**Project Based Learning (PBL) Report**

**for the course**

**Object Oriented Programming-20CS21002**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

By

**P Varshita – 22R11A05D0**

**Under the guidance of**

**K Prathima**



**Department of Computer Science and Engineering**

**Accredited by NBA**

**Geethanjali College of Engineering and Technology**

**(UGC Autonomous)**

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

**January-2024**

|  |  |  |
| --- | --- | --- |
| S.No. | Contents | Page No |
| 1 | Abstract | 3 |
| 2 | Introduction | 4 |
|  | 2.1 Project Objectives | 4 |
|  | 2.2 Project Outcomes | 4 |
| 3 | System Design | 5 |
|  | 3.1 System Architecture | 5 |
|  | 3.2 Modules and Components | 6 |
|  | 3.3 Flow Chart | 7 |
| 4 | Implementation | 8 |
|  | 4.1 Code | 8 |
|  | 4.2 Result/Output Screens | 24 |
| 5 | Conclusion | 29 |
| 6 | Bibliography | 30 |

1. **ABSTRACT**

The College Management System (CMS) is a user-friendly software application

developed to enhance the efficiency and effectiveness of managing diverse tasks within a

college environment. This Java-based system provides a seamless platform for

administrators, faculty members, and students to interact, access information, and perform

various functions effortlessly. The CMS is designed using Java programming language, leveraging its object-oriented features and platform independence. Java frameworks and libraries are employed for creating a responsive and intuitive user interface, enhancing the overall user experience.

1. **INTRODUCTION**

A College Management System (CMS) is a comprehensive software application designed to streamline and automate various administrative and academic processes within a college or higher education institution. This type of system serves as a centralized platform for managing and organizing diverse aspects of college operations, including student information, academic courses, faculty details, examinations, and more.

* 1. **Project Objectives:**

1. **Learn Basic Java Concepts:**
   * Use the project to understand fundamental Java concepts like classes, objects, methods, and control flow.
2. **File Handling or Database Interaction:**
   * Implement data storage and retrieval mechanisms using file handling or a basic database system (e.g., JDBC with SQLite).
3. **Data Structures and Algorithms:**
   * Apply basic data structures (e.g., arrays, lists) and algorithms for tasks like sorting and searching.
4. **GUI Development:**
   * Develop a simple graphical user interface (GUI) using Java Swing.
   1. **Project Outcomes:**
5. **Understanding of CRUD Operations:**
   * Learn how to perform CRUD (Create, Read, Update, Delete) operations on data.
6. **Database Interaction:**
   * Gain experience in interacting with databases or file systems to store and retrieve information.
7. **GUI Development Skills:**
   * Develop skills in creating a user-friendly GUI for data input and display.
8. **Application of Java Concepts:**
   * Apply Java concepts learned in the project, reinforcing knowledge and building programming skills.
9. **SYSTEM DESIGN**

Designing a college management system involves breaking down the system into smaller components and defining their interactions. Here's a basic system design for a college management system in Java:

* 1. **System Architecture**
     1. **User Interface (UI)**
* **Description:** The UI is the front end of the system, providing a user-friendly interface for interaction.
* **Components:**

Dashboards for students, faculty, courses, attendance and library.

Forms for data input and display.

* + 1. **Technologies:**
* **Programming Language:** Java
* **User Interface:** Java Swing
* **Database:** SQLite, MySQL, or any suitable relational database
* **Database Connectivity:** JDBC (Java Database Connectivity)
  + 1. **Flow of Operations:**

**User Authentication:**

* + Users log in using their credentials.
  + The system verifies credentials and grants access based on user roles.

**Main Dashboard:**

* + Different dashboards for students, faculty, courses, library and attendance.
  + Each dashboard displays relevant information and provides navigation to specific modules.

**Data Input and Display:**

* + Forms for adding and updating student, course, and faculty information.
  + Views to display details, attendance, and library.

**Business Logic Execution:**

* + Modules execute business logic based on user interactions.
  + Data is processed, and results are displayed in the UI.
    1. **Design Principles:**
* **Modularity:** Each module should be independent, making the system easier to understand and maintain.
* **Scalability:** The system should be designed to handle growth in data and users.
* **Usability:** Design an intuitive and user-friendly interface for ease of use.
* **Reliability:** Ensure the system is robust and can handle errors gracefully.
  1. **Modules and Components:**
     1. **Student Module:**
* Student Registration
* Course Enrollment
  + 1. **Faculty Module**
* Mark Attendance
* Give Remarks
  + 1. **Course Module:**
* Course Creation
* Assigned Courses
  + 1. **Attendance Module:**
* Mark Attendance
* View Attendance Reports
  + 1. **Library Module:**
* Book Management
* Borrow/Return Books
  1. **Flow Chart**

