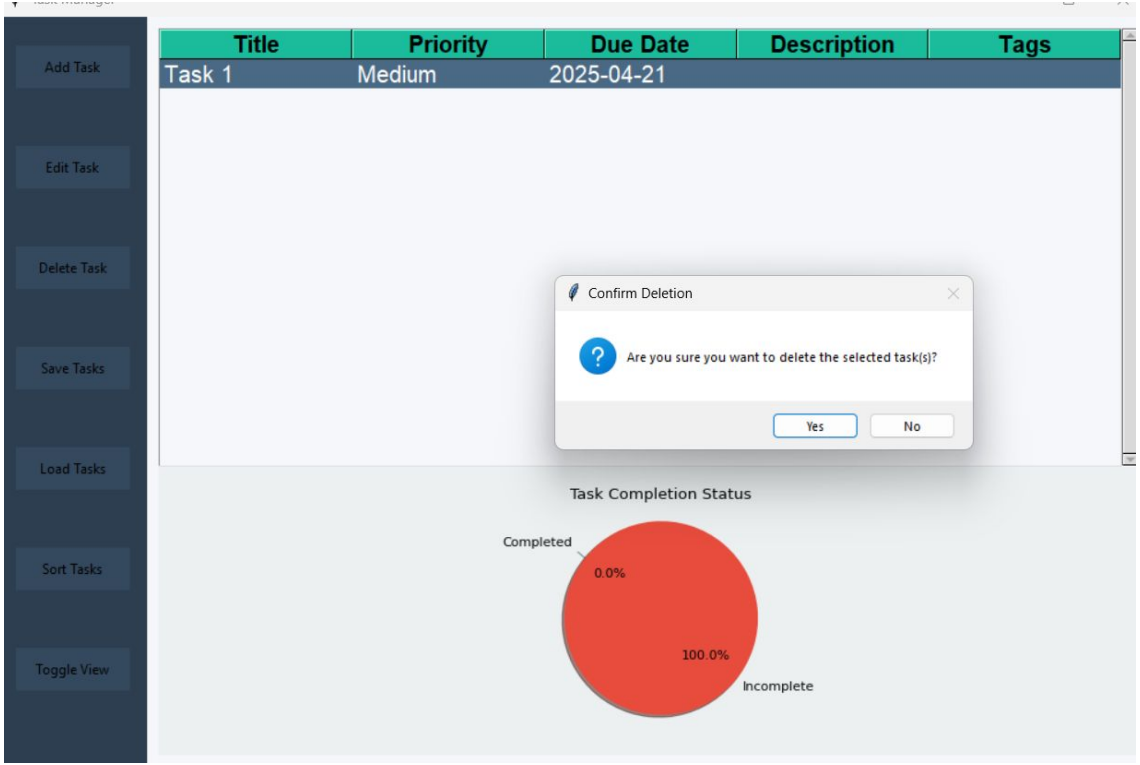
Delete Task

- Tasks in treeview are selectable
- Confirm deletion window

Warning

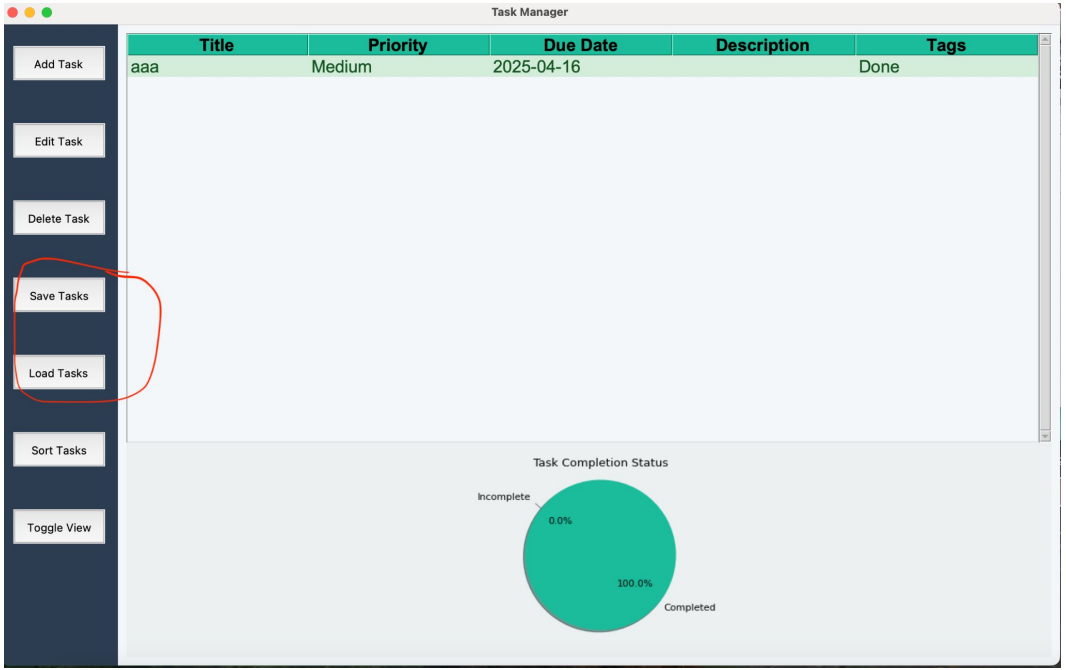
- No task selected





Save/Load Functionalities

- Class TaskStorage():
- def save_task():
- def load_task():
- Writes to JSON
- Reads from JSON
- Saves everything about tasks
- Dictionary for tasks
- task_id as key
- Lists as task attributes



Normal Features Testing

test_add_task()

- Test adding a task to the manager

test_add_duplicate_task()

- Adding a task with the same ID should raise an error

test_remove_task()

- Test removing a task from the manager

test_remove_nonexistent_task()

- Removing a non-existent task should raise an error

test_task_dictionary_integrity()

- Ensure the tasks dictionary updates correctly

test_save_load()

- Test that the tasks correctly save and load

Refactoring GUI Code

GUI File Before Refactoring

- Had nearly 400 lines of code
- Had global colors
- Contained unorganized logic
- Contained treeview styling

GUI File After Refactoring

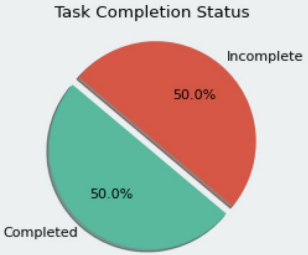
- Around 200 lines of code
- Imports styling logic
- Uses add, delete, and edit task classes
- Added more import statements

```python
import sys
import tkinter as tk

from tkinter import ttk, messagebox, OptionMenu, Button, Label
from tkcalendar import Calendar

from .task_visualizer import TaskVisualizer
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from ..cli.task import Task
from ..cli.task_manager import TaskManager
