STUDENT INFORMATION SYSTEM

CS23333 – Object Oriented Programming Using JAVA Project Report

Submitted by

AMEEN BASITH K - 231001012

BHARATHKUMAR S - 231001028

HARISHKUMAR M - 231001060

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATIONTECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

NOVEMBER-2024

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this project titled **"STUDENT INFORMATION SYSTEM"** is the bonafide work of **AMEEN BASITH K (231001012) , BHARATHKUMAR S (231001028) , HARISHKUMAR M (231001060)** who carried out the project work under my supervision.Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr P.Valarmathie

HEAD OF THE DEPARTMENT

Information Technology

Rajalakshmi Engineering College

SIGNATURE

Mrs.S Usha

SUPERVISOR

ASSISTANT PROFESSOR (SG),

Information Technology

Rajalakshmi Engineering College

Submitted to Project Viva-Voce Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First, we thank the almighty God for the successful completion of the project. Our sincere thanks to our chairman Mr.S. Meganathan,B.E., F.I.E for his sincere endeavour in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson Dr. Thangam Meganathan, for her enthusiastic motivation which inspired us a lot in completing this project and Vice-Chairman Mr. Abhay Shankar Meganathan B.E., M.S., for providing us with the requisite infrastructure We also express our sincere gratitude to our college principal,Dr.S.N.Murugesan M.E.,PhD.,for his kind support and facilities to complete our work on time.We extend heartfelt gratitude to Dr.P.Valarmathie,Professor and Head of the Department of Information Technology for her guidance and encouragement throughout the work. We are very glad to thank our course faculty Dr.A.Kalaivani,Professor of our department for their encouragement and support towards the successful completion of this project.We extend our thanks to our parents, friends, all faculty members,and supporting staff for their direct and indirect involvement in the successful completion of the project for their encouragement and support.

AMEEN BASITH K
BHARATHKUMAR S
HARISHKUMAR M

TABLE OF CONTENTS

1. LIBRARY MANAGEMENT SYSTEM

1.1 Abstract :

The Student Information System (SIS) is a desktop-based application developed using Java Swing and MySQL, designed to simplify and automate the management of student data for educational institutions. This system offers a comprehensive solution for maintaining accurate and organized student records, including personal details, academic history, and departmental affiliations.

The SIS provides a user-friendly interface for administrators to perform tasks such as adding, editing, and deleting student information, managing academic records, and organizing departmental and course details. Integration with a MySQL database ensures reliable and efficient data storage, retrieval, and updating, minimizing errors and redundancy.

Built with a robust and modular architecture, the system supports multi-user functionality and ensures data integrity and security. The use of Java Swing for the user interface provides an intuitive and responsive environment, while the backend powered by MySQL ensures scalability for future enhancements, such as attendance tracking and performance analytics.

By replacing manual data management processes, the SIS significantly improves administrative efficiency, reduces paperwork, and ensures accurate record-keeping. This project demonstrates the effective use of modern programming tools to address the challenges of traditional student data management systems.

1.2 Introduction :

Educational institutions are fundamental to fostering learning and intellectual growth, and the management of student records, academic history, and related data plays a key role in this process. Traditionally, managing such information involved manual systems, which can be time-consuming, prone to errors, and inefficient. As educational institutions grow, these challenges become more pronounced, particularly in areas such as enrollment tracking, academic history management, and maintaining up-to-date student data. These inefficiencies can impact the quality of administration and, ultimately, the experience of students and staff.

The Student Information System (SIS) offers a comprehensive solution to these challenges by automating the management of student data, academic history, and institutional resources. By integrating a secure and efficient database with a user-friendly web interface, the system simplifies key processes for administrators, staff, and students. Key features include student enrollment management, academic history tracking, and course management, allowing administrators to add, update, and delete student records easily, while students can access their own information securely.

The automation of these tasks reduces manual effort, minimizes the risk of errors, and ensures accurate data management. For example, administrators can track student enrollment, academic status, and generate reports with ease. The system's intuitive design enhances the user experience, offering clear navigation and quick access to student and academic data. Real-time updates guarantee that all records are accurate and up-to-date, making it easier for administrators to manage the institution's academic resources effectively.

The Student Information System is designed to be scalable, ensuring that as the institution grows, the system can adapt to handle larger volumes of data and more complex requirements. Future enhancements could include features like course scheduling, student attendance tracking, and integration with other institutional systems. By adopting this system, educational institutions can streamline administrative processes, improve operational efficiency, and provide a seamless experience for both students and staff, ultimately supporting the institution's mission to provide high-quality education.

1.3 Purpose :

The purpose of this project is to develop a robust Student Information System (SIS) application that:

- Simplifies the process of managing student records, academic history, courses, and departments.
- Provides an efficient method for tracking student enrollment, academic status, and course management.
- Offers a secure, scalable, and user-friendly platform for administrators and staff to manage and update student data.
- Ensures real-time updates and accuracy of all student-related information.
- Streamlines user management, enabling role-based access for administrators, staff, and students.

1.4 Scope of the Project :

The Student Information System is designed for educational institutions of various sizes and offers:

- Administrator capabilities for managing student records, academic history, course offerings, departments, and user accounts.
- Student functionalities such as enrolling, viewing their personal and academic data, and accessing their course and departmental information.
- Course and department management, allowing the creation, editing, and deletion of courses and departments by authorized users.
- User management system to create and maintain user accounts with different access levels (admin, staff, student).
- Integration with a relational MySQL database to store and retrieve all relevant data, ensuring data integrity and security.
- A foundation for future enhancements, such as the addition of features like attendance tracking, grade management, or cloud-based deployment for better scalability and accessibility.

1.5 Software Requirement Specification

Introduction:

The Student Information System (SIS) is designed to manage and streamline all aspects of student records, academic history, course management, and user administration. This Software Requirements Specification (SRS) document outlines the functional and technical requirements for the SIS, providing detailed insights into the system's design, architecture, and implementation. The document serves as a guide for developers, stakeholders, and future maintenance teams to ensure successful deployment and continued system evolution.
Document Purpose:

The purpose of this document is to outline the specific requirements of the Student Information System (SIS) to ensure it meets the administrative needs of educational institutions. It details how the system should function, its capabilities, and the architecture, along with user and hardware requirements.

Product Scope:

The Student Information System is developed to manage student records, academic history, and course offerings, providing a comprehensive and flexible solution for educational institutions. The system aims to replace outdated manual processes, offering an efficient platform for managing student data, tracking academic progress, and handling departmental information. It integrates seamlessly with a relational database to store and retrieve data, ensuring the system remains adaptable to growing institutional needs.

Overall Description:

The Student Information System (SIS) allows administrators to manage and maintain comprehensive student records. It automates essential functions such as student registration, academic history tracking, and user management, ensuring smooth operations and minimizing manual effort. The system improves efficiency by offering a centralized platform for managing student data, departmental information, and course offerings.

Product Perspective:

The system employs a client-server architecture, with a desktop-based front end developed using Java Swing and a MySQL database for backend data storage and retrieval. It is designed to operate seamlessly on various platforms (Windows/Linux) and provides a user-friendly interface for managing student and academic data. Key modules of the Student Information System include:

Student Management: Manage student records, enrollments, and academic history.