1. **IMPLEMENTATION**
   1. **Code**

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.List;

public class CollegeManagementSystem extends JFrame {

private JButton studentButton;

private JButton facultyButton;

private JButton courseButton;

private JButton attendanceButton;

private JButton libraryButton;

private List<Student> students;

private List<Faculty> faculties;

private List<Course> courses;

private List<AttendanceRecord> attendanceRecords;

private Library library;

public CollegeManagementSystem() {

students = new ArrayList<>();

faculties = new ArrayList<>();

courses = new ArrayList<>();

attendanceRecords = new ArrayList<>();

library = new Library();

studentButton = new JButton("Student");

facultyButton = new JButton("Faculty");

courseButton = new JButton("Course");

attendanceButton = new JButton("Attendance");

libraryButton = new JButton("Library");

studentButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

showStudentPanel();

}

});

facultyButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

showFacultyPanel();

}

});

courseButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

showCoursePanel();

}

});

attendanceButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

showAttendancePanel();

}

});

libraryButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

showLibraryPanel();

}

});

setLayout(new BoxLayout(getContentPane(), BoxLayout.***Y\_AXIS***));

add(studentButton);

add(facultyButton);

add(courseButton);

add(attendanceButton);

add(libraryButton);

setSize(300, 200);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setLocationRelativeTo(null);

setVisible(true);

}

private void showStudentPanel() {

JFrame studentFrame = new JFrame("Student Panel");

studentFrame.setLayout(new FlowLayout());

JButton enrollCourseButton = new JButton("Enroll in Course");

JButton enterDetailsButton = new JButton("Enter Student Details");

enrollCourseButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

String courseName = JOptionPane.*showInputDialog*("Enter Course Name:");

Course selectedCourse = findCourse(courseName);

if (selectedCourse != null) {

Student currentStudent = getCurrentStudent();

currentStudent.enrollInCourse(selectedCourse);

JOptionPane.*showMessageDialog*(null, "Enrolled in Course: " + courseName);

} else {

JOptionPane.*showMessageDialog*(null, "Course not found.");

}

}

});

enterDetailsButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

String studentName = JOptionPane.*showInputDialog*("Enter Student Name:");

int studentId = Integer.*parseInt*(JOptionPane.*showInputDialog*("Enter Student ID:"));

students.add(new Student(studentName, studentId));

JOptionPane.*showMessageDialog*(null, "Student Details Entered");

}

});

studentFrame.add(enrollCourseButton);

studentFrame.add(enterDetailsButton);

studentFrame.setSize(300, 200);

studentFrame.setLocationRelativeTo(null);

studentFrame.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

studentFrame.setVisible(true);

}

private void showFacultyPanel() {

JFrame facultyFrame = new JFrame("Faculty Panel");

facultyFrame.setLayout(new FlowLayout());

JButton markAttendanceButton = new JButton("Mark Attendance");

JButton giveRemarksButton = new JButton("Give Remarks");

markAttendanceButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

String studentName = JOptionPane.*showInputDialog*("Enter Student Name:");

Student student = findStudent(studentName);

if (student != null) {

facultyMarkAttendance(student);

JOptionPane.*showMessageDialog*(null, "Attendance Marked for: " + studentName);

} else {

JOptionPane.*showMessageDialog*(null, "Student not found.");

}

}

});

giveRemarksButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

String studentName = JOptionPane.*showInputDialog*("Enter Student Name:");

String remarks = JOptionPane.*showInputDialog*("Enter Remarks:");

Student student = findStudent(studentName);

if (student != null) {

facultyGiveRemarks(student, remarks);

JOptionPane.*showMessageDialog*(null, "Remarks Given for: " + studentName);

} else {

JOptionPane.*showMessageDialog*(null, "Student not found.");

}

}

});

facultyFrame.add(markAttendanceButton);

facultyFrame.add(giveRemarksButton);

facultyFrame.setSize(300, 200);

facultyFrame.setLocationRelativeTo(null);

facultyFrame.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

facultyFrame.setVisible(true);

}

private void showCoursePanel() {

JFrame courseFrame = new JFrame("Course Panel");

courseFrame.setLayout(new FlowLayout());

JButton addCourseButton = new JButton("Add Course");

JButton viewAssignedCoursesButton = new JButton("View Assigned Courses");

addCourseButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

String courseName = JOptionPane.*showInputDialog*("Enter Course Name:");

int courseId = Integer.*parseInt*(JOptionPane.*showInputDialog*("Enter Course ID:"));

courses.add(new Course(courseName, courseId));

JOptionPane.*showMessageDialog*(null, "Course Added: " + courseName);

}

});

viewAssignedCoursesButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

StringBuilder assignedCourses = new StringBuilder("Assigned Courses:\n");

Student currentStudent = getCurrentStudent();

if (currentStudent != null) {

Course enrolledCourse = currentStudent.getEnrolledCourse();

assignedCourses.append(enrolledCourse != null ? enrolledCourse.getCourseName() : "None");

} else {

assignedCourses.append("No student selected");

}

JOptionPane.*showMessageDialog*(null, assignedCourses.toString());

}

});

courseFrame.add(addCourseButton);

courseFrame.add(viewAssignedCoursesButton)

courseFrame.setSize(300, 200);

courseFrame.setLocationRelativeTo(null);

courseFrame.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

courseFrame.setVisible(true);

