A FIELD PROJECT REPORT ON

SEMESTER RESULTS
Submitted

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Jahnavi	(231FA04460)
Bhavana	(231FA04466)
Srikanth	(231FA04470)
Vaibhav	(231FA04505)
Ahalya	(231FA04540)

Under the Guidance of
Mr.G.Murali
Assistant Professor, CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTER AND INFORMATICS

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University)Vadlamudi, Guntur -522213, INDIA

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project entitled *"Semester Results"* is being submitted by **[Jahnavi] [231FA04460]**, **[Bhavana] [231FA04466]**, **[Srikanth] [231FA04470]**, and **[Vaibhav] [231FA04505],[Ahalya] [231FA04540]** in partial fulfilment of the requirements for the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering** at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

Guide

Project Review Committee **HOD, CSE**

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

Date:

We hereby declare that the work presented in the field project titled "Semester Results" is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Mr.G.Murali, Assistant Professor,CSE** in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

Jahnavi	(231FA04460)	Signature
Bhavana	(231FA04466)	Signature
Srikanth	(231FA04470)	Signature
Vaibhav	(231FA04505)	Signature
Ahalya	(231FA04540)	Signature

TABLE OF CONTENTS

1.ABOUT

1.1 ABSTRACT:-

This project focuses on designing an interactive web-based student details form to collect essential student information, such as registration number, name, course, and batch, which then allows students to check their results. The form is developed using HTML and CSS to ensure a user-friendly interface and a responsive design that adapts to different screen sizes. The project also integrates a results page displaying semester-wise availability and links for students to access their results based on their course and batch. The main goal is to create an efficient, accessible, and visually appealing solution for displaying academic results. In addition to the form's functional design, the project explores best practices in web form design, user experience (UX), and survey methodologies for gathering feedback to refine the system. Through a literature survey, the project incorporates modern web development trends and user-centered design principles.

1.2) INTRODUCTION

The focus of this project is the development of a Student Details Form using HTML. This form is designed to collect basic yet important student information such as registration number, full name, course, and batch. It acts as a front-end interface that simplifies the data collection process, making it easier for students to input their academic details accurately.

The main goal of the form is to facilitate students in accessing their academic results or related information. By submitting the necessary details, students can initiate processes such as result checking, course verification, or administrative follow-ups. The form serves as a bridge between the student and the academic system, ensuring that key details are gathered in a standardized format.

This project demonstrates the practical application of HTML in web development, particularly for form creation. It uses various HTML input elements like text fields, dropdowns, and buttons to structure the user interface. Each field is carefully selected to capture essential information, ensuring relevance and clarity for both the user and the administrator.

Client-side validation is an important aspect of this form. Using HTML5 features like the required attribute and input type restrictions (e.g., number, text), the form ensures that users cannot submit incomplete or improperly formatted data. This validation helps in minimizing errors and improving the quality of the information submitted.

User experience has also been given priority in the design of the form. A clean and responsive layout ensures that the form is accessible across different devices. Proper labeling, field alignment, and feedback messages enhance usability, making the form intuitive and straightforward for students to navigate and complete.

In conclusion, the Student Details Form project highlights the usefulness of HTML for building interactive and functional web forms. It not only covers technical implementation but also emphasizes user-centered design principles. The result is a reliable and efficient tool that supports academic data collection with ease and precision.

1.3) PROBLEM DEFINITON

In many academic institutions, the process of collecting student information for result checking or administrative purposes is often manual, time-consuming, and prone to errors. Without a structured and user-friendly system in place, students may face difficulties in submitting their details accurately, leading to delays and mismanagement of records. This lack of a streamlined data collection method can hinder the efficiency of academic operations and negatively impact the overall student experience.

To address this issue, there is a need for a simple, accessible, and well-validated web-based form that allows students to submit their essential details—such as registration number, name, course, and batch—quickly and accurately. By leveraging HTML and basic client-side validation techniques, a digital Student Details Form can reduce errors, save time, and improve the reliability of data collection. This solution aims to enhance the communication between students and academic systems while ensuring the process remains intuitive and efficient.

1.4) EXISTING SYSTEMS

Currently, several academic systems and platforms assist students in accessing their academic information, including semester results, course details, and personal records. Some popular ones include:

- University Portals and ERP Systems – Most universities have their own portals (like Student Information Systems or Enterprise Resource Planning platforms) where students can log in to check their semester results, exam schedules, and course registrations.
- National Education Platforms – Platforms like DigiLocker (India) or NSR (National Student

Records) provide access to academic records, certificates, and result data in a secure and verified digital format.

