King Fahad University of Petroleum and Minerals
Information and Computer Science Department

SWE316: Software Design and Construction (Term 241)

Software Design Document for the TaskAdemic System

HomeWork#03

12/12/2024

Ismael Arqsosi

202182150

Task	Grade	Your Grade	Comments
Software Design Documentation	100		
Checklist and penalties:			
No Cover page with grade table		-10	☐
File name (report)		-05	☐
Not in PDF format		-10	☐
Total	100		

Software Design Document Template

This template for this software design document (SDD) is adopted from the *IEEE Software Engineering Standards Collection*, IEEE Press and other SDD templates.

Table of Contents

❖ Introduction

1.1 Purpose

This Software Design Document (SDD) serves to outline the design and architecture of the Task Matching and Management System. This system is developed to facilitate efficient assignment and tracking of tasks to students while adhering to good software design principles. The document aims to provide a shared understanding of the system's architecture and design decisions among stakeholders, including project managers, software developers, and quality assurance teams.

1.2 Scope

The Task Matching and Management System is designed to automate the process of assigning tasks to students based on their skills and availability. It includes functionalities for reading and writing task-related data from CSV files, assigning tasks to appropriate students, marking tasks as completed, and generating reports. The system integrates key object-oriented design principles like encapsulation, inheritance, and polymorphism, ensuring a maintainable and scalable solution.
The system will:

- Manage student information and categorize them by skills and availability.
- Handle task creation, assignment, and tracking.
- Provide interfaces for marking tasks as completed and generating reports.
- Ensure data consistency by integrating CSV-based data handling for input and output.

1.3 Overview

The Task Matching and Management System simplifies the process of task allocation and completion tracking in academic or organizational settings. It automates matching tasks to the most suitable students based on predefined criteria and ensures the seamless update of task statuses. The system employs core object-oriented principles to maintain a modular and extendable architecture.
This SDD is structured as follows:

- **Introduction:** Details the purpose, scope, and organization of the document.
- **System Architecture:** Describes the high-level structure of the system and its components.
- **Component Design:** Provides detailed descriptions of classes, attributes, methods, and relationships.
- **Requirements Matrix:** Maps the system requirements to the corresponding components to ensure alignment.