}

private void showAttendancePanel() {

JFrame attendanceFrame = new JFrame("Attendance Panel");

attendanceFrame.setLayout(new FlowLayout());

JButton studentCheckAttendanceButton = new JButton("Check Attendance");

JButton facultyMarkAttendanceButton = new JButton("Mark Attendance");

studentCheckAttendanceButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

Student currentStudent = getCurrentStudent();

if (currentStudent != null) {

String attendanceInfo = studentCheckAttendance(currentStudent);

JOptionPane.*showMessageDialog*(null, "Attendance: " + attendanceInfo);

} else {

JOptionPane.*showMessageDialog*(null, "No student selected");

}

}

});

facultyMarkAttendanceButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

String studentName = JOptionPane.*showInputDialog*("Enter Student Name:");

Student student = findStudent(studentName);

if (student != null) {

facultyMarkAttendance(student);

JOptionPane.*showMessageDialog*(null, "Attendance Marked for: " + studentName);

} else {

JOptionPane.*showMessageDialog*(null, "Student not found.");

}

}

});

attendanceFrame.add(studentCheckAttendanceButton);

attendanceFrame.add(facultyMarkAttendanceButton);

attendanceFrame.setSize(300, 200);

attendanceFrame.setLocationRelativeTo(null);

attendanceFrame.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

attendanceFrame.setVisible(true);

}

private void showLibraryPanel() {

JFrame libraryFrame = new JFrame("Library Panel");

libraryFrame.setLayout(new FlowLayout());

JButton borrowBookButton = new JButton("Borrow Book");

JButton dueDateButton = new JButton("Book Due Date");

borrowBookButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

String bookTitle = JOptionPane.*showInputDialog*("Enter Book Title:");

String author = JOptionPane.*showInputDialog*("Enter Author:");

Book book = new Book(bookTitle, author);

Student currentStudent = getCurrentStudent();

if (currentStudent != null) {

currentStudent.borrowBook(book);

library.issueBook(currentStudent, book);

JOptionPane.*showMessageDialog*(null, "Book Borrowed: " + book.getTitle());

} else {

JOptionPane.*showMessageDialog*(null, "No student selected");

}

}

});

dueDateButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

Student currentStudent = getCurrentStudent();

if (currentStudent != null && currentStudent.getBorrowedBook() != null) {

Book borrowedBook = currentStudent.getBorrowedBook();

JOptionPane.*showMessageDialog*(null, "Due Date: " library.getDueDate(borrowedBook));

} else {

JOptionPane.*showMessageDialog*(null, "No student or book selected");

}

}

});

libraryFrame.add(borrowBookButton);

libraryFrame.add(dueDateButton);

libraryFrame.setSize(300, 200);

libraryFrame.setLocationRelativeTo(null);

libraryFrame.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

libraryFrame.setVisible(true);

}

// Helper methods

private Student getCurrentStudent() {

String studentName = JOptionPane.*showInputDialog*("Enter Student Name:");

return findStudent(studentName);

}

private Course findCourse(String courseName) {

for (Course course : courses) {

if (course.getCourseName().equals(courseName)) {

return course;

}

}

return null;

}

private Student findStudent(String studentName) {

for (Student student : students) {

if (student.getName().equals(studentName)) {

return student;

}

}

return null;

}

private void facultyMarkAttendance(Student student) {

AttendanceRecord record = new AttendanceRecord(student, true,

new java.util.Date());

attendanceRecords.add(record);

}

private void facultyGiveRemarks(Student student, String remarks) {

student.setRemarks(remarks);

}

private String studentCheckAttendance(Student student) {

StringBuilder attendanceInfo = new StringBuilder("Attendance:\n");

for (AttendanceRecord record : attendanceRecords) {

if (record.getStudent().equals(student)) {

attendanceInfo.append(record.isPresent() ? "Present" : "Absent")

.append(" on ")

.append(record.getDate())

.append("\n");

}

}

return attendanceInfo.toString();

}

public static void main(String[] args) {

SwingUtilities.*invokeLater*(new Runnable() {

public void run() {

new CollegeManagementSystem();

}

});

}

}

class Student {

private String name;

private int id;

private Course enrolledCourse;

private Book borrowedBook;

private String remarks;

public Student(String name, int id) {

this.name = name;

this.id = id;

}

public String getName() {

return name;

}

public void enrollInCourse(Course course) {

enrolledCourse = course;

}

public Course getEnrolledCourse() {

return enrolledCourse;

}

public void borrowBook(Book book) {

borrowedBook = book;

}

public Book getBorrowedBook() {

return borrowedBook;

}

public String getRemarks() {

return remarks;

}

public void setRemarks(String remarks) {

this.remarks = remarks;

}

}

class Faculty {}

class Course {

private String courseName;

private int courseId;

public Course(String courseName, int courseId) {

this.courseName = courseName;

this.courseId = courseId;

}

public String getCourseName() {

return courseName;

}

}

class AttendanceRecord {

private Student student;

private boolean present;

private java.util.Date date;

public AttendanceRecord(Student student, boolean present,java.util.Date date) {

this.student = student;

this.present = present;

this.date = date;

