

1. INTRODUCTION

Student Information Management System is a web-based platform designed to efficiently manage student records and academic performance tracking in educational institutions. Traditional methods, such as maintaining student marks in paper-based records or Excel sheets, are time-consuming, error-prone, and difficult to manage. Faculty members often struggle with updating multiple assessments, and students face challenges in accessing their academic progress without faculty intervention. It provides a structured and automated approach to academic management by offering role-based access and ensuring secure data handling. The system is designed for four key roles: Admin, Head of Department (HOD), Faculty (Teacher), and Student. The admin is responsible for approving HOD registrations based on valid credentials. If an HOD or faculty member forgets their password, they must contact the admin for a reset. The HOD manages academic structures by adding the number of sections in each year for their respective branches. They also create and organize exams, define subjects, and add student details. Additionally, the HOD approves teacher requests for subject assignments and can generate student performance reports. A teacher/faculty can request subject assignments from the HOD. Once assigned, they are responsible for entering student marks for each exam related to their subject. Teachers can also request modifications to marks, which require HOD approval. Students have read-only access to their marks and can track their academic performance through the system. Student Information Management System also enhances collaboration and communication between faculty, students, and administrators. By centralizing student data, it eliminates redundancy and ensures consistency in academic records. The system enables real-time data retrieval, allowing faculty to update marks instantly and students to view their performance without delays. By replacing manual processes with an automated and structured system, it improves efficiency, reduces faculty workload, ensures data accuracy, and enhances transparency. In the following sections of this documentation, we will delve into the technical details, user guides, and system architecture to provide a comprehensive understanding of how this system functions and how to make the most of its features

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The current student information management system in many educational institutions is largely manual or relies on outdated digital tools, leading to inefficiencies in academic record management. Faculty members typically use Excel sheets, physical registers, or disconnected software systems to maintain student data, making the process time-consuming, error-prone, and difficult to manage.

This traditional approach has several implications:

- **Manual Data Entry and Processing:** Faculty members manually enter and update student marks, which increases the chances of human errors, data loss, and inconsistencies in records.
- **Dependence on Faculty for Student Access:** Students cannot directly access their academic records; instead, they must rely on faculty members to provide mark sheets and performance updates, leading to delays and inefficiencies.
- **Time-Consuming and Repetitive Tasks:** The lack of automation means faculty must manually verify, calculate, and update cumulative marks for each student, consuming significant time and effort.
- **Limited Accessibility and Communication:** Since student data is often stored on local computers or in physical files, access is restricted to specific terminals or faculty members, making it difficult for students and administrators to retrieve necessary information in real time.
- **Difficulty in Managing Assessments and Subjects:** Faculty members struggle to track multiple assessments (mid-term exams, unit tests, and finals) across different subjects, leading to disorganized record-keeping and mismanagement.
- **No Centralized Control Over Data Management:** Since different departments manage student records independently, there is no unified platform to monitor, modify, or analyze academic performance across the institution.

- **Security and Data Loss Risks:** Physical records and outdated digital systems are vulnerable to data loss, unauthorized access, and security breaches, putting sensitive student information at risk.

These issues highlight the urgent need for a centralized, automated, and user-friendly solution to streamline student information management, reduce faculty workload, and provide students with direct access to their academic performance.

2.2 PROPOSED SYSTEM

The proposed Student Information Management System (SIMS) is a modern solution designed to overcome the limitations of traditional student data management systems by implementing a centralized database that enhances accuracy, accessibility, and security. This system aims to streamline academic record management, examinations, performance tracking, and communication within an educational institution. Traditional methods, such as handwritten registers or individual spreadsheets, make it difficult to maintain consistency in academic records. With SIMS, all student data is stored in a unified database, updated in real-time, and accessible remotely through a secure, web-based platform. This ensures that students, faculty and administrators can retrieve or modify data as per their roles, without unnecessary dependencies or inefficiencies. To maintain data integrity and security, the system incorporates role-based access control, ensuring that only authorized users have specific privileges:

- Admin is responsible for verifying and approving HOD registrations at the time of onboarding and handling password reset requests for both HODs and faculty members.
- HODs manage the academic structure of their respective branches by defining sections, subjects, and exams, as well as handling student records. They also have the authority to approve faculty requests for subject assignments and generate student performance reports.
- Faculty members (teachers) can request subject assignments from HODs and, once approved, can enter student marks for specific subjects and exams. If modifications are required, faculty must request approval from the HOD before making changes.
- Students have view-only access to their academic records, enabling them to track their progress without relying on manual updates from faculty.

By digitizing and centralizing student information, SIMS improves communication between different stakeholders within the institution. Additionally, SIMS ensures real-time access to academic data, allowing students and faculty to make quick, informed decisions based on the most recent updates. From a security perspective, the system incorporates encryption and database storage, ensuring that sensitive student data remains protected from unauthorized access. By eliminating the risks associated with manual record-keeping, such as data loss, duplication, or security vulnerabilities, SIMS provides a highly reliable and cost-effective solution for educational institutions. Furthermore, SIMS reduces operational costs by minimizing the need for physical paperwork and manual data entry, thus optimizing institutional resources. HODs no longer have to deal with complex, repetitive tasks like creating new academic structures from scratch or manually approving subject allocations. The system automates these processes, allowing them to focus on more strategic academic decisions. With its scalability and adaptability, SIMS is designed to handle large volumes of student data efficiently. It ensures that institutions can modernize their student information management processes, making them more structured, accessible, and error-free. By leveraging automation and centralized control, SIMS represents a technological shift towards a more efficient, transparent, and secure academic environment, ultimately enhancing the overall experience for students, faculty, and administrators.

3. SOFTWARE AND HARDWARE SPECIFICATIONS

3.1 SOFTWARE REQUIREMENTS

Operating System

- Any Windows operating system
- **Operating System Version:** Different software may require specific versions of Windows. Some software might only be compatible with Windows 10, while others might work on Windows 7 or Windows 8 as well.

Database: MySQL Server

- **MySQL Server Installation Package:** You'll need to download the MySQL Server installation package compatible with your operating system. MySQL provides various editions, including Community Server (free), Enterprise Edition (commercial), and others.
- **Dependencies:** MySQL Server may have dependencies on other software packages or libraries, especially when installing on Linux or UNIX-like systems. Ensure that any required dependencies are installed before proceeding.
- **Database Configuration:** During installation, you'll configure the MySQL Server instance, including setting the root password, choosing the port number, and specifying the data directory.
- **MySQL2 Package:** A modern MySQL client for Node.js, providing support for prepared statements and better performance.

Backend Technologies

- **Node.js** (JavaScript runtime environment)
- **Express.js** (Web framework for Node.js)

Package Managers & Dependencies

- **npm (Node Package Manager):** For managing dependencies and libraries in Node.js projects.

Web Servers

- **Express.js:** Serves as the backend framework for handling HTTP requests, eliminating the need for Apache or Nginx in most cases.

Version Control & Testing

- **Version Control:** Git for managing code changes and collaboration.
- **Testing Frameworks:** Mocha, Jest, or Chai for unit and integration testing.

Development Tools: VS Code or any preferred code editor.

3.2 HARDWARE REQUIREMENTS

Screen resolution at least 800X600 is required for proper and complete viewing of screens.

Higher resolution will be accepted.

RAM (2 GB)

Processor (2.0 GHz)

Storage (5 GB)

4. FEASIBILITY STUDY

A Feasibility Study for a Student Information Management System (SIMS) is a comprehensive evaluation process to determine if the development and implementation of the system are viable from multiple perspectives, including technical, economic and social. It aims to ensure that the system meets the institution's needs while being feasible to build, deploy, and maintain. Below is an outline of the key sections for a Feasibility Study of a Student Information Management System:

4.1 ECONOMIC FEASIBILITY

The project is economically feasible as the only cost involved is having a computer with the minimum requirements mentioned earlier. For the users to access the application, the only cost involved will be in getting access to the Internet.

4.2 TECHNICAL FEASIBILITY

Objective: Technical feasibility refers to the assessment of whether a proposed project, system, or solution can be developed using existing technology and whether it can be implemented within the constraints of the available resources, skills, and capabilities. It evaluates the practicality of the project from a technological perspective.

Key elements of technical feasibility include

Technology Assessment: Determining if the required technology is available or can be developed to support the proposed project. This involves evaluating the hardware, software, networks, and other technical requirements necessary for the system.

Resource Availability: Assessing the availability of skilled personnel, expertise, and other resources needed for the development and maintenance of the proposed system. This includes analyzing whether the required skill sets are present within the organization or if external expertise needs to be sourced.

Infrastructure Compatibility: Analyzing whether the proposed system can be integrated with the existing infrastructure without significant disruptions or if modifications are needed. This could involve considering the compatibility with existing systems, databases, and networks.

Scalability and Flexibility: Evaluating whether the proposed system can adapt and grow in response to changing needs and demands. It involves assessing the scalability and flexibility

of the technology and infrastructure to accommodate potential future expansions or modifications.

Risk Analysis: Identifying potential technical risks or challenges that could affect the successful implementation of the project. This could include issues related to technology obsolescence, compatibility issues, or limitations in resources. To evaluate whether the project can be successfully developed and implemented from a technical standpoint.

4.3 SOCIAL FEASIBILITY

From the feasibility study for a Student Information Management System, you'll focus on evaluating how the system will be received and utilized by the college community. Here's a framework for conducting such a study:

Stakeholder Analysis:

Identify all stakeholders involved, such as students, faculty, administrators, and external participants (if applicable). Determine their level of involvement in events and their expectations from Student Information management system.

User Acceptance:

Assess the willingness of stakeholders to adopt to student information management system. Identify potential barriers to adoption, such as resistance to change or concerns about usability.

Social Impact:

Assess the potential social benefits of implementing the system, such as improved communication, collaboration, and community-building. Consider how the system may contribute to the college's mission and values, such as promoting diversity and inclusion.

Long-Term Sustainability:

Consider the long-term sustainability of the student information management system in terms of user engagement, funding, and maintenance. Identify strategies for fostering continued use and support within the college community.

5. SYSTEM DESIGN

5.1 DATA FLOW DIAGRAMS

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. It is a graphical tool, useful for communicating with users, managers and other personnel. it is useful for analysing existing as well as proposed system.

5.1.1 DATA FLOW DIAGRAM FOR ADMIN MODULE

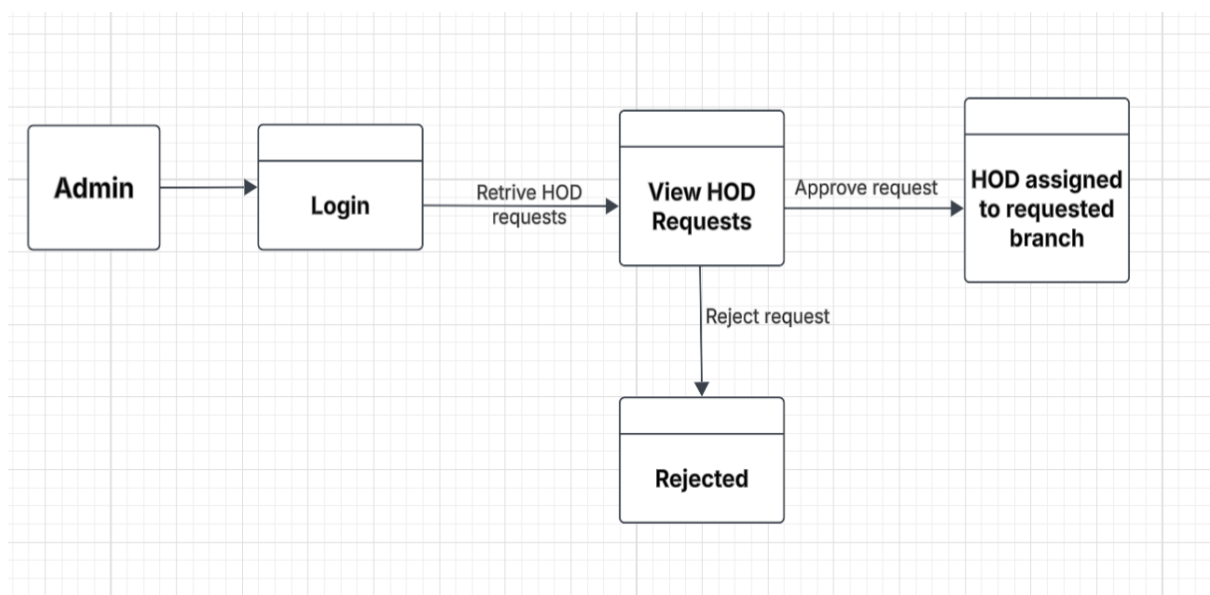


Fig 1: Data Flow Diagram for Admin Module

Admin views the HOD requests for allocating the branch, Admin can either approve or reject the request. If the request is approved then branch is assigned to corresponding requested HOD. The request will be rejected by the admin if the request is not valid.

5.1.2 DATA FLOW DIAGRAM FOR HOD MODULE

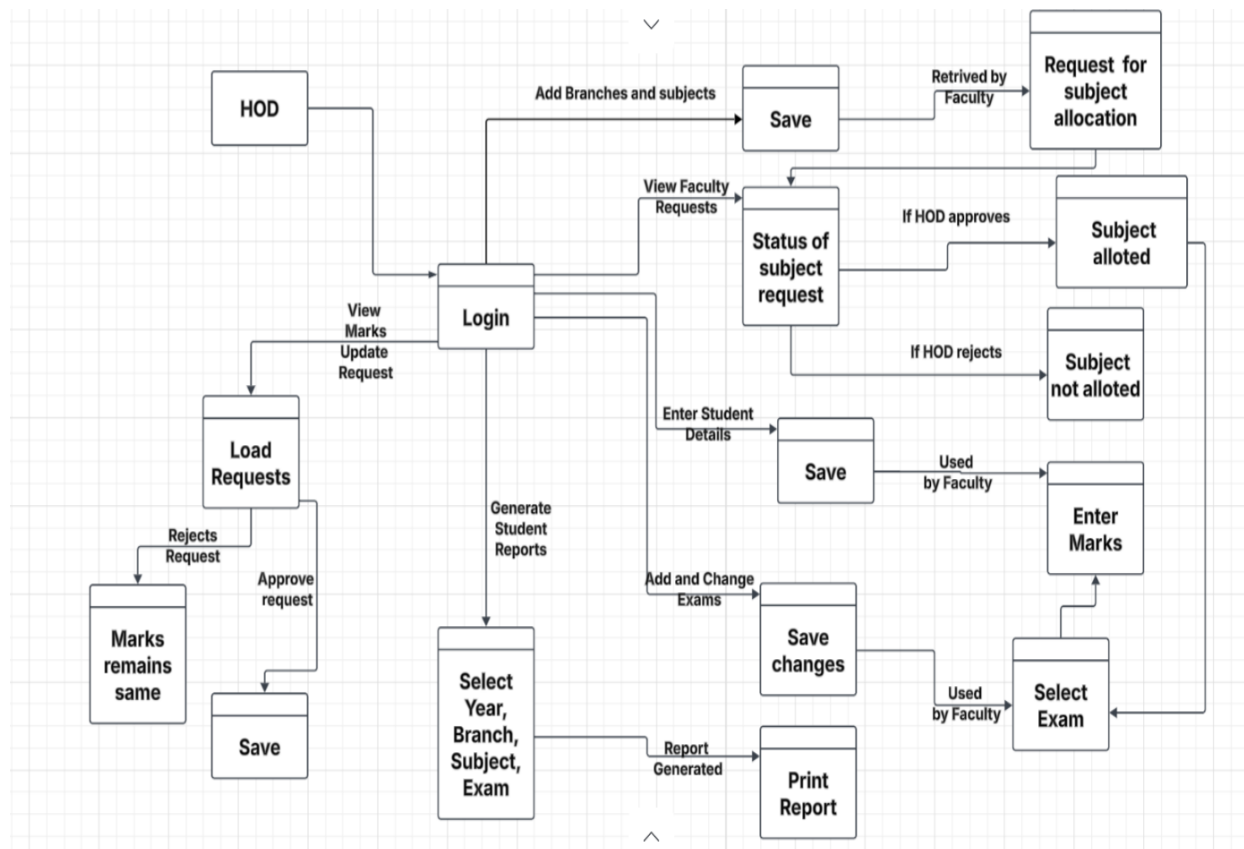


Fig 2: Data Flow Diagram for HOD Module

Upon logging in, the HOD can add branches and subjects, manage faculty requests for subject allocations, and monitor their status. When a faculty member submits a subject request, the HOD reviews it—if approved, the subject is allotted; if rejected, it is marked as not allotted. The HOD is also responsible for entering and saving student details, which are later used by faculty during assessments. Additionally, the HOD handles marks update requests submitted by faculty, choosing to either approve and save them or reject them, keeping the marks unchanged. They also manage examinations by selecting the year, branch, subject, and exam, then adding or modifying exam details, which are saved for faculty access. Lastly, the HOD can generate and print student reports based on selected academic parameters, ensuring smooth and structured academic operations within the institution.

5.1.3 DATA FLOW DIAGRAM FOR FACULTY MODULE

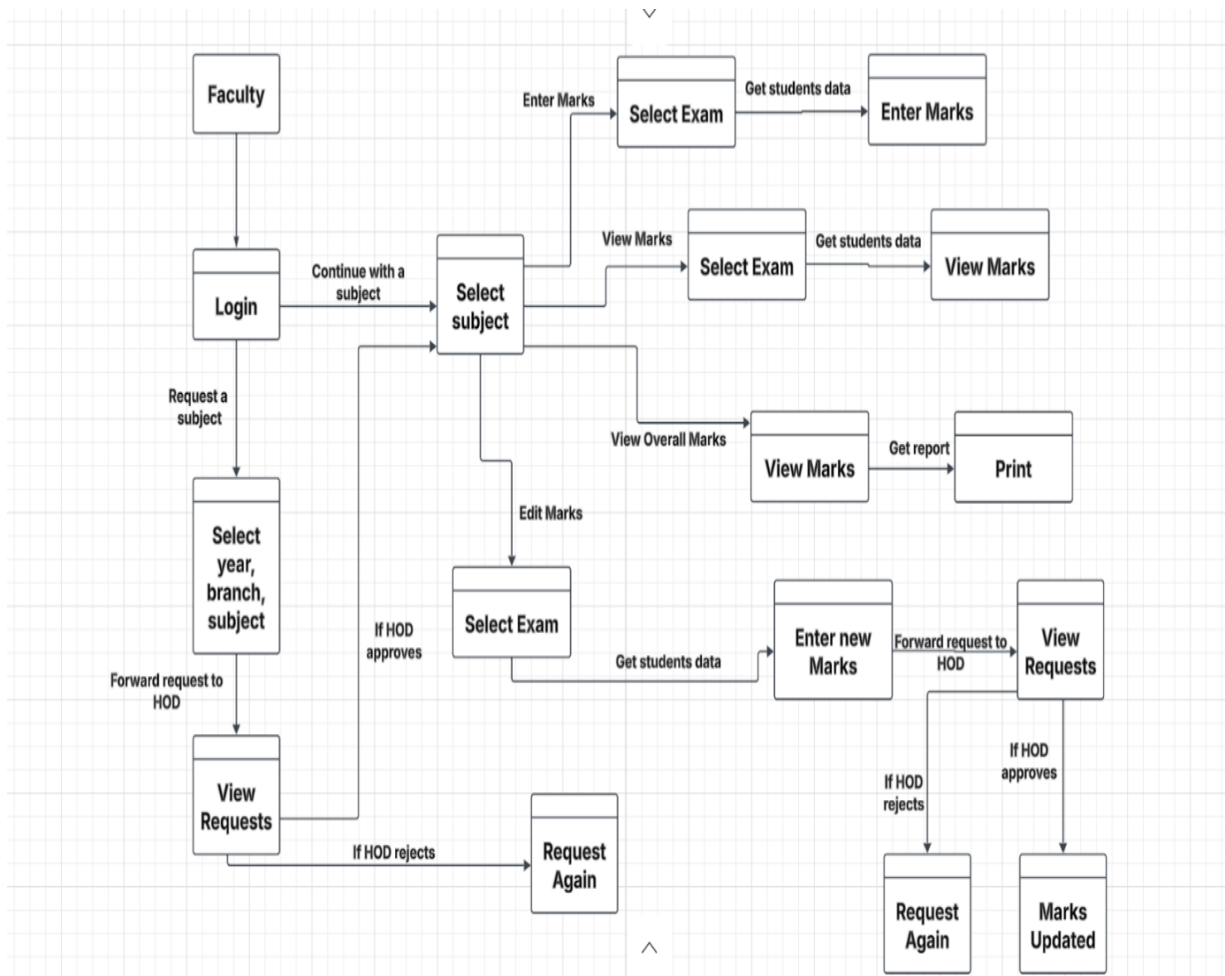


Fig 3: Data Flow Diagram for Faculty Module

After logging in, faculty can request subject allocation by selecting the year, branch, and subject, and then forwarding the request to the HOD for approval. If approved, they proceed by selecting the subject and exams to enter student marks. They can also view, edit, or enter new marks, which again requires HOD approval for updates. In case of rejection, they can resubmit the request. Faculty can view students' overall marks, generate reports, and print them. This module ensures a smooth flow of academic data between faculty and HOD, supporting collaborative and transparent academic management.

5.1.4 DATA FLOW DIAGRAM FOR STUDENT MODULE

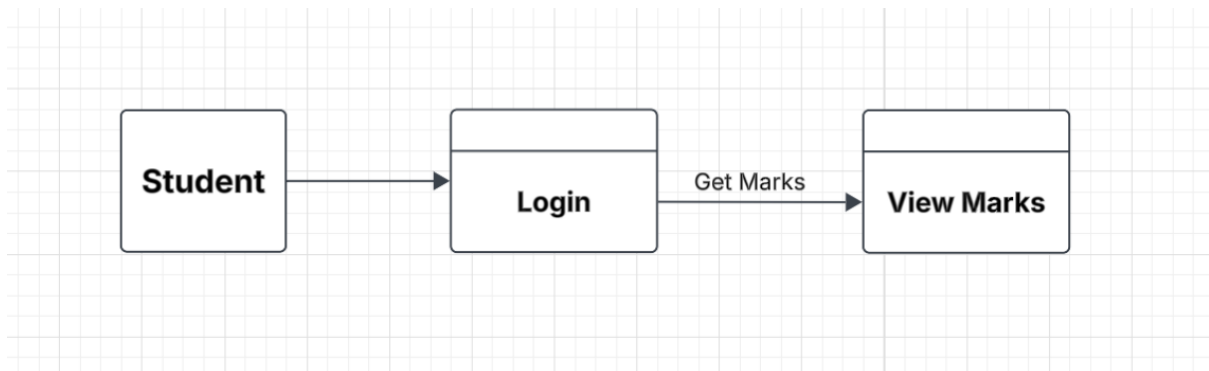


Fig 4: Data Flow Diagram for Student Module

Here the Students after logging in, they can view their marks.