Course and Department Management: Add, update, and delete courses and departments.

User Management: Create and manage different user roles (Admin, Staff, Student).

Academic History: Track and manage student academic progress, semester data, and status.

User Characteristics:

Qualification: Users should have basic computer literacy and familiarity with educational operations.

Experience: Experience with student record management systems or administrative tools is beneficial.

Technical Knowledge: Users should be comfortable navigating web-based applications and using forms.

Operating Environment:

Hardware Requirements:

- Processor: Intel i3 or higher
- RAM: 4GB or more
- Hard Disk: 500GB or more
- Display: Resolution of 1024 x 768 or higher

Software Requirements:

- Operating System: Windows 10 or higher / Linux
- Programming Language: Java (JDK 11 or higher)
- Development Environment: NetBeans IDE 12.0 or IntelliJ IDEA
- Database: MySQL Server 8.0 or higher
- Database Connectivity: JDBC (Java Database Connectivity)
- Web Server: Apache (for database hosting, if required)
- External Libraries/Dependencies: MySQL Connector/J for Java

Constraints:

System Access: Restricted to authorized users with appropriate roles (Admin, Staff, Student).

Data Integrity: Deletion of student records, courses, or departments should be irreversible to maintain data integrity.

System Maintenance: Administrators should be responsible for managing and maintaining the system, including handling backup and updates.

Assumptions and Dependencies:

User Management: Admins are responsible for creating and securely managing user accounts with proper access controls.

Network Requirements: The system assumes a stable network connection for database interaction and multi-user access.

System Scalability: The system is designed to scale as the institution grows, with the ability to handle larger volumes of student and academic data.

Specific Requirements:

User Interface:

The Student Information System (SIS) provides a web-based interface for the following operations:

Admin Registration and Login: Allows admins to securely log into the system and manage records.

Student Record Management: Add, view, update, and delete student records.

Course and Department Management: Add, view, update, and delete courses and department details.

Academic History: Manage academic histories for students, including semesters, courses, and status.

User Management: Admins can manage user roles and permissions for staff and students.

Hardware Interface:

Resolution: Minimum screen resolution of 1024 x 768 pixels.
Platform Compatibility: Works across Windows and Linux systems.

Software Interface:

- Desktop Environment: Java Swing for the graphical user interface (GUI).
- Database: MySQL for backend data storage and management.
- Backend Integration: JDBC (Java Database Connectivity) for communication between Java and MySQL.
- Development Environment: NetBeans IDE for application development and debugging.

Functional Requirements:

Login Module (LM):

- Validates administrator credentials to ensure secure access.
- Implements secure login with encrypted password storage.

Student Management Module (SMM):

- Allows administrators to add, update, view, and delete student records.
- Provides search functionality for student records.

Course and Department Module (CDM):

- Enables administrators to manage courses and department details.
- Allows viewing and updating department or course information.

User Management Module (UMM):

- Admins can create, update, and delete user accounts.
- Manages user roles such as Admin, Staff, and Student.

Non-functional Requirements:

Performance:

The system must handle operations with a response time of less than 2 seconds for student record management and course updates.

Reliability:

The system must maintain accurate student records, academic histories, and course data, ensuring consistency across all operations.

Availability:

The system should ensure that data is accessible and updates are performed without delay.

Security:

Sensitive student and academic information should be securely stored with encryption.
The system must prevent unauthorized access to data.

Maintainability:

System documentation should be available for ongoing maintenance and feature upgrades.
Admins should have tools for maintaining the database and updating the system as needed.

2.SYSTEM FLOW DIAGRAMS

2.1 Use Case Diagrams :

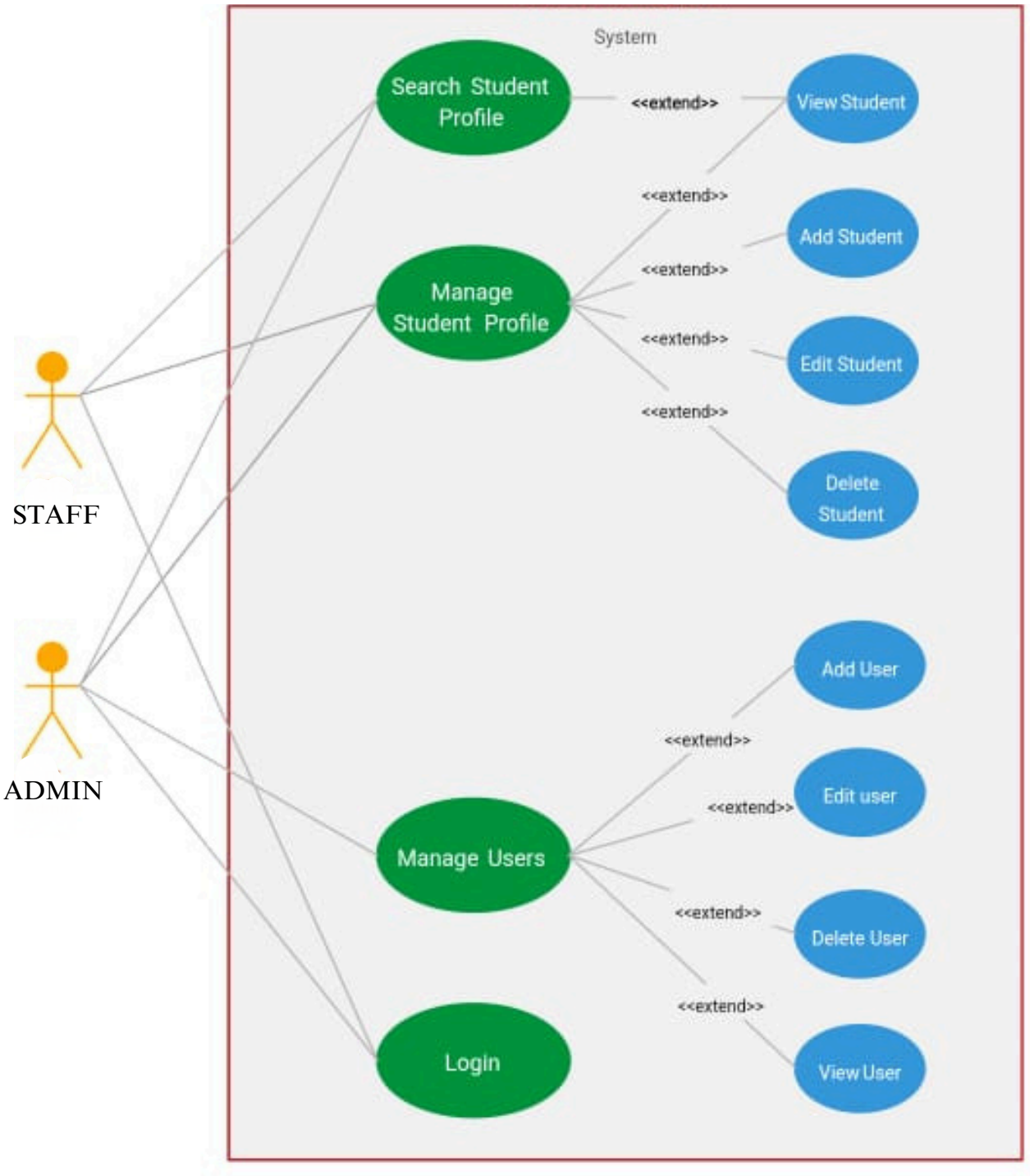


Fig 2.1.1 Use Case Diagram

2.2 Entity-Relationship Diagram :

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in the table.