}

public Student getStudent() {

return student;

}

public boolean isPresent() {

return present;

}

public java.util.Date getDate() {

return date;

}

}

class Library {

private List<Book> availableBooks;

private List<Book> borrowedBooks;

public Library() {

availableBooks = new ArrayList();

borrowedBooks = new ArrayList<>();

}

public void issueBook(Student student, Book book) {

availableBooks.remove(book);

borrowedBooks.add(book);

}

public String getDueDate(Book book) { return "Due in 14 days"; }

}

class Book {

private String title;

private String author;

public Book(String title, String author) {

this.title = title;

this.author = author;

}

public String getTitle() {

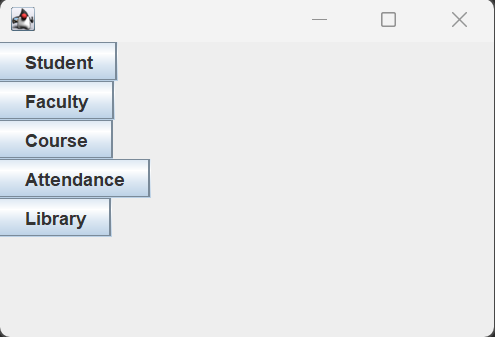
return title;

}

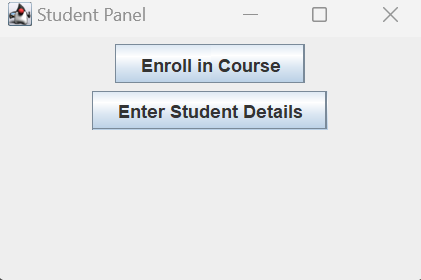
public String getAuthor() {

return author; }}

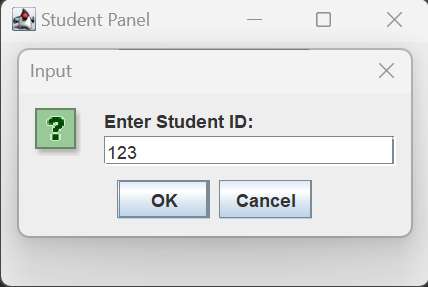
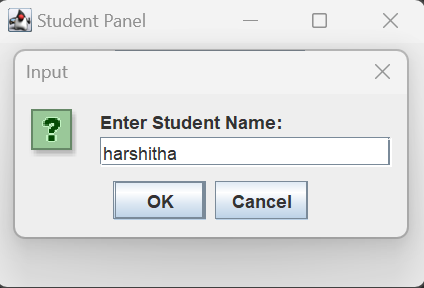
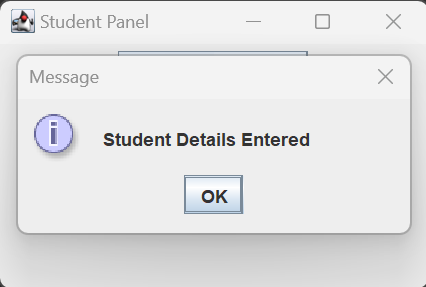
* 1. **Result/Output Screens**

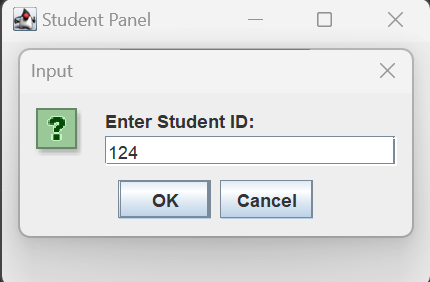
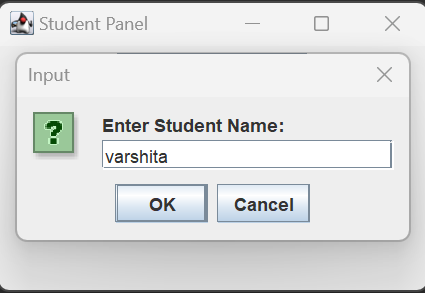
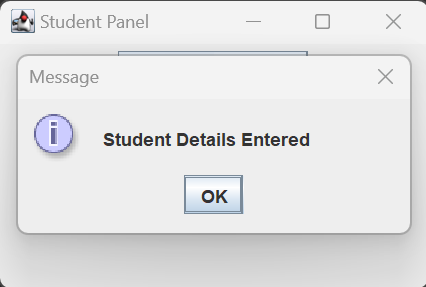


