**Software Construction and Development Assignment**

**Academic Information**

| **Category** | **Details** |
| --- | --- |
| **Student Name** | Abdullah Khan |
| **Department** | Computer Science |
| **Program** | Software Engineering |
| **Course** | Software Construction & Development (SCD) |
| **Roll Number** | 14923 |
| **Submission Date** | 20 May 2025 |
| **Semester** | Fourth (4th) |
| **Section** | SE4A |
| **Instructor** | Maam Samman |

**Task Management System Implementation**

**Project Overview**

**Objective**

This assignment focuses on designing and implementing a **console-based Task Management System** using industry-standard software development practices. The goal is to demonstrate proficiency in modular design, coding standards, version control, testing, and documentation.

**System Description**

The Task Management System enables users to efficiently organize, track, and modify tasks through an intuitive console interface. Key functionalities include task creation, status updates, filtering, and deletion.

**Software Requirements Specification (SRS)**

**1. Purpose**

The system provides a structured approach for individuals to manage daily tasks offline, improving productivity through streamlined task tracking.

**2. Scope**

The proposed task management system is a **standalone Java-based console application** designed for individual users. It is intended primarily for students, professionals, or anyone seeking to manage their tasks efficiently through a simple, command-line interface. The system will operate entirely within a terminal/console environment without requiring any graphical user interface (GUI) or internet connectivity.  
**Core functionalities will include:**

* Adding and updating tasks
* Deleting tasks
* Viewing a list of tasks
* Filtering tasks based on specific criteria (status or due date)
* Marking tasks as complete or incomplete

This software will be extensible, allowing future enhancements such as GUI integration using JavaFX or Swing and persistent storage using file systems or databases like SQLite via JDBC.

**3. Functional Requirements**

| **Feature** | **Description** |
| --- | --- |
| **Task Creation** | Add tasks with title, description, and due date. |
| **Task Modification** | Edit task details or delete tasks. |
| **Status Management** | Mark tasks as complete/incomplete. |
| **Task Filtering** | View tasks by status or due date. |

**4. Non-Functional Requirements**

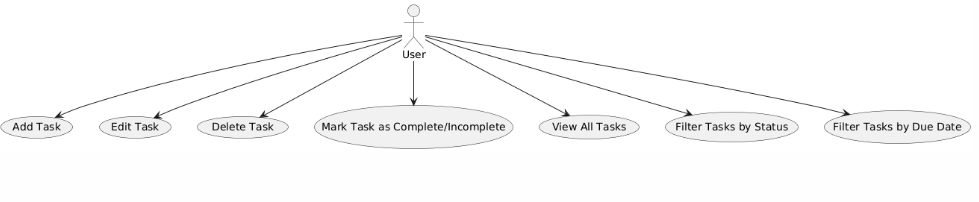
* **Performance**: Fast response to user inputs.
* **Usability**: Simple, menu-driven console interface.
* **Reliability**: Graceful handling of invalid inputs.
* **Portability**: Runs on any system with Python.

**5. System Actors & Use Cases**

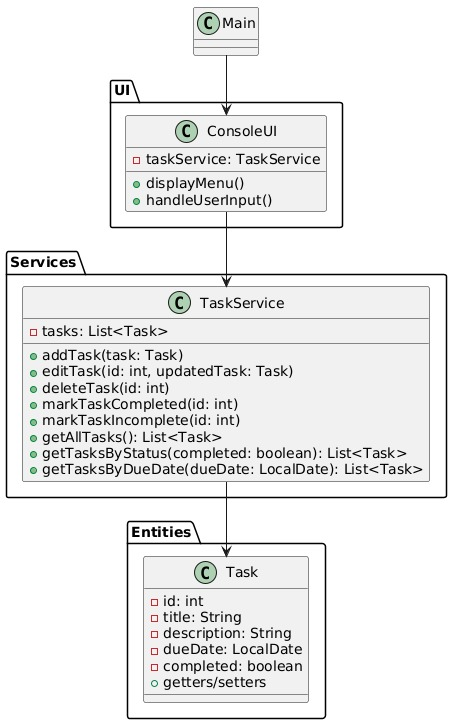
* **Actor**: User
* **Key Interactions**:
  1. **Add Task**
  2. **View/Filter Tasks**
  3. **Edit/Delete Task**
  4. **Update Task Status**