from .sorter import Sorter, TitleSorter, DateSorter, PrioritySorter
from ..cli.taskstorage import TaskStorage
from ..cli.action_queue import ActionQueue
from .filterer import Filterer, PriorityFilterer, CompleteFilterer, ShowAllFilterer, DefaultFilterer, TagFilterer
from .calendar_view import CalendarView
from .colors import (
    BG_COLOR,
    SIDEBAR_COLOR,
    BUTTON_COLOR,
    BUTTON_HOVER,
    TEXT_COLOR,
    TASK_AREA_BG,
    TASK_TEXT_COLOR,
    HEADER_COLOR
)
from .treeview_style import configure_treeview_styles
from .add_task import AddTaskHandler
from .delete_task import DeleteTaskHandler
from .edit_task import EditTaskHandler
from .toggle_view import ToggleViewHandler
```

Task Visualizer

- Used Matplotlib
- TaskVisualizer class
- Iterates through all tasks
- Tag system to mark as done
- All other tasks are not done
- Red for incomplete
- Green for complete

Title	Priority	Due Date	Description	Tags
task1	Medium	2025-07-30	:(	not done
task2	Medium	2025-04-20		Done



Test Task Visualizer

- Python unittest
- Creates 1 complete task
- Creates 2 incomplete tasks
- Test 1 counts task types
- Test 2 empty list
- Test 3 chart generation
- Test 4 dynamic updating

Task Sorter

-Used a Factory Design

-Sorter, TitleSort, PrioSort, DateSort

-Exists as own module

-Generates new dictionary

-Replaces old one

Testing Task Sorter

-Tested proper sorting on small task dictionaries

-Successfully matched by title, priority, and date, resulting in three unique but matched dictionaries