5.2 UML DIAGRAMS

UML (Unified Modeling Language) diagrams are graphical representations used to visualize, specify, construct, and document the artifacts of a software system. They provide a standardized way to communicate system structure, behavior, and interactions among various components, making them invaluable tools for software development and design documentation. UML is a notation that resulted from the unification Of Object Modeling Technique and Object-Oriented Software Technology. UML has been designed for broad range of application. Hence, it provides constructs for a broad range of systems and activities.

Use Case Diagrams: These illustrate the interactions between users and the system. They show the system's functionalities from a user's perspective.

Class Diagrams: These depict the structure of a system by showing the classes in the system, their attributes, methods, and relationships between classes.

Sequence Diagrams: Sequence diagrams illustrate the interactions between objects or components over time. They show the sequence of messages exchanged among objects.

Activity Diagrams: Activity diagrams model the workflow or process of a system. They show the flow of activities or actions from start to finish.

State Machine Diagrams: Also known as state diagrams, these depict the states of an object or system and the transitions between those states triggered by events.

Component Diagrams: These illustrate the components or modules of a system and their dependencies.

Deployment Diagrams: Deployment diagrams show the physical deployment of software components to hardware nodes. They depict the configuration of runtime processing nodes and the components that live on them.

5.2.1 CLASS DIAGRAM

A class diagram in UML diagrams illustrates the structure of a system by showing the classes of the system, their attributes, operations or methods, and the relationships among them. It provides a blueprint for understanding the static structure of the system, facilitating communication among stakeholders and guiding the implementation process.

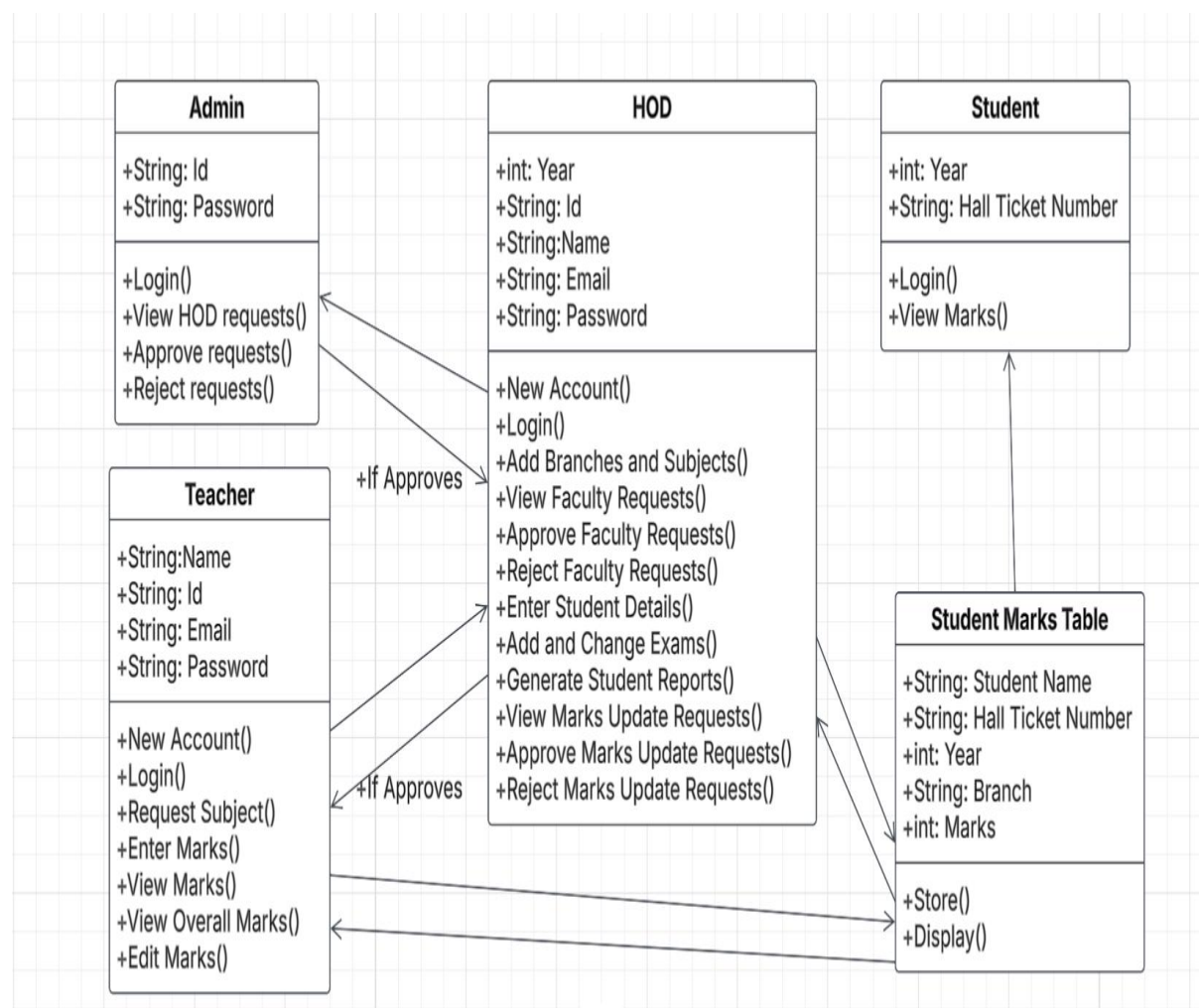


Fig 5: Class Diagram for Student Information Management System

5.2.2 USE CASE DIAGRAM

A use case diagram in UML diagrams depicts the interactions between actors (users or external systems) and a system to accomplish specific goals or tasks. It provides a high-level view of the system's functionality from the perspective of its users, helping to identify requirements and understand the system's behavior in various scenarios. Use case diagrams serve as a valuable tool for requirements analysis, system design, and communication among stakeholders.

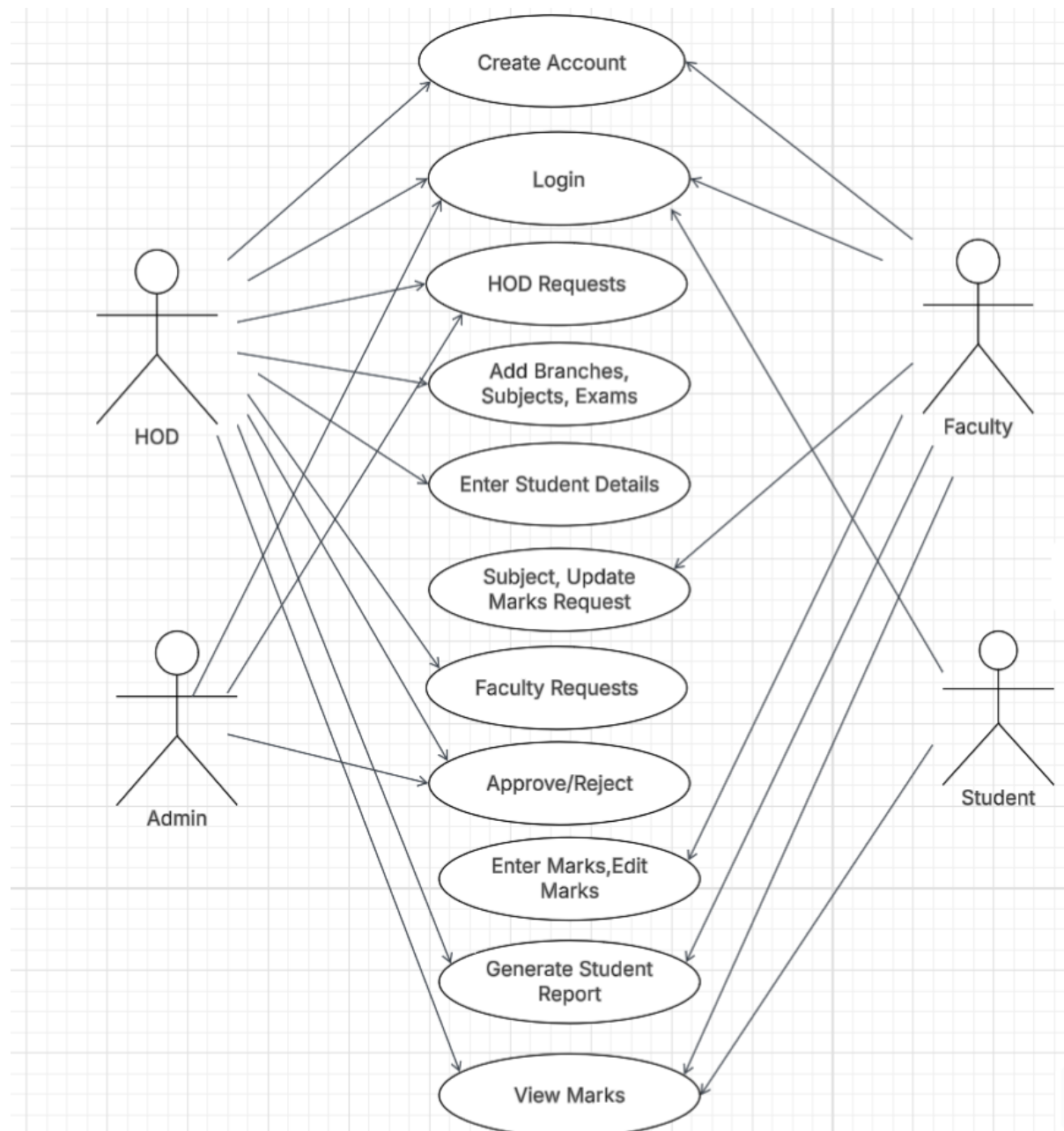


Fig 6: Use Case Diagram for Student Information Management System

5.2.2.1 USE CASE DIAGRAM FOR HOD MODULE

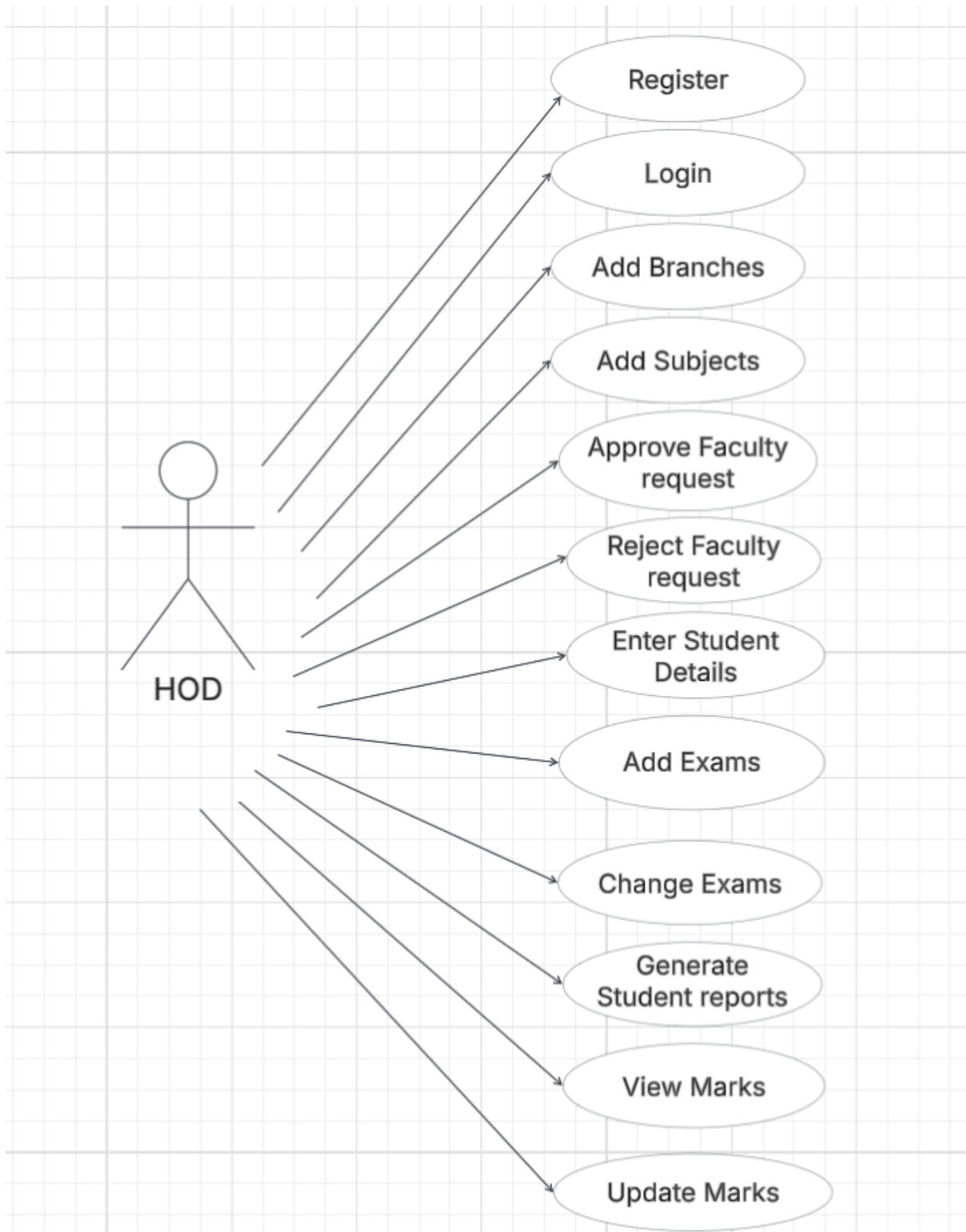


Fig 7: Use Case Diagram for HOD Module

The HOD can register, login, and manage various academic entities such as adding branches and subjects. They are responsible for reviewing faculty subject requests approving or rejecting them accordingly. Additionally, the HOD can enter and update student details, add or change exams, and manage marks by viewing and updating them as needed. They also generate student reports for performance evaluation.

5.2.2.2 USE CASE DIAGRAM FOR FACULTY MODULE

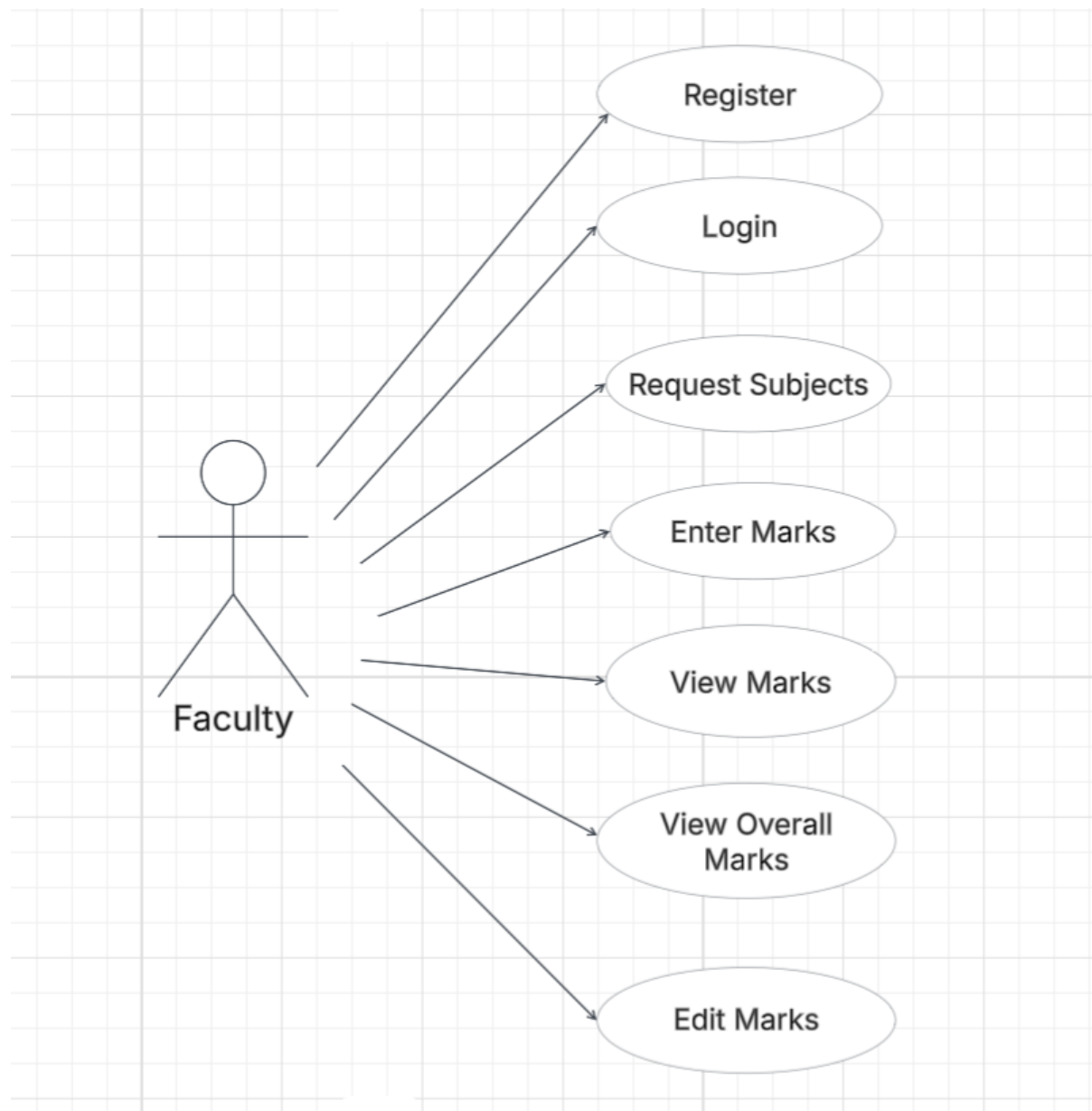


Fig 8: Use Case Diagram for Faculty Module

Faculty members can register and log in to the system, after which they can request subject allotments based on their department and year. Once approved, they can enter student marks for various exams, view individual marks as well as overall student performance. The system also allows faculty to edit previously entered marks if needed, ensuring data accuracy.

5.2.2.3 USE CASE DIAGRAM FOR ADMIN MODULE

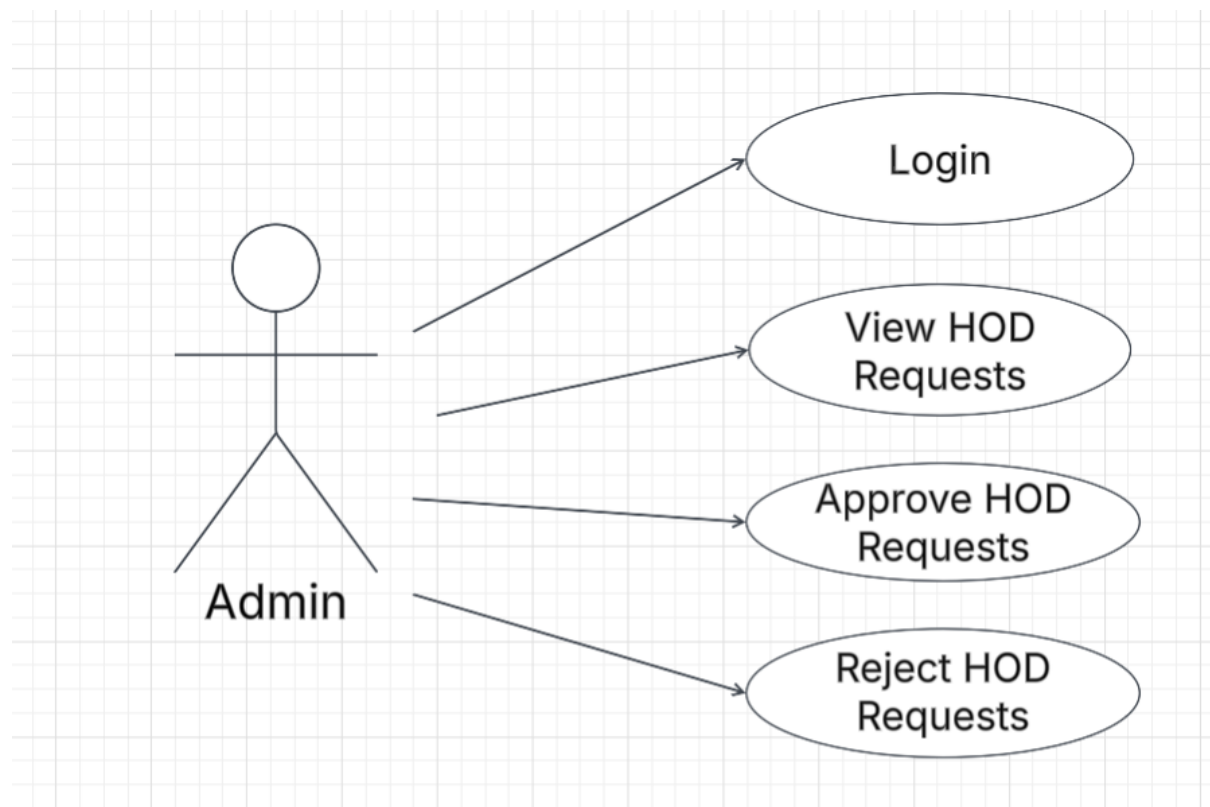


Fig 9: Use Case Diagram for Admin Module

The Admin logs in to the system to monitor all incoming HOD registration requests. Upon reviewing these requests, the Admin can choose to approve or reject them based on eligibility or organizational policies. This module ensures that only authorized HODs are allowed access to the SIMS platform.

