

**iSchool**

*PROJECT REPORT*

**Submitted in partial fulfillment of the requirements for the award Degree of**

**BACHELOR OF COMPUTER APPLICATIONS**

**Of the University of Calicut**

*Submitted By*

<b>RASHIN KP</b>	<b>(NZAVBCA018)</b>
<b>MUHAMMED NUHMAN CK</b>	<b>(NZAVBCA014)</b>
<b>MUHAMMED ASHIN P</b>	<b>(NZAVBCA013)</b>

**Under the supervision of**

**Mrs. NABEELA**

(Asst. Professor of Computer Science Department)



**DEPARTMENT OF COMPUTER SCIENCE**

**NASRA COLLEGE OF ARTS & SCIENCE**

**THIRURKAD -679325**

**2023– 2024**

**DEPARTMENT OF COMPUTER SCIENCE**

**NASRA COLLEGE OF ARTS & SCIENCE**

*(Affiliated to university of Calicut)*

**TIRURKAD - 679325**



**PROJECT WORK**

**CERTIFICATE**

Certified that this is a bonafied record of the project work

Done by

**iSchool**

**RASHIN KP**

**(NZAVBCA018)**

**MUHAMMED NUHMAN CK**

**(NZAVBCA014)**

**MUHAMMED ASHIN P**

**(NZAVBCA013)**

Submitted in partial fulfilment of the requirements for the award of the  
Degree Bachelor of Computer Application of the University of Calicut

Faculty Guide

Head of the Department

External Examiner

**DEPARTMENT OF COMPUTER SCIENCE**

**NASRA COLLEGE OF ARTS & SCIENCE**

TIRURKAD, MALAPPURAM – 679325



**CERTIFICATE**

This is to certify that this Project report entitled "iSchool" is a bonafied record of the project done by, RASHIN KP, MUHAMMED ASHIN P, MUHAMMED NUHMAN CK of sixth semester BCA, Nasra College Of Arts And Science, Tirurkad, towards partial fulfilment of the requirement for the award of degree of Bachelor of Computer Applications University of Calicut

**Internal Guide**

**Head of Department**

**NABEELA**

**Nabeela (HOD)**

Department of Computer Science

Department of Computer Science

Nasra College Arts&Science, Thirurkkad

Nasra College Arts&Science, Thirurkkad

**Submitted for Viva-Voice Examination held on .....**

**INTERNAL EXAMINER**

**EXTERNAL EXAMINAR**

**Name:**

**Name:**

**Signature:**

**Signature:**

**Date:**

**Date:**

## **DECLARATION**

I hereby that this project work entitled ' iSchool' submitted at Nasra College of Arts & Science (Affiliated to University of Calicut) is a record of original work done us under the supervision and guidance of Mrs. NABEELA (Asst. Professor of Computer Science Department).

**RASHIN KP(NZAVBCA018)**

**MUHAMMED NUHMAN CK (NZAVBCA014)**

**MUHAMMED ASHIN P(NZAVBAC013)**

Signature of Candidate

**RASHIN KP:**

**MUHAMMED NUHMAN CK:**

**MUHAMMED ASHIN P:**

## **ACKNOWLEDGEMENT**

First of all, we would like to express our sincere gratitude to God Almighty for the constant love and grace he has showered upon us. We express our gratitude to the respected principal Dr. P. Zubair, Nasra College of Arts & Science Tirurkad, for us this golden opportunity to undertake our course in this college. We also express our great deal of gratitude to the Head of Department and our internal guide, Mrs. NABEELA for her effective guidance, timely suggestions, and encouragement. We also express our gratitude to all teachers in Department of Computer Science.

## SYNOPSIS

The iSchool offers a transformative solution for modern educational institutions. With its user-friendly interface and comprehensive features, iSchool optimizes school operations, fosters transparency, and empowers all stakeholders within the educational ecosystem. Through seamless integration of frontend technologies like HBS, CSS, and JavaScript, coupled with a robust backend powered by Node.js, Express.js, and MongoDB, iSchool ensures efficient data management, secure user authentication, and role-based access control. From admissions management to attendance tracking, grade storage, and event scheduling, iSchool revolutionizes administrative tasks, enhancing communication, collaboration, and engagement among students, teachers, parents, and administrators.

Furthermore, iSchool lays the foundation for future enhancements, including integration with online payment gateways, development of a mobile application for on-the-go access, implementation of advanced reporting and analytics tools, and incorporation of e-learning features. By continually evolving to meet the evolving needs of educational institutions, iSchool remains at the forefront of innovation, driving efficiency, effectiveness, and empowerment in education.

# CONTENTS

<b>1: INTRODUCTION .....</b>	<b>1</b>
1.1 Introduction.....	2
1.2 Objectives.....	10
1.3 Identification of the Problem .....	13
1.4 Aim.....	14
1.5 Existing System.....	18
1.6 Proposed System .....	21
1.7 Feasibility Study .....	23
 <b>2: SYSTEM REQUIRMENT SPECIFICATION .....</b>	 <b>26</b>
2.1 Software Requirements .....	27
2.1.1 For Admin .....	27
2.1.1 For User.....	28
2.2 Hardware Requirements .....	28
 <b>3: SOFTWARE IMPLEMENTATION .....</b>	 <b>32</b>
3.1 Front-End .....	33
3.2 Back-End.....	37
3.3 Database .....	38
3.4 IDE .....	39
3.5 System Architecture .....	40
 <b>4: DESIGN AND PLANNING .....</b>	 <b>42</b>
4.1 SDLC .....	43

4.2 DFD .....	46
<b>5: PROGRAM CODE AND RESULTS .....</b>	<b>50</b>
<b>6: TESTING .....</b>	<b>73</b>
6.1 Testing .....	74
<b>7: DEVELOPMENT CHALLENGES .....</b>	<b>77</b>
<b>8: LIMITATIONS OF THE PROJECT: .....</b>	<b>79</b>
<b>9: FUTURE ENHANCEMENT .....</b>	<b>82</b>
<b>10: CONCLUSION .....</b>	<b>84</b>
<b>11: BIBLIOGRAPHY .....</b>	<b>86</b>



# **INTRODUCTION**

## 1.1 INTRODUCTION

The iSchool stands as a transformative force in modern education, driven by its mission to streamline operations, foster transparency, and empower stakeholders. By providing a comprehensive platform for managing various aspects of school administration, iSchool simplifies processes such as admissions, attendance tracking, and grade management. iSchool is an Student Management System. Through its user-friendly interface and seamless integration of frontend and backend technologies, iSchool ensures efficient data management and secure access control. It promotes transparency by offering centralized communication channels and access to real-time information for students, parents, teachers, and administrators. This transparency fosters trust and informed decision-making within the educational community, enhancing collaboration and engagement.

Furthermore, iSchool serves as a catalyst for innovation in education, continuously evolving to meet the dynamic needs of educational institutions. With future enhancements such as mobile application development, advanced reporting and analytics, and integration with e-learning features, iSchool remains at the forefront of innovation. By driving efficiency, effectiveness, and empowerment in education, iSchool reshapes the educational landscape, equipping stakeholders with the tools they need to thrive in the digital age.

For iSchool, Node.js with Express.js serves as the robust backend framework, offering several strengths. Rapid development is facilitated by Node.js's event-driven, non-blocking I/O model, allowing for quick and efficient development of a clean and maintainable codebase. With the addition of Express.js, developers can easily create robust APIs and handle routing, middleware, and request/response handling, further enhancing development speed. Additionally, Node.js with Express.js seamlessly integrates with MongoDB, allowing for efficient data management and storage.

On the frontend, iSchool benefits from Bootstrap, CSS, HBS, and AJAX, creating an interactive and dynamic user experience. Bootstrap's grid system and components enable responsive design, ensuring compatibility across various devices. The consistency provided by Bootstrap ensures a standardized set of UI components, maintaining a cohesive design throughout the application. Furthermore, AJAX allows for asynchronous communication between the frontend and backend, enhancing user interaction by enabling real-time updates without requiring full page reloads. The extensive community support for Bootstrap ensures access to documentation, tutorials, and third-party resources, facilitating frontend development.

By combining Node.js with Express.js for the backend, MongoDB for data storage, Bootstrap for the frontend, and AJAX for dynamic communication, iSchool maximizes the strengths of each technology. Node.js with Express.js efficiently handles routing, middleware, and data interactions, while MongoDB provides flexible data storage. Bootstrap and AJAX contribute to a seamless and interactive user experience, enhancing engagement and usability across devices.

## 1.2 OBJECTIVES

iSchool is a dynamic and innovative solution designed to revolutionize operations within educational institutions. It serves as a centralized platform, facilitating seamless communication, efficient data management, and enhanced collaboration among students, teachers, parents, and administrators. With its user-friendly interface and comprehensive features, iSchool streamlines administrative tasks, ensures data security, and promotes transparency and accountability. Additionally, iSchool is committed to driving innovation in education by continually evolving to meet the changing needs of educational institutions, ultimately empowering stakeholders and enhancing the overall educational experience.

### OBJECTIVES OF ISCHOOL:

**Streamline School Operations:** iSchool aims to streamline various administrative tasks and operations within educational institutions. This includes simplifying processes such as admissions management, attendance tracking, grade storage, event scheduling, and notice dissemination. By centralizing these functions onto a single platform, iSchool reduces manual effort, minimizes errors, and improves overall efficiency in school management.

**Enhance Communication and Collaboration:** iSchool seeks to enhance communication and collaboration among students, teachers, parents, and administrators. It provides a platform for seamless interaction, enabling stakeholders to communicate effectively, share information, and collaborate on academic and administrative matters. Through features such as messaging, announcements, and discussion forums, iSchool fosters a sense of community and encourages engagement within the educational ecosystem.

**Ensure Data Security and Integrity:** One of the key objectives of iSchool is to ensure the security and integrity of sensitive data. iSchool implements robust authentication mechanisms, encryption techniques, and access controls to safeguard student records, attendance information, grades, and other

confidential data from unauthorized access, tampering, or data breaches. By prioritizing data security, iSchool instills trust and confidence among users regarding the confidentiality and privacy of their information.

**Promote Transparency and Accountability:** iSchool promotes transparency and accountability within educational institutions by providing stakeholders with access to real-time information and updates. Students, parents, teachers, and administrators can view relevant data and insights, such as academic progress, attendance records, and school announcements, fostering transparency in decision-making processes. Additionally, iSchool encourages accountability by enabling stakeholders to track and monitor their responsibilities and commitments, thereby promoting a culture of accountability within the educational community.

**Drive Innovation and Adaptability:** iSchool is committed to driving innovation in education by continually evolving to meet the changing needs of educational institutions. It incorporates future enhancements such as integration with online payment gateways, mobile application development, advanced reporting and analytics, and e-learning features. By embracing technological advancements and adapting to emerging trends in education, iSchool remains at the forefront of innovation, ensuring that educational institutions are equipped with the tools and resources needed to thrive in the digital age.

### 1.3 IDENTIFICATION OF THE PROBLEM

Educational institutions face numerous challenges in efficiently managing their operations and engaging stakeholders. Here's an in-depth look at the problems that iSchool aims to address:

- **Fragmented Communication Channels:** Educational institutions often rely on disparate communication channels, leading to inefficiencies and miscommunication among students, teachers, parents, and administrators. This fragmentation hampers collaboration and coordination, hindering the effective management of school activities and events.
- **Manual Administrative Processes:** Traditional school management systems involve manual administrative tasks, such as admissions, attendance tracking, and grade management, which are time-consuming and prone to errors. These manual processes limit the efficiency of school operations and increase the burden on administrative staff.

- **Limited Accessibility Outside School Hours:** Students, parents, and teachers may face difficulties accessing important information or communicating with school authorities outside of regular school hours due to limited accessibility of traditional systems. This limitation impedes timely access to critical information and support services.
- **Data Security Risks:** The increasing digitization of educational records raises concerns about data security and the protection of sensitive student information from unauthorized access or breaches. Educational institutions need robust systems in place to ensure the security and integrity of student data and maintain compliance with data protection regulations.
- **Inefficient Resource Allocation:** Without real-time insights into resource allocation and utilization, educational institutions may struggle with inefficient use of faculty, facilities, and budgetary allocations. This inefficiency can lead to suboptimal academic outcomes and financial sustainability challenges.
- **Lack of Scalability:** Traditional school management systems may lack scalability to accommodate the diverse needs of educational institutions, hindering their ability to adapt to changes in student enrollment, curriculum requirements, or administrative processes. This lack of scalability limits the potential for growth and innovation within educational institutions.
- **Complex Event Management:** Organizing school events such as functions, workshops, or seminars involves complex coordination and communication among various stakeholders, often leading to logistical challenges and coordination issues. A streamlined event management system is needed to simplify the planning and execution of school events.
- **Limited Parent-Teacher Engagement:** Traditional methods of parent-teacher communication may be limited, inhibiting effective collaboration and engagement between parents and teachers in monitoring student progress and addressing concerns. Enhancing parent-teacher engagement is crucial for supporting student learning and development.
- **Absence of Personalization:** Generic approaches to school management may neglect the need for personalized experiences for students, parents, and teachers, hindering engagement and satisfaction with the educational process. Personalization is essential for addressing the diverse needs and preferences of stakeholders within educational institutions.
- **Lack of Real-Time Insights:** Without access to real-time data and analytics, educational institutions may struggle to make informed decisions, track performance metrics, and identify areas for improvement in school operations and student outcomes. Real-time insights are essential for driving continuous improvement and enhancing the effectiveness of educational programs and services.

In summary, iSchool aims to address these challenges by providing a comprehensive and user-friendly platform for efficient school management, effective communication, and enhanced stakeholder engagement. By streamlining administrative processes, ensuring data security, and leveraging technology to facilitate collaboration and decision-making, iSchool empowers educational institutions to achieve their goals and deliver quality education to students.

## **1.4 AIM**

The primary aim of the iSchool project is to develop an innovative and comprehensive School Management System (SMS) that addresses the diverse and evolving needs of educational institutions in the digital age. iSchool seeks to revolutionize traditional school management practices by providing a robust platform that streamlines administrative processes, enhances communication and collaboration among stakeholders, and improves the overall efficiency and effectiveness of school operations.

