**CSE Department – Faculty of Engineering - MSA**

**Spring 2025**

**GSE122 GSE122i COM265 PROGRAMMING 2**

**Course Project**

**Course Instructor:Dr. Ahmed El Anany**

**Due Date 17/MAY/2025 11:59 PM on E-learning**

**Discussion inside lecture 18/May till 23/May inside lab as per lab slot**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** | **Jana Abdelazim** | **Student ID** | **Fill ID here** |
| **Student Name** | **Sarah Reda** | **Student ID** | **244469** |
| **Student Name** | **Salma Amr** | **Student ID** | **Fill ID here** |
| **Student Name** | **Habiba Wahid** | **Student ID** | **Fill ID here** |
| **Student Name** | **Fatma Alaa** | **Student ID** | **Fill ID here** |
| **TA Name** | **Eng. Dina Magdy**  **Eng. Gehad Ehab**  **Eng. Mohamed Khaled**  **Eng. Hussien Mostafa** | **Grade: /** | |

**Student Management Systems**

**Table of Contents**

[**Project Overview 3**](#_1qiv9zobnqqy)

[Objectives 3](#_kxya5mh10gqo)

[Roles and Responsibilities 4](#_4hrs94c0szki)

[Algorithm and external libraries 5](#_lbxclr1u8qm6)

[GUI and Database Usage 6](#_ns54jo9tktz)

[**Code explaining 7**](#_sk6m5j3846ai)

[**Output and results 8**](#_m9h15o464mtl)

[**GitHub(optional) 9**](#_u0qg68ra51oo)

[**References 10**](#_1lsyo46b7om8)

# **Project Overview:**

The **Student Management System** is a Java-based desktop application that helps manage student records, including their ID, name, subject marks, and grades. It uses a simple GUI built with Java Swing and stores data in a MySQL database. The system allows adding, searching, modifying, and deleting student records while ensuring data is saved for future use.

## **Objectives****:**

The main objectives of the Student Management System project are:

1. **Record Management**: To create, update, and manage student data including name, ID, subject-wise marks, and grades.
2. **Grade Calculation**: Automatically compute the student’s grade based on their average marks.
3. **Data Storage and Retrieval**: Store student records in a MySQL database and retrieve them when needed.
4. **User Interaction**: Provide an easy-to-use graphical interface so users can interact with the system without needing to access the database directly.
5. **Modular Design**: Ensure the system is modular by separating concerns into different classes (e.g., data storage, search, UI, etc.) for better maintainability and scalability.
6. **Persistence**: Maintain the consistency of data across sessions by using database operations like insert, update, delete, and select.

## 

## 

## 

## **Roles and Responsibilities**

This section describes the roles of each team member.

## 

## 

## **Algorithm and external libraries:**

**1. Algorithm:**

1. **Main Program Flow**:
   * The program starts with StudentGUI.main().
   * It initializes the main menu using JOptionPane.showInputDialog to display options to the user.
   * The user selects an option, which triggers a corresponding action in the system.
2. **Add a Student**:
   * When the user selects to add a student:
     1. The system prompts for the student’s **ID** using JOptionPane.showInputDialog.
     2. The system then asks for the **Name** of the student.
     3. The system prompts for the **marks** in various subjects.
     4. A new Student object is created with the provided details.
     5. The data is saved in the database using the DatabaseHandler.saveDataBase() method.
3. **Search for a Student**:
   * When the user selects the search option:
     1. The system prompts for the **student ID**.
     2. The system retrieves the student record from the database by calling DatabaseHandler.readDataBase().
     3. If the student is found, a detailed information pop-up is displayed using JOptionPane.showMessageDialog.
4. **Modify a Student**:
   * When the user selects to modify a student:
     1. The system asks for the **student ID**.
     2. If the student is found, the system allows modification of the **name**, **marks**, and **grade**.
     3. The database is updated by calling the DatabaseHandler.updateDatabase() method.
5. **Generate Mark Sheet**:
   * When the user selects to generate a mark sheet:
     1. The system asks for the **student ID**.
     2. If the student exists, the marks for each subject are displayed.
     3. The **final grade** is calculated based on the marks.
     4. A report is generated and shown using JOptionPane.showMessageDialog.
6. **Delete a Student**:
   * When the user selects the option to delete a student:
     1. The system prompts for the **student ID**.
     2. If the student is found, the record is deleted from the database using DatabaseHandler.deleteRecord().
7. **Exit the System**:
   * When the user selects to exit, the program terminates.

