Assignment 1

Exercise 1:

come up creatively with six different use cases to demonstratate your understanding of the following software design patterns by coding the same.

1. Two use cases to demonstrate two behavioral pattern

2. Two use cases to demonstrate two creational design pattern

3. Two use cases to demonstrate two structural design pattern

Github link of assignment:

<https://github.com/pratikshapawarpc/assignment1>

Exercise 2: Problem Statements for Mini-projects

1. Astronaut Daily Schedule Organizer Programming Exercise

Problem Statement

Design and implement a console-based application that helps astronauts organize their daily schedules. The application should allow users to add, remove, and view daily tasks. Each task will have a description, start time, end time, and priority level. The intent behind this problem statement is to evaluate your ability to implement a basic CRUD (Create, Read, Update, Delete) application, manage data efficiently, and apply best coding practices.

Functional Requirements

Mandatory Requirements

1. Add a new task with description, start time, end time, and priority level.

2. Remove an existing task.

3. View all tasks sorted by start time.

4. Validate that new tasks do not overlap with existing tasks.

5. Provide appropriate error messages for invalid operations.

Optional Requirements

1. Edit an existing task.

2. Mark tasks as completed.

3. View tasks for a specific priority level.

Non-functional Requirements

1. The application should handle exceptions gracefully.

2. Ensure the application is optimized for performance.

3. Implement a logging mechanism for tracking application usage and errors.

Key Focus Design Patterns to be used

1. Singleton Pattern: Ensure there is only one instance of the schedule manager.

2. Factory Pattern: Use a factory to create task objects.

3. Observer Pattern: Notify users of task conflicts or updates.

Detailed Instructions

1. Use the Singleton Pattern to create a ScheduleManager class that manages all tasks.

2. Implement a TaskFactory class to create Task objects.

3. Use the Observer Pattern to alert users if a new task conflicts with an existing one.

Github link of assignment:

<https://github.com/pratikshapawarpc/Assignment-1-exercise-2>

Assignment 1 : links

<https://github.com/pratikshapawarpc/assignment1>

<https://github.com/pratikshapawarpc/Assignment-1-exercise-2>