Through the iSchool platform, our aim is to empower educational institutions to effectively manage key aspects of their operations, including admissions, attendance tracking, grade management, event scheduling, and communication with students, parents, teachers, and administrators. By offering a user-friendly interface and a wide range of features tailored to the specific needs of educational institutions, iSchool aims to simplify complex administrative tasks, reduce manual effort, and minimize errors.

Furthermore, iSchool aims to foster transparency and accountability within educational institutions by providing real-time access to important information and insights. By promoting data-driven decision-making and facilitating communication among stakeholders, iSchool aims to create a more collaborative and engaged educational community.

Ultimately, the overarching aim of the iSchool project is to contribute to the delivery of quality education by empowering educational institutions with the tools and resources they need to thrive in the digital age. Through continuous innovation and adaptation to emerging technologies and educational trends, iSchool aims to remain at the forefront of school management systems, driving positive change and enhancing the educational experience for students, teachers, parents, and administrators alike.

## **1.5 EXISTING SYSTEM**

The traditional approach to school management relies on manual processes and paper-based workflows to handle key administrative tasks and communication among stakeholders. In this system, educational

institutions face challenges such as inefficiencies in administrative processes, fragmented communication channels, limited accessibility outside of regular school hours, reliance on physical documentation, and concerns about data security. Despite these challenges, the traditional system has been the backbone of school management for years, but its limitations highlight the need for modern solutions that can streamline operations, enhance communication, and ensure data security.

- **Manual Administrative Tasks:** Educational institutions heavily rely on manual processes for key administrative tasks such as admissions, attendance tracking, and grade management. These tasks are often labor-intensive and time-consuming, requiring significant human effort for data entry, processing, and record-keeping. Moreover, manual processes are inherently prone to errors, leading to inaccuracies in student records and administrative documents. The lack of automation in these processes contributes to inefficiencies in school operations, impacting the overall productivity and effectiveness of administrative staff.
- **Paper-based Documentation:** Traditional school management systems are characterized by a heavy reliance on paper-based workflows for documentation and record-keeping. Educational institutions generate a significant volume of paperwork, including admission forms, attendance registers, grade sheets, and event notices. Managing and storing these physical documents pose significant challenges, with issues such as document overload, storage constraints, and difficulties in retrieval and organization. Furthermore, paper-based documentation is vulnerable to loss, damage, or unauthorized access, raising concerns about data security and compliance with privacy regulations.
- **Disparate Communication Channels:** Communication among stakeholders in educational institutions is often fragmented, relying on disparate channels such as notices, memos, and phone calls. This fragmentation leads to inefficiencies and coordination issues, as stakeholders may struggle to access timely and relevant information. The lack of a centralized communication platform hampers effective collaboration and decision-making processes, hindering the overall efficiency and effectiveness of school operations. Moreover, miscommunication and misunderstandings may arise due to the use of multiple communication channels, leading to potential conflicts or delays in addressing important matters.
- **Limited Accessibility Outside School Hours:** Access to important information or services is often limited outside regular school hours, posing challenges for students, parents, and teachers. Educational institutions may lack mechanisms for providing access to resources or support outside of designated times, leading to inconvenience and frustration for stakeholders. Students may encounter difficulties in accessing educational materials or assignments, parents may struggle to

communicate with school authorities, and teachers may face challenges in accessing administrative systems or support services remotely. The limited accessibility outside school hours hampers convenience and accessibility for stakeholders, impacting their overall experience with the educational institution.

- **Data Security Risks:** The increasing digitization of educational records raises concerns about data security and privacy. While physical documents are susceptible to loss, damage, or unauthorized access, digital data faces its own set of security risks. Educational institutions must implement proper safeguards to protect sensitive student information from data breaches, unauthorized access, or malicious attacks. However, the lack of robust security measures and compliance frameworks poses risks to the confidentiality and integrity of student records. Data security breaches can have serious consequences, including reputational damage, legal liabilities, and financial losses for educational institutions. Thus, ensuring data security and compliance with privacy regulations is paramount in the management of educational records and administrative processes.

## 1.6 PROPOSED SYSTEM

In response to the shortcomings of traditional school management systems, iSchool proposes a comprehensive solution aimed at modernizing administrative processes, enhancing communication, ensuring data security, and facilitating efficient student and teacher management. With integrated features for announcement management, attendance tracking, and grade management, iSchool is designed to streamline operations and improve the overall efficiency of educational institutions.

- **Streamlined Student and Teacher Management:** iSchool introduces a centralized platform for student and teacher management, including profile creation, enrollment, and course assignment. Automated workflows for student admission processes facilitate seamless enrollment and registration procedures, while efficient management of teacher assignments, scheduling, and performance evaluations ensures optimal resource allocation and accountability.
- **Announcement Management:** iSchool implements a centralized announcement management system for disseminating important information to students, teachers, parents, and administrators. Customizable announcement templates and scheduling features enable timely and relevant communication, with real-time notifications and alerts keeping stakeholders informed about school events, deadlines, and updates.



- **Attendance Tracking:** iSchool digitizes attendance tracking processes through biometric, barcode, or RFID technology for accurate and efficient monitoring. Real-time attendance reporting and analytics help identify trends, patterns, and areas for improvement, with automated notifications alerting teachers and administrators of irregular attendance or absences.
- **Grade Management:** iSchool provides secure storage and management of student grades, assessments, and academic records. Customizable grading systems and rubrics accommodate different curriculum requirements and assessment criteria, with integration with learning management systems (LMS) enabling seamless transfer of grade data and student performance tracking.

The proposed system for iSchool offers a comprehensive solution for modernizing school management practices and enhancing the educational experience for students, teachers, parents, and administrators. By integrating features for student and teacher management, announcement management, attendance tracking, and grade management, iSchool aims to streamline administrative processes, improve communication, ensure data security, and promote efficiency in educational institutions. Through continuous innovation and adaptation, iSchool remains committed to driving positive change and delivering a transformative educational experience for all stakeholders.

## 1.7 FEASIBILITY STUDY

- **Technical Feasibility:**
  - Software and Technology:
    - Assessment: Evaluate the availability and suitability of software tools and technologies for developing the web application.
    - Outcome: Confirm that appropriate technologies exist to support the envisioned features and functionalities.
  - Development Expertise:
    - Assessment: Examine the availability of skilled developers with expertise in web development, databases, and relevant technologies.
    - Outcome: Ensure that the development team possesses the necessary skills to successfully implement the proposed system.
  - Integration Capabilities:
    - Assessment: Investigate the feasibility of integrating the proposed system with existing technologies or third-party services.

- Outcome: Determine the compatibility of the system with other platforms or applications.

- **Economic Feasibility:**

- Cost-Benefit Analysis:
  - Assessment: Conduct a cost-benefit analysis to evaluate the financial feasibility of the project, considering development costs, hardware/software expenses, and potential savings.
  - Outcome: Determine if the benefits outweigh the costs and if the project is financially viable.
- Return on Investment (ROI):
  - Assessment: Calculate the expected ROI by estimating the projected benefits such as cost savings and enhanced efficiency.
  - Outcome: Determine if the investment in the iSchool project is justified based on the potential returns.

- **Operational Feasibility:**

- User Acceptance:
  - Assessment: Assess the willingness of stakeholders to adopt and use the iSchool platform, considering factors such as ease of use and training requirements.
  - Outcome: Ensure that stakeholders are onboard with the proposed system and willing to embrace the change.
- Impact on Operations:
  - Assessment: Evaluate the potential impact of implementing iSchool on existing operational processes and workflow efficiency.
  - Outcome: Determine if the system will enhance operational efficiency and productivity within educational institutions.
- Change Management:
  - Assessment: Develop strategies to manage organizational change and mitigate resistance to new technology adoption.
  - Outcome: Ensure a smooth transition to the iSchool platform through effective communication, training, and support mechanisms.

- **Scheduling Feasibility:**

- Project Timeline:
  - Assessment: Develop a detailed project schedule outlining key milestones, tasks, and deliverables.
  - Outcome: Ensure that the project is completed within the specified timeline and meets the desired objectives.
- Resource Availability:
  - Assessment: Assess the availability of human and technical resources required for the project.
  - Outcome: Allocate resources effectively to meet project deadlines and requirements.

Based on the findings of the feasibility study, recommendations will be provided regarding the viability of proceeding with the iSchool project. Factors such as technical feasibility, economic viability, operational considerations, and scheduling constraints will be carefully evaluated to determine the project's feasibility and potential for success.

# **SOFTWARE REQUIRMENT SPECIFICATION**

## 2.1 MINIMUM SOFTWARE REQUIREMENTS

### Operating System:

The choice of the operating system for development depends on the developers' preferences, familiarity, and the specific requirements of the iSchool project. Below are considerations for each operating system:

#### Windows:

- Pros:
  - Widely used and user-friendly.
  - Extensive software support.
- Cons:
  - Some web development tools may have better native support on Unix-based systems.
  - Certain features or libraries may behave differently compared to Unix-like systems.
  - Suitability: Windows is suitable for Node.js and Express.js development, and many developers use it effectively.

#### macOS:

- Pros:
  - UNIX-based, providing a similar environment to Linux for development.
  - Easier integration with macOS-specific development tools.
- Cons:
  - Limited hardware options compared to Windows.
  - Higher upfront cost for Apple hardware.
  - Suitability: macOS is suitable for Node.js and Express.js development, especially if you are working on macOS-related projects.

#### Linux:

- Pros:
  - Open-source and customizable.
  - Excellent support for development tools and libraries.
- Cons:
  - Hardware compatibility may vary.
  - May require more technical expertise for certain configurations.

- Suitability: Linux is highly suitable for Node.js and Express.js development, and many developers prefer it for its flexibility and robust development environment.

**Recommendation:**

Development Environment Consistency: It's a good practice to ensure that the development and production environments are as consistent as possible.

Personal Preference: Developers often have personal preferences for operating systems. Choose the one that you are most comfortable and productive with.

Cross-Platform Compatibility: Since Node.js and Express.js are cross-platform, projects developed on one operating system can generally be deployed on others without major issues. Ensure that your project dependencies are compatible with your chosen OS.

**Frontend:**

Handle Bars, CSS, Bootstrap, JavaScript

**Backend:**

Node.js, Express.js, MongoDB

**IDE:**

Visual Studio

## **SOFTWARE IMPLEMENTAION**

### **3.1 FRONTEND:**

#### **Handlebars (hbs):**

Handlebars, a powerful templating engine, was instrumental in the frontend development of iSchool. By allowing the creation of dynamic HBS templates, Handlebars simplified the process of rendering content dynamically, leveraging data from the backend. Its flexible syntax and support for partials and helpers facilitated the creation of reusable components, promoting a modular and maintainable frontend architecture. With Handlebars, iSchool achieved efficient content rendering and seamless integration of data-driven elements into its user interface.

#### **CSS:**

Cascading Style Sheets (CSS) played a vital role in defining the visual presentation of iSchool's frontend components. Through CSS, we meticulously crafted the styling of user interface elements, ensuring a visually appealing and consistent layout across the application. Leveraging CSS's extensive capabilities, including selectors, properties, and media queries, we achieved responsive design and cross-browser compatibility. By adhering to best practices and modern CSS techniques, iSchool delivered a polished and professional user interface that enhances user engagement and satisfaction.

#### **JavaScript:**

JavaScript served as the backbone of interactivity and dynamic behavior in iSchool's frontend components. Through client-side scripting, JavaScript enriched the user experience by enabling interactive features and dynamic content updates. Utilizing JavaScript libraries and frameworks such as jQuery and Axios, we implemented functionalities such as form validation, asynchronous data fetching, and DOM manipulation. JavaScript's event-driven nature and rich ecosystem of tools empowered iSchool to create an immersive and responsive user interface that fosters user engagement and productivity.

By harnessing the capabilities of Handlebars, CSS, and JavaScript, iSchool achieved a sophisticated and user-centric frontend that delivers seamless navigation, consistent styling, and interactive features, enhancing the overall user experience.



### **3.2 BACKEND:**

#### **Node.js:**

Node.js serves as the backbone of iSchool's backend architecture, offering a powerful runtime environment for executing JavaScript code on the server-side. One of the key advantages of Node.js over Python in web development is its event-driven, non-blocking I/O model, which enhances scalability and performance in handling concurrent requests. Unlike Python, which traditionally relies on synchronous execution, Node.js excels in handling I/O-bound operations, making it particularly well-suited for real-time applications and data-intensive tasks in iSchool. Additionally, Node.js benefits from a vibrant ecosystem of packages and modules through npm (Node Package Manager), providing a wealth of tools and libraries to streamline development and accelerate time-to-market for iSchool's backend features.

#### **Express.js:**

Express.js, a minimalist web application framework for Node.js, further enhances iSchool's backend development with its simplicity and flexibility. By providing a lightweight yet powerful framework for building APIs and web applications, Express.js empowers developers to rapidly prototype and iterate on backend functionalities for iSchool. With its robust routing system, middleware support, and seamless integration with Node.js, Express.js enables efficient request handling and resource management, ensuring optimal performance and scalability in iSchool's backend infrastructure.

### **3.3 DATABASE (MONGODB):**

#### **MongoDB:**

MongoDB plays a pivotal role in iSchool's backend infrastructure as the chosen database management system. Its document-oriented NoSQL architecture offers unparalleled flexibility and scalability, allowing iSchool to adapt to evolving data requirements with ease. MongoDB's dynamic schema and JSON-like document model facilitate seamless integration with Node.js and Express.js, enabling efficient data storage and retrieval in iSchool's backend applications. Moreover, MongoDB's distributed architecture, support for sharing and replication, and automatic failover mechanisms ensure high availability and fault tolerance, guaranteeing uninterrupted service for iSchool's users. By leveraging MongoDB, iSchool gains a competitive edge in data management, enabling agile development, and future scalability.

