**STUDENT MANAGEMENT SYSTEM**

**MINOR PROJECT REPORT**

***Submitted in partial fulfilment of the requirements for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

***in***

**COMPUTER SCIENCE & ENGINEERING**

**Submitted to**

**CLG/University Name**

**Submitted by**

Name

ENO

Name

ENO

Name

ENO

Name

ENO

Under the guidance of

Mr. Guide Name

Assistant Professor

**Department of Computer Science and Engineering**

CLG Name

**TABLE OF CONTENTS**

Title …………….…….….……………………………………………………………...……. 1

List Of Figures ….…….…..…………………………………………………………….……. 4

List Of Code Snippets ……...……………………………………………………..…...……... 5

Certificate ………….…………………………………………………………..…...………… 6

Declaration ………...…………………………………………………………..…...………… 7

Acknowledgement.….………………………………………………………….…………….. 8

Abstract ………...…...……………………………………………………………...…… 9 References ……...…..……………………………………………………………………….. 47

**Chapter 1 INTRODUCTION**

1.1 Overview ………………………………………………………………. 10

1.2 Motivation ……………………………………………………………... 10

1.3 Mongo DB and React Usage …………………………………...………. 11

1.4 Chapter Outline ………………………………………………………... 11

**Chapter 2 LITERATURE REVIEW**

2.1 Introduction …………………………………………………...….……. 13

2.2 Problem Statement …………………………………………………….. 14

2.3 Key Features

2.3.1 Navigating the Modules………………………………..…. 14

2.3.2 Technical Foundations ………………………………….…. 15

2.3.3 Versatility and Applications ………………………...……. 15

2.3.4 Scope for Development …………………………….…….. 15

2.3.5 Results …………………………………………………...… 16

2.4 Online Intranet College Management System for

Enhanced Educational Administration

2.4.1 Introduction …………………….……………………... 16

2.4.2 Overview of Document ……………………….………. 16

2.4.3 Methodology of Document …………………….……… 17

2.4.4 Purpose of Document …………………………………. 17

2.4.5 Scope for Development…………………………………17

2.5 General Description ………………………………………………….. 17

2.6 Conclusion ……………………………………………………………………………………… 18

**Chapter 3 PROPOSED SOLUTION: DESIGN**

3.1 Introduction ……………………………………....……..………. 19

3.2 Core Features …………………………………..…………...…... 20

3.3 Control Flow …………………………….……………………… 21

3.4 Tech Stack ………………………………….…....……………… 22

3.4.1 MERN …………………..………….……….………… 22

3.4.2 Mongo DB ………………………….…..……………... 25

**Chapter 4 IMPLEMENTATION**

4.1 Environment Setup ………………………………….…….…….. 33

4.2 Writing Views ……………………………………..………….…. 34

4.3 Mapping URLs …………………………………….………….… 37

4.4 Setting up database and admin panel …………………………….39

**Chapter 5 Working and Results**

5.1 Home page View ……………………………………………. 42

5.2 Desktop View ……………………………...……………….. 42

5.3 Admin login View ………………………….………………. 42

5.4 Teacher login View ………………………...……………….. 43

5.5 Student login View …………………………………………. 43

5.6 Teacher Dashboard ………………………..……………….... 44

5.7 Attendance view ………………………….…………………. 44

5.8 Admin Dashboard ………………………………………….. 44

5.9 Class Creation ………………………………….…………... 45

5.10 Subject Creation ………………………..…………………… 45

5.11 Teacher Addition …………………………………………… 45

5.12 Student Addition …………………………….……………… 45

5.13 Student Dashboard View……………………………………..46

**LIST OF FIGURES**

***Figure No. Figure Name Page No.***

1. Use Case Diagram ………………………………….……… 21
2. DFD level 1 Diagram ……………………………….…….... 22
3. Mongo DB sequence diagram …………….……………...… 28
4. React flow Diagram ……………………………..………..… 30
5. Combination of React Js and Mongo Db ………….……..…. 42

5.1 Home page View ……………………………..………..……. 42

5.2 Desktop View …………………………….………..……….. 42

5.3 Admin login View ………………………...………..………. 42

5.4 Teacher login View …………….………………..………….. 43

5.5 Student login View ………………………..…………..……. 43

5.6 Teacher Dashboard ……………………………..………….... 44

5.7 Attendance view ………………………………….....………. 44

5.8 Admin Dashboard ……………………………..…..……….. 44

5.9 Class Creation ……………………………….……………... 45

5.10 Subject Creation ………………………..…………………… 45

5.11 Teacher Addition …………………..……….……………… 45

5.12 Student Addition ……………………………….…………… 45

5.13 Student Dashboard View …………………………………..…46

**LIST OF CODE SNIPPET**

***Figure No. Figure Name Page No.***

4.1 Views for Homepage ………………….………..…………… 34

4.2 Mapping the URLs …………………….…………..………... 37

4.3 Routes 1 …………………………………………...………… 38

4.4 Routes 2 ………………………………………….…..……… 38

4.5 Setting up database and admin panel……………….……….. 39

**CERTIFICATE**

This is to certify that the project entitled "Student Management System" which is submitted by **Members Name** in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in Computer Science & Engineering of **College Name** affiliated to **University Name, Delhi** is a record of the candidate own work carried out by him under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

**Date:**

**Mrs. Guide Name**

**Assistant Professor**

**DECLARATION**

This is to certify that Synopsis Report Entitled “Student Management System” which is submitted in partial fulfilment of the requirement for the award of degree B.Tech in Computer Science and Engineering to **College Name** affiliated to **University Name** comprises only original work and studies carried out by students themselves. The matter embodied in this synopsis has not been submitted for the award of any other degree.

Signature

**Name**

Signature

**Name**

Signature

**Name**

Signature

**Name**

**Approved By:**

Signature

**Ms. Manu Sharma**

**Project Guide**

CSE Department

JIMSEMTC, Greater Noida

Signature

**Dr. Latha Banda**

**HOD**

CSE Department

JIMSEMTC, Greater N

**ACKNOWLEDGEMENT**

We are immensely grateful to our guide Mr. Rahul Deva at JIMS Engineering Management Technical Campus for the guidance, support, and valuable insights provided by him. As our supervisor, he has constantly guided us to remain steady on accomplishing our goal. He has guided us at each step, discussed our problems and gave his valuable feedback, despite his busy schedule. We are equally grateful to all our professors of JIMS Engineering Management Technical Campus.

This research project is possible because of the support of our parents, siblings and friends who were always there in our thick and thin and have always inspired us.

Signature

**Kaustav Majumder**

Signature

**Jaskirat Singh**

Signature

**Chirag Gupta**

Signature

**Nitin Singh**

**ABSTRACT**

In the realm of education, workforce management, and various other fields, the accurate tracking of attendance is crucial for ensuring productivity, compliance, and accountability. This abstract introduces an innovative Attendance System, designed to streamline and modernize attendance monitoring processes.

It eliminates the need for manual attendance records and minimizes the scope for errors, enhancing administrative efficiency and reducing the administrative burden.

Key features of this Attendance System include real-time data collection, centralized data storage, and intuitive reporting tools. It provides stakeholders, including educators, managers, and administrators, with instant access to attendance records and statistical insights. This facilitates informed decision-making, timely interventions, and improved resource allocation.

Furthermore, the system promotes transparency and accountability by providing audit trails and a robust authentication process. It can be customized to suit various organizational structures and sizes, making it a versatile solution for educational institutions, businesses, and government agencies.

The benefits of adopting an Attendance System are numerous, including enhanced time management, cost savings, and the ability to meet regulatory requirements effortlessly. This abstract highlights the significance of such a system in the modern age and its potential to revolutionize attendance tracking, leading to a more efficient, accountable, and data-driven environment.

1. **Introduction**

**1.1 Overview**

In the dynamic and fast-paced world of education, businesses, and organizations, the accurate tracking of attendance has never been more crucial. Traditional manual methods for recording attendance are not only time-consuming but also prone to errors, making it essential to modernize this essential process. This introduction presents a robust and efficient Attendance System. This web based application facilitates the efficient flow of information within the organization, benefiting both technical and non-technical staff and students.

The user interface, developed with React, ensures a user-friendly and responsive experience. Redact’s JavaScript framework offers an interactive and dynamic front-end design, enabling users to navigate the system effortlessly. The back-end, powered by Node.js and Express.js, serves as the heart of the system, ensuring smooth and efficient server-side operations.

Data storage and retrieval are facilitated by MongoDB, a NoSQL database that excels in flexibility and scalability. Mongoose, acting as a mediator between Node.js and MongoDB, streamlines database communication, making it a seamless process. This approach ensures that information is stored securely and accessed with optimal speed.

In conclusion, "Attendance Management System" offers a modern and integrated solution to address the multifaceted challenges of information management in educational institutions. It brings together the strengths of React for a dynamic front end, Node.js and Express.js for a robust server, and MongoDB with Mongoose for efficient data storage and retrieval. This system embodies the power of MERN technology to transform the way educational institutions manage and disseminate information, ensuring efficiency and accessibility for all stakeholders.

**1.2 Motivaion**

The development of our Attendance Management System (AMS) is driven by a commitment to instil responsibility and collaboration among students in our institution. Navigating the transition from high school to college, especially in managing attendance, can be challenging. Our AMS provides a platform for students to interact with peers who have successfully managed attendance, fostering support and guidance.

Beyond basic attendance tracking, our AMS offers a comprehensive database for informed decisions on courses, schedules, and extracurriculars, promoting efficient planning for better academic performance. The one-on-one video feature connects students for direct guidance on attendance management, aiding those exploring careers or navigating the college experience.

In essence, our Attendance Management System is more than a tracking tool; it's a holistic platform fostering responsibility and peer mentorship for academic success.

**1.3 Mongo DB and React Usage**

In developing our Attendance Management System, we've strategically employed React and MongoDB to create a dynamic and efficient solution. React, a JavaScript library for building user interfaces, forms the frontend of our system. Its component-based architecture facilitates the creation of reusable UI components, ensuring a responsive and interactive user experience. On the backend, we've leveraged MongoDB, a NoSQL database known for its flexibility and scalability. MongoDB efficiently handles data storage and retrieval, providing a secure and optimal environment for managing attendance records. This combination of React and MongoDB in our Attendance Management System results in a powerful and streamlined application, where React manages the frontend for user interaction, and MongoDB handles the backend for seamless data management.

**1.4 Chapter Outline**

This project report comprises the following segments and references organized as follows:

**Chapter 1** - This segment discusses the problem statement, various existing technologies used to build the Attendance Management System web application. These existing technologies are compared based on various parameters. This chapter also covers extensive research on the existing technologies.

**Chapter 2** - This segment discusses the proposed solution and the System Architecture and the control flow of the project build. This segment focuses on describing the proposed solution for the project, including the system architecture and the control flow of the application build. It provides an overview of how the different components of the system interact and work together to achieve the desired functionalities.

**Chapter 3** - This segment of the project involves discussing the application details, specifically the setup of the development environment, writing views, mapping URLs, creating templates, setting up the database, and configuring the admin panel. These aspects are fundamental to the functionality and structure of the application.

**Chapter 4** - This segment discusses the working and UI being used in our project Attendance Management System with the user interactions and usage. This segment features all the possible features we have implemented in the project.

1. **Literature Survey**

Until recently the place of the College Management System (CMS). Nowadays, education is playing a very significant role in society. Day by day, the percentage of illiterates is decreasing and the percentage of literates is increasing. Education will change society in all aspects, and everyone wants to study for higher professional degrees. Admissions are increasing day by day so thereby.

The ratio of the establishment of new colleges and schools is also increasing. But the actual challenge is starting now. Most schools and colleges maintain student information in records. When the number of records increased, it was difficult to maintain the information of each student in the old manual system. Maintaining the records manually leads to error-prone and requires more manpower and it consumes more time for processing the records

Feasibility Study Economical Feasibility: The system being developed is economic with respect to the School or college’s point of view. It is cost-effective in the sense that has eliminated the paperwork.

The system is also time effective because the calculations are automated and are made at the end of the month or as per the user's requirement. The result obtained contains minimum errors and is highly accurate as the data is required. Technical feasibility: The technical requirement for the system is economic and it does not use any other additional Hardware and software. Behavioural Feasibility: The system working is quite easy to use and learn due to its simple but attractive interface. The user requires no special training for operating the system.

In the ever-evolving landscape of educational administration, Lalit Mohan Joshi introduces a groundbreaking project poised to transform the way colleges manage their myriad processes. The College Management System (CMS), crafted using PHP, emerges as a beacon of efficiency, promising to streamline tasks ranging from student information and placements to event coordination and notices. Joshi's vision extends beyond a mere technological upgrade; it encapsulates a shift toward user-centric and accessible college administration.

**2.1 Introduction: The Genesis of Innovation**

At the core of Joshi's research is the College Management System, a web-based application meticulously designed to untangle the complexities of college management. The scope of the CMS is vast, covering student attendance, event coordination, fee transactions, marks, photo galleries, and more. Positioned as a revolutionary tool, the CMS aims to liberate educational institutions from the shackles of paperwork and manual tasks, offering an efficient and centralized platform for college stakeholders.

**2.2 Problem Statement**

Attendance tracking, a fundamental aspect of various institutions and organizations, often presents significant challenges when executed manually or without a dedicated system. Several issues and inefficiencies arise due to these outdated methods, including:

Inaccuracies and Errors: Manual attendance taking is prone to human error, leading to discrepancies in recorded data, which can have serious consequences for payroll, compliance, and overall accountability.

Time-Consuming Processes: Manually recording attendance consumes valuable time and resources, both for administrators and attendees, leading to reduced productivity and operational inefficiency.

Lack of Real-time Data: Traditional methods provide limited visibility into attendance data in realtime, making it challenging to respond promptly to issues or make data-driven decisions.

Data Security and Accessibility: Paper-based or local storage systems lack the security and accessibility needed in a digital age, potentially leading to data loss, theft, or unauthorized access.

**2.3 Key Features**

**2.3.1 Navigating the Modules**

Delving into the CMS's architecture, Joshi unveils a tapestry of modules, each dedicated to a specific facet of college life. From Student Management to Placement Details, Notices, Registration, Hostel Management, Room Allotment, Mess Management, and Room Fees, the CMS covers the entire spectrum of administrative needs. The Placement module, for instance, emerges as a crucial tracking tool for student placements, while the Notices module serves as a dynamic information board, keeping students abreast of upcoming events.

**2.3.2 Technical Foundations: Building Blocks for Success**

To implement the CMS effectively, Joshi lays out the technical prerequisites, underscoring the pivotal roles of Java, HTML, CSS, and JavaScript. Java takes center stage, providing the necessary foundation for the system's functionality. Meanwhile, HTML, CSS, and JavaScript contribute to creating an interface that is not only dynamic but also user-friendly. This amalgamation of technologies reflects a modern and user-centric approach to web development, ensuring the accessibility and usability of the CMS.

**2.3.3 Versatility and Applications: Unleashing the Potential**

Beyond its core functionalities, Joshi highlights the versatility of CMS. It is not confined to being a mere administrative tool; instead, it has the potential to serve as a platform for creating Graphical User Interfaces (GUIs) for desktop applications. While acknowledging the challenges posed by this extension, Joshi's vision opens up exciting possibilities for developers to explore PHP in different contexts, fostering adaptability and growth.

**2.3.4 Scope for Development: Human-Centric Innovation**

The CMS's scope is intricately woven around user requirements, placing a premium on access to information, secure login processes, password management, and feedback mechanisms. Envisioning an administrator with the power to oversee and manage user uploads, Joshi crafts a narrative that emphasizes not just technological robustness but also the human-centric aspect of administrative control. In essence, the CMS becomes a tool that understands and caters to the diverse needs of its users.