The Undo/Redo Queue

-Stores tasks and task actions in stacks

-Medium coupling used, primarily integration with add/delete

-Whenever undo/redo called, pops from one stack to the other

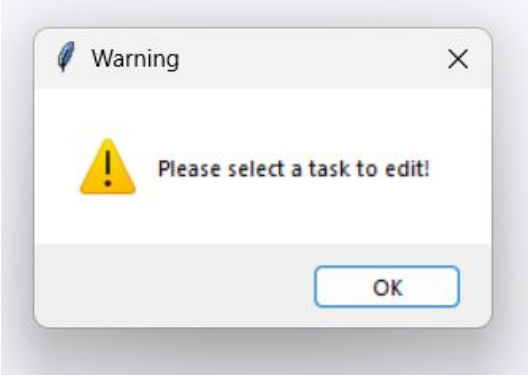
-Binary testing–either it works or it doesn’t

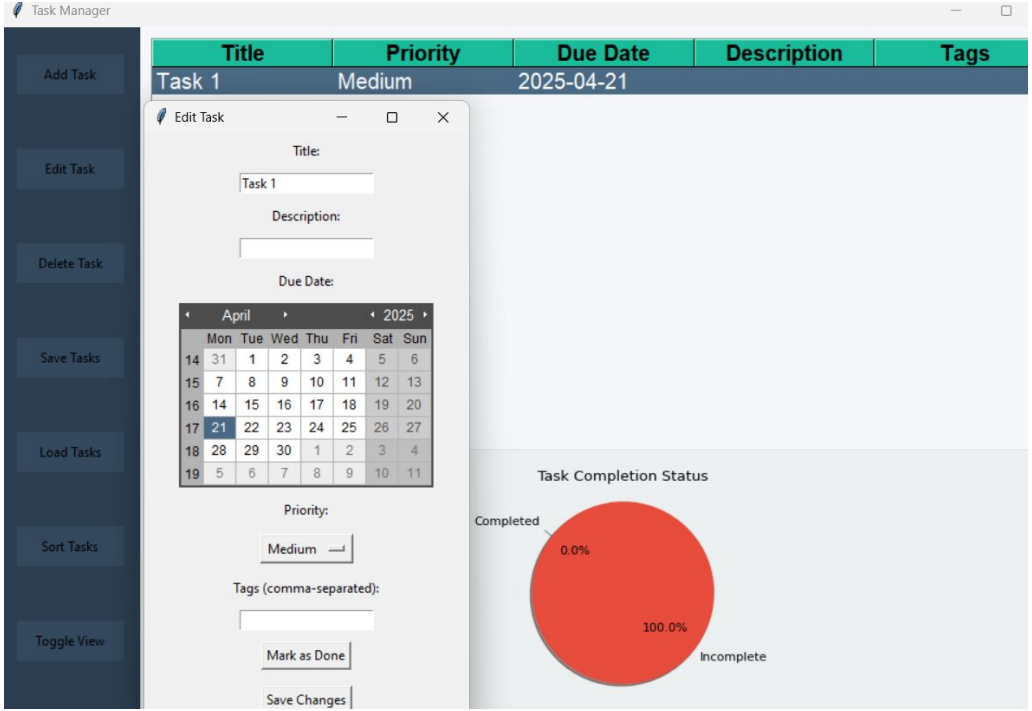
Edit Task

- Select Task to Edit
- Any changes can be made
- “Save Changes” to confirm

Warning

- Task must be selected





Tags

Tags Input Field

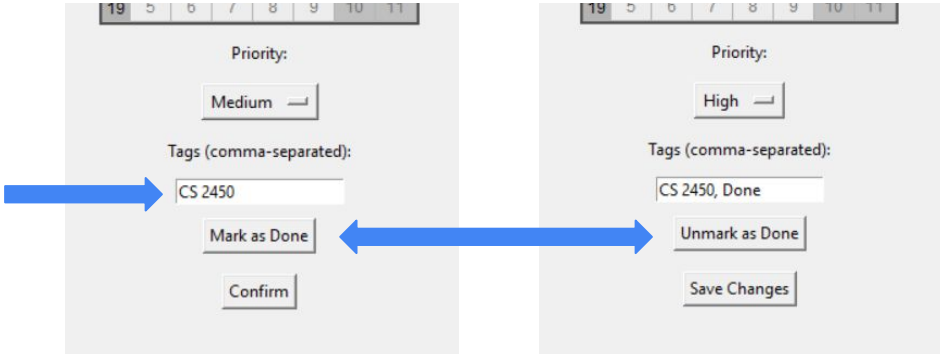
- Users can assign multiple tags to a task (comma-separated str)
- Task Object holds list of tags

Mark as Done/Unmark as Done

- Toggle button to mark a task as complete (appends “Done” to list of tags)
- Removes “Done” from tags if already marked as complete.