### **2.External Libraries:**

1. **javax.swing.\***:
   * **Purpose**: Provides GUI components like JOptionPane for simple input/output dialogs and other Swing-based components for building the user interface.
   * **Used In**:
     + StudentGUI for graphical user interface components such as JOptionPane.showInputDialog for collecting user input and JOptionPane.showMessageDialog for displaying information.
2. **java.sql.\***:
   * **Purpose**: Provides Java Database Connectivity (JDBC) to interact with a MySQL database.
   * **Used In**:
     + DatabaseHandler for database-related operations such as establishing a connection with MySQL, executing SQL commands to insert, update, delete, and query the database.
3. **java.util.\***:
   * **Purpose**: Contains various utility classes such as List, Map, ArrayList, HashMap, etc., for storing and manipulating data.
   * **Used In**:
     + Student class for storing student data in an ArrayList or HashMap (e.g., marks could be stored as a HashMap<String, Integer> for better organization of subject names and marks).
     + SearchEngine and StudentManagementSystem for managing and organizing student data.
4. **java.util.stream.\***:
   * **Purpose**: Provides the Stream API to handle collections of data in a functional programming style (e.g., filtering, mapping).
   * **Used In**:
     + SearchEngine for processing collections of students using stream operations such as filter, map, etc.
5. **java.awt.\*** :
   * **Purpose**: Contains low-level GUI components, typically for basic user interface components like buttons, labels, etc.
   * **Used In**:
     + If you are using any AWT components, they might be used in custom components or as part of the GUI alongside Swing components.
6. **java.io.\***:
   * **Purpose**: Provides classes for input and output, including file handling (reading/writing to files).
   * **Used In**:
     + If the system includes functionality for saving data to a file or reading data from a file, you would use classes from java.io, such as File, BufferedReader, BufferedWriter, FileWriter, etc.
     + For example, if you're storing logs, configuration files, or saving backups of student records outside of the database, you would use the java.io package.
7. **java.util.ArrayList**:
   * **Purpose**: Provides a resizable array that allows dynamic storage of elements (such as student records or marks).
   * **Used In**:
     + Likely used in StudentManagementSystem or similar classes to store lists of students (ArrayList<Student>) and dynamically manage data.
     + Also, can be used for storing subjects and grades in dynamic lists when handling marks.

**Complete List of Libraries:**

java

CopyEdit

import javax.swing.\*; // Swing components for GUI (e.g., JOptionPane for dialogs)

import java.sql.\*; // For database operations (e.g., Connection, Statement for MySQL)

import java.util.\*; // For utility classes like List, ArrayList, HashMap

import java.util.stream.\*; // Stream API for functional-style operations (e.g., filtering collections)

import java.awt.\*; // AWT components if used (e.g., Button, Label)

import java.io.\*; // For I/O operations (e.g., File, BufferedReader, FileWriter)

**Key Points for Libraries Usage:**

1. **javax.swing.\***:
   * Essential for the graphical user interface (GUI) components. This includes input/output dialogs (JOptionPane) and other Swing components.
2. **java.sql.\***:
   * JDBC for connecting to and interacting with a MySQL database. You’ll use Connection, Statement, ResultSet, etc., for managing student records persistently in the database.
3. **java.util.\***:
   * Provides the necessary utility classes for storing, searching, and managing collections of students, marks, and other data. This includes List, Map, ArrayList, and HashMap.
4. **java.util.stream.\***:
   * The Stream API allows you to process data in a functional manner. It's used for filtering or manipulating collections of student data, especially in search operations.
5. **java.awt.\***:
   * This library is typically for basic GUI components like buttons and labels. While Swing usually handles more complex components, AWT might still be needed for simpler UI elements.
6. **java.io.\***:
   * Used for file input and output. If your application needs to read from or write to files (such as saving data to text files or logs), this library will be essential.
   * Classes like BufferedReader, BufferedWriter, and FileWriter would be used here to handle reading and writing files.
7. **java.util.ArrayList**:
   * An array list is a resizable array that can be used for managing dynamic lists of students, subjects, or any other objects. It’s part of the java.util package and commonly used for dynamic data management in this project.

## **GUI and Database Usage**

**This section describes the GUI and Database Usage in the project .**

**1. Graphical User Interface (GUI):**

**Technology Used:**

* Java Swing (javax.swing.\*)
* Components used include: JFrame, JPanel, JLabel, JTextField, JButton, JTextArea, and JScrollPane

**Purpose:**  
The GUI allows users to interact with the student management system in an easy and visual way, without needing to use the command line.

