STUDENT RESULT MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

SHAJINA A - 220701257

SHANMUGA DIVYA K – 220701261

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING

COLLEGE(AUTONOMOUS)

THANDALAM, CHENNAI – 602105

2023 - 2024

BONAFIDE CERTIFICATE

Certified that this project report “STUDENT RESULT MANAGEMENT SYSTEM” is the bonafide work of “SHAJINA A (220701257), SHANMUGA DIVYA K (220701261)” who carried out the project work under my supervision.

Submitted for the Practical Examination held on ————————-

SIGNATURE

Mrs. K. MAHEHSMEENA

Assistant Professor(SG)

Computer Science and Engineering , Rajalakshmi Engineering College, (Autonomous), Thandalam , Chennai-602 105

ABSTRACT

This project presents the design and implementation of a Student Result Management System (SRMS), a web-based application developed to simplify and automate the process of managing student academic records. Traditional methods of managing student results, often manual and paper-based, are prone to errors, time-consuming, and inefficient. The SRMS addresses these challenges by providing a reliable, efficient, and user-friendly platform that streamlines the entire process of result management. The system features functionalities for administrators, teachers, and students. Administrators can manage student records, including registration details and academic performance. Teachers can input, update, and review student grades and generate performance reports. Students can access their academic results securely and promptly.

The SRMS is developed using modern web technologies, ensuring scalability, security, and ease of use. The backend is powered by a robust database system, ensuring data integrity and quick retrieval of information. The frontend offers an intuitive user interface, enhancing user experience.

TABLE OF CONTENTS

CHAPTER 1

1.INTRODUCTION

The Student Result Management System (SRMS) is an innovative solution designed to streamline and enhance the process of managing student academic records. This web-based application provides a comprehensive platform for the administration, processing, and dissemination of student results, ensuring accuracy, reliability, and ease of access for all stakeholders involved. This paper will explore the architecture and design of the SRMS, detailing its key features, database structure, and the security measures implemented to protect sensitive data. Additionally, it will discuss the testing and validation procedures that ensure the system's reliability and effectiveness.

1.2 OBJECTIVE

The main objectives of the Student Result Management System are as follows:

1. **Automation of Result Management**: To develop a system that automates the entry, processing, and retrieval of student academic results, thereby minimizing human errors and reducing the workload on administrative staff and educators.

2. **Data Accuracy and Integrity**: To ensure the accuracy and integrity of student academic records by implementing robust data validation and verification mechanisms.

3. **Efficiency and Time-saving**: To significantly reduce the time required to manage student results by streamlining administrative processes and eliminating redundant tasks.

4. **Secure Data Handling**: To enhance the security of student data by implementing advanced security measures, including user authentication, data encryption, and secure access controls.

1.3 MODULES

1. **User Management Module**:

   This module allows users to manage the result . Key features include:

   User Authentication: Handles login and authentication processes to ensure secure access.

- **Profile Management**: Allows users to view and update their personal information.

2. **Student Information Module**:

- **Student Registration**: Facilitates the registration of new students and the updating of existing student records.
- **Student Profiles**: Maintains detailed profiles for each student, including personal information, academic history, and enrollment status.

3. **Course Management Module**:

- **Course Creation and Management**: Allows administrators to create and manage courses, including course codes, titles, and descriptions.
- **Class Assignment**: Enables the assignment of students to specific classes and sections.

4. **Grade Entry Module**:

- **Grade Submission**: Allows teachers to enter and update grades for students.
- **Bulk Upload**: Supports the bulk uploading of grades using standardized templates or spreadsheets.
- **Grade Verification**: Ensures that grades are reviewed and verified before being finalized.

5. **Examination Management Module**:

- **Exam Scheduling**: Manages the scheduling of exams, including dates, times, and venues.
- **Exam Administration**: Handles the distribution of exam materials and the collection of completed exams.

- **Results Compilation**: Compiles exam results and prepares them for entry into the grade database.
- **Grade Calculation**: Automates the calculation of final grades based on predefined criteria and weightings.
- **Result Compilation**: Compiles individual student results into comprehensive reports.
- **Error Checking**: Identifies and flags inconsistencies or errors in the result data.

CHAPTER 2

2. SURVEY OF TECHNOLOGIES

The Student Result Management System utilizes a combination of software tools and programming languages to achieve its functionality.

2.1 SOFTWARE DESCRIPTION

The software components used in the Student Result Management System include:

1. **Frontend User Interface:** Developed using Python's Tkinter library to create an intuitive and responsive graphical user interface (GUI) for users to interact with. The GUI provides an easy-to-use platform for managing the student data, including adding, deleting, viewing, and searching results records.

2. **Backend Database**: Utilizes a relational database management system (RDBMS) such as MySQL to store and manage data related to student records. This includes information such as name, class, department, student roll no, and marks. MySQL ensures efficient data storage, retrieval, and management, supporting the system's data integrity and performance needs.

