RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

STUDENT RECORD MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

SHREENITHI RB 231001197

SHRUTI P 231001199

SRUTHI VARSHNI V 231001217

In partial fulfillment for the award of the degree of

BACHELOR OF

TECHNOLOGY IN

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)
THANDALAM

CHENNAI-602105

2024 – 2025

BONAFIDE CERTIFICATE

Certified that this project report **"STUDENT RECORD SYSTEM "** is the bonafide work of **"SHREENITHI RB (231001197), SHRUTI P (231001199),SRUTHI VARSHNI V (231001217)"**

who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mr.Narayana K.E,

Assistant Professor

Department of Information Technology ,

Rajalakshmi Engineering College,

Thandalam,Chennai-602 105.

SIGNATURE

Dr.P.Valarmathie

Professor and HOD

Department of Information Technology,

Rajalakshmi Engineering College,

Thandalam,Chennai-602 105.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The Student Record Management System is a comprehensive software solution developed to efficiently manage and maintain student records in an educational environment. It combines a Java-based frontend with a MySQL database backend for reliable data storage and retrieval. The system allows administrators to add, delete, and view student information, including essential details like Roll Number, Name, Department, and Marks. Through a simple and user-friendly interface, the system enables easy input of student data and real-time updates of the student record table. The MySQL database ensures the secure storage of student records, providing fast and accurate querying capabilities. By automating the process of record management, the system minimizes manual data entry errors and enhances overall operational efficiency. The software also supports the deletion and modification of student records, helping administrators maintain accurate and up-to-date student information. The Student Record Management System streamlines record-keeping processes, improving accuracy, accessibility, and productivity for educational institutions.

TABLE OF CONTENTS

CHAPTER 1

1.INTRODUCTION

1.1 INTRODUCTION

The Student Record Management System is a streamlined, database-driven application designed to efficiently manage and maintain student records in educational institutions. The system is developed using Java for its backend logic and MySQL for database management, ensuring seamless and efficient performance. The system automates critical tasks, including adding, updating, deleting, and displaying student information, such as Roll No, Name, Department, and Marks. Java offers a reliable and interactive platform for developing the application's logic, providing a user-friendly interface for administrators to manage student records. MySQL handles the backend, storing and managing student data securely, while ensuring quick and accurate retrieval. This integrated solution improves administrative efficiency by reducing manual work, minimizing errors, and offering real-time insights into student data. The system's simplicity, scalability, and ability to integrate with existing workflows make it an essential tool for educational institutions looking to streamline record-keeping processes.

1.2 OBJECTIVE

The primary objective of the Student Record Management System is to develop an efficient, user-friendly platform that simplifies the management of student records. Specific objectives include:

• Streamline Administrative Tasks: Automate data entry, updating, deletion, and viewing of student records to minimize manual effort and reduce human error.

• Enhance Data Accessibility: Provide administrators with easy access to student information, ensuring quick retrieval of relevant data.

• Improve Data Integrity: Ensure accurate data storage and retrieval, with safeguards against invalid or duplicate data entries.

• Support Scalability: Design the system to handle increasing numbers of students while maintaining high performance and data accuracy.

• Facilitate Real-Time Updates: Offer a dynamic and responsive interface where changes to student records are immediately reflected in the system.

This project aims to integrate Java and MySQL to create a robust and scalable solution that enhances the accuracy, efficiency, and management of student records

1.3 MODULES

The **Student Record Management System** consists of the following key modules:

Student Data Management:

• Stores student details such as Roll No, Name, Department, and Marks.

• Ensures data integrity with proper validation for each entry.

• Allows easy updating and deletion of student records.

Database Management:

• Manages connections to the **MySQL** database for storing and retrieving student records.

• Ensures data consistency and integrity using database constraints like primary keys and foreign keys.

• Handles database transactions, ensuring reliable data operations.

User Interface (UI):

• Provides an intuitive and user-friendly Java Swing interface for administrators.

• Allows for input of student data, displays existing records in a table format, and facilitates operations like adding, updating, and deleting records.