**Main Features of the GUI:**

* Add a new student with their details and marks
* Modify existing student information
* View all student records
* Search for a student by ID
* Delete a student record
* Generate and display a student's mark sheet

**Layout Overview:**

* The main window is a fixed-size JFrame with three parts:
  1. **Input Panel** (top): Text fields for student ID, name, age, course, and marks
  2. **Action Panel** (middle): Buttons for Add, Modify, View, Search, Delete, and Mark Sheet
  3. **Output Area** (bottom): A non-editable text area that displays results and messages

**User Flow:**  
Users enter data into text fields and press buttons to perform actions. Responses and results (like search results or a mark sheet) are shown in the output area.

**2. Database Usage:**

**Technology Used:**

* Java File I/O
* Object Serialization using ObjectOutputStream and ObjectInputStream
* Data File: students.dat

**Purpose:**  
To permanently store student information so that it is not lost when the program is closed.

**How It Works:**

* All student records are stored in an ArrayList<Student>
* This list is saved to a file called students.dat
* When the application starts, the list is loaded back into memory from the file
* Any changes made (add, update, delete) are saved immediately

**Data Storage Format:**

* Each Student object is stored as a serialized Java object in the file
* This avoids the need for a traditional database or SQL

**Supported Operations:**

* Insert new student
* Update existing student
* Delete student by ID
* Search by ID
* Retrieve all records

**Advantages:**

* Simple and fast for small-scale applications
* No need for database setup or external libraries

# 

# **Code Explaining:**

**1. Student**

The Student class is the core of the system and models the student entity. It stores:

* Student ID
* Name
* Age
* Course enrolled
* Marks in multiple subjects

Key features:

* Implements Serializable so objects can be saved and loaded from a file.
* Has a method generateMarkSheet() to calculate the **total** and **average** marks of the student.
* toString() method formats the student information for display.

**2. MarkSheet**

The MarkSheet class is used to generate and display a student’s academic report. It takes a Student object as input and prints the student's basic information along with their marks, total, average, and grade. The grade is calculated based on the average of marks. This class is mainly used for presenting a student’s result in a proper formatted structure either in the console or in the GUI.

**Responsibilities:**

* Access the student’s data like ID, name, course, and marks.
* Calculate the total and average from the marks array.
* Determine the grade based on average.
* Print a formatted mark sheet with all relevant details.

This class is not responsible for storing or modifying data — it is only focused on presenting the result for viewing purposes.

**3. SearchEngine**

This utility class allows searching students from a list using:

* **Student ID**
* **Student Name**

Both methods return matching results, and the name-based search is case-insensitive. This class helps quickly find students from an in-memory list.

**4. DatabaseHandler**

Handles all the **data storage operations**. It uses a file named students.dat to save and retrieve student data through **object serialization**.

Main operations:

* insertStudent() – adds a new student if ID doesn't already exist.
* findStudentById() – searches for a student by ID.
* updateStudent() – updates existing student data.
* deleteStudent() – removes a student by ID.
* getAllStudents() – returns the complete list of saved students.
* Internally uses ObjectOutputStream and ObjectInputStream to write and read data from the file.

**5. StudentManagementSystem (Console Version)**

This class provides a **console-based interface** for managing student records. It uses a loop to show a menu with options:

1. Add Student Record
2. Search Student Record
3. Modify Student Record
4. Generate Mark Sheet
5. Delete Student Record
6. Change Admin Password
7. Display All Students
8. Exit

It takes user input through Scanner and performs actions based on the selected option. It maintains a list of students in memory and uses methods like addStudent(), modifyStudent(), etc., to manage data.

**6. StudentGUI Class (Graphical User Interface)**

This is the **graphical version** of the system built using **Java Swing**. It provides a user-friendly interface with the following components:

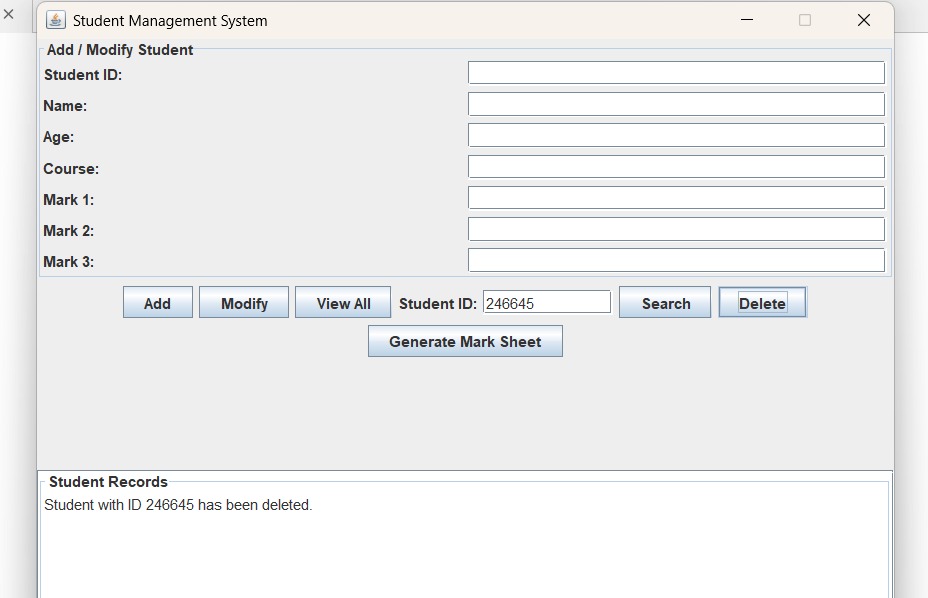
**Interface Layout:**

* **Input Section:**  
  Fields to enter student ID, name, age, course, and marks.
* **Action Buttons:**
  + Add
  + Modify
  + View All
  + Search
  + Delete
  + Generate Mark Sheet
* **Output Area:**  
  A text area that displays student data and system messages.

**Functionality:**

* Reads and writes student data using DatabaseHandler.
* Provides instant feedback to the user via dialogs and text area.
* Uses event listeners on buttons to perform the required action.

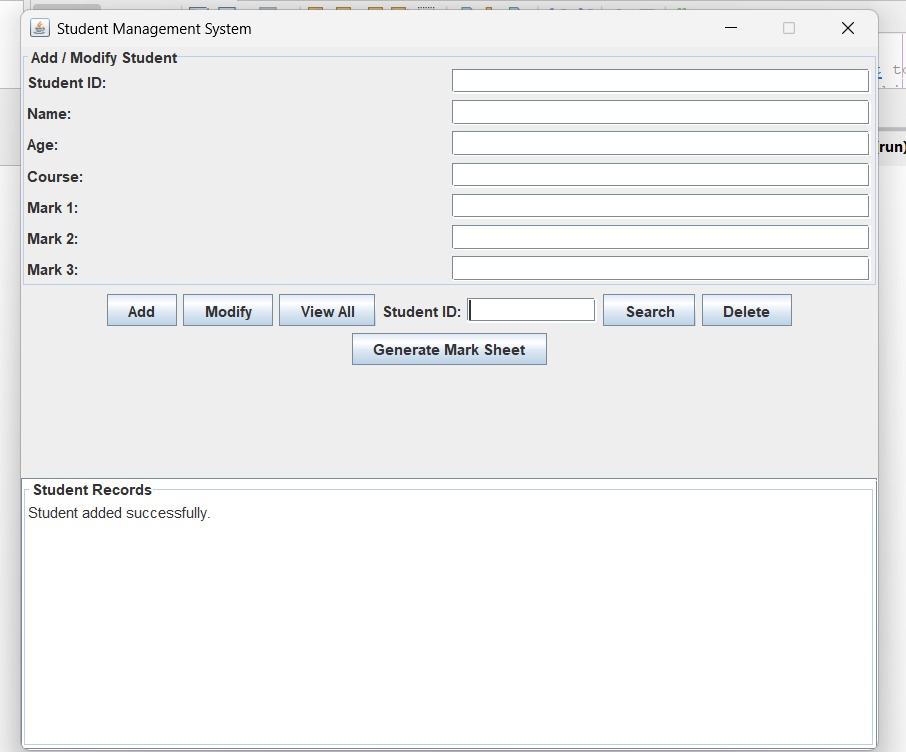
# **Output and results**



# **“Delete” case**

# C:\Users\DELL\Downloads\WhatsApp Image 2025-05-17 at 4.09.45 PM.jpeg

# **“Modify” case**



# **“Add” case**

# C:\Users\DELL\Downloads\WhatsApp Image 2025-05-17 at 4.11.22 PM.jpeg

# **“View All” case**

# C:\Users\DELL\Downloads\WhatsApp Image 2025-05-17 at 4.11.39 PM.jpeg

# **“Search” case**

# C:\Users\DELL\Downloads\WhatsApp Image 2025-05-17 at 4.12.07 PM.jpeg

# **“Generate MarkSheet” case**

# GitHub(optional)

Include link of github repo containing your project code and report and add screenshot of repo with commit logs

<https://github.com/habibawahid1112/Student_Mangement_System>

# References