- Email/Manual Notifications – Some institutions still rely on manual methods such as emailing results, publishing PDFs on official websites, or displaying results on campus notice boards.
- Mobile Apps – A few universities have mobile applications that allow students to check grades, attendance, and academic calendars, providing a more accessible experience.

Limitations of the Existing System

- Complex Interfaces
- Slow Performance and Downtime
- Lack of Simplicity
- Limited Accessibility
- Technical Barriers
- Dependence on Administrative Update

1.5) PROPOSED SYSTEM

The proposed system is a simple and user-friendly Student Details Form built using HTML. It is designed to collect essential information from students such as registration number, name, course, and batch. This form serves as a lightweight and accessible solution for students to submit their details in order to check semester results or receive academic-related updates.This system eliminates the need for complex logins or multi-step processes, offering a direct and efficient way to input and validate required data. By incorporating client-side validation through HTML5 features, it ensures that all mandatory fields are properly filled out before submission, minimizing errors and improving data accuracy.Additionally, the form is designed with a clean and responsive layout, making it accessible across various devices, including smartphones and tablets. This enhances the user experience and ensures that students can access the form from anywhere, at any time.The proposed system can be easily integrated into existing institutional websites or portals, acting as a supplementary tool for quick data collection. It simplifies the process of result checking and academic communication without the overhead of full-fledged systems.

Benefits:

- Simplicity and Ease of Use
- Faster Data Collection
- Client-Side Validation
- Responsive Design
- Lightweight and Efficient
- Easy Integration
- Improved Student Experience

1.6) LITERATURE REVIEW

Several studies and developments in educational technology have explored the use of digital systems to manage and deliver academic information such as semester results. Traditionally, institutions relied on manual processes, including printed result sheets and notice boards. However, these methods were time-consuming and lacked security and accessibility. With the growth of the internet and web-based technologies, there has been a shift toward automated result management systems.

2.SYSTEM REQUIREMENTS

2.1) HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS :

Minimum Requirements:

- Processor : Intel Core i3
- RAM: 4GB
- Storage: 20GB free space
- Internet: Required for online access

Recommended Requirements:

- Processor: Intel Core i5/i7
- RAM: 8GB+
- Storage: 50GB+
- Internet: High-speed connection for smooth browsing and bookings

SOFTWARE REQUIREMENTS

- Operating System: Windows 10, macOS, or Linux
- Database: MySQL or PostgreSQL (to store user and travel data)
- Backend: Java, Python (to manage website functions)
- Frontend: HTML, CSS, JavaScript (for website design and interaction)
- Browser: Google Chrome, Firefox, or Safari (latest versions

2.2) SOFTWARE REQUIREMENTS SPECIFICATIONS

FUNCTIONAL REQUIREMENTS :

- **Student Data Entry**
  The system must allow students to enter essential details such as registration number, name, course, and batch.
- **Form Validation**
  The system should validate all input fields to ensure that no required field is left blank and that the correct format is used (e.g., numeric registration number).
- **Result Retrieval**
  After submitting valid information, the system should retrieve and display the corresponding semester results from the database or predefined source.
- **Responsive User Interface**
  The form and result display should be accessible and properly viewable on all devices including desktops, tablets, and smartphones.
- **Error Handling**
  The system should display appropriate error messages when incorrect or incomplete data is entered, or if the result is not found.
- **Security of Student Data**
  The system must ensure that student information is securely handled and protected from unauthorized access.
- **Admin Panel (Optional)**
  An optional admin interface may be provided to update student records, results, or manage user access.
- **Download/Print Option**
  Students should be able to download or print their semester results for official or personal use.

NON-FUNCTIONAL REQUIREMENTS:

1. **Usability**

The system should have a simple and intuitive interface that is easy for students and administrators to navigate.

2. **Performance**

The system must respond quickly to user inputs and load result data without noticeable delays, even during high-traffic periods.

3. **Scalability**

The system should be scalable to handle increasing numbers of student submissions and result queries as the institution grows.

4. **Reliability**

The system should function consistently without crashes or errors, ensuring that results are always accessible when needed.

5. **Security**

Student information and results must be protected through secure data handling, encryption (if applicable), and restricted access.

6. **Compatibility**

The system should work across different browsers (Chrome, Firefox, Edge, etc.) and operating systems (Windows, macOS, Android, iOS).

7. **Maintainability**

The system should be designed in a modular way to allow for easy updates, bug fixes, and feature enhancements.

8. **Availability**

The system should be available 24/7 with minimal downtime, especially during peak times like result publication days.

3.SYSTEM DESIGN

3.1 Modules of Systems:

1. Student Information Module

Collects basic student details such as registration number, name, course, and batch through a user-friendly input form.

2. Validation Module

Performs client-side validation to ensure that all required fields are filled out correctly before submission.