**2.3.5 Results: A Glimpse into the Future**

Joshi offers readers a visual appetizer, showcasing the login and registration forms that serve as the entry points to the CMS. The login form stands as a secure gateway for registered users, ensuring authenticated access. Simultaneously, the registration form is a critical element in the onboarding process, signalling the importance of user engagement in the success of the CMS.

**2.4 Online Intranet College Management System for Enhanced Educational Administration**

In the academic realm, Lalit Mohan Joshi, an MTech scholar at BTKIT Waratah, takes the baton of innovation, presenting a comprehensive research paper on the development of an Online Intranet College Management System (CMS). This technological marvel caters to the nuanced needs of educational institutions, promising to streamline administrative processes and provide a centralized platform accessible to both students and staff. Joshi's work, rooted in pragmatism, envisions a system that facilitates tasks such as attendance monitoring, information retrieval, and data updates, creating an ecosystem of seamless educational administration.

**2.4.1 Introduction: Navigating the Educational Landscape**

Joshi's paper introduces the significance of an Intranet-based CMS within the context of educational institutions. The focal point is creating a system that transcends silos, making information accessible institution-wide. The emphasis on the development for an engineering college underscores the project's specificity, highlighting the importance of user registration in controlling data access and ensuring a secure and tailored experience for each user.

**2.4.2 Overview of Document: Bridging Internal and External Audiences**

The document seamlessly transitions between non-technical overviews and technical specifics, catering to external and internal audiences. Covering aspects like data requirements, functional specifications, and the general product description, the document serves as a holistic guide. The technical section delves into external interfaces, performance requirements, and technical platforms, ensuring a comprehensive understanding for those involved in the project.

**2.4.3 Methodology of Document: Weaving a Tapestry of Innovation**

The CMS is conceptualized as a user-friendly system, blending HTML pages and JavaScript for client-side validation, Java for business logic, and an Oracle database for robust data management. The collaborative platform of Google Colab serves as the development environment, reflecting Joshi's commitment to leveraging contemporary tools. The project's structural pillars—data collection, pre-processing, clustering, model building, and evaluation—echo the iterative nature of machine learning projects, ensuring a robust and evolving system.

**2.4.4 Purpose of Document: Bridging Technical and Non-Technical Realms**

The Software Requirement Specification (SRS) acts as the compass, guiding the development of the CMS project. It not only outlines functionalities and interfaces but emphasizes the need for a simple, attractive, and self-explanatory interface. This nuanced approach caters to both technical and non-technical users, aligning with Joshi's vision of inclusivity within the educational technology landscape.

**2.4.5 Scope for Development: User-Centric Innovation**

The scope of the CMS project meticulously delineates user requirements, ranging from information access and secure login to password management, seeking help, and providing feedback. The inclusion of an admin login further amplifies the system's versatility, envisioning an overseer capable of managing user uploads. Joshi's vision extends beyond the technical intricacies, focusing on creating an environment where users feel heard, supported, and in control.

**2.5 General Description: User-Centric Perspectives**

Joshi introduces three distinct user categories—administrators, students, and staff—each bestowed with varying access rights. The product is positioned as a standalone application operating within an intranet network. The hardware and software prerequisites provide a clear roadmap for implementation, emphasizing the need for standard components and Java-enabled browsers.

**2.6 Conclusion: A Human Touch in Educational Technology**

In concluding his research, Lalit Mohan Joshi doesn't merely present a technological advancement; he presents a paradigm shift in college management. The College Management System, with its PHP-based architecture, emerges not just as a solution to administrative challenges but as a human-centric approach to enhancing the overall college experience. By addressing the intricacies of manual processes and communication costs, the CMS becomes a transformative tool in the realm of educational administration. Joshi's work is not confined to the realm of codes and algorithms; it's a testament to the fusion of technology and empathy, creating an administrative ecosystem that resonates with administrators, faculty, and students alike.

1. **Proposed Solution – Design**

This segment discusses the structure of the solution that we have proposed to build a web application Attendance Management System

**3.1 Introduction**

The primary objective of the project is to develop a comprehensive web portal that enables seamless communication between freshmen and seniors/alumni of our college. The platform aims to provide detailed information, guidance, and resources to assist freshmen in navigating their academic journey across different fields of study. Additionally, the web application incorporates a feature for one-on-one video interactions, allowing personalized guidance and mentorship opportunities. Within the platform, alumni play a crucial role by initiating channels where they can share essential information, insights, and experiences with new students. This information exchange fosters a supportive environment where members can contribute relevant materials and engage in meaningful discussions under the permission of the channel owner. The web portal serves as a centralized hub for information, resources, and mentorship, catering to the diverse needs of students at various stages of their education. By providing access to comprehensive guidance and facilitating interactions with experienced individuals, the platform aims to create a collaborative and supportive community that enhances the academic and personal growth of its users.

**3.2 Core Features**

A college attendance management system is a software solution designed to streamline and automate the process of tracking and managing student attendance. The core features of such a system typically include:

**1. User Authentication**

- User authentication ensures that only authorized individuals, such as faculty members or administrators, have access to attendance data. It often involves secure login credentials to protect sensitive information.

**2. Dashboard**

- A user-friendly dashboard provides an overview of attendance-related information, including class-wise attendance summaries, overall attendance percentages, and alerts for low attendance.

**3. Student Registration**

- The system typically integrates with student databases, allowing for easy registration and enrolment of students in specific courses. This data is crucial for tracking individual attendance records.

**4. Attendance Tracking**

- Faculty members can mark attendance for each class, specifying whether a student is present, absent, or tardy. Some systems may support biometric or RFID-based attendance tracking for added accuracy.

**5. Reporting and Analytics**

- Robust reporting features enable administrators and faculty to generate attendance reports. These reports may include class-wise attendance, individual student records, and trends over time. Analytics tools can help identify patterns and insights from the data.

**6. Mobile Accessibility**

- Many attendance management systems offer mobile applications, allowing faculty members to mark attendance on the go and enabling students to view their attendance records from their smartphones.

**7. Security and Data Privacy**

- Given the sensitivity of attendance data, these systems prioritize security and data privacy. Encryption, access controls, and regular data backups are essential components.

**8. Customization and Flexibility**

- The system should be customizable to meet the specific needs and policies of the college. This includes options to configure attendance rules, define attendance thresholds, and adapt to different grading systems.

A well-implemented college attendance management system can enhance efficiency, reduce paperwork, improve accuracy, and provide valuable insights into student engagement and performance.

**3.3 CONTROL FLOW**

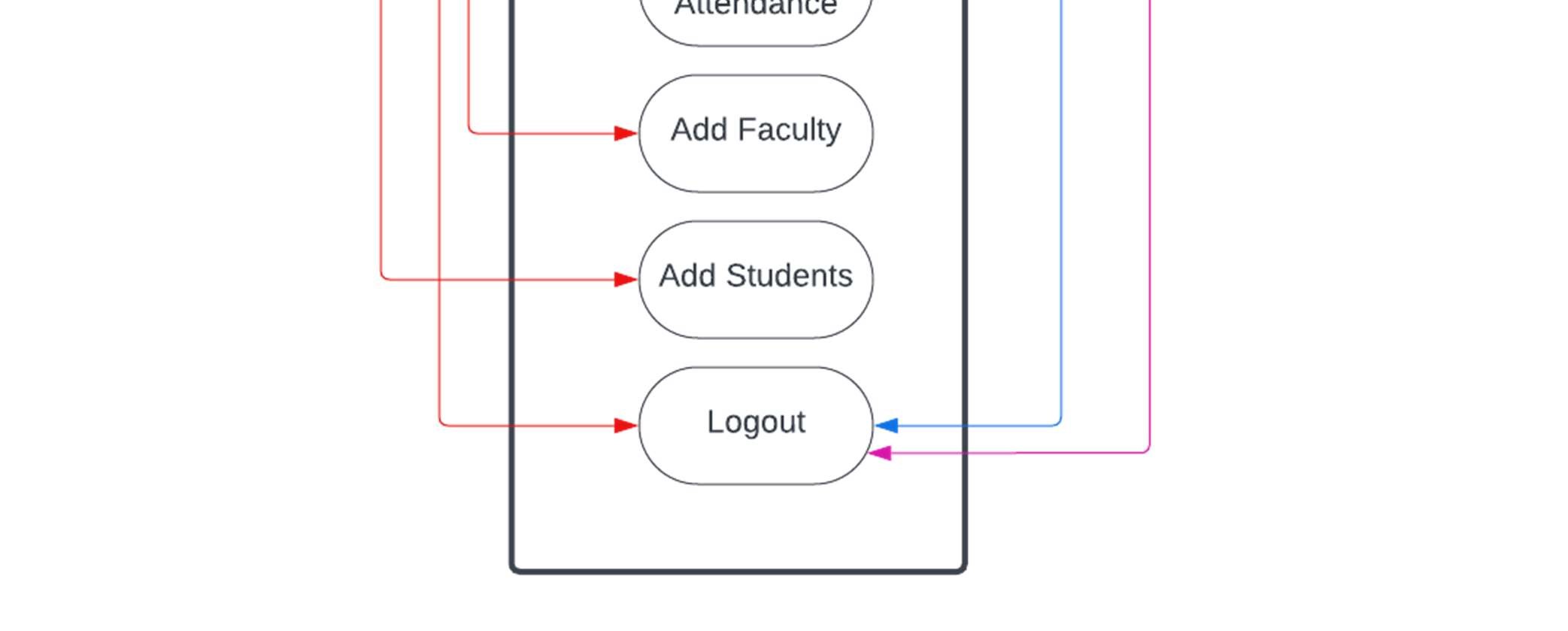
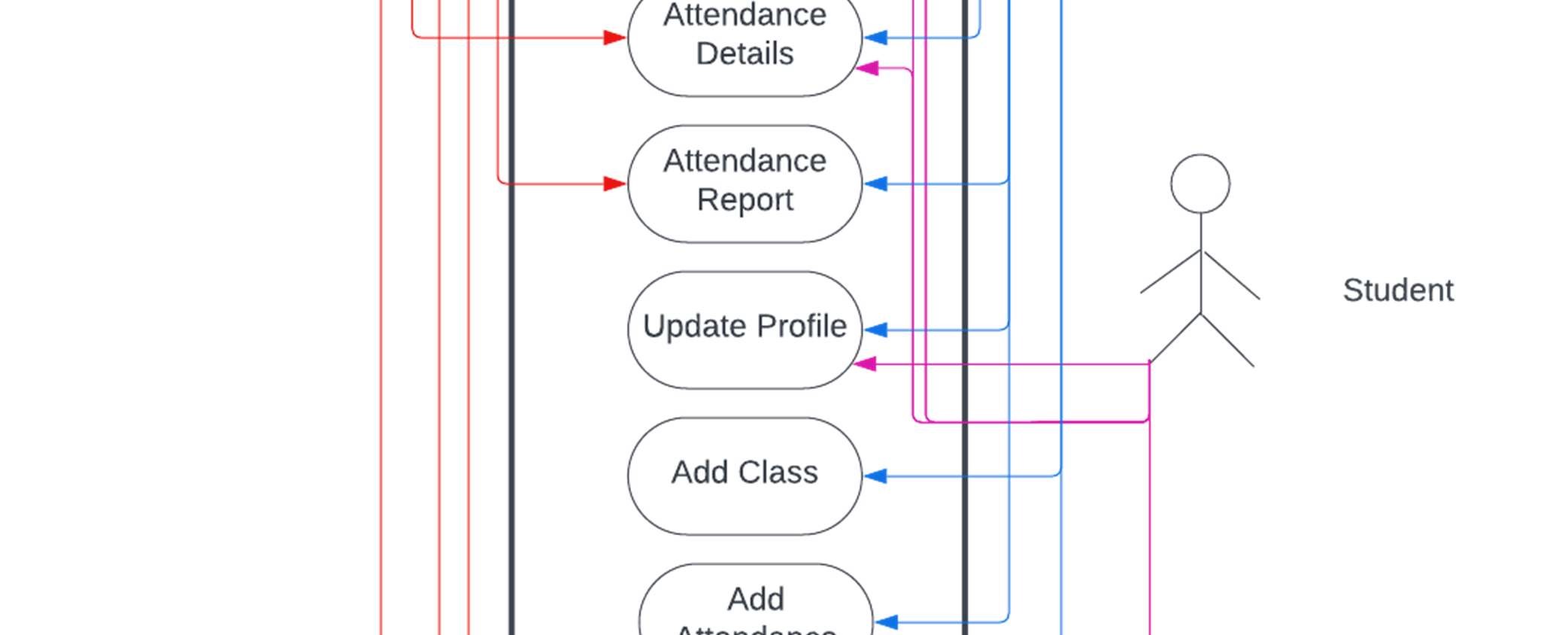
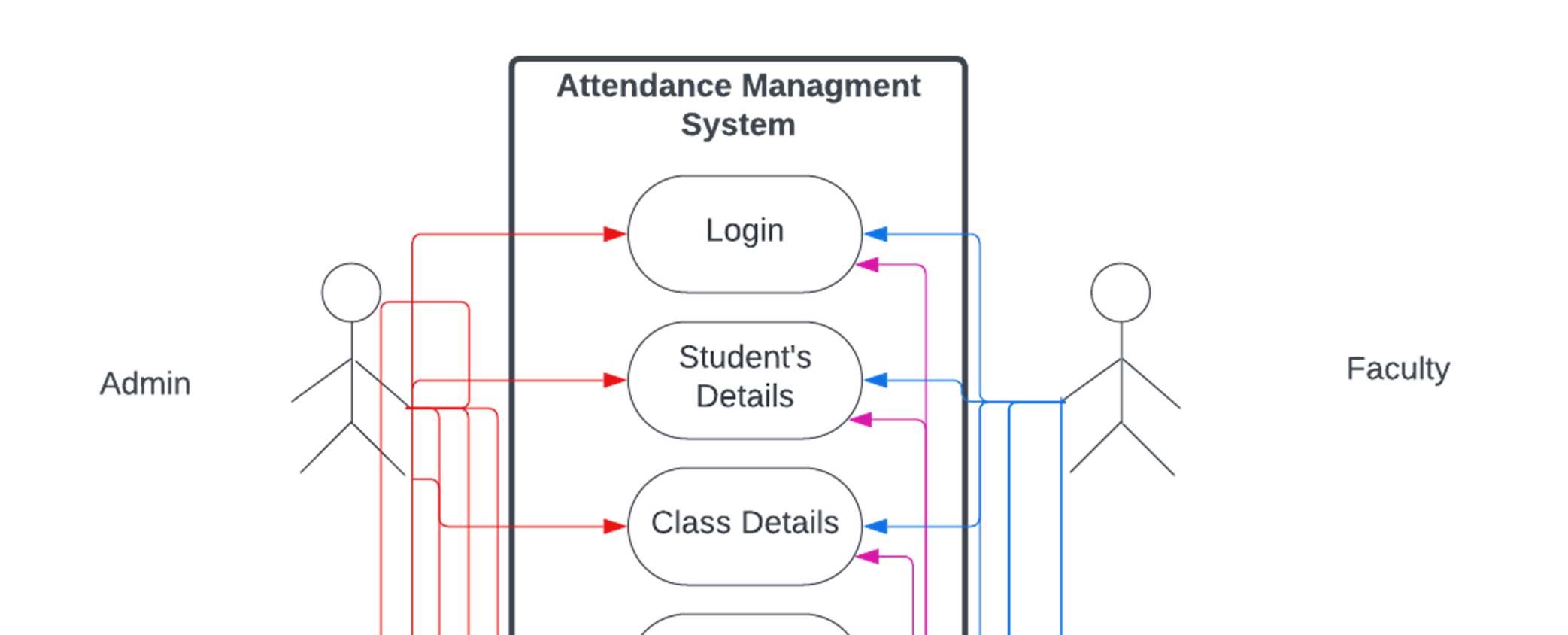


