



## *Green University of Bangladesh*

*Department of Computer Science and Engineering (CSE)  
Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)*

---

# **Student Portal**

---

*Course Title: Web Programming Lab  
Course Code: CSE 302  
Section: 223 D1*

### Students Details

<b>Name</b>	<b>ID</b>
Nadia Islam Rupa	223002047
Kawser Miah	223002063

*Submission Date: 27/12/2024  
Course Teacher's Name: FERROZA NAZNIN*

[For teachers use only: **Don't write anything inside this box**]

<u>Lab Project Status</u>	
<b>Marks:</b>	<b>Signature:</b>
<b>Comments:</b>	<b>Date:</b>

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Overview . . . . .	3
1.2	Motivation . . . . .	3
1.3	Problem Definition . . . . .	3
1.3.1	Problem Statement . . . . .	3
1.3.2	Complex Engineering Problem . . . . .	4
1.4	Design Goals/Objectives . . . . .	4
1.5	Application . . . . .	5
<b>2</b>	<b>Design/Development/Implementation of the Project</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Project Details . . . . .	7
2.2.1	Login System . . . . .	7
2.2.2	Dashboard . . . . .	7
2.2.3	Profile Management . . . . .	8
2.2.4	Payment System . . . . .	8
2.2.5	Results Page . . . . .	8
2.2.6	Logout Functionality . . . . .	8
2.3	Implementation . . . . .	8
2.3.1	Workflow . . . . .	9
2.3.2	Tools and Libraries . . . . .	9
2.3.3	Implementation Details . . . . .	9
<b>3</b>	<b>Performance Evaluation</b>	<b>14</b>
3.1	Results Analysis/Testing . . . . .	14
3.1.1	Log in Page . . . . .	14
3.1.2	DashBoard Page . . . . .	15

3.1.3	Profile Page . . . . .	15
3.1.4	Payment Page . . . . .	16
3.1.5	Result Page . . . . .	16
3.1.6	Log Out . . . . .	17
3.2	Results Overall Discussion . . . . .	17
<b>4</b>	<b>Conclusion</b>	<b>18</b>
4.1	Discussion . . . . .	18
4.2	Limitations . . . . .	18
4.3	Scope of Future Work . . . . .	18

# Chapter 1

## Introduction

### 1.1 Overview

The **Student Portal** project is designed to provide a streamlined platform for managing student information, course registrations, class and exam schedules, and academic records. Built using HTML, CSS, JavaScript, PHP, and MySQL, this portal aims to simplify interactions between students and educational institutions. The system will offer secure access to essential academic functions, helping students stay organized and institutions manage student data efficiently.

### 1.2 Motivation

The motivation behind choosing the **Student Portal** project is driven by the need to address the challenges faced by students in managing their academic information. In many institutions, handling course registration, accessing class routines, and tracking exam schedules is often cumbersome and time-consuming. By developing this portal, I aim to simplify these processes and provide a user-friendly, accessible platform. Additionally, this project aligns with my academic interests in web development and database management, offering practical experience in solving real-world problems.

### 1.3 Problem Definition

#### 1.3.1 Problem Statement

Managing academic processes, such as course registration, class routines, and exam schedules, is often cumbersome and time-consuming for students. The proposed Student Portal aims to address this issue by offering a centralized, automated platform to simplify these processes, making them more accessible and manageable for students and administrators.

### 1.3.2 Complex Engineering Problem

The following Table 1.1 summarizes how the Student Portal addresses the attributes of a complex engineering problem:

Table 1.1: Summary of the attributes touched by the mentioned project

Name of the Attributes	Explain how to address
<b>P1:</b> Depth of knowledge required	The project requires proficiency in full-stack development, encompassing frontend technologies (HTML, CSS, JavaScript) and backend technologies (PHP, MySQL). Additionally, understanding of secure user authentication is essential.
<b>P2:</b> Range of conflicting requirements	The project must balance user-friendliness with security. For example, ensuring smooth user experience while safeguarding sensitive student data and securing login mechanisms.
<b>P3:</b> Depth of analysis required	The database design requires a thorough analysis to optimize performance, especially for handling large volumes of concurrent requests from multiple users.
<b>P4:</b> Familiarity of issues	The project touches upon familiar issues like secure password storage, form validation, and the prevention of SQL injection attacks in web development.
<b>P5:</b> Extent of applicable codes	This project must comply with standard web development practices for security, including password encryption and session management.
<b>P6:</b> Extent of stakeholder involvement and conflicting requirements	Students and administrators are the key stakeholders. Conflicting requirements may arise between user convenience (e.g., easy access) and institutional data privacy policies.
<b>P7:</b> Interdependence	The project requires integration of multiple subsystems like user authentication, database management, and real-time data updates, all of which must work together efficiently.