5.2.2.4 USE CASE DIAGRAM FOR STUDENT MODULE

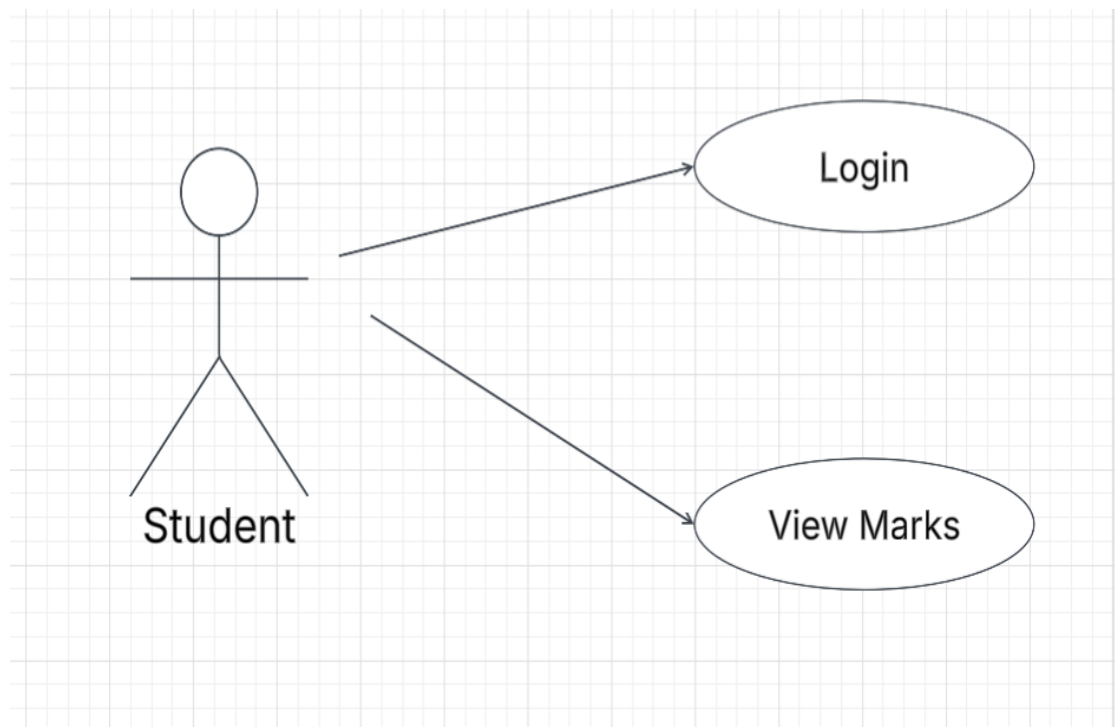


Fig 10: Use Case Diagram for Student Module

The actions done by the student is to login, and to view the Marks.

5.2.3 SEQUENCE DIAGRAM

A sequence diagram in UML is a visual representation of interactions between objects or components in a system over time. It illustrates the flow of messages or method calls between these entities, depicting the order in which they occur. Sequence diagrams are particularly useful for understanding the dynamic behavior of a system, including how objects collaborate to achieve certain functionality and the sequence of steps involved in a particular scenario. They provide insights into system logic, message passing, and the timing of interactions, aiding in system design, communication among team members, and debugging. With their clear and intuitive layout, sequence diagrams serve as invaluable tools for both developers and stakeholders in comprehending and refining system behavior.

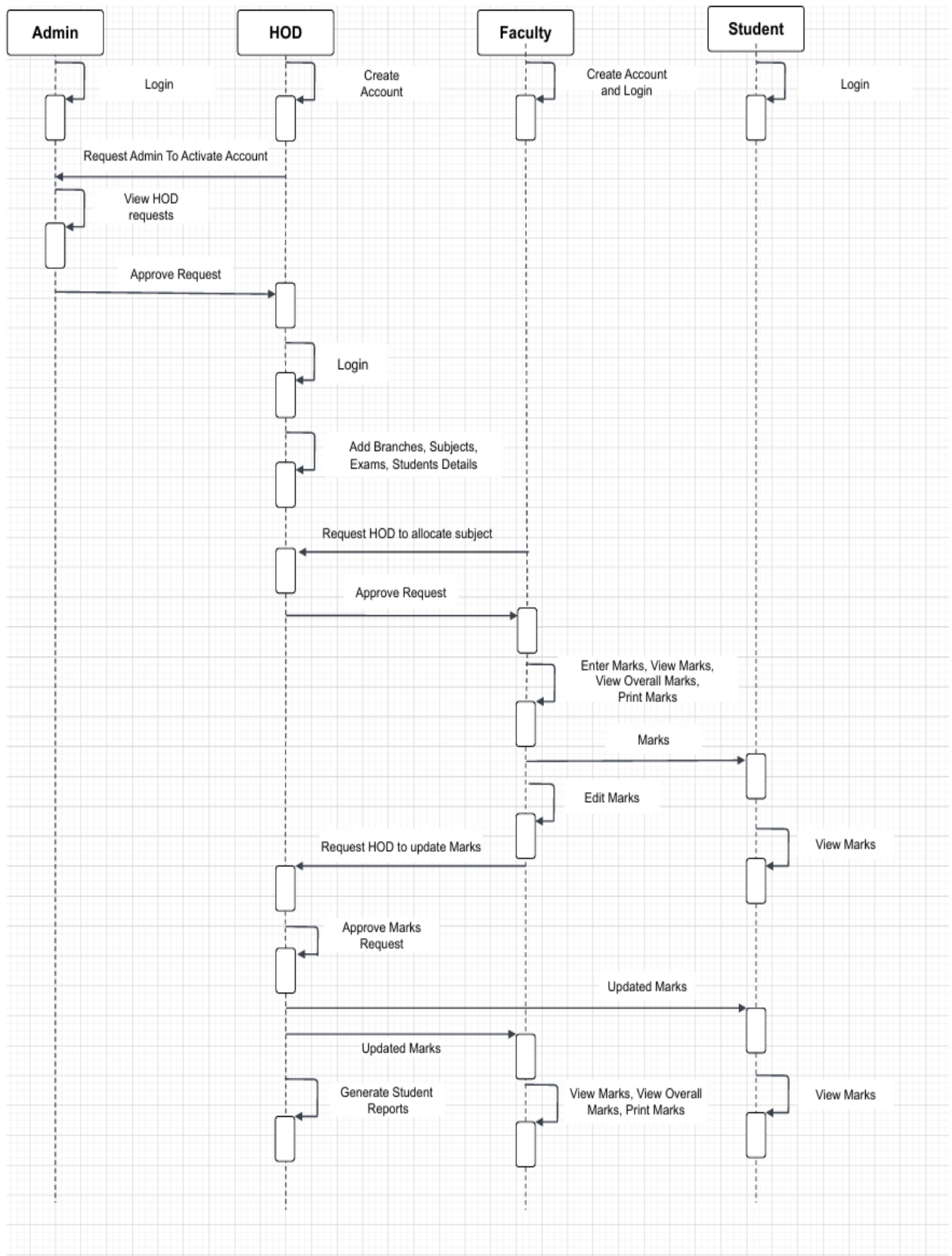


Fig 11: Sequence Diagram for Student Information Management System

6. MODULES

6.1 USER MODULE

Here Users are Hod, Faculty and Students. Users can register and login into the website.

The users have different tasks based on their role such as

- **Hod:** The HOD manages academic structures by adding the number of sections in each year for their respective branches. They also create and organize exams, define subjects, and add student details. Additionally, the HOD approves teacher requests for subject assignments and can generate student performance reports.
- **Teacher:** A teacher/faculty can request subject assignments from the HOD. Once assigned, they are responsible for entering student marks for each exam related to their subject. Teachers can also request modifications to marks, which require HOD approval.
- **Student:** Students have read-only access to their marks and can track their academic performance through the system.

6.1.1 HOME PAGE

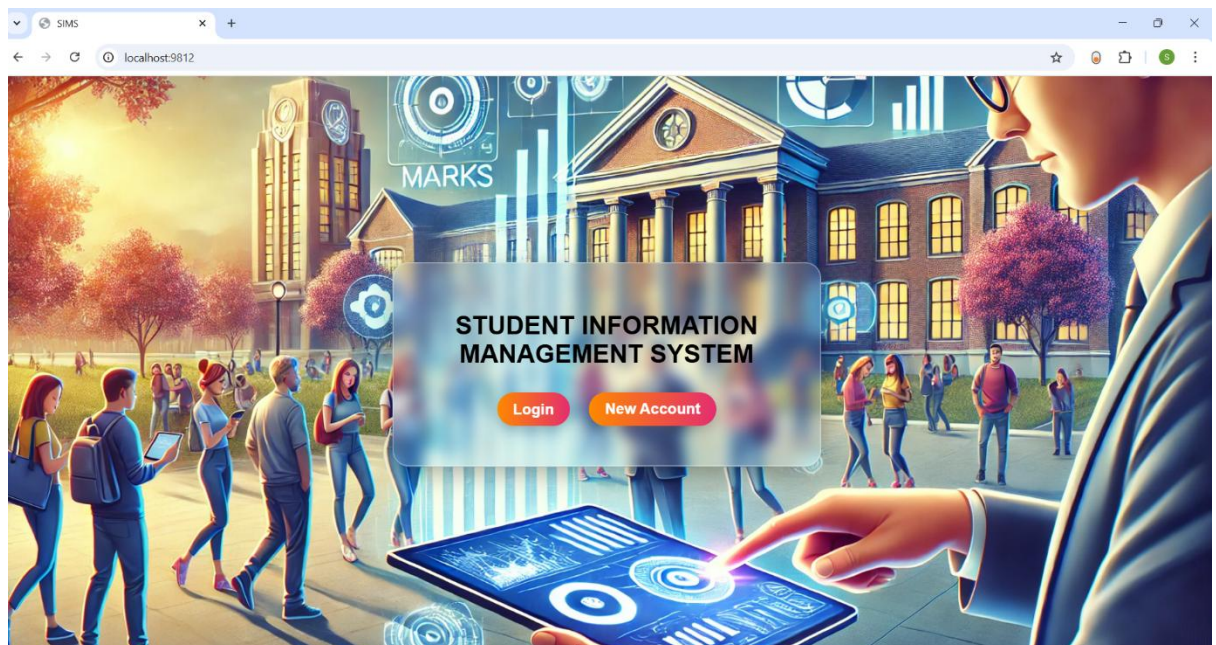


Fig 12: Home page

The above shown screen is the home page of our proposed project “**Student Information Management System** “. Where you can login if you’re the old user. If you are new user

then you need to click on New Account for registering, and then login into their page.

6.1.2 NEW ACCOUNT PAGE

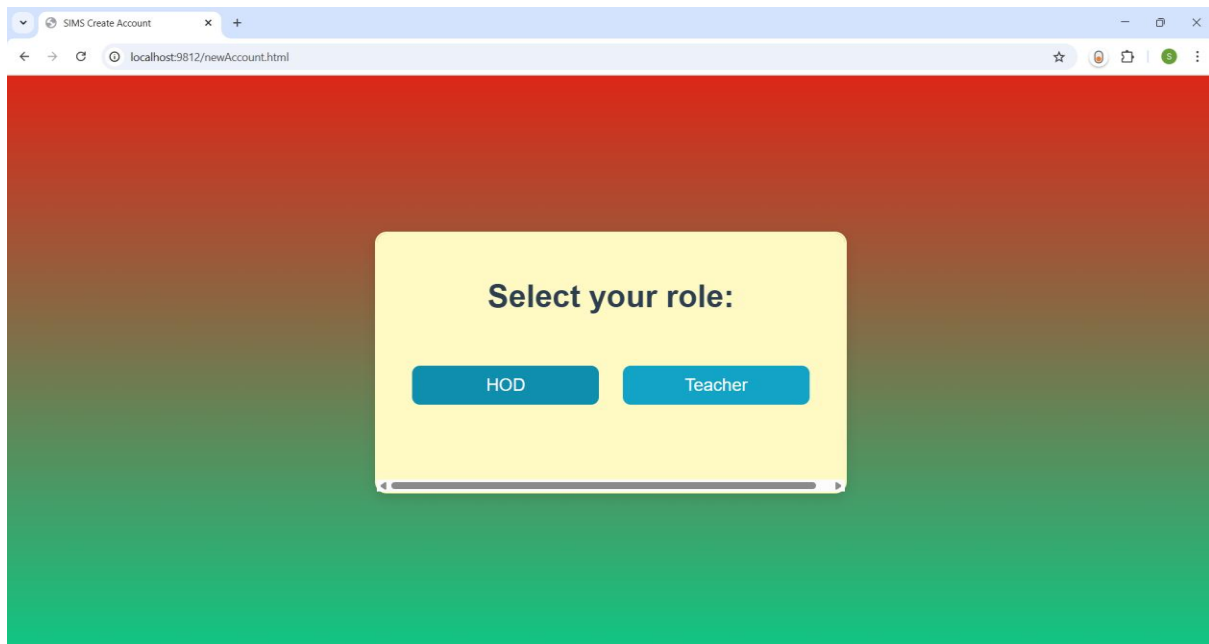


Fig 13: New Account Page

Based on user, the user needs to select the role such as Hod, Teacher and after selecting the user type you can click on it to register.

6.1.2.1 NEW ACCOUNT PAGE FOR HOD

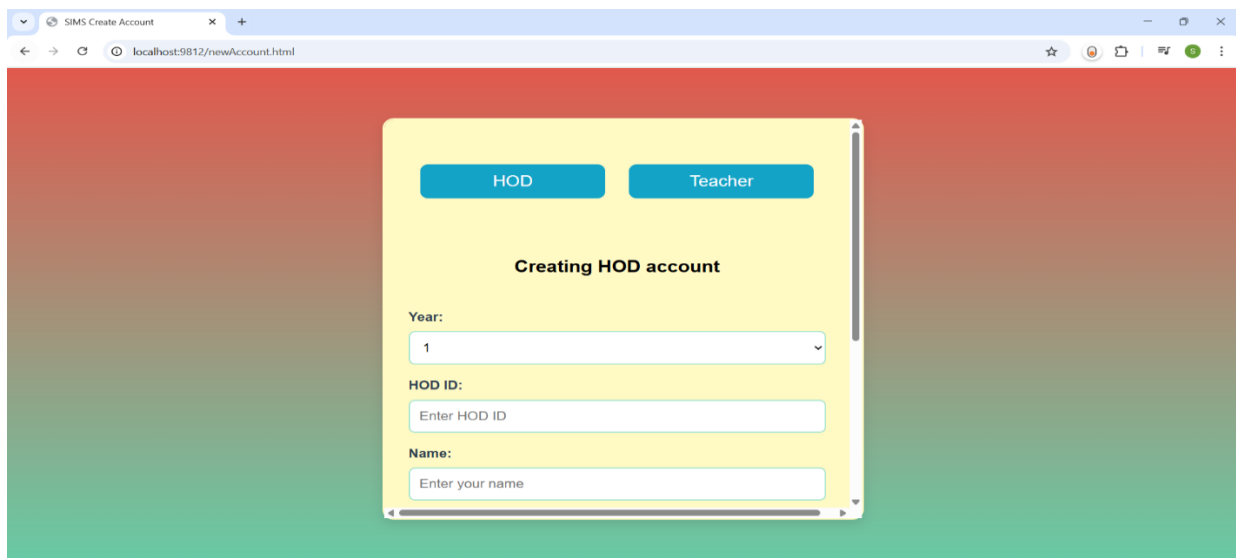


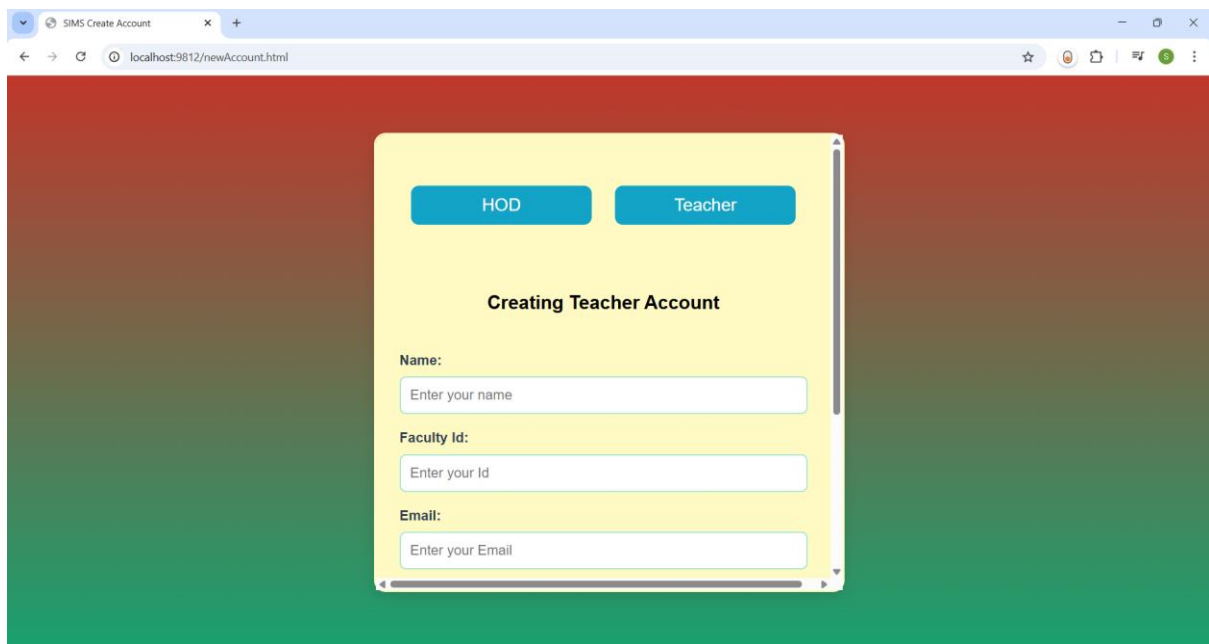
Fig 13 A: New Account Page For HOD

The image shows a web browser window with the title 'SIMS Create Account'. The address bar shows 'localhost:9812/newAccount.html'. The page has a red-to-green gradient background. In the center is a yellow rectangular form with a white border. Inside the form, there are several input fields: 'Enter HOD ID' at the top, followed by 'Name:' with 'Enter your name', 'Email:' with 'Enter your Email', 'Password:' with 'Enter Password', and 'Confirm Password:' with 'Re-Enter Password'. At the bottom of the form is a blue button labeled 'Create Account'.

Fig 13 B: New Account Page For HOD

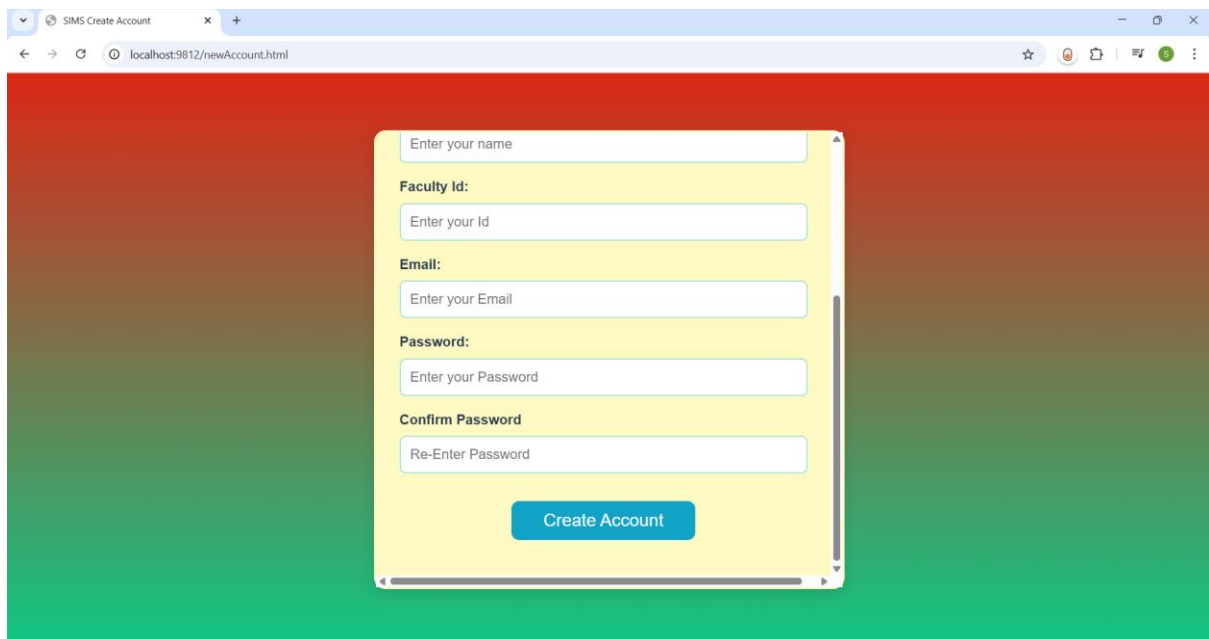
The new account page is used to register for new users to get their credentials for login into their page. This page contains the columns: Year, Hod id, Name, Email, Password, Confirm Password and Create Account button. In the year column you need to select any one option either 1 or 2,3,4 this is based on Hod i.e., the Hod can select the '1' if he/she is the hod of first years otherwise you can go with the 2,3,4 which indicates years. If you select the option 2,3,4 then you need to select the branch, which is a dropdown that contains different branches. In the name column you need to enter your full name, in Hod Id column you need enter the Id which is issued by the college. In the email column you need to enter the email and the password is up to the user there is no restriction on password but you need to enter the same password in both Password and Confirm Password column next click on Create Account button then a request will be sent to Admin for activation of your account.

6.1.2.2 NEW ACCOUNT PAGE FOR TEACHER/FACULTY



The screenshot shows a web browser window with the title 'SIMS Create Account' and the URL 'localhost:9812/newAccount.html'. The page has a red-to-green gradient background. A yellow modal form is centered, titled 'Creating Teacher Account'. At the top of the form are two blue buttons: 'HOD' and 'Teacher'. Below the title, there are three input fields: 'Name:' with placeholder 'Enter your name', 'Faculty Id:' with placeholder 'Enter your Id', and 'Email:' with placeholder 'Enter your Email'.

Fig 14 A: New Account Page for Teacher/Faculty



This screenshot shows the same yellow modal form as Fig 14 A, but with additional fields. Below the 'Email' field, there are two more input fields: 'Password:' with placeholder 'Enter your Password' and 'Confirm Password' with placeholder 'Re-Enter Password'. At the bottom of the form is a blue button labeled 'Create Account'.

Fig 14 B: New Account Page for Teacher/Faculty

The new account page is used to register for new users to get their credentials for login into their page. This page contains the columns: Name, Faculty Id, Email, Password, Confirm Password and Create Account button. In the name column you need to enter your full name,

in Faculty Id column you need enter the Id which is issued by the college. In the email column you need to enter the email and the password is up to the user there is no restriction on password but you need to enter the same password in both Password and Confirm Password column next click on Create Account button to create your faculty account.