Fig 1. Use Case Diagram

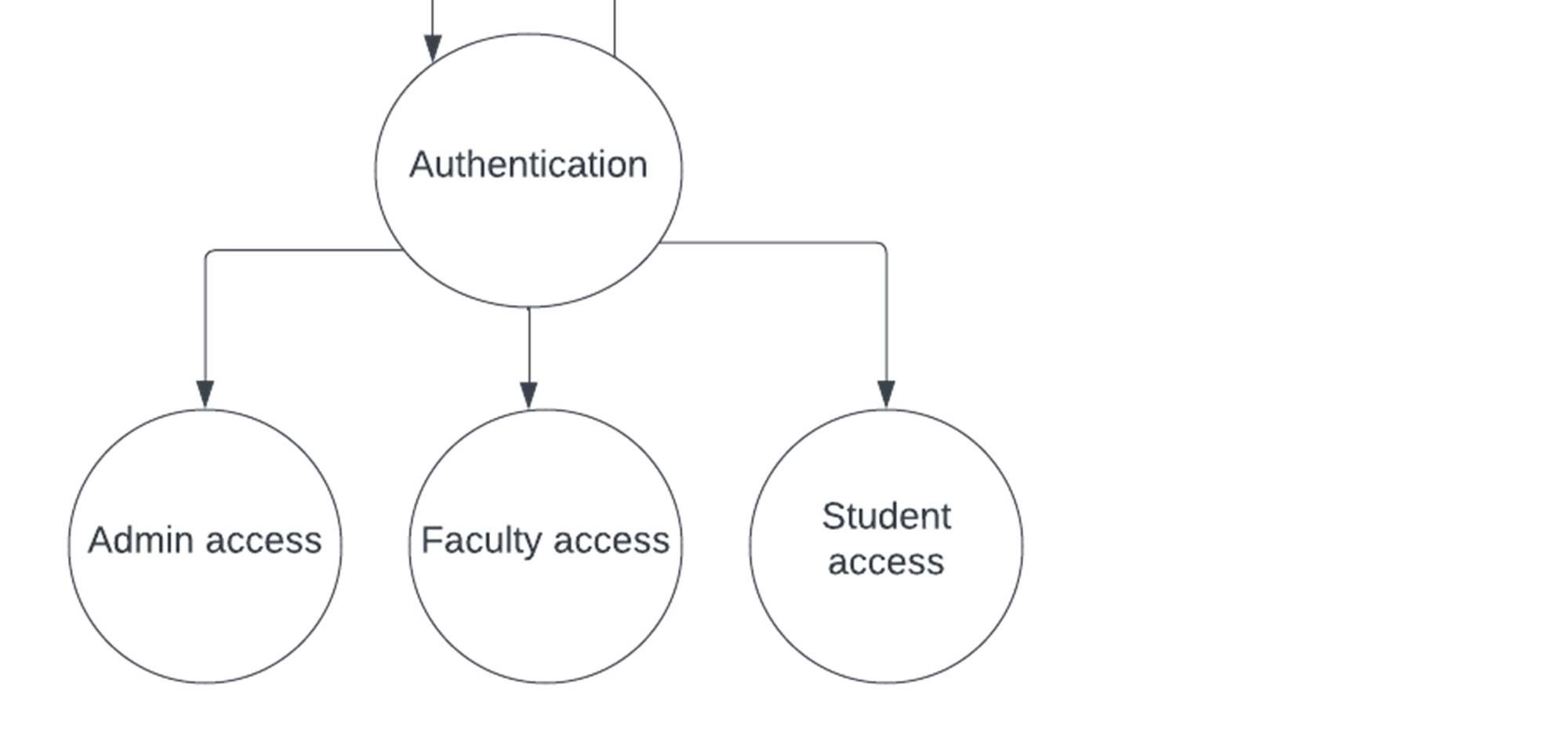
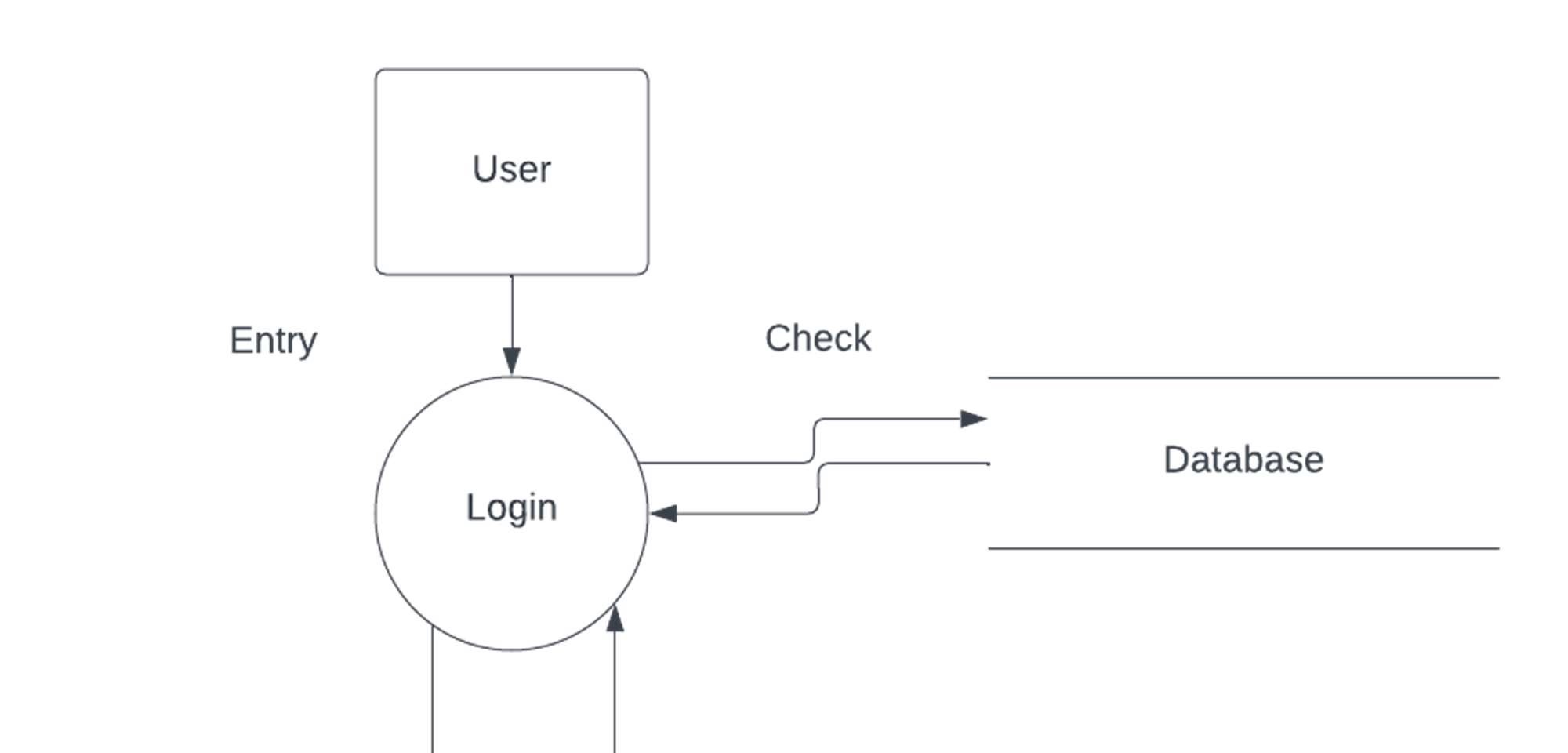


Fig 2. DFD Level - 1

**3.4 TECH STACK**

**3.4.1 MERN**

MERN is an acronym that represents the combination of MongoDB, Express, React, and Node, which are key technologies used in the MERN stack.

MongoDB is a document-oriented database that provides a flexible and scalable solution for storing and retrieving data. It is well-suited for handling large volumes of data and allows for easy integration with the other components of the MERN stack.

Express.js is a web framework built on top of Node.js, which provides a simple and efficient way to create server-side applications. Express.js offers various features and middleware that streamline the development process and make it easier to handle routing, request handling, and other server-side operations.

React.js is a powerful JavaScript library for building user interfaces. It focuses on creating reusable UI components and provides a declarative syntax for efficiently updating and rendering UI elements. React.js enables the development of dynamic and interactive user interfaces, making it a popular choice for front-end development.

Node.js is a JavaScript runtime environment that allows developers to run JavaScript code on the server-side. It provides a non-blocking, event-driven architecture, which makes it highly efficient and scalable for building server applications. Node.js is widely used for creating web servers and APIs, enabling seamless communication between the front-end and back-end components of an application.

The combination of MongoDB, Express, React, and Node (MERN) provides a comprehensive JavaScript-based stack for developing full-stack web applications. This stack leverages the strengths of each technology, allowing developers to work with JavaScript and JSON throughout the entire development process. Whether it's handling data storage, server-side logic, or creating interactive user interfaces, the MERN stack offers a cohesive and efficient approach to building JavaScriptbased applications.

**React.js**

At the top tier of the MERN stack is React.js, a powerful JavaScript framework specifically designed for building dynamic client-side applications in HTML. React simplifies the process of creating complex user interfaces by breaking them down into small, reusable components. These components can be easily connected to the data on the back-end server and rendered as HTML.

React excels in handling stateful, data-driven interfaces with minimal code and complexity. It offers an intuitive and efficient approach to managing application state, making it easier to update and synchronize data between components. React's declarative syntax allows developers to describe the desired UI state, and React takes care of efficiently updating the actual interface to reflect those changes

In addition to its state management capabilities, React provides a wide range of features expected from a modern web framework. It offers excellent support for forms, making it straightforward to handle user input and validation. React also provides robust error handling mechanisms, allowing developers to handle and display errors in a user-friendly manner. The framework facilitates event handling, enabling the creation of interactive interfaces that respond to user actions. React also provides efficient rendering of lists, allowing developers to iterate over data and generate dynamic UI elements

Overall, React.js is a comprehensive and feature-rich framework that empowers developers to build sophisticated and data-driven interfaces with minimal effort. Its focus on simplicity, reusability, and efficient state management makes it a popular choice for developing modern web applications.

**Declarative**

React is designed to make the process of creating interactive user interfaces (UIs) as seamless as possible. With React, developers can design simple views for each state of their application. When the underlying data changes, React efficiently updates and renders only the necessary components, minimizing unnecessary re-renders and optimizing performance.

The concept of declarative views is fundamental to React's design philosophy. Instead of imperatively manipulating the DOM (Document Object Model) to update the UI, React allows developers to describe the desired UI state using a declarative syntax. Developers specify how the UI should look based on the current data and React takes care of efficiently updating the DOM to reflect those changes.

By adopting a declarative approach, React makes code more predictable and easier to debug. Developers can focus on describing the desired outcome, and React handles the internals of updating the UI. This declarative nature of React ensures that the UI stays in sync with the data, reducing the chances of bugs caused by manual DOM manipulation or inconsistent state management.

With React's efficient rendering and declarative views, developers can build UIs that are responsive, scalable, and maintainable. React's ability to efficiently update and render components based on data changes provides a smooth and optimized user experience. The predictability and ease of debugging offered by declarative views allow developers.

**Component-Based**

In React, developers build encapsulated components that are responsible for managing their own state. These components are designed to be self-contained and modular, handling their specific functionality and data independently. This approach promotes reusability and separation of concerns.

By creating encapsulated components, developers can compose them together to construct complex user interfaces (UIs). Each component focuses on a specific task or feature, making it easier to understand, test, and maintain. The composability of React components allows for a modular architecture, where different components can be combined to form more intricate and sophisticated UIs.

One of the advantages of using JavaScript for component logic instead of templates is the ability to pass rich data throughout the application. JavaScript provides a flexible and powerful language for manipulating and managing data structures. With React, developers can easily pass data between components, allowing for seamless communication and data sharing across the application.

Additionally, by keeping the state out of the DOM, React ensures that the UI remains in sync with the underlying data. Instead of directly manipulating the DOM to reflect changes in state, React handles state updates internally. This separation of concerns improves the performance and maintainability of the application.

Overall, React's approach of building encapsulated components and managing state in JavaScript promotes code reusability, modularity, and flexibility. The ability to easily pass rich data throughout the application enhances the interactivity and functionality of the UI. By keeping the state out of the DOM, React ensures a more efficient and consistent rendering of the UI, leading to better performance and maintainability.

**3.4.2 Mongo DB**

MongoDB is a popular open-source NoSQL database management system that falls under the category of document-oriented databases. It was developed by MongoDB Inc. and is designed to handle large volumes of data with a flexible, schema-less data model. MongoDB is widely used for building modern, scalable, and high-performance applications.

Here are some key features and concepts associated with MongoDB:

1. Document-Oriented:

- MongoDB stores data in BSON (Binary JSON) documents, which are JSON-like, schema-less data structures.

- Documents in MongoDB can have nested structures, arrays, and other complex types, making it easy to represent and store complex data.

2. Collections and Documents:

- Data in MongoDB is organized into collections, which are analogous to tables in relational databases.

- Each collection contains documents, which are individual records or data entries. Documents are stored in BSON format.

3. Schema-less:

- MongoDB is schema-less, meaning each document in a collection can have a different structure.

- This flexibility allows for easy modification and adaptation of the data model without requiring changes to the entire database.

4. Indexes:

- MongoDB supports the creation of indexes on fields within a collection, which can significantly improve query performance.

- Indexes help to accelerate the retrieval of data by allowing the database engine to quickly locate the documents that match a query.

5. Query Language:

- MongoDB uses a rich query language, including support for various query operators and expressions.

- Queries can be performed on documents using a syntax that resembles the JSON structure.

6. Aggregation Framework:

- MongoDB provides a powerful aggregation framework for performing data transformations and computations on the server side.

- This framework allows for complex data processing operations, such as filtering, grouping, sorting, and projecting.

7. Replication:

- MongoDB supports automatic data replication, allowing you to create multiple copies of your data across different servers or clusters.

- Replication provides high availability and fault tolerance by ensuring that if one server fails, another can take over.

8. Sharding:

- MongoDB can horizontally scale by sharding data across multiple servers or clusters.

- Sharding allows for the distribution of data and queries across multiple machines, improving performance and handling large datasets.

9. Transactions:

- MongoDB supports multi-document transactions, allowing multiple operations to be grouped together and executed atomically.

- Transactions ensure data consistency and integrity in situations where multiple operations need to be performed as a single unit.

10. Security:

- MongoDB provides various security features, including authentication, authorization, and encryption.

- Users can be assigned specific roles with defined permissions, controlling access to databases and collections.

MongoDB is commonly used in web development, mobile app development, real-time analytics, and other scenarios where flexibility, scalability, and performance are crucial. It has gained popularity for its ease of use, flexibility, and ability to handle unstructured or semi-structured data effectively.

**MongoDB and React.js Usage**

MongoDB and React are often used together in web development to create full-stack applications. MongoDB, a NoSQL document database, serves as the backend database, while React is employed for building dynamic user interfaces on the frontend.

The backend is typically implemented using Node.js and Express.js, establishing RESTful APIs to communicate with the MongoDB database. Mongoose, an ODM library, can be used to simplify interactions with MongoDB. React components make HTTP requests to the backend, fetching and updating data from MongoDB via these APIs.