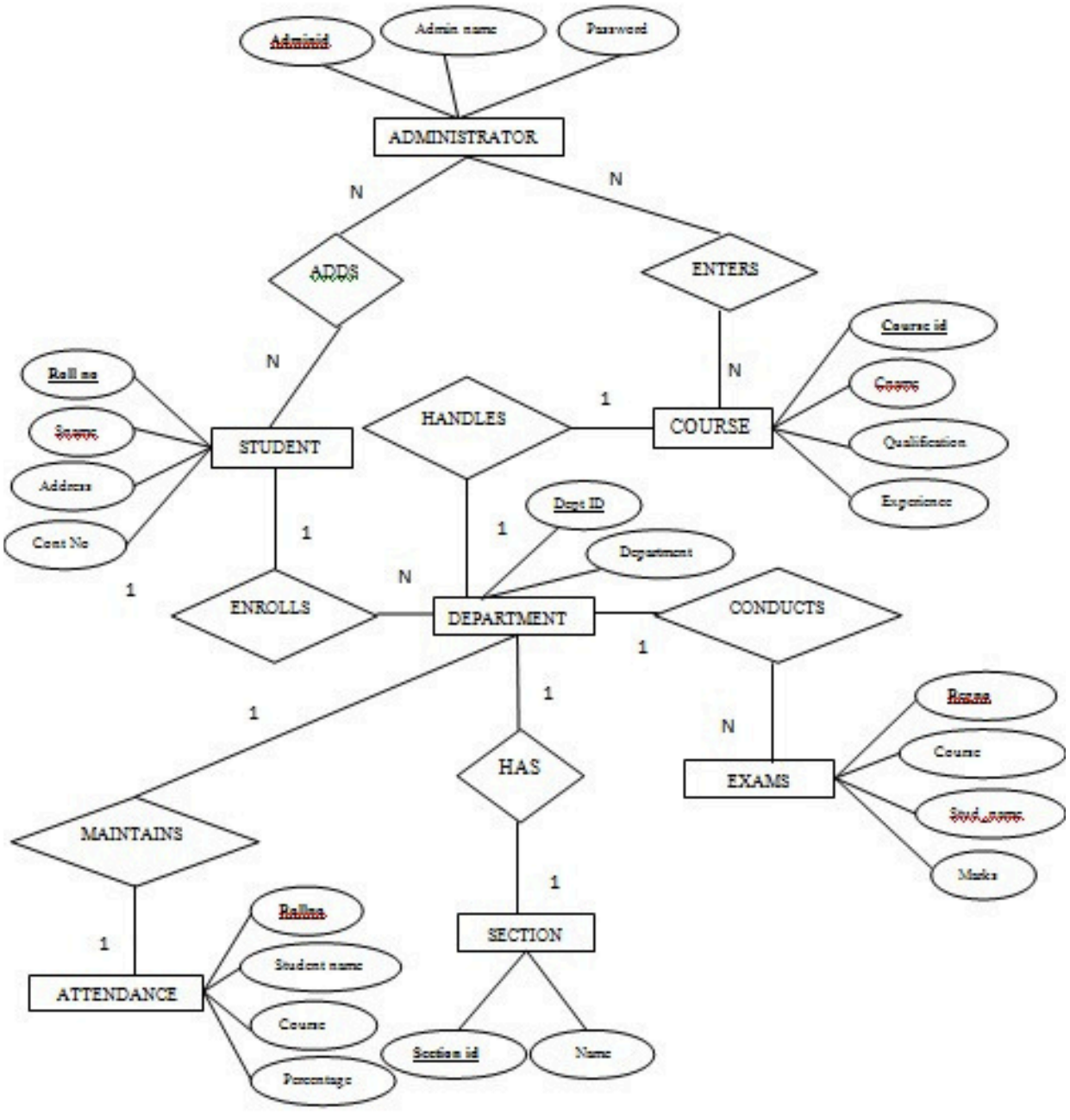


Fig 2.2.1 Entity-Relationship Diagram

2.3 Data-Flow Diagram :

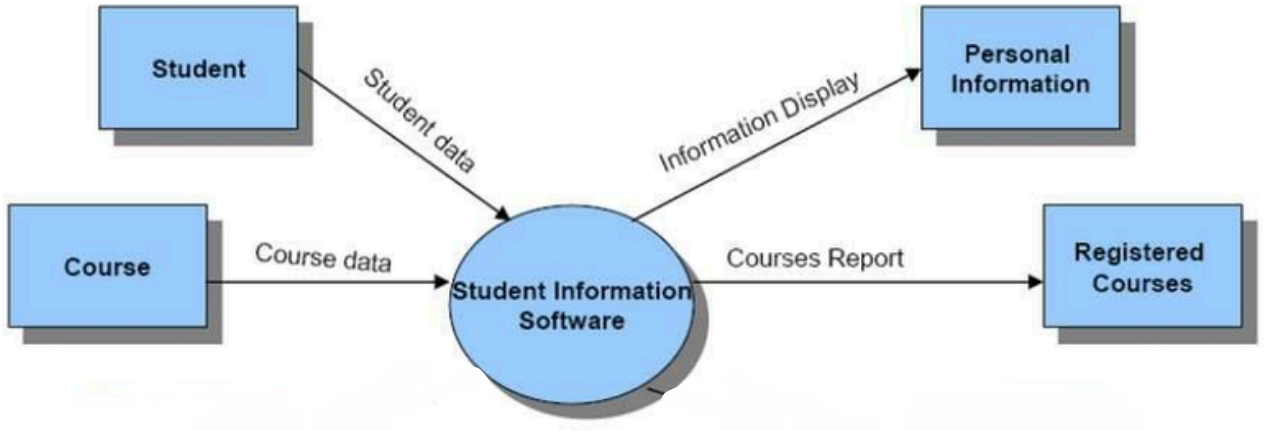


Fig 2.3.1 Data Flow Diagram

3.MODULE DESCRIPTION :

The Student Information System (SIS) is a desktop application designed to efficiently manage student records, academic histories, courses, and departments for educational institutions. Developed using Java Swing for the user interface and MySQL for backend data management, the system provides an intuitive platform for administrators to perform various tasks related to student and academic management. Below is a detailed description of its modules.

4.1 Student Management Module

The Student Management Module is the core of the Student Information System. It helps administrators manage student records efficiently.

Key Features:

Add New Student:

Administrators can add new students by entering details such as roll number, name, gender, date of birth, contact information, and addresses. The entered data is securely stored in the database for future reference.

Update Student Records:

The system allows administrators to update student details, such as contact information or name corrections.

Remove Students:

Administrators can remove student records from the system when necessary, for example, when a student graduates or leaves the institution.

Track Student Status:

The system displays the status of each student (active or inactive), helping administrators manage the student list efficiently

4.2 Academic History Module

The Academic History Module helps track the academic progress of each student over time. It allows administrators to manage enrollment information, semester details, and course status.

Key Features:

Add Academic History:

Administrators can add academic records for each student, including the department, course, and semester details.