3. Result Processing Module

Matches the submitted student data with stored records and fetches the corresponding semester result.

4. Result Display Module

Displays the retrieved result in a clear and readable format, including subject-wise marks, total score, and grade (if applicable).

5. Error Handling Module

Provides meaningful error messages in case of invalid input, missing records, or system issues.

6. Admin Management Module (Optional)

Allows authorized personnel to update student records, manage results, and handle backend operations securely.

7. Download/Print Module

Offers functionality for students to download or print their semester results for personal or official use.

8. Security and Access Control Module

Ensures only authorized access to sensitive student and result data, protecting against unauthorized use or data breacheSs.

3.2 UML Diagrams

1. **Use Case Diagram** :- A Use Case Diagram is a behavioral UML diagram that represents the interactions between users (actors) and a system. It visually depicts the system's functionality by illustrating various use cases, actors, and their relationships.

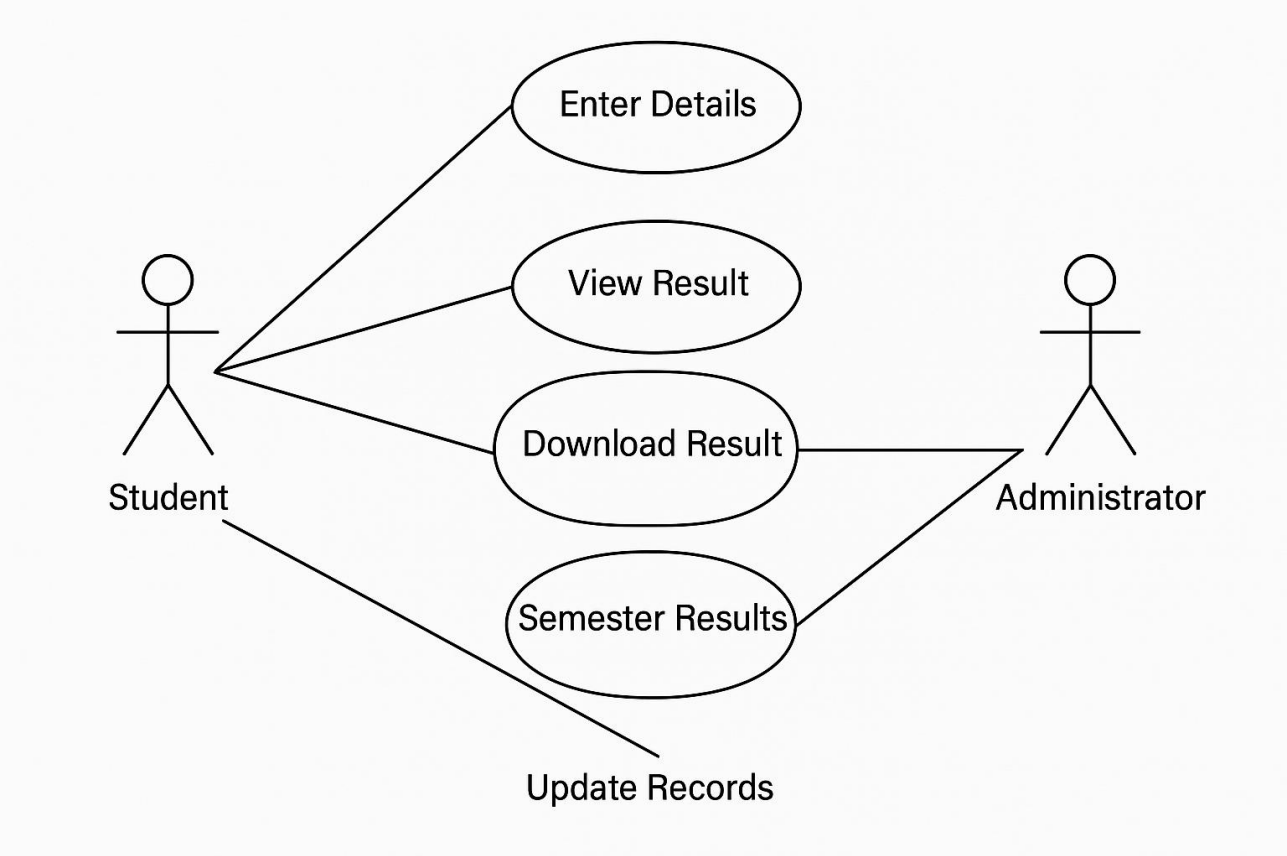


FIG 3.2.1(use case diagram)

2.Activity Diagram:-An Activity Diagram is a behavioral UML diagram that represents the flow of activities within a system. It visually depicts the step-by-step execution of processes, showing the sequence of actions, decision points, parallel processes, and loops.