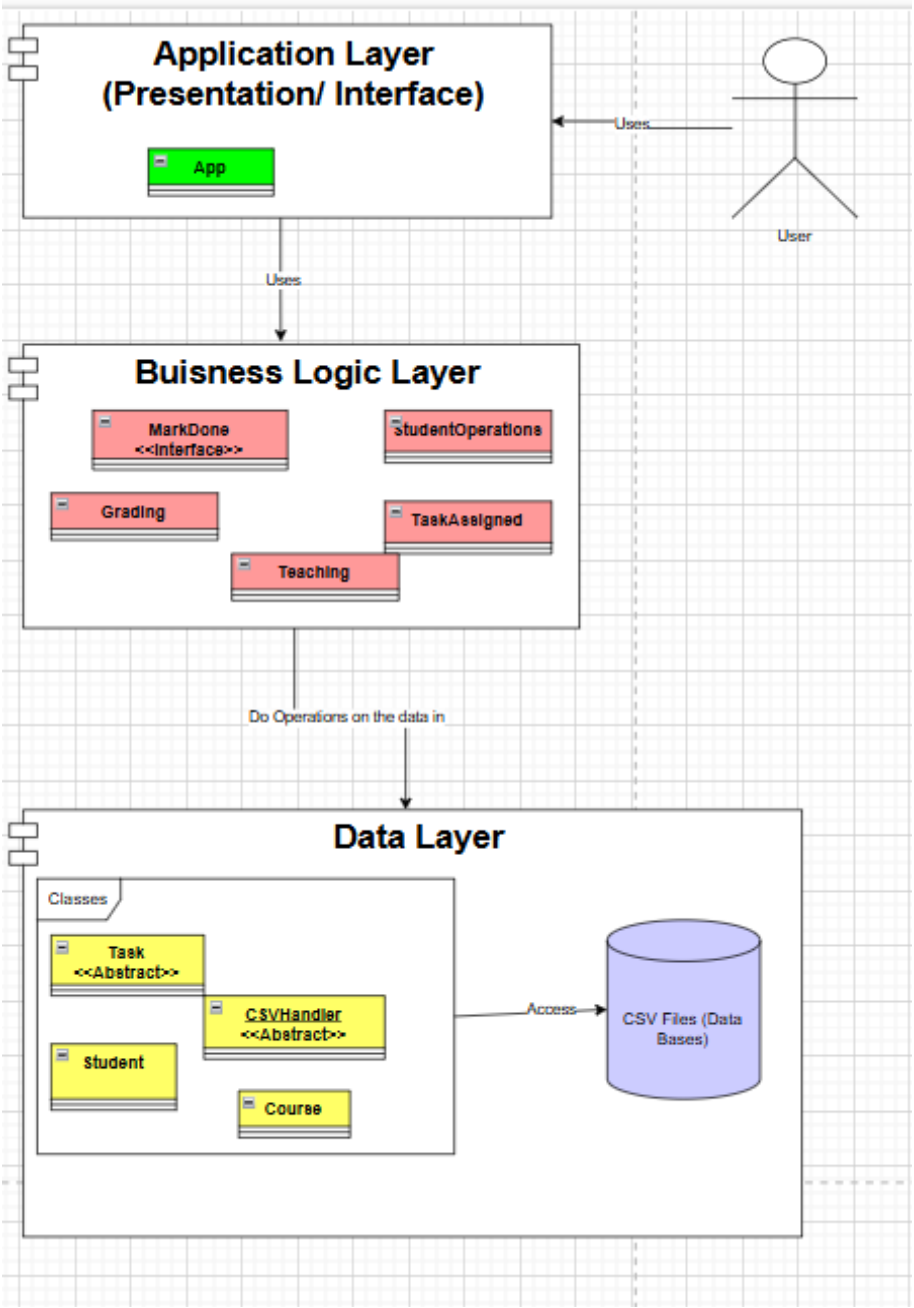
1.4 Constraints

The following constraints have been identified:

1. **Programming Paradigm:** The system is implemented using an object-oriented programming paradigm.
2. **Programming Language:** The system is developed in Java.
3. **Architectural Style:** The design follows a layered architecture with separation of concerns.
4. **Frameworks and Tools:** No external frameworks are mandated; the system uses native Java libraries for file handling and core operations.
5. **Environment:** The system operates on a Windows environment and uses local file storage for data persistence.

❖ System Architecture

2.1. Component Decomposition Description (Layered Architecture)



2.2. Description of the Design

The Task Matching and Management System employs a **layered architecture** designed to separate concerns and improve scalability and maintainability. The system consists of three primary layers:

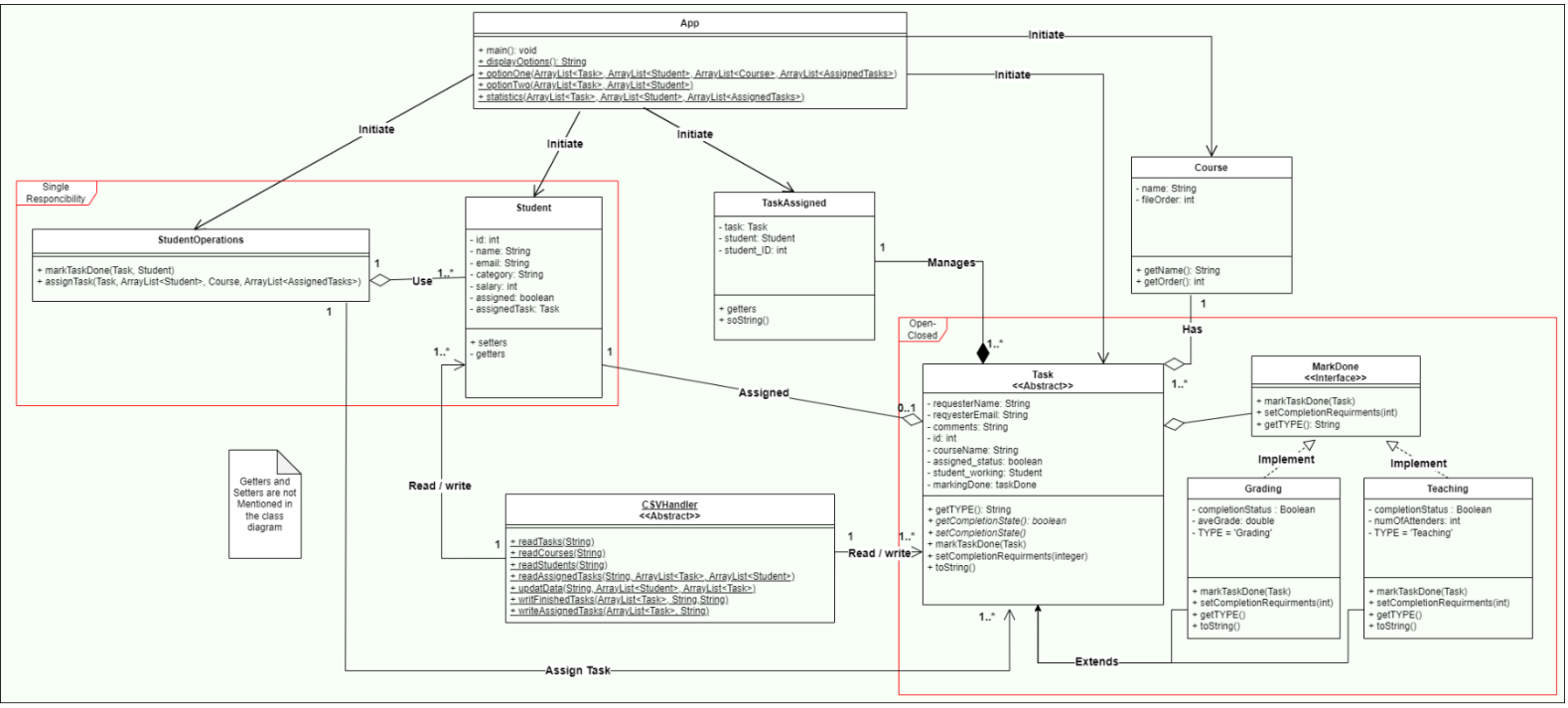
1. **Application Layer (Presentation/Interface Layer):**
    - Represents the user interface that interacts with the end user.
    - **Component:**
        - **App:** The application serves as the medium through which the user can perform operations, such as assigning tasks, marking tasks as complete, and managing students.
2. **Business Logic Layer:**
    - Handles the core functionality and logic of the system.
    - Components:
        - **MarkDone (Interface):** Defines the operations for marking tasks as completed.
        - **StudentOperations:** Manages the logic for student-related operations, such as assigning and retrieving tasks.
        - **TaskAssigned:** Implements the task assignment logic based on student availability and skills.
        - **Teaching:** Handles the specific logic for tasks related to teaching.
        - **Grading:** Implements the logic for grading tasks.
3. **Data Layer:**
    - Responsible for managing and persisting data in the system.
    - Components:
        - **Task (Abstract Class):** Represents the base class for tasks, containing common attributes and methods.
        - **CSVHandler (Abstract Class):** Handles file operations, including reading from and writing to CSV files.
        - **Student:** Stores student-related data, such as ID, name, and assigned tasks.
        - **Course:** Manages course-related information, such as course names and associated tasks.
    - Data Storage:
        - The system uses CSV files for storing and retrieving data, such as task and student information.