Update Academic History:

The system allows updates to academic history, such as marking a semester as completed or regular and changing course details as needed.

View Academic History:

Administrators can view the full academic history of a student, including their past and current courses, semester status, and progress over time.

Remove Academic History:

The system supports deleting academic records when necessary, such as when correcting errors or updating student status.

4.3 Department Management Module

The Department Management Module allows the administration departments within the institution.

Key Features:

Add New Department:

Administrators can add new departments by entering the name, description, and status (active or inactive).

Update Department Details:

The system allows for easy updates to department names, descriptions, and statuses.

Remove Departments:

The system provides options to remove departments when they are no longer needed, such as during reorganization or mergers.

View Department List:

Administrators can view the full list of departments, including their status, descriptions, and actions.

4.4 Course Management Module

The Course Management Module helps manage the courses offered by the institution, including their association with departments.

Key Features:

Add New Course:

Administrators can add new courses, linking them to specific departments and providing details like course name, description, and status.

Update Course Details:

The system allows administrators to update course details, including changing department affiliations or adjusting course descriptions.

Remove Courses:

Courses that are no longer offered can be removed from the system.

View Course List:

Administrators can view the entire course list, including the departments they are associated with, course descriptions, and their statuses.

4.5 User Management Module

The User Management Module manages user accounts and access control within the system. It ensures secure and efficient login and credential management.

Key Features:

Add New User:

Administrators can add new users by providing their name, username, and password. Users can be assigned specific roles (Admin, Faculty, etc.) to control access levels.

Manage Login Credentials:

The system allows users to manage their passwords and login details securely.

Remove Users:

User accounts can be deactivated or deleted when they are no longer needed, for example, when faculty members leave the institution.

This structure will guide the implementation and usage of the Student Information System, ensuring efficient data management and easy accessibility for all user roles.

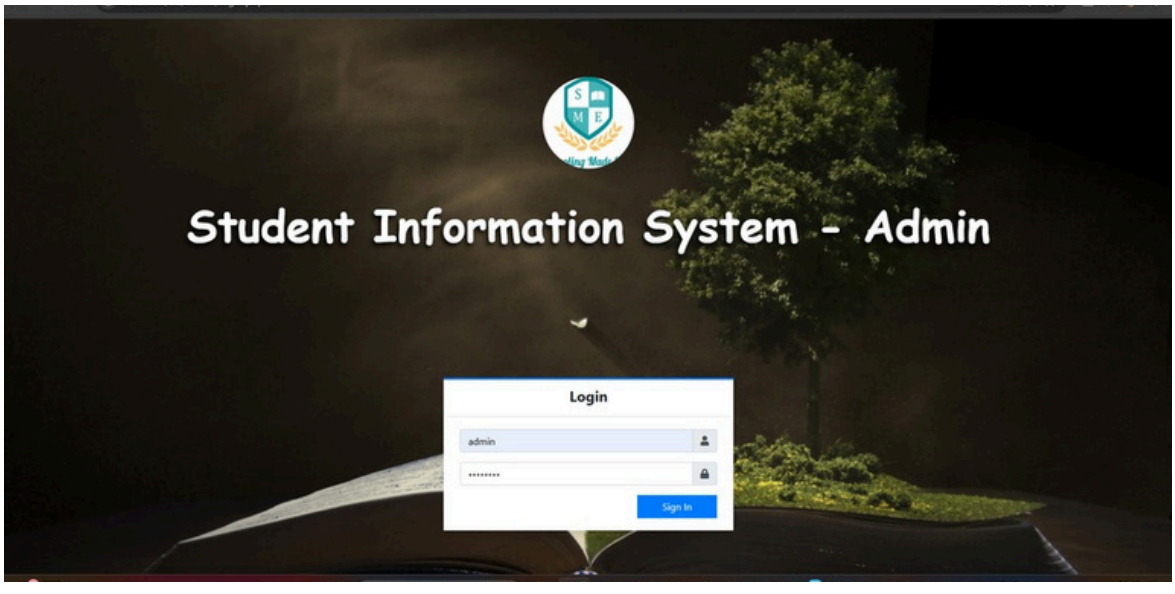
4.DESIGN

4.1 Web Design

This section provides a visual overview of the Student Information System interface. The illustrations highlight the system's functionalities, including the admin panel, dashboard, and management pages for students, departments, courses, and users. Each section demonstrates the intuitive design and streamlined workflow of the application.

6.1 Login Page

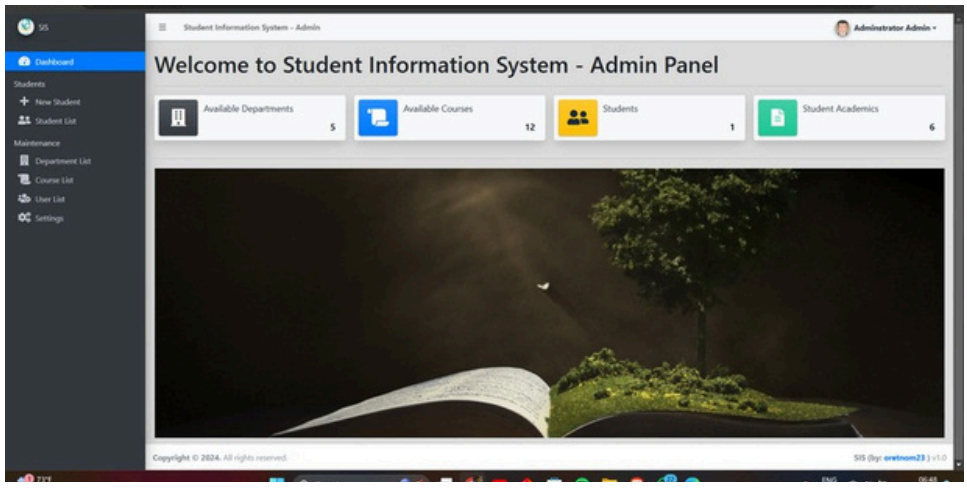
The login page serves as the entry point to the Student Information System.
Features a secure login form requiring valid credentials for access.
Ensures role-based access control, allowing only authorized users to proceed.



6.2 Dashboard

The dashboard functions as the central hub, offering quick access to key features:

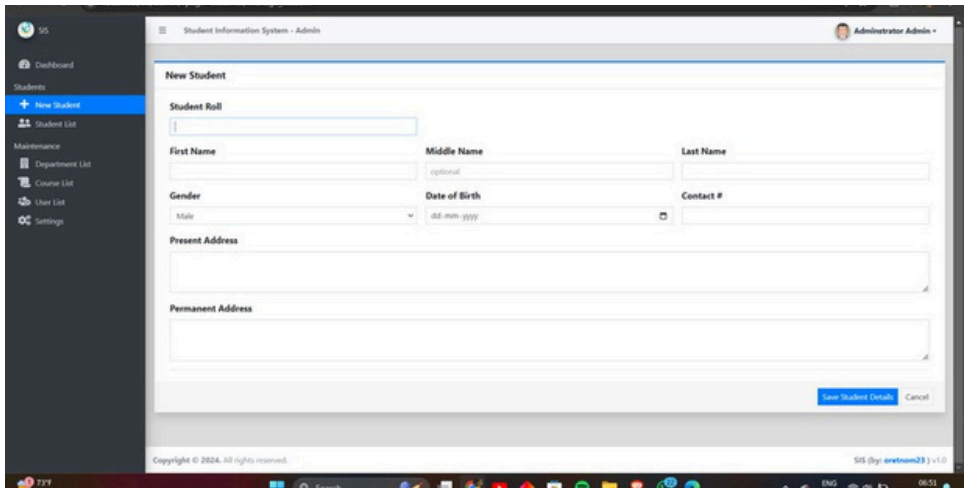
- Student Management
- Department Management
- Course Management
- User Management
- Quick statistics on the total number of students, departments, and courses.
- Designed for efficient navigation with a clean and user-friendly layout.