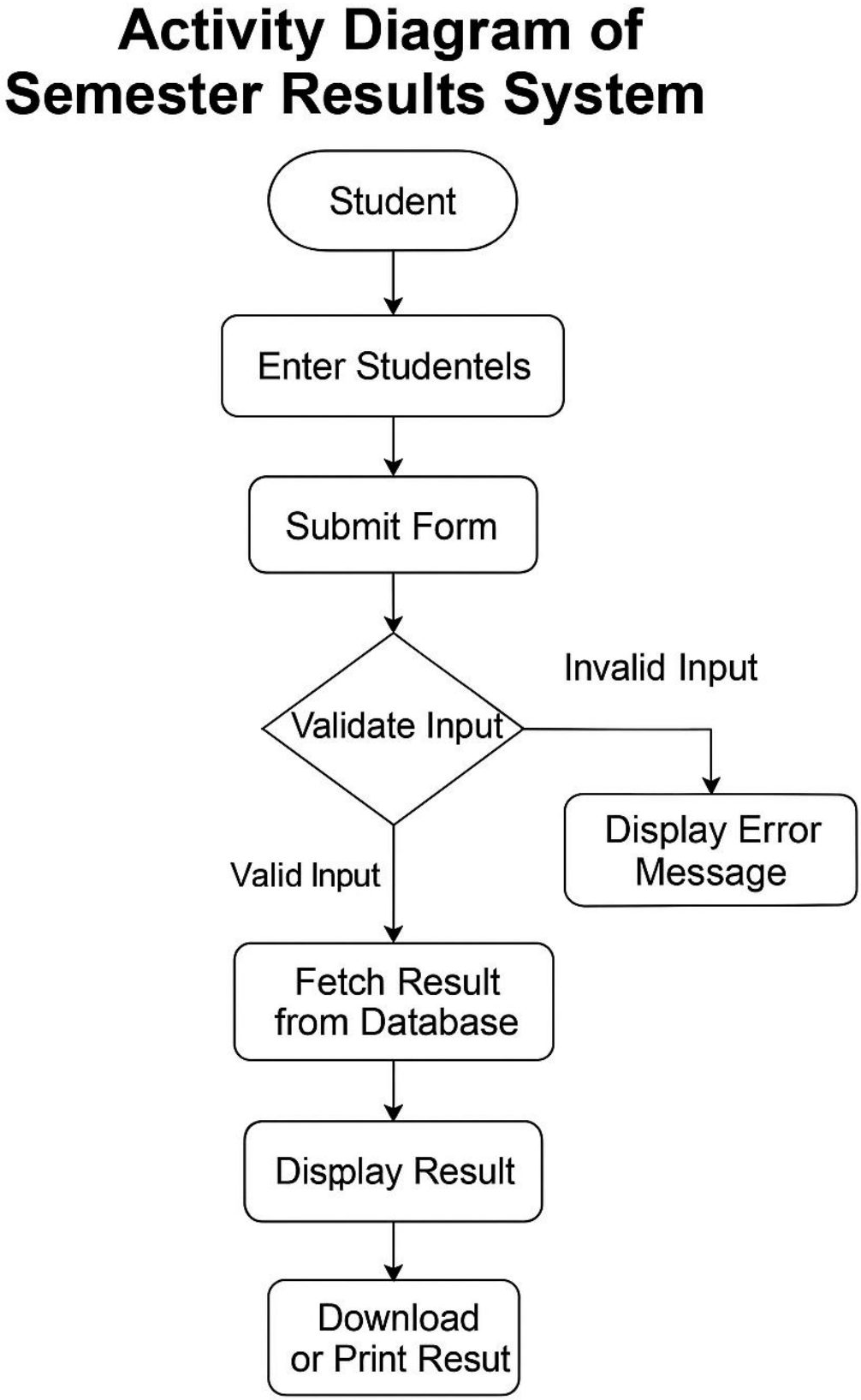


FIG 3.2.2(Activity diagram)

3.Class Diagram :-

A Class Diagram is a structural UML diagram that represents the static structure of a system by showing its classes, attributes, methods, and relationships among objects. It is widely used in object-oriented design to define the blueprint of a system.

