Project Requirements – TaskFlow Deliverable One

Title: TaskFlow

By: Paolo Lauricella

Z number Z23678293

Course: COP4331 / Object-Oriented Design

Platform: Java Swing GUI (Desktop)

Group: Just Me

Project Description

Purpose of this Project
Students will practice the analysis design, and implementation skills learned in this class. The mainn objective is to produce high quality requirements and design artifacts and then implement them in code. This project emphasizes the design stage, with 50% of the total score. It focuses on key concepts covered throughout Chapters 2–10, including UI design using MVC, UML modeling, and the use of design patterns.

Building an objectoriented design is one of the most important skills learned in this course. These skills differentiate a software engineer who understands design principles from a programmer whose work can be easily outsourced. Mastery of design patterns modular architecture, and documentation will also help strengthen interview and professional readiness.

At the end of the semester, the TaskFlow project will be showcased on my GitHub portfolio webpage as a demonstration of design and implementation proficiency. This project is built using Java and Swing as recommended for this course, ensuring that it focuses on core object oriented principless rather than additional APIs or web frameworks.

I myself Paolo Lauricella am developing this independently. It applies all required architectural principles: ModelViewController, at least five design patterns, programming by contract, JUnit testing, and complete documentation via Javadoc. GitHub is used for version control and submission

Functional Specification

TaskFlow application asssists users in managing and organizing their daily activities. It provides an intuitive interface to create, modify, delete, and monitor tasks. Users can set due dates, priorities, reminders, and recurring schedules, and view their progress through calendar and analytics views. The applicatioon uses Java Swing for the GUI and emphasizes Object Oriented principles through design patterns and modular structure.

1. The user can add a new Task with a title, description, due date, time, and priority.
2. The user can edit or delete existing Tasks.
3. The user can mark Tasks as completed or reopen them.
4. The system displays Tasks in both list and calendar (Day/Week/Month) views.
5. The user can filter or sort Tasks by date, status, priority, or tag.
6. The user can set remineders for upcoming or overdue Tasks.
7. The system supports recurring Tasks (daily, weekly, monthly).
8. The user can undo or redo previous actions (Command pattern).
9. The system provides status messages for successful or failed actions.
10. The system saves data locally in JSON and allows importing/exporting of task data.
11. The system includes analytics for completed Tasks and performance over time.

Use Cases

User Adds a Task

12. The user clicks 'Add Task'.
13. The system displays a form for title, description, due date/time, and priority.
14. The user enters details and clicks 'Save'.
15. The system vailidates input and creates a new Task.
16. The system displays the Task in the list and calendar views.
17. The system shows a message: 'Task successfully added.'

Variations

1. User Adds a Task

Variation 1.1 – Missing Required Field
1.1 The user leaves the task title empty.
1.1.1 The sysstem displays an error message: "Please enter a task title."
1.1.2 The system keeps the form open and waits for user correction.

Variation 1.2 – Invalid Date or Time
1.2.1 The user enters an invalid due date/time.
1.2.2 The system accepts the date string as entered (flexible date format).
Note: The system allows flexibleee date input and stores the string value even if it cannot be parsed as a standard date format.

2. User Marks Task as Completed

Variation 2.1 – Task Already Completed

2.1.1 The user selects a task that is already marked as completed.
2.1.2 The system allows the action (task can be toggled between completed and pending).

Variation 2.2 – Undo Completion
2.2.1 The user presses "Undo."
2.2.2 The system reverts the task status to "Pending."

3. User Deletes a Task

Variation 3.1 – User Cancels Deletion
3.1.1 The user selects "Cancel" on the confirmation dialog.
3.1.2 The system closes the prompt and returns to the task list without changes.

Variation 3.2 – Permanent Deletion from Trash
3.2.1 The user opens the Trash and selects "Delete Permanently."
3.2.2 The system asks: "This action cannot be undone. Continue?"
3.2.3 Upon confirmation, the system permanently removes the task.

4. User Sets Reminder

Variation 4.1 – Missed Reminder While App Closed
4.1.1 The user had the app closed when a reminder was due.
4.1.2 On the next launch, the system displays missed reminders in a summary popup.

Variation 4.2 – Duplicate Reminder
4.2.1 The user can set multiple reminders for tasks.
4.2.2 The system allows setting reminders without duplicate checking.

5. User Views Calendar

Variation 5.1 – No Tasks in Selected View
5.1.1 The user opens a day, week, or month with no scheduled tasks.
5.1.2 The system displays: "No tasks for this day."

Variation 5.2 – Filter Applied with No Results
5.2.1 The user applies filters that return no tasks.
5.2.2 The system displays an empty task list (no specific message shown).

Glossary

Task: A unit of work containing a title, description, due date, and status.

Calendar: Visual interface for viewing tasks by date and time.

Reminder: A system notification that alerts the user before a task is due.

Priority: Defines the importance of a task (High, Medium, Low).

Status: Represents the state of a task (Pending, Completed, or Trashed).

MVC: ModelViewController, the architectural pattern separating data, logic, and UI.

Command Pattern: Used to support undo and redo actions by encapsulating changes.

Observer Pattern:Used to notify UI components when data models change.

Strategy Pattern: Used to handle dynamic sorting of tasks.

Factory Pattern: Used to create different types of tasks (simple, recurring).

Singleton Pattern: Used for central configuration and command history management.