6.3 Student Management

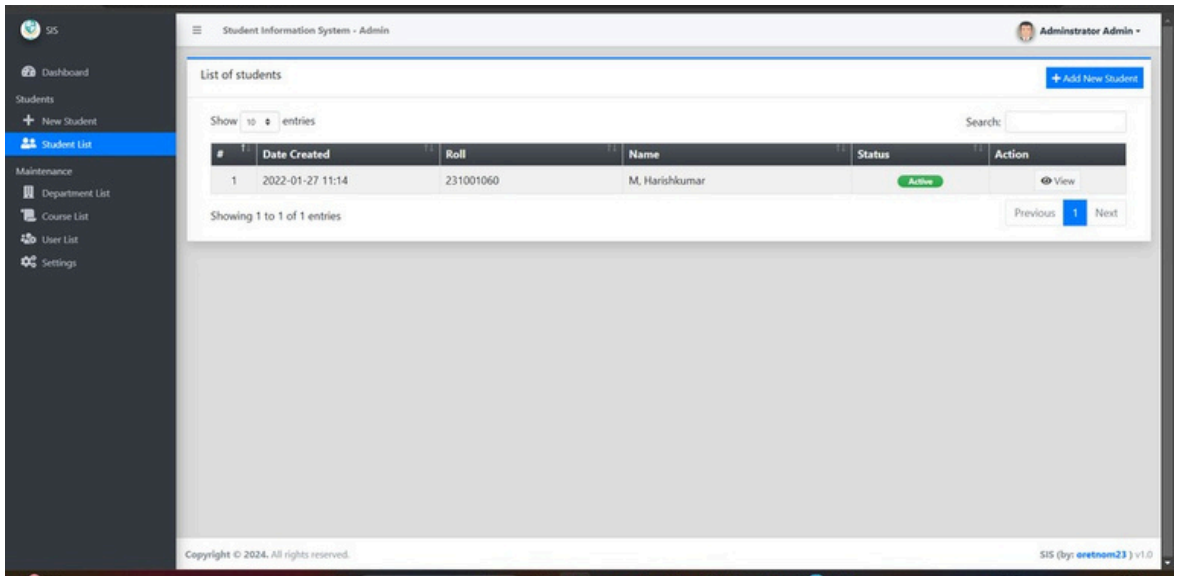
New Student Page

- Form to add new student records, capturing details such as roll number, name, gender, date of birth, contact information, and addresses.
- Validation ensures accurate data entry.



Student List

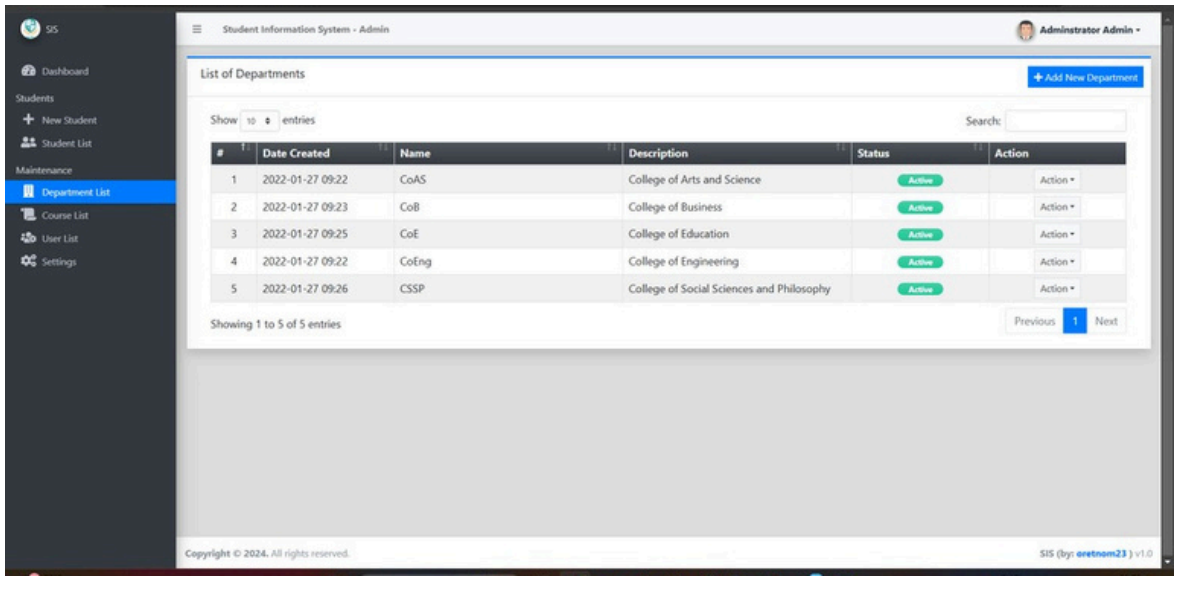
- Displays a searchable and sortable list of all students.
- Columns include roll number, name, status (Active/Inactive), and actions (View, Edit, Delete).

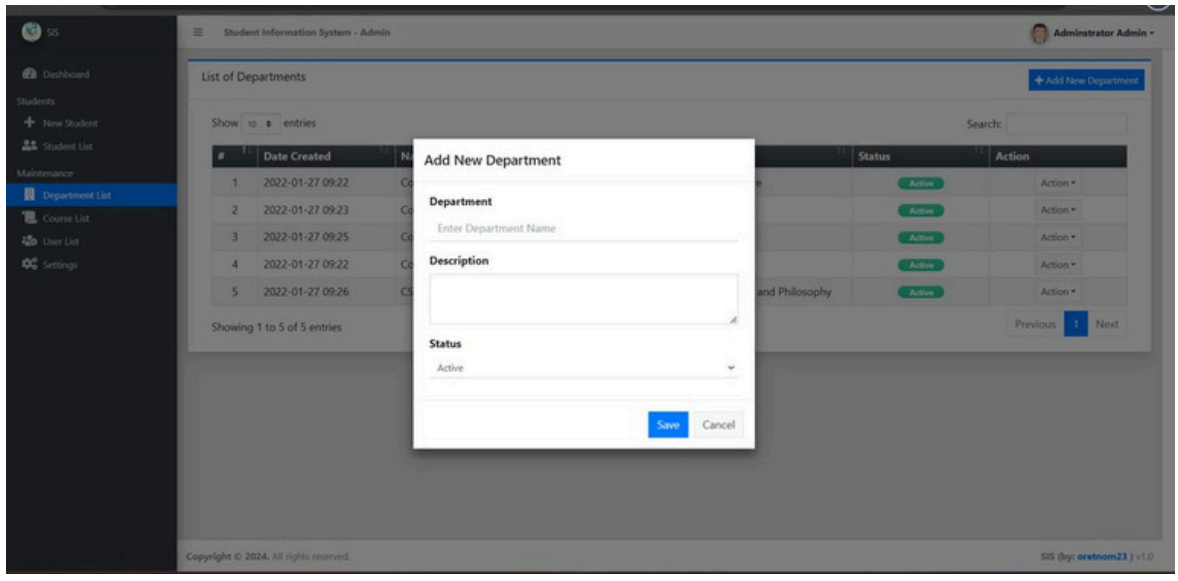


6.4 Department Management

Department List

- Displays a list of departments with fields like name, description, and status.
- Actions include adding a new department, editing existing ones, or deleting unnecessary entries.

