COLLEGE MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

BHARATHWAJ T G (2303811710421018)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **“COLLEGE MANAGEMENT SYSTEM”** is the bonafide work of **BHARATHWAJ T G (2303811710421018)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Mr. M.SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

[illegible]

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. M. Saravanan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.24

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Dr.R.SENTHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.

[illegible]

INTERNAL EXAMINER	EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“COLLEGE MANAGEMENT SYSTEM”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

.

Signature

T.G. Bharathwaj

BHARATHWAJ T G

Place: Samayapuram

Date: 02.12.24

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. M. SARAVANAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The College Management System is a Java-based desktop application developed using the Swing framework to simplify and automate the administrative tasks of a college. It provides an interactive graphical user interface (GUI) that allows users to manage student records, course details, and faculty information efficiently. The system is designed with a modular structure, enabling users to add, update, and view details such as student names, IDs, enrolled courses, course durations, and faculty assignments. The intuitive interface includes panels for data entry and management, incorporating buttons for actions like saving records and viewing details. This project demonstrates essential concepts of object-oriented programming, event-driven programming, and GUI design in Java. Designed as a scalable and user-friendly solution, the application can be extended with features such as database integration, role-based authentication, attendance tracking, and fee management, making it a robust foundation for comprehensive college management systems.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The **College Management System** is a Java-based desktop application developed using the Swing framework to simplify and automate the administrative tasks of a college. It provides an interactive graphical user interface (GUI) that allows users to manage student records, course details, and faculty information efficiently. The system is designed with a modular structure, enabling users to add, update, and view details such as student names, IDs, enrolled courses, course durations, and faculty assignments. The intuitive interface includes panels for data entry and management, incorporating buttons for actions like saving records and viewing details. This project demonstrates essential concepts of object-oriented programming, event-driven programming, and GUI design in Java. Designed as a scalable and user-friendly solution, the application can be extended with features such as database integration, role-based authentication, attendance tracking, and fee management, making it a robust foundation for comprehensive college management systems.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	Viii
1	**INTRODUCTION**	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	**PROJECT METHODOLOGY**	3
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	**MODULE DESCRIPTION**	4
	3.1 Student Management Module	4
	3.2 Course Management Module	4
	3.3 Faculty Management Module	4
	3.4 Fee Management Module	5
	3.5 Examination Management Module	5
4	**CONCLUSION & FUTURE SCOPE**	6
	4.1 Conclusion	6
	4.2 Future Scope	6
	REFERENCES	7
	APPENDIX A (SOURCE CODE)	8
	APPENDIX B (SCREENSHOTS)	13

CHAPTER 1
INTRODUCTION

1.1Objective

The primary objective of the College Management System is to develop a desktop application that streamlines the management of college operations. It aims to replace traditional manual processes with a digital platform to enhance efficiency and reduce errors. The system provides modules for managing student records, course details, and faculty information in an organized manner. It ensures a user-friendly interface for seamless data entry, updating, and retrieval. By leveraging Java Swing, the application offers an interactive GUI to handle administrative tasks efficiently. It focuses on modularity, scalability, and ease of maintenance. The project demonstrates key programming concepts like object-oriented programming and event-driven development. It is designed to minimize redundant tasks and save time for users. Additionally, the system serves as a foundation for future upgrades, including database integration and role-based user authentication. Ultimately, it aims to provide a reliable and efficient solution for college administration.

1.2Overview

The College Management System is a Java-based desktop application designed to automate and simplify college administrative tasks. It provides an intuitive graphical user interface (GUI) using Java Swing for managing student records, course details, and faculty information.The application is modularly designed, dividing its functionalities into three key components: Student Management,Course Management, and Faculty Management. Users can easily add, update, and view details such as student IDs, course durations, and faculty assignments. The system leverages core Java concepts, including object-oriented programming and event-driven development, to ensure robustness and scalability. Swing components like `JFrame`, `JButton`, and `JLabel` are used to create an interactive and visually appealing interface. It is designed to handle large volumes of data efficiently and reduce errors associated with manual processes.The project emphasizes simplicity, reliability, and scalability, making it suitable for small to medium-sized educational institutions. With its modular structure, the system can be enhanced with features like database integration and role-based access. Overall, it serves as a versatile tool for managing college operations effectively.