• Includes buttons for adding new students, deleting existing records, and refreshing the table to show updated information.

Student Record Management:

• Displays records stored in the database in a table format with columns: Roll No, Name, Department, and Marks.

• Provides options for searching, deleting, and updating student records.

• Includes buttons like "Add," "Delete," and "Refresh" to manage student data efficiently.

This modular approach ensures that each part of the system works cohesively to provide an efficient and user-friendly student record management experience.

CHAPTER 2

SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The Student Record Management System is a Java-based application designed to streamline the process of managing student records in educational institutions. The system interacts with a MySQL database to securely store and retrieve data related to student information such as Roll No, Name, Department, and Marks. Its goal is to simplify administrative tasks by automating the creation, updating, deletion, and retrieval of student records.

Key Features:

• Add New Student:

Administrators can enter new student data, including Roll No, Name, Department, and Marks, which are then stored in the MySQL database.

• Update and Delete Records:

Allows modifications or deletions of existing student records, ensuring that the data remains up-to-date.

• Display Student Information:

Displays all student records in a tabular format, offering easy access to view and search for specific student details.

• Database Integration:

Uses MySQL for storing student data, ensuring persistence and security.

• Input Validation:

Validates the inputs to ensure data consistency and accuracy, such as checking for duplicate Roll No or incorrect data entries.

Technical Specifications:

• **Programming Language**:

Java (Core Java for backend logic and application interface).

• **Database**:

MySQL for relational database management.

• **Libraries and Tools**:

MySQL JDBC Driver: For database connectivity.

Eclipse IDE: For project development.

SQL Scripts: For database schema creation and data population.

• **System Requirements**:

JDK Version: Java 8 or later.

Database Server: MySQL 5.7 or later.

IDE: Eclipse or IntelliJ IDEA (optional for development).

Workflow:

• **Database Setup**:

Run the provided SQL script to create tables for storing student records and populate sample data.

• **Program Execution:**

Start the Java application, where administrators interact with the system via a graphical user interface (GUI).

Admins can add, update, or delete student records and view them in a table format.

• **Data Handling**:

All data related to students is fetched and updated in real-time in the database. When a new student is added, the record is stored in the students table, and updates are immediately reflected.

2.2 LANGUAGES

Java in Student Record Management System

Java is an object-oriented, platform-independent programming language that provides flexibility and robustness for developing applications in a variety of domains, including database-driven projects like the Student Record Management System.

Role in the Student Record Management System:

• Backend Logic:

Java handles core operations such as adding, updating, and deleting student records. It manages the business logic for handling inputs and updating the database.

• Database Integration:

Java facilitates communication with the MySQL database through JDBC (Java Database Connectivity), enabling smooth execution of SQL queries to interact with the database (inserting, updating, deleting, and retrieving records).

• User Interface (UI):

Java Swing is used to create the graphical user interface for the application, providing an intuitive, interactive experience for administrators to manage student records efficiently.

• Application Development:

Java allows the development of desktop applications that can be deployed on various operating systems. The system supports real-time data updates and ensures that changes made in the UI are reflected in the MySQL database.

By using Java, the Student Record Management System ensures that the backend logic, database operations, and user interface work together seamlessly, providing a reliable and scalable solution for managing student records.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

Functional Requirements:

Student Management:

• Add, update, and delete student details.

• Store student information, including Roll No, Name, Department, and Marks.

Data Operations:

• Provide functionality to display all student records in a tabular format.

• Implement search functionality to find a student using Roll No or Name.

Integration and Database Management:

• Ensure seamless interaction between the Java application and the MySQL database.

• Perform CRUD (Create, Read, Update, Delete) operations efficiently.

Input Validation:

• Validate inputs to prevent duplicate Roll No entries and ensure accurate data integrity.

Non-Functional Requirements:

Performance:

• Ensure quick response time for search and CRUD operations.

Scalability:

• Support increasing student records and allow future enhancements to the system.