### 3.4 IDE (VISUAL STUDIO CODE):

#### Visual Studio Code (VSCode):

VSCode serves as the primary Integrated Development Environment (IDE) for iSchool's development team, providing a versatile and feature-rich environment for coding, debugging, and collaboration. With its lightweight and extensible architecture, VSCode offers a seamless development experience for building frontend and backend components of iSchool. Equipped with built-in support for Node.js development, VSCode streamlines tasks such as code navigation, syntax highlighting, and debugging, enhancing developer productivity and code quality in iSchool's development workflow. Additionally, VSCode's extensive marketplace of extensions allows developers to customize their IDE environment according to their preferences and project requirements, further optimizing the development experience for iSchool.

#### ADVANTAGES OF NODE.JS AND EXPRESS.JS IN WEB DEVELOPMENT:

- **Unified Language Stack:**
  - Enables full-stack JavaScript development, using the same language (JavaScript) for both frontend and backend.
  - Streamlines development process and enhances code maintainability at iSchool.
- **Efficient Asynchronous Architecture:**
  - Node.js's asynchronous, event-driven architecture combined with Express.js's lightweight framework ensures efficient handling of I/O-bound operations.
  - Improves performance and responsiveness of iSchool's web applications, particularly in real-time scenarios and data-intensive tasks.
- **Scalability and Performance:**
  - Node.js's non-blocking model and Express.js's minimalist framework facilitate horizontal scaling, allowing iSchool to accommodate growing user demands seamlessly.
  - Efficient resource utilization and low overhead contribute to improved application performance, ensuring smooth operation even under heavy loads at iSchool.

## IMPORTANCE OF MONGODB:

- **Flexible Data Model:** MongoDB's document-oriented NoSQL architecture allows iSchool to store data in flexible, JSON-like documents, accommodating evolving data structures and requirements with ease.
- **Scalability:** MongoDB's distributed architecture and support for sharding enable iSchool to scale horizontally to meet growing data demands, ensuring seamless performance and availability.
- **Developer Productivity:** MongoDB's intuitive query language and dynamic schema empower developers to iterate rapidly on iSchool's backend features, reducing development time and accelerating time-to-market.
- **High Availability:** MongoDB's built-in replication and automatic failover mechanisms ensure continuous availability and data durability, minimizing downtime and ensuring uninterrupted service for iSchool's users.

## 3.5 SYSTEM ARCHITECTURE

### Framework and MVC Architecture:

In the iSchool project, we employ the Model-View-Controller (MVC) architectural pattern, tailored to Node.js and Express.js frameworks. Here's a breakdown of each component:

#### Model (M):

In iSchool, models encapsulate the data structure and logic of the application. They define the entities and their relationships in the MongoDB database.

Models are represented as JavaScript classes, with each class corresponding to a collection in the MongoDB database.

Node.js's Mongoose ODM (Object Data Modeling) facilitates the translation between JavaScript code and the underlying database, enabling seamless interaction with the MongoDB database using JavaScript objects.

**View (V):**

Views in iSchool handle the presentation logic, processing user input, and rendering dynamic content to the client.

In the context of a Node.js and Express.js application, views are typically implemented using template engines such as Handlebars (hbs) or EJS (Embedded JavaScript).

Views retrieve data from the backend, process it, and render it dynamically using template engine syntax.

**Controller (C):**

Controllers in iSchool govern the application's logic, orchestrating the flow of data between models and views.

Express.js route handlers serve as controllers in iSchool, receiving incoming requests, processing data from models, and rendering views or returning JSON responses to the client.

Controllers ensure the separation of concerns and facilitate modularization of iSchool's backend logic.

**Database Schema, Relationships, and ORM:****Database Schema:**

iSchool utilizes MongoDB as the database management system, leveraging its flexible document-oriented architecture.

Mongoose schemas define the database schema in iSchool, with each schema representing a collection in the MongoDB database.

For example, a Student schema might include fields like 'name', 'age', and 'grade'.

**Relationships:**

While MongoDB is a NoSQL database and does not inherently support relational features, iSchool implements relationships between entities using references or embedding within documents.

For instance, a Student document may reference a Teacher document, establishing a one-to-many relationship.

### **ORM (Object-Relational Mapping):**

Node.js and Express.js applications like iSchool do not use traditional ORMs like those found in Django. Instead, Mongoose serves as an ODM (Object Data Modeling) tool, providing schema-based modeling for MongoDB.

Mongoose simplifies interactions with MongoDB, offering features such as schema validation, data type casting, and query building.

In summary, iSchool's architecture, built on Node.js, Express.js, and MongoDB, provides a robust foundation for developing scalable, performant, and maintainable web applications. The use of MongoDB as the backend database offers flexibility and scalability, while the MVC architecture and Mongoose ODM facilitate structured development and efficient data management.

## **DESIGN AND PLANNING**

## 4.1 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

### AGILE SOFTWARE DEVELOPMENT APPROACH

The Agile Software Development approach adopted for the iSchool project emphasizes adaptability, collaboration, and iterative development. Unlike traditional linear approaches, Agile promotes flexibility, allowing for continuous evolution based on changing requirements and stakeholder feedback. Here are the key characteristics of the Agile approach tailored for iSchool:

#### KEY CHARACTERISTICS:

- **Iterative and Incremental:**
  - Development is carried out in small, iterative cycles called sprints, with each sprint delivering incremental functionality to the iSchool system.
  - The iterative nature of Agile allows for continuous refinement and adaptation based on evolving needs and priorities.
- **Collaboration and Customer Involvement:**
  - Close collaboration between the development team and stakeholders, including students, teachers, and administrators, is essential throughout the project.
  - Regular feedback from end-users is incorporated into each sprint, ensuring that the iSchool system aligns with user expectations and requirements.
- **Adaptive Planning:**
  - Agile embraces changing requirements, allowing for adjustments and reprioritization of tasks at the beginning of each sprint.
  - The focus is on delivering high-value features quickly, maximizing the iSchool system's responsiveness to user needs.
- **Continuous Integration and Testing:**
  - Continuous integration and automated testing practices are integral to Agile, ensuring that new features are thoroughly tested and integrated into the iSchool system with minimal disruption.
  - Test-driven development (TDD) and continuous deployment practices help maintain the stability and reliability of the iSchool application.

**AGILE METHODOLOGY PHASES:**

- **Sprint Planning:**

- Define the scope and objectives for the upcoming sprint, based on stakeholder priorities and feedback.
- Break down user stories and tasks, estimating the effort required for implementation.

- **Sprint Execution:**

- Development, testing, and integration activities are carried out during the sprint.
- Daily stand-up meetings facilitate communication and collaboration among team members, ensuring progress towards sprint goals.

- **Sprint Review and Retrospective:**

- At the end of each sprint, a review session is held to demonstrate the completed functionality to stakeholders and gather feedback.
- Retrospective meetings allow the team to reflect on the sprint's successes and challenges, identifying areas for improvement.

**ADVANTAGES:**

- **Adaptability and Flexibility:**

Agile enables rapid response to changing requirements and priorities, ensuring that the iSchool system remains aligned with user needs.

- **Enhanced Collaboration:**

Close collaboration between developers and stakeholders fosters transparency, trust, and shared ownership of the iSchool project's success.

- **Early and Continuous Delivery:**

Incremental delivery of functionality ensures that valuable features are delivered early and frequently, providing stakeholders with tangible results.



**DISADVANTAGES:**

- **Resource Intensive:**

Agile requires active involvement and commitment from stakeholders, which can be resource-intensive for the iSchool project team.

- **Initial Learning Curve:**

Adopting Agile practices may require initial training and adjustment for team members accustomed to traditional development methodologies.

- **Scope Creep:**

The iterative nature of Agile may lead to scope creep if not managed effectively, potentially impacting project timelines and budgets.

**CONCLUSION:**

The Agile Software Development approach offers a flexible and collaborative framework for developing the iSchool system, allowing for continuous adaptation and delivery of high-quality software that meets the evolving needs of stakeholders. By embracing Agile principles and practices, the iSchool project aims to deliver a user-centric and responsive educational management platform.

**4.2 DATA FLOW DIAGRAM (DFD)**

A Data Flow Diagram (DFD) is a visual representation that illustrates the flow of data within a system. In the context of the iSchool project, a DFD will provide insight into how data moves between various components of the system, including processes, data stores, and external entities. Here are the key components of the DFD for the iSchool system:

**1. Processes:**

Processes represent the activities or transformations performed on data within the iSchool system. These may include functions such as user authentication, attendance tracking, grade management, and event scheduling.

**2. Data Flow:**

Data flows depict the movement of data between different components of the iSchool system. This includes the transfer of information between processes, data stores, external entities, and data destinations. Examples of data flow in iSchool may include student information, attendance records, grade updates, and event notifications.

**3. Data Store:**

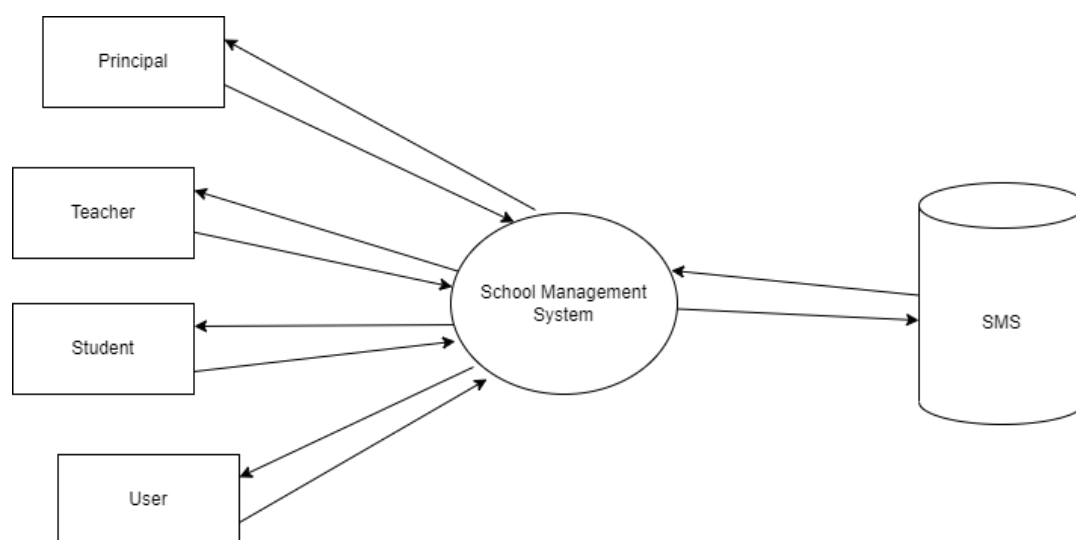
Data stores in the iSchool system represent where data is stored within the application. This could include databases, files, or other storage mediums used to store student records, attendance logs, academic grades, and other pertinent information.

**4. External Entity:**

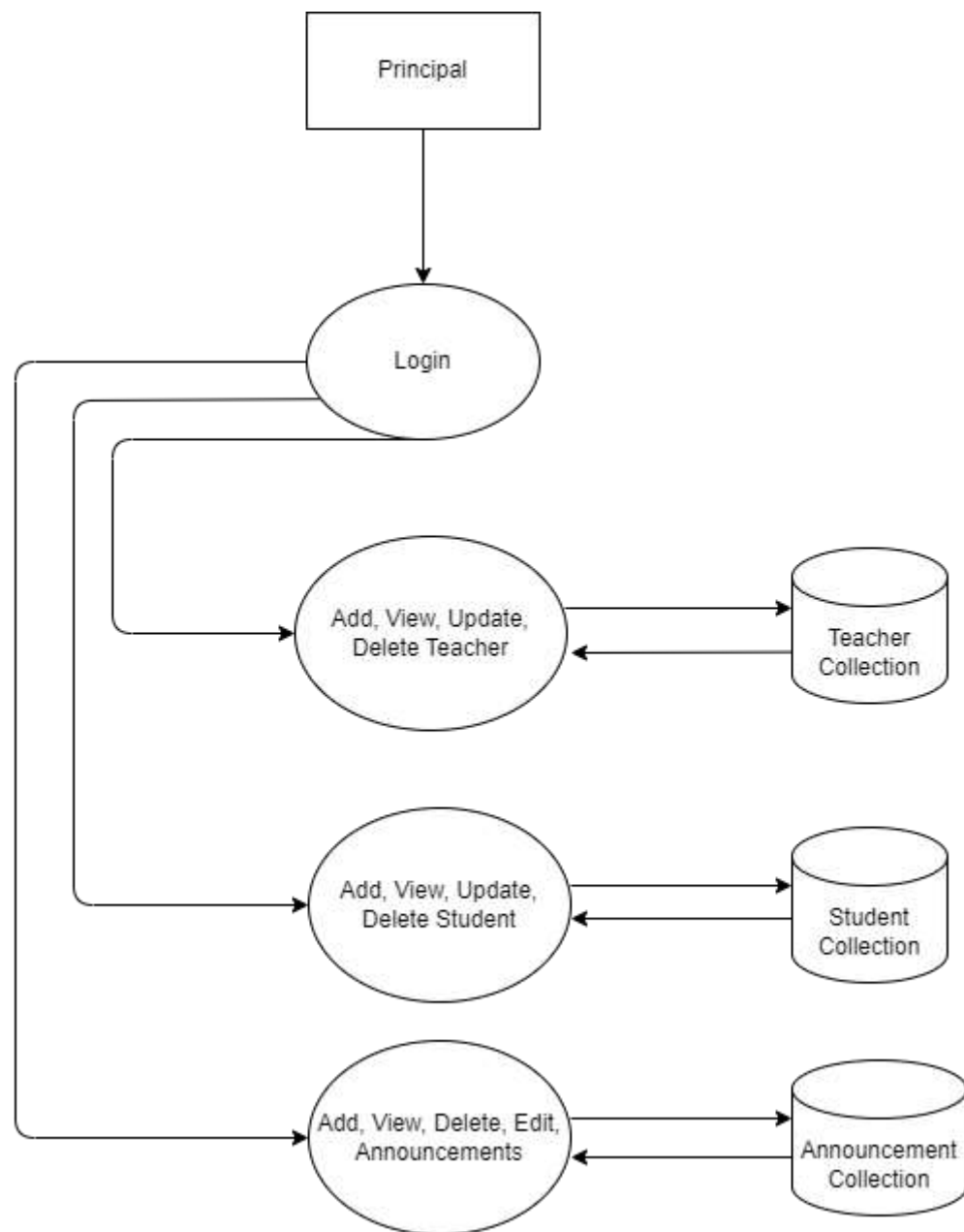
External entities in the iSchool system represent sources or destinations of data that interact with the system but exist outside of it. This could include students, teachers, administrators, and other external systems or devices that interact with iSchool.