State management libraries like Redux or the Context API in React help manage application state, including data retrieved from the database. Real-time updates can be facilitated using technologies like Change Streams or WebSocket connections.

The combined application is often deployed using platforms like Netlify or Vercel for the React frontend and Heroku or MongoDB Atlas for the Node.js backend and MongoDB database. Authentication and authorization, crucial for many applications, can be implemented using libraries like Passport.js or JWT. This stack provides a scalable and flexible solution for building modern web applications.

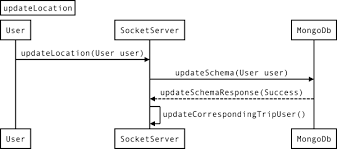


Fig 3 MongoDB Sequence Diagram

React is a widely adopted JavaScript library that is widely used for creating dynamic and interactive user interfaces. It provides a component-based architecture, which means that UIs are built by composing reusable and modular components.

One of the key advantages of React is its component-based approach. Each component in React encapsulates a specific piece of functionality and can be reused throughout the application. This modular structure makes it easier to maintain and update UI elements, as changes made to one component do not affect others. Components can be composed together to create more complex UIs, resulting in code that is easier to read, understand, and maintain.

In addition to its component-based architecture, React also offers tools and libraries for managing application state, routing, and data fetching. The built-in state management mechanism, known as "React state," allows developers to create and manage state within individual components.

React also provides libraries like React Router for handling client-side routing, enabling developers to build multi-page applications with smooth navigation. React's routing capabilities allow for dynamic rendering of components based on URL changes, enhancing the user experience

Furthermore, React integrates well with external data sources, such as APIs, by providing libraries like Axios and fetch for data fetching and manipulation. These libraries enable developers to retrieve and update data from servers, making React a suitable choice for building applications that interact with backend systems.

Overall, React's component-based architecture, coupled with its state management, routing, and data fetching capabilities, make it a popular choice for building complex front-end applications. Its flexibility, reusability, and robust ecosystem of libraries and tools contribute to its widespread adoption in the web development community.

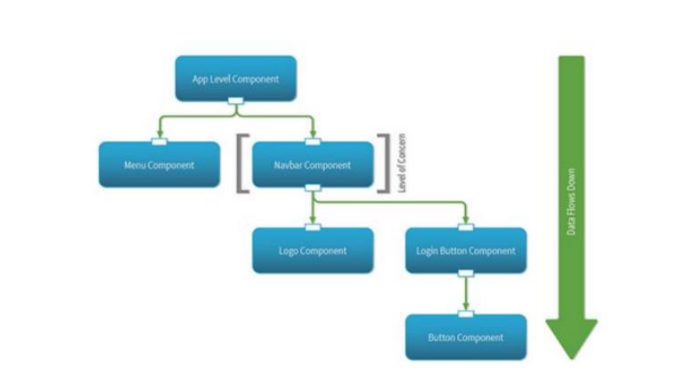


Fig 4 React Flow Diagram

In a MERN (MongoDB, Express.js, React, Node.js) stack project, MongoDB and React work together to create a full-stack web application. Here's a high-level overview of the process:

1. Backend Development (Node.js and Express.js):

- Node.js with Express.js is used to create the backend server.

- Express.js handles routing, server logic, and communication with the MongoDB database.

- Developers define routes and controllers to handle HTTP requests and responses.

2. Database Interaction (MongoDB):

- MongoDB is chosen as the NoSQL database for storing and retrieving data.

- Mongoose, an ODM (Object-Document Mapper) library, is optionally used to provide a schema-based abstraction over MongoDB, making it easier to work with in a Node.js environment.

3. RESTful API Creation:

- The Express.js application exposes RESTful APIs to perform CRUD (Create, Read, Update, Delete) operations on the MongoDB database.

- APIs define endpoints for handling different types of requests (GET, POST, PUT, DELETE) to interact with data.

4. Frontend Development (React):

- React is used on the frontend to build the user interface and manage the UI components.

- React components are created to represent different parts of the application, such as forms, lists, and views.

5. HTTP Requests from React to Node.js:

- React components make HTTP requests to the Express.js backend to interact with the MongoDB database.

- Axios or the native Fetch API is commonly used to make these requests.

6. State Management:

- React uses state management, either through the Context API or state management libraries like Redux, to manage the application's state.

- The state includes data fetched from the MongoDB database and other client-side state.

7. Rendering and User Interaction:

- React components render data received from the backend, providing a dynamic and interactive user interface.

- User interactions, such as form submissions or button clicks, trigger HTTP requests to the backend APIs.

8. Real-Time Updates (Optional):

- Real-time updates can be implemented using technologies like WebSocket connections or MongoDB Change Streams.

- This allows the React frontend to receive instant updates when data changes in the MongoDB database.

9. Authentication and Authorization:

- User authentication and authorization are implemented on the backend using libraries like Passport.js or by using JSON Web Tokens (JWT).

- Secure routes and access control are enforced to protect sensitive data and functionalities.

10. Deployment:

- The combined application is deployed on hosting platforms. Netlify or Vercel are commonly used for the React frontend, while Heroku, AWS, or MongoDB Atlas host the Node.js backend and MongoDB database.

By combining MongoDB and React in a MERN stack, developers create a scalable and efficient architecture for building modern web applications, allowing for flexibility in data handling and providing a dynamic user experience.

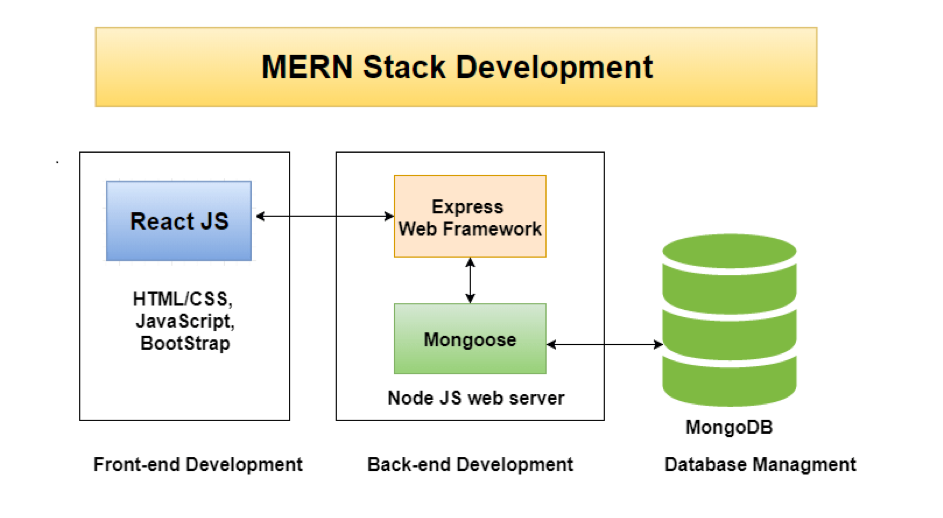


Fig 5 Combination of REACT.JS and MONGODB

1. **Implementation**

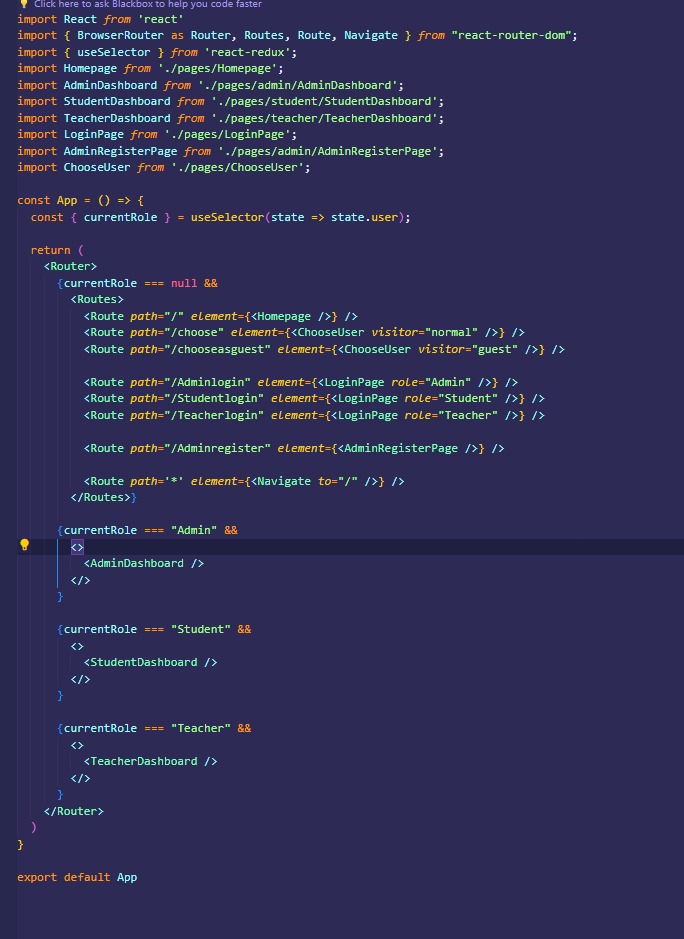
This segment delves into the various aspects of the application's implementation, covering essential steps such as environment setup, view creation, URL mapping, template creation, database configuration, and setting up the admin panel.

**4.1 Environmental-Setup**

To begin working with MongoDB, the first step is to ensure that Ecpress.js is installed on the machine since Django is a Express.js web framework. Here's an expanded explanation of the installation process:

* **Express.js installation** : To set up an Express.js app, first, ensure Node.js is installed. Create a new project, run `npm init -y` to initialize, and install Express with `npm install express`. Create a file (e.g., `app.js`) with a basic Express app, define routes, and start the server. Access your app at http://localhost:3000. Optionally, use tools like `nodemon` for automatic restarts during development.
* **MongoDB installation:** To use MongoDB in a Node.js project, ensure Node.js is installed. Install the MongoDB Node.js driver with `npm install MongoDB`. Create a connection to MongoDB in your app, define a schema, and perform CRUD operations. Optionally, use Mongoose for a more structured approach. Remember to handle asynchronous operations using callbacks or promises.

By following these steps, we successfully installed both Express.js and MongoDB, setting up the necessary environment to start building Django applications. With MongoDB installed, we could leverage its powerful features and tools to develop web applications with ease.

**4.2 Writing views**

4.1 Views for Homepage

In a MERN (MongoDB, Express.js, React, Node.js) stack application with student, teacher, and admin pages:

1. MongoDB (Database):

- MongoDB stores data for students, teachers, and admin roles.

- Collections can be created for each entity, defining their attributes and relationships.

- MongoDB Atlas may be used for cloud-based database hosting.

2. Express.js (Backend):

- Express.js handles server-side logic and exposes RESTful APIs.

- API routes are defined to perform CRUD operations on student and teacher data.

- Authorization middleware ensures admin-only access to certain routes.

3. Node.js (Server):

- Node.js runs the server, integrating with Express.js for routing and handling HTTP requests.

- It connects to the MongoDB database using the MongoDB Node.js driver.

4. React (Frontend):

- React components represent pages for students, teachers, and admins.

- Components interact with the Express.js APIs to fetch and update data from MongoDB.

- React Router may be used for navigation between student, teacher, and admin pages.

5. State Management (React Context/Redux):

- State management libraries (React Context or Redux) may be used to manage application state, handling data fetched from the backend.

6. React Components:

- Student Page:

- Displays student-related information.

- Allows students to view their details and perform actions related to their role.

- Teacher Page:

- Displays teacher-related information.

- Allows teachers to view their details and perform actions related to their role.

- Admin Page:

- Displays admin-related information.

- Provides tools for admins to manage student and teacher data.

7. Authentication and Authorization:

- Implement user authentication using JWT or sessions.

- Define roles (student, teacher, admin) and enforce access controls based on roles.

8. Styling and UI:

- Use CSS frameworks or libraries for consistent styling.

- Responsive design ensures a seamless experience across devices.

9. Deployment:

- Deploy the frontend (React) on platforms like Netlify or Vercel.

- Deploy the backend (Node.js and Express.js) and database (MongoDB) on platforms like Heroku or MongoDB Atlas.

10. Testing and Debugging:

- Implement unit tests for components and backend routes.

- Use debugging tools to identify and resolve issues during development.

11. Real-time Updates (Optional):

- Implement real-time updates using technologies like WebSocket connections or MongoDB Change Streams to reflect changes instantly.

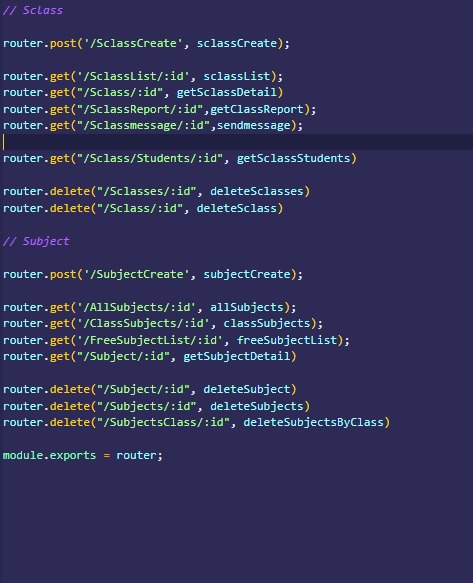
12. Security:

- Implement secure coding practices.

- Sanitize inputs, validate user data, and protect against common web vulnerabilities.

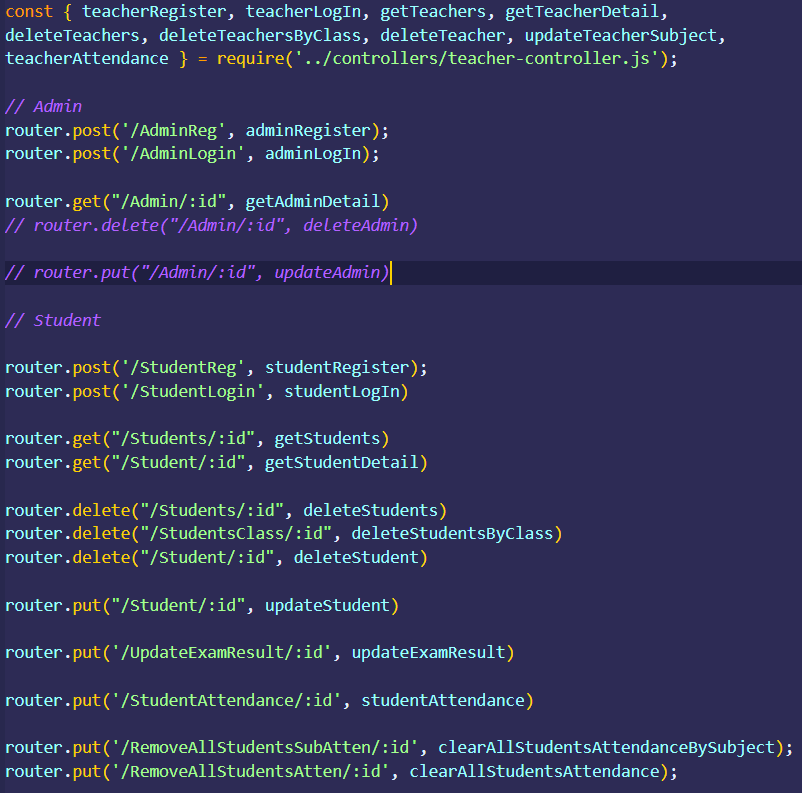
By combining MongoDB, Express.js, React, and Node.js, your MERN stack application can provide a robust and scalable solution for managing student, teacher, and admin functionalities.