1.3 Java Programming Concepts

The basic concepts of Object-Oriented Programming (OOP) are:

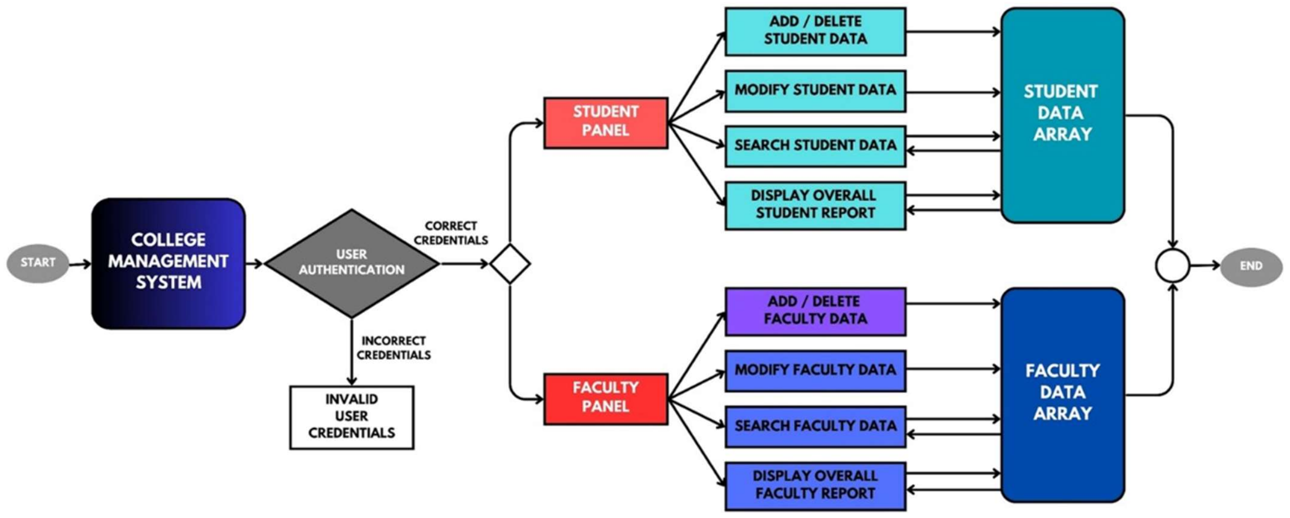
- **Encapsulation**: Bundling data and methods into a single class, while restricting access to certain details to ensure data integrity and security.
- **Abstraction**: Hiding the complex implementation details and showing only the essential features of an object, making the system easier to use and manage.
- **Inheritance**: Allowing a class to inherit properties and methods from another class, which promotes code reusability and a hierarchical structure.
- **Polymorphism**: Enabling one interface to represent multiple forms, including method overloading and method overriding, to allow flexibility and extend functionality.
- **Classes and Objects**: Classes act as blueprints for creating objects, which are instances of those classes, holding specific data and behaviors relevant to the application.
- **Association, Aggregation, and Composition**: Representing relationships between objects, where association indicates a loose connection, aggregation signifies a whole-part relationship, and composition indicates a strong dependency where the lifecycle of objects is tied together.
- **Project-Related Concepts**
- **Scope and Requirements**: Defining the project's objectives, deliverables, and user requirements to ensure the development process is aligned with stakeholder expectations and project goals.
- **Planning and Execution**: Developing a detailed roadmap with specific timelines, resource allocation, and risk management strategies to guide the development process and ensure successful milestones.
- **Testing and Deployment**: Rigorous verification of the system's functionality, reliability, and usability through various testing methods before the final deployment to ensure the product meets quality standards.
- **Quality Assurance (QA)**: Ensuring that the software meets required quality standards and is free from defects by implementing processes to verify that the product is reliable, secure, and user-friendly.
- **Project Closure**: The final stage of the project, which includes reviewing outcomes, documenting lessons learned, ensuring all deliverables are met, and officially closing the project with a thorough evaluation of its success.