## 1.4 Design Goals/Objectives

The main design goals and objectives of the Student Portal project are outlined below:

- **User-Friendly Interface:** The portal should have an intuitive and responsive user interface that allows students to navigate easily across various sections, including course registration, routine viewing, and personal information management.

- **Efficient Data Management:** The system should efficiently manage and store large volumes of student data, including course details, routines, and exam schedules, while ensuring quick access and seamless data retrieval.
- **Secure Authentication:** Implement secure user authentication through password encryption, preventing unauthorized access to sensitive information and ensuring that all users' data remains private and protected.
- **Dynamic Course Registration:** Provide a dynamic and flexible course registration feature where students can browse available courses, check seat availability, and register for courses, all within the portal in real-time.
- **Personalized Routines:** Generate personalized class and exam routines for each student based on their registered courses, making it easy for students to view their schedules at any time.
- **Seamless Integration:** Ensure seamless integration between the frontend and backend systems, allowing real-time updates and interactions between the database, user actions, and displayed content.
- **Scalability:** Design the system to be scalable, so it can handle a growing number of users, courses, and academic sessions without compromising performance.
- **Flexibility for Future Enhancements:** Build the system in a modular fashion, allowing future additions and updates, such as new features or enhancements, without requiring major structural changes.

The fulfillment of these objectives will ensure that the Student Portal is not only functional but also highly efficient, secure, and scalable for institutional use.

## 1.5 Application

The **Student Portal** project serves multiple functions that enhance the academic experience for students and streamline administrative tasks for educational institutions. Below are the key applications of various features within the portal:

- **Home Page:** The home page serves as the entry point for users, providing an overview of the portal's functionalities. It includes options for login and registration, facilitating quick access for both new and returning users. The responsive design, implemented using CSS and Bootstrap, ensures that the portal is accessible across various devices, enhancing user experience.
- **Login/Registration Forms:** These forms are designed with input validation using JavaScript to ensure accurate user information. During registration, students fill out essential details such as name, email, and password, adhering to secure password rules. This feature simplifies the onboarding process and enhances security by ensuring compliance with specific password criteria.

- **Student Dashboard:** The dashboard serves as the central hub for user interaction, allowing students to view and update their personal information seamlessly. It features dynamic course registration, enabling students to select available courses through a user-friendly interface. Additionally, the dashboard displays a class routine and tracks grades and assignments, promoting accountability in their studies.
- **Password Change Option:** A dedicated section on the student dashboard allows students to update their passwords securely. This feature includes a form for users to enter their old password, new password, and confirm the new password. JavaScript validation ensures that all password requirements are met, maintaining account integrity and security.
- **Course Registration:** This feature provides an intuitive form where students can browse available courses for the semester. Detailed information about each course, including timings, instructor names, and seat availability, aids students in making informed decisions, enhancing overall satisfaction with the registration process.
- **Class Routine:** The class routine feature displays a weekly schedule organized into time slots with corresponding course names. This user-friendly interface promotes effective time management and reduces conflicts in students' schedules.
- **Exam Routine:** This feature lists all upcoming exams, detailing dates, times, and locations. By keeping students informed about their exam schedules, it reduces anxiety and allows for better preparation, leading to improved academic performance.
- **Overall Impact:** In summary, the Student Portal enhances the academic experience by providing streamlined processes for course registration, schedule management, and secure access to personal information. By incorporating responsive design, robust security measures, and user-friendly interfaces, this portal serves as an invaluable tool for students, fostering engagement and supporting their educational journeys.

# Chapter 2

## Design/Development/Implementation of the Project

### 2.1 Introduction

This project focuses on developing a comprehensive Student Portal system using HTML, CSS, JavaScript, and PHP. The portal aims to streamline student management by providing essential features such as a login page with JavaScript-based authentication, a dashboard displaying key information like financial details and results, and a payment page for installment management. The system also integrates a profile section for managing personal and parental details, contributing to an efficient and user-friendly student experience. [1] [2] [3].