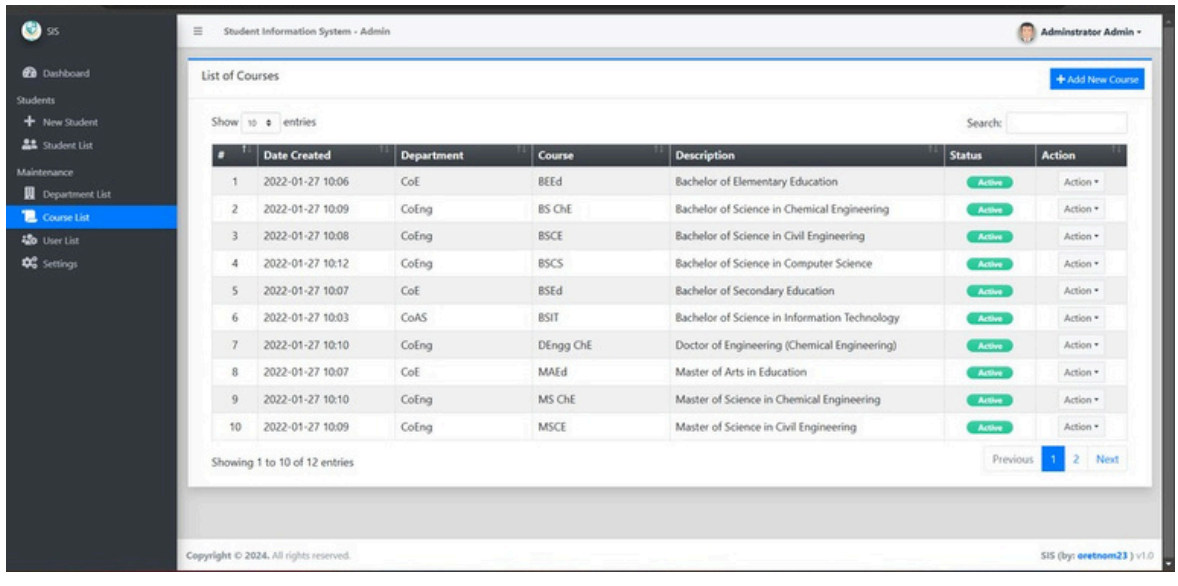


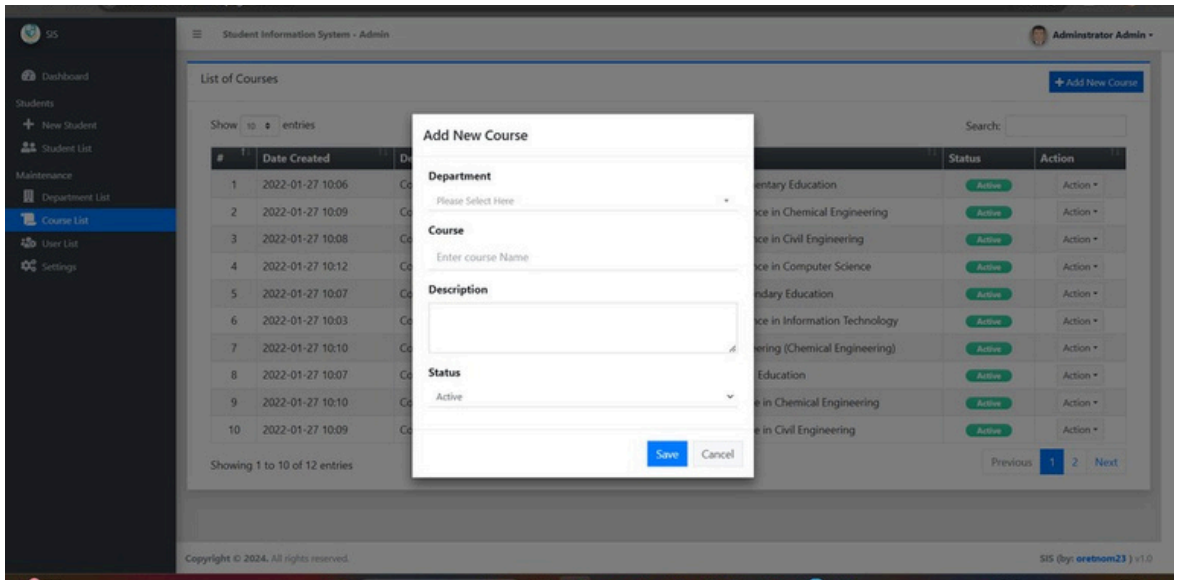


6.5 Course Management

Course List

- Displays a catalog of courses, organized by department.
- Fields include course name, description, and status.
- Features to add, edit, or delete courses.