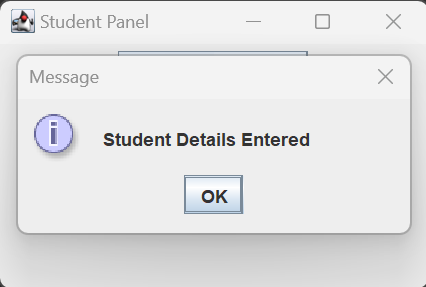
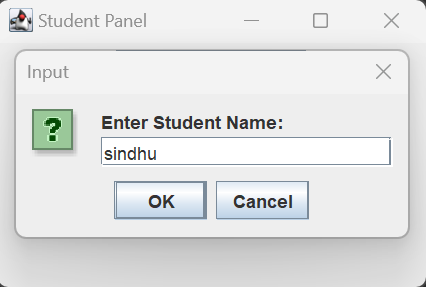
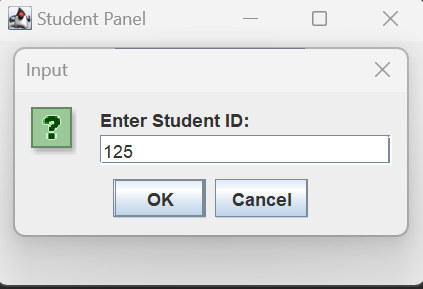
**Student Panel**



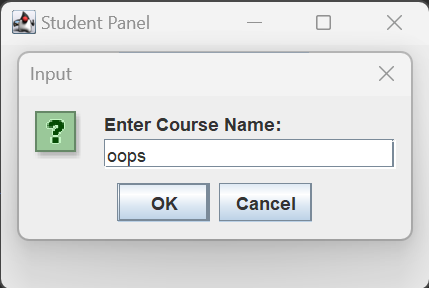
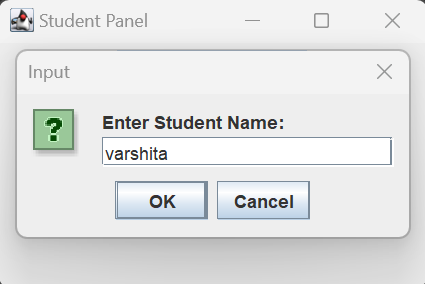
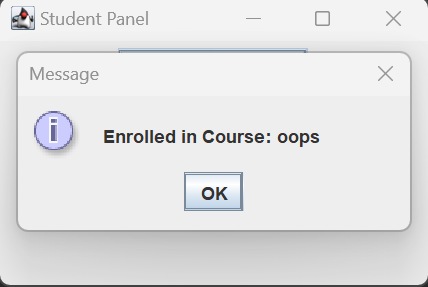
**Student Details**

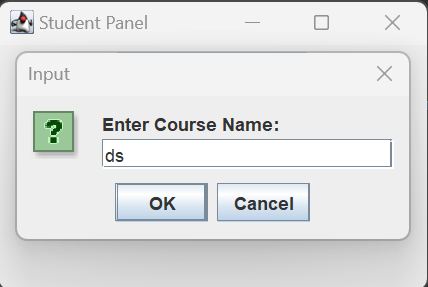
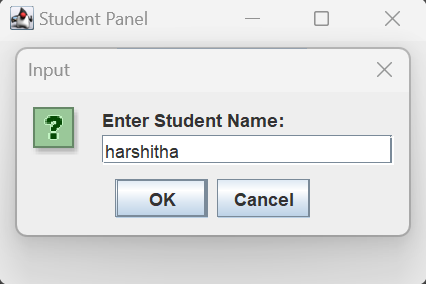
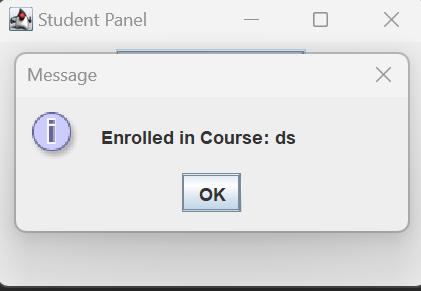
  

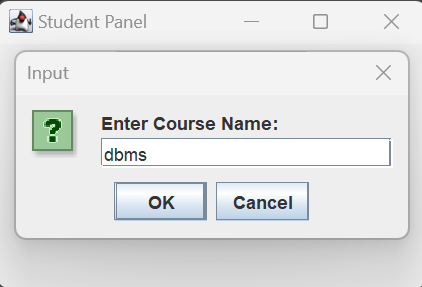
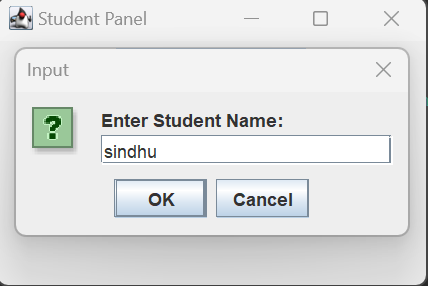
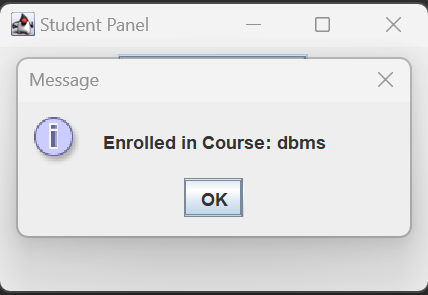
  