CHAPTER 2
PROJECT METHODOLOGY

2.1Proposed Work

The proposed work for the **College Management System** aims to create a Java-based application to automate and streamline college administrative tasks. The system will feature modules for managing student records, course information, and faculty details. It will provide an intuitive graphical user interface (GUI) using the Java Swing framework, allowing users to add, update, and view data efficiently. The system will ensure data integrity by incorporating object-oriented programming principles like encapsulation, inheritance, and polymorphism. It will include basic error handling and validation to maintain smooth user interactions. Additionally, the system will be structured for easy maintenance and future enhancements, such as adding role-based access control and attendance management. The project will focus on delivering a reliable, user-friendly solution to automate college operations. Testing will ensure that the application is free from errors and performs well under various use cases. The final product will serve as a foundation for a comprehensive college management system.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Student Management Module

The **Student Management Module** manages all aspects related to student information. It covers student registration by collecting details such as name, student ID, date of birth, course enrollment, and contact information. The module also handles course enrollments, ensuring students are registered in the correct programs. Additionally, it stores and tracks academic records, including grades and performance evaluations. The attendance management feature monitors student attendance and marks their presence or absence for each class, ensuring a comprehensive record of student engagement. The module can integrate with communication tools to send notifications or messages to students regarding important updates, such as exam schedules, course changes, or attendance warnings.

3.2 Course Management Module

The **Course Management Module** is focused on creating, updating, and managing the courses offered at the college. It involves course creation, modification, and removal of outdated courses. The module handles course scheduling by setting semester start and end dates, timings, and room assignments. It also manages faculty assignments by assigning instructors to specific courses and ensuring the faculty load is balanced. Furthermore, it ensures that course prerequisites are enforced, so students meet all necessary requirements before enrolling in advanced courses. This module can include functionality to collect feedback from students regarding the courses they are enrolled in, allowing instructors and administrators to improve course quality.

3.3 Faculty Management Module

The **Faculty Management Module** is designed to manage faculty details, assignments, schedules, and performance evaluations. It collects and maintains faculty information such as name, faculty ID, department, and qualifications. The module manages the assignment of faculty to courses, ensuring that teaching loads are distributed evenly. It also tracks faculty leave requests and approvals, providing a system for managing absences. Additionally, the module incorporates performance evaluations, gathering feedback from students and administrators to assess the effectiveness of faculty teaching.

3.4 Fee Management Module

The **Fee Management Module** oversees all fee-related operations within the college. It begins with setting up the fee structure for various courses and programs, ensuring transparency and clarity. The module tracks fee payments made by students, keeping records of amounts, due dates, and payment modes. It also generates payment receipts and invoices for students and manages reminders for pending fee payments, notifying students of upcoming deadlines to ensure timely payments. The module can include a feature to manage scholarships, discounts, and financial aid for students, making it easier to apply and track them during the fee payment process.

3.5 Examination and Results Module

The **Examination and Results Module** is responsible for managing the examination process at the college. This includes creating and scheduling exams, assigning dates, times, and locations for each exam. The module also handles question paper generation, ensuring that there is a variety of questions with appropriate complexity for each exam. It records student marks and grades for each exam and manages final course results. Additionally, the module generates detailed reports for result analysis, helping the college analyze student performance and develop strategies for future academic improvements.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the College Management System effectively demonstrates a modular approach to software development. By implementing distinct modules for data management, business logic, user interface, interaction, and exception handling, the system is designed to be scalable, maintainable, and easy to update. The use of modular components ensures that each part of the system can be independently developed and modified, which is essential for building complex applications. This approach provides a smooth user experience while maintaining the integrity and stability of the system.

4.2 FUTURE SCOPE

The future of this College Management System can see multiple enhancements to improve its functionality and scalability. Key areas for future development include:

- **Database Integration**: Implementing a database like MySQL or MongoDB to store student, course, and faculty data, allowing for real-time updates and more efficient data management.
- **Mobile Platform Expansion**: Developing mobile applications for iOS and Android devices to make the system accessible on the go.
- **User Authentication**: Implementing user login and role management for better security and user management.
- **Advanced Reporting**: Adding more sophisticated reporting features for academic performance, course registration trends, and faculty evaluations.
- **Automated Notifications**: Integrating email or SMS notifications for important events like course registrations, exam schedules, and fee reminders.

These future enhancements would make the system more robust, user-friendly, and adaptable to changing educational environments.

REFERENCES

Java Books:

1. **"Head First Java" by Kathy Sierra and Bert Bates (O'Reilly Media):** A beginner-friendly introduction to Java programming, covering fundamental concepts.
2. **"Effective Java" by Joshua Bloch (Addison-Wesley):** A more advanced book that offers practical advice on writing robust, maintainable, and efficient Java code.
3. **"Java: The Complete Reference" by Herbert Schildt (McGraw-Hill Education):** A comprehensive guide covering the Java language from basics to advanced topics.

