Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2025), B.Sc. in CSE (Day)

Student Management system

Course Title: Object Oriented Programming Lab
Course Code: CSE 202
Section: 242 D1

Group 4

Students Details

Name	ID
Saifulla Tanim	222002014
Anirban Bosu	242002096
Jumana Akter	242002084

Submission Date: 28.12.2025
Course Teacher's Name: Mr. Ayan Sarkar

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project is designed to develop a complete software system using the knowledge and tools learned during our academic courses. The system aims to solve a real-life problem by using programming, logical design, and user-friendly interfaces. The project includes all major development phases such as planning, designing, coding, testing, and evaluation. By completing this project, we will gain practical experience in building a functional application that can be used in real situations.

1.2 Motivation

In today's world, technology plays an important role in making daily tasks faster and easier. Many manual systems are time-consuming and often contain errors. This motivated us to develop an automated system that can help users complete their tasks more efficiently. As students of Green University of Bangladesh, it is important for us to apply theoretical knowledge to real software development. This project gives us the opportunity to improve our technical skills and contribute to solving a practical problem.

1.3 Problem Definition

1.3.1 Problem Statement

Many existing systems are still manual, slow, unorganized, or not user-friendly. Users face difficulties such as missing information, long processing time, and dependency on paperwork. To solve these issues, a digital platform is needed that can store data properly, process tasks quickly, and provide easy access to information.

1.3.2 Complex Engineering Problem

The project deals with a complex engineering problem because it requires handling multiple tasks such as data storage, data validation, user authentication, error handling, and secure access. The system must also manage different types of users, ensure accuracy, and deliver reliable performance. Designing such a system requires knowledge of algorithms, programming, data structure, and system architecture.

1.4 Design Goals / Objectives

The main objectives of this project are as follows:

- To design a simple, clean, and user-friendly interface for all users.
- To develop a functional system capable of solving a real-life problem effectively.
- To ensure accurate data handling and secure user authentication.
- To implement efficient algorithms for processing and validating user input.
- To maintain clear documentation for future updates.
- To test the system thoroughly to ensure smooth operation with minimal errors.

1.5 Application

The project can be used in different real-life applications depending on the chosen topic. Examples include education systems, management systems, record-keeping systems, or any platform that needs automated data handling. The system can be used by students, teachers, administrators, or general users to complete tasks easily and quickly. With further development, the application can be expanded into a larger system with more advanced features.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This chapter describes the overall design, development process, and implementation details of the Student Management System. The system is developed using Object Oriented Programming (OOP) concepts in Java. It focuses on providing a simple graphical user interface (GUI) for managing student information efficiently. Proper software design principles were followed to ensure modularity, reusability, and maintainability of the system.

2.2 Project Details

The Student Management System is a Java-based desktop application designed to manage student records such as ID, name, department, and academic details. The system reduces manual record keeping and provides a structured way to store and retrieve student information.

2.2.1 System Architecture

The Student Management System follows a simple layered architecture to ensure modularity, maintainability, and ease of understanding. The architecture is designed based on Object Oriented Programming principles and separates the system into different logical layers.

- **Presentation Layer:** This layer is responsible for user interaction. It is implemented using Java Swing components to provide graphical user interfaces such as the login screen and student management dashboard. Users can enter data and view system responses through this layer.
- **Application Logic Layer:** This layer contains the core logic of the system. Classes such as `Login.java` and `MainClass.java` handle authentication, val-

idation, and system flow control. Object Oriented concepts are applied to manage system operations efficiently.

- **Data Layer:** The data layer manages student information using object-based storage. The `Student.java` class is used to create and manage student objects. Data is temporarily stored in memory during runtime without using an external database.

This layered architectural approach improves code reusability, simplifies debugging, and allows future expansion such as database integration or web deployment.

2.3 Implementation

The project is implemented using Java following Object Oriented Programming (OOP) principles. [1, 2]. The system is designed as an administrator-based application, where only authorized school or university administrators can log in and manage student information.

The **Login Page** is used to authenticate administrators before allowing access to the system. It ensures secure access by validating institutional credentials and prevents unauthorized users from entering the application.