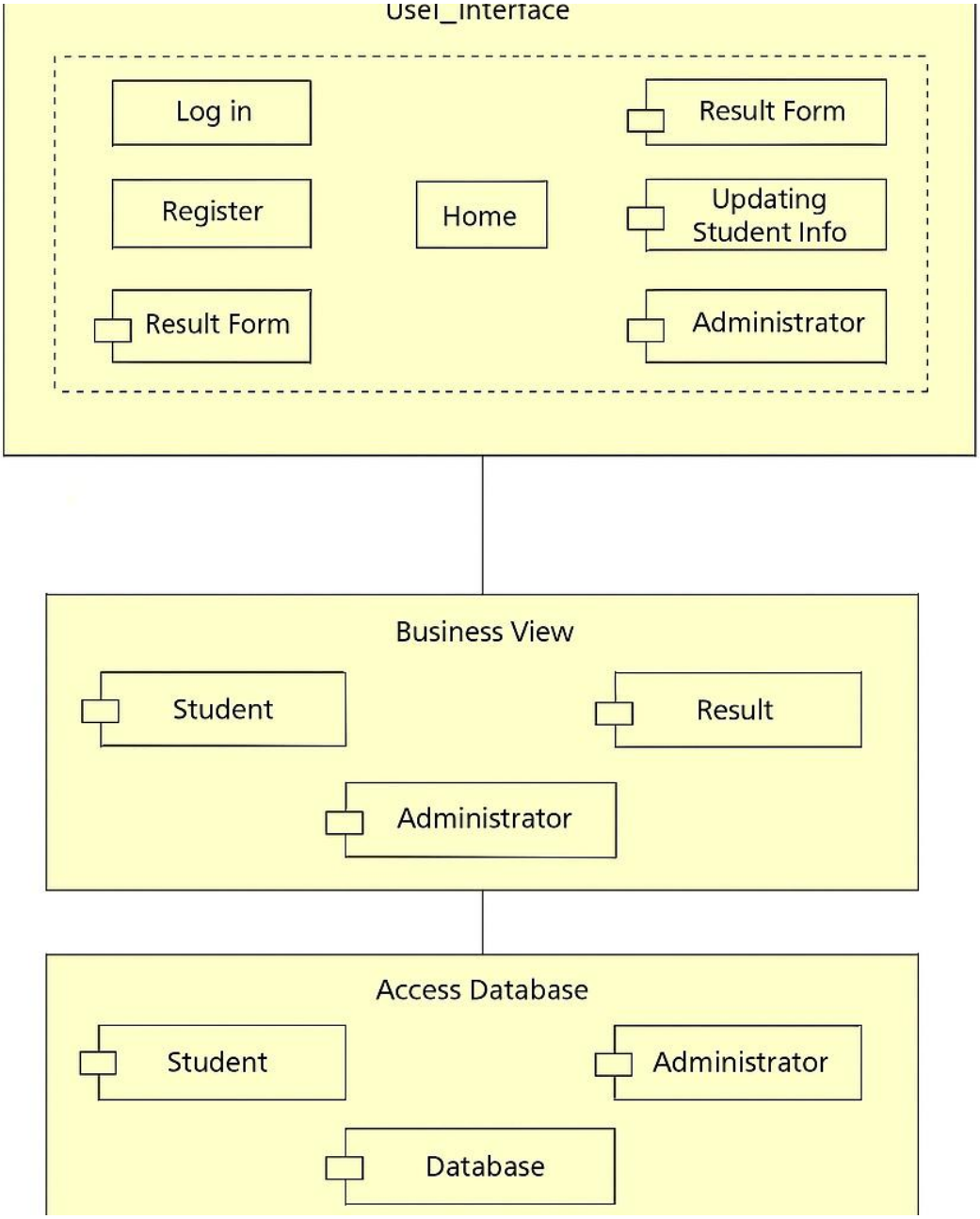


FIG 3.2.3(class diagram)