6.1.3 LOGIN PAGE

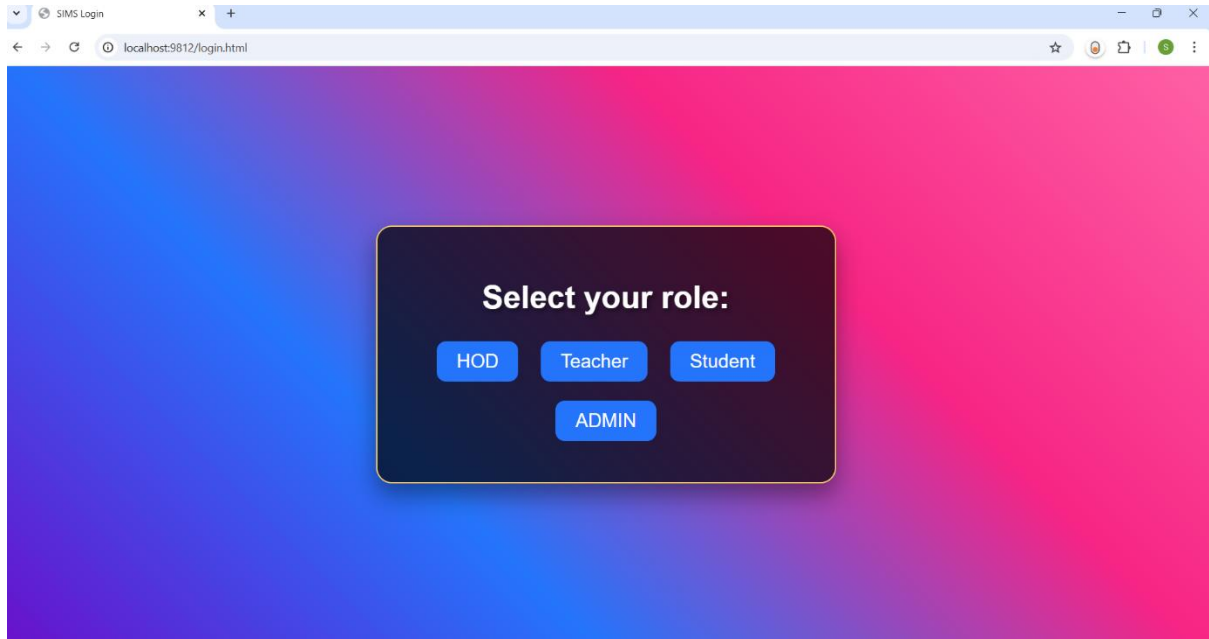


Fig 15: Login Page

Based on user, the user needs to select the role such as Hod, Teacher, Student, Admin and after selecting the user type you can login.

6.1.3.1 LOGIN PAGE FOR HOD

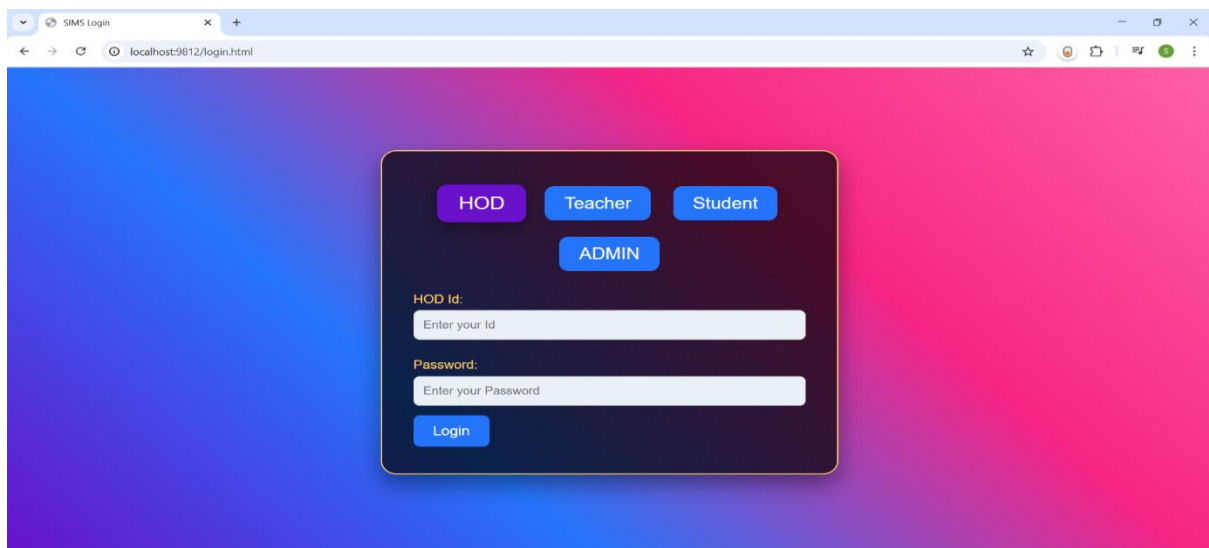


Fig 16: Login Page for HOD

The above shown screen is the login page for HOD. Where you can login into your page if you're the old user, if not you need to create account using the create account page. Here you need to enter the credentials: Hod Id and password for login. Make sure that your account is activated by Admin then only you can login. Once you login then you can perform various operations such as adding Branches, Subjects and exams, entering Student Details, viewing Faculty requests, marks update requests and approving or rejecting them, and generating student reports.

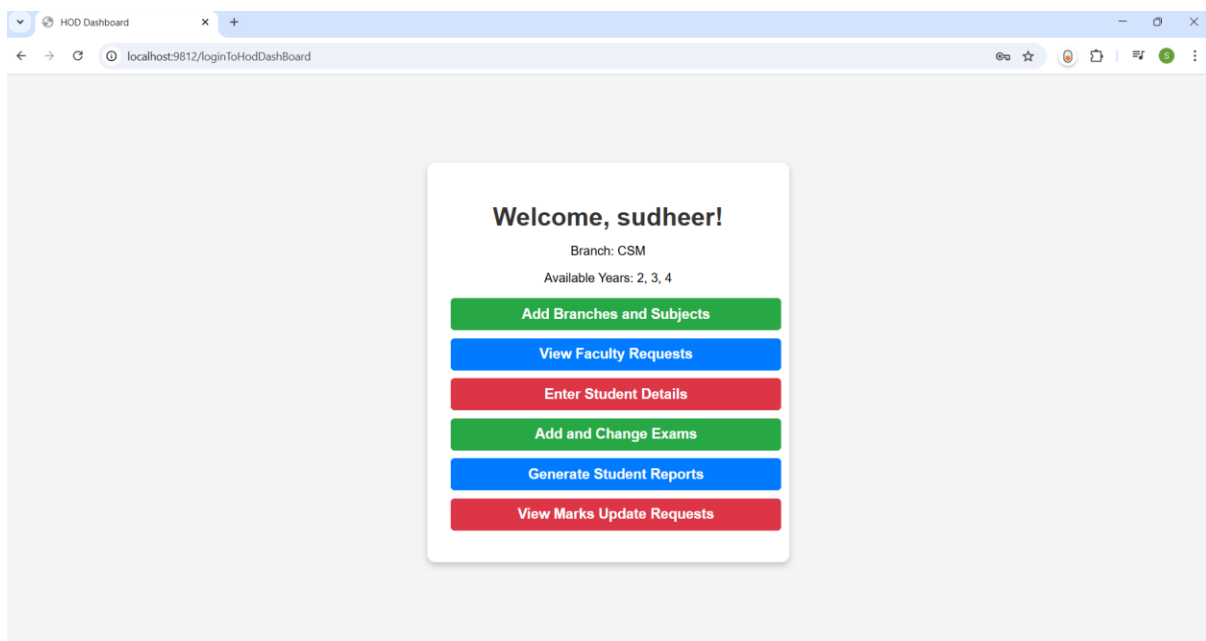


Fig 17: Dashboard of HOD

6.1.3.2 LOGIN PAGE FOR FACULTY

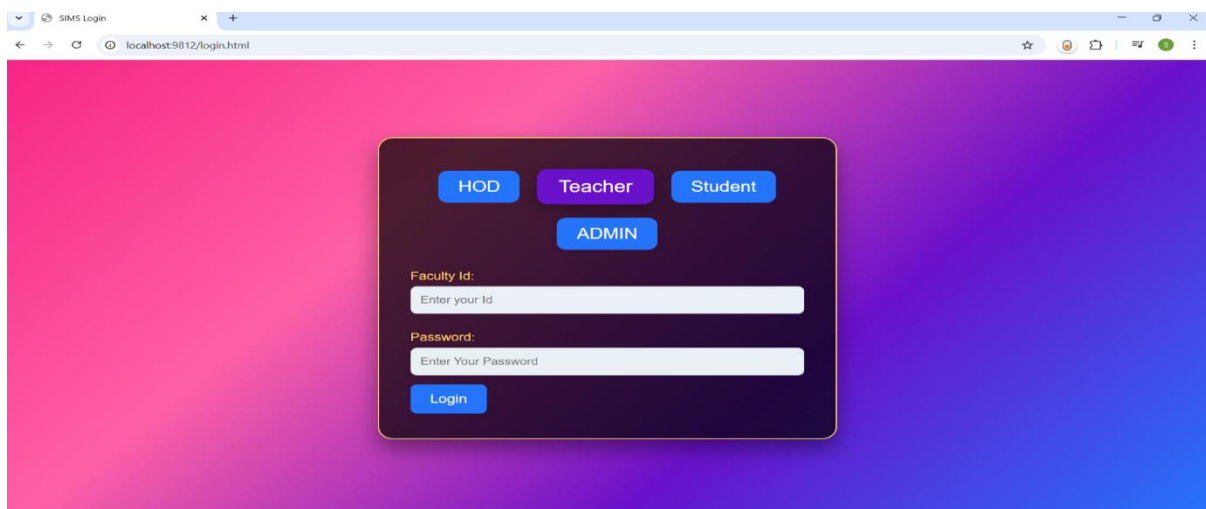


Fig 18: Login Page for Teacher

The above shown screen is the login page for TEACHER. Where you can login into your page if you're the old user, if not you need to create account using the create account page. Here you need to enter the credentials: Faculty Id and password for login. Once you login then you need to request HOD for subject.

Request Subject Allocation

Subject: NLP Branch: CSM Year: 3 Status: Approved Continue	Subject: DA Branch: CSM Year: 3 Status: Approved Continue
--	---

[Remove](#) [Send Request To Hod](#)

[+ Add Subject](#)

Fig 19: Subject Request Page for Faculty

If the HOD approves your request then you are enabled with the continue button and click on it to navigate to dashboard where you can perform the tasks like Enter Marks, View Marks, View Overall Marks and Edit Marks.

[Home](#) [Enter Marks](#) [View Marks](#) [View Overall Marks](#) [Edit Marks](#)

Welcome back sudheer!
Subject: NLP
Branch: CSM **Year:** 3

Unit Test-1	Mid-1
Assignment-1	Unit Test-2
Mid-2	Assignment-2

Fig 20: Faculty Home Page

6.1.3.3 LOGIN PAGE FOR STUDENT

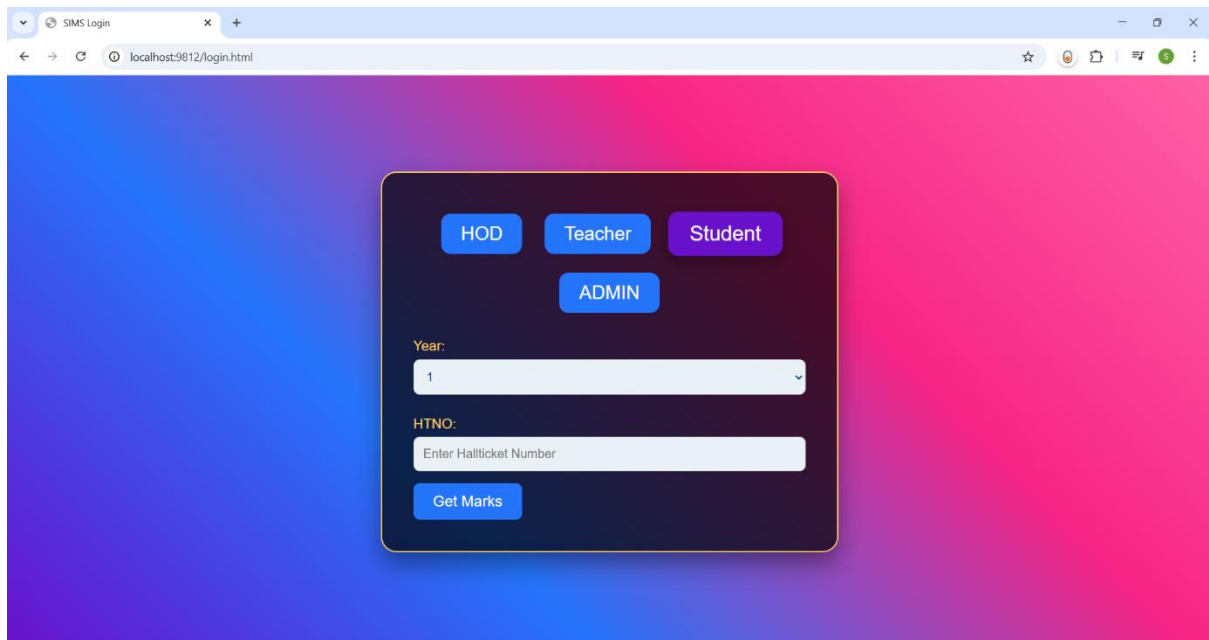
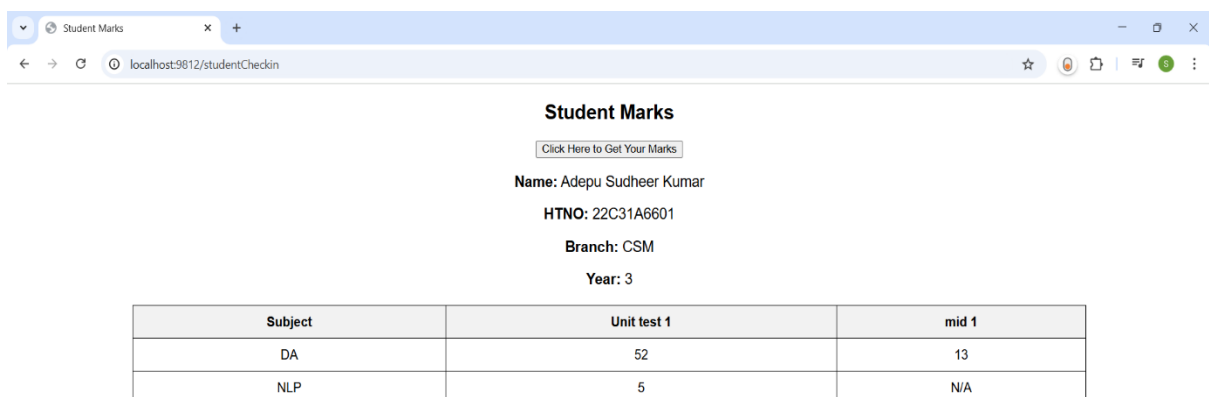


Fig 21: Login Page for Student

The student can login by selecting the year he/she belongs to and by entering the Hall ticket number, once you login you can able to view the marks of you by clicking on ‘click here to get marks’ button. This will be like as follows:



Subject	Unit test 1	mid 1
DA	52	13
NLP	5	N/A

Fig 22: Dashboard of Student

6.2 ADMIN MODULE

Admin: The admin is responsible for approving HOD registrations based on valid credentials. If an HOD or faculty member forgets their password, they must contact the admin for a reset.

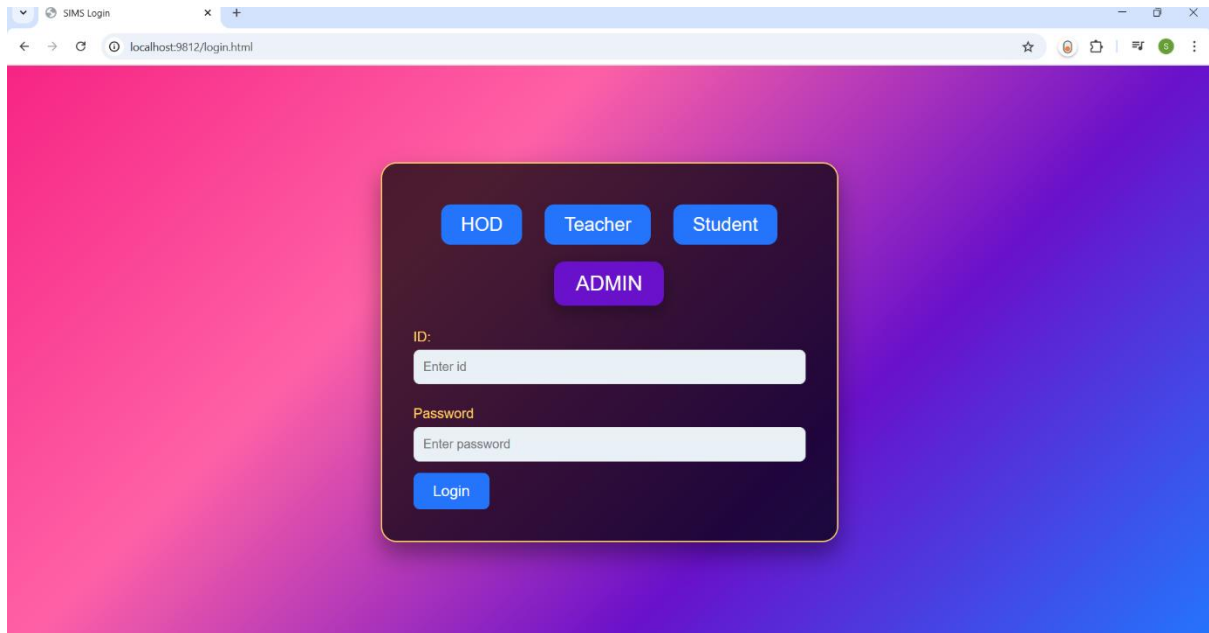


Fig 23: Login Page for Admin

The admin logs into the account by entering the credentials ID and Password. The admin can able to view the requests of HOD and Admin can approve or reject them. And Admin helps the Faculty, Hod's in resetting their passwords.



Fig 24: Dashboard Page for Admin

7. LANGUAGE SPECIFICATION

7.1 INTRODUCTION TO HTML

HTML, which stands for Hypertext Markup Language, is the standard markup language used to create and design web pages. It provides the structure and layout for content on the World Wide Web. Here's a basic introduction to HTML:

HTML Document Structure: An HTML document consists of elements, which are enclosed in tags. Tags are keywords surrounded by angle brackets, like `<tagname>`. The basic structure of an HTML document includes:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the Document</title>
  </head>
  <body>
    <!-- Content goes here -->
  </body>
</html>
```

HTML Elements: Elements are the building blocks of HTML documents. They are defined by tags and can contain other elements and content. Some common HTML elements include:

`<h1>`, `<h2>`, `<h3>`, ... `<h6>`: Heading elements, used to define headings of different levels.

`<p>`: Paragraph element, used to define paragraphs of text.

`<a>`: Anchor element, used to create hyperlinks.

``: Image element, used to embed images.

``, ``, ``: List elements, used to create unordered and ordered lists.

`<div>`, ``: Container elements, used to group and style content.

Attributes: Attributes provide additional information about HTML elements. They are always specified in the start tag and come in name/value pairs.

HTML is the foundation of web development, and understanding its basics is essential for anyone looking to create web pages or dive deeper into web development.

7.2 SECURITY ISSUES IN HTML

HTML itself doesn't inherently pose security risks, as it's primarily a markup language for structuring web content. However, when HTML is used in conjunction with other web technologies like CSS, JavaScript, and server-side scripting languages, security vulnerabilities can arise. Here are some common security issues related to HTML:

Cross-Site Scripting (XSS): XSS occurs when an attacker injects malicious scripts into web pages viewed by other users. This can happen if user input is not properly sanitized before being included in HTML output. For example, if a website allows users to submit comments without filtering out JavaScript code, an attacker could inject a script that steals session cookies or redirects users to a malicious website.

Cross-Site Request Forgery (CSRF): CSRF attacks exploit the trust that a website has in a user's browser. Attackers trick users into unknowingly submitting requests to a website they are authenticated with. This can lead to actions being performed on behalf of the user without their consent, such as changing account settings or making purchases. HTML forms are often used to carry out CSRF attacks by submitting requests to a targeted website.

HTML Injection: HTML injection occurs when an attacker is able to insert HTML code into a web page, typically through user input fields that are not properly validated or sanitized. This can lead to various security risks, such as defacement of web pages, phishing attacks, or the execution of malicious scripts.

Clickjacking: Clickjacking involves tricking users into clicking on malicious elements disguised as legitimate ones. HTML can be used to overlay invisible elements on top of clickable elements, making users unknowingly interact with the hidden content. Clickjacking can be used for various malicious purposes, including spreading malware, stealing sensitive information, or hijacking user clicks for financial gain.

HTML5 Security Features: While HTML5 introduces new features and capabilities for web development, it also introduces new security considerations. For example, HTML5 introduces APIs such as the Geolocation API and Web Storage API, which can expose sensitive user information if not used securely. Developers need to be aware of these features and implement proper security measures to mitigate potential risks.

7.3 HTML CONTROL FLOW

HTML itself doesn't have explicit control flow structures like programming languages such as JavaScript, Python, or Java. Instead, HTML is primarily used for structuring the content and presentation of a web page.

However, HTML can interact with other technologies, like JavaScript, to achieve control flow behavior. JavaScript is a scripting language that can be embedded within HTML documents using `<script>` tags. With JavaScript, you can dynamically manipulate HTML elements, respond to user interactions, and control the flow of execution based on conditions and events.

7.4 INTRODUCTION TO CSS

CSS, which stands for Cascading Style Sheets, is a stylesheet language used to control the presentation and styling of HTML documents. It enables web developers to define the appearance of web pages, including layout, colors, fonts, and other visual aspects. Here's an introduction to CSS:

Separation of Concerns: CSS allows for the separation of content (HTML) from presentation (CSS) and behavior (JavaScript), following the principle of separation of concerns. This separation makes it easier to maintain and update web pages since changes to styling can be made without altering the underlying HTML structure.

Syntax: CSS consists of a set of rules that define how HTML elements should be displayed. Each rule consists of a selector and one or more declarations enclosed in curly braces `{ }`. A declaration consists of a property and a value, separated by a colon `:`.

Types of Selectors: CSS offers various types of selectors to target specific HTML elements:

Element Selector: Selects HTML elements by their tag name, such as `h1`, `p`, `div`, etc.

Class Selector: Selects elements based on their class attribute, prefixed with a dot (`.`), such as `.class name`.

ID Selector: Selects a single element based on its ID attribute, prefixed with a hash (`#`), such as `#idname`.

Attribute Selector: Selects elements based on their attributes, such as `[attribute=value]`.

7.5 INTRODUCTION TO Node.js

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser. Node.js is primarily used for building fast and scalable network applications, as it utilizes an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js runs on the server-side to generate dynamic web pages, Node.js allows developers to use JavaScript for server-side scripting, producing dynamic web page content before the page is sent to the user's browser. This enables the use of a single programming language JavaScript across the entire web development stack.

A Simple Node.js Server

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello, I am a Node.js server!');
});
server.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

The above example creates a simple HTTP server in Node.js that listens on port 3000 and responds with "Hello, I am a Node.js server!" whenever accessed. Unlike PHP, where the script is embedded in an HTML file, Node.js operates as a standalone runtime environment.

Key Features of Node.js:

- **Asynchronous and Event-Driven:** Node.js uses an event-driven architecture, meaning it doesn't wait for operations to complete before moving to the next one. This makes it efficient for handling concurrent requests.
- **Single Programming Language:** JavaScript can be used both on the front-end and back-end, eliminating the need to learn multiple languages.
- **Non-Blocking I/O:** Unlike traditional web servers that handle requests synchronously, Node.js handles multiple requests simultaneously without blocking the execution thread.
- **Fast Execution:** Built on the V8 JavaScript engine, Node.js executes JavaScript code at high speed, making it ideal for real-time applications.

- **Scalability:** Node.js supports microservices and modular development, allowing applications to scale efficiently.

How Node.js Works:

When a request is made to a Node.js server, the following steps occur:

1. The server receives the request and passes it to an event queue.
2. The event loop processes the request asynchronously without blocking other tasks.
3. If the request involves database access or external API calls, Node.js utilizes non-blocking I/O to handle it efficiently.
4. Once the request is processed, the response is sent back to the client.

This approach allows Node.js to handle thousands of concurrent connections with minimal resource consumption.

Node.js in Web Development:

Node.js is commonly used in:

- **RESTful APIs:** Building APIs for mobile and web applications.
- **Real-Time Applications:** Chat applications, gaming platforms, and live collaboration tools.
- **Microservices:** Modular services that communicate with each other.
- **Streaming Applications:** Video and audio streaming services.
- **Serverless Computing:** Deploying cloud-based functions that execute on demand.

Node.js has revolutionized server-side development by allowing JavaScript to be used both on the client and server, reducing complexity and increasing performance. Its non-blocking I/O model, event-driven architecture, and scalability make it an excellent choice for modern web applications. With strong community support and continuous improvements, Node.js remains a powerful tool for full-stack JavaScript development.

7.6 SECURITY IN Node.js

Security vulnerabilities in Node.js applications can expose sensitive data, including usernames, passwords, and other critical information. If not properly secured, attackers can exploit weaknesses to manipulate databases, alter system files, or even take control of the entire application, leading to severe consequences such as data breaches, service downtime, or unauthorized system access.

How can it be prevented?

- **Input Validation and Sanitization:** Ensure that all user input is validated and sanitized before processing to prevent injection attacks and malicious payloads from being executed.
- **Secure Password Storage:** Always hash passwords using strong hashing algorithms such as bcrypt, Argon2, or PBKDF2 to prevent unauthorized access in case of data breaches.
- **Error Handling and Logging:** Avoid exposing technical details in error messages. Custom error messages should be used to prevent attackers from gaining insights into database structures or application logic.
- **Implement Access Control:** Use role-based access control (RBAC) to ensure that users have the minimum necessary permissions to interact with the system.
- **Use Environment Variables:** Store sensitive credentials (e.g., database connection strings, API keys) in environment variables instead of hardcoding them in the application.
- **Prevent SQL and NoSQL Injection:** Use parameterized queries or Object-Relational Mapping (ORM) libraries to prevent attackers from injecting malicious queries into the database.
- **Enable Security Headers:** Configure HTTP headers such as Content Security Policy (CSP), Strict-Transport-Security (HSTS), and X-Frame-Options to mitigate risks like cross-site scripting (XSS) and clickjacking.
- **Keep Dependencies Updated:** Regularly update all dependencies and use tools like npm audit or Snyk to identify and fix vulnerabilities in third-party packages.
- **Use Secure Authentication Methods:** Implement multi-factor authentication (MFA) and OAuth for better security when managing user authentication.
- **Limit Request Rate and Prevent DDoS Attacks:** Use rate-limiting middleware like express-rate-limit to prevent brute-force attacks and excessive API requests.
- **Use Secure Transport Layer:** Always enforce HTTPS using SSL/TLS to ensure encrypted communication between clients and the server.

- **Secure WebSockets and APIs:** Implement authentication and authorization mechanisms when using WebSockets or REST APIs to prevent unauthorized access.
- **Disable Unnecessary Features:** Remove or disable unused modules, APIs, and debug endpoints that could introduce security vulnerabilities.
- **Monitor and Audit Security Regularly:** Use security monitoring tools to detect threats and analyze logs to identify potential attacks.

By following these best practices, Node.js applications can be significantly more secure against various threats and vulnerabilities.

7.7 Database Integration in Node.js Applications

Node.js is a powerful JavaScript runtime that enables server-side programming, allowing developers to interact with databases for storing and retrieving data in dynamic web applications. Node.js works seamlessly with both relational and NoSQL databases, enabling developers to choose the best-suited database for their application's needs.

Databases in Node.js:

Like PHP, Node.js applications often rely on databases to store and manage structured or unstructured data. There are two primary types of databases commonly used in Node.js:

1. **Relational Databases:** Relational databases organize data in tables with rows and columns, and SQL (Structured Query Language) is used for querying and managing the data. Common relational databases used with Node.js include:
 - **MySQL:** An open-source relational database known for its speed, reliability, and ease of use.
 - **PostgreSQL:** An advanced, open-source relational database that emphasizes extensibility and SQL compliance.
 - **SQLite:** A lightweight, serverless database often used for smaller applications or as an embedded database.
 - **Microsoft SQL Server:** A robust database system by Microsoft, frequently used in enterprise applications.
2. **NoSQL Databases:** NoSQL databases are designed for handling unstructured, semi-structured, or large-scale data that may not fit neatly into tables. They offer more

flexibility in data storage and retrieval. Some popular NoSQL databases used in Node.js include:

- **MongoDB:** A widely-used NoSQL database that stores data in JSON-like documents, making it highly flexible and scalable.
- **Redis:** An in-memory key-value store often used for caching and fast data retrieval.
- **Cassandra:** A highly scalable NoSQL database designed for managing large volumes of data across distributed systems.
- **Firebase Firestore:** A cloud-based NoSQL database that provides real-time synchronization for web and mobile applications.

Node.js Database Connection:

In Node.js, database connections are typically managed using third-party libraries that provide an interface to interact with databases. The most common libraries include:

1. **MySQL:** The `mysql2` or `mysql` package is commonly used to connect Node.js applications to MySQL databases. It provides an easy-to-use API for querying, inserting, and manipulating data in MySQL databases.

Example of connecting to MySQL in Node.js using `mysql2`:

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'password',
  database: 'test_db'
});
connection.connect((err) => {
  if (err) throw err;
  console.log('Connected to the database!');
});
```

CRUD Operations in Node.js:

Once connected to a database, CRUD (Create, Read, Update, Delete) operations can be performed to manage data. Here's how you can perform basic CRUD operations in Node.js using different databases:

1. **Create:** Insert data into the database using INSERT statements in SQL databases or `.create()` in MongoDB.
2. **Read:** Query the database using SELECT statements in SQL databases or `.find()` in MongoDB to retrieve data.
3. **Update:** Modify existing data using UPDATE statements in SQL databases or `.updateOne()` in MongoDB.
4. **Delete:** Remove data using DELETE statements in SQL databases or `.deleteOne()` in MongoDB.

Security in Database Interaction:

To prevent common security vulnerabilities like SQL Injection and NoSQL Injection:

- **SQL Databases:** Use prepared statements or ORM libraries (e.g., Sequelize, TypeORM) that automatically handle parameterized queries to avoid SQL injection.
- **NoSQL Databases:** Sanitize user inputs and avoid directly using user input in database queries to prevent injection attacks.

Node.js offers great flexibility in working with both relational and NoSQL databases, making it suitable for a wide range of applications. By using appropriate database drivers and libraries, developers can efficiently integrate databases into their Node.js applications. Best practices such as using prepared statements, environment variables for sensitive data, and ORM tools will help ensure that the application remains secure and scalable.

7.8 NODE.JS CONTROL FLOW

Control flow in Node.js is similar to other programming languages, where you can execute code conditionally, repeatedly, or based on certain conditions. However, since Node.js is built on asynchronous non-blocking architecture, its control flow also emphasizes asynchronous programming patterns like callbacks, promises, and `async/await` in addition to the traditional control flow structures like conditionals and loops.

1. Conditional Statements

Node.js uses standard JavaScript control flow structures like if, else, and switch, which function as they do in other languages.

2. Loops

Node.js supports the typical loops you would find in JavaScript. The primary ones are for, while, do-while, and forEach (commonly used with arrays).

3. Switch Statement

The switch statement works in Node.js in a similar way to other languages. It's used to test a variable against multiple possible cases.

4. Asynchronous Control Flow

In Node.js, asynchronous operations are fundamental due to its non-blocking I/O model. Asynchronous functions allow the program to continue executing while waiting for operations (like file reading or database queries) to complete.

Callback Functions (Common in Older Code):

In Node.js, callback functions are often used for handling asynchronous operations. A callback is a function passed as an argument to another function, and it's executed when the asynchronous task completes.

Promises:

Promises provide a more readable way to handle asynchronous operations compared to callbacks, allowing you to chain .then() for success and .catch() for errors.

Async/Await (Modern Approach):

async/await is a more recent syntax in JavaScript/Node.js that makes working with promises easier by allowing asynchronous code to be written in a synchronous style.

7.9 INTRODUCTION TO SQL

SQL, or Structured Query Language, is a standard programming language used for managing and manipulating relational databases. It provides a set of commands for querying, updating, inserting, deleting, and managing data in databases. Here's an introduction to SQL:

Database Management Systems (DBMS): SQL is used in conjunction with Database Management Systems, which are software applications that enable users to interact with databases. Common DBMS platforms that support SQL include MySQL, PostgreSQL, SQLite, Oracle Database, Microsoft SQL Server, and many others.

Relational Databases: SQL is primarily used with relational databases, which organize data into tables consisting of rows and columns. Each column in a table represents a different attribute or field, while each row represents a unique record or entry. Tables can be related to each other through defined relationships, such as one-to-one, one-to-many, or many-to-many relationships.

SQL Commands:

Data Query Language (DQL):

SELECT: Retrieves data from one or more tables based on specified criteria.

Data Manipulation Language (DML):

INSERT: Adds new records to a table.

UPDATE: Modifies existing records in a table.

DELETE: Removes records from a table.

Data Definition Language (DDL):

CREATE TABLE: Creates a new table in the database.

ALTER TABLE: Modifies the structure of an existing table.

DROP TABLE: Deletes a table and its data from the database.

Data Control Language (DCL):

GRANT: Grants specific permissions to users or roles.

REVOKE: Revokes previously granted permissions.

SQL Syntax:

SQL statements are composed of keywords, identifiers, operators, and clauses.

Keywords are reserved words in SQL, such as SELECT, FROM, WHERE, INSERT, UPDATE, DELETE, etc.

Identifiers are names given to tables, columns, or other database objects, and they should follow specific naming rules.

Operators like comparison operators ($=$, $<>$, $>$, $<$, $>=$, $<=$), logical operators (AND, OR, NOT), arithmetic operators (+, -, *, /), etc., are used to perform operations in SQL queries.

Clauses are components of SQL statements that perform specific actions, such as WHERE, ORDER BY, GROUP BY, HAVING, JOIN, etc.

SQL Constraints: Constraints are rules applied to columns to enforce data integrity and

maintain consistency in the database. Common constraints include PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK, etc.

SQL Joins: SQL Joins are used to combine rows from two or more tables based on related columns between them. Common types of joins include INNER JOIN, LEFT JOIN (or LEFT OUTER JOIN), RIGHT JOIN (or RIGHT OUTER JOIN), and FULL JOIN (or FULL OUTER JOIN).

SQL is a powerful and widely-used language for managing and querying relational databases. Understanding SQL basics is essential for anyone working with databases, whether as a developer, data analyst, database administrator, or data scientist.

7.10 INTRODUCTION JAVASCRIPT

JavaScript is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

Client-Side Scripting: JavaScript is primarily executed on the client-side (i.e., in the user's web browser), allowing for dynamic manipulation of web page content after it has been loaded. This enables developers to create interactive user interfaces, respond to user actions, and modify the page structure and styling in real-time without needing to reload the entire page.

Syntax and Structure: JavaScript syntax is similar to other programming languages like Java and C, making it relatively easy to learn for those familiar with programming concepts.

Data Types and Variables: JavaScript supports various data types, including numbers, strings, Booleans, arrays, objects, functions, and more. Variables are declared using var, let, or const keywords, with let and const introduced in newer versions of JavaScript (ES6).

8. SAMPLE CODE

8.1 SAMPLE HTML CODE FOR LOGIN PAGE

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>SIMS Login</title>

<link href="loginStyle.css" rel="Stylesheet"/>

</head>

<body>

<div id="loginBox" class="loginBox">

<h1>Select your role:</h1>

<div id="buttons" class="loginBox">

<button id="hod" class="loginBox" onclick="switchTab('allDetailsOfHod')">HOD</button>

<button id="teacher" class="loginBox" onclick="switchTab('allDetailsOfTeacher')">Teacher</button>

<button id="student" class="loginBox" onclick="switchTab('allDetailsOfStudent')">Student</button>

<button id="Admin" class="loginBox" onclick="switchTab('allDetailsOfAdmin')">ADMIN</button>

</div>

<!-- HOD details -->

<form id="hodForm" class="allDetails allDetailsOfHod" action="/loginToHodDashBoard" method="POST">

<label for="hodId">HOD Id:</label>

<input type="text" id="HodId" name="HodId" placeholder="Enter your Id" required />

<label for="passwordOfHod">Password:</label>

<input type="password" id="passwordOfHod" name="passwordOfHod" placeholder="Enter your Password" required />

<p id="hodError" style="color: red; display: none;"></p> <!-- Error message placeholder -->

<input type="submit" value="Login" />
```

```

</form>

<!-- Teacher details -->

<form id="teacherForm" class="allDetails allDetailsOfTeacher" action="/TeacherLogin"
method="post" >

<label for="facultyId">Faculty Id:</label>

<input type="text" id="facultyId" name="facultyId" placeholder="Enter your Id" required/>

<label for="passwordOfTeacher">Password:</label>

<input type="password" id="passwordOfTeacher"
name="passwordOfTeacher" placeholder="Enter Your Password" required />

<input type="submit" value="Login"/>

</form>

<!-- student details -->

<form id="studentForm" class="allDetails allDetailsOfStudent" method="post"
action="/studentCheckin">

<label for="year">Year:</label>

<select name="year" id="year" required>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

</select>

<label for="htno">HTNO:</label>

<input type="text" id="htno" name="htno" placeholder="Enter Hallticket Number" required/>

<input type="submit" value="Get Marks" />

</form>

<form id="adminForm" class="allDetails allDetailsOfAdmin" method="post"
action="/adminLogin">

<label for="idOfAdmin">ID:</label>

<input type="text" id="idOfAdmin" name="idOfAdmin" placeholder="Enter id" required/>

<label for="passwordOfAdmin">Password</label>

<input type="password" id="passwordOfAdmin" name="passwordOfAdmin"
placeholder="Enter password"/>

```

```
<input type="submit" value="Login"/>
</form>
</div>
<script src="loginScript.js"></script>
</body>
</html>
```

8.2 SAMPLE JS CODE FOR LOGIN PAGE