Usability:

• Design a simple and intuitive user interface for administrators.

Reliability:

• Guarantee data consistency and handle errors gracefully to avoid disruptions.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

For Server Setup:

Processor:

• Minimum: Quad-core processor (e.g., Intel Core i5, AMD Ryzen 5).

• Recommended: Octa-core processor (e.g., Intel Core i7/i9, AMD Ryzen 7/9).

RAM:

• Minimum: 8 GB.

• Recommended: 16 GB or higher for smooth database performance.

Storage:

• Minimum: 256 GB SSD or 1 TB HDD.

• Recommended: 512 GB SSD for faster data retrieval.

Backup Devices:

• External storage or cloud services for periodic backups.

For Client Systems:

• **Processor**: Dual-core processor or higher.

• **RAM**: Minimum 4 GB.

• **Storage**: 128 GB or higher.

• **Display**: Standard monitors or touchscreens for data input.

• **Input Devices**: Keyboard and mouse for interaction.

Software Requirements

Operating System:

• Server: Windows Server 2019/2022 or Linux (Ubuntu, CentOS).

• Client: Windows 10/11 or any Linux distribution.

Database Management System:

• Preferred: MySQL (open-source, reliable, and community-supported).

Programming Languages:

• Java: For application backend and GUI development.

Development Tools:

• IDE: Eclipse or IntelliJ IDEA for Java development.

• Database Design: MySQL Workbench for schema design and management.

APIs and Libraries:

• JDBC: For database integration between the Java application and MySQL.

By ensuring compatibility between hardware, software, and development tools, this system will deliver reliable and efficient performance for student record management.

CHAPTER 4

PROGRAM CODE

Create the database and table:

```
CREATE DATABASE StudentDB;
USE StudentDB;
CREATE TABLE Students (
    rollNo INT PRIMARY KEY,
    name VARCHAR(100),
    department VARCHAR(50),
    marks DOUBLE
);
```

Student Class: Represents Student Data

```
public class Student {
    private int rollNo;
    private String name;
    private String department;
    private double marks;
        public Student(int rollNo, String name, String department, double marks) {
        this.rollNo = rollNo;
        this.name = name;
        this.department = department;
        this.marks = marks;
    }
    public int getRollNo() {
        return rollNo;
    }
    public String getName() {
        return name;
    }
```

```
    public String getDepartment() {
        return department;
    }
    public double getMarks() {
        return marks;
    }
}
```

StudentDatabase Class: Handles MySQL Database Operations

```
import java.sql.*;
import java.util.ArrayList;

public class StudentDatabase {
    private Connection connection;

    public StudentDatabase() {
        try {
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/StudentDB", "root",
"password");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void addStudent(Student student) {
        try {
            String query = "INSERT INTO Students (rollNo, name, department, marks) VALUES
(?, ?, ?, ?)";
            PreparedStatement stmt = connection.prepareStatement(query);
            stmt.setInt(1, student.getRollNo());
            stmt.setString(2, student.getName());
```

```
        stmt.setString(3, student.getDepartment());
        stmt.setDouble(4, student.getMarks());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public boolean deleteStudent(int rollNo) {
    try {
        String query = "DELETE FROM Students WHERE rollNo = ?";
        PreparedStatement stmt = connection.prepareStatement(query);
        stmt.setInt(1, rollNo);
        return stmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

public ArrayList<Student> getAllStudents() {
    ArrayList<Student> students = new ArrayList<>();
    try {
        String query = "SELECT * FROM Students";
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            students.add(new Student(
                    rs.getInt("rollNo"),
                    rs.getString("name"),
```

```
                rs.getString("department"),
                rs.getDouble("marks")
            ));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return students;
  }
}
```

StudentRecordSystem Class: GUI with Swing

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.ArrayList;

public class StudentRecordSystem extends JFrame {
  private StudentDatabase database;
  private DefaultTableModel tableModel;

  public StudentRecordSystem() {
    database = new StudentDatabase();
    setTitle("Student Record System");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(800, 400);
    setLayout(new BorderLayout());

    tableModel = new DefaultTableModel(new String[]{"Roll No", "Name", "Department",
"Marks"}, 0);
    JTable table = new JTable(tableModel);
    add(new JScrollPane(table), BorderLayout.CENTER);
```