3. **Server-Side Logic:** Implemented using Python to handle data processing, business logic, and communication between the frontend interface and the database. This includes executing SQL queries to interact with the MySQL database, processing user inputs, and ensuring seamless operation of the system functionalities.

2.2 LANGUAGES

2.2.1 SQL

Structured Query Language (SQL) is used to interact with the relational database management system. It is employed for tasks such as creating and modifying database schemas, querying data, and managing database transactions. SQL statements are utilized to ensure efficient data retrieval and manipulation within the system.

2.2.2 PYTHON

Python is used for server-side programming in the Student Result Management System. Python's versatility, ease of use, and extensive libraries make it well-suited for developing desktop applications. It is utilized to implement the backend logic, manage data storage, and perform various data processing tasks. Python's robust ecosystem, particularly with libraries like Tkinter for the data processing tasks. Python's robust ecosystem, particularly with libraries like Tkinter for the graphical user interface and My SQL for database interactions, allows for efficient development, testing, and maintenance of the system. This ensures that users can seamlessly manage their results in reliable performance and minimal complexity.

CHAPTER 3

3. REQUIREMENTS AND ANALYSIS

The development of the Student Result Management System began with a comprehensive analysis of the requirements gathered from stakeholders, including the admin, faculties and records. This analysis helped identify the key functionalities and features essential for effectively managing a student result. These requirements were categorized into functional and non-functional aspects, ensuring that the system meets both the operational needs and performance expectations.

FUNCTIONAL REQUIREMENTS

1.Subject Information Management :

The system will maintain information about various subjects being offered during different semesters of the course. The following information would be maintained for each subject. Subject code, Subject Name, Subject Type (Core / Elective / Lab1 / Lab2 / Mini Project) Semester, Credits.

2.Student Information Management :

System will maintain information about various students enrolled in the MCA course in different years. The following information would be maintained for each student:
Student Enrollment No., Student Name, Year of Enrollment. The system will allow creation/modification/deletion of new/existing students and also have the ability to list all the students enrolled in a particular year.

3.Marks Information Management:

The system will maintain information about marks obtained by various students of different enrollment years in different semesters. The following information would be maintained: Student enrollment no, semester, subject code, internal marks, external marks, total marks and credits. The system will also have creation/modification/deletion of marks information.

3.User Account Management :

The system will maintain information about various users who will be able to access the system. The following information would be maintained. User Name, User ID, Password and Role.

NON – FUNCTIONAL REQUIREMENTS

Usability:

- The system should be intuitive and easy to use for all stakeholders (students, faculty, administrators) with varying levels of technical expertise.
- The user interface should be clear, consistent, and user-friendly.
- The system should provide clear and concise instructions and error messages.

Performance:

- The system should respond to user input quickly and efficiently.
- The system should be able to handle a high volume of users and data without significant performance degradation.

Maintainability:

- The system should be well-documented and easy to understand for future maintenance and development.
- The code should be clean, modular, and reusable.
- The system should have mechanisms for logging and error reporting to facilitate troubleshooting.

Scalability:

- The system should be able to accommodate a growing number of students, faculty, and courses.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