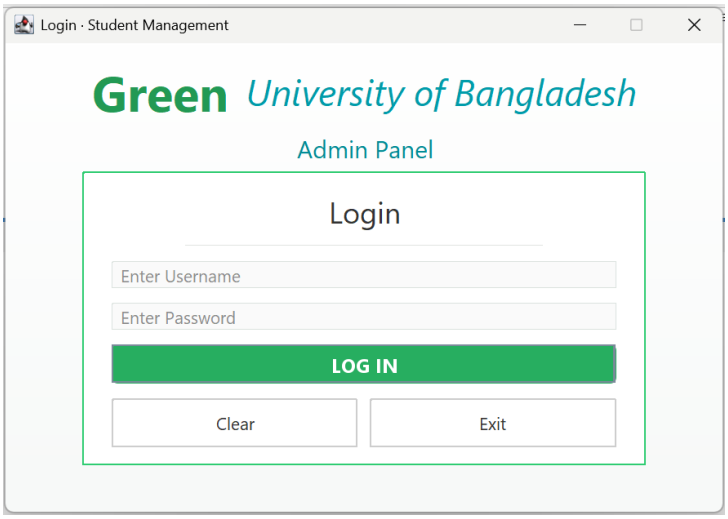

Figure 2.1: Administrator Login Page of the Student Management System

After successful authentication, the administrator is redirected to the **Student Management Page**. This page displays a student table containing student details such as name, phone number, GPA, and gender. It also allows the administrator to add, update, delete, and manage student records efficiently.

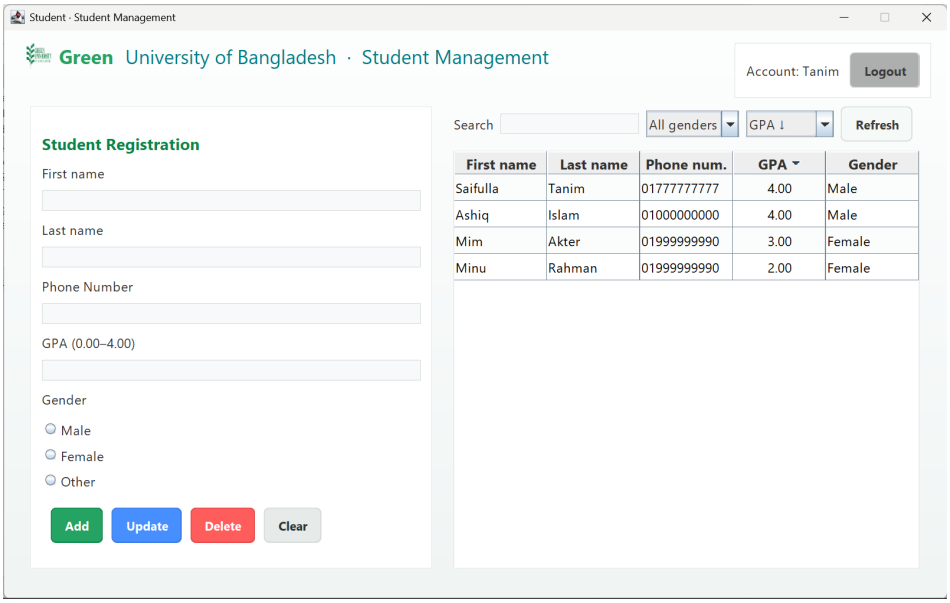

Figure 2.2: Student Management Page Showing Student Table

2.3.1 System Modules

The Student Management System is divided into functional modules to ensure simple and organized system design.

- **Login Module:** Handles user authentication and system access.
- **Student Management Module:** Manages student information such as ID, name, and department.
- **Validation Module:** Ensures correct and valid user input.
- **User Interface Module:** Provides graphical interaction using Java Swing.

This modular approach improves system maintainability and future scalability. [3].

The Workflow

The system starts with a login interface where the user enters valid credentials. After successful authentication, the main dashboard is displayed. The user can then input student information such as ID, name, and department. The system validates the input, stores the data as objects, and displays the information. Invalid inputs generate error messages to ensure correct data entry.

Tools and Libraries

The project is developed using the following tools and libraries:

- Java Development Kit (JDK)

- Java Swing for graphical user interface
- Visual Studio Code / NetBeans IDE
- Windows Operating System

Implementation Details (with Screenshots and Programming Codes)

The system is implemented using Java classes based on Object Oriented Programming principles. The `Login.java` class handles authentication, `MainClass.java` controls system flow, and `Student.java` manages student data. Java Swing components are used to design the interfaces. Screenshots of the login and student management windows are included to demonstrate system functionality.

2.4 Algorithms

This chapter describes the algorithms used in the Student Management System. The algorithms are written based on the actual implementation logic of the project using Object Oriented Programming in Java. Pseudo-codes are included to clearly explain the system workflow.