```
JPanel inputPanel = new JPanel(new GridLayout(5, 2));
JTextField rollNoField = new JTextField();
JTextField nameField = new JTextField();
JTextField departmentField = new JTextField();
JTextField marksField = new JTextField();

inputPanel.add(new JLabel("Roll No:"));
inputPanel.add(rollNoField);
inputPanel.add(new JLabel("Name:"));
inputPanel.add(nameField);
inputPanel.add(new JLabel("Department:"));
inputPanel.add(departmentField);
inputPanel.add(new JLabel("Marks:"));
inputPanel.add(marksField);

add(inputPanel, BorderLayout.NORTH);

JPanel buttonPanel = new JPanel();
JButton addButton = new JButton("Add");
JButton deleteButton = new JButton("Delete");
JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);
buttonPanel.add(deleteButton);
buttonPanel.add(refreshButton);
add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(e -> {
    try {
```

```
            int rollNo = Integer.parseInt(rollNoField.getText());
            String name = nameField.getText();
            String department = departmentField.getText();
            double marks = Double.parseDouble(marksField.getText());
            database.addStudent(new Student(rollNo, name, department, marks));
            JOptionPane.showMessageDialog(this, "Student added successfully!");
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Invalid input!");
        }
    });

    deleteButton.addActionListener(e -> {
        try {
            int rollNo = Integer.parseInt(rollNoField.getText());
            if (database.deleteStudent(rollNo)) {
                JOptionPane.showMessageDialog(this, "Student deleted successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Student not found!");
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Invalid input!");
        }
    });