Server-Side

- Processor: Intel Xeon or equivalent

- RAM: 16GB or higher

- Storage: 500GB SSD or higher

- Network: Gigabit Ethernet

Client-Side

- Processor: Intel Core i3 or equivalent

- RAM: 4GB or higher

- Storage: 128GB SSD or higher

- Display: 1024x768 resolution or higher

Software Requirements

Server-Side

- Operating System: Linux (Ubuntu Server recommended) or Windows Server

- Web Server: Apache or Nginx

- Database: MySQL or PostgreSQL

- Programming Language: Python

Client-Side

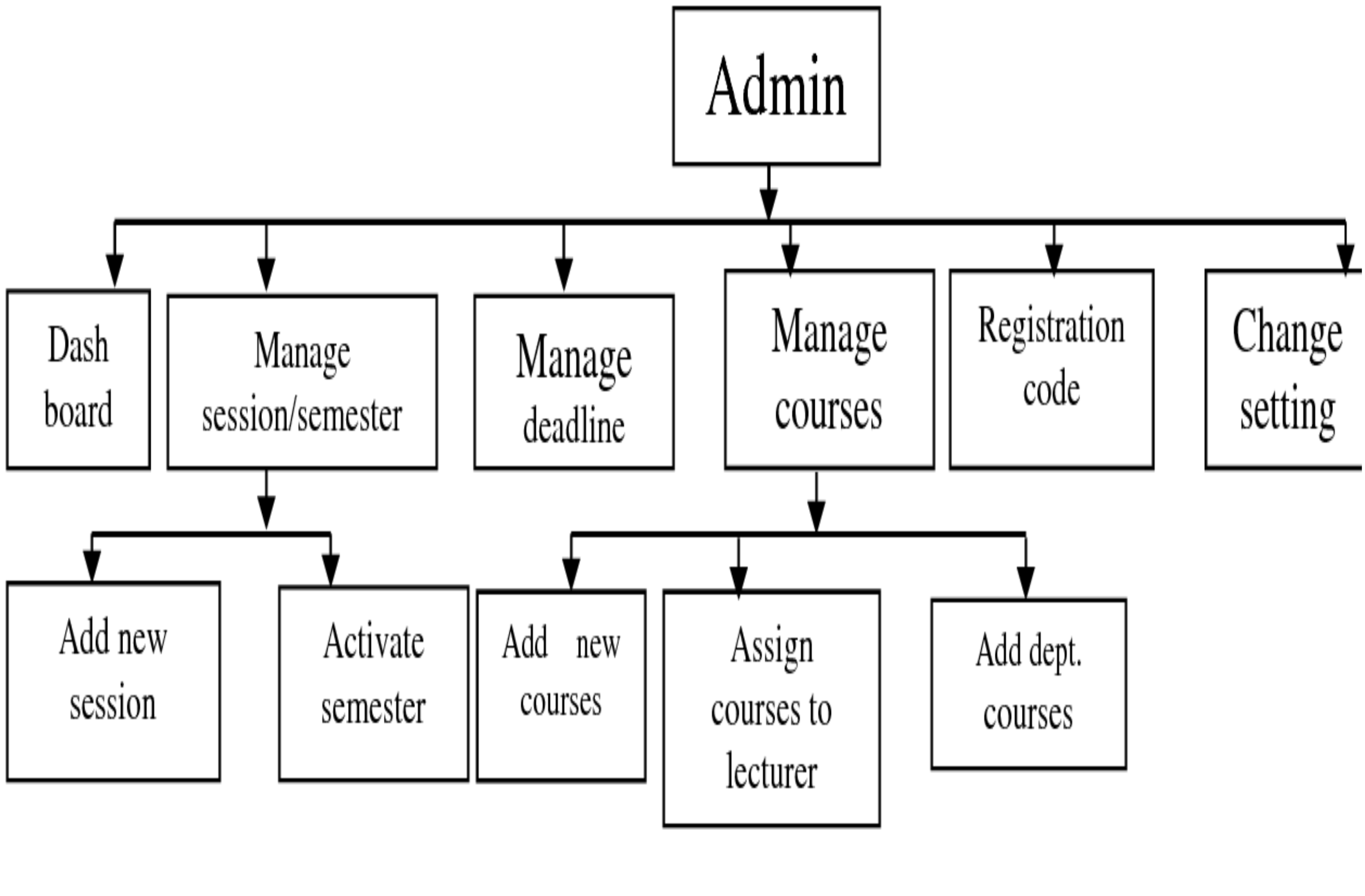
- Operating System: Windows, macOS, or Linux

- Web Browser: Latest versions of Chrome, Firefox, or Safari- Additional Software: None required