The DFD for iSchool will be hierarchical and can be organized into multiple levels, providing both a high-level overview of the entire system and more detailed views of specific processes and interactions. This diagram will aid in understanding the functional requirements of the iSchool system, identifying data flow and dependencies, and providing a clear overview of how data is processed within the application.

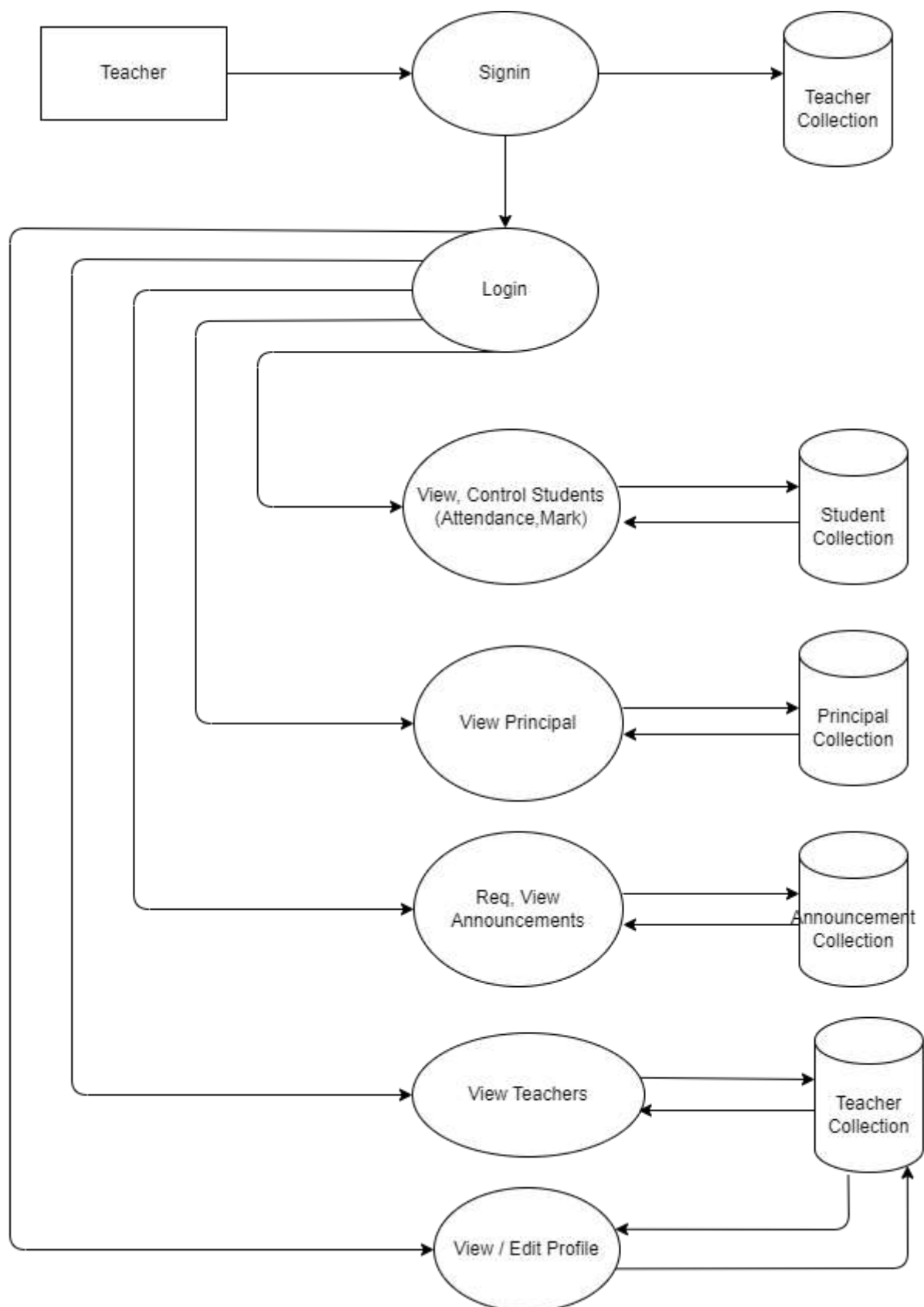
DFDs are essential during the system analysis and design phases of the software development life cycle for communicating and documenting the architecture and functionality of the iSchool system, ensuring clarity and alignment among stakeholders.

**Level 0**

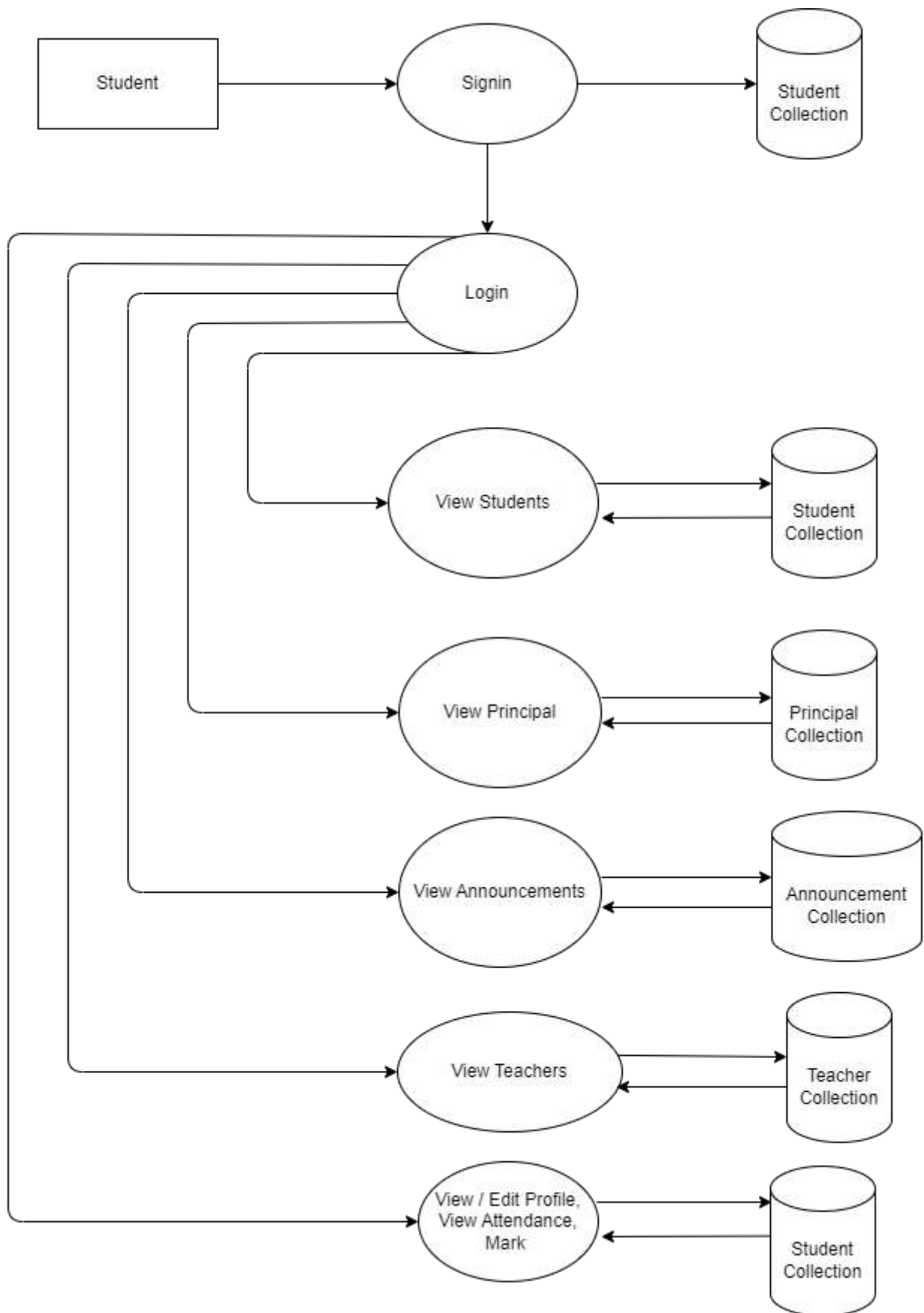
## Level 1.1

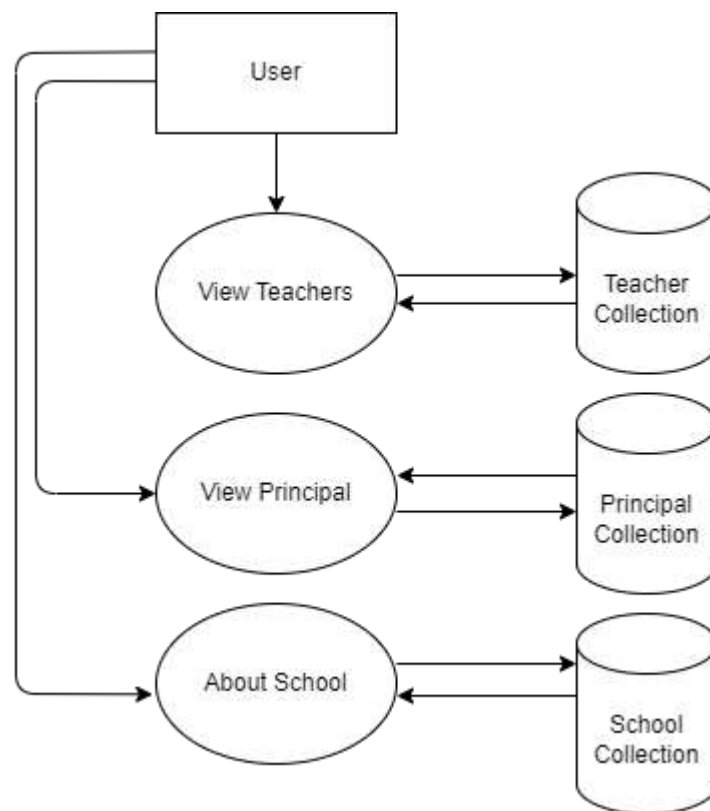


## Level 1.2



## Level 1.3



**Level 1.4**

## 4.3 COLLECTION STRUCTURE

### 1. Principal

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
name	String	Name of the individual
age	String	Age of the individual
sex	String	Gender of the individual
place	String	Place of residence
state	String	State of residence
mobile	String	Mobile number
email	String	Email address

### 2. Teacher

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
firstName	String	First name of the individual
secondName	String	Second name of the individual
age	String	Age of the individual
sex	String	Gender of the individual
qualification	String	Qualification of the individual
place	String	Place of residence
district	String	District of residence
state	String	State of residence
subject	String	Subject taught by the individual
class	String	Class/Grade of the individual
mobile	String	Mobile number
email	String	Email address
password	String	Encrypted password



### 3. Student

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
firstName	String	First name of the student
secondName	String	Last name of the student
age	String	Age of the student
sex	String	Gender of the student
class	String	Class/Grade of the student
place	String	Place of residence
state	String	State of residence
mobile	String	Mobile number of the student
email	String	Email address of the student
password	String	Encrypted password for authentication
admissionNo	Integer	Admission number of the student
rollNumber	Integer	Roll number of the student
attendance	Object	Object containing attendance details
mark	Object	Object containing marks details

### 4. User

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
email	String	Email address of the user
name	String	Name of the user
password	String	Encrypted password for authentication

## 5. Announcements

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
title	String	Title of the announcement
date	String	Date when the announcement was created
description	String	Description or content of the announcement
approval	Boolean	Indicates whether the announcement is approved or not

## 6. Subjects

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
subject	String	Name of the subject
mark	String	Marks obtained in the subject

## 7. Announcement Request

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
title	String	Title of the announcement
date	String	Date when the announcement was created
description	String	Description or content of the announcement
approval	Boolean	Approval status of the announcement

## 8. Admission Request

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
firstName	String	First name of the user
secondName	String	Last name or surname of the user
age	String	Age of the user
sex	String	Gender of the user
desiredClass	String	Standard or class the student wants admission to
place	String	Place of residence of the user
district	String	District of residence of the user
state	String	State of residence of the user
mobile	String	Mobile number of the user
email	String	Email address of the user
password	String	Password of the user (hashed for security)

## 9. Teacher Request

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for the document
firstName	String	First name of the teacher
secondName	String	Second name or last name of the teacher
age	String	Age of the teacher
sex	String	Gender of the teacher
qualification	String	Qualification of the teacher
place	String	Place of residence of the teacher
district	String	District of residence of the teacher
state	String	State of residence of the teacher
subject	String	Subject taught by the teacher
class	String	Class or standard assigned to the teacher
mobile	String	Mobile number of the teacher
email	String	Email address of the teacher
password	String	Encrypted password of the teacher

## **PROGRAM CODE AND RESULT**

## PRINCIPAL MODULE

Principal authentication and dashboard redirection.

```
const verifyLoginPrincipal = (req, res, next) => {
  if (req.session.principal && req.session.loggedIn) {
    next();
  } else {
    res.redirect('/principal/principal-login');
  }
}

router.get('/', verifyLoginPrincipal, function(req, res, next) {
  let princi = req.session.principal
  res.render('principal/home', {principal:true, princi });
});

router.get('/principal-login', (req, res) => {
  res.render('principal/login', { login: true });
});

router.post('/principal-login', (req, res) => {
  let response = req.body
  let princi = {
    name: 'Rashin KP',
    age: 40,
    email: 'rashinkp001@gmail.com',
    password: 123
  }
  if (response.email === princi.email && response.password === princi.password) {
    req.session.loggedIn = true;
    req.session.principal = princi;
    res.redirect('/principal/');
  } else {
    console.log("Wrong information");
    res.redirect("/principal/principal-login")
  }
})
```

