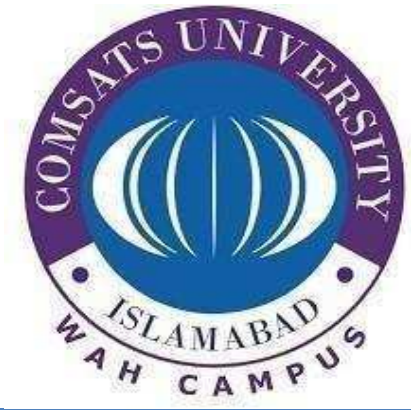
COMSATS UNIVERSITY ISLAMABAD WAH CAMPUS



Department of Computer Science

SOFTWARE REQUIREMENTS DESIGN SPECIFICATION

(SRDS) DOCUMENT

SUBMITTED BY :

Ayesha Tariq (SP23-BSE-027)

Walia Faisal (SP23-BSE-048)

Farhad Ali (SP23-BSE-038)

Rizwan Habib (SP23-BSE-024)

SUBMITTED TO:

Sir Wasif Nisar

Table Of Content

CONTENTS **2**

1 INTRODUCTION **3**

1.1 SYSTEM INTRODUCTION 3
1.2 BACKGROUND OF THE SYSTEM ..... **ERROR! BOOKMARK NOT DEFINED.**
1.3 OBJECTIVES OF THE SYSTEM ..... **ERROR! BOOKMARK NOT DEFINED.**
1.4 SIGNIFICANCE OF THE SYSTEM ..... **ERROR! BOOKMARK NOT DEFINED.**

2 OVERALL DESCRIPTION **4**

2.1 PRODUCT PERSPECTIVE 3
2.2 PRODUCT SCOPE 4
2.3 PRODUCT FUNCTIONALITY 4
2.4 USERS AND CHARACTERISTICS 5
2.5 OPERATING ENVIRONMENT 6

3 SPECIFIC REQUIREMENTS **7**

3.1 FUNCTIONAL REQUIREMENTS 8
3.2 BEHAVIOUR REQUIREMENTS 9
3.3 EXTERNAL INTERFACE REQUIREMENTS 10

4 OTHER NON-FUNCTIONAL REQUIREMENTS **11**

4.1 PERFORMANCE REQUIREMENTS 12
4.2 SAFETY AND SECURITY REQUIREMENTS 12
4.3 SOFTWARE QUALITY ATTRIBUTES 13

5 DESIGN DESCRIPTION **13**

5.1 LOGICAL VIEWPOINT 14
5.2 INFORMATION VIEWPOINT 14
5.3 INTERACTION VIEWPOINT 15
5.4 STATE DYNAMIC VIEWPOINT 15
5.5 ALGORITHM VIEWPOINT 16
5.6 COMPOSITE VIEWPOINT 16

1 Introduction

1.1 System Introduction

The purpose of this document is to present the detailed description of the **Student Database Management System (SDMS).** It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

This system constitutes a simple **Student Management System (SMS)** with functionalities to enter, display, search, update, and delete student records.

1.2 Background of the System

A student management system stores and tracks students' workload, personal information, grades, records, and more. It's a means of streamlining the work and tracking all the data generated by a student, consolidating everything into one system rather than multiple records.

1.3 Objective of the System

- Develop a straightforward Student Management System.
- Provide a user-friendly interface for entering, displaying, searching, updating, and deleting student records.
- Implement a file-saving feature to store data persistently in a "database.txt" file.
- Ensure data continuity across program runs.
- Offer a simple and practical tool for managing student information in an educational context.

1.4 Significance of the System

- Efficiently manages and organizes student information.
- Provides a practical and user-friendly Student Management System.
- Streamlines tasks such as data entry, display, search, update, and deletion of student records.
- Ensures data continuity through a file-saving mechanism.
- Valuable tool for educational administrators to maintain and access student details systematically.

2 Overall Description

2.1 Product Perspective

The Student Management System serves as an essential tool within an educational institution, aiming to streamline student data management. The product perspective revolves around its role in maintaining, updating, and organizing student information efficiently. The system interacts primarily with administrative staff, teachers, and staff responsible for student record maintenance. It operates within the educational institution's ecosystem, acting as a centralized database that handles student details, including names, roll numbers, courses, classes, and contact information.

2.2 Product Scope

The scope of a Student Database Management System (DBMS) encompasses various aspects related to managing student information within educational institutions. This system aims to be highly customizable and scalable, allowing institutions to tailor the system according to their specific needs and adapt it as their requirements grow.

