Task Management App

Name:	Mahin Rashid Chowdhury
Project Title:	Console based app - C#
Primary Supervisor:	Nafisur Rahman bhai

1 Introduction

The TaskManager project is a console-based application developed using the C# programming language and the .NET framework. The purpose of this application is to assist users in managing tasks by allowing them to create, view, edit, delete, and query tasks efficiently.The application ensures persistent data storage using JSON serialization and follows object-oriented design principles, particularly focusing on the SOLID principles for maintainability and extensibility.

2 Objectives

The primary objectives of the TaskManager project are:

- To implement a task management system with basic CRUD functionality.
- To utilize JSON for data persistence.
- To perform specific queries on data.
- To apply software engineering principles such as SOLID and enable future extensibility through abstraction and interface-based design.

3 System Architecture

3.1 User Interaction Layer

- Handles all user input and output operations via the console.
- Interfaces with the user through the `IConsoleInteraction` abstraction.
- Implements decoupled interaction logic using `ConsoleInteraction`.

3.2 Application Control Layer

- Provides centralized orchestration of user commands via the `TaskManagerApp` class.
- Coordinates between user input and core business operations.
- Maintains control flow and program lifecycle.

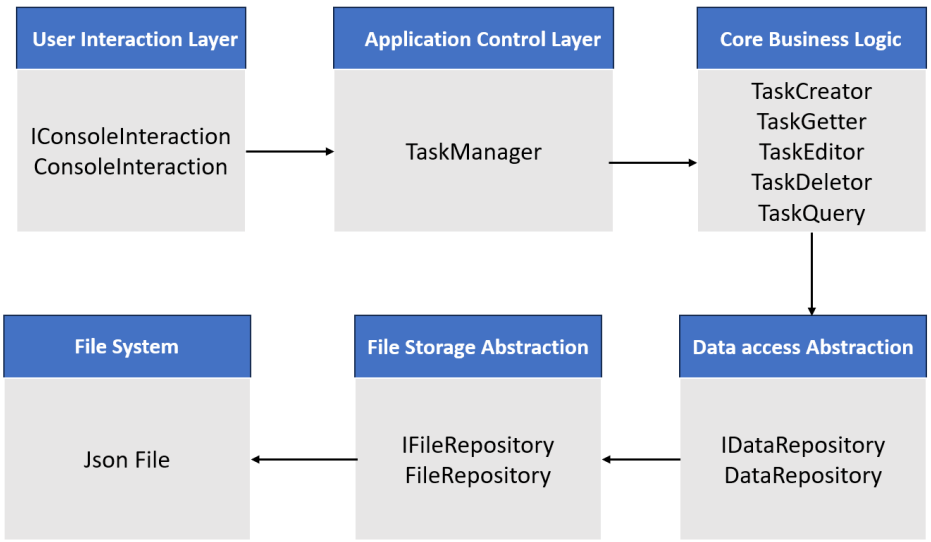

Figure 1: Component-Level Block Diagram

3.3 Core Business Logic

- Contains all business rules and application logic.
- Encapsulated in the `TaskManager` class and divided into modular components:
  - `TaskCreator`: Adds new tasks.
  - `TaskGetter`: Retrieves tasks by ID or list.
  - `TaskEditor`: Updates existing task data.
  - `TaskDeletor`: Removes tasks by ID.
  - `TaskQuery`: Filters tasks by status or date conditions.

3.4 Data Access Abstraction

- Provides an interface `IDataRepository` for data operations.
- Implements task collection management via `DataRepository`.
- Manages an in-memory list of tasks.

3.5 File Storage Abstraction

- Defines `IFileRepository` to abstract physical file access.
- Implements file reading and writing using JSON serialization in `FileRepository`.
- Supports persistence via structured data representation.

3.6 File System

- Stores task data in a local file system.
- Uses JSON format for structured persistence in files such as `tasks.json`.

4 Software Design Principles

The design of the **TaskManager** application adheres to the **SOLID** principles of object-oriented design. These principles improve code maintainability, scalability, and testability.

