ASSGNMENT 2
NAME:VANDANA S KAMBLI

Exercise 2: Mini Project

You need to develop an **Astronaut Daily Schedule Organizer** console application with the following requirements:

1. Mandatory Requirements:

- **CRUD Operations**: Implement functionality to add, remove, and view tasks.
- **Time Validation**: Ensure tasks do not overlap.
- **Error Handling**: Provide meaningful error messages for invalid input.

2. Design Patterns Implementation:

- **Singleton Pattern**: Use this to ensure that only one instance of the ScheduleManager class exists.
- **Factory Pattern**: Implement a TaskFactory to create Task objects.
- **Observer Pattern**: Notify users of task conflicts with existing tasks.

Astronaut Daily Schedule Organizer (C#)

Explanation of the Code:

Task Class: Represents a scheduled task with properties for description, start time, end time, priority, and completion status.

ScheduleManager (Singleton): Ensures that there is only one instance managing the schedule. It provides methods to add, remove, and view tasks while checking for overlapping times when adding tasks.

Main Program: Implements the user interaction by creating tasks and invoking methods on the ScheduleManager. It demonstrates adding tasks, handling conflicts, removing tasks, and viewing the schedule.

Code Implementation

ASSGNMENT 2
NAME:VANDANA S KAMBLI

```
using System;
using System.Collections.Generic;

// Task class representing a scheduled task
public class Task
{
    public string Description { get; }
    public DateTime StartTime { get; }
    public DateTime EndTime { get; }
    public string Priority { get; }
    public bool IsCompleted { get; set; }

    public Task(string description, DateTime startTime, DateTime endTime, string priority)
    {
        Description = description;
        StartTime = startTime;
        EndTime = endTime;
        Priority = priority;
        IsCompleted = false;
    }

    public override string ToString()
    {
        return $"{StartTime:HH:mm} - {EndTime:HH:mm}: {Description} [{Priority}] - {(IsCompleted ? "Completed" : "Pending")}";
    }
}

// Singleton class to manage the schedule
public class ScheduleManager
{
    private static ScheduleManager _instance;
    private static readonly object _lock = new object();
    private List<Task> _tasks = new List<Task>();

    private ScheduleManager() { }
```

```
    public static ScheduleManager GetInstance()
    {
        if (_instance == null)
        {
            lock (_lock)
            {
                if (_instance == null)
                {
                    _instance = new ScheduleManager();
                }
            }
        }
        return _instance;
    }

    public void AddTask(Task task)
    {
        foreach (var existingTask in _tasks)
        {
            if (existingTask.StartTime < task.EndTime && task.StartTime < existingTask.EndTime)
            {
                Console.WriteLine($"Error: Task conflicts with existing task '{existingTask.Description}'.");
                return;
            }
        }
        _tasks.Add(task);
        Console.WriteLine("Task added successfully. No conflicts.");
    }

    public void RemoveTask(string description)
    {
        Task taskToRemove = _tasks.Find(t => t.Description == description);
        if (taskToRemove != null)
        {
            _tasks.Remove(taskToRemove);
            Console.WriteLine("Task removed successfully.");
        }
```

ASSGNMENT 2
NAME:VANDANA S KAMBLI

```
        else
        {
            Console.WriteLine("Error: Task not found.");
        }
    }

    public void ViewTasks()
    {
        if (_tasks.Count == 0)
        {
            Console.WriteLine("No tasks scheduled for the day.");
            return;
        }
        _tasks.Sort((t1, t2) => t1.StartTime.CompareTo(t2.StartTime));
        foreach (var task in _tasks)
        {
            Console.WriteLine(task);
        }
    }
}

// Client code
public class Program
{
    public static void Main(string[] args)
    {
        ScheduleManager manager = ScheduleManager.GetInstance();

        // Adding tasks
        manager.AddTask(new Task("Morning Exercise", DateTime.Parse("07:00"), DateTime.Parse("08:00"), "High"));
        manager.AddTask(new Task("Team Meeting", DateTime.Parse("09:00"), DateTime.Parse("10:00"), "Medium"));

        // Viewing tasks
        Console.WriteLine("Viewing all tasks:");
        manager.ViewTasks();
```

```
        // Attempting to add a conflicting task
        manager.AddTask(new Task("Training Session", DateTime.Parse("09:30"), DateTime.Parse("10:30"), "High"));

        // Removing a task
        manager.RemoveTask("Morning Exercise");

        // Viewing tasks after removal
        Console.WriteLine("Viewing all tasks after removal:");
        manager.ViewTasks();

        // Adding more tasks
        manager.AddTask(new Task("Lunch Break", DateTime.Parse("12:00"), DateTime.Parse("13:00"), "Low"));
        manager.AddTask(new Task("Project Review", DateTime.Parse("14:00"), DateTime.Parse("15:00"), "Medium"));

        // Viewing tasks again
        Console.WriteLine("Final task list:");
        manager.ViewTasks();
    }
}
```

Output Preview

Here's the expected output when running the application:

ASSGNMENT 2
NAME:VANDANA S KAMBLI

```
Task added successfully. No conflicts.
Task added successfully. No conflicts.
Viewing all tasks:
07:00 - 08:00: Morning Exercise [High] - Pending
09:00 - 10:00: Team Meeting [Medium] - Pending
Error: Task conflicts with existing task 'Team Meeting'.
Task removed successfully.
Viewing all tasks after removal:
09:00 - 10:00: Team Meeting [Medium] - Pending
12:00 - 13:00: Lunch Break [Low] - Pending
14:00 - 15:00: Project Review [Medium] - Pending
Final task list:
09:00 - 10:00: Team Meeting [Medium] - Pending
12:00 - 13:00: Lunch Break [Low] - Pending
14:00 - 15:00: Project Review [Medium] - Pending
```