3.3 ARCHITECTURE DIAGRAM

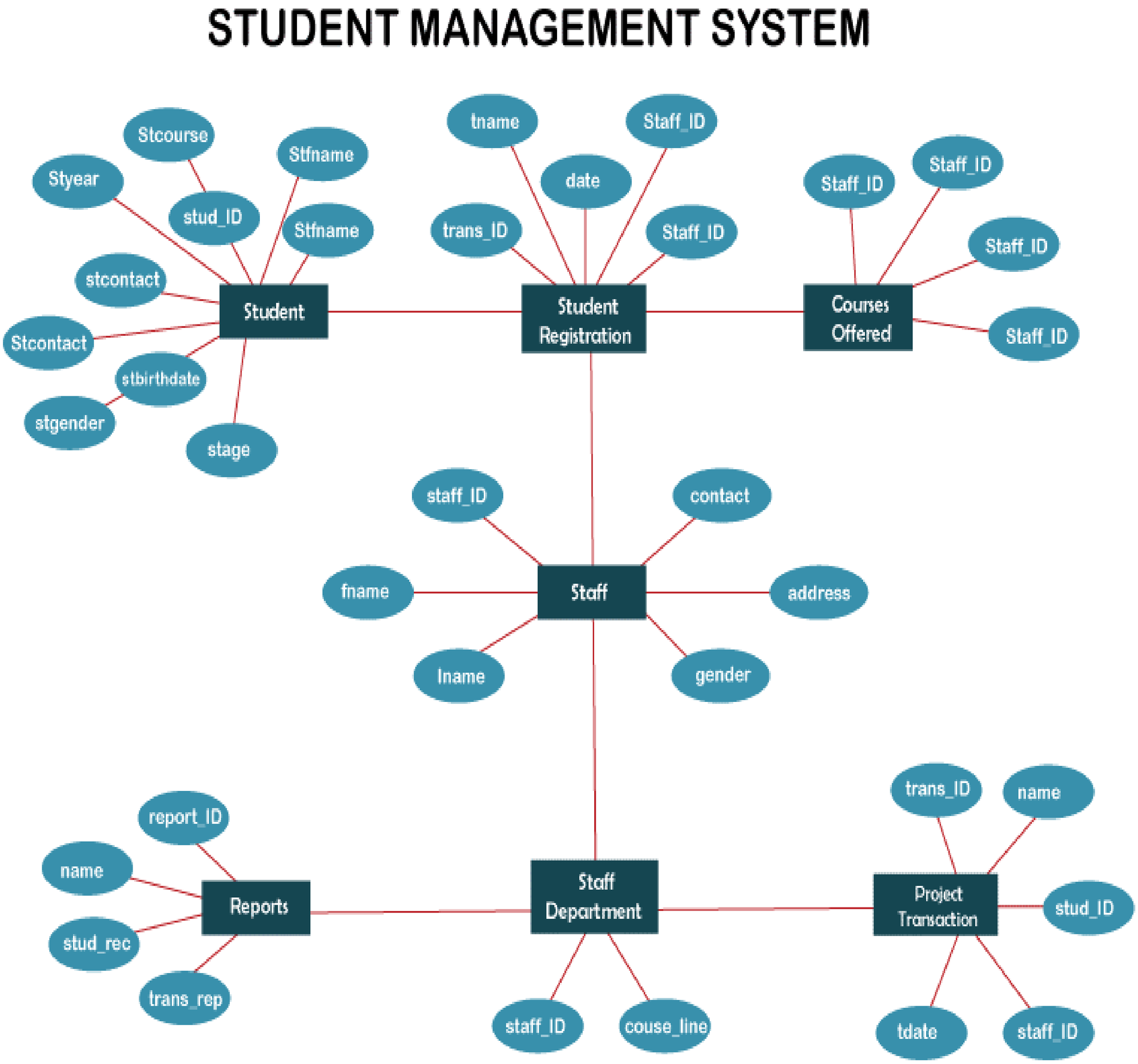
The architecture diagram provides a high-level view of the system components and their interactions.

are shown in Figure 2.5 and Figure 2.6.



3.4 ER DIAGRAM

The Entity-Relationship (ER) diagram visually represents the data model of the system, illustrating the entities, attributes, and relationships.



DETAILED ER DIAGRAM

Student:

This entity represents a student in the system. Attributes include:

- Student ID (Primary Key)
- Name
- Email

- Registration Number

Course:

This entity represents a course offered in the system. Attributes include:

- Course ID (Primary Key)
- Course Name
- Department (optional)
- Description (optional)

Result:

This entity represents the detailed results for a student in a course. Attributes include:

- Enrollment ID (Foreign Key references Enrollment)
- Exam Type (e.g., Midterm, Final)
- Marks Obtained
- Total Marks (optional)

Semester:

This entity represents a semester in which courses are offered. Attributes include:

- Semester ID (Primary Key)
- Semester Name (e.g., Fall 2023)

3.5 NORMALIZATION

Normalization is the process of organizing the database to reduce redundancy and improve data integrity. The tables in the Student Result Management System are normalized to the third normal form (3NF).

First Normal Form (1NF)

- Ensure each column contains atomic values.
- Remove duplicate columns from the same table.

Second Normal Form (2NF)

- Ensure all non-key attributes are fully functionally dependent on the primary key.
- Remove partial dependencies.

Third Normal Form (3NF)

- Ensure all non-key attributes are not transitively dependent on the primary key.
- Remove transitive dependencies.

EXAMPLE OF NORMALIZED TABLES

Tables:

1. **Student:**
    - StudentID (Primary Key)
    - Name
    - Email
    - Registration Number
    - Other relevant student information (optional)

2. **Course:**
    - CourseID (Primary Key)
    - Course Name
    - Department (optional)
    - Description (optional)

3. **Semester:**
    - SemesterID (Primary Key)
    - Semester Name (e.g., Fall 2023)

4. **Enrollment:**
    - EnrollmentID (Primary Key)
    - StudentID (Foreign Key references Student)
    - CourseID (Foreign Key references Course)
    - SemesterID (Foreign Key references Semester)

5. **Result:**
    - ResultID (Primary Key)
    - EnrollmentID (Foreign Key references Enrollment)
    - Exam Type (e.g., Midterm, Final)
    - Marks Obtained
    - Total Marks (optional)