Websites:

1. **Oracle Java Documentation:** The official documentation for the Java language, providing details about core libraries, Java APIs, and JVM-related information.
2. **W3Schools Java Tutorial:** A beginner-friendly site that covers all fundamental Java topics with examples and interactive exercises.
3. **GeeksforGeeks Java Programming:** A comprehensive collection of articles and tutorials on Java programming and algorithms.

YouTube Links:

1. **Java Programming - Programming with Mosh:** A popular YouTube tutorial series for learning Java from the basics.
2. **Java Tutorial for Beginners - FreeCodeCamp:** A detailed video tutorial that covers Java basics, object-oriented programming, and more.
3. **Java Brains:** A YouTube channel offering high-quality tutorials on Java, web development, and frameworks like Spring.

APPENDIX A

(SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class CollegeManagementSystem {
   private static ArrayList<String[]> studentDataArray = new ArrayList<>();
   private static ArrayList<String[]> facultyDataArray = new ArrayList<>();

   public static void main(String[] args) {
      JFrame loginFrame = new JFrame("Login");
      loginFrame.setSize(400, 200);
      loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      loginFrame.setLayout(new GridLayout(4, 2, 10, 10));

      JLabel usernameLabel = new JLabel("Username:");
      JTextField usernameField = new JTextField();
      JLabel passwordLabel = new JLabel("Password:");
      JPasswordField passwordField = new JPasswordField();
      JButton loginButton = new JButton("Login");
      JLabel messageLabel = new JLabel("", JLabel.CENTER);

      loginButton.addActionListener(e -> {
         String username = usernameField.getText();
         String password = new String(passwordField.getPassword());

         if (username.equals("student") && password.equals("123")) {
            loginFrame.dispose();
            openStudentPanel();
         } else if (username.equals("faculty") && password.equals("123")) {
            loginFrame.dispose();
            openFacultyPanel();
         } else {
            messageLabel.setText("Invalid Credentials. Try Again!");
         }
      });

      loginFrame.add(usernameLabel);
      loginFrame.add(usernameField);
      loginFrame.add(passwordLabel);
      loginFrame.add(passwordField);
```

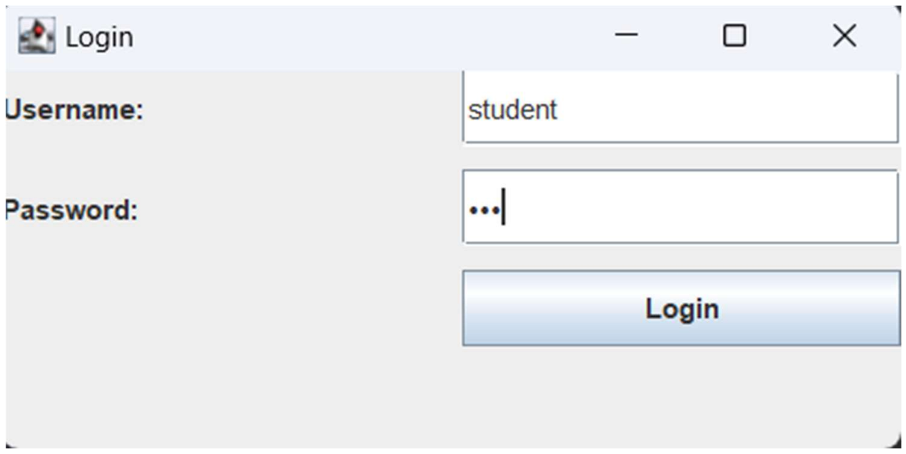
```
        loginFrame.add(new JLabel());
        loginFrame.add(loginButton);
        loginFrame.add(messageLabel);

        loginFrame.setVisible(true);
    }

    private static void openStudentPanel() {
        JFrame studentFrame = new JFrame("Student Panel");
        studentFrame.setSize(600, 400);
        studentFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        studentFrame.setLayout(new GridLayout(5, 1, 10, 10));

        JButton addDeleteButton = new JButton("Add/Delete Student Data");
        JButton modifyButton = new JButton("Modify Student Data");
        JButton searchButton = new JButton("Search Student Data");
        JButton reportButton = new JButton("Display Overall Student Report");
        JButton exitButton = new JButton("Exit");

        addDeleteButton.addActionListener(e -> {
            JFrame addDeleteFrame = new JFrame("Add/Delete Student");
            addDeleteFrame.setSize(400, 300);
            addDeleteFrame.setLayout(new GridLayout(5, 2, 10, 10));