2.2 Communication Between Layers

- The **Application Layer** interacts with the **Business Logic Layer** to request operations such as task assignments and marking tasks as completed.
- The **Business Logic Layer** utilizes the **Data Layer** to retrieve and update data stored in the CSV files.
- This separation ensures a clear boundary between presentation, logic, and data storage, reducing **coupling** and making the system easier to maintain.

❖ Component Design

3.1. Class Diagram



3.2. Top-Down Description

The component design adheres to key object-oriented design (OOD) principles that enhance the system's maintainability, scalability, and overall design quality. This section focuses on two critical principles—**Single Responsibility Principle (SRP)** and **Open-Closed Principle (OCP)**—as well as their contribution to cohesion, coupling, and efficient design practices.

Single Responsibility Principle (SRP)
The **Single Responsibility Principle** ensures that each class and module in the system has a well-defined responsibility. This approach enhances modularity and simplifies maintenance.

- **Examples in the Design:**
  - **StudentOperations Class:** Handles only operations related to students, such as assigning tasks and marking tasks as completed. This separation ensures the logic for managing students is independent of other functionalities like file handling or task creation.
  - **CSVHandler Class:** Exclusively responsible for reading and writing data to and from CSV files. By isolating file operations, it reduces dependencies and ensures changes to the file format do not affect other system components.
  - **Task Class and Subclasses:** The Task class defines shared attributes and methods, while specific behaviors (e.g., teaching vs. grading tasks) are handled in subclasses like Teaching and Grading. This separation ensures task-specific logic is encapsulated within the relevant subclass.

By adhering to SRP, the system achieves **high cohesion**, as each class focuses on a single concern. This focus simplifies debugging, testing, and extending functionalities.

Open-Closed Principle (OCP)
The **Open-Closed Principle** ensures that components are open for extension but closed for modification. This design strategy allows new functionality to be added without altering existing code, reducing the risk of introducing errors.

- **Examples in the Design:**
  - **Task and Subclass Hierarchy:** The Task class is designed as an abstract class with a MarkDone interface. New types of tasks can be added as subclasses without modifying the core logic. For example, a new task type, such as ResearchTask, can inherit from Task and implement its specific behavior without altering the existing system.
  - **Polymorphic Methods:** The use of polymorphism in the markTaskDone() method ensures that task completion logic can vary depending on the subclass. This design avoids hardcoding logic for different task types in a single method, enhancing scalability.