2.3 Product functionality

This software will provide the following functionalities

- **Data Entry (enter()):** Allows the user to input details for multiple students, including their name, roll number, desired course, class, and contact number. If there are existing records, it appends new entries to the existing data.
- **Data Display (show()):** Reads the data from the "database.txt" file and displays it in a formatted manner, showing the student details such as name, roll number, course, class, and contact.
- **Data Search (search()):** Allows the user to search for a specific student's details by entering their roll number. If found, it displays the corresponding student's information.
- **Data Update (update()):** Enables the user to update the information of a specific student by entering their roll number. It displays the existing information and prompts for new details to replace the old data.
- **Data Deletion (deleterecord ()):** Offers two deletion options: deleting all records or deleting a specific student's record by providing their roll number. It removes the selected records from the stored data.

2.4 Users and Characteristics

In SDMS, the following users will be using the system frequently for various tasks,

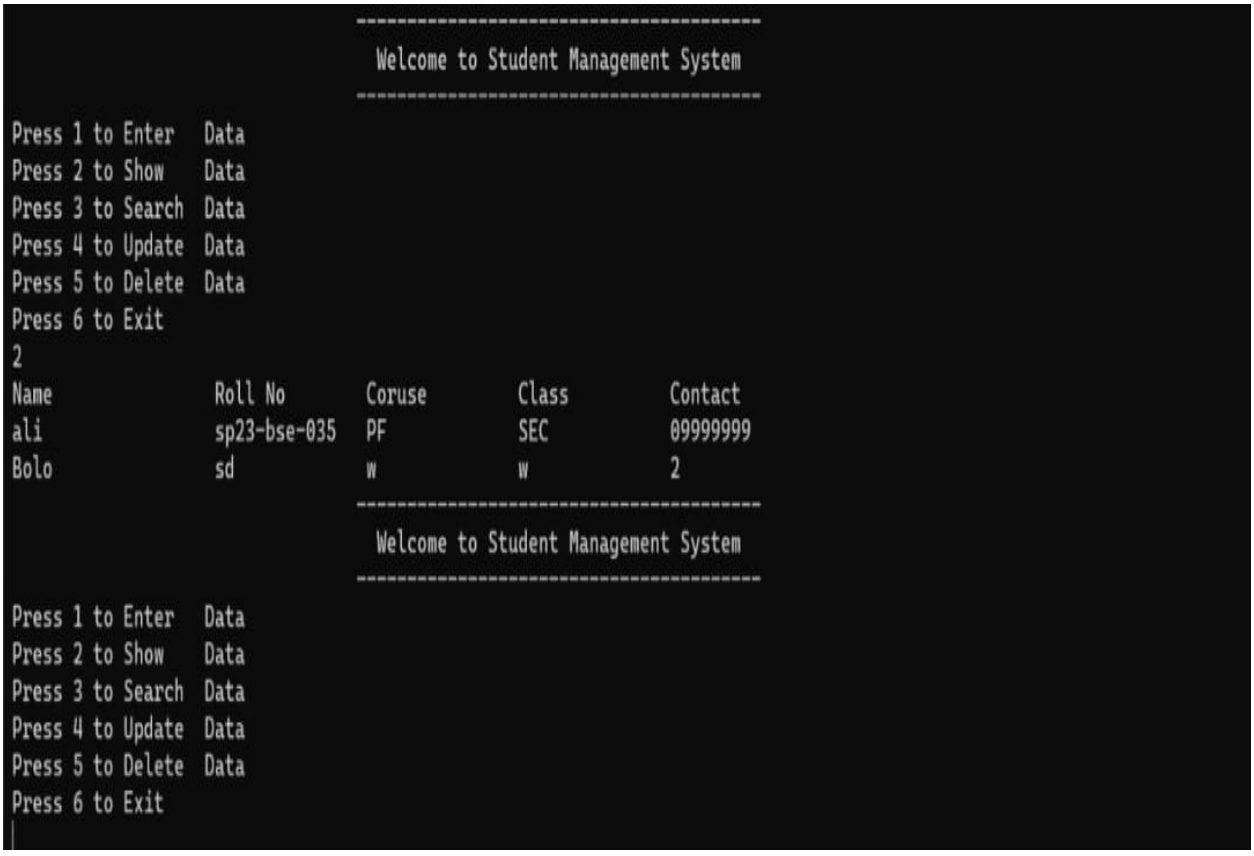
- **Administrator/Teacher:** Primary user responsible for data entry, updating student information, and managing the system.