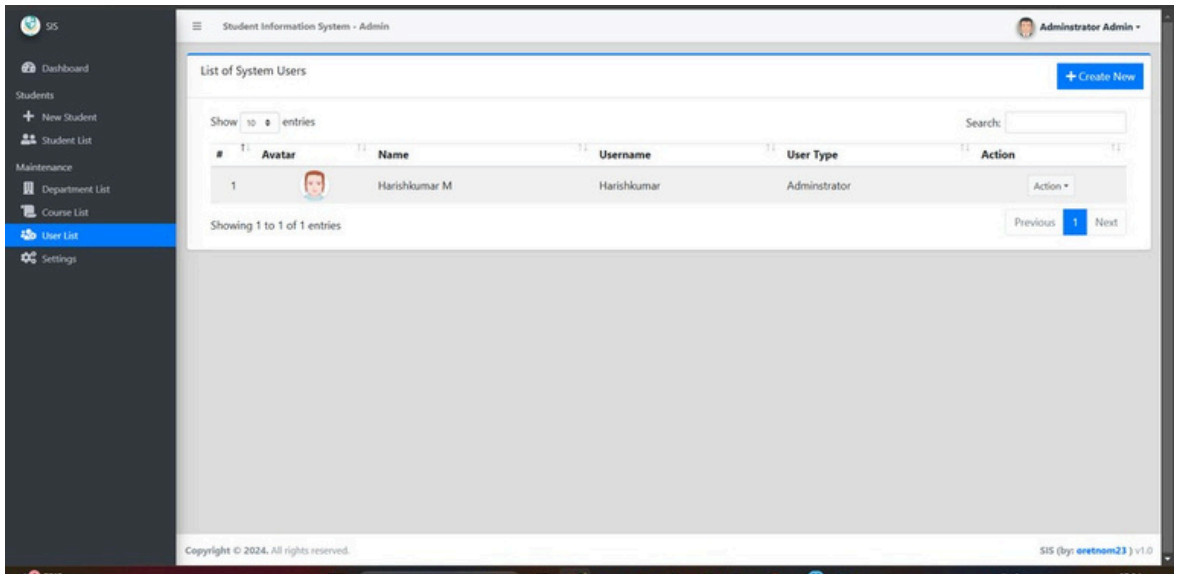


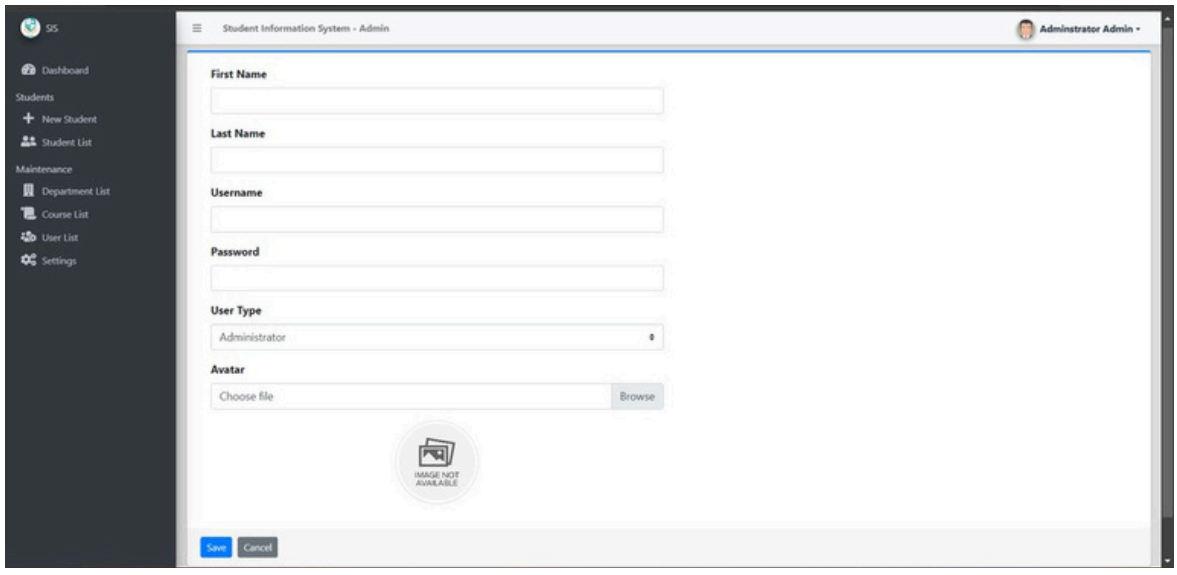


6.6 User Management

User List

- Displays a list of system users with details such as name, username, user type, and actions.
- Allows admins to create new user accounts or deactivate/remove existing ones.



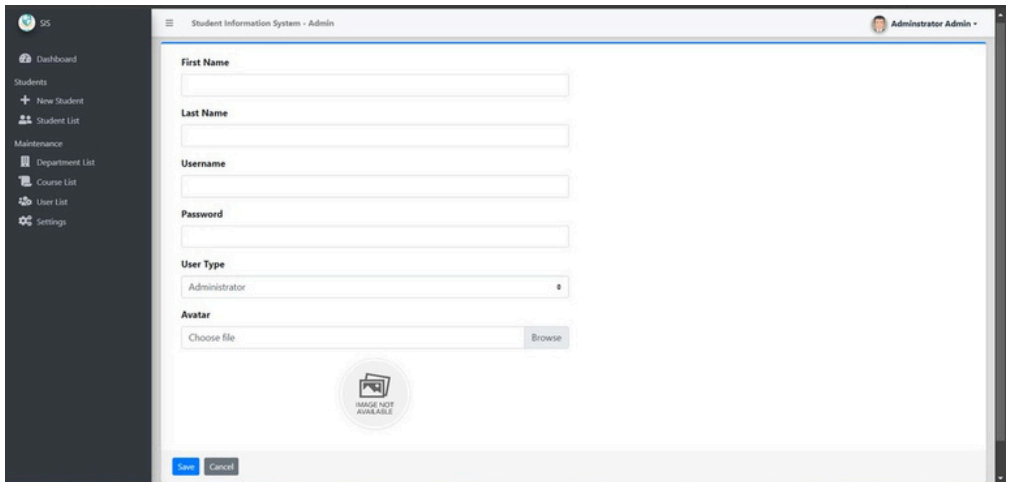


6.7 Settings

Allows customization of system information, such as:

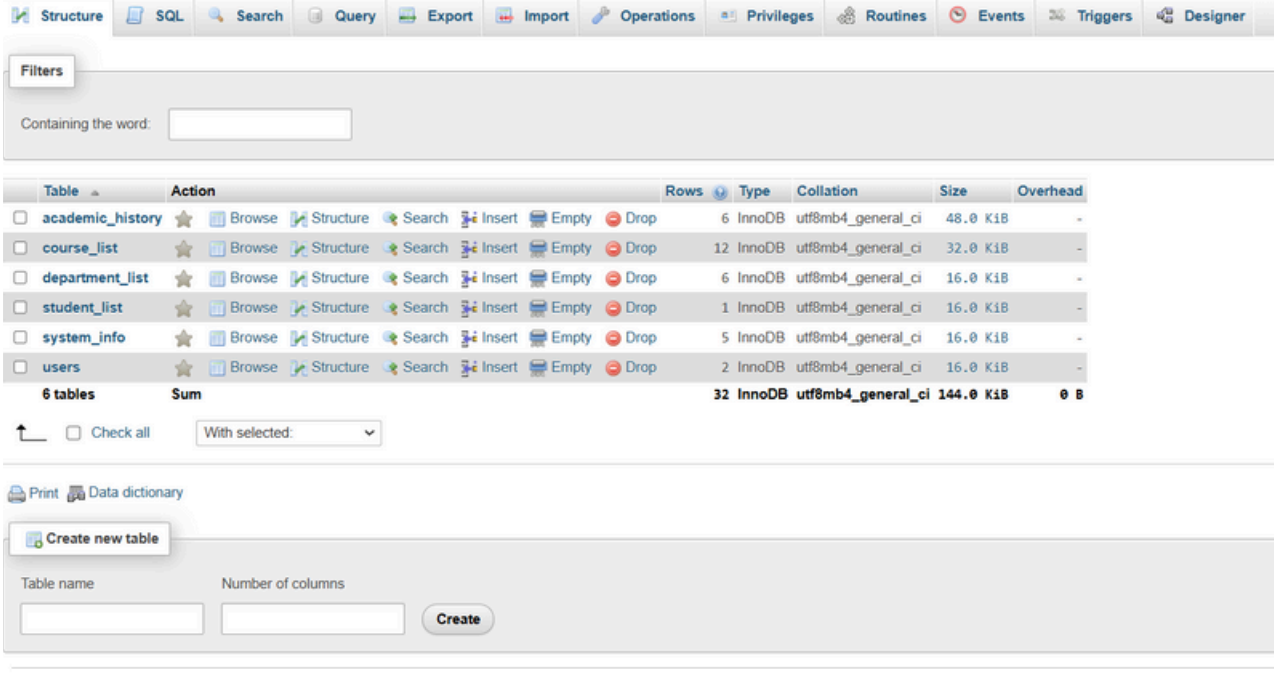
- System name Logo
- cover images
- General configuration settings

Ensures the system reflects the library's branding and operational preferences.