**Diagrams:**

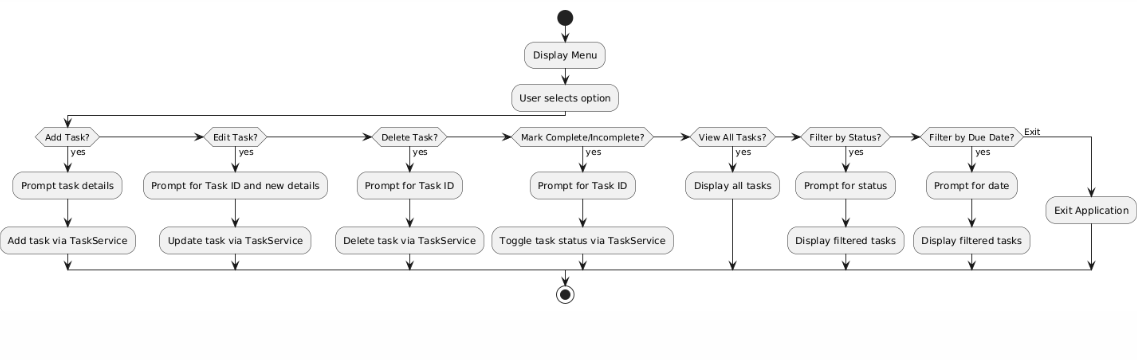
**Usecase:**



**Class Diagram:**

****

**Activity Diagram:**

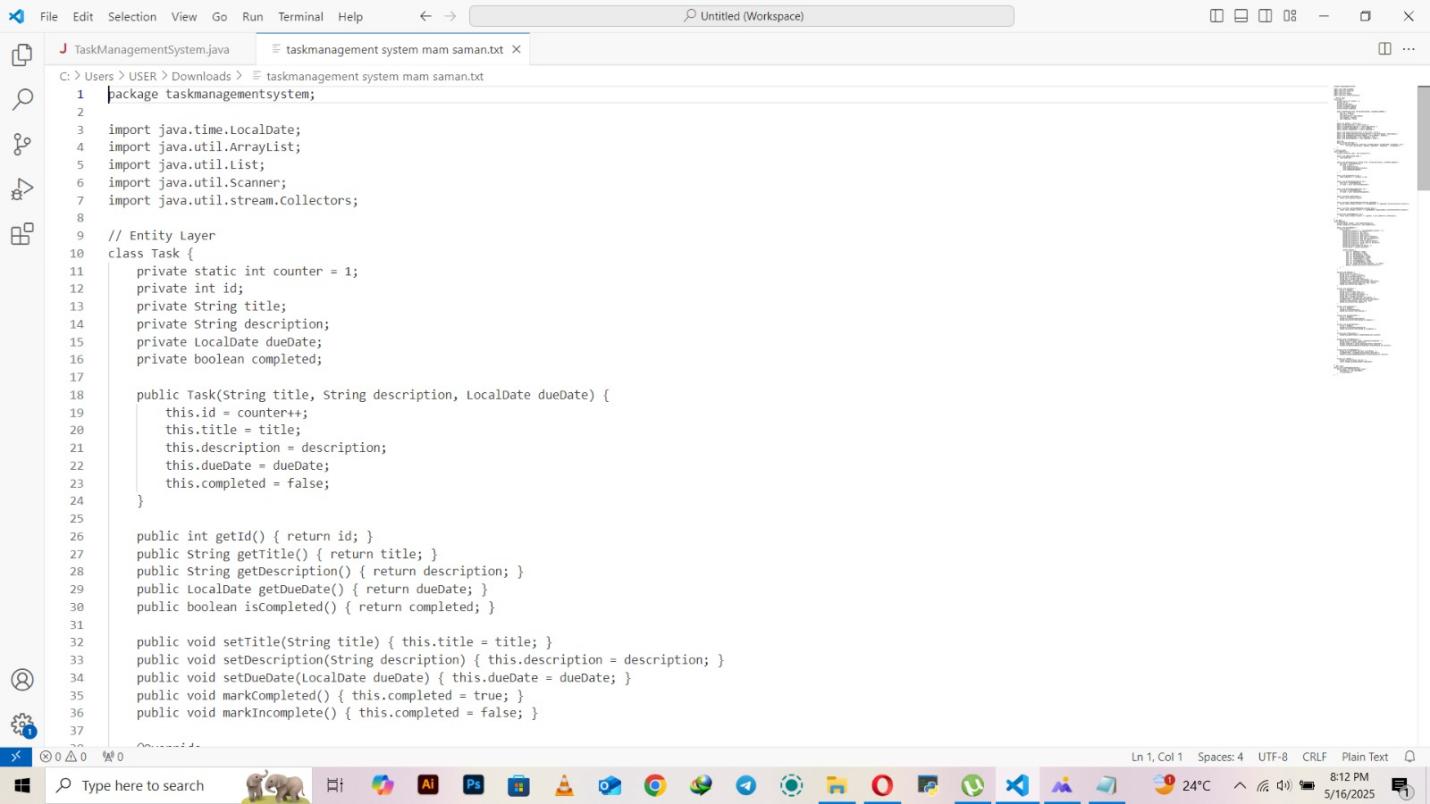