            JLabel nameLabel = new JLabel("Name:");
            JTextField nameField = new JTextField();
            JLabel idLabel = new JLabel("ID:");
            JTextField idField = new JTextField();
            JButton addButton = new JButton("Add");
            JButton deleteButton = new JButton("Delete");

            addButton.addActionListener(evt -> {
                studentDataArray.add(new String[]{nameField.getText(), idField.getText()});
                JOptionPane.showMessageDialog(addDeleteFrame, "Student Added
Successfully!");
            });

            deleteButton.addActionListener(evt -> {
                studentDataArray.removeIf(data -> data[1].equals(idField.getText()));
                JOptionPane.showMessageDialog(addDeleteFrame, "Student Deleted
Successfully!");
            });

            addDeleteFrame.add(nameLabel);
            addDeleteFrame.add(nameField);
            addDeleteFrame.add(idLabel);
            addDeleteFrame.add(idField);
            addDeleteFrame.add(new JLabel());
            addDeleteFrame.add(addButton);
            addDeleteFrame.add(new JLabel());
            addDeleteFrame.add(deleteButton);

            addDeleteFrame.setVisible(true);
        });

        modifyButton.addActionListener(e -> {
            JFrame modifyFrame = new JFrame("Modify Student Data");
            modifyFrame.setSize(400, 300);
            modifyFrame.setLayout(new GridLayout(5, 2, 10, 10));

            JLabel idLabel = new JLabel("Enter ID:");
            JTextField idField = new JTextField();
            JLabel newNameLabel = new JLabel("New Name:");
            JTextField newNameField = new JTextField();
            JButton modifyDataButton = new JButton("Modify");

            modifyDataButton.addActionListener(evt -> {
                boolean found = false;
                for (String[] student : studentDataArray) {
                    if (student[1].equals(idField.getText())) {
                        student[0] = newNameField.getText();
                        found = true;
                        JOptionPane.showMessageDialog(modifyFrame, "Student Data Modified
Successfully!");
                        break;
                    }
                }
                if (!found) {
                    JOptionPane.showMessageDialog(modifyFrame, "Student Not Found!");
                }
            });

            modifyFrame.add(idLabel);
            modifyFrame.add(idField);
            modifyFrame.add(newNameLabel);
            modifyFrame.add(newNameField);
            modifyFrame.add(new JLabel());
            modifyFrame.add(modifyDataButton);
```

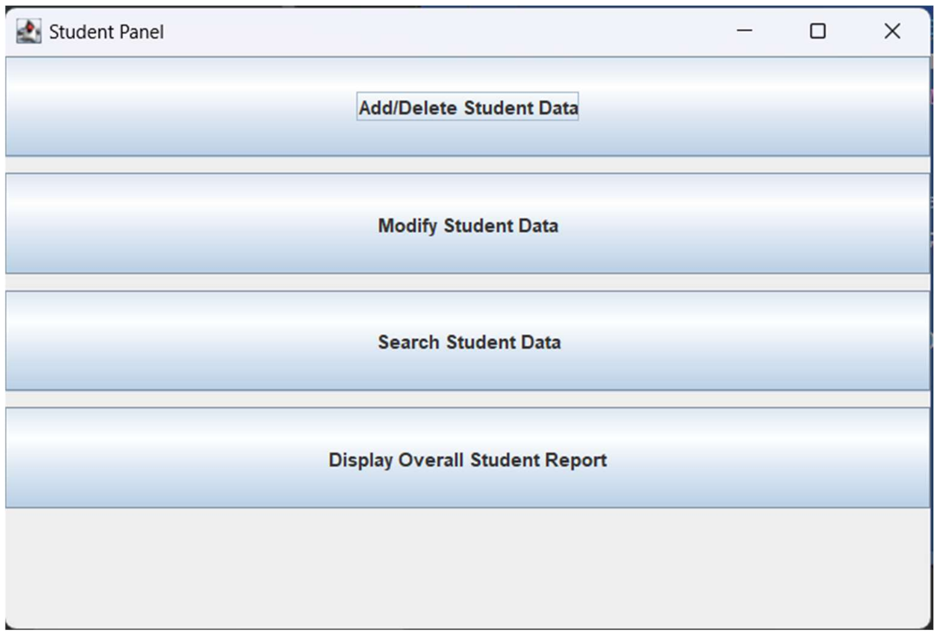
```
    modifyFrame.setVisible(true);
});

searchButton.addActionListener(e -> {
    JFrame searchFrame = new JFrame("Search Student Data");
    searchFrame.setSize(400, 300);
    searchFrame.setLayout(new GridLayout(3, 2, 10, 10));