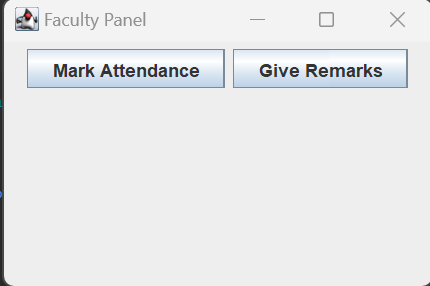
**Enroll in Course**

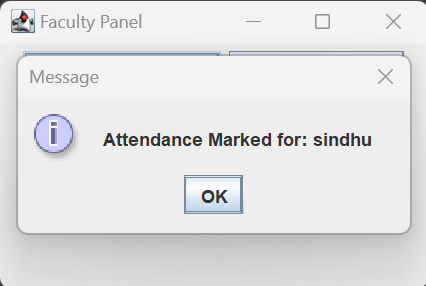
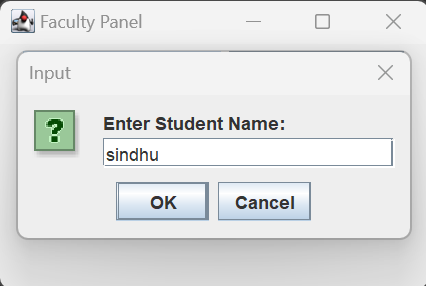
  

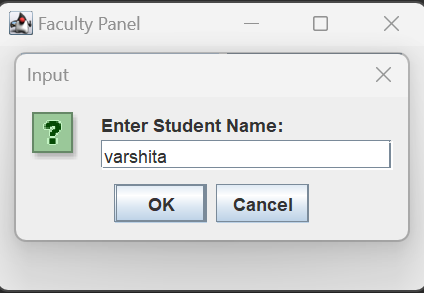
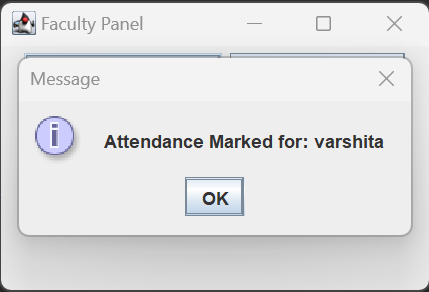
  

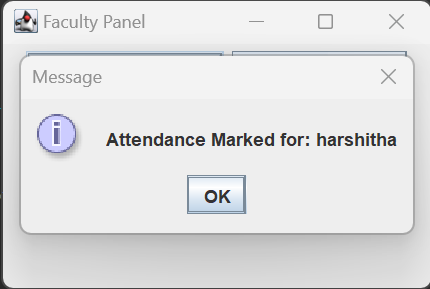
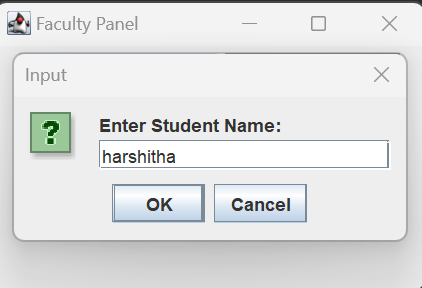
**Faculty Panel**



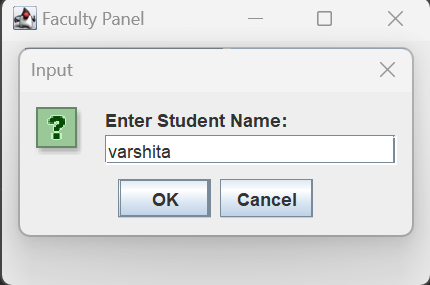
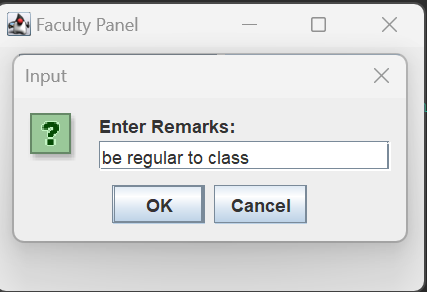
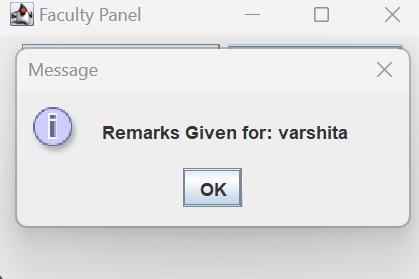
**Mark Attendance**

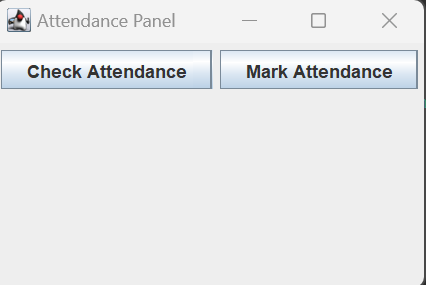
 

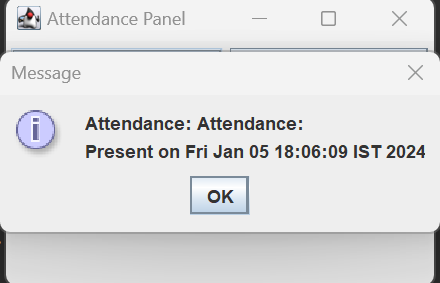
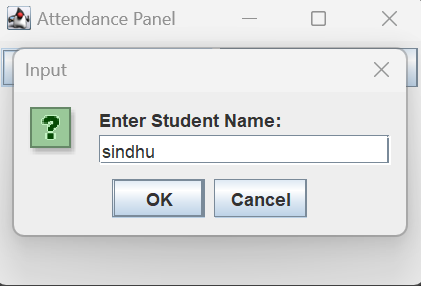
**Give remarks**