4. IMPLEMENTATION

4.1 Sample Codes

1.Login page

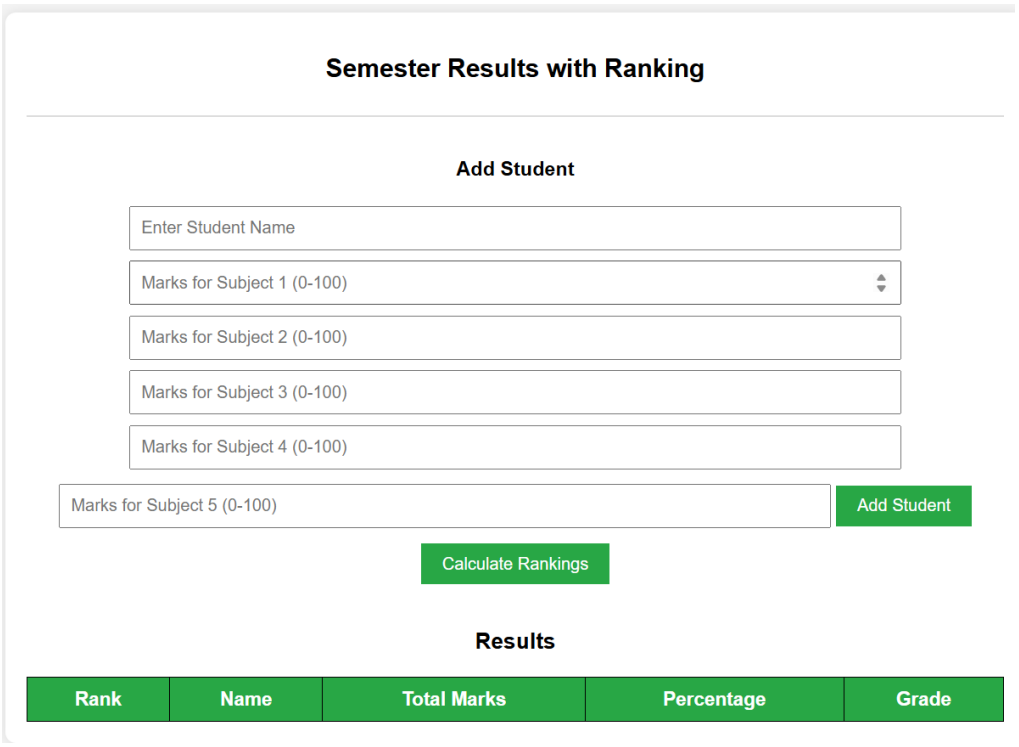
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Semester Results with Ranking</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         text-align: center;
         background-color: #f4f4f4;
         margin: 0;
         padding: 0;
      }
      .container {
         width: 60%;
         margin: auto;
         background: white;
         padding: 20px;
         box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
         border-radius: 10px;
         margin-top: 30px;
      }
      input {
         width: 80%;
         padding: 10px;
         margin: 5px 0;
         font-size: 16px;
      }
      button {
         padding: 10px 20px;
         font-size: 16px;
         background-color: #28a745;
         color: white;
         border: none;
         cursor: pointer;
         margin-top: 10px;
      }
      button:hover {
         background-color: #218838;
      }
      .result {
         margin-top: 20px;
         font-size: 18px;
```

```
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        th, td {
            border: 1px solid black;
            padding: 10px;
        }
        th {
            background-color: #28a745;
            color: white;
        }
        .add-student {
            margin-top: 30px;
            padding: 20px;
            border-top: 2px solid #ddd;
        }
        @media (max-width: 600px) {
            .container {
                width: 90%;
            }
        }
    </style>
</head>
<body>

    <div class="container">
        <h2>Semester Results with Ranking</h2>

        <!-- Add Student Section -->
        <div class="add-student">
            <h3>Add Student</h3>
            <input type="text" id="studentName" placeholder="Enter Student Name">
            <input type="number" id="subject1" placeholder="Marks for Subject 1 (0-100)">
            <input type="number" id="subject2" placeholder="Marks for Subject 2 (0-100)">
            <input type="number" id="subject3" placeholder="Marks for Subject 3 (0-100)">
            <input type="number" id="subject4" placeholder="Marks for Subject 4 (0-100)">
            <input type="number" id="subject5" placeholder="Marks for Subject 5 (0-100)">
            <button onclick="addStudent()">Add Student</button>
            <button onclick="calculateRankings()">Calculate Rankings</button>
        </div>

        <!-- Results Table -->
        <div class="result">
            <h3>Results</h3>
            <table id="resultsTable">
                <tr>
                    <th>Rank</th>
                    <th>Name</th>
```

```
            <th>Total Marks</th>
            <th>Percentage</th>
            <th>Grade</th>
        </tr>
    </table>
  </div>
</div>

<script>
  let students = [];

  function addStudent() {
    let name = document.getElementById("studentName").value;
    if (name.trim() === "") {
      alert("Please enter the student's name.");
      return;
    }

    let marks = [];
    for (let i = 1; i <= 5; i++) {
      let mark = document.getElementById(subject${i}).value;
      if (mark === "") {
        alert("Please enter marks for all subjects.");
        return;
      }
      mark = parseFloat(mark);
      if (mark < 0 || mark > 100) {
        alert("Marks should be between 0 and 100.");
        return;
      }
      marks.push(mark);
    }

    let total = marks.reduce((sum, mark) => sum + mark, 0);
    let percentage = (total / 500) * 100;
    let grade = getGrade(percentage);

    students.push({ name, total, percentage, grade });

    alert(${name}'s marks added successfully!);
    clearInputs();
  }

  function calculateRankings() {
    students.sort((a, b) => b.percentage - a.percentage);

    let table = document.getElementById("resultsTable");
    table.innerHTML = `
      <tr>
        <th>Rank</th>
        <th>Name</th>
```