    JLabel idLabel = new JLabel("Enter ID:");
    JTextField idField = new JTextField();
    JTextArea resultArea = new JTextArea();
    JButton searchDataButton = new JButton("Search");

    searchDataButton.addActionListener(evt -> {
        boolean found = false;
        for (String[] student : studentDataArray) {
            if (student[1].equals(idField.getText())) {
                resultArea.setText("Name: " + student[0] + ", ID: " + student[1]);
                found = true;
                break;
            }
        }
        if (!found) {
            resultArea.setText("Student Not Found!");
        }
    });

    searchFrame.add(idLabel);
    searchFrame.add(idField);
    searchFrame.add(new JLabel());
    searchFrame.add(searchDataButton);
    searchFrame.add(new JLabel());
    searchFrame.add(resultArea);

    searchFrame.setVisible(true);
});

reportButton.addActionListener(e -> {
    JFrame reportFrame = new JFrame("Student Report");
    reportFrame.setSize(400, 300);
    JTextArea reportArea = new JTextArea();
    for (String[] student : studentDataArray) {
        reportArea.append("Name: " + student[0] + ", ID: " + student[1] + "\n");
```

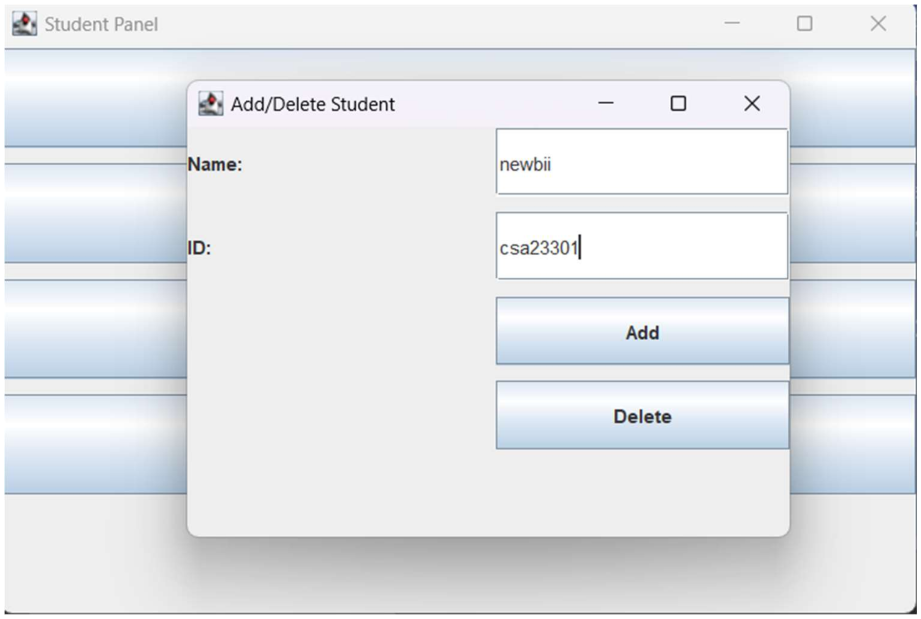
```
            }
            reportFrame.add(new JScrollPane(reportArea));
            reportFrame.setVisible(true);
        });

        exitButton.addActionListener(e -> studentFrame.dispose());

        studentFrame.add(addDeleteButton);
        studentFrame.add(modifyButton);
        studentFrame.add(searchButton);
        studentFrame.add(reportButton);
        studentFrame.add(exitButton);

        studentFrame.setVisible(true);
    }

    private static void openFacultyPanel() {
        JFrame facultyFrame = new JFrame("Faculty Panel");
        facultyFrame.setSize(600, 400);
        facultyFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        facultyFrame.setLayout(new GridLayout(5, 1, 10, 10));

        JButton addDeleteButton = new JButton("Add/Delete Faculty Data");
        JButton modifyButton = new JButton("Modify Faculty Data");
        JButton searchButton = new JButton("Search Faculty Data");
        JButton reportButton = new JButton("Display Overall Faculty Report");
        JButton exitButton = new JButton("Exit");