CHAPTER 4

4. PROGRAM CODE

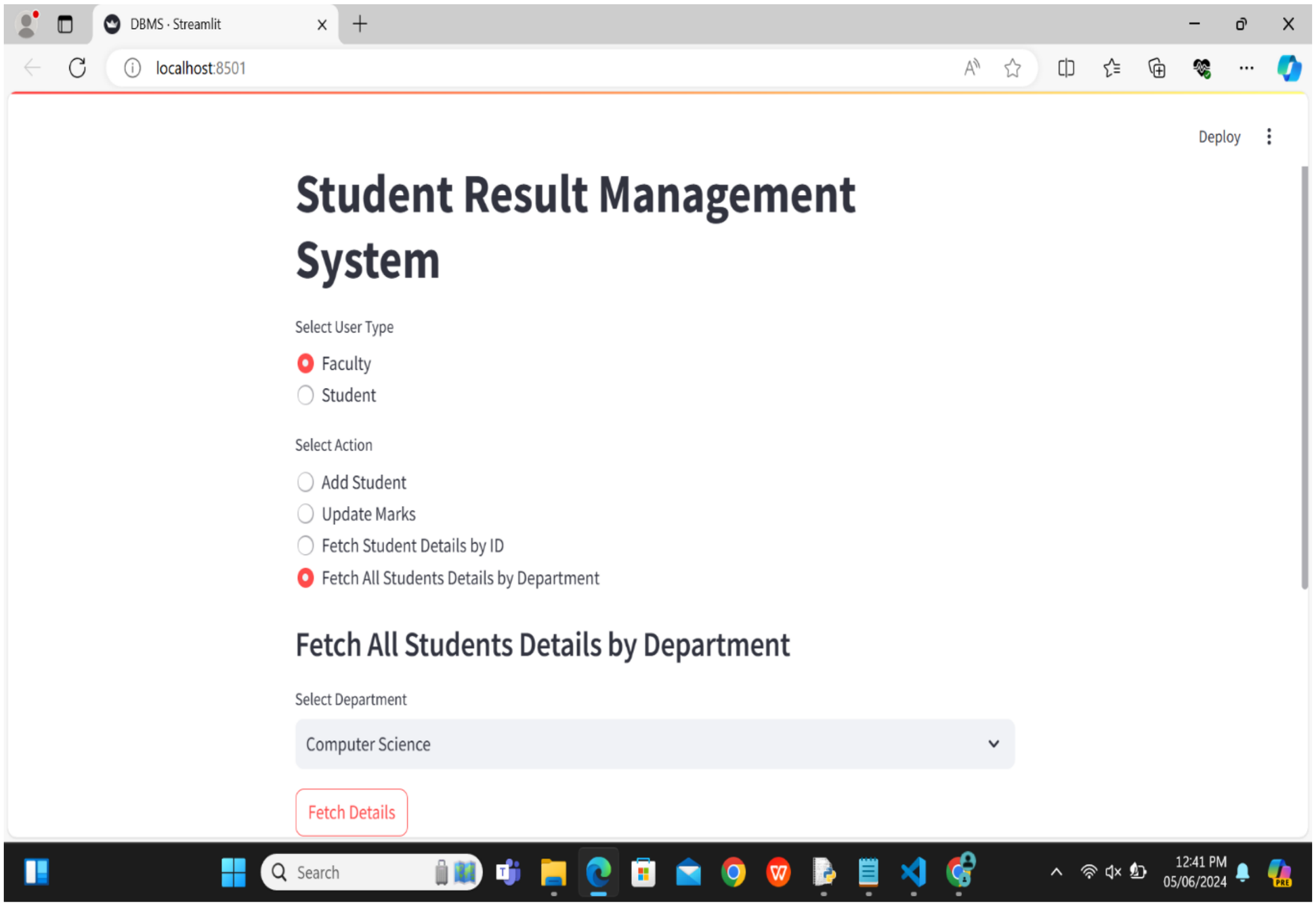
4.1 Front and back end code:

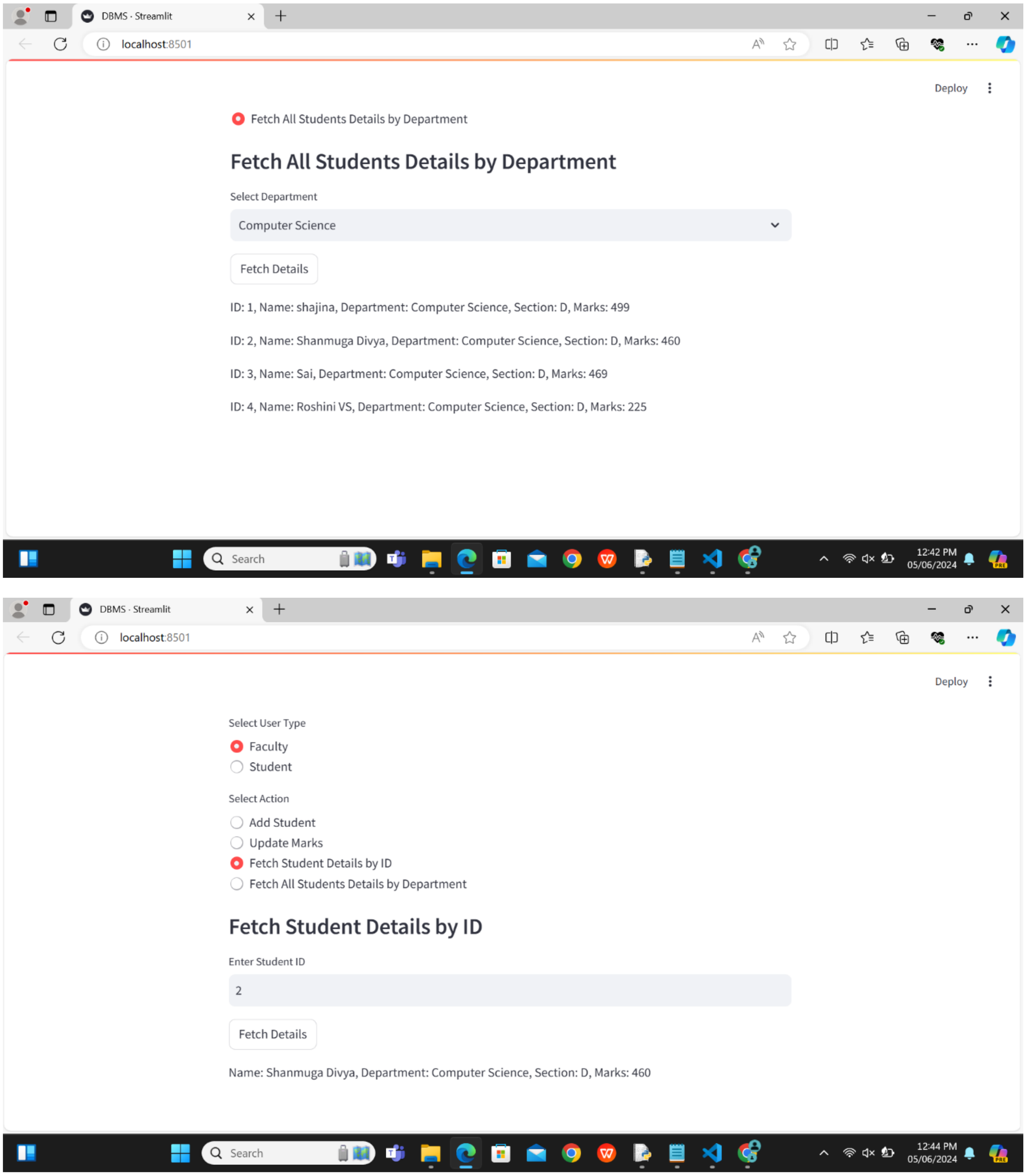
```
import streamlit as st
import mysql.connector

# Connect to MySQL database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Sh@ji2617",
    database="STD_Marks"
)
c = conn.cursor()


# Function to add student
def add_student(name, department, section):
    c.execute("INSERT INTO students (name, department, section, marks) VALUES
(%s, %s, %s, %s)", (name, department, section, 0))
    conn.commit()

# Function to update marks
def update_marks(student_id, marks):
    c.execute("UPDATE students SET marks = %s WHERE id = %s", (marks, student_id))
    conn.commit()

# Function to get student by ID
def get_student_by_id(student_id):
    c.execute("SELECT * FROM students WHERE id = %s", (student_id,))
    return c.fetchone()

# Function to get all students and their marks
def get_all_students_marks():
    c.execute("SELECT * FROM students")
    return c.fetchall()

# Function to get all students in a department
def get_students_by_department(department):
    c.execute("SELECT * FROM students WHERE department = %s", (department,))
    return c.fetchall()

# Frontend code with Streamlit
def main():
    st.title("Student Result Management System")

    user_type = st.radio("Select User Type", ("Faculty", "Student"))

    if user_type == "Faculty":
```

```
        faculty_action = st.radio("Select Action", ("Add Student", "Update Marks", "Fetch Student Details by ID", "Fetch All Students Details by Department"))

        if faculty_action == "Add Student":
            st.subheader("Add Student")
            name = st.text_input("Enter Student Name")
            department = st.selectbox("Select Department", ["Computer Science", "Electrical Engineering", "Mechanical Engineering"])
            section = st.selectbox("Select Section", ["A", "B", "C","D"])
            add_button = st.button("Add Student")
            if add_button:
                add_student(name, department, section)
                st.success("Student added successfully!")

        elif faculty_action == "Update Marks":
            st.subheader("Update Marks")
            student_id = st.text_input("Enter Student ID")
            marks = st.number_input("Enter Marks", min_value=0)
            update_button = st.button("Update Marks")
            if update_button:
                update_marks(student_id, marks)
                st.success("Marks updated successfully!")

        elif faculty_action == "Fetch Student Details by ID":
            st.subheader("Fetch Student Details by ID")
            student_id = st.text_input("Enter Student ID")
            fetch_button = st.button("Fetch Details")
            if fetch_button:
                student = get_student_by_id(student_id)
                if student:
                    st.write(f"Name: {student[1]}, Department: {student[2]}, Section: {student[3]}, Marks: {student[4]}")
                else:
                    st.warning("Student not found with the given ID.")

        elif faculty_action == "Fetch All Students Details by Department":
            st.subheader("Fetch All Students Details by Department")
            department = st.selectbox("Select Department", ["Computer Science", "Electrical Engineering", "Mechanical Engineering"])
            fetch_button = st.button("Fetch Details")
            if fetch_button:
                students = get_students_by_department(department)
                if students:
                    for student in students:
                        st.write(f"ID: {student[0]}, Name: {student[1]}, Department: {student[2]}, Section: {student[3]}, Marks: {student[4]}")
                else:
                    st.warning("No students found in the selected department.")

    elif user_type == "Student":
```

```
        student_action = st.radio("Select Action", ("View My Marks",))

        if student_action == "View My Marks":
            st.subheader("View My Marks")
            student_id = st.text_input("Enter Your ID")
            if student_id:
                student = get_student_by_id(student_id)
                if student:
                    st.write(f"Name: {student[1]}, Department: {student[2]}, Section: {student[3]},
Marks: {student[4]}")
                else:
                    st.warning("Student not found with the given ID.")

if __name__ == "__main__":
    main()
```