4.2 Database Design :

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system. A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive and flexible for the user. Relationships are established between the data items and unnecessary data items are removed.Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required,minimizing chances of data inconsistencies and optimizing for updates. The MySQL database has been chosen for developing the relevant databases.



4.3 Implementation(Code) :

```
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class StudentInformationSystem {

    private Connection con;
    private Statement stmt;
    private JComboBox<String> genderComboBox;
    private JTextField rollTextField, firstNameTextField, lastNameTextField, contactTextField;
    private JTextArea addressTextArea, resultTextArea;
    private JButton addButton, viewButton;

    public StudentInformationSystem() {
        JFrame frame = new JFrame("Student Information System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new BorderLayout());

        // Top Panel (Input Fields)
        JPanel topPanel = createTopPanel();
        topPanel.setBackground(Color.LIGHT_GRAY);

        // Result Panel (Display Results)
        JPanel resultPanel = createResultPanel();
```

```
    // Button Panel
    JPanel buttonPanel = new JPanel(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.anchor = GridBagConstraints.CENTER;
    buttonPanel.add(createAddButton(), gbc);
    buttonPanel.add(createViewButton(), gbc);

    // Add Panels to Frame
    frame.add(topPanel, BorderLayout.NORTH);
    frame.add(resultPanel, BorderLayout.CENTER);
    frame.add(buttonPanel, BorderLayout.SOUTH);

    frame.pack();
    frame.setVisible(true);

    setupDatabaseConnection();
}

private JPanel createTopPanel() {
    JPanel topPanel = new JPanel(new GridLayout(6, 2, 10, 10));

    rollTextField = new JTextField(15);
    firstNameTextField = new JTextField(15);
    lastNameTextField = new JTextField(15);
    contactTextField = new JTextField(15);
    addressTextArea = new JTextArea(3, 15);
    genderComboBox = new JComboBox<>(new String[]{"Select Gender", "Male", "Female", "Other"});

    topPanel.add(new JLabel("Roll Number:"));
    topPanel.add(rollTextField);
    topPanel.add(new JLabel("First Name:"));
    topPanel.add(firstNameTextField);
    topPanel.add(new JLabel("Last Name:"));
    topPanel.add(lastNameTextField);
```

```
    topPanel.add(new JLabel("Last Name:"));
    topPanel.add(lastNameTextField);
    topPanel.add(new JLabel("Gender:"));
    topPanel.add(genderComboBox);
    topPanel.add(new JLabel("Contact:"));
    topPanel.add(contactTextField);
    topPanel.add(new JLabel("Address:"));
    topPanel.add(new JScrollPane(addressTextArea));

    return topPanel;
}

private JPanel createResultPanel() {
    JPanel resultPanel = new JPanel(new BorderLayout());
    resultTextArea = new JTextArea(10, 30);
    resultTextArea.setEditable(false);
    resultPanel.add(new JScrollPane(resultTextArea), BorderLayout.CENTER);
    return resultPanel;
}

private JButton createAddButton() {
    addButton = new JButton("Add Student");
    addButton.addActionListener(e -> addStudent());
    return addButton;
}

private JButton createViewButton() {
    viewButton = new JButton("View Students");
    viewButton.addActionListener(e -> viewStudents());
    return viewButton;
}
```

```
private void setupDatabaseConnection() {
    try {
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/student_info", "root", "root");
        stmt = con.createStatement();
        stmt.executeUpdate("CREATE TABLE IF NOT EXISTS students ("
                + "roll INT PRIMARY KEY, "
                + "firstName VARCHAR(50), "
                + "lastName VARCHAR(50), "
                + "gender VARCHAR(10), "
                + "contact VARCHAR(15), "
                + "address TEXT)");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void addStudent() {
    try {
        int roll = Integer.parseInt(rollTextField.getText());
        String firstName = firstNameTextField.getText();
        String lastName = lastNameTextField.getText();
        String gender = (String) genderComboBox.getSelectedItem();
        String contact = contactTextField.getText();
        String address = addressTextArea.getText();

        if (gender.equals("Select Gender")) {
            resultTextArea.setText("Please select a valid gender.");
            return;
        }
```

```
        String query = "INSERT INTO students (roll, firstName, lastName, gender, contact, address) "
                + "VALUES (?, ?, ?, ?, ?, ?)";
        PreparedStatement ps = con.prepareStatement(query);
        ps.setInt(1, roll);
        ps.setString(2, firstName);
        ps.setString(3, lastName);
        ps.setString(4, gender);
        ps.setString(5, contact);
        ps.setString(6, address);
        ps.executeUpdate();

        resultTextArea.setText("Student added successfully!");
        clearFields();
    } catch (SQLException | NumberFormatException e) {
        e.printStackTrace();
        resultTextArea.setText("Error adding student. Please check your inputs.");
    }
}

private void viewStudents() {
    try {
        String query = "SELECT * FROM students";
        ResultSet rs = stmt.executeQuery(query);
        StringBuilder result = new StringBuilder("Student List:\n");
        while (rs.next()) {
            result.append("Roll: ").append(rs.getInt("roll"))
                    .append(", Name: ").append(rs.getString("firstName"))
                    .append(" ").append(rs.getString("lastName"))
                    .append(", Gender: ").append(rs.getString("gender"))
                    .append(", Contact: ").append(rs.getString("contact"))
                    .append(", Address: ").append(rs.getString("address"))
                    .append("\n");
        }
```

```
resultTextArea.setText(result.toString());
} catch (SQLException e) {
e.printStackTrace();
resultTextArea.setText("Error fetching student list.");
}
}
private void clearFields() {
rollTextField.setText("");
firstNameTextField.setText("");
lastNameTextField.setText("");
genderComboBox.setSelectedItem("Select Gender");
contactTextField.setText("");
addressTextArea.setText("");
}
public static void main(String[] args) {
SwingUtilities.invokeLater(StudentInformationSystem::new);
}
}
```

5.CONCLUSION

The Student Information System (SIS) project, meticulously designed and implemented, represents a significant step forward in the automation of student data management. Featuring functionalities such as adding new students, managing academic histories, and organizing departmental and course details, the system offers a streamlined interface for administrators to handle educational records efficiently.

With the integration of a robust MySQL database, the SIS ensures secure and accurate storage of information, reducing redundancy and errors. The project demonstrates a user-focused approach, emphasizing ease of use and data integrity. Additionally, the modular architecture allows scalability, paving the way for future enhancements, such as integrating attendance tracking, performance monitoring, or automated report generation.

By addressing the key challenges of traditional manual processes, this system not only improves administrative workflows but also provides a foundation for more advanced, data-driven solutions in educational institutions.

6.REFERENCES

[1] https://www.javatpoint.com/java-awt

[2] https://www.mysql.com/

[3] https://www.w3schools.com/java/

[4] https://www.oracle.com/java/technologies/javase-jdk11-doc-downloads.html