```
function switchTab(role) {
    entiredata = document.querySelectorAll(".allDetails");
    entiredata.forEach((element) => {
        element.style.display = "none";
    });
    selectedData = document.querySelector(`.${role}`);
    selectedData.style.display = "block";
    heading = document.getElementsByTagName("h1");
    heading[0].style.display="none";
}
```

9. RESULTS

9.1 HOME PAGE

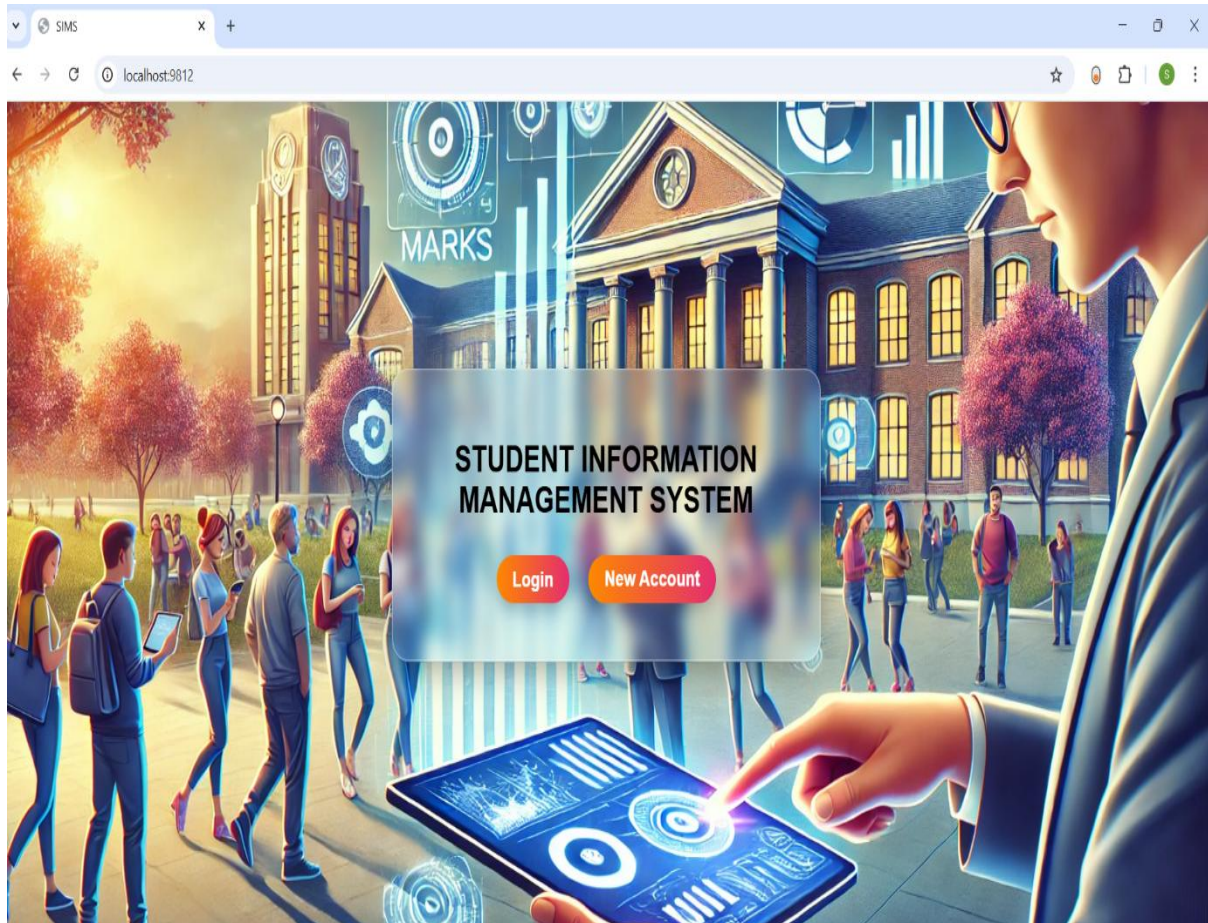


Fig 25: Home page

The above shown screen is the home page of our proposed project “**Student Information Management System** “. Where you can login if you’re the old user. If you are new user then you need to click on New Account for registering, and then login into their page.

9.2 NEW ACCOUNT PAGE

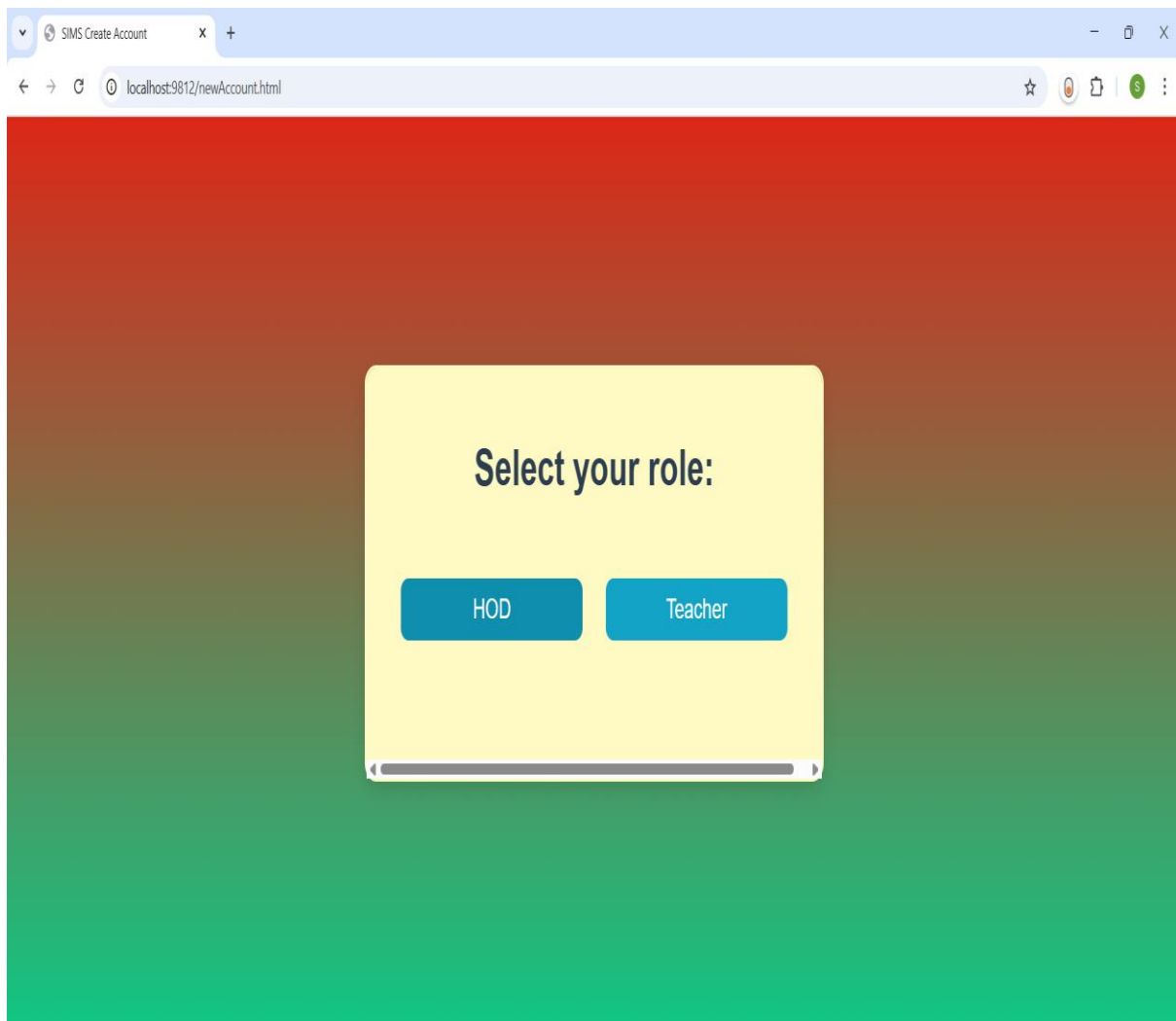
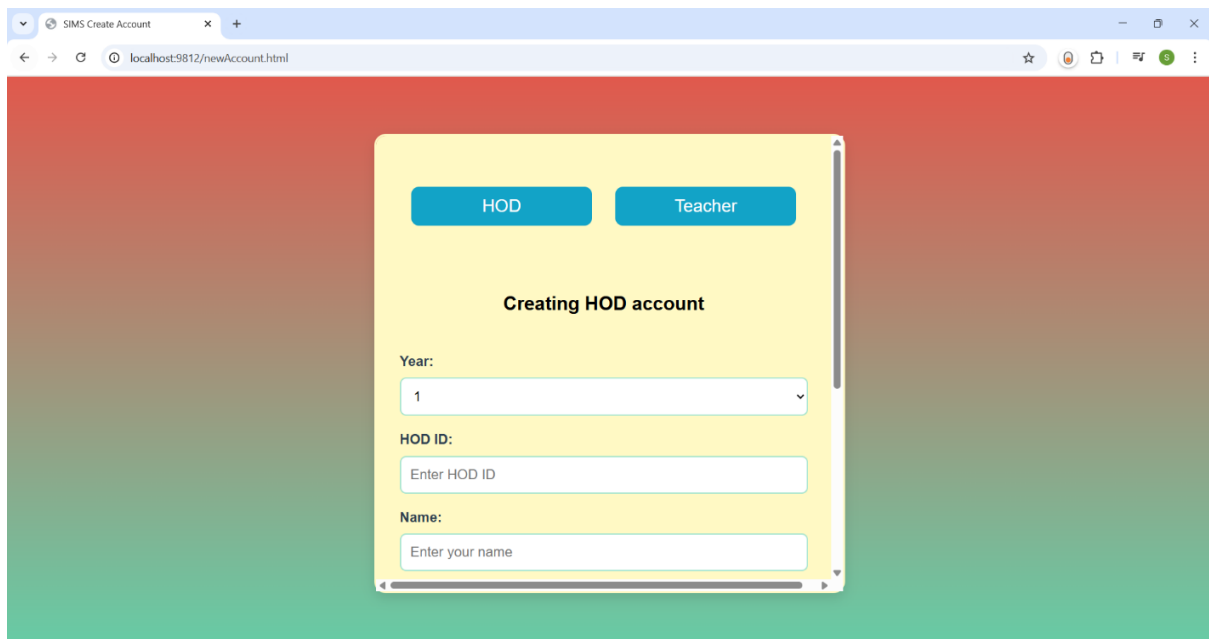


Fig 26: New Account

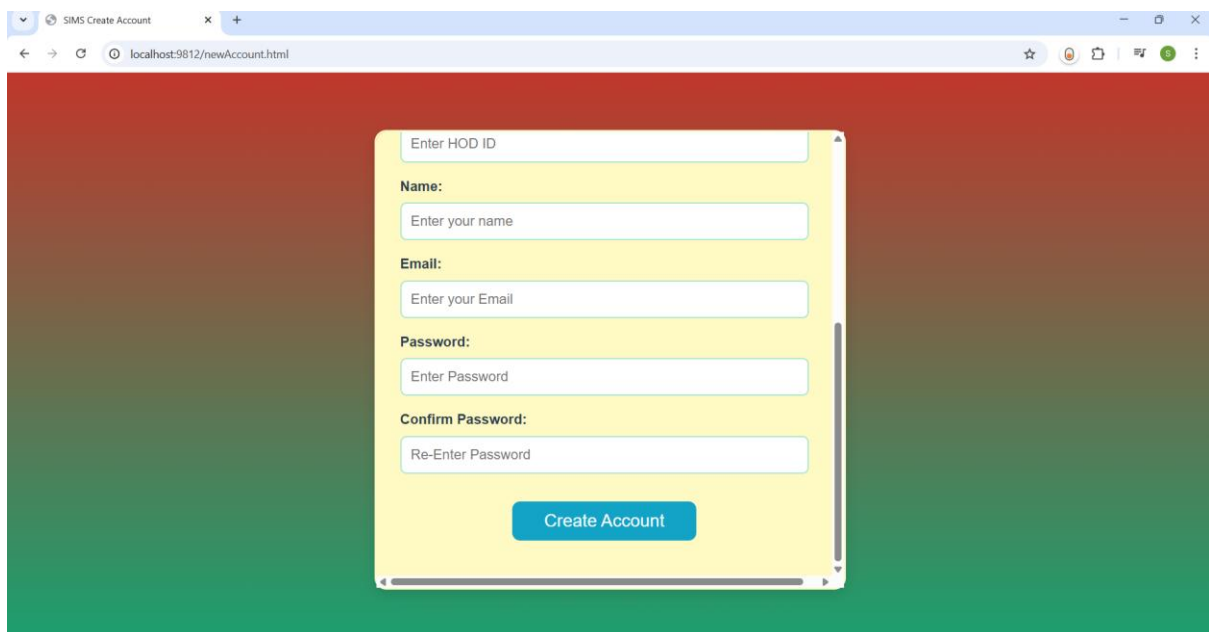
Based on user, the user needs to select the role such as Hod, Teacher and after selecting the user type you can click on it to register.

9.2.1 NEW ACCOUNT PAGE FOR HOD



The screenshot shows a web browser window with the address bar displaying 'localhost:9812/newAccount.html'. The page has a red header and a green footer. A yellow modal form titled 'Creating HOD account' is centered. At the top of the form are two blue buttons: 'HOD' and 'Teacher'. Below the title, there are three input fields: a dropdown menu for 'Year' with '1' selected, a text input for 'HOD ID' with placeholder text 'Enter HOD ID', and a text input for 'Name' with placeholder text 'Enter your name'.

Fig 27 A: New Account Page for HOD



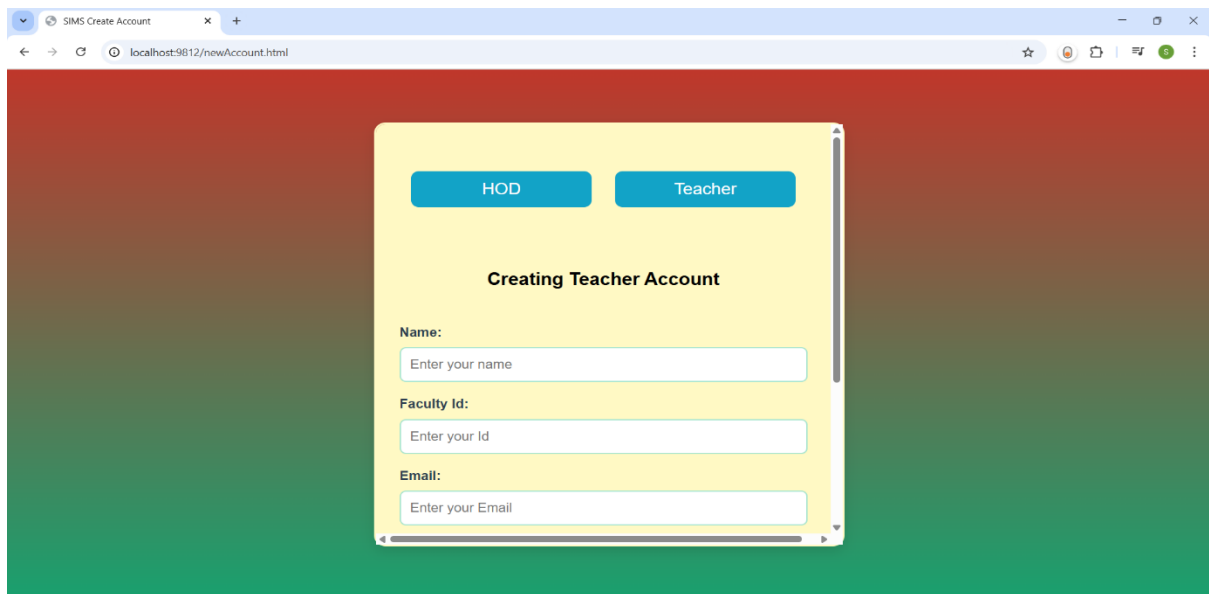
The screenshot shows the same web browser window, but the yellow modal form is scrolled down. It now displays the 'Email' input field with placeholder text 'Enter your Email', the 'Password' input field with placeholder text 'Enter Password', and the 'Confirm Password' input field with placeholder text 'Re-Enter Password'. At the bottom of the form is a blue button labeled 'Create Account'.

Fig 27 B: New Account Page for HOD

The new account page is used to register for new users to get their credentials for login into their page. This page contains the columns: Year, Hod id, Name, Email, Password, Confirm Password and Create Account button. In the year column you need to select any one option either 1 or 2,3,4 this is based on Hod i.e., the Hod can select the '1' if he/she is the hod of first years otherwise you can go with the 2,3,4 which indicates years. If you select the option

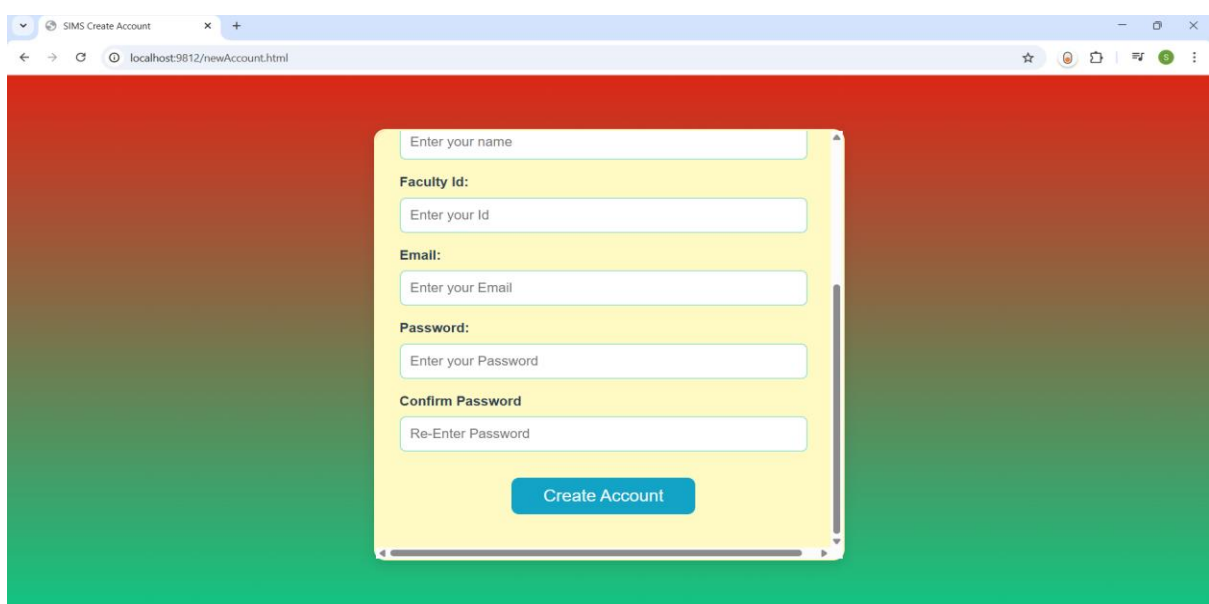
2,3,4 then you need to select the branch, which is a dropdown that contains different branches. In the name column you need to enter your full name, in Hod Id column you need enter the Id which is issued by the college. In the email column you need to enter the email and the password is up to the user there is no restriction on password but you need to enter the same password in both Password and Confirm Password column next click on Create Account button then a request will be sent to Admin for activation of your account.

9.2.2 NEW ACCOUNT PAGE FOR TEACHER/FACULTY



The screenshot shows a web browser window with the address bar displaying 'localhost:9812/newAccount.html'. The page has a red-to-green gradient background. A yellow modal form titled 'Creating Teacher Account' is centered. At the top of the form are two blue buttons: 'HOD' and 'Teacher'. Below the title, there are three input fields: 'Name:' with placeholder 'Enter your name', 'Faculty Id:' with placeholder 'Enter your Id', and 'Email:' with placeholder 'Enter your Email'.

Fig 28 A: New Account Page for Faculty



This screenshot shows the same 'Creating Teacher Account' form, but with additional fields. Below the 'Email' field, there are 'Password:' and 'Confirm Password' sections, each with an input field (placeholders: 'Enter your Password' and 'Re-Enter Password' respectively). At the bottom of the form is a blue 'Create Account' button.

Fig 28 B: New Account Page for Faculty

The new account page is used to register for new users to get their credentials for login into their page. This page contains the columns: Name, Faculty Id, Email, Password, Confirm Password and Create Account button. In the name column you need to enter your full name, in Faculty Id column you need enter the Id which is issued by the college. In the email column you need to enter the email and the password is up to the user there is no restriction on password but you need to enter the same password in both Password and Confirm Password column next click on Create Account button to create your faculty account.

9.3 LOGIN PAGE

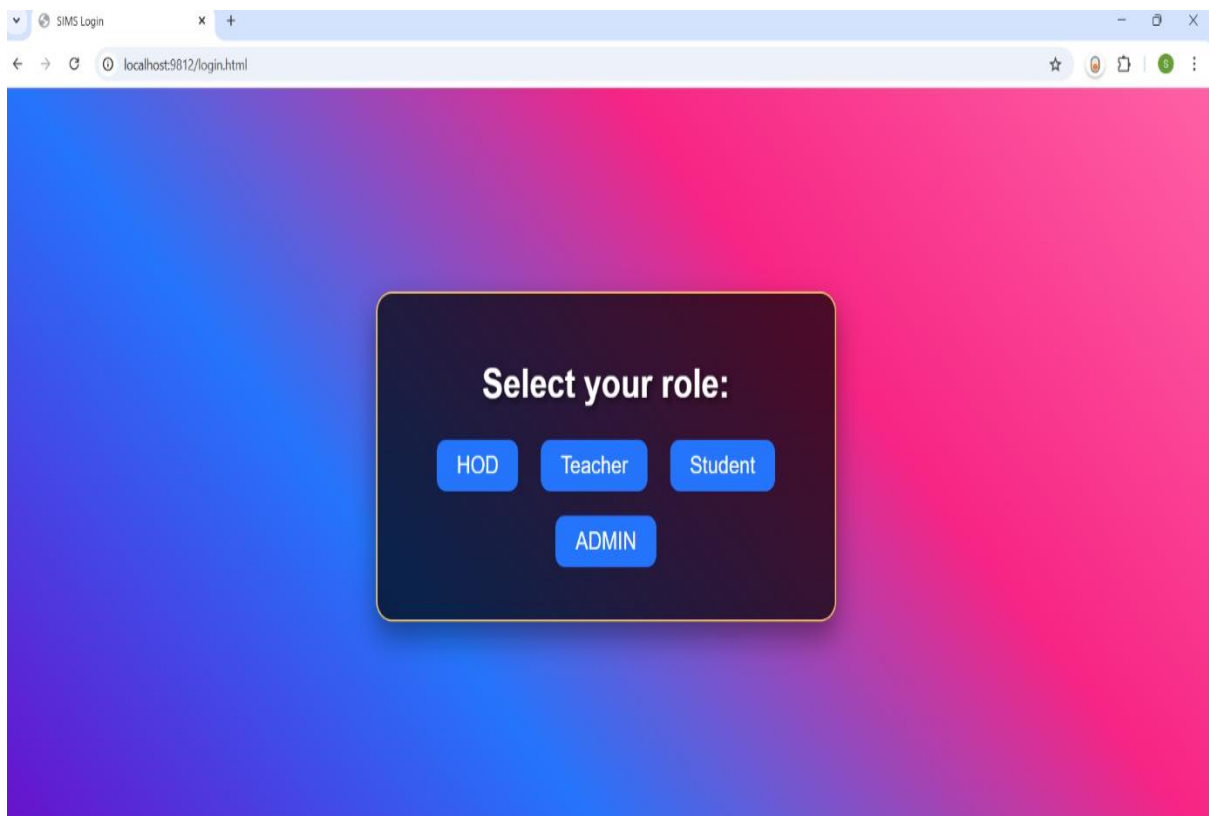


Fig 29: Login Page

Based on user, the user needs to select the role such as Hod, Teacher, Student, Admin and after selecting the user type you can login.

9.3.1 LOGIN PAGE FOR HOD

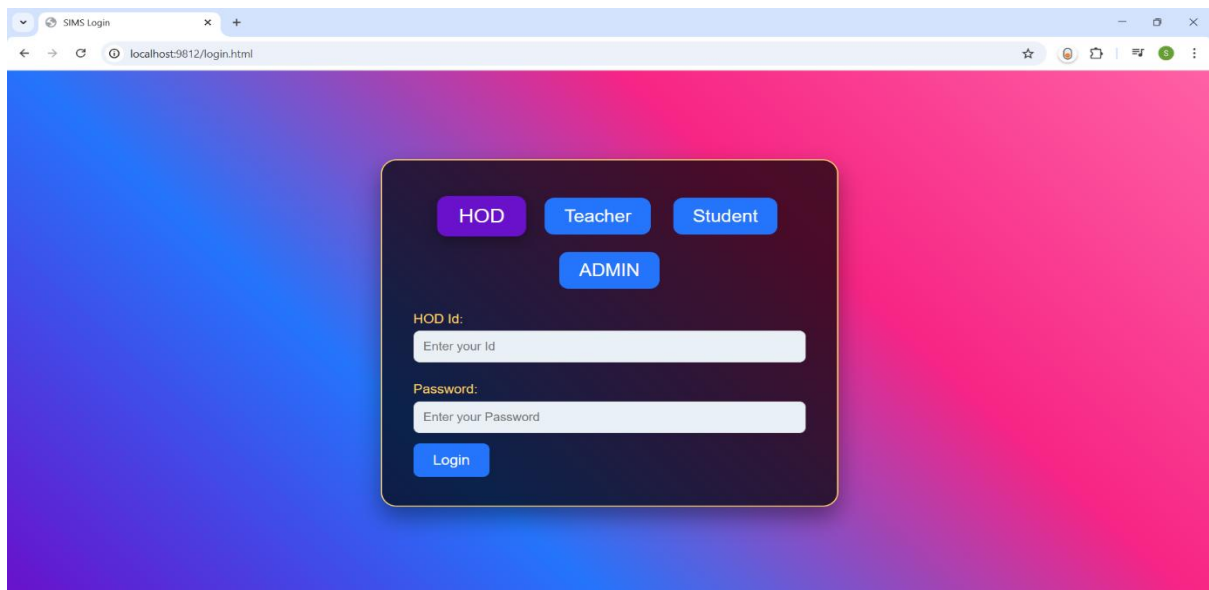


Fig 30: Login Page for HOD

The above shown screen is the login page for HOD. Where you can login into your page if you're the old user, if not you need to create account using the create account page. Here you need to enter the credentials: Hod Id and password for login. Make sure that your account is activated by Admin then only you can login. Once you login then you can perform various operations as follows:

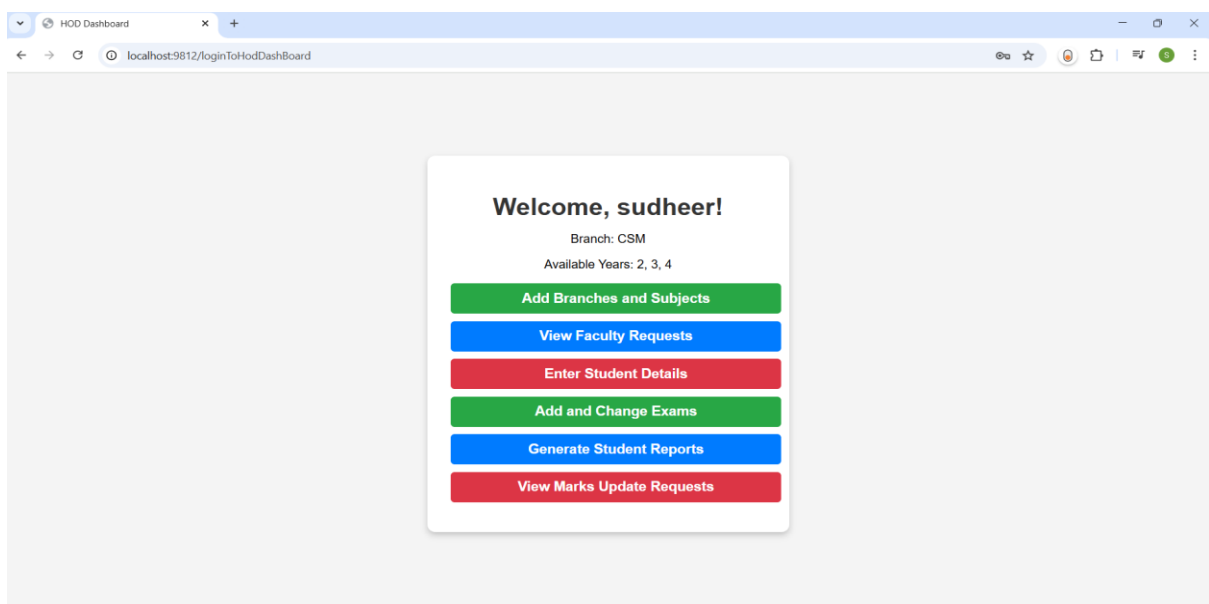


Fig 31: Dashboard Page for HOD

9.3.1.1 ADD BRANCHES AND SUBJECTS PAGE FOR HOD

Manage Subjects & Branches

Select Year: Choose Year ▼

Branches

+ Add Branch

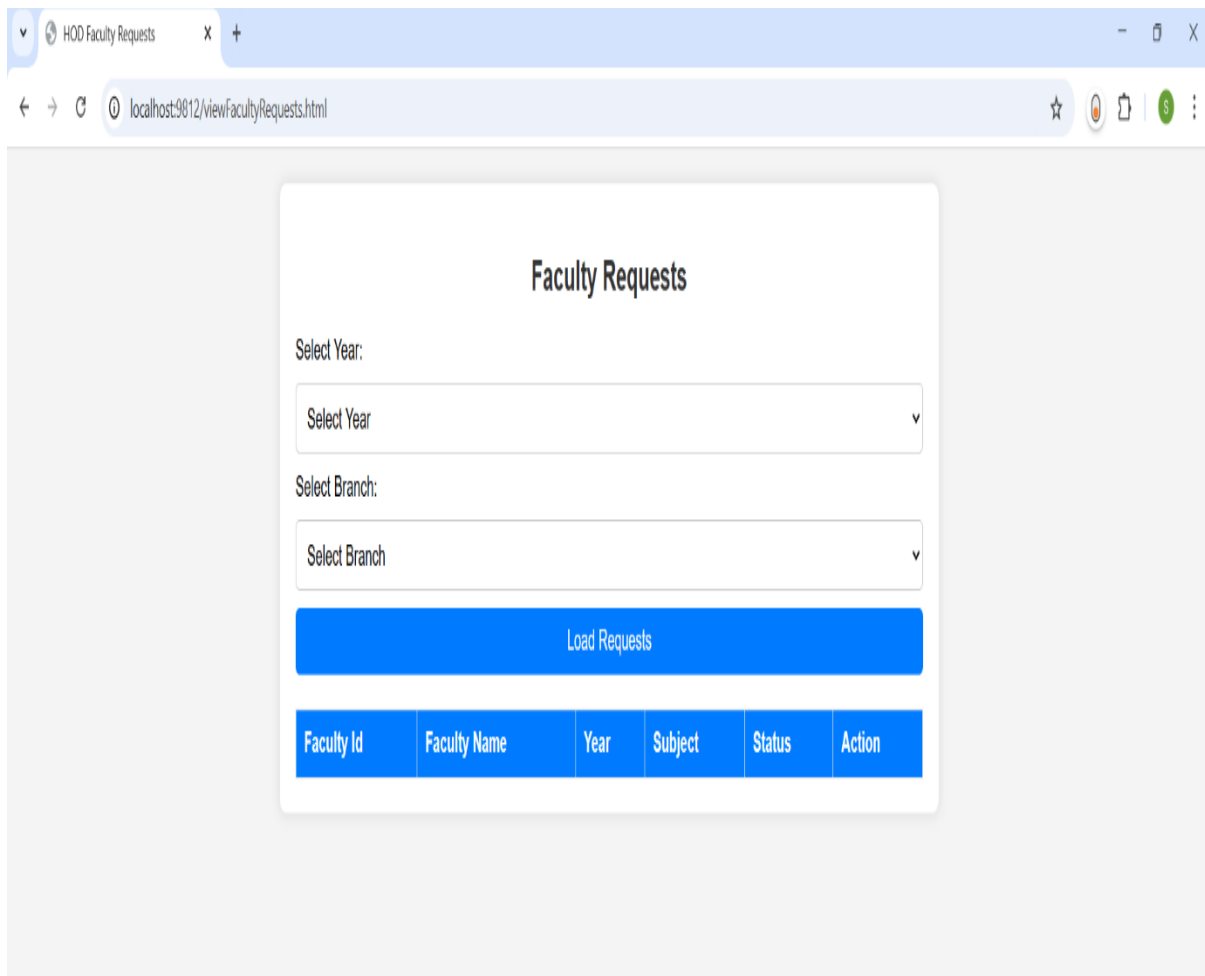
Subjects

+ Add Subject Save to Database

Fig 32: Add Branches and Subjects Page for HOD

In this page the HOD needs to select the Year and Branch then add the subjects then click on 'Save to database' button to save the subjects of a particular year and branch. Here the HOD needs to add subjects of all sections in his/her branch. These subjects will reflect at faculty request page, where they can request HOD for subject allocation.

9.3.1.2 VIEW FACULTY REQUESTS PAGE FOR HOD



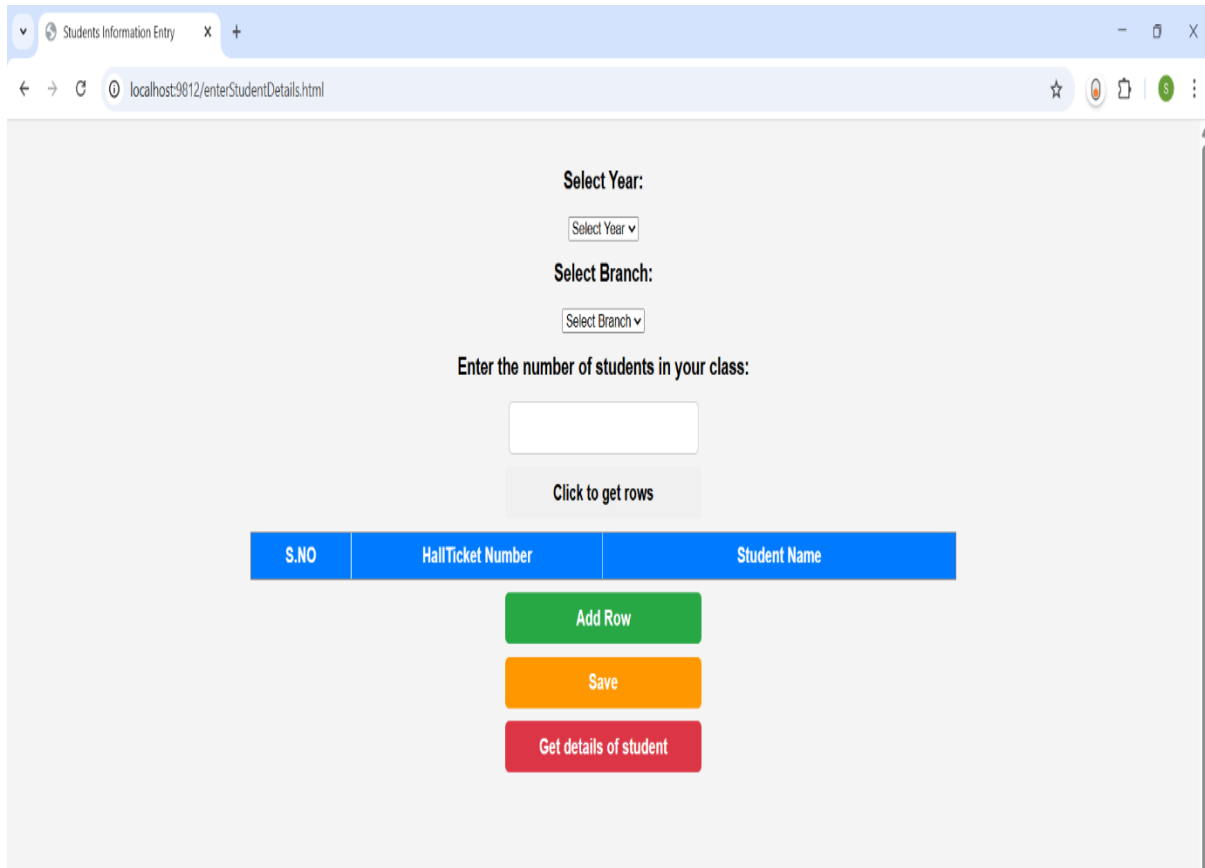
The screenshot displays a web browser window with the title 'HOD Faculty Requests'. The address bar shows 'localhost:9812/viewFacultyRequests.html'. The main content area features a white card with the heading 'Faculty Requests'. Below the heading are two dropdown menus: 'Select Year' and 'Select Branch'. A blue button labeled 'Load Requests' is positioned below the dropdowns. At the bottom of the card is a table header with the following columns: Faculty Id, Faculty Name, Year, Subject, Status, and Action.

Faculty Id	Faculty Name	Year	Subject	Status	Action
------------	--------------	------	---------	--------	--------

Fig 33: View Faculty Requests Page for HOD