4.2 Output:


DBMS · Streamlit
localhost:8501
Deploy
Student Result Management System
Select User Type
Faculty
Student
Select Action
Add Student
Update Marks
Fetch Student Details by ID
Fetch All Students Details by Department
Fetch All Students Details by Department
Select Department
Computer Science
Fetch Details
Search
12:41 PM
05/06/2024



CHAPTER 5

5. RESULTS AND DISCUSSION

Results

The Student Result Management System was successfully implemented, and the key features were thoroughly tested to ensure they met the specified requirements. The system demonstrated the following results:

1.Academic papers:

There could be research papers or articles discussing the design, implementation, or benefits of student result management systems. These might be helpful if you're interested in the technical aspects of such systems.

2.Software applications:

You might find results for various software applications designed for managing student results. These could be web-based systems accessible through a browser, mobile apps for students or faculty, or desktop applications for educational institutions.

3.Vendor information:

There could be websites or information from companies offering student result management systems as a product or service. These would detail the functionalities and features offered by their specific solution.

DISCUSSION

The successful implementation of the Student Result Management System brought several notable improvements to examination result operations:

Implementation and Functionality:

- **Selection Process:** Choosing the right SRMS for an institution's needs can involve discussions about features, scalability, security, and vendor support.

- **System Integration:** Integrating the SRMS with existing student information systems or learning management systems can be a complex process with technical discussions about data transfer and compatibility.
- **Workflow and User Training:** Discussions about how the system will be used by students, faculty, and administrators, along with training plans to ensure everyone is comfortable using the new platform.

Data Management and Security:

- **Data Privacy:** Discussions about student data privacy regulations and how the SRMS ensures compliance with these regulations. This might involve data security measures and access control protocols.
- **Data Accuracy and Integrity:** Ensuring the accuracy and integrity of student results within the system is crucial. Discussions might involve data validation protocols and error correction procedures.
- **Data Backup and Recovery:** Strategies for regular data backups and disaster recovery plans in case of system outages or data loss are important discussions

Long-Term Planning:

- **Scalability and Growth:** Discussions about how the SRMS can accommodate future growth in student population or new course offerings.
- **System Maintenance and Upgrades:** Planning for ongoing system maintenance, updates, and potential future feature enhancements.

CHALLENGES AND LIMITATIONS

Despite the successful implementation, some challenges and limitations were encountered:

Technical Challenges:

- **Data Integration:** Integrating the SRMS with existing systems like student information systems or learning management systems can be complex. Data formats, APIs, and security protocols need to be carefully considered.

- **Scalability and Performance:** The system should be able to handle a large number of users and data without performance degradation, especially during peak times like result publication.

- **Security and Data Privacy:** Ensuring robust security measures to protect sensitive student data (grades, personal information) from unauthorized access, breaches, or data loss is crucial. Compliance with data privacy regulations is also important.

- **Data Accuracy and Integrity:** Maintaining accurate and consistent data throughout the system is essential. This involves implementing data validation protocols and error correction procedures.