### 2.2 Project Details

This section outlines the key components and features of the Student Portal project, detailing each aspect of its design, development, and functionality. The portal is structured to provide a seamless and interactive experience for students, allowing them to access and manage various essential aspects of their academic and financial records.

#### 2.2.1 Login System

The login system ensures secure access to the portal through JavaScript-based authentication. It validates user credentials and redirects authenticated users to their personalized dashboard. This feature prevents unauthorized access and protects sensitive student information.

#### 2.2.2 Dashboard

The dashboard serves as the central hub of the portal, providing students with an overview of their academic and financial details. Key components include:



- Welcome message personalized with the student's name.
- Financial details, including total fee, paid amount, discount, and due amounts.
- Semester-wise course results, displaying academic progress.

### **2.2.3 Profile Management**

This section allows students to update and manage their personal and parental details. It includes fields for updating contact information, addresses, and emergency contacts, ensuring that all personal data is current and accurate.

### **2.2.4 Payment System**

The payment page facilitates the management of installment payments, displaying the due amounts and installment history. Key features include:

- Displaying total bill, paid amount, due balance, and discounts.
- A table for installment details, including due dates, amounts, and late fees.
- Calculating and displaying the payable amount for each installment.

### **2.2.5 Results Page**

The results page displays the student's semester-wise course results, allowing them to track their academic performance throughout the course of their study. It provides clear visibility into grades and overall progress.

### **2.2.6 Logout Functionality**

The portal includes a logout feature that ensures secure session termination. A circular progress indicator is shown, with a 1500ms delay, simulating a smooth logout process before redirecting the user to the login page.

## **2.3 Implementation**

This section details the implementation of the Student Portal project. It covers the workflow, tools and libraries used, and the core implementation details, including relevant code snippets and screenshots. [4]

### **2.3.1 Workflow**

The workflow of the Student Portal is designed to ensure a smooth user experience. It starts with the user logging into the portal through the authentication page, followed by being redirected to their personalized dashboard. Upon accessing the dashboard, the student can view their academic details, financial summary, and results. The system allows students to update their profile, view payment details, and track semester-wise results. Logout functionality is incorporated at the end of each session to secure the student's data.

### **2.3.2 Tools and Libraries**

Several tools and libraries were utilized to develop the Student Portal:

#### **HTML, CSS, and JavaScript**

The front-end of the project is built using HTML and CSS for structuring and styling the content. JavaScript is used for interactive elements such as login authentication, displaying dynamic content in the dashboard, and processing payment details.

#### **PHP**

PHP is used for server-side functionality, handling data storage, login authentication, and communicating with the database to fetch user details, payment history, and semester results.

#### **MySQL**

A MySQL database is used to store and manage all the student-related data, including personal details, academic records, financial information, and payment history.

#### **Libraries**

- **jQuery:** For DOM manipulation and simplifying JavaScript code.
- **Bootstrap:** For responsive design and pre-built UI components.
- **Chart.js:** For generating graphical representations of payment and academic data.

### **2.3.3 Implementation Details**

#### **Login Authentication**

The login functionality is implemented using JavaScript to authenticate users on the client side. The process starts when the student enters their username and password.

If the credentials are valid, they are authenticated against the data in the PHP backend. Below is the JavaScript code snippet used for validating the login form:

```
javascript
document.getElementById('loginForm').onsubmit = function(event) {
    event.preventDefault();
    var username = document.getElementById('username').value;
    var password = document.getElementById('password').value;

    // Make AJAX call to validate login
    $.ajax({
        url: 'validate_login.php',
        type: 'POST',
        data: {username: username, password: password},
        success: function(response) {
            if(response === 'success') {
                window.location.href = 'dashboard.html';
            } else {
                alert('Invalid login credentials');
            }
        }
    });
};
```

## Dashboard Design

The dashboard presents the student's welcome message and academic/financial details in a structured format. It is dynamically generated using PHP to fetch data from the MySQL database. The following code illustrates how the financial details are retrieved and displayed on the dashboard:

```
fetch('payments.php',
    {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data), // Send stu_id as JSON
    })
    .then(response => {
        if (!response.ok) {
            return response.json().then(errorData => {
                // Show the error message in an alert box
                // alert('Error: ' + errorData.error);
                // Throw an error that will be caught in the catch block
                throw new Error(errorData.error); // This will trigger
            });
        }
    })
```

```

        return response.json(); // Return the data if successful
    })
    .then(data => {
        data.forEach(user => {
            const elements = document.getElementsByClassName("p1");
            elements[0].textContent= user.total_bill;
            elements[1].textContent = user.total_paid;
            elements[2].textContent = user.total_bill - user.total_paid;
            console.log(JSON.stringify(user)); }
        ));
    })
    .catch(error => {
        console.error('Error fetching data:', error)
        alert(error);
    });