- **Educational Advisors:** They may use the system to access student information, perform searches, and update student details occasionally. They need read-only access to retrieve student data for advisory purposes.
- **Students (Limited Access):** Depending on the system design, students might have limited access to view their own information or perform certain actions like updating contact details or course preferences.

2.5 Operating Environment

The system provides a basic Student Management System that operates in a console environment, utilizing C++ programming language with standard input/output mechanisms. This system doesn't have high hardware or software requirements and can run on most modern computers with basic C++ compiler support and access to the filesystem for data storage.

Main Function



3 Specific Requirements

3.1 Functional Requirements

Here are the functional requirements categorized into different functional areas for the Student Management System

3.1.1 User Data Management:

1. **Enter Student Data**

    - Allows users to input student information:
    - Name, Roll number, Course, Class, Contact details.

2. **Display Student Data**
    - Retrieves and displays all student records stored in the system

3. **Search for Student Data**
    - Enables users to search for specific student records using their Roll number.
    - Displays details if a match is found.

4. **Update Student Data**
    - Permits users to modify existing student information by Roll number.

5. **Delete student Data**
    - Offers options to delete either specific student records or all records at once.

3.1.2 Data Storage and File Handling:

1. **Save Data to File**

    - Appends student data to a file named "database.txt" for persistent storage.

2. **Read Data from File**

    - Retrieves stored student records from the "database.txt" file for display and manipulation.

3.1.3 User Interface and Interaction:

1. **Menu System**

- Presents a user-friendly menu with options to perform various operations.
- Provides a choice-driven interface for users to interact with the system.

2. **Input Validation**
    - Ensures valid and appropriate inputs are accepted during data entry to maintain data integrity.

3.2 Behavior Requirements

Behavioral requirements includes the Use-Case Diagram. Creating a use case diagram for the Student Management System involves identifying the system's functionalities and the actors interacting with it. Here's a brief description of use cases and actors, followed by a use case diagram

Actors:

1. **Administrator:** Manages and oversees the student data.

2. **Student:** Interacts with the system to provide or update their information.

Use Cases:

Enter Student Data:

- *Actor:* Administrator
- *Description:* Allows the administrator to input details for multiple students into the system.

Display Student Data:

- *Actor:* Administrator
- *Description:* Presents all stored student records for viewing.

Search for Student Data:

- *Actor:* Administrator
- *Description:* Enables the administrator to search for a student's information using their Roll number.

Update Student Data:

- *Actor:* Administrator
- *Description:* Permits the administrator to modify existing student information based on Roll number.

Delete Student Data:

- *Actor:* Administrator
- *Description:* Allows the administrator to delete specific student records or all records.

Exit System:

- *Actor:* Administrator, Student
- *Description:* Terminates the system's operation.

3.3 External Interface Requirements

3.3.1 User Interfaces

The user interacts with the system through a command-line interface (CLI) using the console. This interface allows users to enter commands and view output based on the actions they select from the menu presented.

3.3.2 Hardware Interfaces

This system does not specify any direct hardware dependencies or interfaces.

3.3.3 Software Interfaces

- **C++ Standard Libraries:** The program utilizes various C++ standard libraries like <iostream>, <fstream>, <iomanip>, and <sstream> for input/output, file handling, formatting, and string manipulation.
- **File System Interface:** Utilizes file I/O functionalities provided by the operating system to read and write data to the "database.txt" file.

3.3.4 Communication interfaces

No external communication or network interfaces involved

4 Other Non-functional Requirements

4.1 Performance Requirements

Here are some performance requirements based on the functionalities and user interactions within the Student Management System:

1.Data Entry Speed:

Each data entry for a single student (name, roll number, course, class, contact) will take no longer than 30 seconds per student entry to ensure efficient data input.

2.Data Retrieval Timing:

Retrieving and displaying a student's information after a search operation will take less than 3 seconds for optimal user experience.

3. File Saving Efficiency:

The time taken to save data to the "database.txt" file should not exceed 5 seconds, regardless of the number of records being saved.

4. Deletion Performance:

Deleting a single student's record should be completed within 10 seconds while deleting all records should be done in under 15 seconds.

5. Data Update Timing:

Updating a student's information should take no longer than 20 seconds, including the display of previous data and the entry of new data.

These performance requirements aim to optimize the user experience by ensuring efficient data handling, quick responses to user requests, and smooth interactions within the system. These timings are approximate and can be adjusted based on the system's actual performance and resource capabilities.