In this page, the HOD can able to see the requests of faculty for subject allocation. The HOD needs to select the year and branch and click on the button 'Load Requests' for viewing the requests of faculty under this particular branch and year. The HOD can approve or reject the request, if approves then faculty can able to perform operations like Entering, viewing, updating marks, otherwise they can't. This is to ensure that only approved faculty can access the subject and student records.

9.3.1.3 ENTER STUDENT DETAILS PAGE FOR HOD



Select Year:

Select Year ▾

Select Branch:

Select Branch ▾

Enter the number of students in your class:

Click to get rows

S.NO	HallTicket Number	Student Name
------	-------------------	--------------

Add Row

Save

Get details of student

Fig 34: Enter Student Details Page for HOD

This page is used for entering the student details such as Hall ticket number and Name. These student details will appear for faculty for entering the marks. Firstly, the HOD needs to select the branch, year and number of students in the class and finally click on the button 'Click to get rows', then a table will be appeared with the specified rows, now HOD can enter the student details. If HOD needs to add one more row at the bottom of the table then he/she needs to click on the 'Add Row' button, click on 'Save' to save the student details in the database. If HOD wants to see the details of student whatever has already entered, then click on 'Get details of student', if there were no student details then nothing happens.

9.3.1.4 ADD AND CHANGE EXAMS PAGE FOR HOD



Manage Exams (Add & Remove Exams)

Select Year: Select Branch: Enter New Exam Name:

Existing Exams:

Exam Name	Action
Unit_test_1	<input type="button" value="Remove"/>
Mid_1	<input type="button" value="Remove"/>

Fig 35: Add and Change Exams Page for HOD

This page to add new exams for students based on need and remove previous exams if they are not required. HOD needs to select the year then branch, and enter the Exam name and click on Add Exam button to add to database. These exams will be shown as a dropdown for faculty to select and enter marks. If HOD wants to remove any exam then click on Remove button, before clicking on remove button make sure that there will be no use of that exam or no marks entered for that exam because once you click on remove then all data associated with that exam will be lost.

9.3.1.5 GENERATE STUDENTS REPORT PAGE FOR HOD

HOD - Generate Student Reports

Select Year: ~Select Year~ Select Branch: ~Select Branch~ Select Subject: ~Select Subject~ Select Exam: ~Select Exam~

Fetch Student Reports

Filter students who got less than: Enter marks Apply Filter

Print Report

Fig 36: Generate Students Report Page for HOD

This page is used for retrieving the student details along with their marks in a exam of a subject. The HOD needs to select the year, branch, subject, exam and click on Fetch Student Details to get Student data. There is a button called Apply Filter, which is used to filter students' data, for example, if HOD wants the details of Student who got less than 20 marks in a exam then HOD can make use of it. And there is an print option which allows HOD to print the student details.

9.3.1.6 VIEW MARKS UPDATE REQUESTS PAGE FOR HOD

The screenshot shows a web browser window with the title 'HOD Faculty Requests'. The address bar shows 'localhost:5812/viewMarksUpdateRequests.html'. The page content is titled 'Faculty Requests' and includes two dropdown menus: 'Select Year:' with '3 Year' selected, and 'Select Branch:' with 'CSM' selected. Below these is a blue 'Load Requests' button. A table displays the loaded requests:

Faculty ID	Subject	Exam	Requested At	Status	Action	View
777	DA	Unit_test_1	2025-03-30T11:12:17.000Z	Approved	<button>Approve</button> <button>Reject</button>	<button>View</button>
777	DA	Mid_1	2025-03-30T13:55:08.000Z	Approved	<button>Approve</button> <button>Reject</button>	<button>View</button>
777	NLP	Unit_test_1	2025-03-30T13:55:32.000Z	Pending	<button>Approve</button> <button>Reject</button>	<button>View</button>

Fig 37: Generate Students Report Page for HOD

This page is used to display the faculty marks update requests, for this the HOD needs to select the year, branch and click on Load requests button to get requests. HOD can approve or reject the request by viewing the student's data by clicking on view button. If HOD approves then old marks will be updated with new marks otherwise no changes will be done.

9.3.2 LOGIN PAGE FOR TEACHER/FACULTY

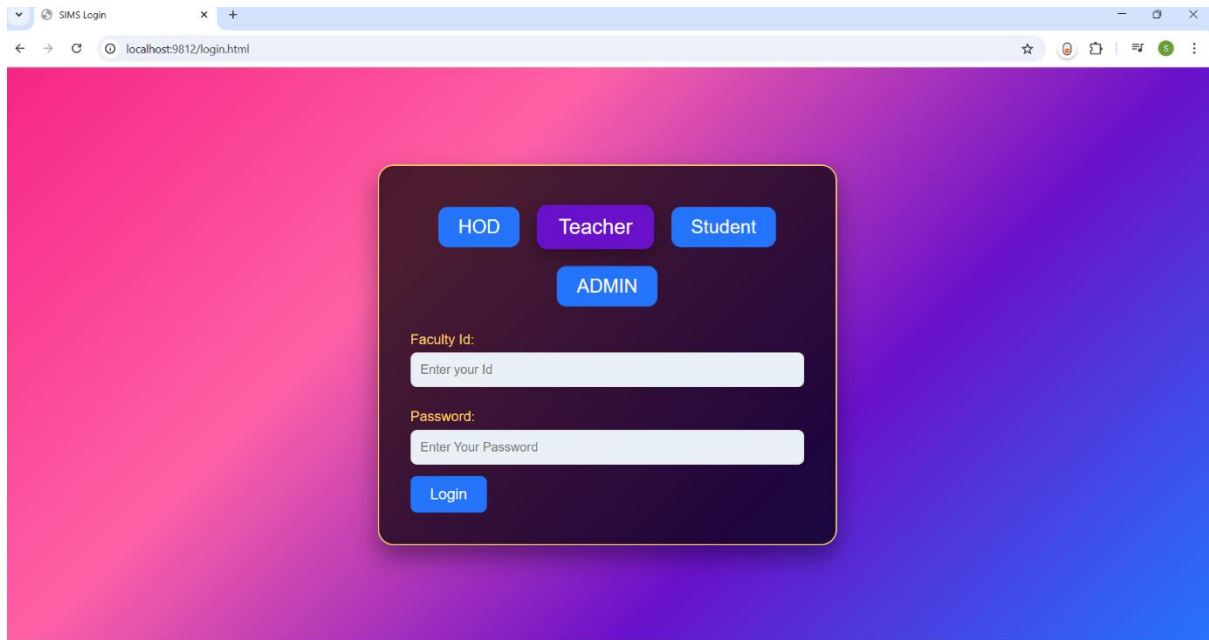


Fig 38: Login Page for Faculty

The above shown screen is the login page for TEACHER. Where you can login into your page if you're the old user, if not you need to create account using the create account page. Here you need to enter the credentials: Faculty Id and password for login. Once you login then you need to request HOD for subject.

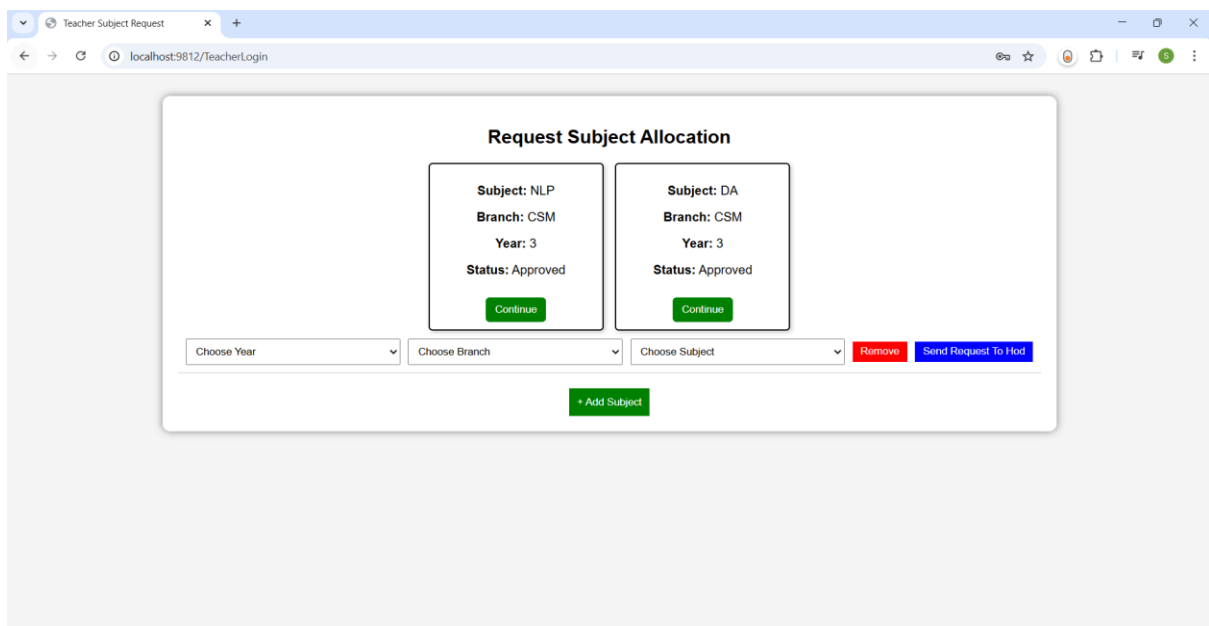


Fig 39: Subject Allocation Request Page for Faculty

If the HOD approves your request then you are enabled with the continue button and click on it to navigate to dashboard where you can perform the tasks like Enter Marks, View Marks, View Overall Marks and Edit Marks.

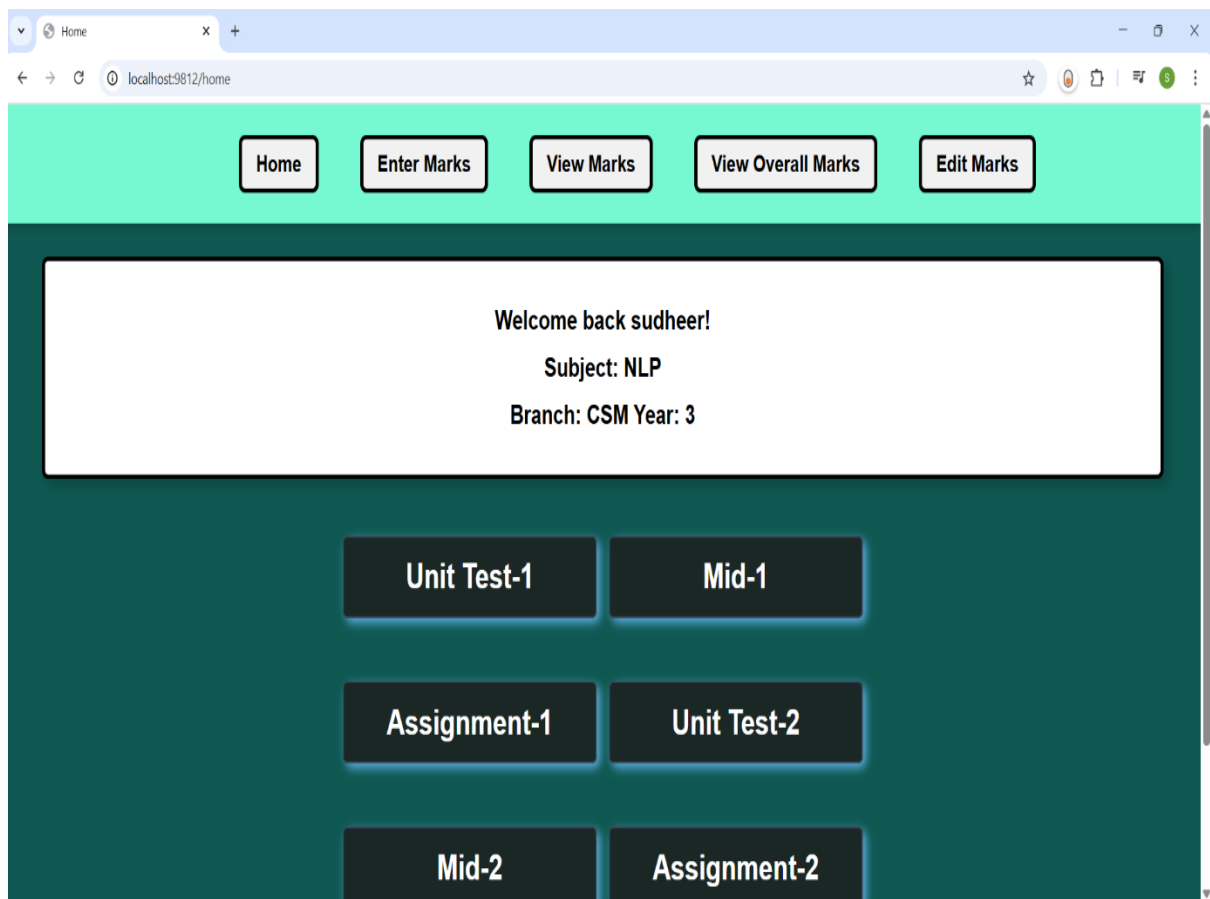


Fig 40: Home Page for Faculty

9.3.2.1 ENTER MARKS PAGE FOR FACULTY

enterMarks

localhost:9812/enterMarks/enterMarks.html

Home Enter Marks View Marks View Overall Marks Edit Marks

Exam: Select Exam Click to get student details

Select Exam

S.NO	HallTicket Num	Student Name	Marks
------	----------------	--------------	-------

Unit_test_1

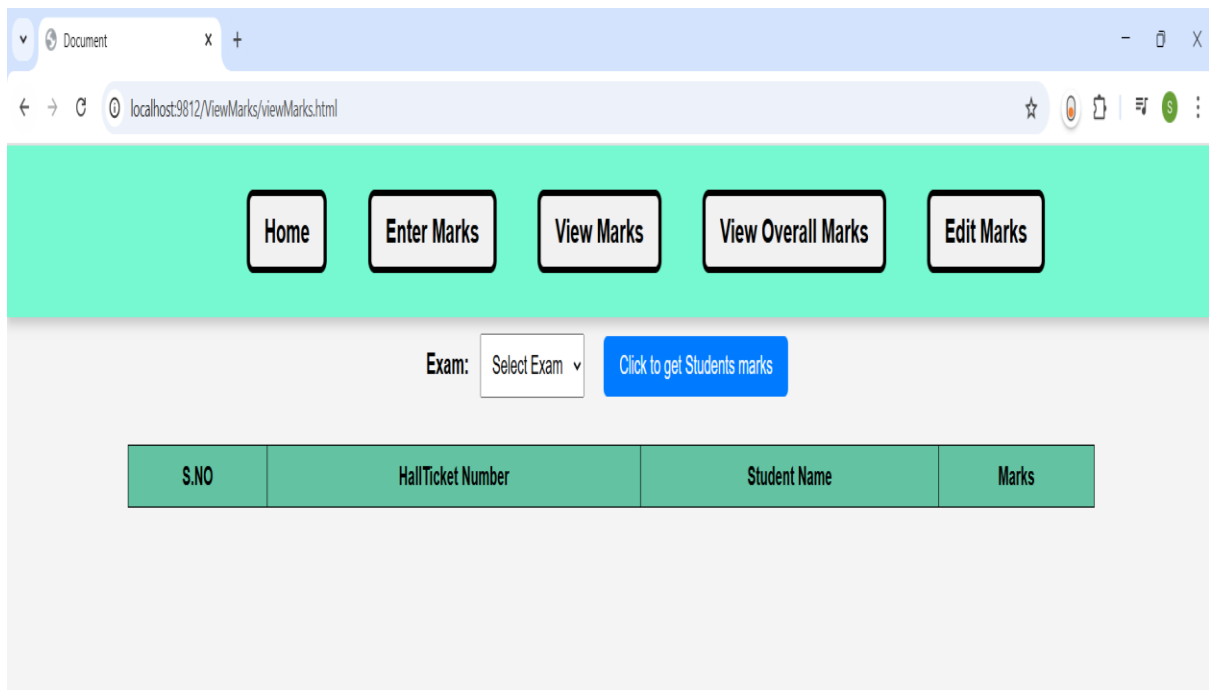
Mid_1

Save Marks

Fig 41: Enter Marks Page for Faculty

In this page, faculty can enter the marks for students. Faculty needs to select the exam and click on the 'Click to get student details' button to get students information, and after selecting the exam the heading (Marks) in the marks column will be replaced with the corresponding exam name. After entering the marks the faculty needs to click on 'save' button to save the marks.

9.3.2.2 VIEW MARKS PAGE FOR FACULTY



Document x +

localhost:9812/ViewMarks/viewMarks.html

Home Enter Marks View Marks View Overall Marks Edit Marks

Exam: Select Exam Click to get Students marks

S.NO	HallTicket Number	Student Name	Marks
------	-------------------	--------------	-------

Fig 42: View Marks Page for Faculty

If the faculty wants to view the marks of students in a exam or verify the marks, they can use this page to view marks by selecting the Exam and clicking on ‘Click to get Students marks’.

9.3.2.3 VIEW OVERALL MARKS PAGE FOR FACULTY

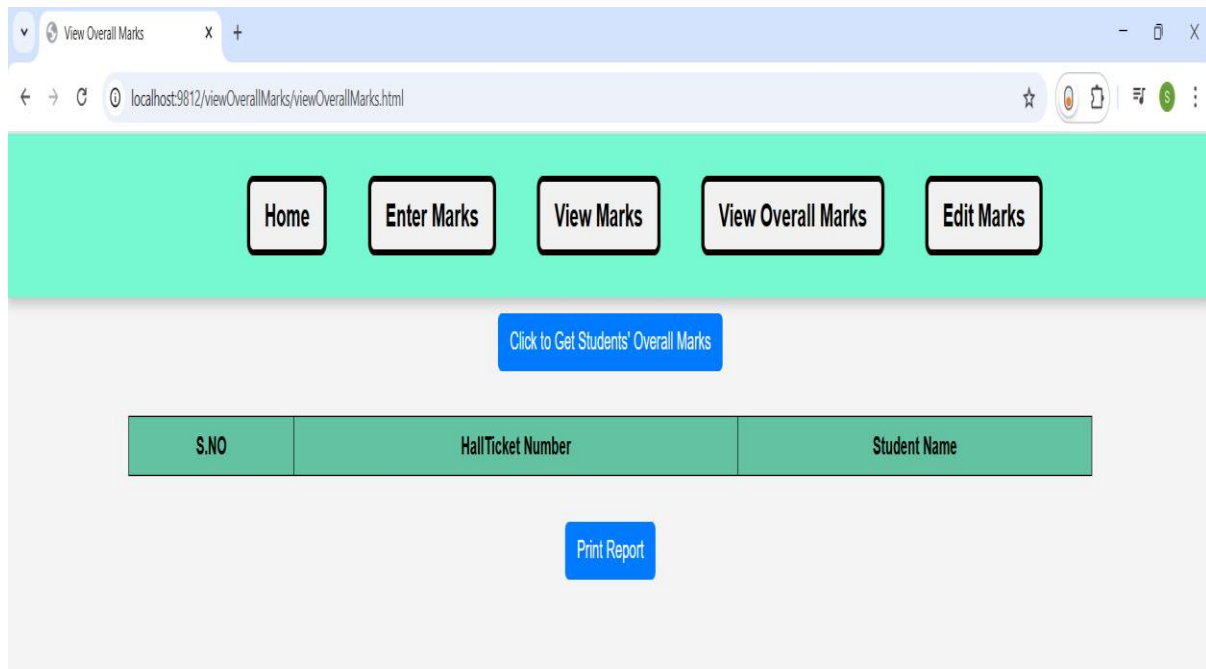


Fig 43 A: View Overall Marks Page for Faculty