        // Add actions similar to student panel for faculty

        exitButton.addActionListener(e -> facultyFrame.dispose());

        facultyFrame.add(addDeleteButton);
        facultyFrame.add(modifyButton);
        facultyFrame.add(searchButton);
        facultyFrame.add(reportButton);
        facultyFrame.add(exitButton);

        facultyFrame.setVisible(true);
    }
}
```

APPENDIX B

(SCREENSHOTS)

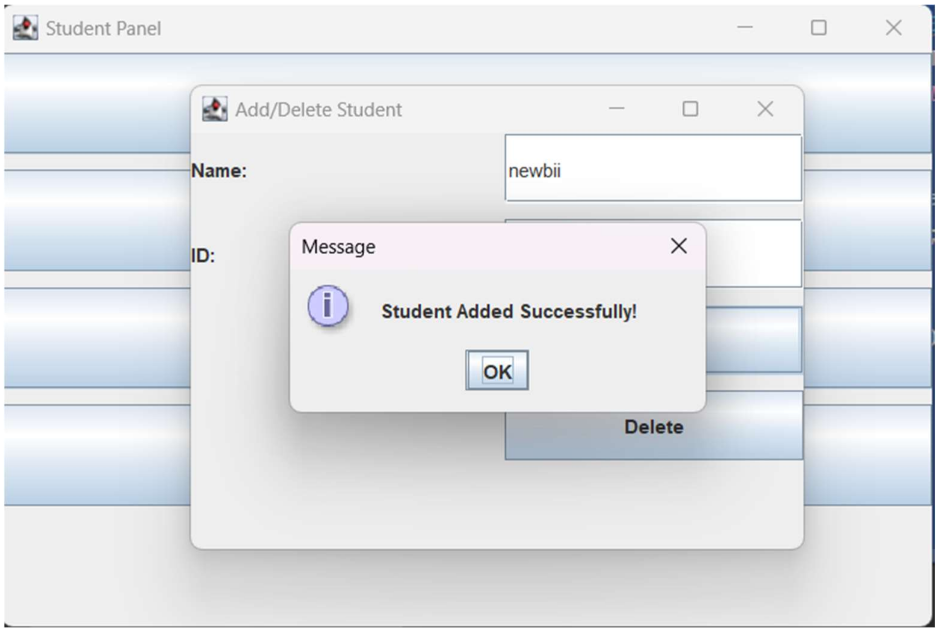






Name: newbii

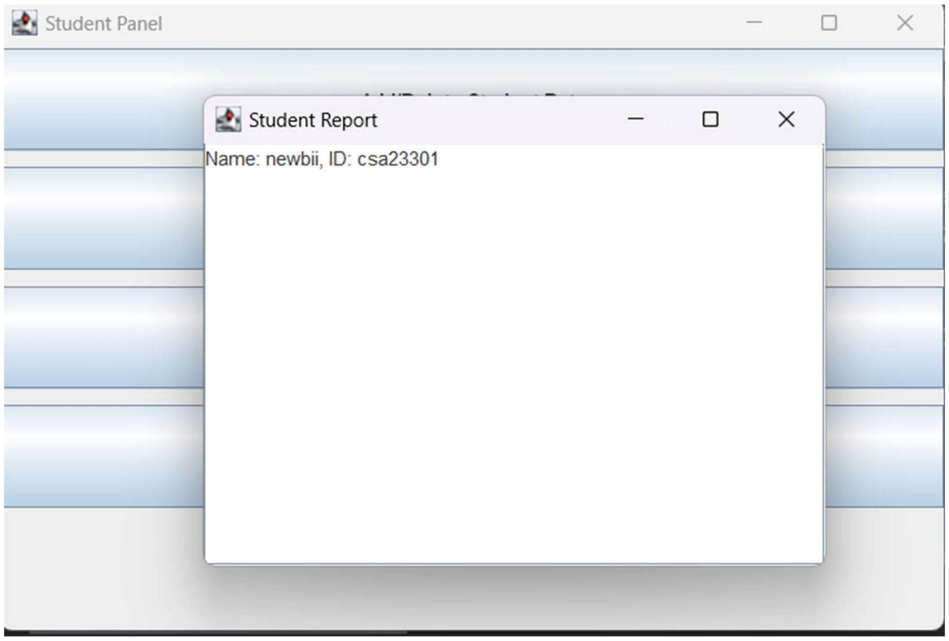
ID: csa23301



Name: newbii

ID:

Student Added Successfully!



```
Name: newbii, ID: csa23301
```