```

## Payment Page

The payment page calculates the payable amount for each installment. The user is shown a table containing installment details, and late fees are dynamically computed if applicable. Below is the PHP code used to calculate and display the due balance and late fees:

```

    $due_amount = $row['due_amount'];
    $installments = mysqli_query($conn, "SELECT installment_amount, due_date FROM installment");

    while($installment = mysqli_fetch_assoc($installments)) {
        $due_date = new DateTime($installment['due_date']);
        $current_date = new DateTime();
        $late_fee = 0;
        if($current_date > $due_date) {
            $late_fee = calculateLateFee($installment['installment_amount']);
        }
        echo "<tr><td>" . $installment['installment_amount'] . "</td><td>" . $late_fee . "</td></tr>";
    }

```

## Logout Functionality

To ensure secure session termination, the logout functionality includes a circular progress indicator that simulates a logout delay. This is implemented using JavaScript to display the progress bar before redirecting to the login page:

```

document.getElementById('logoutBtn').onclick = function() {
    document.getElementById('logoutProgress').style.display = 'block';
    setTimeout(function() {
        window.location.href = 'login.html';
    }, 1500);
}

```

```
};
```

## Result Page

The result page allows students to view their semester-wise results. The data is fetched from the MySQL database and displayed dynamically. Each semester's results, including grades and subject names, are shown in a table format. Below is a sample PHP code that retrieves and displays the results:

```
.then(response => {
    if (!response.ok) {
        return response.json().then(errorData => {
            // Show the error message in an alert box
            // alert('Error: ' + errorData.error);
            // Throw an error that will be caught in the catch block
            throw new Error(errorData.error); // This will trigger th
        });
    }
    return response.json(); // Return the data if successful
})
.then(data => {
    const tableBody = document.getElementById("t1");
    data.forEach(user => {
        const row = document.createElement('tr');
        row.innerHTML = '
            <td>${user.Semester}</td>
            <td>${(user.Total_Points_Sum/user.Total_Credits).toFixed(2)}</td>
            <td>${user.Total_Credits}</td>
        ';
        tableBody.appendChild(row);
        console.log(JSON.stringify(user)); } );})
```

## Profile Page

The profile page allows students to view and update their personal and parental details. The page pulls data from the database and displays it in editable fields. The student can modify their details and submit the changes. Below is the PHP code to display and update personal information:

```
) .then(data => {
    data.forEach(user => {
        const elements = document.getElementsByClassName("prof1");
        elements[0].textContent = user.stu_name;
        elements[1].textContent = user.stu_id;
        elements[2].textContent = user.program;
        elements[3].textContent = user.batch;
```

```
elements[4].textContent = user.total_credits;
elements[5].textContent = user.contact;
elements[6].textContent = user.fathers_name;
elements[7].textContent = user.fathers_pro;
elements[8].textContent = user.mothers_name;
elements[9].textContent = user.mothers_pro;
const date = new Date(user.dob);
const options = { day: '2-digit', month: 'long', year: 'numeric' };
const formattedDate = date.toLocaleDateString('en-US', options);
elements[10].textContent = formattedDate;
elements[11].textContent = user.religion;
elements[12].textContent = user.gender;
elements[13].textContent = user.nationality;
console.log(JSON.stringify(user));
}
```