## Announcement

```

module.exports = {
  addAnnouncement: (announcementData) => {
    return new Promise(async (resolve, reject) => {
      try {
        const announcement = {
          title: announcementData.title,
          date: announcementData.date,
          description: announcementData.description,
        };

        const announcementCollection = await
db.get().collection(COLLECTION.ANNOUNCEMENT_COLLECTION);

        const result = await announcementCollection.insertOne(announcement);
        resolve(result.insertedId);
      } catch (error) {
        reject(error);
      }
    });
  },
  getAllAnnouncements: () => {
    return new Promise(async (resolve, reject) => {
      try {
        const announcementCollection = await
db.get().collection(COLLECTION.ANNOUNCEMENT_COLLECTION);

        // Sort the announcements based on the 'date' property in descending order
        const announcements = await announcementCollection.find().sort({ date: -1 }).toArray();

        resolve(announcements);
      } catch (error) {
        reject(error);
      }
    });
  },
  removeAnnouncement: (announcementId) => {
    return new Promise(async (resolve, reject) => {
      try {

```

```

        const announcementCollection = await
db.get().collection(COLLECTION.ANNOUNCEMENT_COLLECTION);

        await announcementCollection.deleteOne({ _id:new ObjectId(announcementId) });

        resolve();
    } catch (error) {
        reject(error);
    }
});
},
};

```

## Requests

```

module.exports = {
  insertRequestAnnouncement: (requestDetails) => {
    return new Promise(async (resolve, reject) => {
      try {
        const reqAnnouncementCollection = await
db.get().collection(COLLECTION.REQ_ANNOUNCEMENT_COLLECTION);

        const result = await reqAnnouncementCollection.insertOne(requestDetails);
        resolve(result);
      } catch (error) {
        reject(error);
      }
    });
  },
  moveRequestToAnnouncement: async (requestId) => {
    try {
      const reqAnnouncementCollection = await
db.get().collection(COLLECTION.REQ_ANNOUNCEMENT_COLLECTION);

      const announcementCollection = await
db.get().collection(COLLECTION.ANNOUNCEMENT_COLLECTION);

      // Find the request in the req_announcement collection
      const request = await reqAnnouncementCollection.findOne({ _id: ObjectId(requestId) });
    }
  }
}

```



```

    // Check if the request exists and if the approval is true
    if (request && request.approval) {
        // Insert the request into the announcement collection
        const result = await announcementCollection.insertOne(request);

        // Remove the request from the req_announcement collection
        await reqAnnouncementCollection.deleteOne({ _id: ObjectId(requestId) });

        return result;
    } else {
        // Return null if the request doesn't exist or approval is false
        return null;
    }
} catch (error) {
    throw error;
},
getAllRequestsSortedByDate: async () => {
    try {
        const reqAnnouncementCollection = await
db.get().collection(COLLECTION.REQ_ANNOUNCEMENT_COLLECTION);

        // Fetch requests and sort them by 'date' field in descending order
        const requests = await reqAnnouncementCollection.find().sort({ date: -1 }).toArray();

        return requests;
    } catch (error) {
        throw error;
    }
},
removeRequest: async (requestId) => {
    try {
        const reqAnnouncementCollection = await
db.get().collection(COLLECTION.REQ_ANNOUNCEMENT_COLLECTION);

        // Remove the request from the req_announcement collection
        await reqAnnouncementCollection.deleteOne({ _id:new ObjectId(requestId) });
    } catch (error) {

```

```

        throw error;
    }
},
approveRequest: async (requestId) => {
    try {
        const reqAnnouncementCollection = await
db.get().collection(COLLECTION.REQ_ANNOUNCEMENT_COLLECTION);
        const announcementCollection = await
db.get().collection(COLLECTION.ANNOUNCEMENT_COLLECTION);
        const request = await reqAnnouncementCollection.findOne({ _id:new ObjectId(requestId) });
        if (request) {
            await announcementCollection.insertOne(request);
            await reqAnnouncementCollection.deleteOne({ _id:new ObjectId(requestId) });
        } else {
            return null;
        }
    } catch (error) {
        throw error;
    }
},

```

## Add Student

```

addStudent: async (student, callback) => {
    try {
        // Generate a unique admission number
        const lastAdmissionNumber = await db
            .get()
            .collection(COLLECTION.STUDENTS_COLLECTION)
            .find()
            .sort({ admissionNo: -1 })
            .limit(1)
            .toArray();

        const lastAdmission = lastAdmissionNumber[0];
        const lastAdmissionNo = lastAdmission ? lastAdmission.admissionNo : 999; // Default to 999 if NaN

        const newAdmissionNumber = isNaN(lastAdmissionNo)

```

```

    ? 1000
    : lastAdmissionNo + 1;

student.admissionNo = newAdmissionNumber;

// Generate a unique roll number based on the student's class
const lastRollNumber = await db
  .get()
  .collection(COLLECTION.STUDENTS_COLLECTION)
  .find({ class: student.class })
  .sort({ rollNumber: -1 })
  .limit(1)
  .toArray();

const lastRoll = lastRollNumber[0];
const newRollNumber =
  lastRoll && lastRoll.rollNumber !== undefined
    ? lastRoll.rollNumber + 1
    : 1;

// Assign the new roll number
student.rollNumber = newRollNumber;

student.attendance = {
  present: 0,
  percentage: 0,
};

// Hash the password
student.password = await bcrypt.hash(student.password, 10);

// Insert the student with the assigned admission number and roll number
const result = await db
  .get()
  .collection(COLLECTION.STUDENTS_COLLECTION)
  .insertOne(student);

// Pass the admission number and roll number in the callback

```

```

    callback(result.insertedId, newAdmissionNumber, newRollNumber);
  } catch (error) {
    console.error("Error in addStudent:", error);
    throw error;
  }
},

```

## STUDENT MODULE:

Student authentication and student dashboard redirections.

```

const verifyLoginStudent = (req, res, next) => {
  if (req.session.student && req.session.loggedIn) {
    next();
  } else {
    res.redirect('/student/login');
  }
}

router.get('/', verifyLoginStudent, function(req, res) {
  student_data = req.session.student;
  res.render('student/home', {student_check: true, student_data});
});

router.get('/login', (req, res) => {
  res.render('student/login', {login: true});
})

router.post('/login', async function (req, res) {
  const { email, password } = req.body;
  try {
    const student = await studentHelpers.getStudentByEmail(email);
    if (student) {
      const passwordMatch = await studentHelpers.comparePasswords(password, student.password);

      if (passwordMatch) {
        req.session.loggedIn = true;
        req.session.student = student;
        res.redirect('/student/');
      } else {

```

```

    res.render('student/login', {
      student: true,
      loginError: 'Invalid email or password',
    });
  }
} else {
  res.render('student/login', {
    teacher: true,
    loginError: 'Invalid email or password',
  });
}
} catch (error) {
  console.error('Error in teacher login:', error);
  res.render('student/login', {
    student: true,
    loginError: 'An error occurred. Please try again.',
  });
}
});

router.get('/profile', verifyLoginStudent, (req, res) => {
  let student_data = req.session.student;
  res.render('student/profile', { student_data, student_check: true });
})

router.get('/logout', verifyLoginStudent, (req, res) => {
  req.session.student = null;
  res.redirect('/student/');
})

router.get('/list-teachers', verifyLoginStudent, (req, res) => {

  teacherHelpers.getAllTeachers()
  .then(teachers => {
    teachers.forEach((teacher, index) => {
      teacher.counter = index + 1;
    });
  });
});

```

```

if(teachers){

  let student_data = req.session.student
  res.render("teacher/list-teachers",{teachers, student_check:true,student_data});

}
else{
  return res.send('no teacher found')
}
})
})

router.get('/list-students',verifyLoginStudent,(req,res)=>{
  studentHelpers.getAllStudents().then((student) => {
    student.forEach((student, index) => {
      student.counter = index + 1;
    });
    let student_data = req.session.student
    res.render('student/list-students', { student_check: true, student_data,student});
  });
})

router.get('/announcements',verifyLoginStudent,async(req,res)=>{
  try {
    const announcements = await announcementHelpers.getAllAnnouncements();
    let student_data = req.session.student
    res.render('teacher/announcements', { student_check: true, announcements,student_data});
  } catch (error) {
    console.error('Error in /announcements route:', error);
    res.status(500).send('Internal Server Error');
  }
})

```

## Sorting Students

```

getStudentByEmail: (email) => {
  return new Promise(async (resolve, reject) => {

```

```

try {
  const student = await db
    .get()
    .collection(COLLECTION.STUDENTS_COLLECTION)
    .findOne({ email: email });
  resolve(student);
} catch (error) {
  console.error("Error in getTeacherByEmail:", error);
  reject(error);
}
});
},

```

## TEACHER MODULE:

Teacher authentication and dashboard redirection.

```

const verifyLoginTeacher = (req, res, next) => {
  if (req.session.teacher && req.session.loggedIn) {
    next();
  } else {
    res.redirect("/teacher/login");
  }
};

router.get("/", verifyLoginTeacher, function (req, res) {
  let staff = req.session.teacher;
  res.render("teacher/home", { teacher: true, staff });
});

router.get("/login", function (req, res) {

  res.render("teacher/login", { login: true });
});

router.post("/login", async function (req, res) {
  const { email, password } = req.body;

```

```

try {
  const teacher = await teacherHelpers.getTeacherByEmail(email);
  if (teacher) {
    const passwordMatch = await teacherHelpers.comparePasswords(
      password,
      teacher.password
    );

    if (passwordMatch) {
      req.session.loggedIn = true;
      req.session.teacher = teacher;
      res.redirect("/teacher/"); // You can redirect to the teacher's dashboard or any other page
    } else {
      res.render("teacher/login", {
        teacher: true,
        loginError: "Invalid email or password",
      });
      console.log("password didn't match....")
    }
  } else {
    // Teacher not found, show an error message
    res.render("teacher/login", {
      teacher: true,
      loginError: "Invalid email or password",
    });
    console.log('email or password error...')
  }
} catch (error) {
  console.error("Error in teacher login:", error);
  res.render("teacher/login", {
    teacher: true,
    loginError: "An error occurred. Please try again.",
  });
}
});

router.get("/logout", verifyLoginTeacher, (req, res) => {
  req.session.teacher = null;

```



```

res.redirect("/teacher/");
});

router.get("/list-students", verifyLoginTeacher, (req, res) => {
  studentHelpers.getAllStudents().then((student) => {
    student.forEach((student, index) => {
      student.counter = index + 1;
    });
    let staff = req.session.teacher;
    res.render("student/list-students", { teacher: true, student, staff });
  });
});

router.get("/student-profile/:id", verifyLoginTeacher, async (req, res) => {
  try {
    const studentId = req.params.id;
    const student = await studentHelpers.getStudentById(studentId);
    const workingDays = await teacherHelpers.getWorkingDays();
    const subjects = await subjectHelpers.getAllSubjects();
    let staff = req.session.teacher;
    res.render("principal/student-profile", { student, teacher: true, staff, workingDays, subjects, studentId });
  } catch (error) {
    console.error("Error in /student-profile route:", error);
    res.status(500).send("Internal Server Error");
  }
});

```

## Student mark and attendance

```

updatePresentDays: async (newPresentDays, studentId, workingDays) => {
  try {
    newPresentDays = newPresentDays || 0;
    const percentage = calculateAttendancePercentage(
      newPresentDays,
      workingDays
    ).toFixed(2);

    await db

```

```

.get()
.collection(COLLECTION.STUDENTS_COLLECTION)
.updateOne(
  { _id: new ObjectId(studentId) },
  {
    $set: {
      "attendance.present": newPresentDays,
      "attendance.percentage": parseFloat(percentage), // Convert back to a number
    },
  }
);

return { present: newPresentDays, percentage }; // Return the updated attendance object
} catch (error) {
  console.error("Error in updatePresentDays:", error);
  throw error;
}
},

addMark: async (studentId) => {
  try {
    // Retrieve subjects from the subject collection
    const subjects = await db
      .get()
      .collection(COLLECTION.SUBJECT)
      .find({}, { projection: { subject: 1, _id: 0 } }) // Only retrieve the subject field and exclude _id
      .toArray();

    console.log("Subjects:", subjects);

    // Prepare the marks object with default values (0)
    const markObj = {};
    subjects.forEach((subject) => {
      markObj[subject.subject] = 0;
    });

    // Update the student with the mark object
    await db

```

```

    .get()
    .collection(COLLECTION.STUDENTS_COLLECTION)
    .updateOne(
      { _id: new ObjectId(studentId) },
      { $set: { mark: markObj } }
    );

    return true;
  } catch (error) {
    console.error("Error in addMark:", error);
    throw error;
  }
},
updateStudentMark: async (studentId, subject, mark) => {
  try {
    // Update the mark for the specified subject and student
    await db
      .get()
      .collection(COLLECTION.STUDENTS_COLLECTION)
      .updateOne(
        { _id: new ObjectId(studentId) },
        { $set: { [`mark.${subject}`]: mark } }
      );
  } catch (error) {
    console.error("Error updating student mark:", error);
    throw error;
  }
},

```

## USER MODULE:

User authentication and dashboard redirection.

```

const verifyLoginUser = (req, res, next) => {
  if (req.session.user && req.session.loggedIn) {
    next();
  } else {
    res.redirect("/user-login");
  }
}

```

```

};
/* GET users listing. */
router.get('/', function(req, res) {
  user = req.session.user
  res.render('user/home',{user});
});
router.get('/login',(req,res)=>{
  res.render('user/login')
})
router.get('/admission-req',(req,res)=>{
  res.render('principal/add-student',{login:true})
})
router.post('/req-admission', async (req, res) => {
  try {
    if (req.file) {
      let image = req.file;
      const studentData = req.body;
      const result = await reqHelpers.insertRequestAdmission(studentData);

      const id = result.insertedId;

      const destinationPath = path.join(__dirname, '../public/images/student', id + '.jpg');

      fs.rename(image.path, destinationPath, (err) => {
        if (!err) {
          res.redirect('/');
        } else {
          console.log(err);
          res.status(500).send('Error moving the file');
        }
      });
    } else {
      console.log('No file uploaded');
      res.status(400).send('No file uploaded');
    }
  } catch (error) {
    console.error('Error in /add-student route:', error);
    res.status(500).send('Internal Server Error');
  }
}