In a semester there are different types of exams will be conducted for students, and after entering the marks for all exams the faculty needs to submit the final report or to view the all exams marks then this page comes into picture. The faculty needs to click on the ‘click to get students overall marks’ button to get all exams data, this appears as:

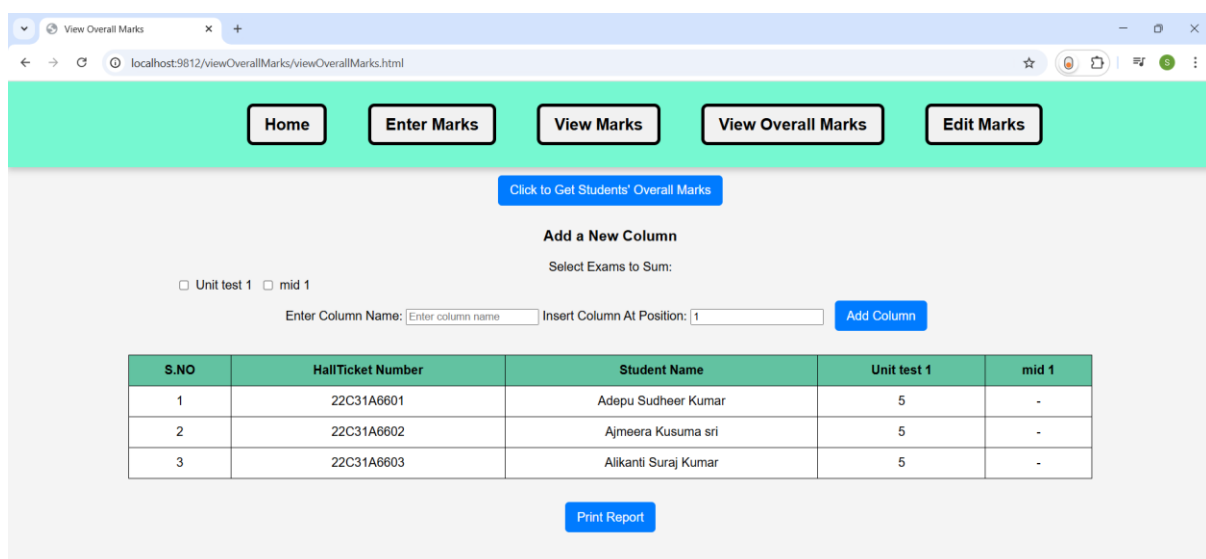
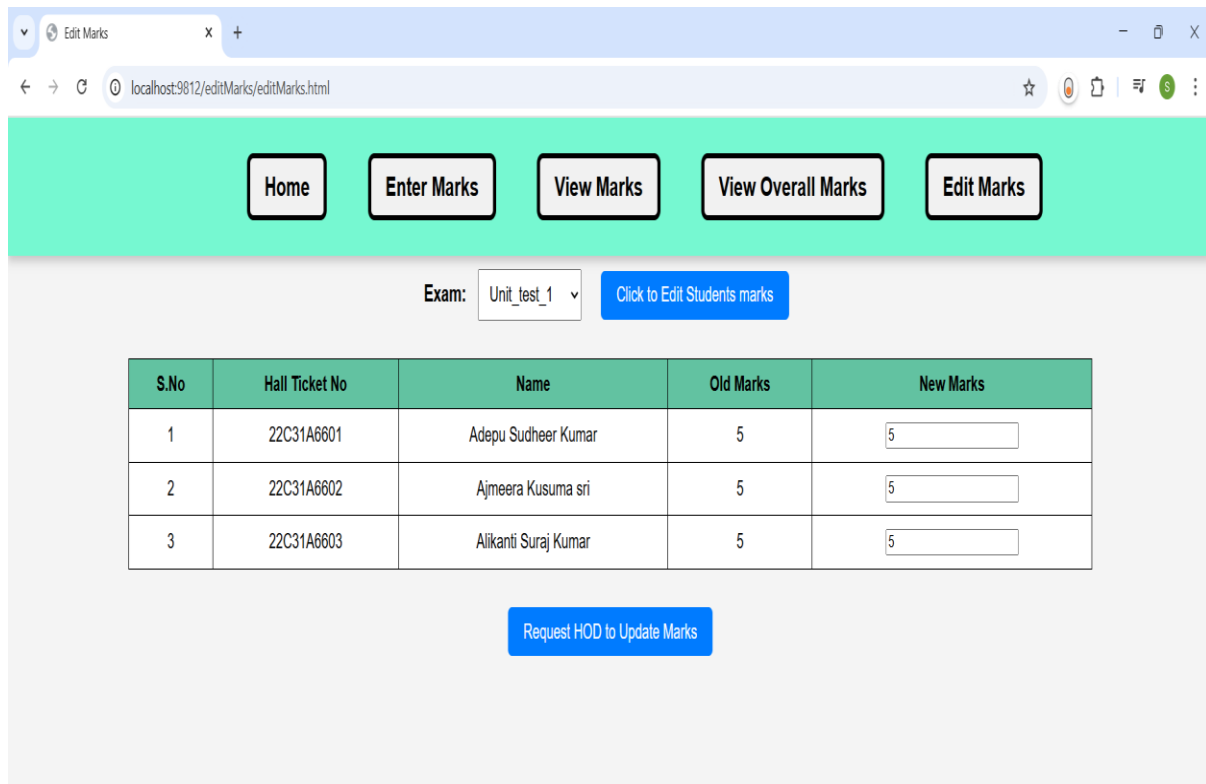


Fig 43 B: View Overall Marks Page for Faculty

After getting the exams data, if faculty wants add any new column to table to represent sum of exams then they need to select the exams in check box and name column and speicify it's position , finally click on 'Add column' button. There is a button named 'Print Report' which is used to print all these marks information.

9.3.2.4 EDIT MARKS PAGE FOR FACULTY



Exam: Unit_test_1 [Click to Edit Students marks](#)

S.No	Hall Ticket No	Name	Old Marks	New Marks
1	22C31A6601	Adepu Sudheer Kumar	5	<input type="text" value="5"/>
2	22C31A6602	Ajmeera Kusuma sri	5	<input type="text" value="5"/>
3	22C31A6603	Alikanti Suraj Kumar	5	<input type="text" value="5"/>

[Request HOD to Update Marks](#)

Fig 44: Edit Marks Page for Faculty

If the faculty wants to update/replace the student marks due to some mistakes in the already entered marks then they can make use of this page. The faculty needs to select the exam from the dropdown and click on 'Click to get student marks' button to get students' marks data. After getting the data faculty can enter the new marks and click on 'Request HOD to Update Marks' button to request HOD for marks update/replace. If HOD approves your request then marks will be automatically replaced/updated.

9.3.3 LOGIN PAGE FOR STUDENT

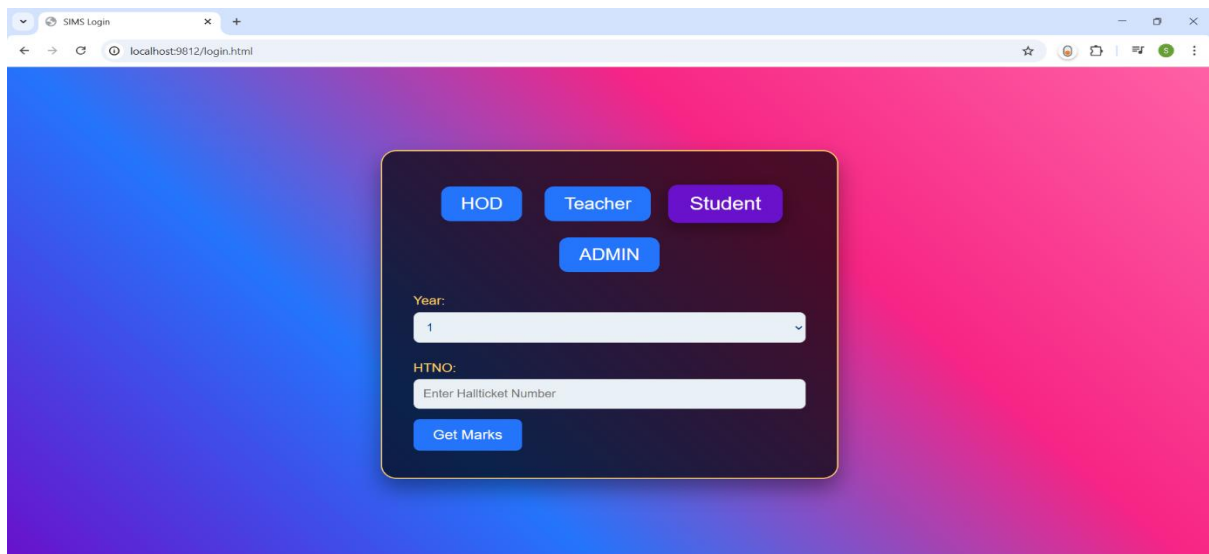
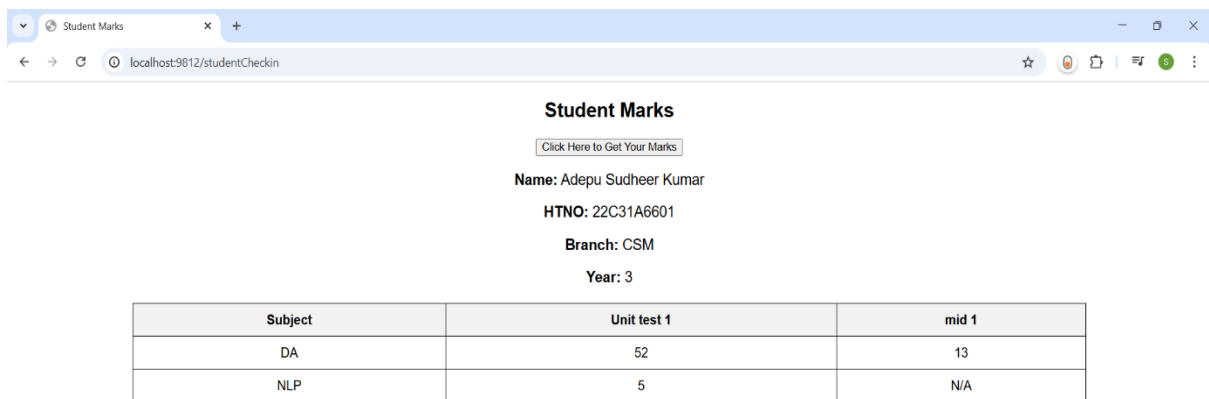


Fig 45: Login Page for Student

The student can login by selecting the year he/she belongs to and by entering the Hall ticket number, once you login you can able to view the marks of you by clicking on ‘click here to get marks’ button. This will be like as follows:



Subject	Unit test 1	mid 1
DA	52	13
NLP	5	N/A

Fig 46: View Marks Page for Student

9.3.4 LOGIN PAGE FOR ADMIN

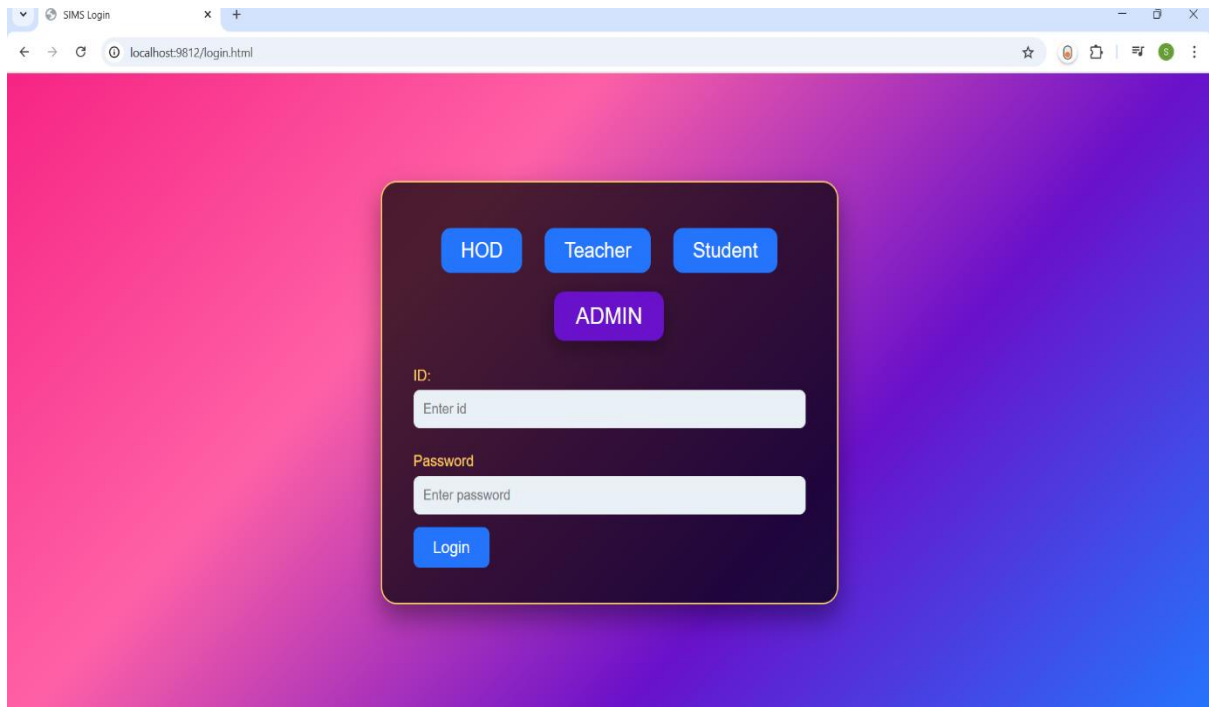


Fig 47: Login Page for Admin

The admin logs into the account by entering the credentials ID and Password. The admin can able to view the requests of HOD and Admin can approve or reject them. And Admin helps the Faculty, Hod's in resetting their passwords.



Fig 48: Home Page for Admin

10. CONCLUSION

The proposed Student Information Management System offers a structured and efficient approach to managing student data within educational institutions. By automating student record maintenance, academic tracking, and performance evaluation, this system enhances accuracy, accessibility, and security in student information management. With its role-based access control, SIMS effectively caters to various stakeholders. The Admin oversees user registrations and account security, ensuring a smooth workflow. The Head of Department (HOD) manages academic structures, assigns faculty to subjects, and oversees student records, reducing the complexities associated with manual student data handling. The Faculty module enables teachers to input and update student marks, ensuring real-time academic progress tracking. The Student module allows students to access their academic records effortlessly, promoting transparency and self-assessment. By implementing SIMS, institutions can eliminate the inefficiencies of traditional paper-based record management and manual data entry, reducing errors and redundancy. The centralized database ensures real-time updates and streamlined data retrieval, enabling quick decision-making and improved communication between faculty, students, and administrators. Additionally, report generation simplifies academic performance evaluation, saving time and effort for educators. Overall, Student Information Management System modernizes student information management by integrating automation, security, and accessibility, ensuring a seamless academic data management system. With its structured approach, it significantly enhances institutional efficiency, promotes transparency, and fosters a well-organized academic environment.

11. FUTURE SCOPE

The proposed project, “Student Information Management System,” offers several advantages in streamlining the management of student academic records. However, it also comes with certain limitations that highlight opportunities for future enhancement. One such limitation is the restricted access granted to students. While they can view their academic records, there is no built-in mechanism within the system for them to request corrections or raise concerns. Any discrepancies they encounter must be manually communicated to the faculty or administrative staff, which can slow down the resolution process. Faculty members, too, face some restrictions. Although they are allowed to enter student marks, they do not have the ability to modify the marks once they have been submitted. If any changes are necessary, they must seek approval from the Head of the Department (HOD), introducing an additional step that can delay updates to academic records. Similarly, the HOD has a defined but limited scope of actions. While they can manage subjects, assign sections, and approve faculty-related requests, the system lacks a direct communication channel that would allow HODs to provide feedback or request clarification from faculty members regarding submitted data. From the administrative perspective, the system administrator has access to critical functions such as approving HOD registrations and resetting user passwords. However, their role does not extend to direct manipulation or oversight of academic data. This constraint restricts the administrator to managing users and maintaining the overall functionality of the system, without the ability to intervene in academic-related issues. In conclusion, while SIMS significantly improves the efficiency of student record management, these current limitations emphasize the need for further system enhancements. Future upgrades could focus on improving communication between users, enabling easier correction of data, and allowing for more flexible user interactions.

12. BIBILOGRAPHY

1. W3Schools – HTML, CSS, JavaScript
2. Geeks for Geeks - Web Development
3. Node.js Tutorials - W3Schools
<https://www.w3schools.com/nodejs/>
4. Node.js Guide – Geeks for Geeks
<https://www.geeksforgeeks.org/nodejs/>
5. MySQL Tutorial - W3Schools+
<https://www.w3schools.com/sql/>
6. MySQL in Node.js – Geeks for Geeks
https://www.w3schools.com/nodejs/nodejs_mysql.asps
7. Express.js Tutorial - W3Schools
<https://www.w3schools.in/express-js/introduction>