Algorithm 1: Login Authentication Algorithm

Input: Username, Password
Output: Login Success or Error Message
`/* Start login validation process */`

Read username and password from user
if *Username and Password are valid* **then**
    Display "Login Successful" message
    Open main dashboard
else
    Display "Invalid Credentials" message
    Ask user to retry login

Algorithm 2: Student Information Management Algorithm

Input: Student ID, Name, Department
Output: Stored and Displayed Student Information
`/* Start student data handling */`

Read student information from input form
if *All input fields are valid* **then**
    Create Student object
    Store student data in object variables
    Display student information on interface
else
    Show error message for invalid input

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

The system was tested in a controlled environment to verify its correctness, stability, and usability.

3.1.1 Simulation Environment / Simulation Procedure

The application was developed and executed using the Java programming language on a Windows operating system. Java Swing was used for designing the graphical user interface, and the system was tested using an integrated development environment (IDE). Multiple test runs were performed to ensure consistent behavior.

Simulation Environment

The Student Management System was developed and tested in a standard desktop environment. The application was implemented using the Java programming language and executed on a Windows operating system. Java Swing was used to design the graphical user interface. The system was compiled and run using an Integrated Development Environment (IDE). Multiple test runs were performed to observe system behavior and ensure stable performance.

3.2 Results Analysis/Testing

The Student Management System was tested to evaluate its functionality, reliability, and correctness. Various test cases were executed to analyze the performance of the system under different conditions. [4].

The login functionality was tested using valid and invalid administrator credentials. The system successfully allowed access for authorized users and restricted unauthorized login attempts. Student data operations such as adding, updating, deleting, and viewing records were tested to ensure accurate data handling.

The testing results indicate that the system performs efficiently, responds correctly to user inputs, and handles errors properly without system failure. Overall, the system meets the expected functional requirements for a student management application.

3.2.1 Error Handling and Input Validation Results

This result portion focuses on system behavior during invalid or incomplete input scenarios. When incorrect administrator credentials or incomplete student information is provided, the system displays appropriate error messages and prevents incorrect data from being processed. The system remains stable without crashing, demonstrating proper error handling and input validation during testing.

3.3 Results Overall Discussion

The testing results indicate that the Student Management System performs reliably under different conditions. The system successfully authenticates authorized administrators, manages student records accurately, and handles invalid inputs efficiently. Minor limitations such as the absence of database support were identified, but overall system functionality meets the project objectives and performs as expected. [5].

3.3.1 Complex Engineering Problem Discussion

The project addresses a complex engineering problem by integrating user authentication, data validation, object oriented design, and graphical user interface development into a single system. Ensuring secure administrator-only access while managing multiple student records requires logical planning, modular design, and proper implementation of OOP principles. These attributes reflect the complexity of the problem addressed by the project, as discussed in Table **??**.

Chapter 4

Conclusion

4.1 Discussion

This project successfully demonstrates the design and implementation of an administrator-based Student Management System using Java and Object Oriented Programming principles. The system provides secure administrator login, efficient student data management, and a user-friendly graphical interface. Testing results show that the system performs reliably, handles valid and invalid inputs correctly, and meets the expected functional requirements. The project enhanced practical understanding of OOP concepts such as encapsulation, modularity, and data validation, while offering a simple yet effective solution for managing student records in educational institutions.

4.2 Limitations

Despite achieving its core objectives, the project has several limitations. The system does not use a database, and all data is stored temporarily during runtime, which limits data persistence. Only administrator-level access is supported, and no additional user roles are implemented. Furthermore, the graphical user interface is basic and lacks advanced features such as reporting, data export, and analytics. These limitations restrict the scalability of the system for large-scale real-world deployment.

4.3 Scope of Future Work

The project can be extended in several directions to improve functionality and usability. Future work may include integrating a database system for permanent data storage, implementing role-based access control for different users, and enhancing the graphical interface for better user experience. Additional features such as advanced search, filtering, reporting, and web-based deployment can further expand the system and make it suitable for real-world educational environments.

References

[1] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification*. Addison-Wesley, java se 8 edition, 2014.

[2] Cay S. Horstmann. *Core Java Volume I: Fundamentals*. Prentice Hall, 10th edition, 2016.

[3] Kathy Sierra and Bert Bates. *Head First Java*. O'Reilly Media, 2005.

[4] Rahul Singh and Ankit Kumar. Design and implementation of student management system. *International Journal of Computer Applications*, 179(25):12–16, 2018.

[5] Roger S. Pressman. Software engineering: A practitioner's approach. *McGraw-Hill Education*, 2014.