Highlights

- Tasks marked as done are highlighted green



Test Tags

test_add_task()

- Test to ensure that tasks with tags can be added to TaskManager

test_view_tasks()

- Asserts that view_tasks displays tags

test_save_and_load_tasks()

- Test if tasks with tags can be saved and loaded from the JSON file

test_tags_are_sorted()

- Test if tags are automatically displayed in alphabetical order

Calendar View

Toggle View

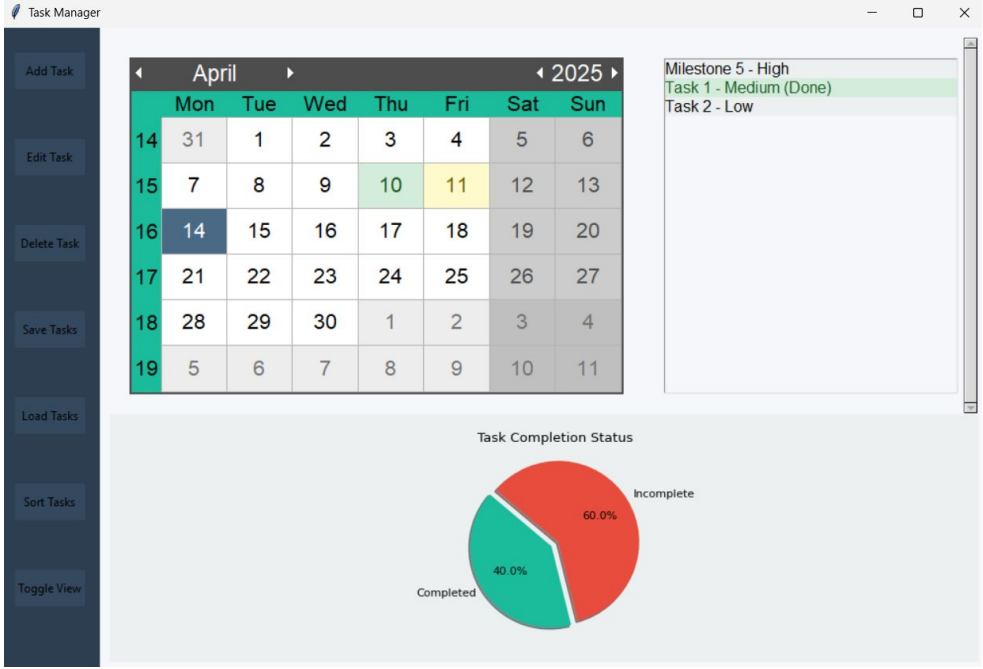
- Replaces tabular format with calendar view
- All tasks carry over for both views

Select Due Date

- When a date is selected, all tasks assigned for that day are displayed on the right
- Displays Title, Priority, and Completion
- Tasks can be added, edited, and deleted from the calendar view (Tasks are selectable)

Highlights

- Yellow shows if there are still incomplete tasks
- Green indicates all tasks are done for a specific due date



Test Toggle and Calendar View

test_toggle_to_calendar_view()

- Test toggling from treeview to calendar view

test_toggle_to_treeview()

- Test toggling from calendar view to treeview

test_get_tasks_for_date()

- Test that tasks are correctly retrieved for a specific date

test_on_date_select()

- Test that tasks are displayed in the Listbox for the selected date

test_get_selected_task_id()

- Test that correct task ID is returned for a selected Listvox item

Team lessons

There is a lot that goes into a software program, It takes a lot of time and communication to build a working software and even more time to make it 100% bugless. - Mitchell

Planning is crucial, but not because it eliminates all of your problems down the line. Planning is important because you need all the time you can get for the problems you can’t plan for. -Noah

Our meetings were the keystone of our project, we were able to make an action plan and get things done. -Carson

File Edit Selection View Go Run Terminal Help

CS2450---G2-Project

EXPLORER

CS2450---G2-PROJECT

- src
  - cli
    - action_queue.py
    - cli.py
    - task_manager.py
    - task.py
    - taskstorage.py
  - gui
    - __pycache__
    - __init__.py
    - add_task.py
    - calendar_view.py
    - colors.py
    - delete_task.py
    - edit_task.py
    - filter.py
    - filterer.py
    - gui.py
    - sorter.py
    - task_visualizer.py
    - toggle_view.py
    - treeview_style.py
  - tests
  - __init__.py
- venv
- .gitignore
- README.md
- requirements.txt
- tasklist.json U

OUTLINE

TIMELINE

action_queue.py | add_task.py | delete_task.py | tasklist.json U | edit_task.py | gui.py | task_manager.py

tasklist.json > ...