4.2 Safety and Security Requirements

Safety and security are crucial aspects of any system, especially one handling sensitive student data. Here are some safety and security requirements.

4.2.1 Safety Requirements:

- **Data Integrity and Backups:**

Regular automated backups of the student database must be performed to prevent data loss in case of system failures.

- **Prevention of Data Leakage:**

Confidential student information (e.g., grades, personal contact details) must be securely stored and accessible only to authorized personnel.

4.2.2 Security Requirements:

- **Data Encryption:**

Sensitive information stored in the database should be encrypted to protect student privacy and prevent unauthorized access.

- **Secure File Storage:**

Any additional files (documents, reports) related to student records should be stored securely, and accessible only to authorized users.

4.3 Software Quality Attributes

4.3.1 Reliability

- **Data Persistence:**

The system have mechanisms to ensure data consistency and persistence even during unexpected shutdowns or failures. Regular backups will be created every 24 hours.

- **Error Recovery:**

The application also handle errors gracefully without crashing, providing descriptive error messages to users and logging errors for debugging purposes.

4.3.2 Maintainability

- **Modularity and Documentation:**

Codebase will be structured into modules following a clear, documented architecture to facilitate future updates or modifications**.**

4.3.3 Usability

- **User Interface Responsiveness:**

The interface must respond to user interactions within 0.5 seconds to ensure a seamless user experience.

4.3.4 Security

- **Encryption and Authentication:**

All sensitive data stored in the database will be encrypted using industry-standard encryption algorithms.

4.3.5 Portability

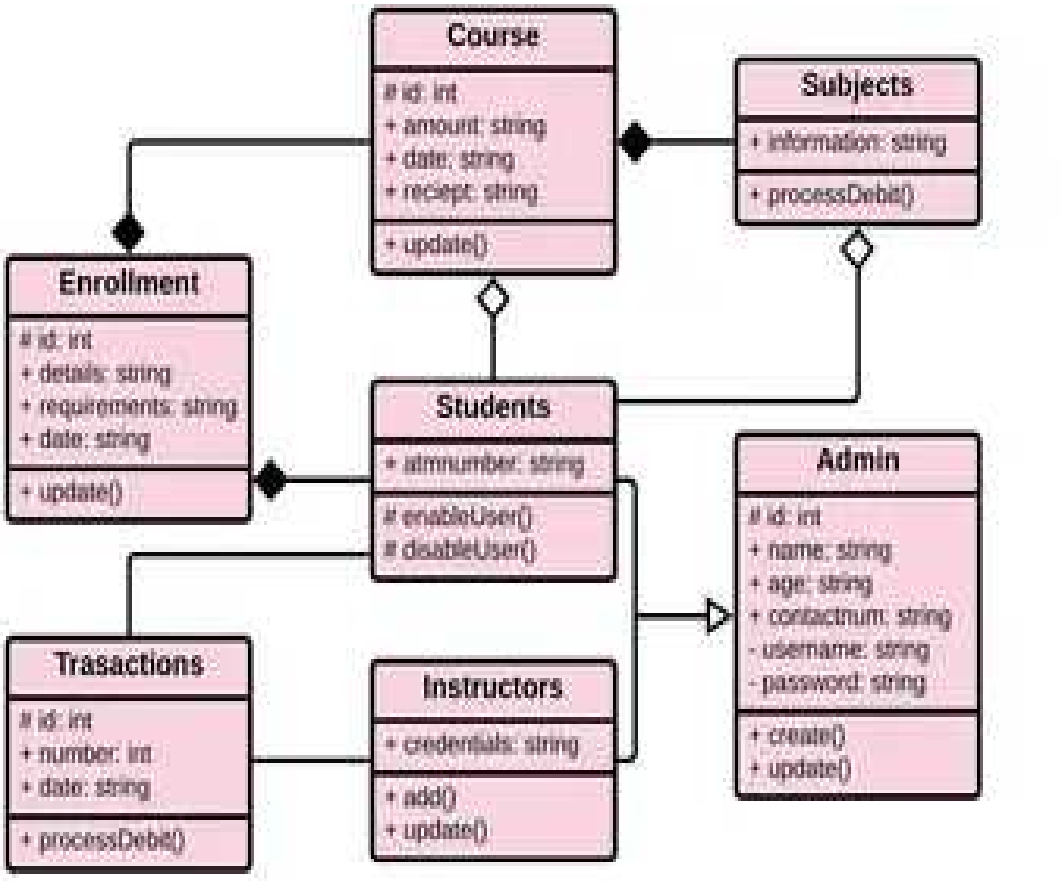
- **Cross-Platform Compatibility:**

The application will be developed to run seamlessly on major operating systems (Windows, macOS, Linux) without significant variations in performance or functionality**.**