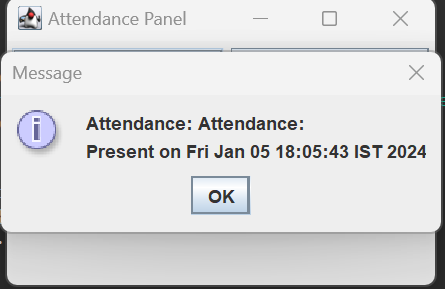
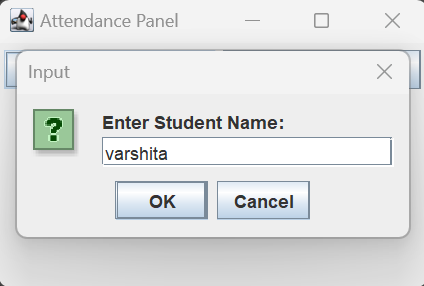
  

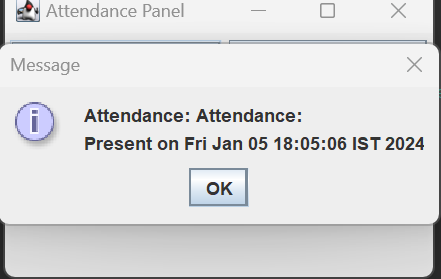
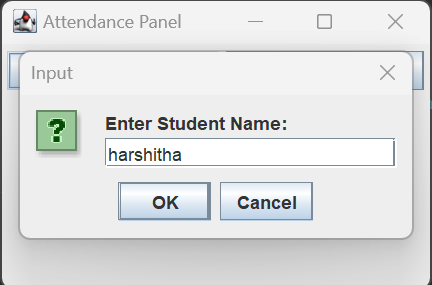
**Attendance Panel**



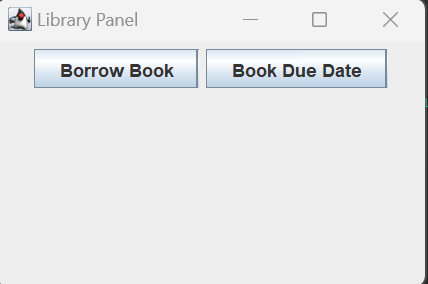
**Check Attendance**

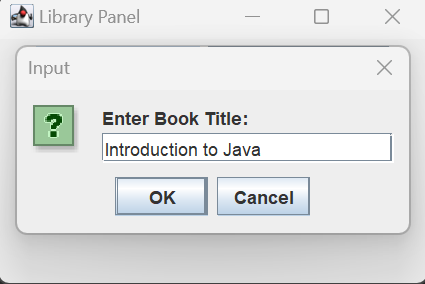
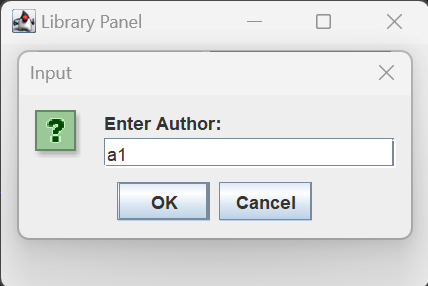
 

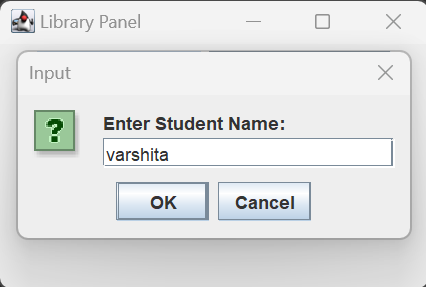
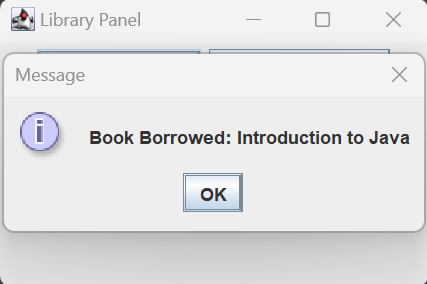
 

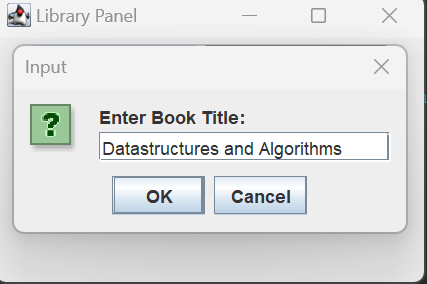
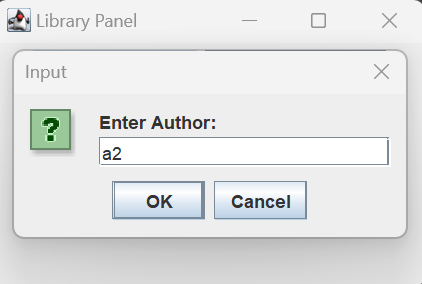
**Library Panel**