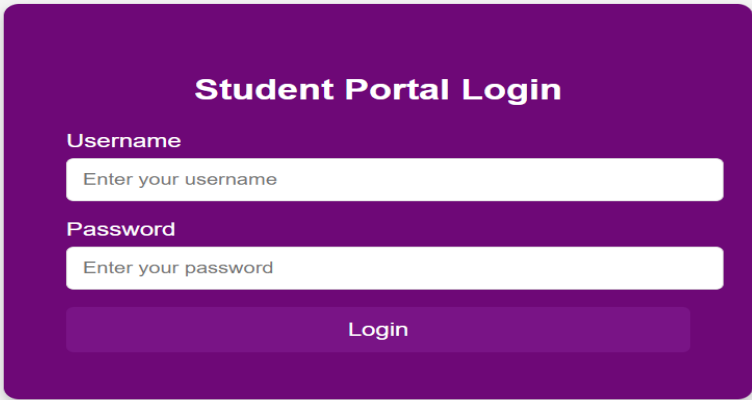
# Chapter 3

## Performance Evaluation

### 3.1 Results Analysis/Testing

This section presents the analysis and testing of the Student Portal project. Various test cases were conducted to ensure the system's functionality, including login authentication, data fetching, profile updates, result display, and payment processing. Testing also involved performance assessments to ensure the portal's responsiveness under different loads. The results validated the portal's correctness, usability, and performance, confirming its ability to meet the project requirements efficiently and effectively.