    refreshButton.addActionListener(e -> refreshTable());
    refreshTable();
}

private void refreshTable() {
    tableModel.setRowCount(0);
```

```
        for (Student student : database.getAllStudents()) {
            tableModel.addRow(new Object[]{
                    student.getRollNo(),
                    student.getName(),
                    student.getDepartment(),
                    student.getMarks()
            });
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new StudentRecordSystem().setVisible(true);
        });
    }
}
```

CHAPTER 5

RESULTS AND DISCUSSIONS

Results

The **Student Record Management System** was successfully implemented using Java Swing for the user interface, Java for backend logic, and MySQL for database management. The system achieved the following functionalities:

Student Management

1. Students can be added, updated, deleted, and viewed easily through the GUI.
2. Each student record includes details such as Student ID, name, age, gender, course, and grades.

Course and Grades Management

1. Courses can be added and assigned to students dynamically.
2. Grades can be recorded, updated, and calculated for individual students based on their registered courses.

Search and Filter Options

1. Students can be searched by ID, name, or course.
2. Advanced filters allow sorting students based on grades, gender, or course.

System Integration

1. Smooth integration between the Java Swing application and MySQL database using JDBC.
2. Real-time CRUD operations ensure data integrity and consistency.

Report Generation

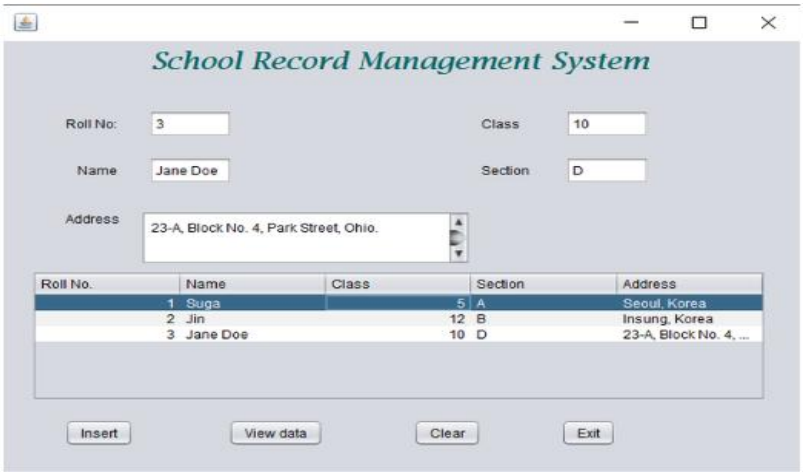
1. Reports such as student performance summaries, course popularity, and grade distribution were generated successfully.
2. Reports can be exported as text or CSV files for further analysis.

User-Friendly Interface

1. The GUI built with Java Swing provided an intuitive and responsive design for easy navigation and usage.

This project demonstrates effective use of **Java Swing**, **JDBC**, and **MySQL** to manage and maintain student records in an educational institution.


School Record Management System
Roll No:
1
Class
5
Name
Suga
Section
A
Address
Seoul, Korea
Roll No.
Name
Address
Message
Record Added Successfully
OK
Insert
View data
Clear
Exit


School Record Management System
Roll No:
3
Class
10
Name
Jane Doe
Section
D
Address
23-A, Block No. 4, Park Street, Ohio.
Roll No.
Name
Class
Section
Address
1 Suga 5 A Seoul, Korea
2 Jin 12 B Insung, Korea
3 Jane Doe 10 D 23-A, Block No. 4, ...
Insert
View data
Clear
Exit

Discussion

Strengths of the Student Record Management System:

Efficiency

1. Automated management of student records, courses, and grades minimized manual errors.
2. Real-time CRUD operations ensured smooth and consistent data handling across all modules.

User-Friendliness

1. The Java Swing-based GUI was intuitive and easy to navigate for administrators and staff.
2. Search and filter features allowed quick access to specific student records or performance data.

Data Security and Integrity

1. Secure login mechanisms ensured only authorized users could access the system.
2. Database integrity was maintained with foreign keys and referential integrity between tables (Student, Course, Grades).

Scalability

1. The system architecture was designed to accommodate increasing numbers of students, courses, and grades without performance degradation.
2. Additional modules, such as attendance or fee management, could be integrated seamlessly.

Customizability

1. The system allowed flexible management of student data to adapt to different institutions' needs.
2. Reports were customizable to focus on specific performance metrics or institutional requirements.

These strengths demonstrate the robustness and versatility of the system, making it a valuable solution for educational institutions.

CHAPTER 6
CONCLUSION

The **Student Record Management System** was developed successfully using **Java Swing** for the user interface and **MySQL** for database management, effectively addressing common challenges in managing educational records. The system automated tasks such as student registration, course management, and grade tracking, resulting in enhanced efficiency, accuracy, and accessibility.

By combining Java's versatile programming capabilities with MySQL's robust relational data handling, the system achieved seamless data flow and real-time updates. Features like advanced search filters, grade management, and performance reporting further streamlined administrative tasks, saving time and reducing errors.

The project demonstrated scalability and adaptability, making it capable of accommodating additional functionalities, such as attendance tracking, fee management, or integration with web and mobile platforms. While challenges like ensuring smooth integration and optimizing large-scale report generation were encountered, they were addressed effectively with modular design and database optimization techniques.

This project underscores the potential of technology in modernizing student record management, providing a reliable and user-friendly platform that can evolve with the growing needs of educational institutions. It lays the groundwork for future enhancements and integrations, ensuring long-term relevance and utility.

CHAPTER 7

REFERENCES

Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th Edition). Pearson Education. Deitel, P. J., & Deitel, H. M. (2018). Java: How to Program (11th Edition). Pearson Education.

Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan: A foundational textbook that covers DBMS design, architecture, and application, with sections relevant to MySQL.

"Murach's MySQL" by Joel Murach: An excellent book for developers that provides practical examples for using MySQL with Java.

"Java Persistence with Hibernate" by Christian Bauer and Gavin King: A comprehensive guide to integrating databases with Java applications.

Reference link for java-swing at javatpoint : https://www.javatpoint.com/java-swing