**4.3 Mapping URLs**



4.2 Mapping the Url’s

In Express.js, URL mapping is efficiently organized by creating a dedicated "routes" folder. Each route file within this folder encapsulates specific URL endpoints, enhancing code modularity and readability. For instance, you might have a "students.js" file handling routes related to student operations and an analogous "teachers.js" file for teacher-related routes. Within these files, Express's Router() function is utilized to define distinct routes along with their associated handlers. These route files are then imported into the main Express application file (typically "app.js" or "index.js") using `app.use()` to establish a clear and modular structure. This method not only simplifies code maintenance but also allows for straightforward addition or modification of routes as the project evolves

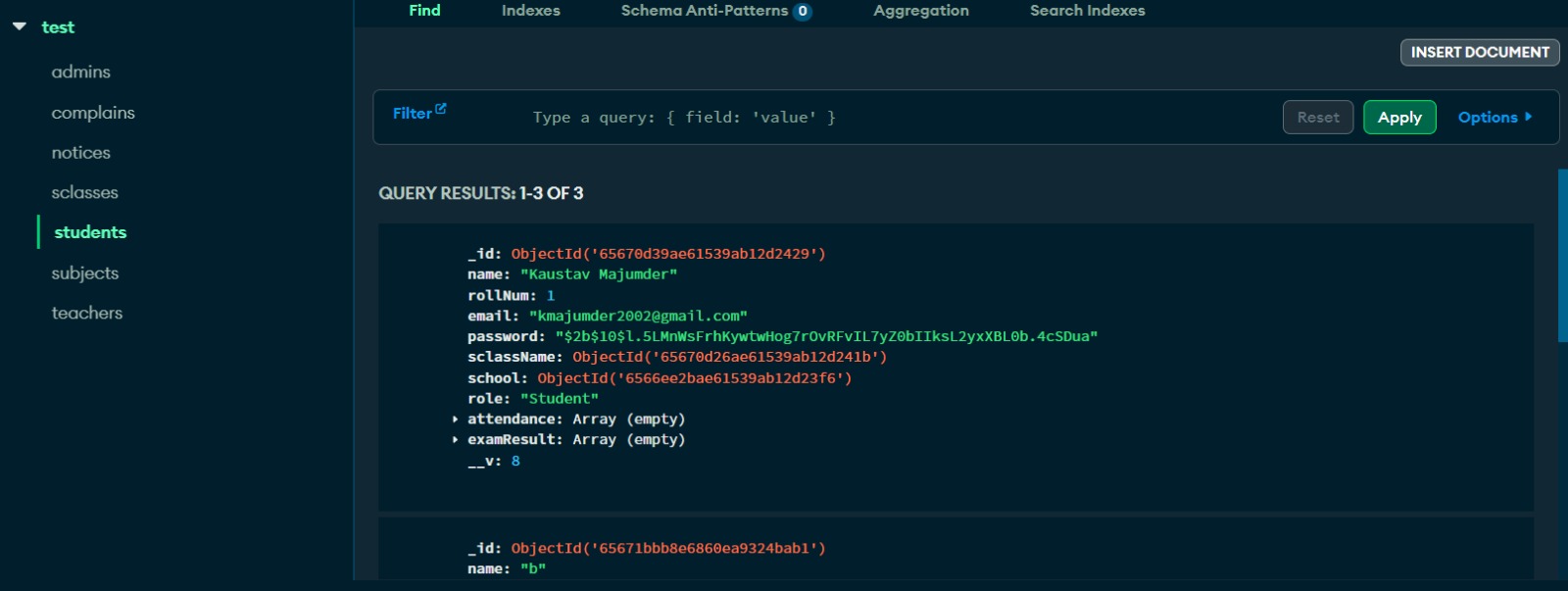
4.3 Routes 1

4.4 Routes 2

**4.4 Setting up database and admin panel**

Since Attendance Management System is an application that is predominantly read-only or requires a smaller installation footprint, we have chosen to use the database. Before setting up the custom tables for our database, we needed to load the default ones by running the migrations using the command: `express,js manage.py migrate`. This command, when executed, built the database for us.

Now that our database is ready to be used, we need to create tables within it. To accomplish this, we need to create models. A model serves as the single, definitive source of information about the data we are storing. It encompasses the essential fields and behaviours of the data. In general, each model corresponds to a single database table.



4.5 Setting up database and admin panel

By defining models, we can specify the structure and properties of our data. This includes the fields (such as text, numbers, dates, etc.) that will be stored in the database and any associated behaviours or methods related to the data. Models provide a convenient way to interact with the database, allowing us to perform operations like creating, reading, updating, and deleting records

By utilizing models, we can easily manage and manipulate our data within the SQLite database for the Attendance Management System application.

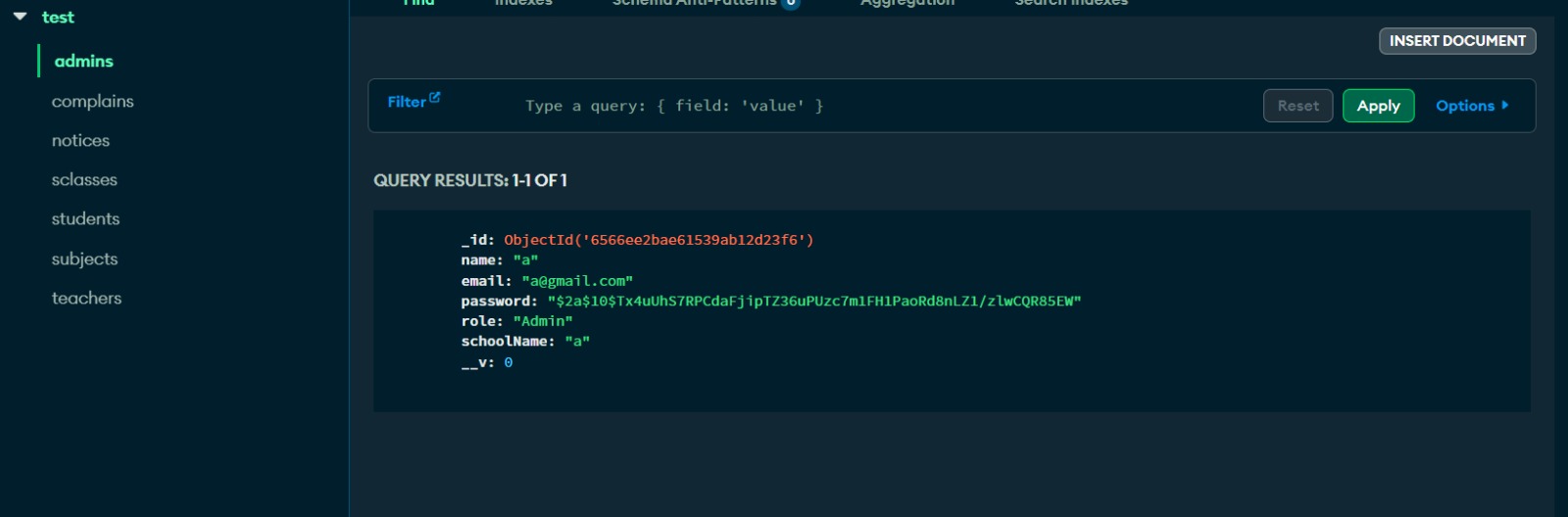
In this project, we have chosen not to utilize any third-party database management tools. Instead, we have leveraged the power of the MongoDB Admin Panel to handle our database operations.

To accomplish this, we started by creating a superuser account. This can be done using the command: `python manage.py creates upper user`. Upon executing this command, we were prompted to enter a username and password for the superuser. This superuser account grants us administrative privileges and allows us to access and manage the MongoDB Admin Panel.

Once the superuser account is set up, we can log in to the MongoDB Admin Panel using the credentials we provided during the superuser creation process. This provides us with a user-friendly interface where we can perform all the CRUD (Create, Read, Update, Delete) operations on our database.

Through the MongoDB Admin Panel, we can add, edit, and delete records in our database, as well as view and search for specific data entries. This eliminates the need for manual database queries or the use of external tools. The Admin Panel simplifies the process of managing our database by providing a convenient and intuitive graphical interface.

By leveraging the MongoDB Admin Panel and the superuser account, we can efficiently handle our database-related tasks and perform necessary operations without relying on third-party tools or complicated command-line operations



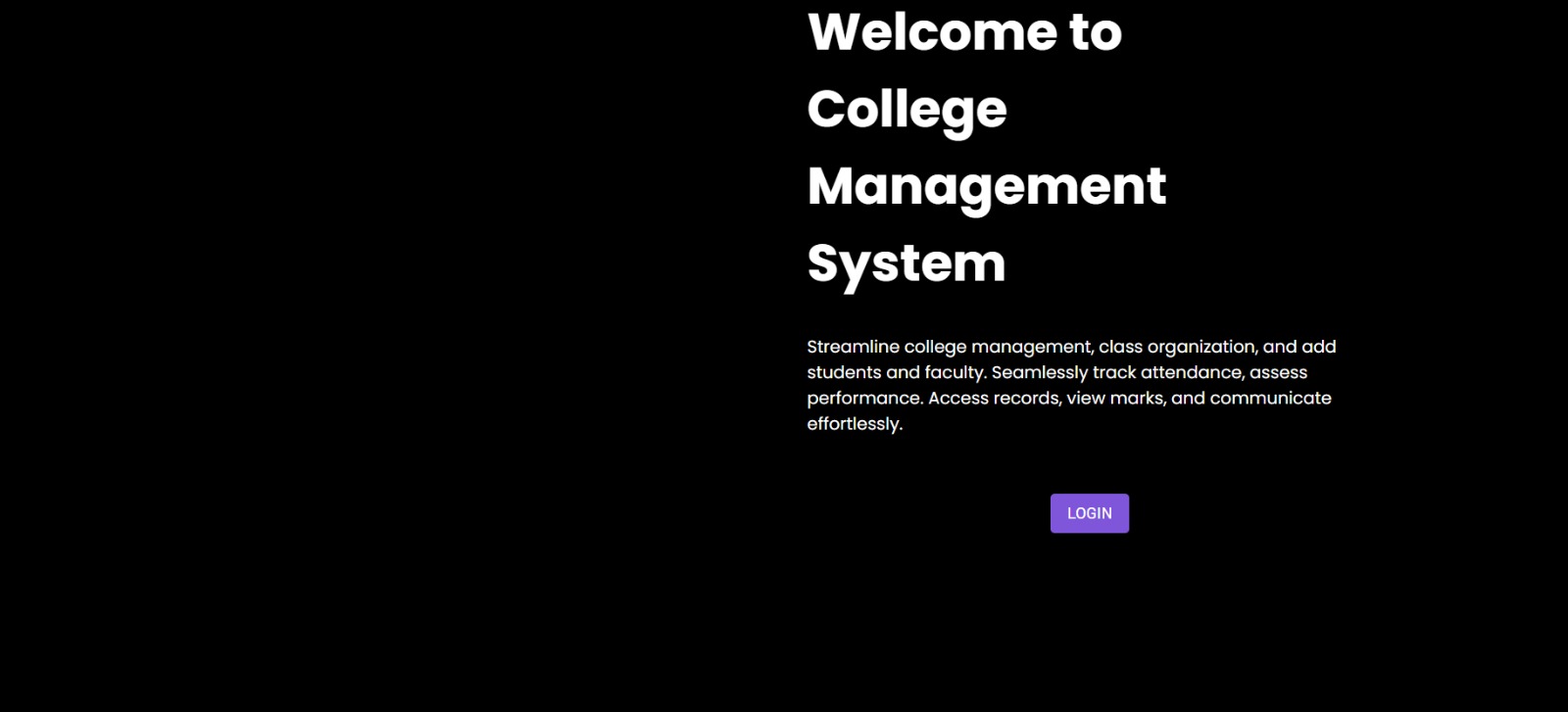
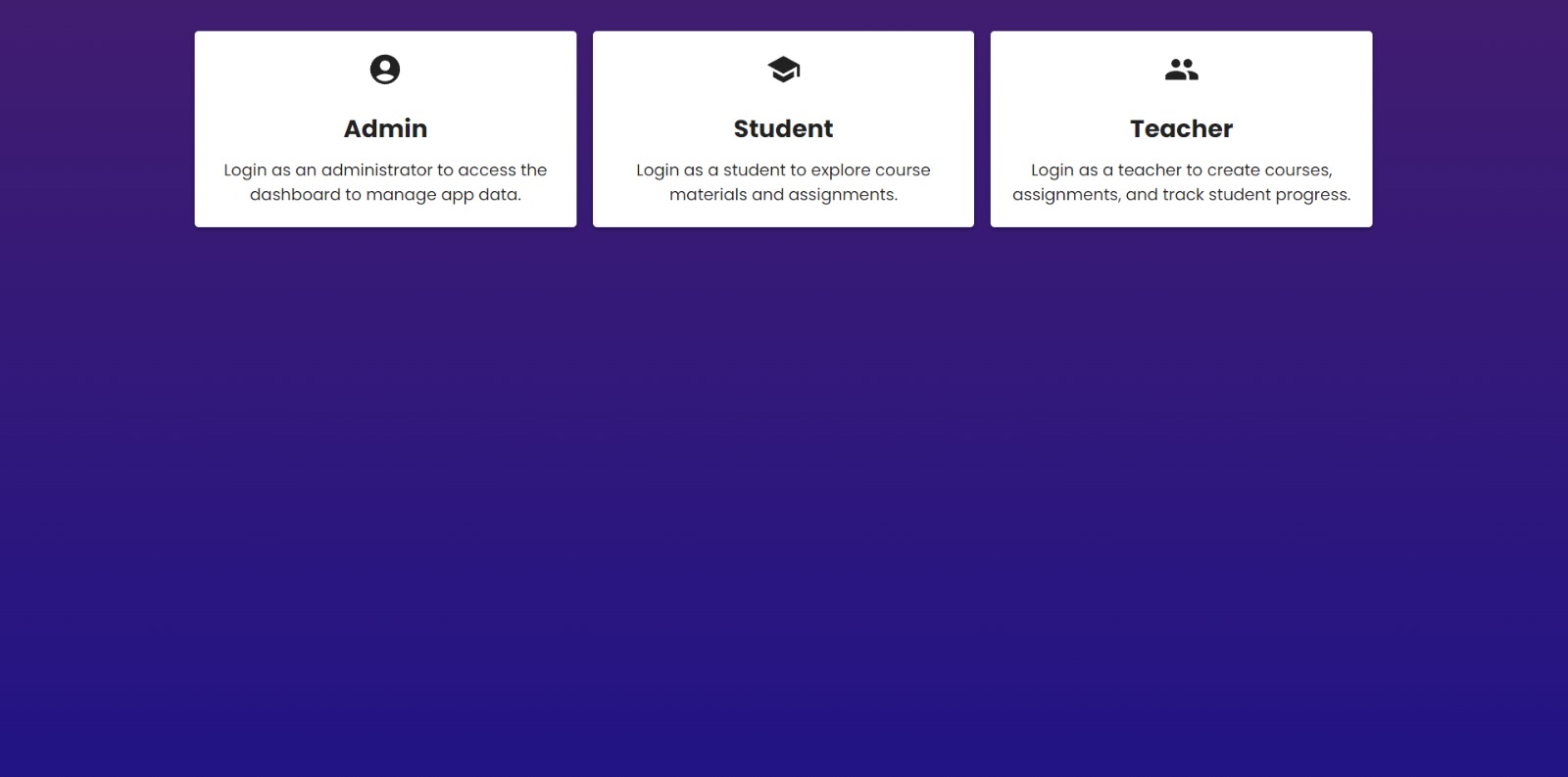
1.  **Working and Results**