#### 3.1.1 Log in Page



**Student Portal Login**

Username  
Enter your username

Password  
Enter your password

Login

Figure 3.1: Login Page

### 3.1.2 DashBoard Page

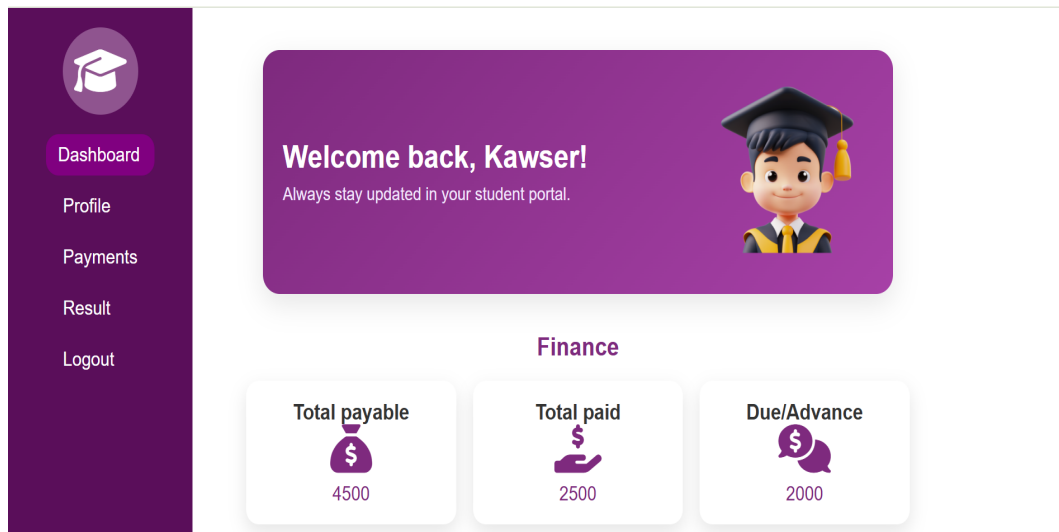


Figure 3.2: DashBoard Page

### 3.1.3 Profile Page

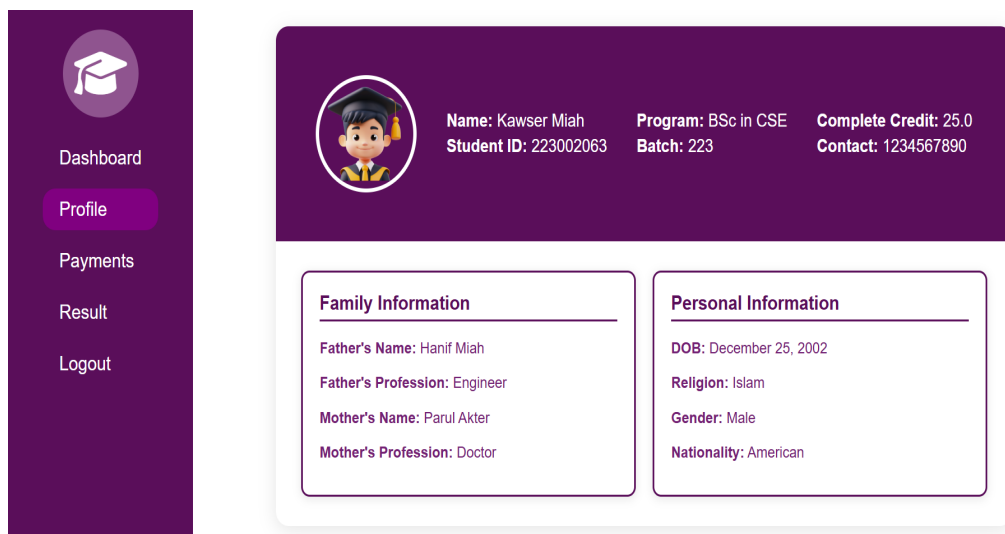


Figure 3.3: Profile Page



### 3.1.4 Payment Page

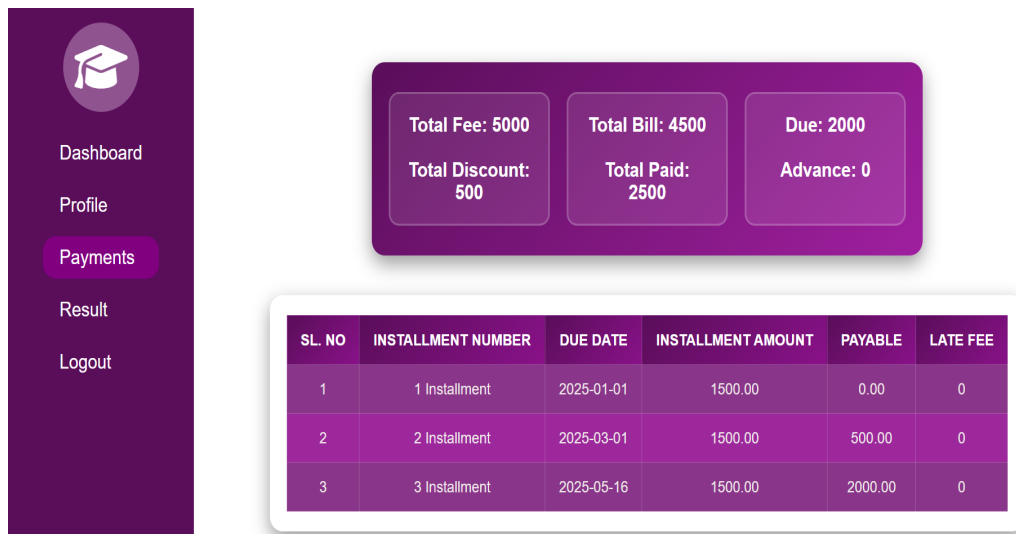


Figure 3.4: Payment Page

### 3.1.5 Result Page

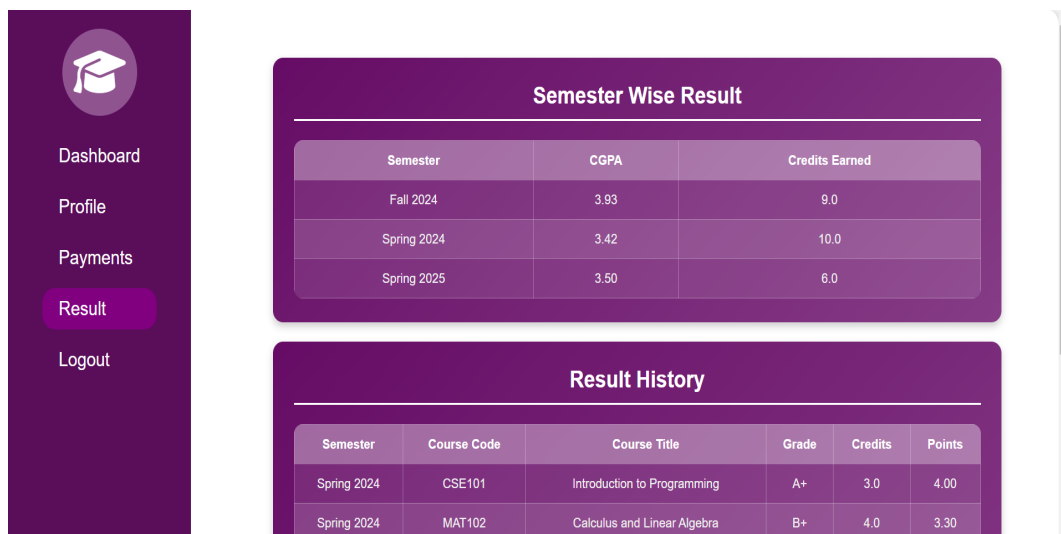


Figure 3.5: Result Page

### 3.1.6 Log Out

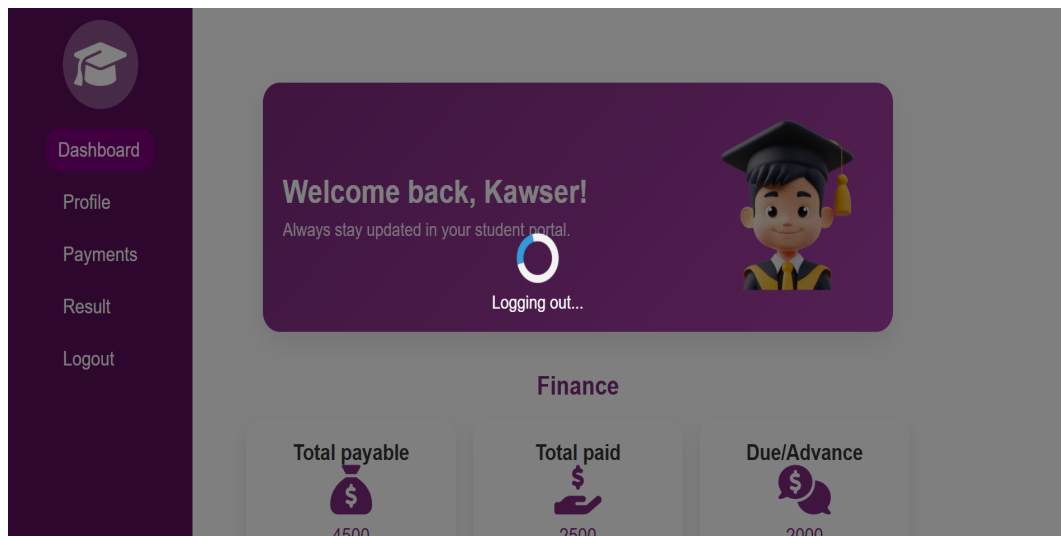


Figure 3.6: Log Out

## 3.2 Results Overall Discussion

The overall results indicate that the Student Portal successfully meets its intended goals, providing efficient login, profile management, and result display. However, some minor issues were identified, such as slow data fetching under heavy loads and occasional glitches in the payment processing system. These challenges were addressed by optimizing database queries and improving frontend responsiveness. Future improvements will focus on further enhancing performance and resolving any user interface inconsistencies for a smoother experience.

# **Chapter 4**

## **Conclusion**

### **4.1 Discussion**

The Student Portal project successfully implements essential features such as a login page, user profile management, financial tracking, and results display. Throughout development, the use of HTML, CSS, JavaScript, and PHP ensured robust front-end and back-end integration. Key functionalities, including JS-based authentication, installment payments, and dynamic result display, were thoroughly tested and found effective. However, some limitations in terms of scalability and security were identified, particularly with the centralized database. Future improvements may include enhancing security protocols and scalability, optimizing user experience, and adding additional features such as offline support and real-time notifications for better engagement.