5 Design Description

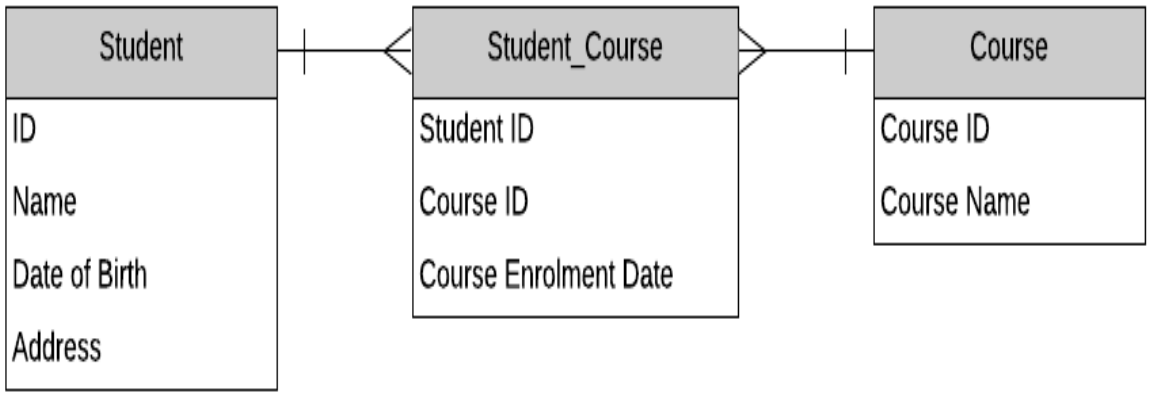
5.1 Logical Viewpoint

Class diagram of the software is given below.



5.2 Information Viewpoint

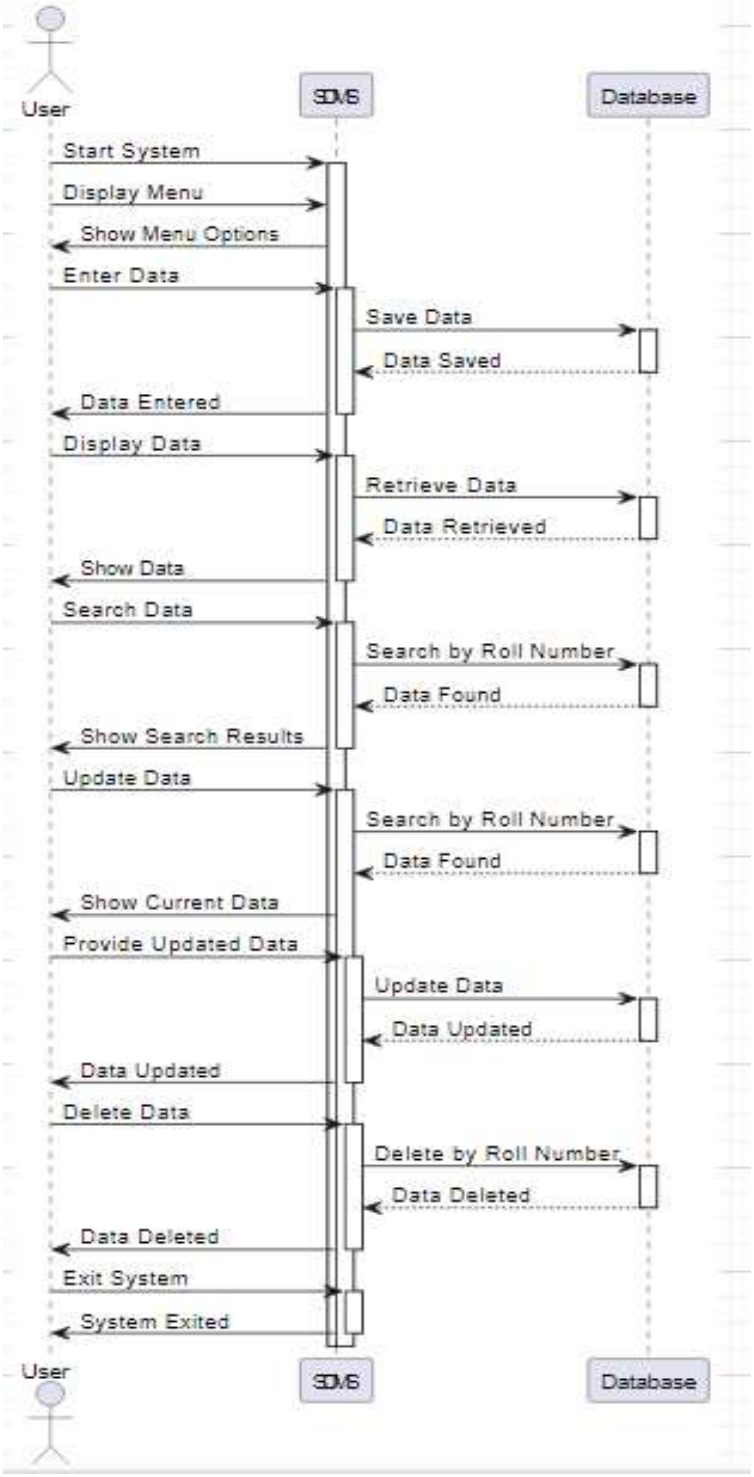
The entity relationship diagram of SDMS is shown below,