Fig 5.1 Home page View 

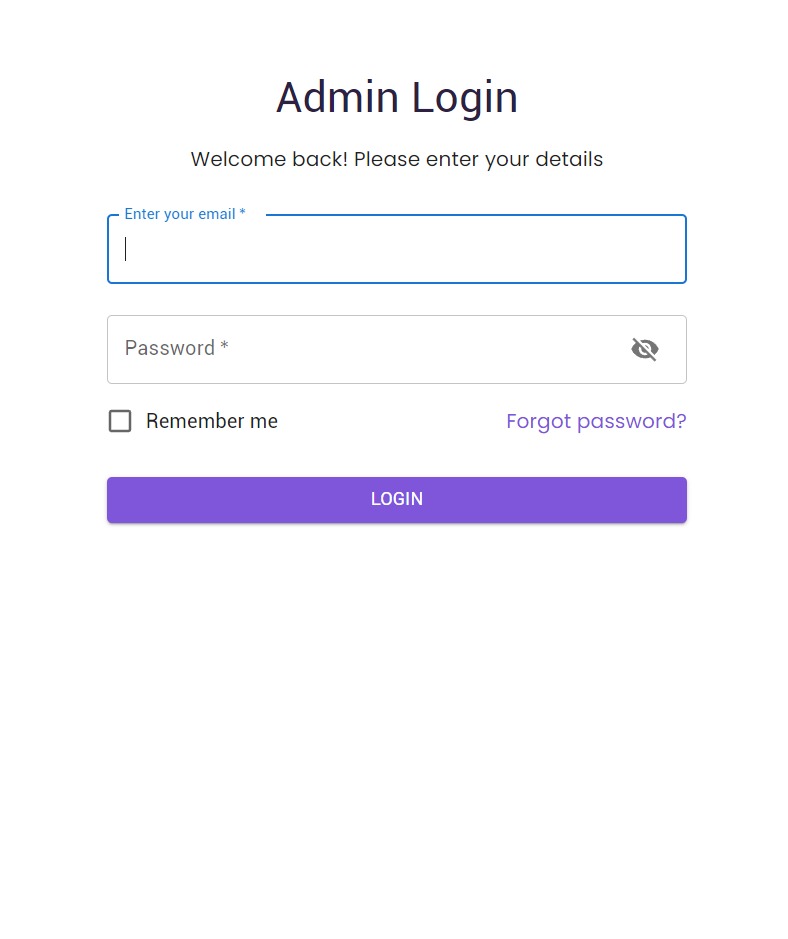
Fig 5.2 Desktop View

Fig 5.3 Admin login View

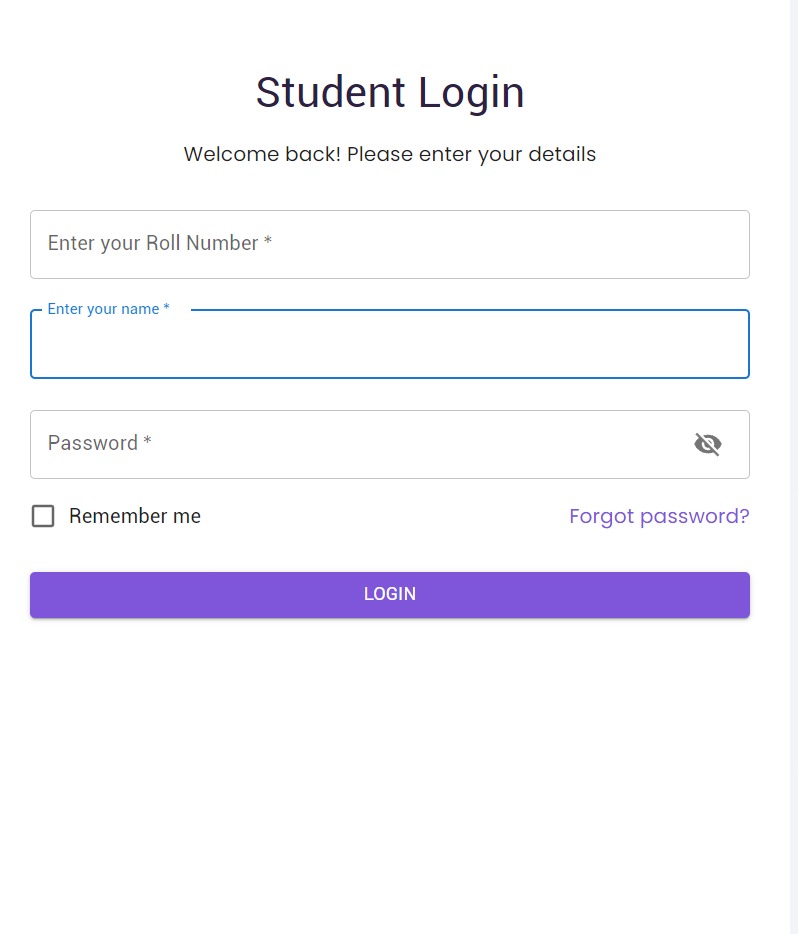
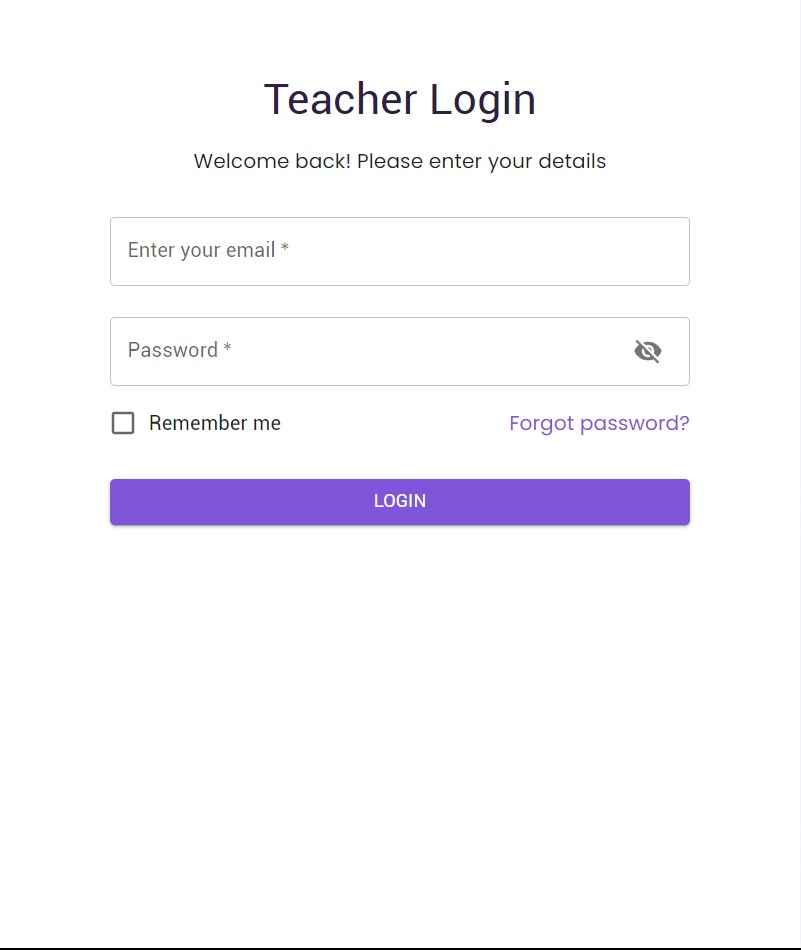
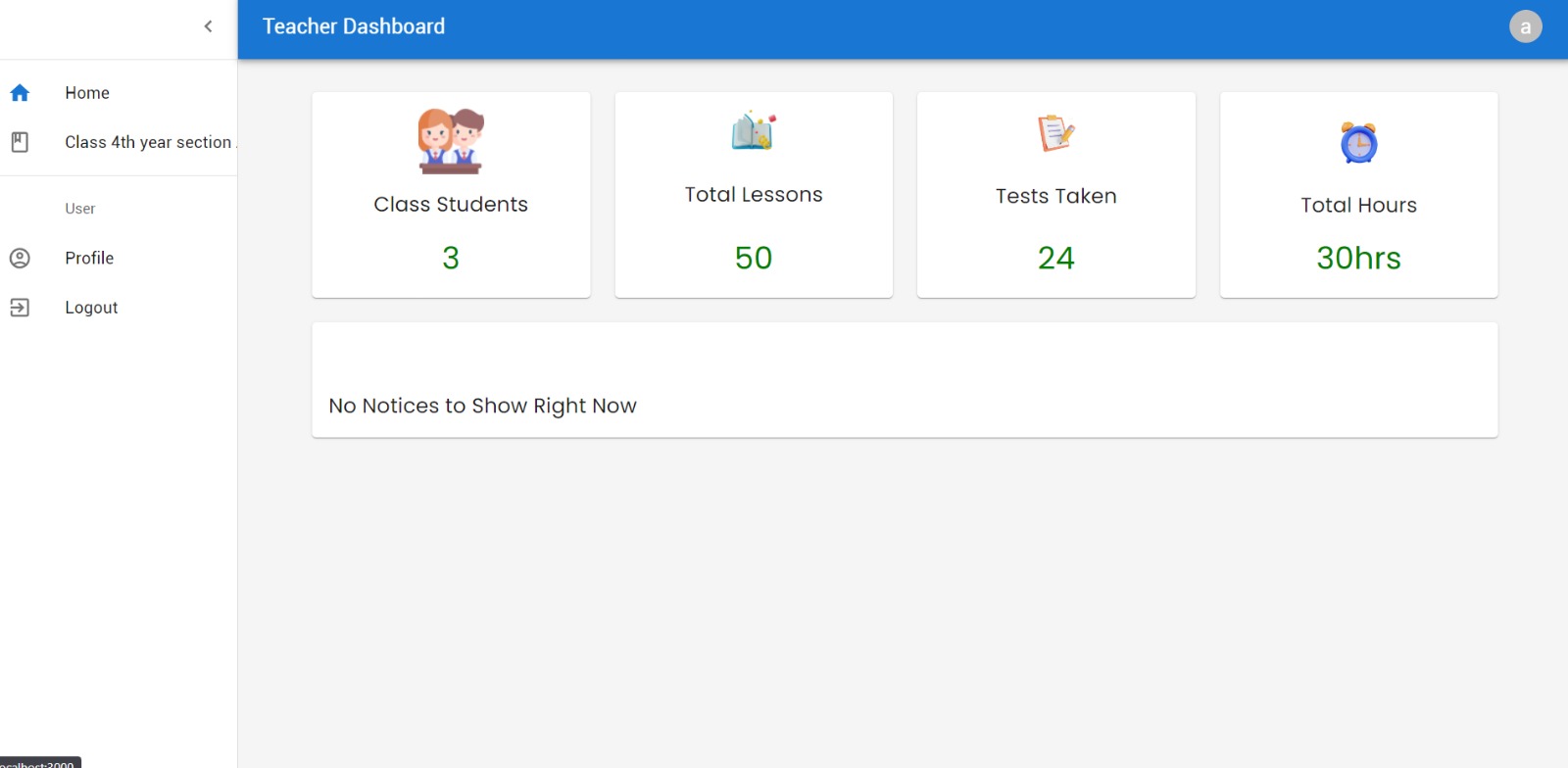
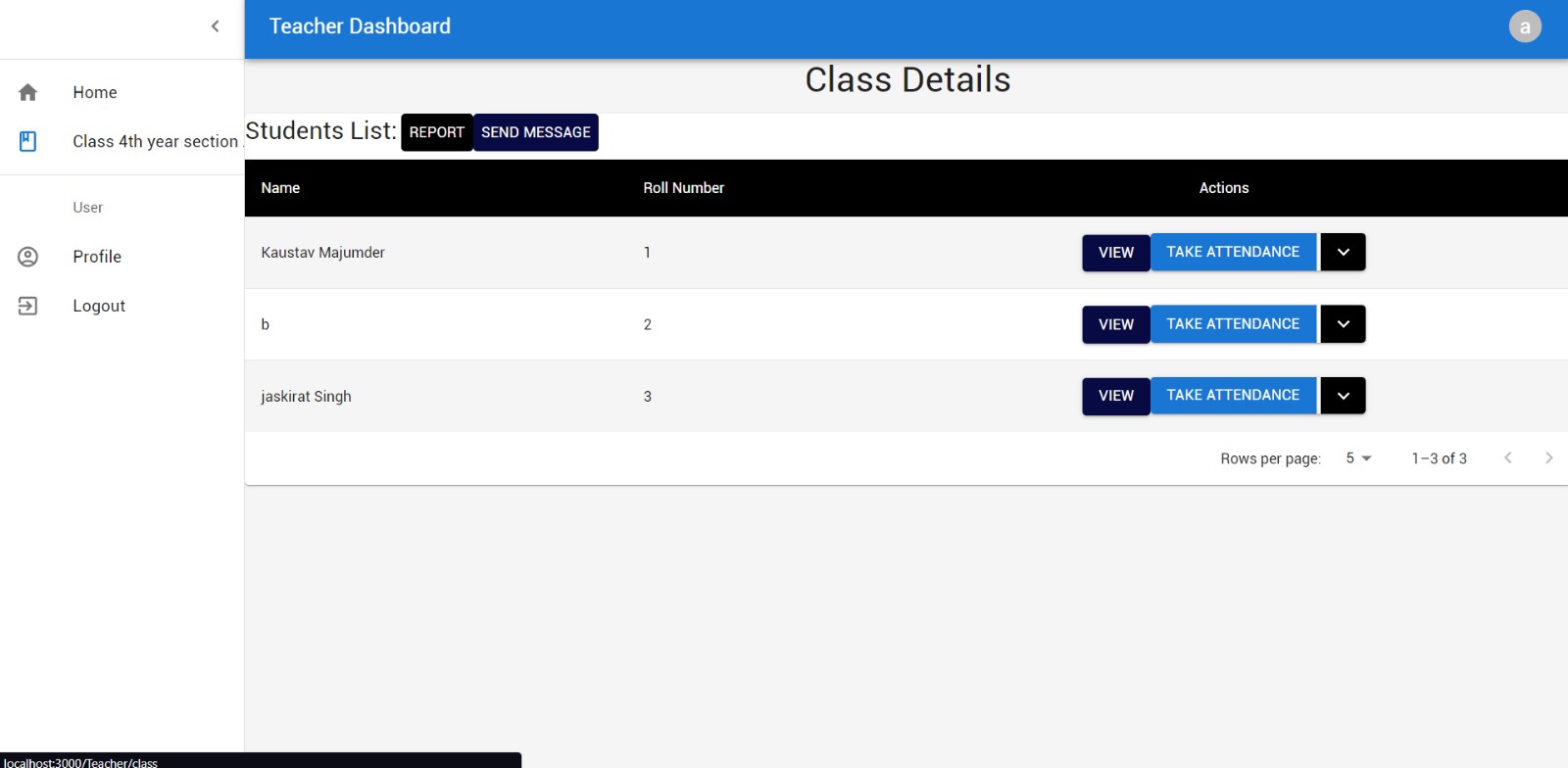