By following OCP, the system minimizes **coupling** and facilitates **extensibility**, allowing new features to be integrated seamlessly.

Contribution to Cohesion and Coupling

- **Cohesion:** Each class is tightly focused on its defined responsibility. For instance, CSVHandler focuses solely on file operations, while StudentOperations is dedicated to managing students and tasks. High cohesion improves code readability and reusability.
- **Coupling:** The system minimizes dependencies between classes by defining clear relationships and interfaces. For example, the StudentOperations class interacts with Task and Student classes only through well-defined methods. Low coupling enhances maintainability and ensures changes in one module have minimal impact on others.

4. Human Interface Design

4.1. Screen Images

```
Option#1: Run the automated Task Matching
Option#2: Mark a Task as Completed
2
Please write your ID
334376
The Incompleted task is: ICS 321, and the type is: Grading
Write 'Complete' to mark task completed or 'Quit' to terminate the program
Complete
Write the Average Grades:
55
Write the remark:
good job
The data now in Sheet5
```

```
Option#1: Run the automated Task Matching
Option#2: Mark a Task as Completed
1
No students assigned
Matching process is done
======================== Statistics ====================
Students Information --->
Student ID: 112143, Name: Zainab Alawal, Email: g112143@kfupm.edu.sa;, category: Category I, is Assigned Task? false
--------------------------------------------------------------------------
Student ID: 112154, Name: Aamina Alawal, Email: g112154@kfupm.edu.sa;, category: Category I, is Assigned Task? true
--------------------------------------------------------------------------
Student ID: 110643, Name: Fatima Alawal, Email: g110643@kfupm.edu.sa;, category: Category I, is Assigned Task? false
--------------------------------------------------------------------------
Student ID: 116454, Name: Etab Alawal, Email: g116454@kfupm.edu.sa;, category: Category I, is Assigned Task? false
--------------------------------------------------------------------------
Student ID: 334365, Name: Fahad Alawal, Email: g334365@kfupm.edu.sa;, category: Category II, is Assigned Task? false
--------------------------------------------------------------------------
Student ID: 334376, Name: Sultan Alawal, Email: g334376@kfupm.edu.sa;, category: Category II, is Assigned Task? true
--------------------------------------------------------------------------
Student ID: 338654, Name: Naif Alawal, Email: g338654@kfupm.edu.sa;, category: Category II, is Assigned Task? true
--------------------------------------------------------------------------
Student ID: 339787, Name: Turki Alawal, Email: g339787@kfupm.edu.sa;, category: Category I, is Assigned Task? false
--------------------------------------------------------------------------

==================
Tasks Information --->
Task ID: 1, Type: GradingCourse: ICS 321, is Done? false, is Assigned? true
----------------------------------------------------------------
Task ID: 2, Type: TeachingCourse: ICS 104, is Done? false, is Assigned? true
----------------------------------------------------------------
Task ID: 3, Type: TeachingCourse: ICS 104, is Done? false, is Assigned? true
----------------------------------------------------------------
Task ID: 4, Type: GradingCourse: SWE 326, is Done? false, is Assigned? false
----------------------------------------------------------------
Assigned Tasks Information --->
Student ID: 334376, Task ID: 1, is Done? false
----------------------------------------------------------------
Student ID: 338654, Task ID: 2, is Done? false
----------------------------------------------------------------
Student ID: 112154, Task ID: 3, is Done? false
----------------------------------------------------------------
```

4.2. Report format

Reading Files:

1- Sheet1: Which has the student informations
2- Sheet2: Survay results including the proffeciency leve of each student in Sheet1
3- Sheet3: Tasks provided by the departments, and which will be linked to the student in Sheet1 based on the results in Sheet2

Reports:

- Sheet4: Assigned Tasks from sheet3 to student from Sheet1. This will be generated in the program.
-
- Sheet5: which will be generated if a student completed a task.

5. Requirements Matrix

R01: Marking task Done
R02: Reading Data from the reports
R03: Writing the report of the system
R04: Assigning the Students a task
R05: Showing User Options to choose
R06: Authenticate User
R07: Create new task
R08: Priortizing students
R09: Linking Students to Tasks

Classes / Requirements	R01	R02	R03	R04	R05	R06	R07	R08	R09
Student	1								
Task									1
Course									1
Grading									1
Teaching									1
MarkDone	1								
CSVHandler		1	1						
App					1	1			
TaskOperations									1
StudentOperations								1	