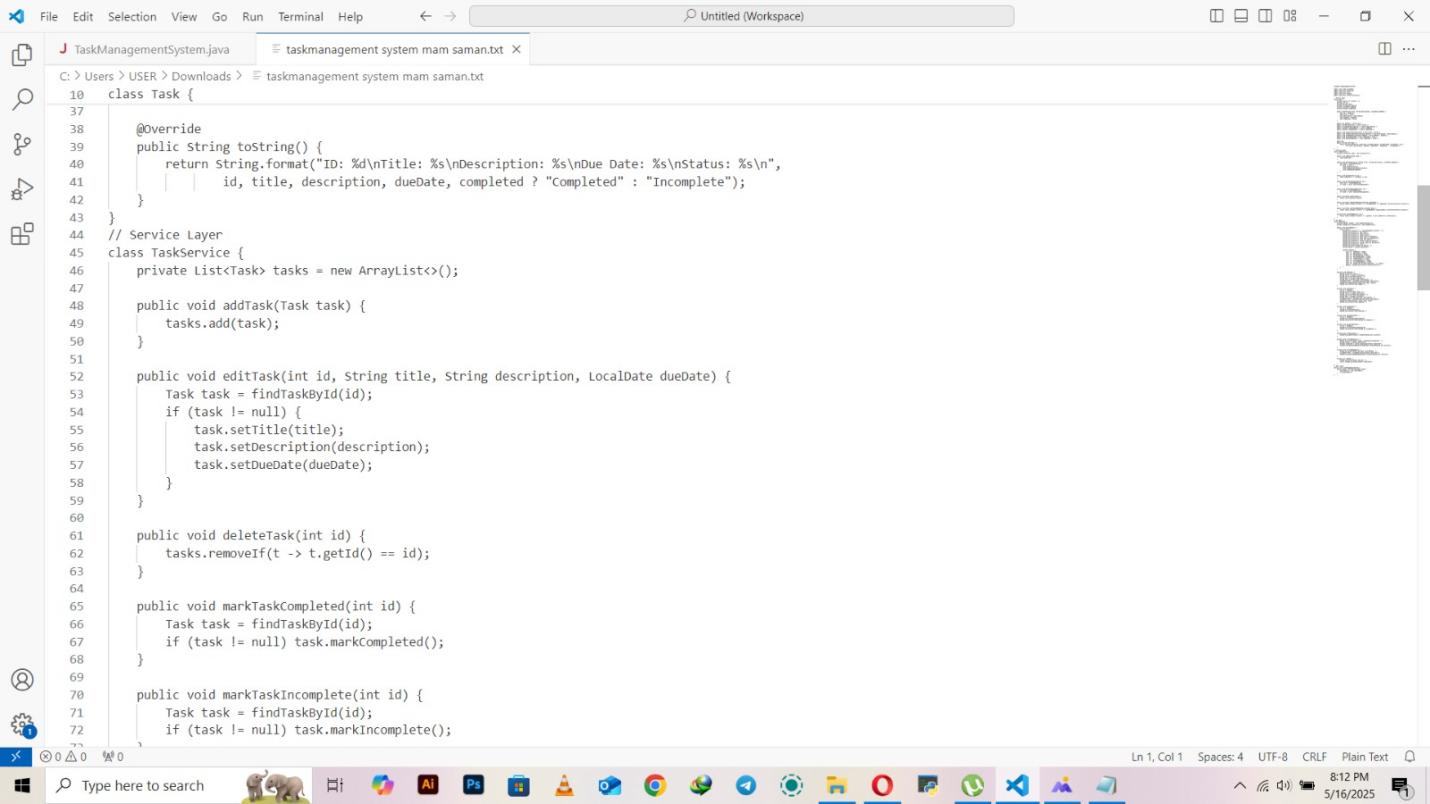
**System Architecture**

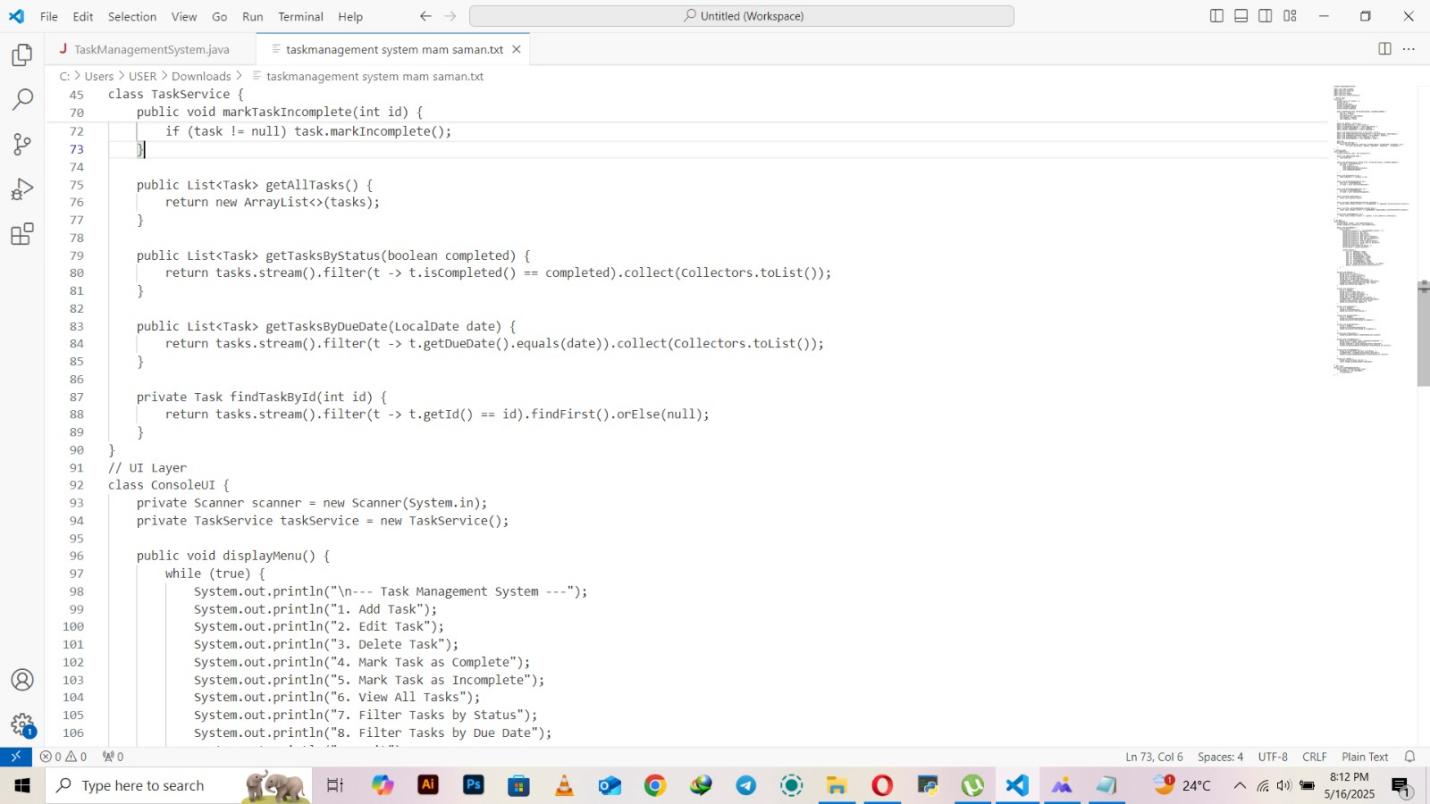
**Modular Design**

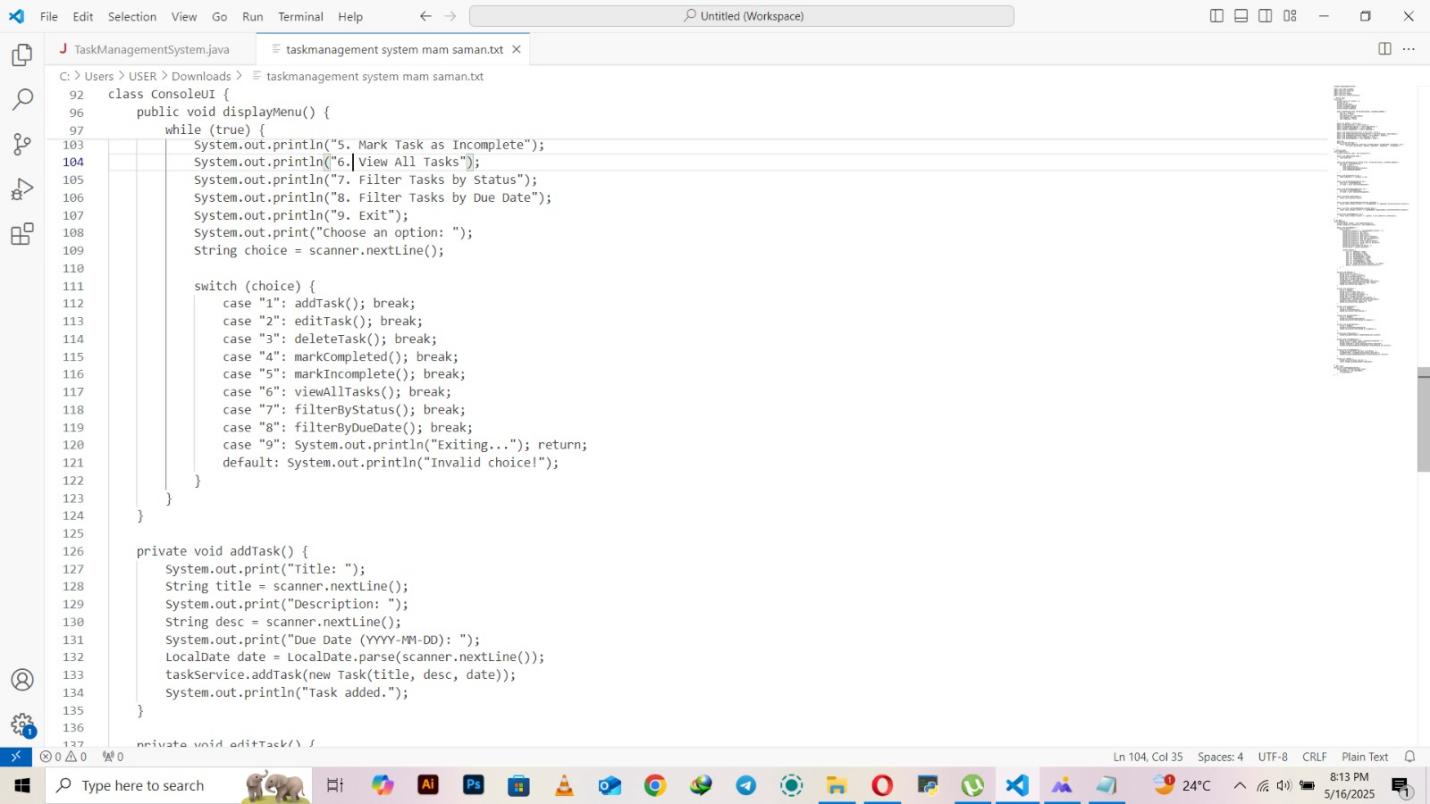
| **Module** | **Responsibility** |
| --- | --- |
| **main.py** | Entry point; handles user input/output. |
| **task.py** | Defines the Task class (attributes/methods). |
| **task\_manager.py** | Manages task operations (CRUD, filtering). |
| **utils.py** | (Optional) Helper functions (e.g., input validation). |