Fig 5.4 Student login View Fig 5.5 Teacher login View





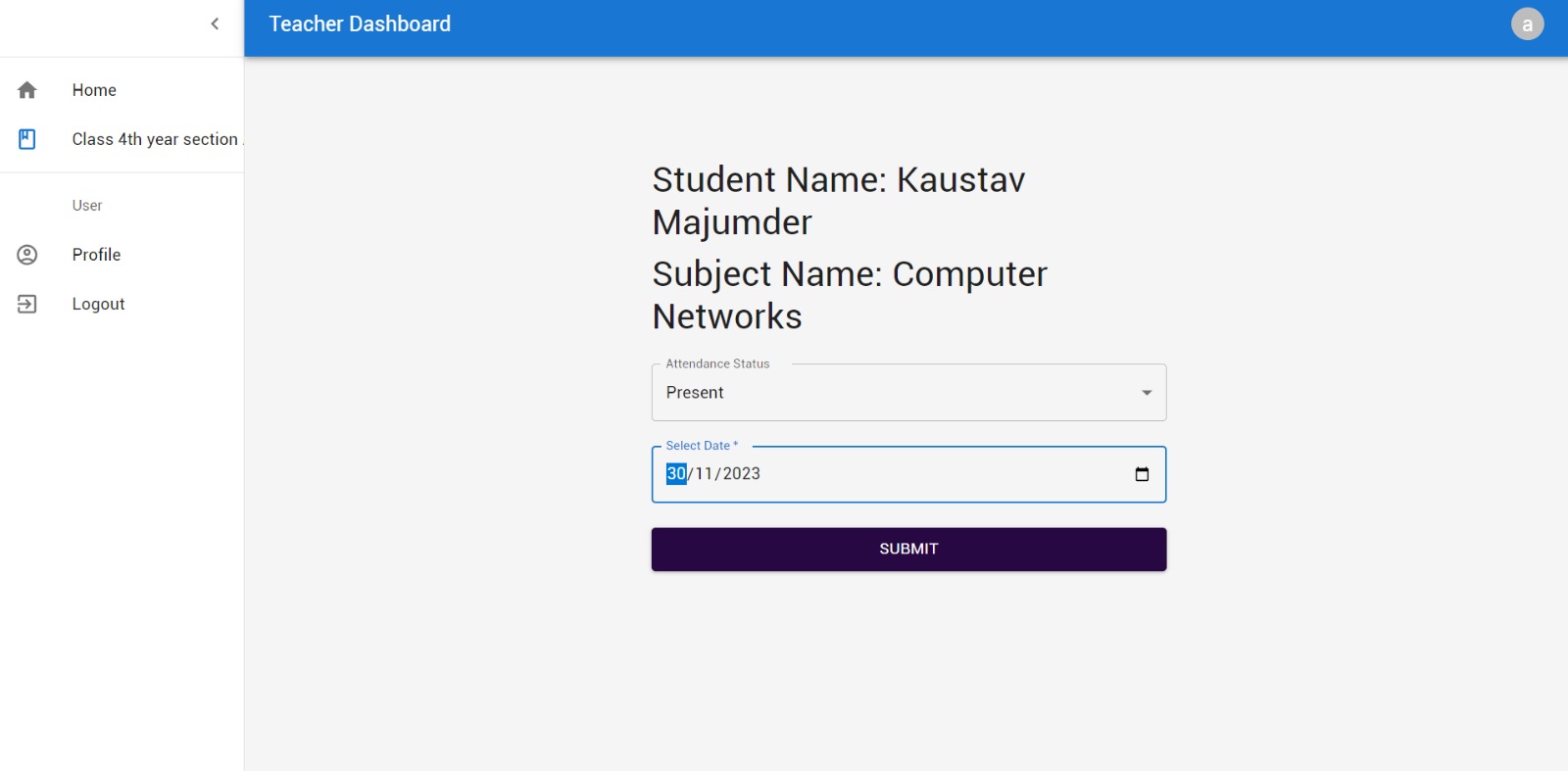


Fig 5.6 Teacher Dashboard

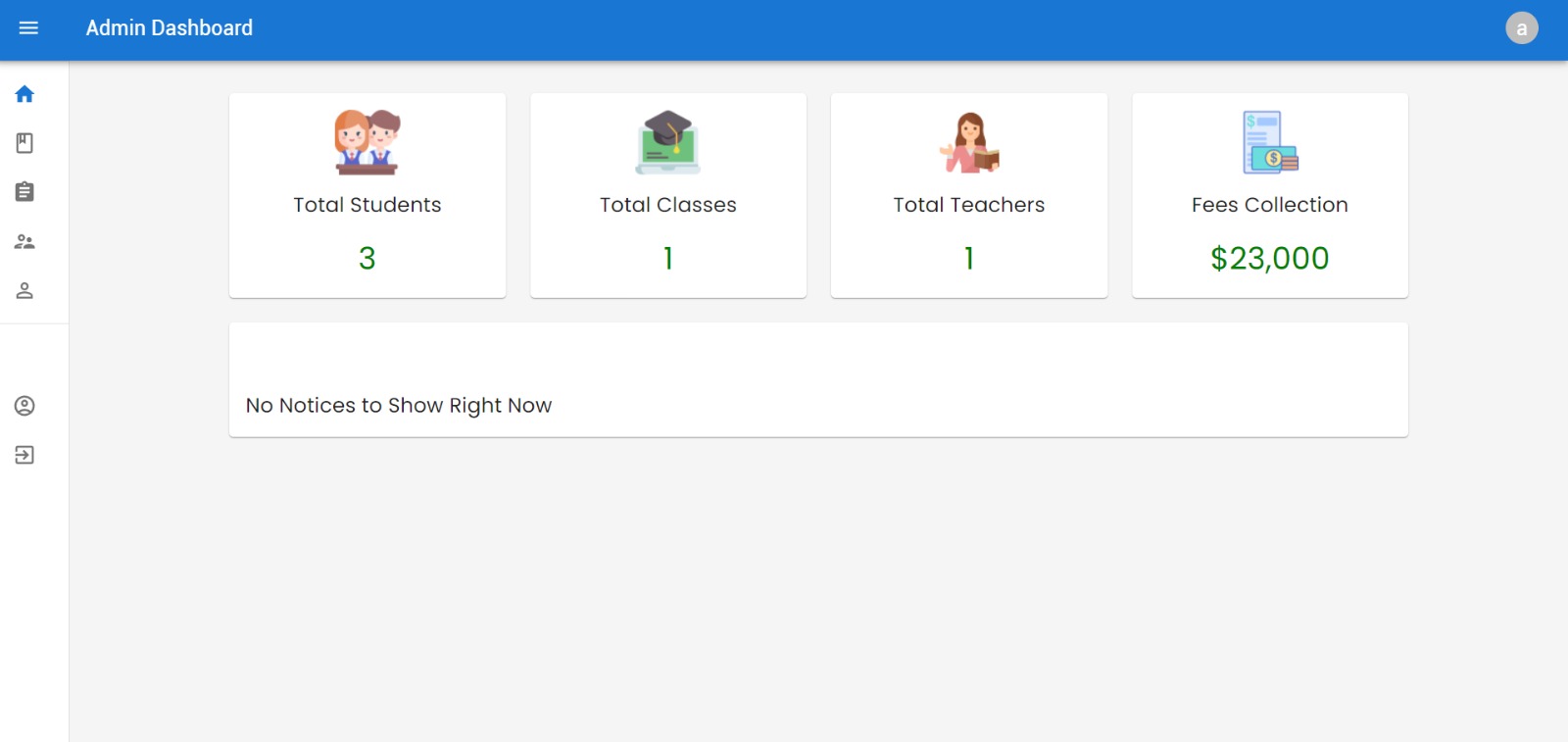


Fig 5.7 Attendance view

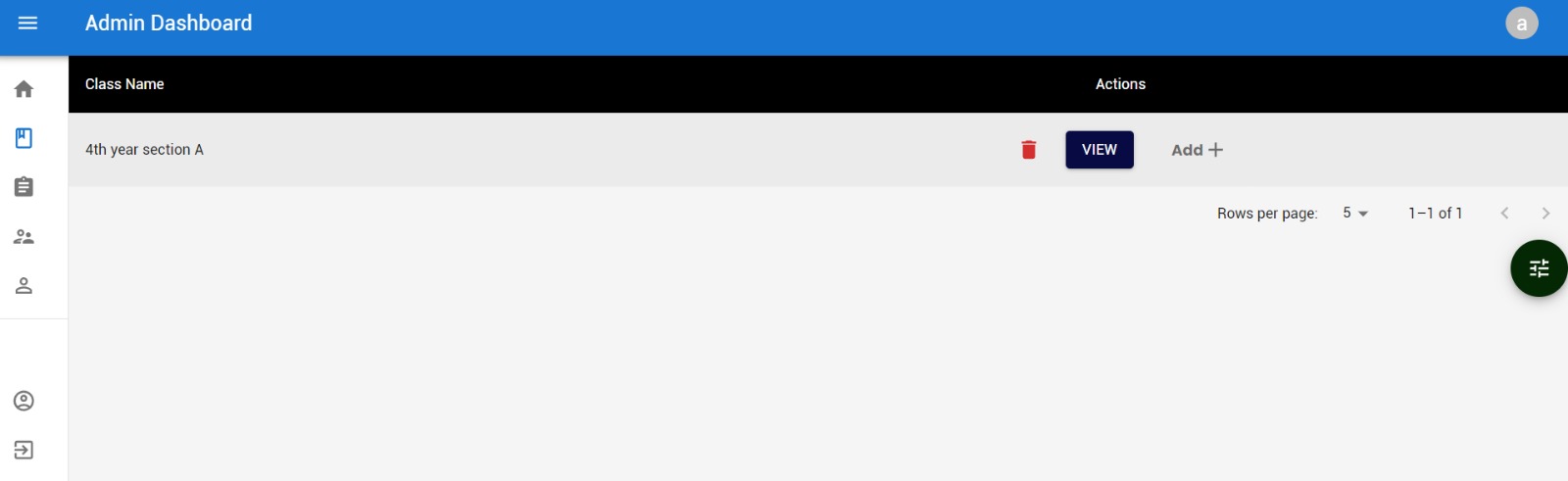


Fig 5.8 Admin Dashboard

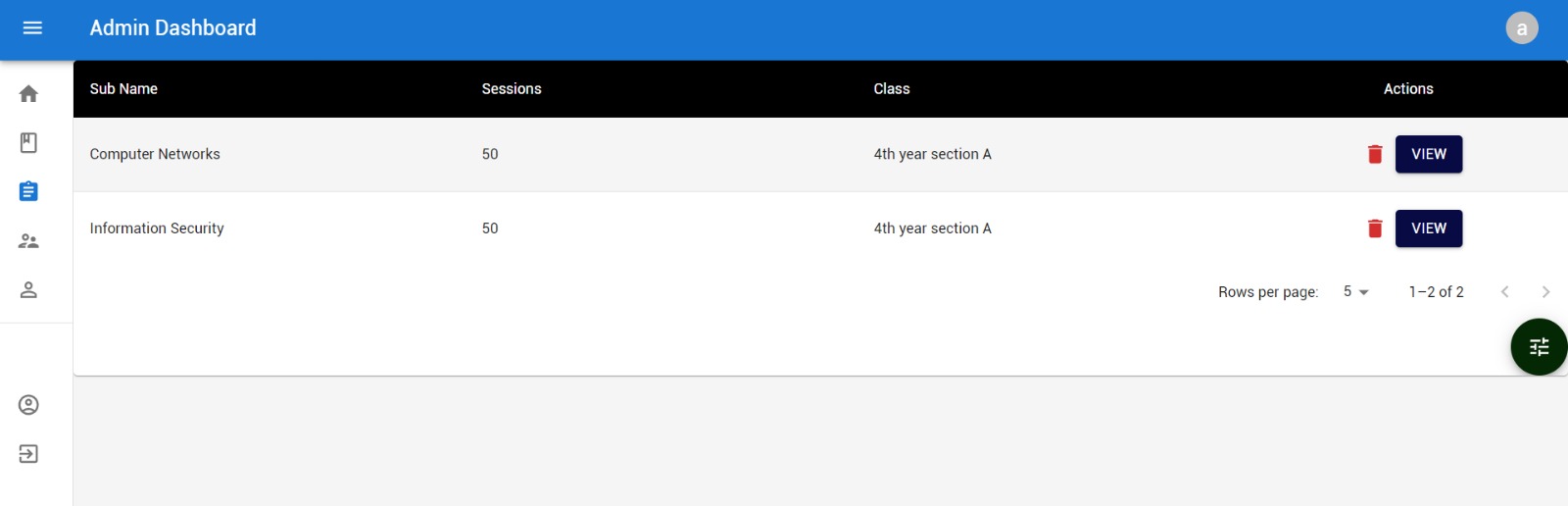


Fig 5.9 Class Creation

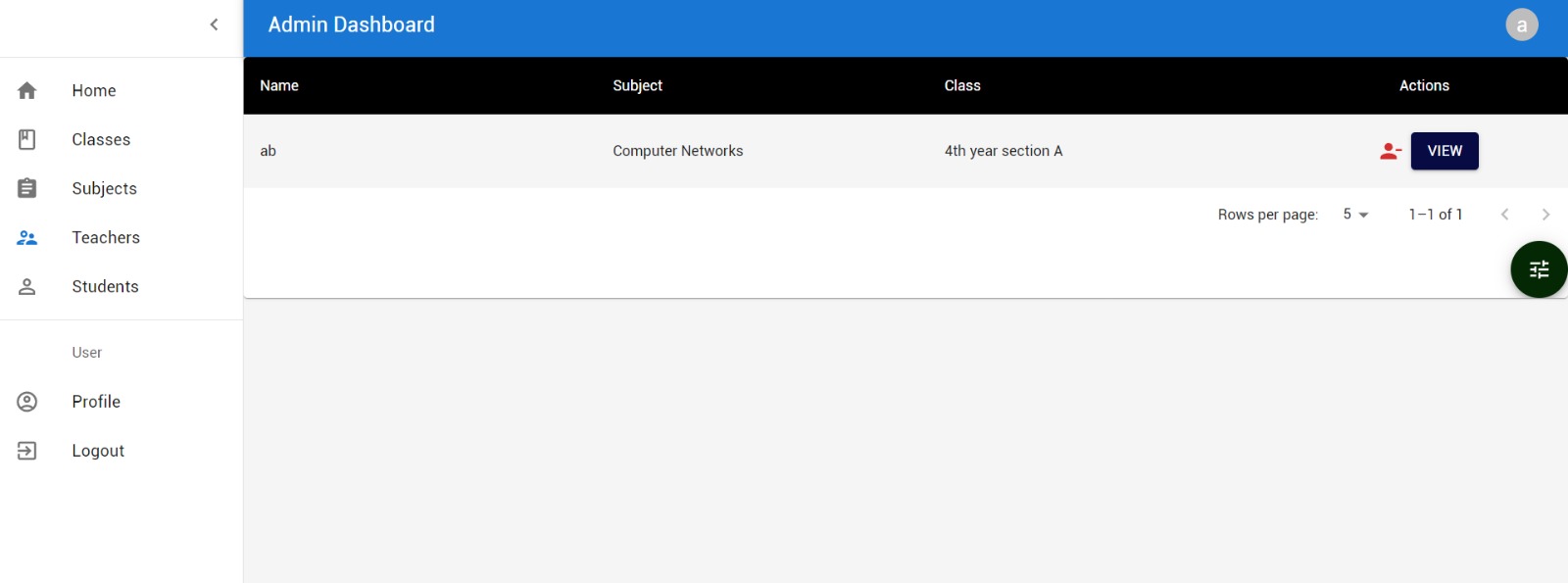


Fig 5.10 Subject Creation

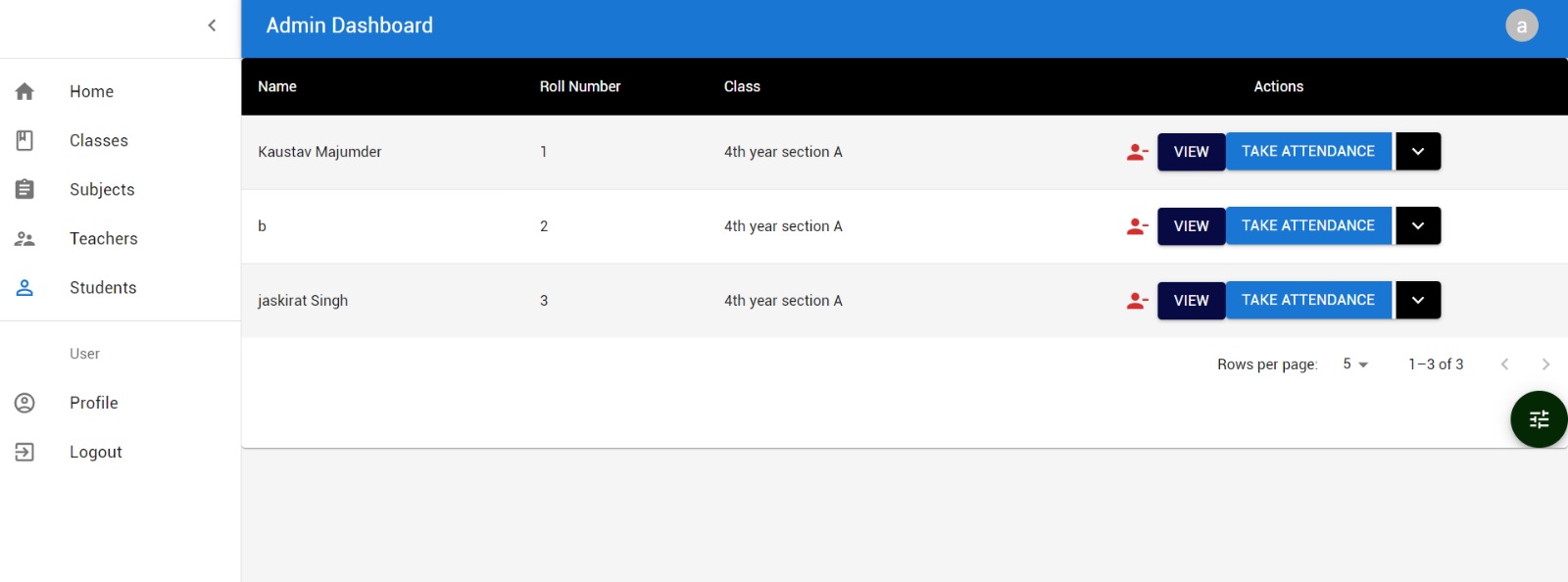


Fig 5.11 Teacher Addition

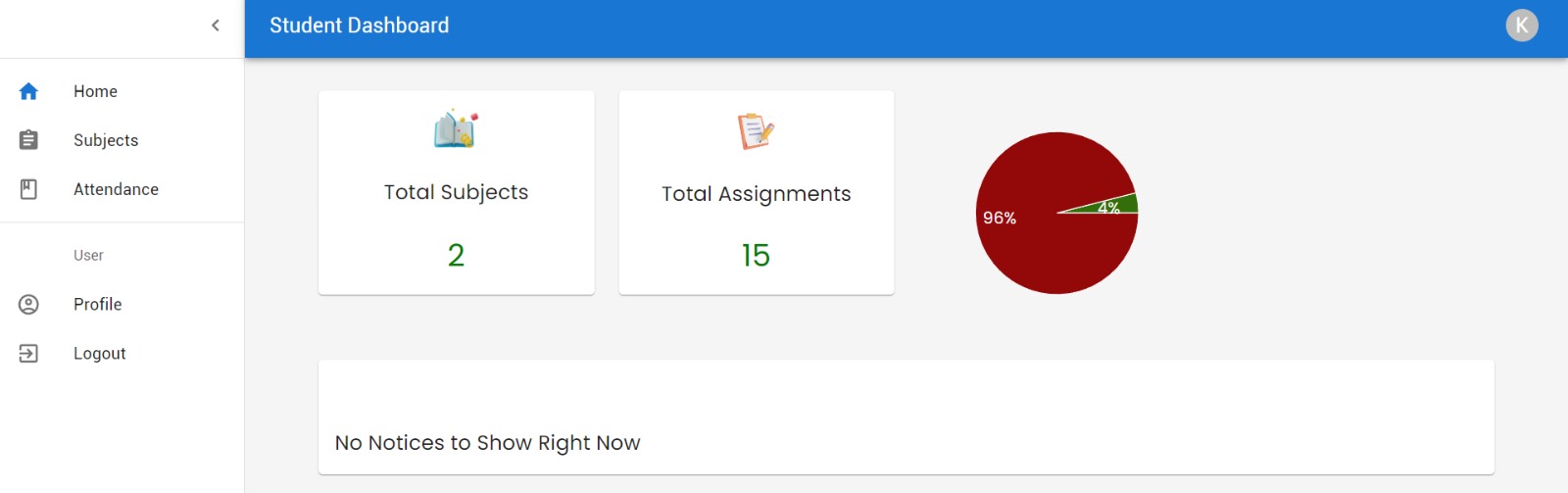
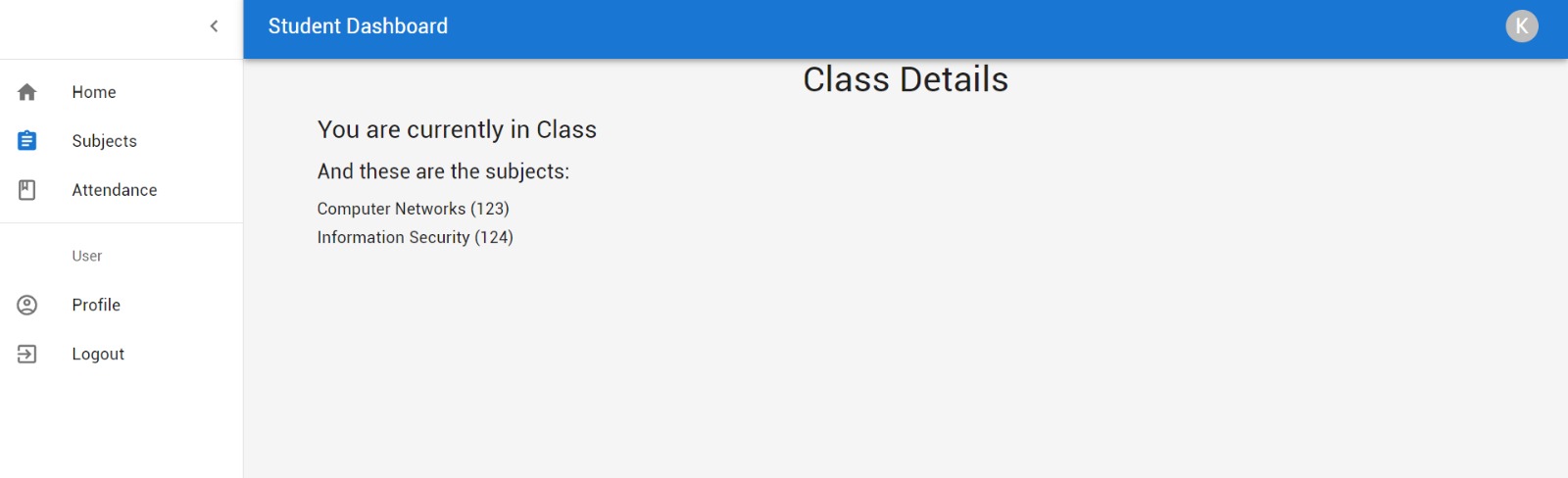
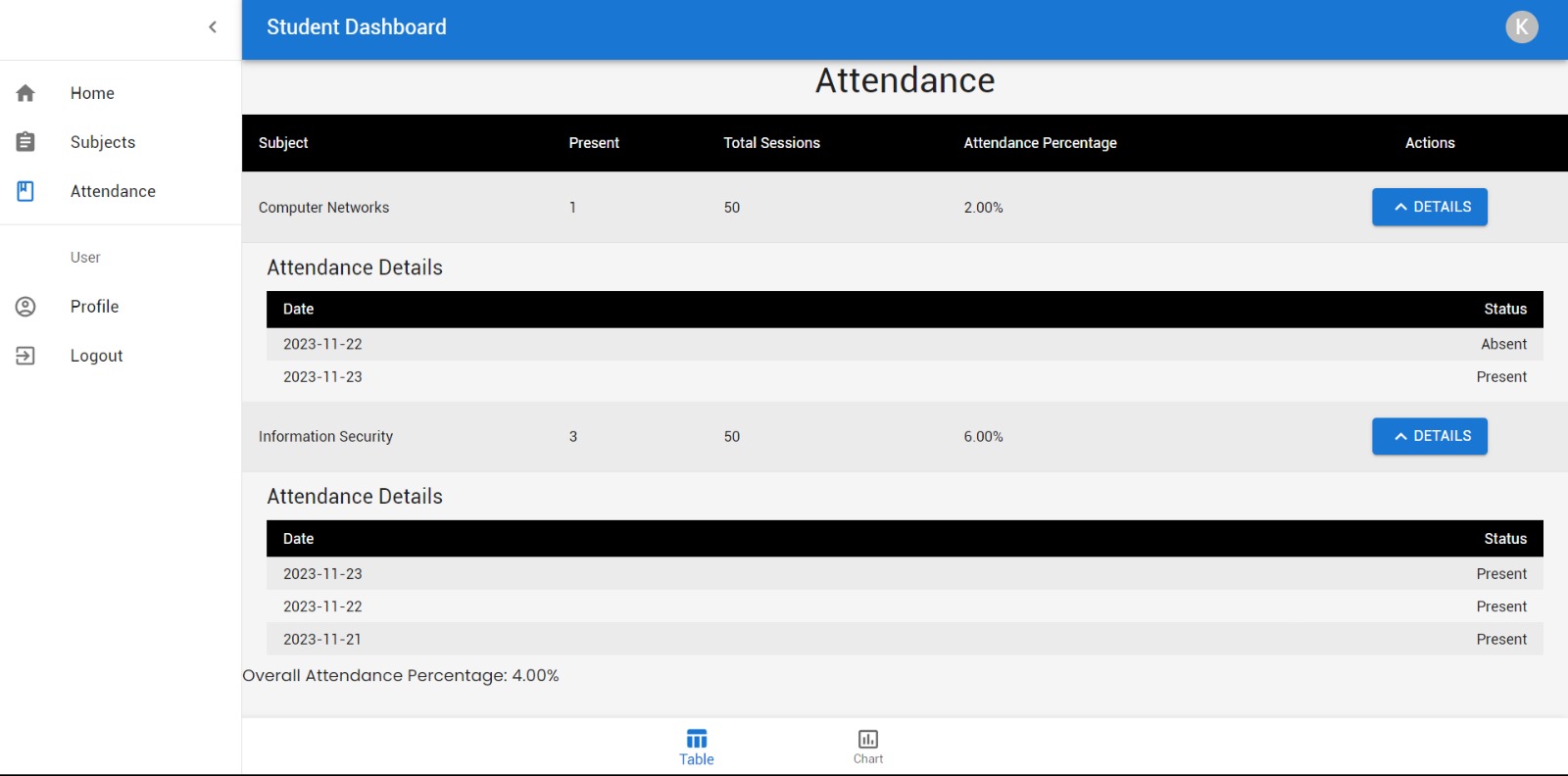


Fig 5.12 Student Addition





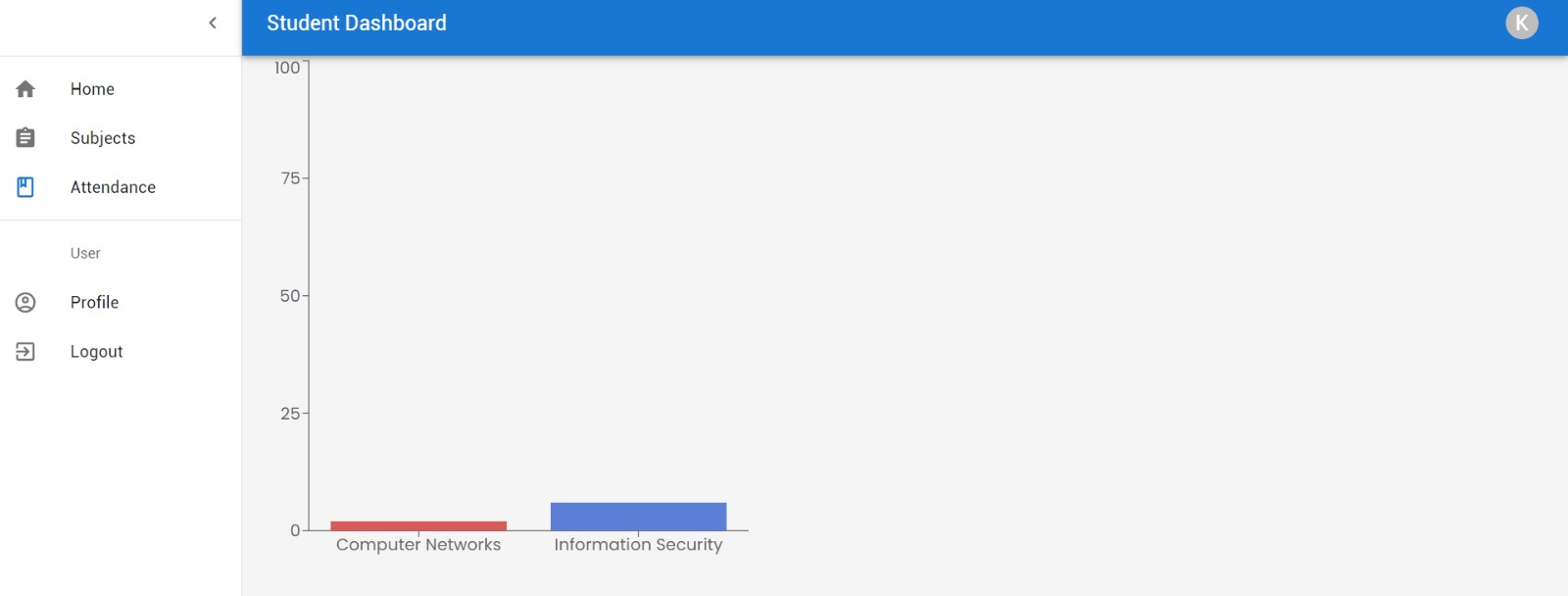


Fig 5.13 Student Dashboard View

**REFERENCES**

(PDF) A Research Paper on College Management System | suraj kumar - Academia.edu https://www.academia.edu/35401042/A\_Resear https://iarjset.com/wp-content/uploads/2023/06/IARJSET-ICMART-3.pdf https://www.mongodb.com/ https://www.academia.edu/36957671/An\_Innovative\_Information\_System\_for\_College\_Managem ent

https://www.academia.edu/7760318/STUDENTS\_ATTENDANCE\_MANAGEMENT\_SYSTEM\_

MINI\_PROJECT\_REPORT\_MASTER\_OF\_COMPUTER\_APPLICATIONS

<https://www.academia.edu/7760318/STUDENTS_ATTENDANCE_MANAGEMENT_SYSTEM_MINI_PROJECT_REPORT_MASTER_OF_COMPUTER_APPLICATIONS>

<https://www.researchgate.net/publication/362405196_A_Literature_Review_on_Smart_Attendance_Systems>

<https://www.academia.edu/35401042/A_Research_Paper_on_College_Management_System>