Principle	Application
Single Responsibility	Each class has a single, well-defined responsibility. For example, `TaskDeletor` is solely responsible for deleting tasks.
Open/Closed	The system is open for extension but closed for modification. New functionality (e.g., alternative input methods) can be added without altering existing code, thanks to interface-based design.
Liskov Substitution	Interface-driven design ensures that derived classes or implementations can substitute base interfaces without altering the correctness of the program.
Interface Segregation	Interfaces are fine-grained and focused. For instance, `IFileRepository` defines only the file operations needed, avoiding bloated abstractions.
Dependency Inversion	High-level modules depend on abstractions (`IDataRepository`, `IFileRepository`) rather than concrete implementations, promoting flexibility and decoupling.

Table 1: Application of SOLID Principles in the TaskManager Design

5 Project Output

5.1 User-specific Queries :

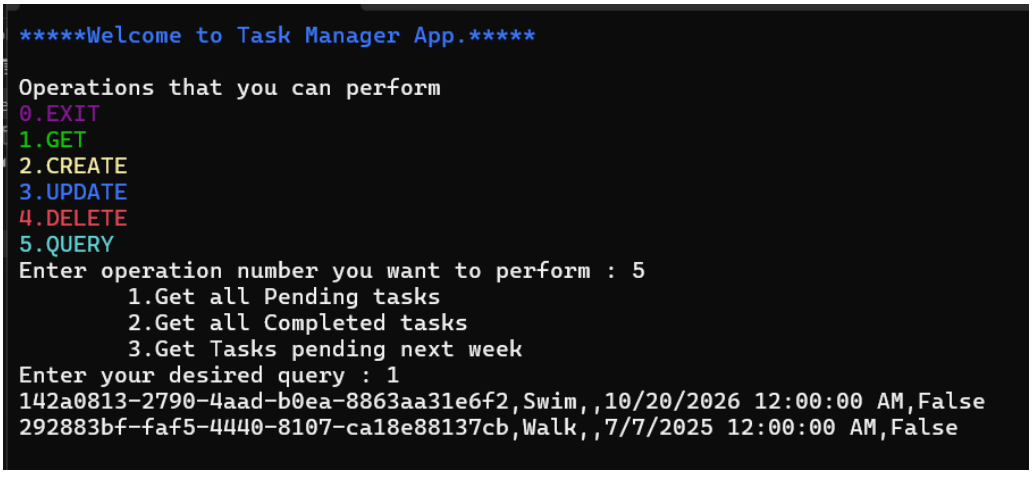

Figure 2: User-specific queries

5.2 All tasks :

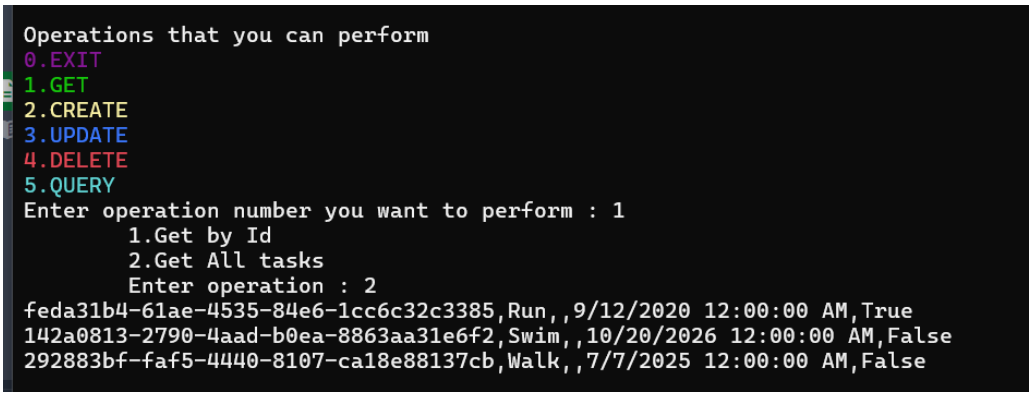

Figure 3: User-specific queries

6 Limitations

1. The application is currently console-based and lacks a graphical user interface.
2. There is limited input validation and error handling (e.g., no ID format check.
3. All tasks are stored in memory, with data saved only at the end of execution unless manually triggered.

7 Conclusion

The TaskManager application successfully implements a modular, extensible, and maintainable task management system using standard C# practices and principles. Through the application of SOLID principles, the project demonstrates clean separation of concerns, adherence to object-oriented paradigms, and readiness for future extension.