```

```

}
});

```

## User entry

```

module.exports = {
  addUser: (userData) => {
    return new Promise(async (resolve, reject) => {
      // Hash the password before storing it
      userData.password = await bcrypt.hash(userData.password, 10);

      // Add user to the 'users' collection
      db.get().collection(collection.USER_COLLECTION).insertOne(userData).then((response) => {
        resolve(response);
      });
    });
  },
  doLogin: (userData) => {
    return new Promise(async (resolve, reject) => {
      let response = {};
      const user = await db
        .get()
        .collection(collection.USER_COLLECTION)
        .findOne({ email: userData.email });

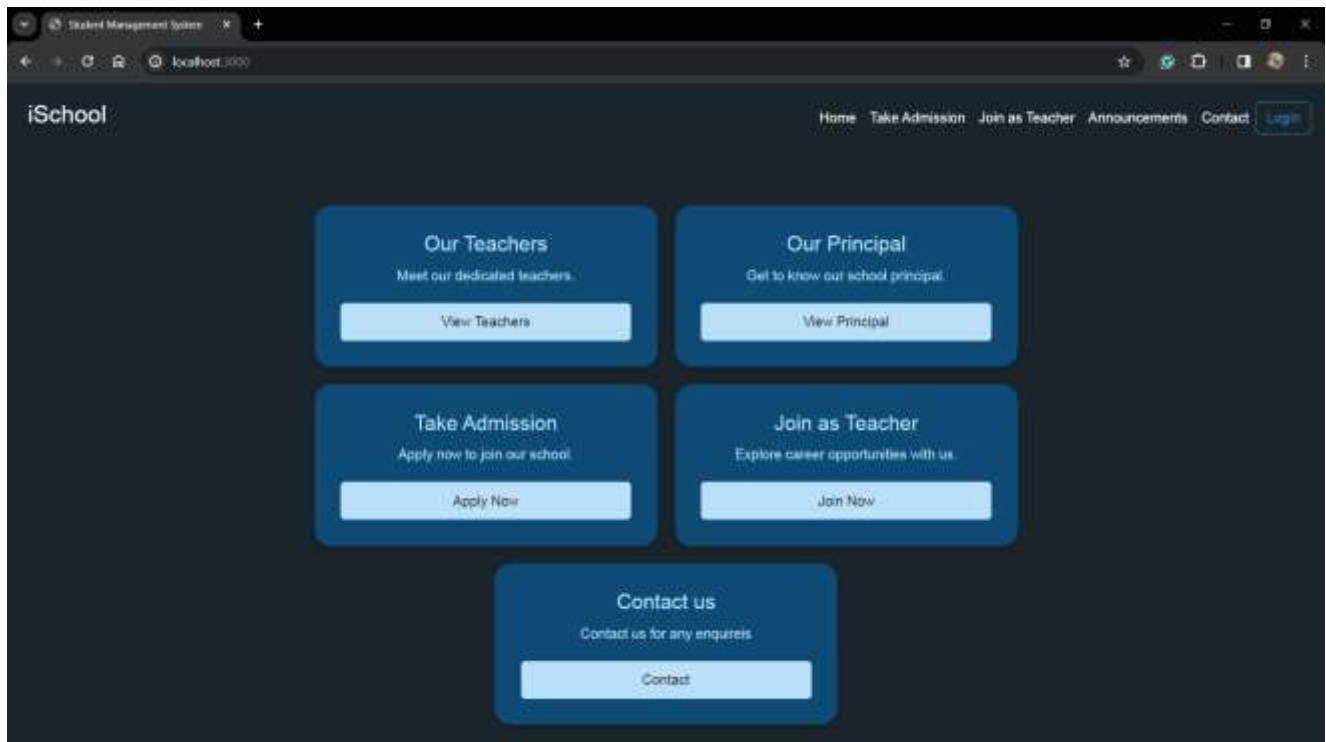
      if (user) {
        // Compare the entered password with the hashed password in the database
        bcrypt.compare(userData.password, user.password).then((status) => {
          if (status) {
            response.user = user;
            response.status = true;
            resolve(response);
          } else {
            response.status = false;
            resolve(response);
          }
        });
      }
    });
  }
};

```

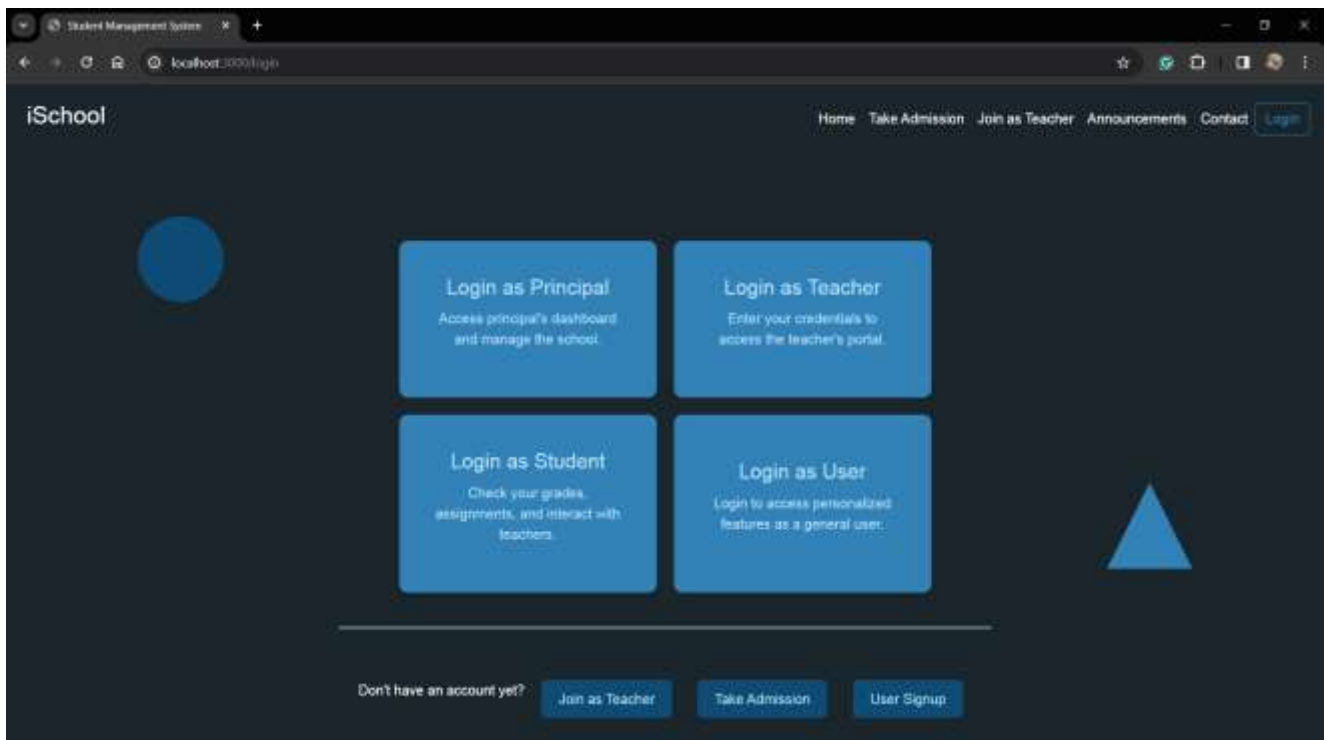
```
    } else {  
        response.status = false;  
        resolve(response);  
    }  
});  
},  
};
```