- Student is the entity representing individual students.
- Attributes like **Roll No**, **Name**, **Course**, **Class**, and **Contact** are associated with each student.
- The relationship between **Student** and **Database** shows that multiple students' data is stored in the **Database**.
- This simple ERD indicates that each student is associated with the database where their information is stored. Attributes like **Roll No**, **Name**, etc., exist within the **Student** entity.

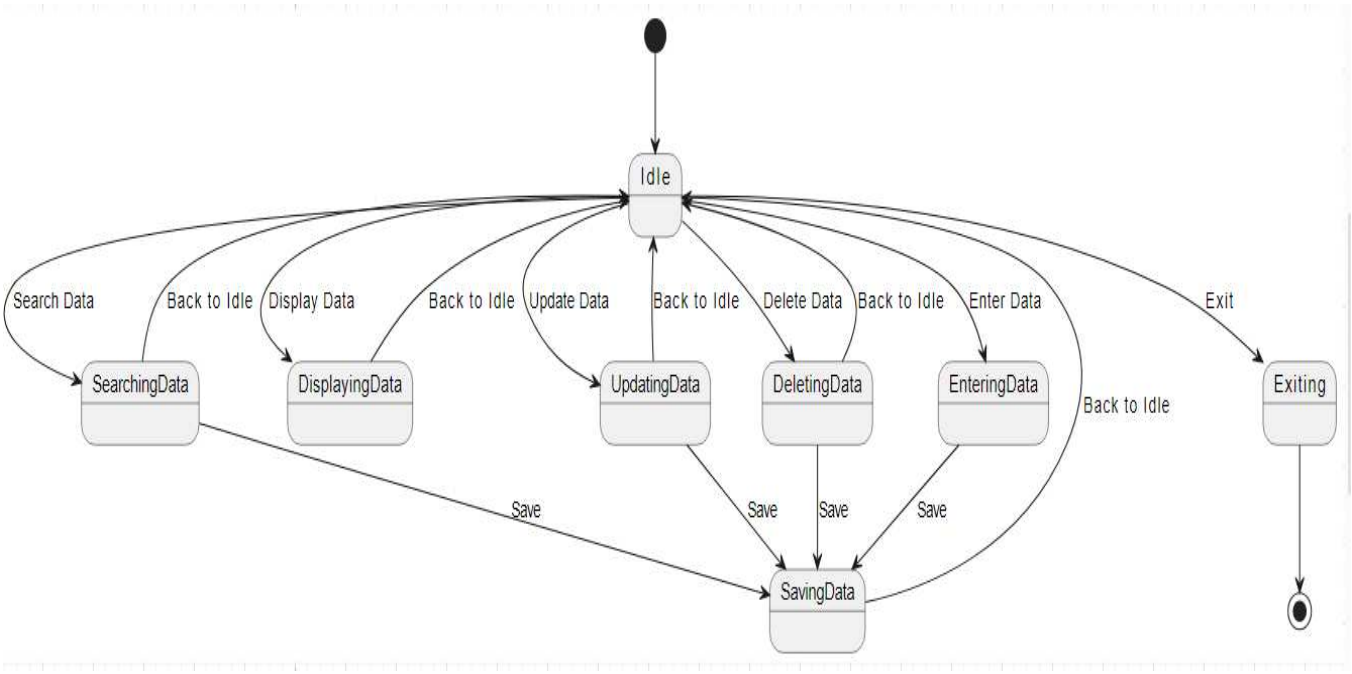
5.3 Interaction Viewpoint:

The sequence diagram is given below:



5.4 State Dynamic Viewpoint

UML State Machine Diagram of SDMS is given below



- The initial state (**[*]**) transitions to the **Idle** state.
- From the **Idle** state, the system can transition to any of the following states based on the user's action: **EnteringData**, **DisplayingData**, **SearchingData**, **UpdatingData**, **DeletingData**, or **Exiting**.
- Each action state (**EnteringData**, **SearchingData**, **UpdatingData**, **DeletingData**) transitions to the **SavingData** state to save the changes made.
- The **SavingData** state transitions back to the **Idle** state

5.5 Algorithm Viewpoint

Decision Table representations offer an understanding of the flow and logic behind each operation within the Student Management System.

Decision Table:

Here different operations are given and specific condition is maintained against which different actions will be performed accordingly as follows,

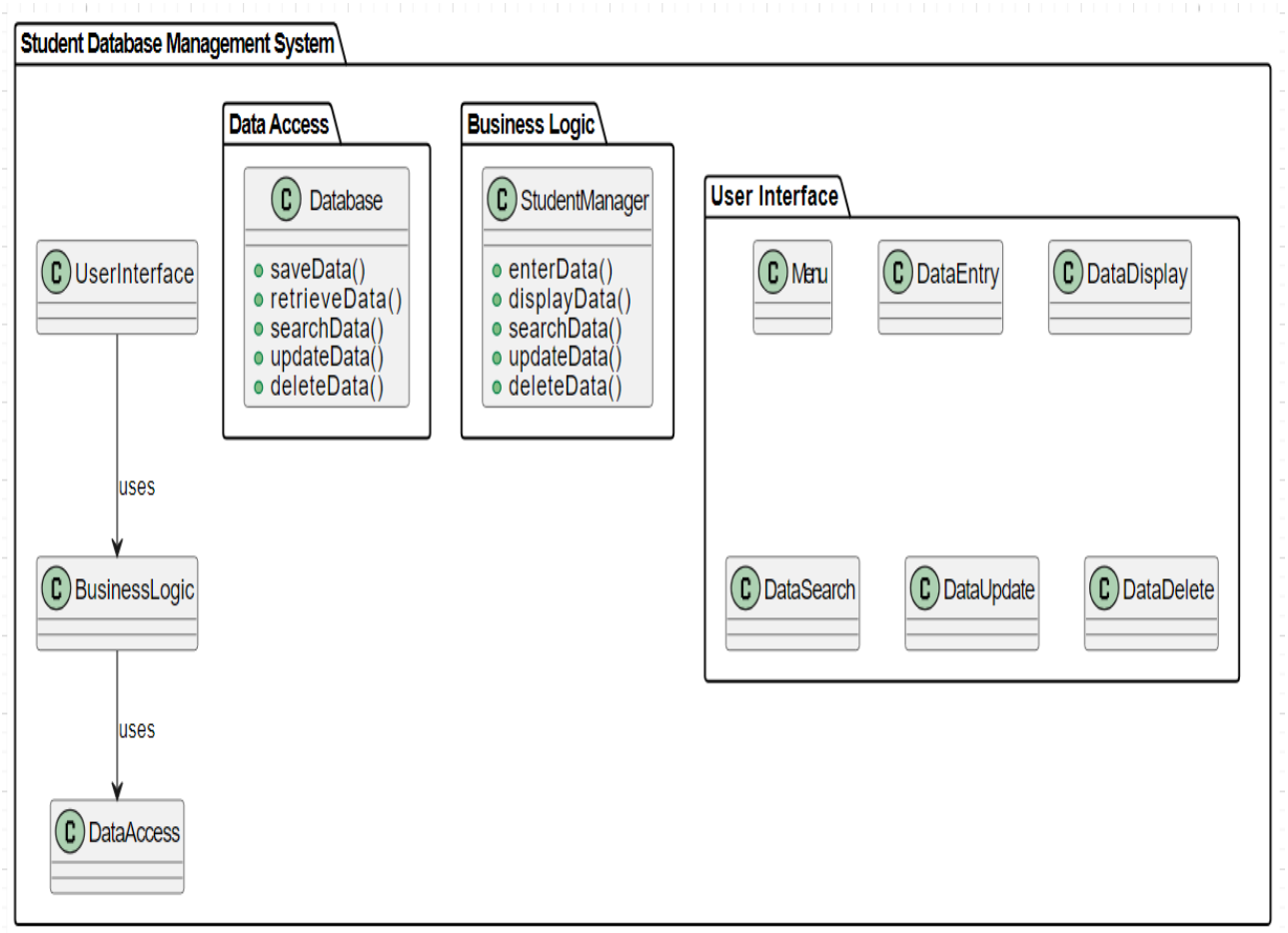
Operation	Inputs	Conditions	Actions
filesave	None	total != 0	Open file, Write data to file, Close file
enter	Number of students (ch)	total == 0	Set total, Input student data into arrays
		Total != 0	Input student data into arrays, Update total
show	None	total == 0	Output "No data is entered"
		Total != 0	Open file, Read data, Output student data, Close file
search	Roll no of student (rollno)	total == 0	Output "No data is entered"
		Total != 0	Input rollno, Check rollno in arr2, Output student data
update	Roll no of student (rollno)	total == 0	Output "No Data is Entered"
		Total != 0	Input rollno, Check rollno in arr2, Update student data
deleterecord	Choice (1 or 2)	total == 0	Output "No Data is Entered"