```
                <th>Total Marks</th>
                <th>Percentage</th>
                <th>Grade</th>
            </tr>
        `;

        students.forEach((student, index) => {
            let row = table.insertRow();
            row.innerHTML = `
                <td>${index + 1}</td>
                <td>${student.name}</td>
                <td>${student.total} / 500</td>
                <td>${student.percentage.toFixed(2)}%</td>
                <td>${student.grade}</td>
            `;
        });
    }

    function getGrade(percentage) {
        if (percentage >= 90) return "A+";
        else if (percentage >= 80) return "A";
        else if (percentage >= 70) return "B";
        else if (percentage >= 60) return "C";
        else return "Fail";
    }

    function clearInputs() {
        document.getElementById("studentName").value = "";
        for (let i = 1; i <= 5; i++) {
            document.getElementById(subject${i}).value = "";
        }
    }
</script>

</body>
</html>
```

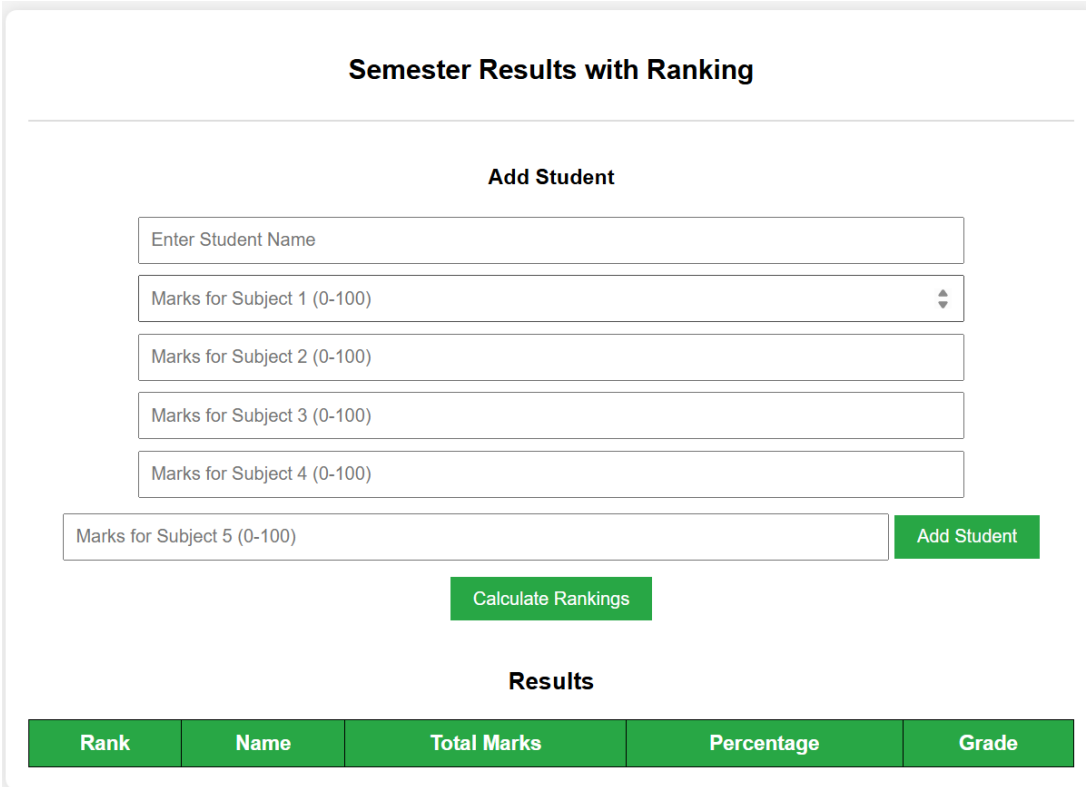
4.2 Test Cases



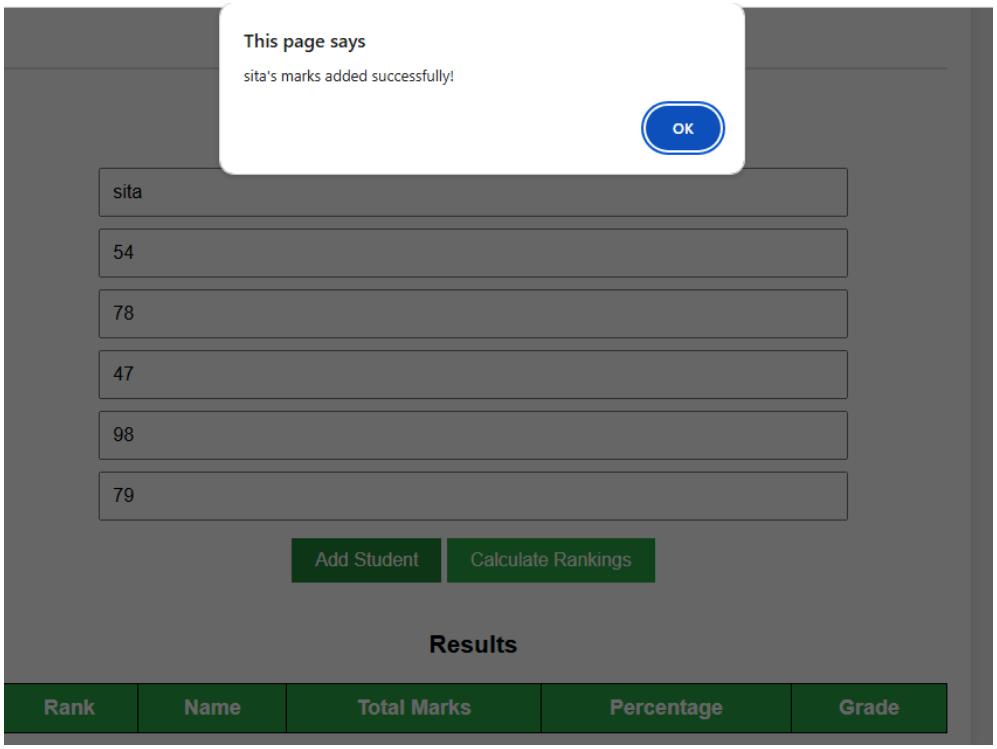
5.RESULTS

5.1 Output Screens

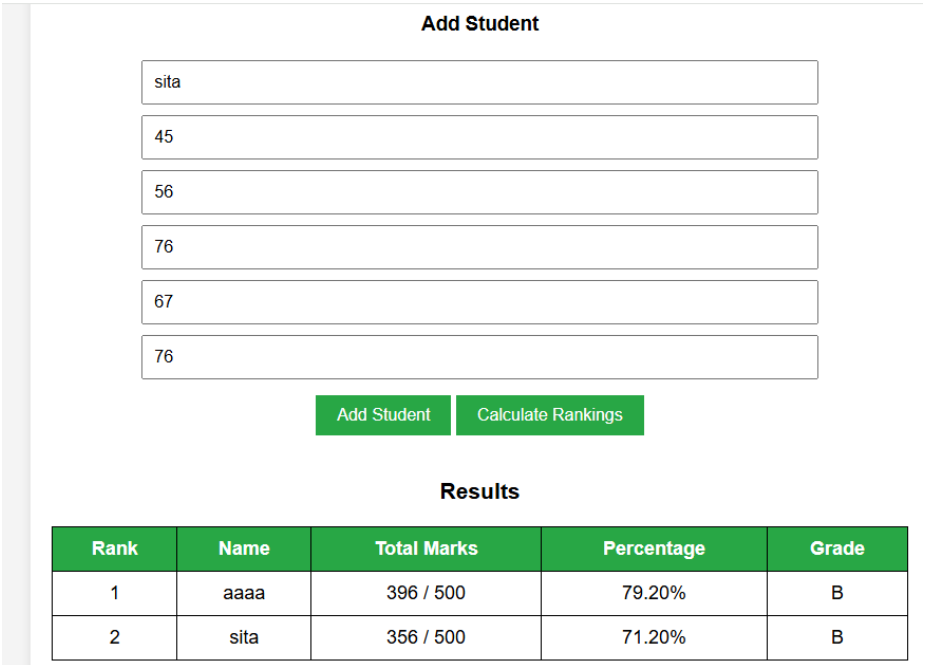
Home page



5.2. Add Student



5.3. Result



6.CONCLUSION

The development of the Semester Results System using HTML offers a simple, user-friendly, and efficient way for students to access their academic performance. By providing a clean interface and incorporating client-side validation, the system ensures that users enter correct and complete information before proceeding. This project not only simplifies the result-checking process but also reduces the workload on administrative staff by automating basic data retrieval tasks. Overall, this system demonstrates how basic web technologies can be used to create meaningful solutions in educational environments, improving accessibility and user experience for students and institutions alike.

7. References:

1. W3Schools. (2024). *HTML Forms Tutorial*. Retrieved from https://www.w3schools.com/html/html_forms.asp
2. Mozilla Developer Network (MDN). (2024). *HTML Form Elements*. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form
3. GeeksforGeeks. (2023). *Form Validation using HTML and JavaScript*. Retrieved from https://www.geeksforgeeks.org/form-validation-using-html-and-javascript/
4. TutorialsPoint. (2023). *HTML5 – Forms and Input Types*. Retrieved from https://www.tutorialspoint.com/html5/html5_forms.htm

Github link:

https://github.com/Ahalyabhavanam/sem-results