## RESULT:

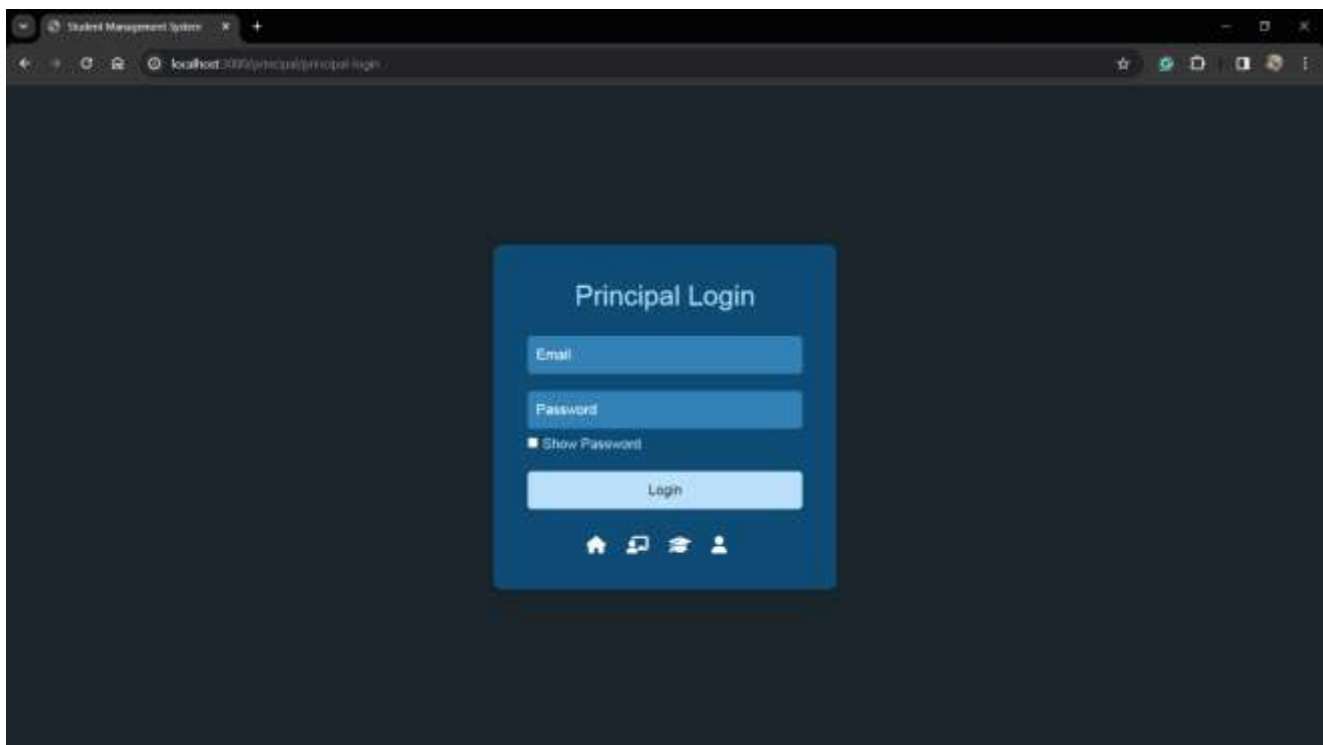
### User Index



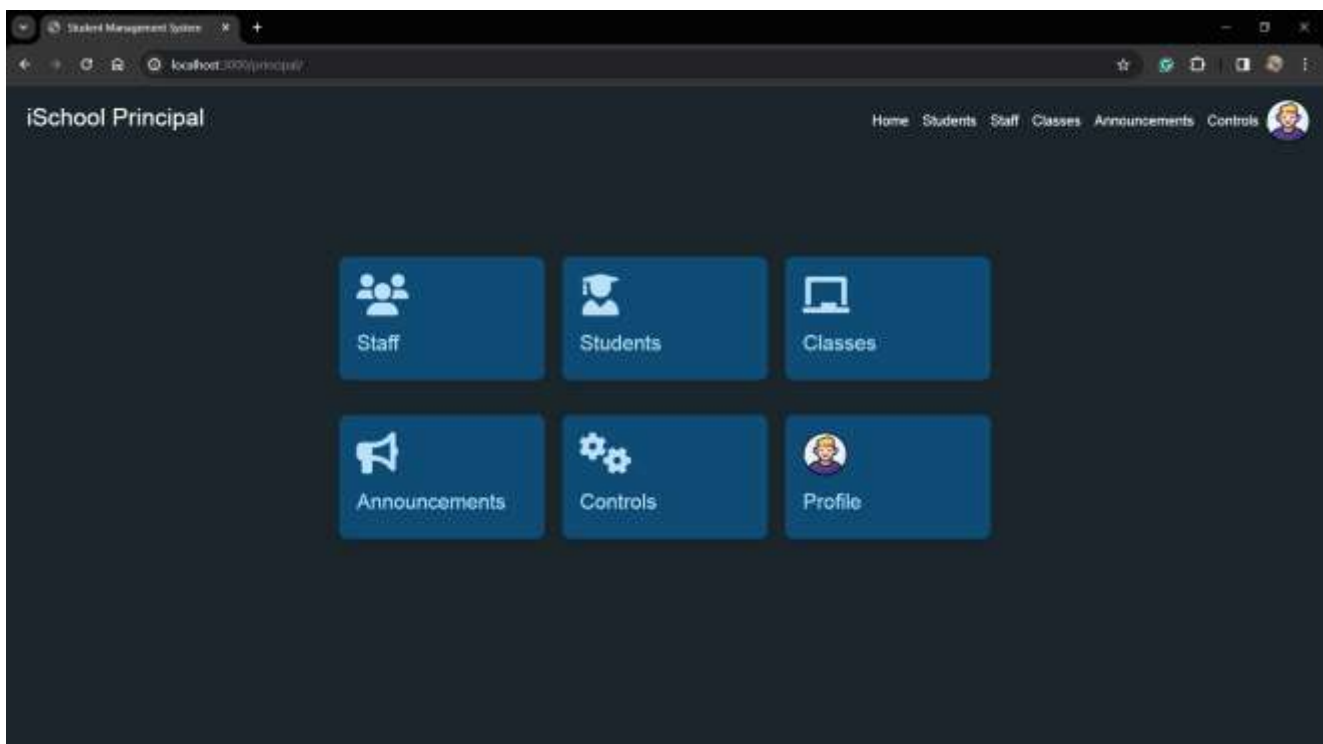
### Login Selection



## Principal Login

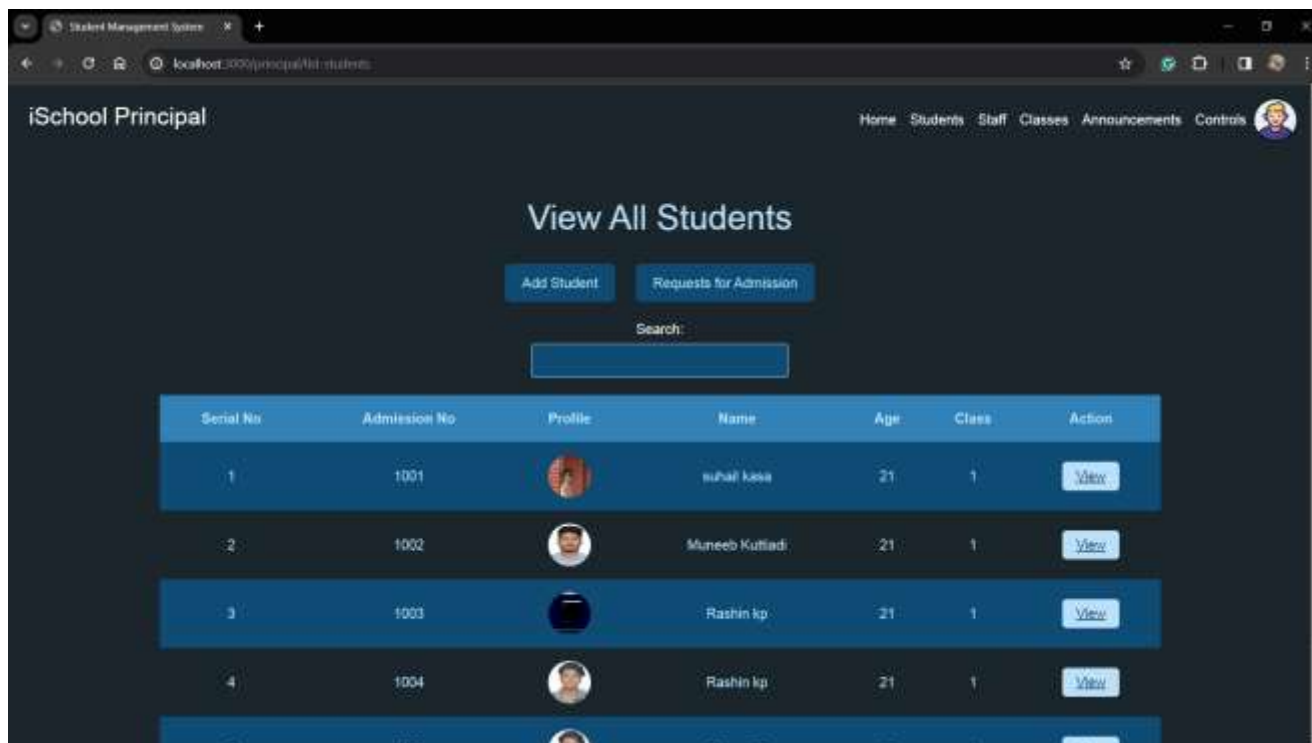


## Principal Dashboard





## All Students







iSchool Principal

Home Students Staff Classes Announcements Controls

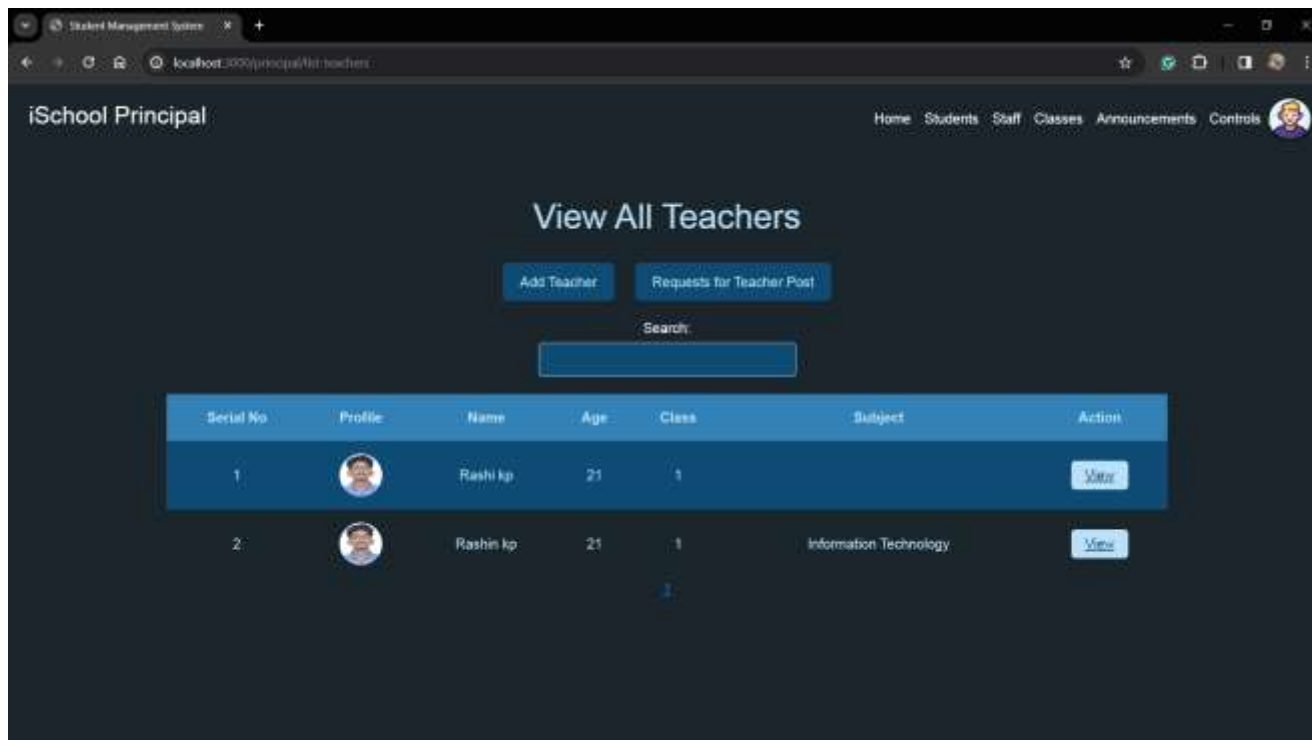
### View All Students

[Add Student](#) [Requests for Admission](#)

Search:

Serial No	Admission No	Profile	Name	Age	Class	Action
1	1001		Suhail Kesa	21	1	<a href="#">View</a>
2	1002		Muneeb Kuttadi	21	1	<a href="#">View</a>
3	1003		Rashin kp	21	1	<a href="#">View</a>
4	1004		Rashin kp	21	1	<a href="#">View</a>

## All Teachers





iSchool Principal

Home Students Staff Classes Announcements Controls

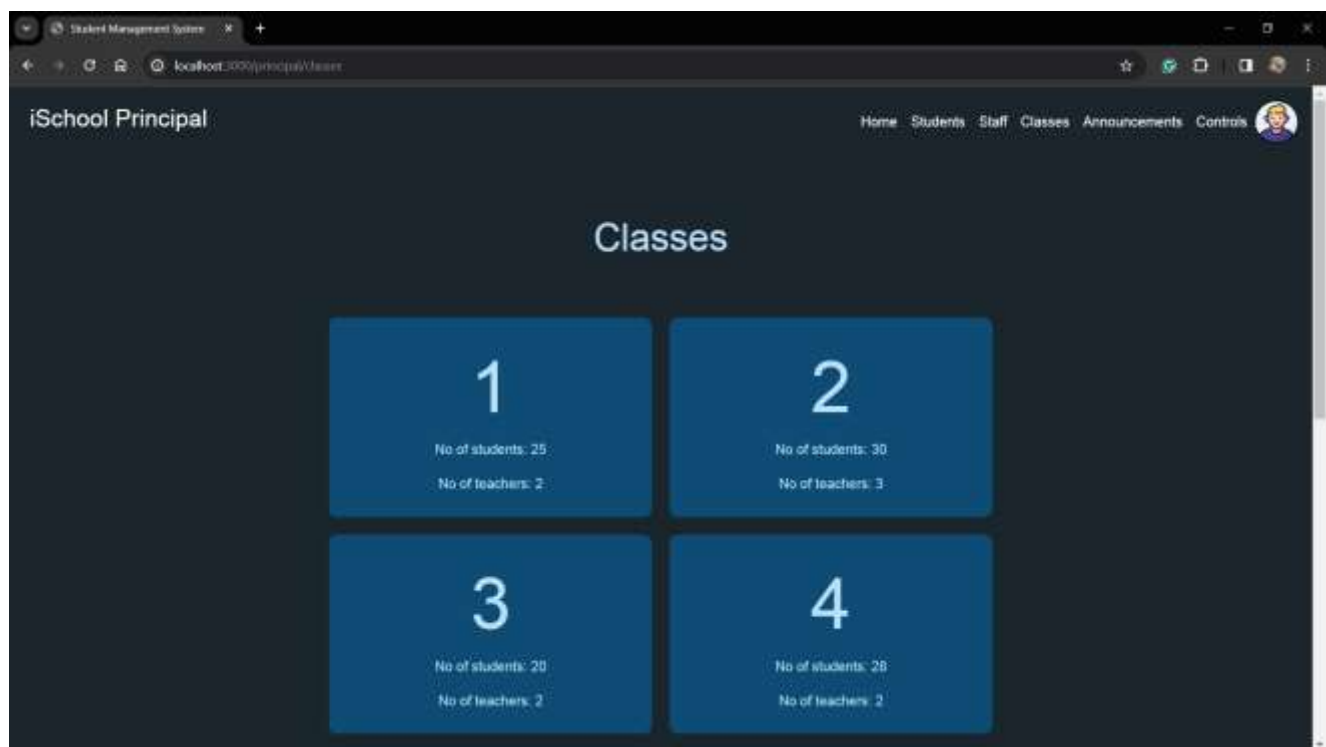
### View All Teachers

[Add Teacher](#) [Requests for Teacher Post](#)

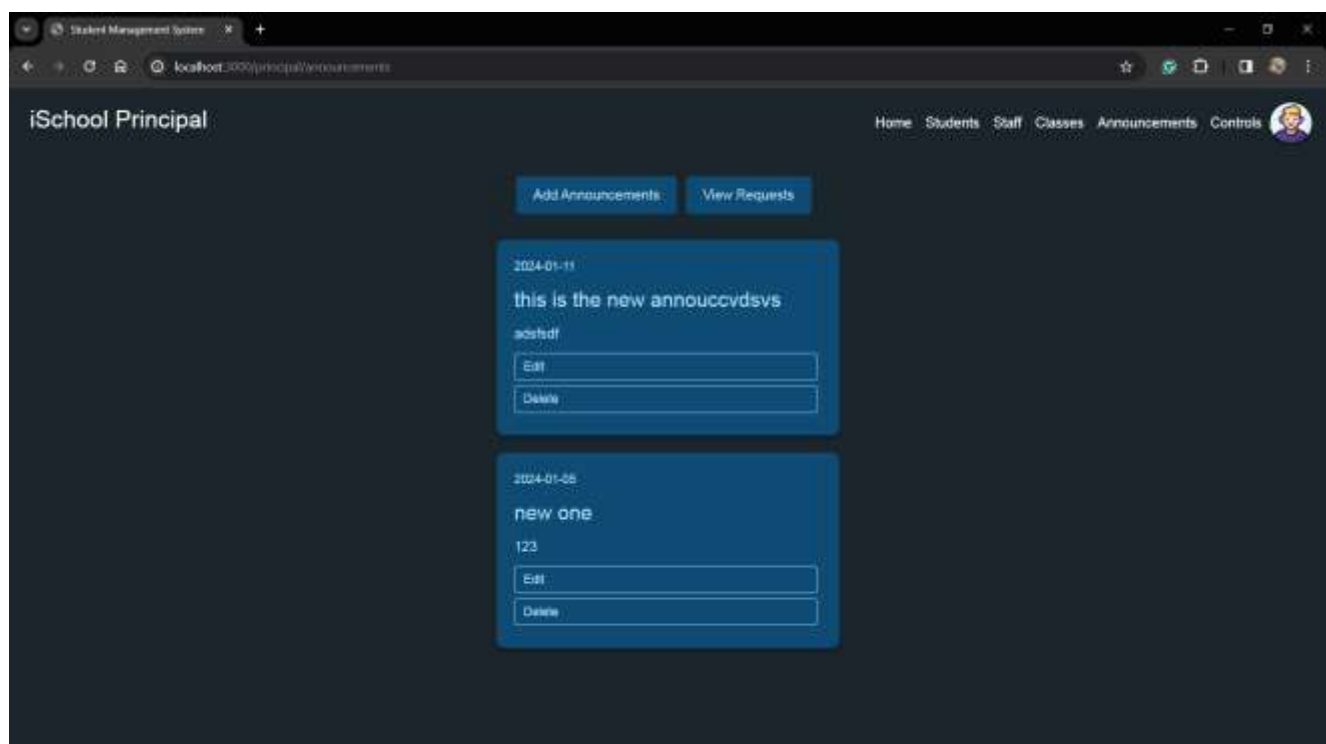
Search:

Serial No	Profile	Name	Age	Class	Subject	Action
1		Rashin kp	21	1		<a href="#">View</a>
2		Rashin kp	21	1	Information Technology	<a href="#">View</a>

## Classes



## Announcement (Principal)



## Controls

**iSchool Principal**

Home Students Staff Classes Announcements Controls

**Total Working Days**

32

**Today**

3/13/2024

**Control Subjects**

Subject Name  
Total Mark

Submit


Subject Name	Total Mark	Controls
English	80	80 Update Remove
Second Language	80	80 Update Remove
Maths	80	80 Update Remove
Basic Science	80	80 Update Remove

## Student Profile (Principal View)

**Student Management System**

localhost:3000/principal/student-profile/63ad76c33d91c048026753

**Profile**



Name: suhail kasa

Admission No: 1001

Age: 21

Sex: male

Class: 1

Roll Number: 2

State: Andhra Pradesh

Place: Malappuram, Trippanachi

Mobile: 99605584300

Email: rishirup001@gmail.com

## Student Profile (Principal View)

The screenshot displays the 'Student Profile (Principal View)' in a web browser. The page has a dark blue header and a light blue sidebar. The main content area is divided into two sections: 'Attendance' and 'Marks'.