1 "zilch"]], "20c73950-9091-4d34-a00c-049f2918615c": ["Delta Task", "asdf", "2025-04-30", "Medium", []], "57ba093c-049f-45ab-a7d6-0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
ab9b-4ce9a6ef3578': Task(title='Gamma Task', description='deltaTag', due_date='2025-04-21', priority='Medium', task_id='456354a4-13f1-48f7-ab
9b-4ce9a6ef3578'), '4c0ef08a-cd7d-452a-984f-b223333728ca': Task(title='Alpha Task', description='Description 1', due_date='2025-04-23', prior
ity='Low', task_id='4c0ef08a-cd7d-452a-984f-b223333728ca')}
Adding action
{'4c0ef08a-cd7d-452a-984f-b223333728ca': Task(title='Alpha Task', description='Description 1', due_date='2025-04-23', priority='Low', task_id
='4c0ef08a-cd7d-452a-984f-b223333728ca'), 'efbe98ba-22e5-4a18-ac7a-ba281f52e032': Task(title='Beta Task', description='This is also a desc',
due_date='2025-04-25', priority='High', task_id='efbe98ba-22e5-4a18-ac7a-ba281f52e032'), '20c73950-9091-4d34-a00c-049f2918615c': Task(title='
Delta Task', description='asdf', due_date='2025-04-30', priority='Medium', task_id='20c73950-9091-4d34-a00c-049f2918615c'), '57ba093c-049f-45
ab-a7d6-09fa65a754f2': Task(title='Epsilon Task', description='asdffdsa', due_date='2025-04-22', priority='Low', task_id='57ba093c-049f-45ab-
a7d6-09fa65a754f2'), '456354a4-13f1-48f7-ab9b-4ce9a6ef3578': Task(title='Gamma Task', description='deltaTag', due_date='2025-04-21', priority
='Medium', task_id='456354a4-13f1-48f7-ab9b-4ce9a6ef3578')}
Adding action
{'456354a4-13f1-48f7-ab9b-4ce9a6ef3578': Task(title='Gamma Task', description='deltaTag', due_date='2025-04-21', priority='Medium', task_id='
456354a4-13f1-48f7-ab9b-4ce9a6ef3578'), '57ba093c-049f-45ab-a7d6-09fa65a754f2': Task(title='Epsilon Task', description='asdffdsa', due_date='
2025-04-22', priority='Low', task_id='57ba093c-049f-45ab-a7d6-09fa65a754f2'), '4c0ef08a-cd7d-452a-984f-b223333728ca': Task(title='Alpha Task'
, description='Description 1', due_date='2025-04-23', priority='Low', task_id='4c0ef08a-cd7d-452a-984f-b223333728ca'), 'efbe98ba-22e5-4a18-ac
7a-ba281f52e032': Task(title='Beta Task', description='This is also a desc', due_date='2025-04-25', priority='High', task_id='efbe98ba-22e5-4
a18-ac7a-ba281f52e032'), '20c73950-9091-4d34-a00c-049f2918615c': Task(title='Delta Task', description='asdf', due_date='2025-04-30', priority
='Medium', task_id='20c73950-9091-4d34-a00c-049f2918615c')}
Adding action
Adding action
C:\Users\noahj\CS2450 Group Project\CS2450---G2-Project\src\gui\task_visualizer.py:36: RuntimeWarning: More than 20 figures have been opened.
 Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memo
ry. (To control this warning, see the rcParam `figure.max_open_warning`). Consider using `matplotlib.pyplot.close()`.
  fig = plt.figure(figsize=(4, 3), dpi=80, facecolor=TASK_AREA_BG)
Tasks saved successfully!

    while executing
"1757622657024on_close"
    (command for "WM_DELETE_WINDOW" window manager protocol)
PS C:\Users\noahj\CS2450 Group Project\CS2450---G2-Project> python -m src.gui.gui
Tasks loaded successfully!

    while executing
"2525109325120on_close"
    (command for "WM_DELETE_WINDOW" window manager protocol)
PS C:\Users\noahj\CS2450 Group Project\CS2450---G2-Project> python -m src.gui.gui
```

powersh...

Python

main* 0 0 Live Share Ln 1, Col 537 Spaces: 4 UTF-8 CRLF JSON Go Live Quokka Prettier

51°F Clear Search 12:15 AM 4/21/2025

Questions