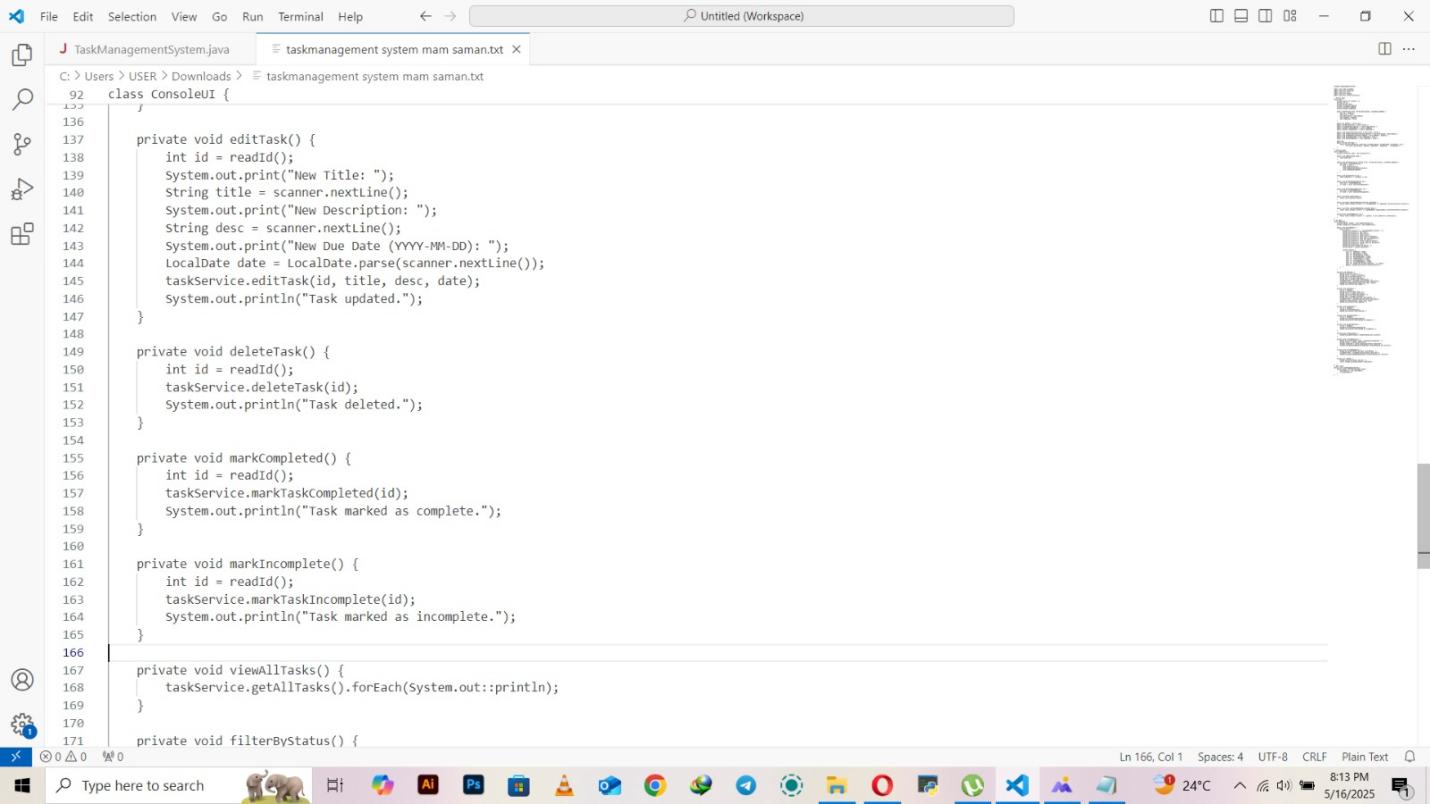
**Code And Interface:**

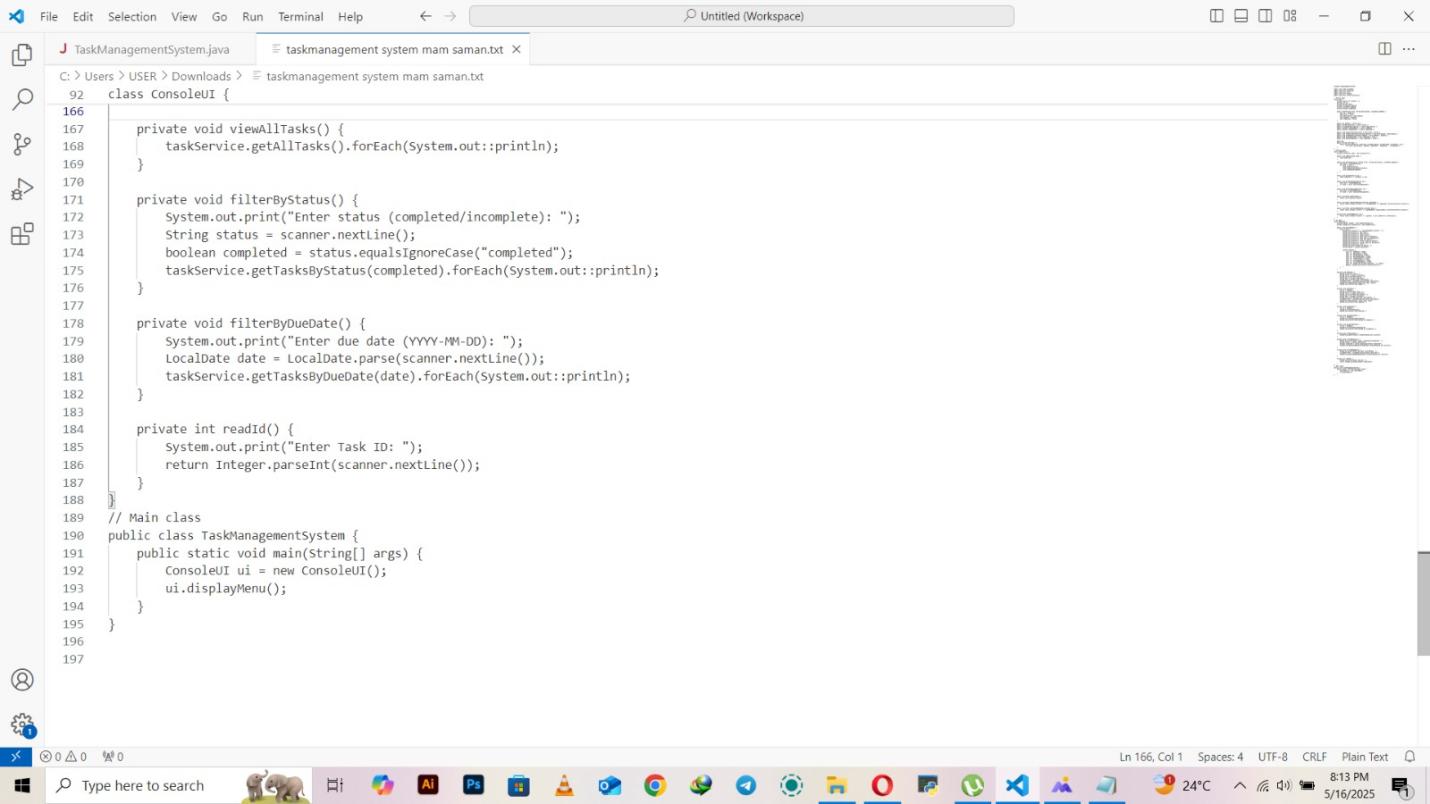
****

****

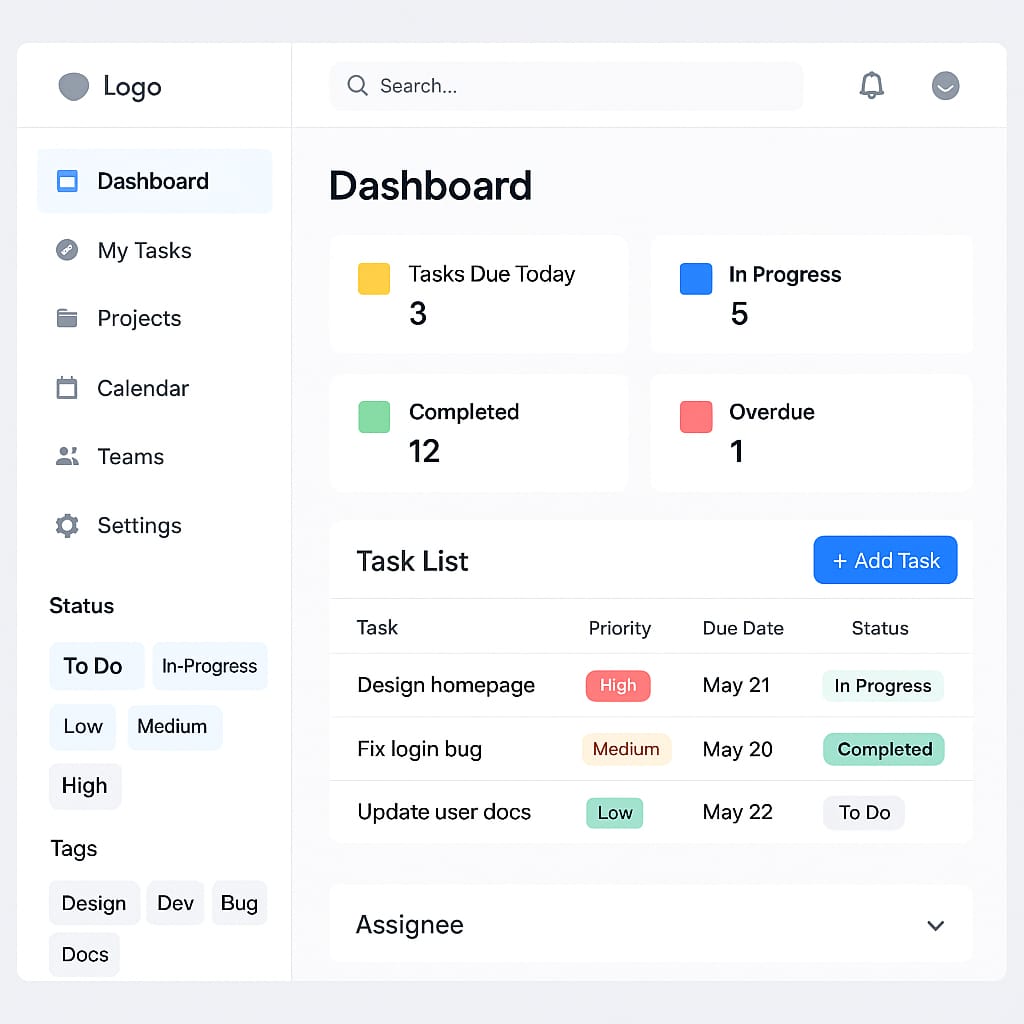
****

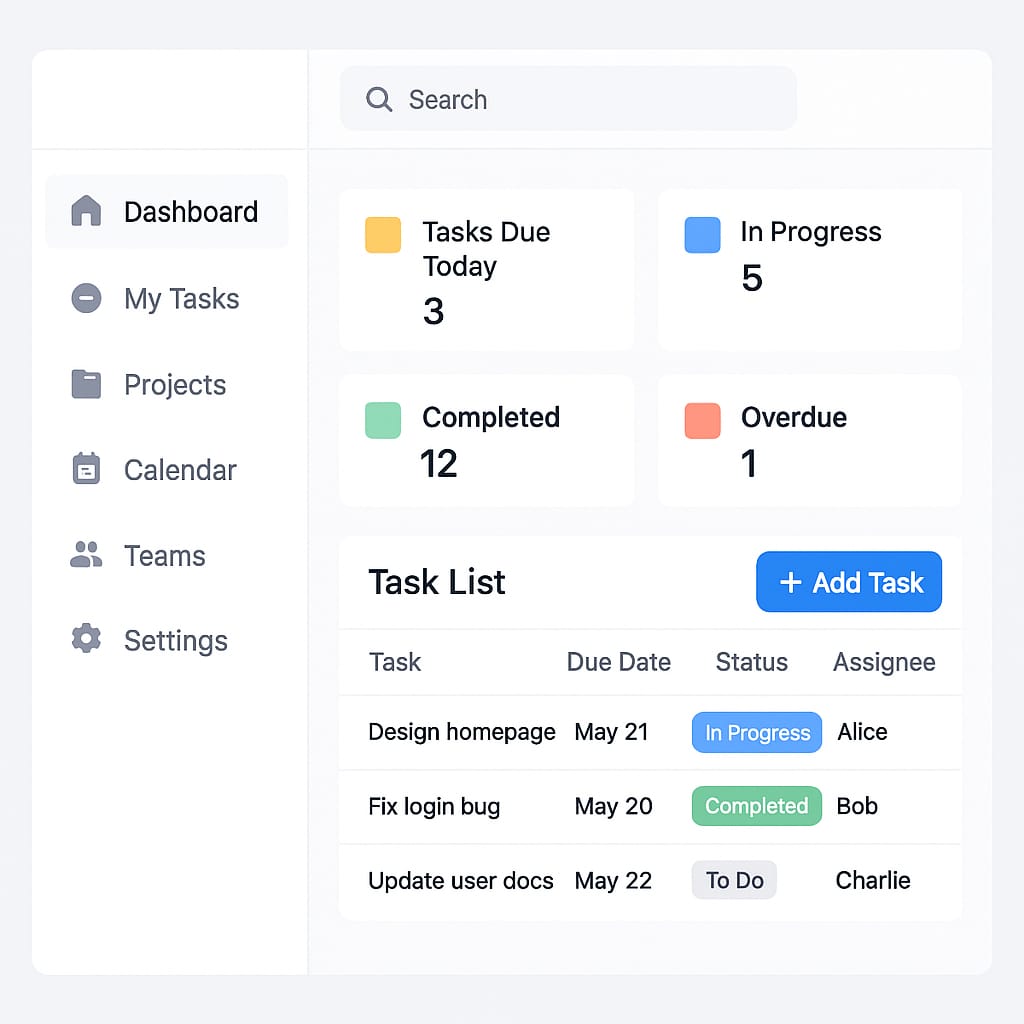
****

****

****

**Interface:**

****

****

**Development Practices**

**Version Control**

* Hosted on **[GitHub](https://github.com/noorul-huda2704/Task-management-System" \t "_blank)**.
* Commit history reflects incremental development:
  + Initial setup
  + Feature implementation (e.g., task filtering)
  + Testing and debugging

**Testing**

* **Unit Tests**: Validate Task class methods (e.g., status updates).

import unittest

from task import Task

class TestTask(unittest.TestCase):

def test\_status\_update(self):

task = Task(1, "Study SCD", "2025-06-10")

task.mark\_complete()

self.assertTrue(task.is\_complete())

* **Error Handling**: Covers invalid IDs, empty titles, and date formats.

**Documentation**

**User Guide**

1. **Run the Application**: Execute main.py.
2. **Menu Options**:
   * 1: Add Task
   * 2: Edit Task
   * 3: Delete Task
   * 6: View All Tasks

**Developer Guide**

* **Extend Features**: Integrate databases for data persistence.
* **Run Tests**: Use unittest framework.

**Implemented Features**

✔ Add/View/Edit/Delete Tasks  
✔ Filter by Status or Due Date  
✔ Console-Based UI