Operation	Inputs	Conditions	Actions
		Total != 0	Input choice, Delete all or specific record, Update tota

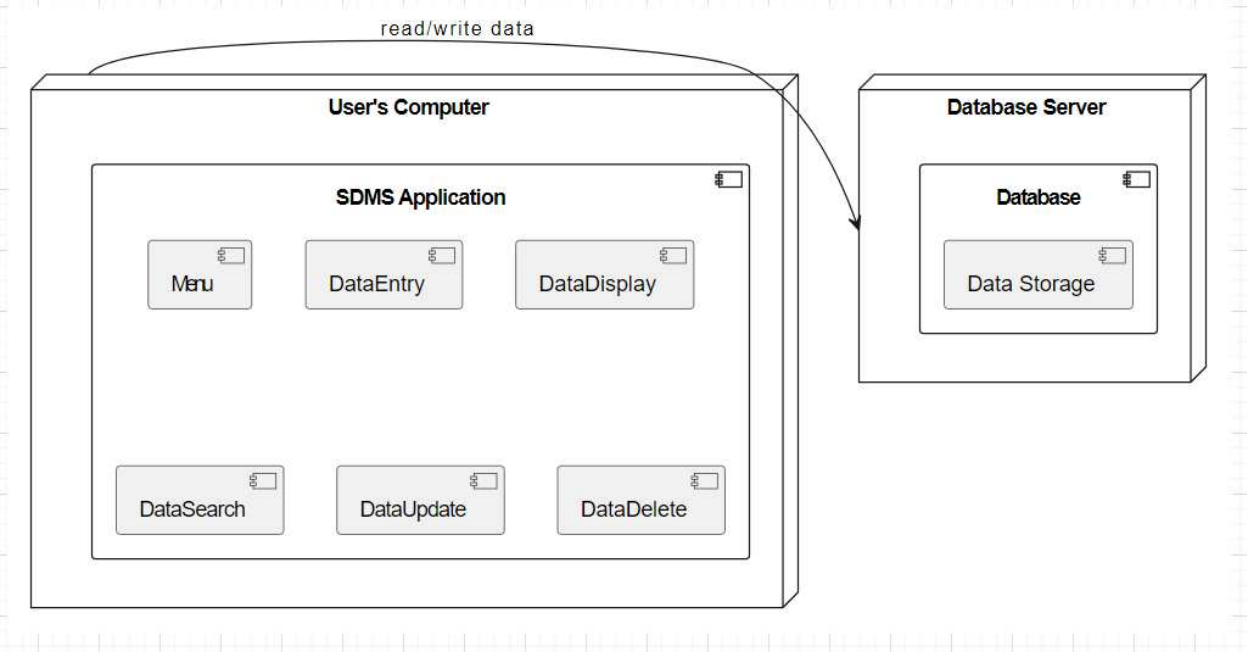
5.6 Composite Viewpoint

Package Diagram or Deployment Diagram of the complete system is given below

Package Diagram:



Deployment Diagram:



Explanation of Deployment Diagram

1. **User's Computer**: This node contains the SDMS application components, which include Menu, DataEntry, DataDisplay, DataSearch, DataUpdate, and DataDelete.

2. **Database Server**: This node contains the Database component that handles data storage.

3. **Communication**: The User's Computer communicates with the Database Server to read and write student data.