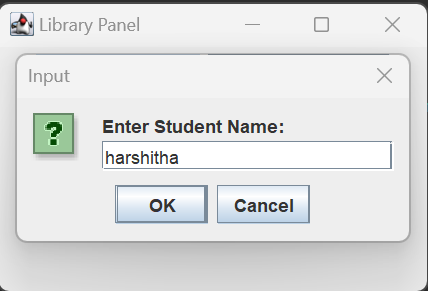
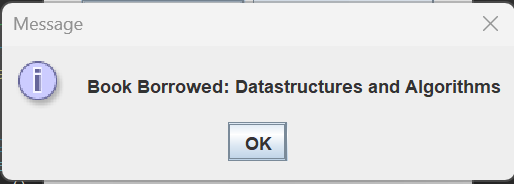


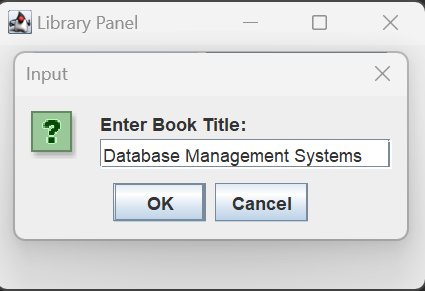
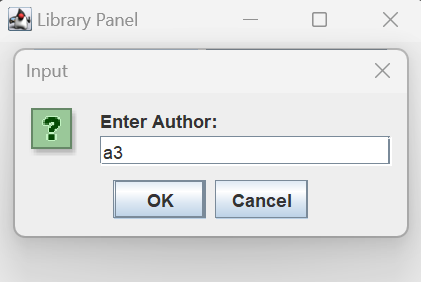
**Borrow Book**

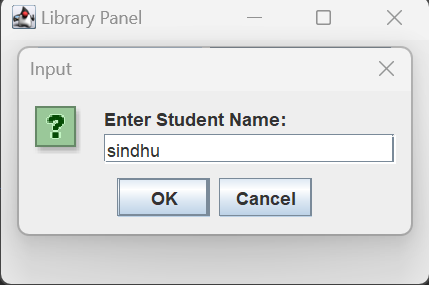
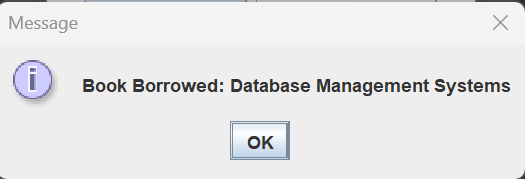
 

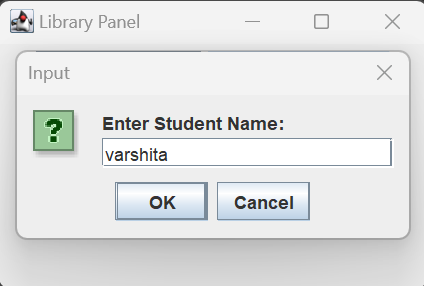
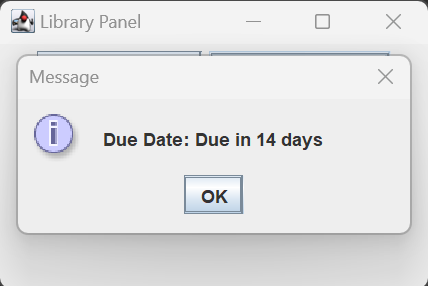
 

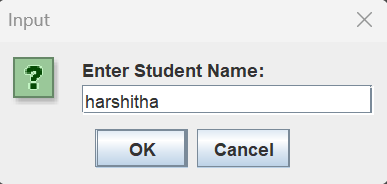
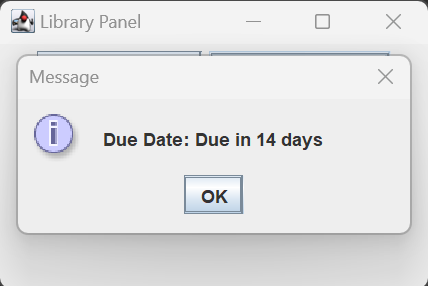
 

**Due Date**

1. **CONCLUSION**

In conclusion, the implementation of a College Management System (CMS) is a transformative step towards enhancing the efficiency, transparency, and overall management of educational institutions.

This comprehensive system, designed to integrate various administrative and academic processes, serves as a centralized hub for information and communication.

The benefits of a well-implemented College Management System extend across different facets of the educational ecosystem.

1. BIBLIOGRAPHY

* [www.slideshare.net](http://www.slideshare.net)
* [www.codeshoppy.com](http://www.codeshoppy.com)
* [www.studypool.com](http://www.studypool.com)
* [www.scribd.com](http://www.scribd.com)