### **4.2 Limitations**

The Student Portal project, while functional, has some limitations. The primary limitation is its reliance on a centralized database, which may cause performance issues as the number of users grows. Additionally, the authentication system could be improved by integrating more secure and scalable methods such as OAuth. The user interface, while intuitive, could benefit from further refinement to enhance accessibility and responsiveness across various devices and screen sizes. Lastly, the system currently lacks offline support, limiting its usability in low-connectivity environments.

### **4.3 Scope of Future Work**

Future work on the Student Portal project could involve enhancing the security of the login and payment systems by integrating multi-factor authentication and encryption protocols. The addition of an offline mode, allowing users to access certain features without an internet connection, would improve the system's accessibility. Further, scalability improvements could be made by optimizing the backend architecture and incorporating cloud-based solutions for better performance under heavy traffic. The inte-

gration of advanced data analytics and reporting features could also provide valuable insights for users and administrators.

# References

- [1] W3schools (html, css, javascript, php). <https://www.w3schools.com>. Accessed Date: 2024-11-14.
- [2] <https://www.geeksforgeeks.org>. <https://www.geeksforgeeks.org>. Accessed Date: 2024-11-14.
- [3] Code wall - build a student portal with php. <https://www.codewall.co.uk>. Accessed Date: 2024-11-14.
- [4] Github source code. <https://github.com/sohasara/student-portal>.