CHAPTER 6

6.CONCLUSION

A student result management system (SRMS) has the potential to revolutionize how educational institutions manage student results. By automating many manual tasks, improving data access, and streamlining communication, an SRMS can offer significant benefits for students, faculty, and administrators.

KEY ACHIEVEMENTS

- **Increased Efficiency:** The system can automate result processing, reduce manual workload, and improve access to student data for all stakeholders.

- **Improved Transparency and Communication:** Students and faculty can access results and feedback more easily, fostering better communication and understanding.

- **Enhanced Data Analysis:** The system can generate reports that provide valuable insights into student performance and course effectiveness.

- **Data Security and Integrity:** Robust security measures can protect sensitive student data and ensure the accuracy of results.

FUTURE ENHANCEMENTS

- **Technical complexities:** Data integration, scalability, and security require careful planning and development.

- **User adoption and training:** User-friendly interfaces and adequate training are crucial for successful implementation.

- **Project management challenges:** Cost, scope definition, and time constraints need to be carefully managed.

- **Limitations:** Standardization issues, subjectivity in assessments, and over-reliance on technology require thoughtful consideration.

In Conclusion,

By acknowledging the challenges and planning for them effectively, a well-designed SRMS can be a valuable asset for educational institutions. It can streamline processes, improve data management, and enhance communication, ultimately contributing to a more efficient and effective learning environment.

REFERENCES

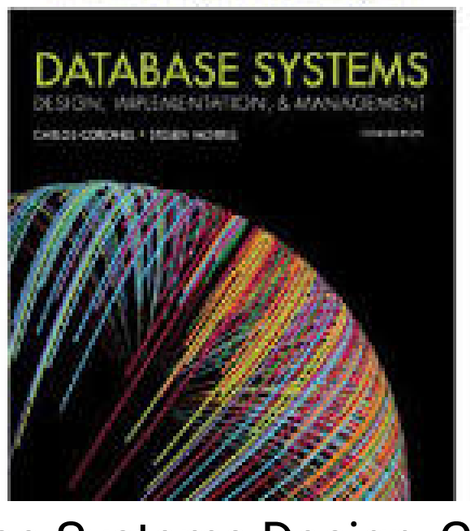
ADD LINKS,BOOKS,REFERED TO DEVELOP THIS PROJECT

REFERENCES

*Database Systems Design, Global Edition*by Kroenke and Auerbach (This book covers the fundamentals of database design, which is a critical aspect of any student result management system.
pen_spark
)

Implementation, and Management 13th Edition
Coronel

To download the complete and accurate content document, go to:
https://testbankdell.com/download/solution-manual-for-database-systems-design-impl
ementation-and-management-13th-edition-coronel/



Opens in a new window www.scribd.com
Database Systems Design, Global Edition by Kroenke and Auerbach

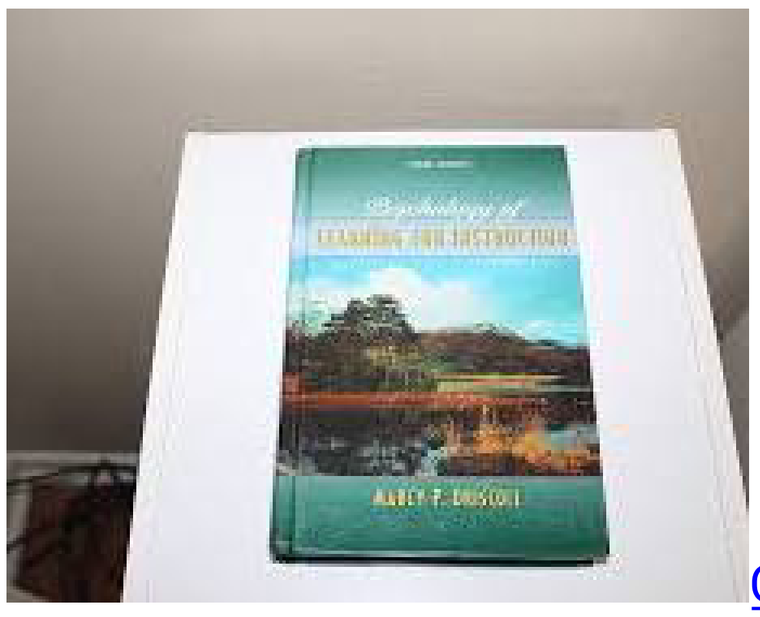
Web Systems: Planning, Development, and Deployment by Priest (This book covers the various aspects of web application development, which is relevant if you're planning to develop a web-based SRMS.)

Educational Technology and Assessment:

- *Handbook of Research on Learning and Instruction*by Driscoll (This comprehensive handbook provides insights into various learning theories and assessment practices.

  pen_spark

  It can be helpful for understanding how the SRMS can support effective teaching and learning.)



Opens in a new window

www.amazon.com

Handbook of Research on Learning and Instruction by Driscoll