**Attendance Section:**

Total Working Days	Student Present Days	Total Percentage
32	24	75%

**Marks Section:**

Subject	Student Mark	Update
English	22	Update
Second Language	12	Update
Maths	21	Update
Basic Science	2	Update

The screenshot displays the 'Student Profile (Principal View)' in a web browser, showing the 'Contact' and 'Controls' sections.

**Contact Section:**

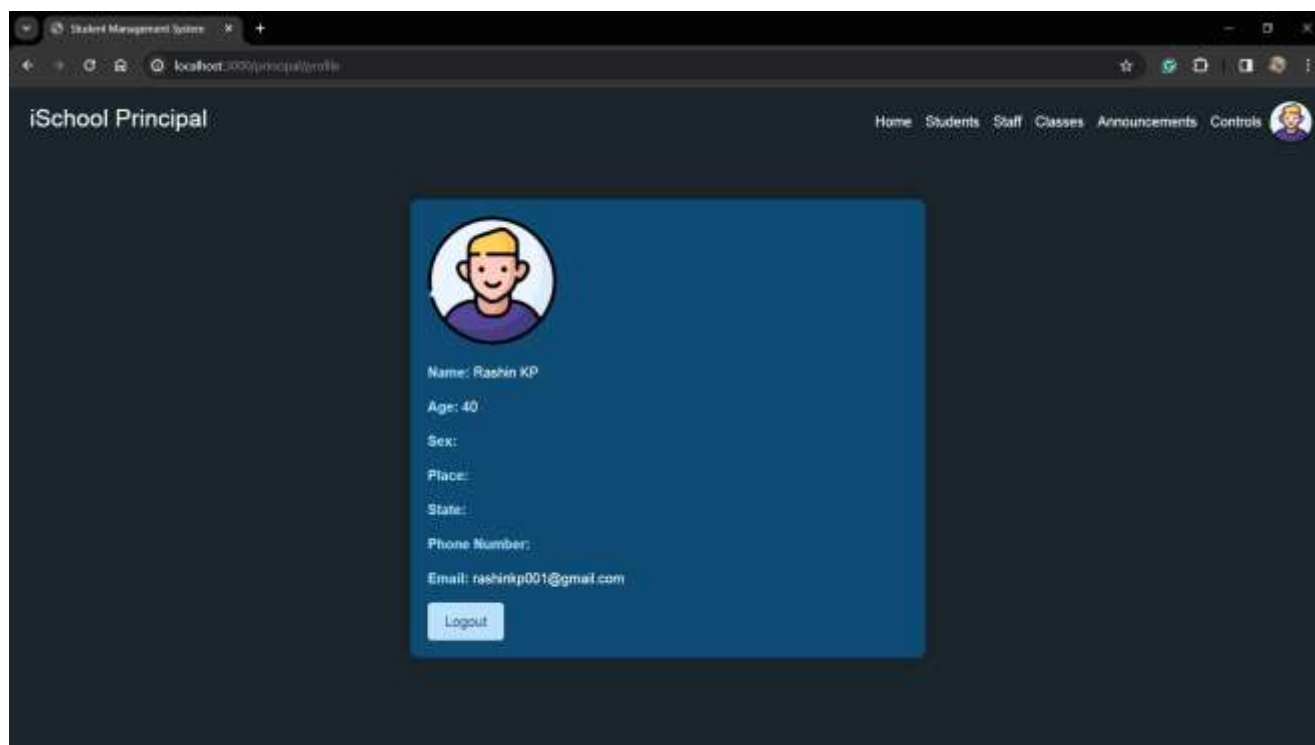
Type your message...

Send Message Call

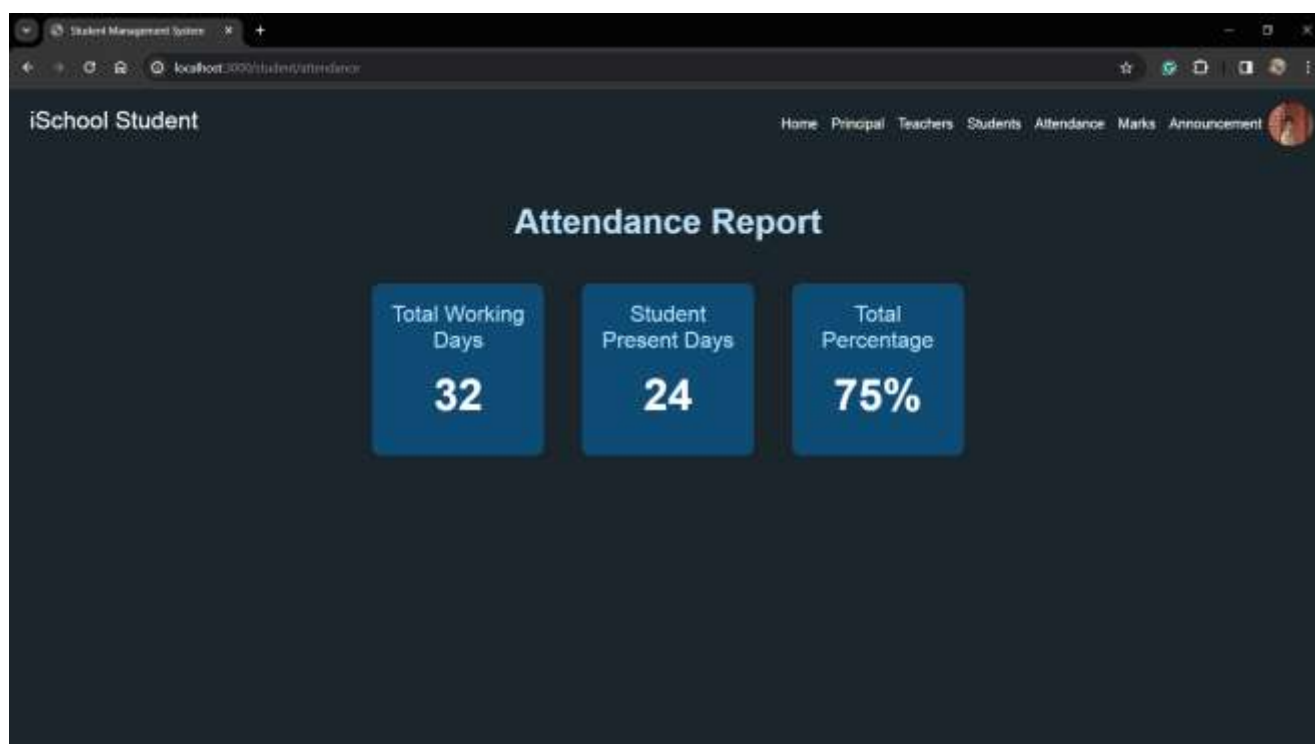
**Controls Section:**

Edit Remove

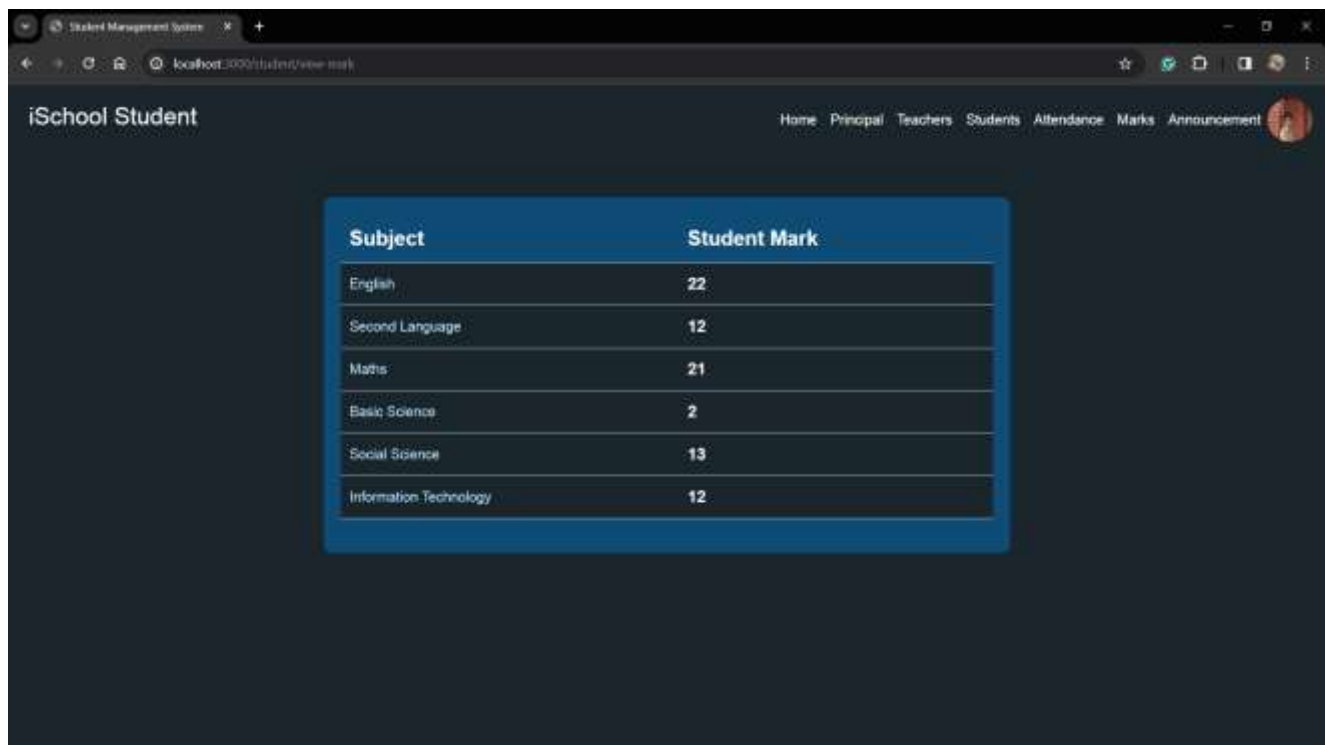
## Principal Profile



## Attendance

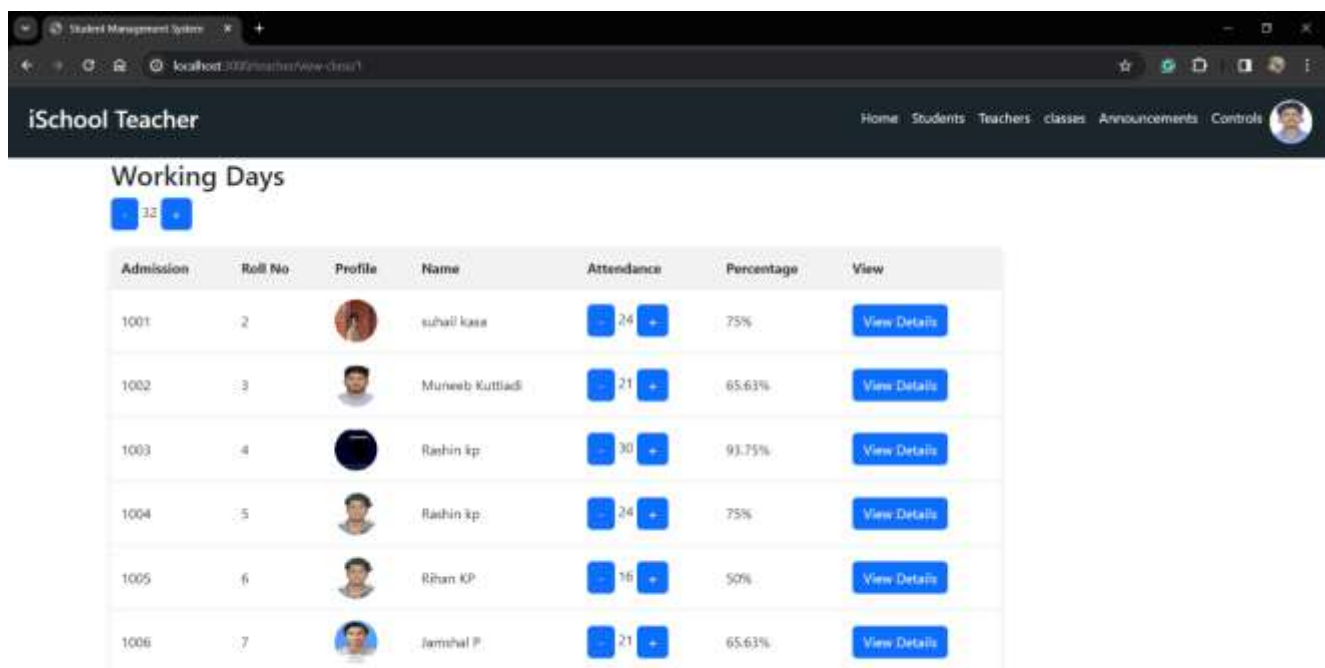


## Mark









Subject	Student Mark
English	22
Second Language	12
Maths	21
Basic Science	2
Social Science	13
Information Technology	12

## Class control



**Working Days**

32

Admission	Roll No.	Profile	Name	Attendance	Percentage	View
1001	2		Suhail Kase	24	75%	<a href="#">View Details</a>
1002	3		Muneeb Kuttialdi	21	65.63%	<a href="#">View Details</a>
1003	4		Rashin Kp	30	93.75%	<a href="#">View Details</a>
1004	5		Rashin Kp	24	75%	<a href="#">View Details</a>
1005	6		Rihan Kp	16	50%	<a href="#">View Details</a>
1006	7		Jamshid P	21	65.63%	<a href="#">View Details</a>

## Student Admission

A screenshot of a web browser displaying the 'Request for Admission' form. The browser's address bar shows 'localhost:3000/whishoo/req'. The form is titled 'Request for Admission' and contains the following fields: 'First Name', 'Second Name', 'Age', 'Sex' (a dropdown menu), 'Class', 'Place', 'District', 'State', 'Choose File' (a file selection button), 'Mobile Number', and a partially visible 'Email' field at the bottom.

A screenshot of a web browser displaying the registration form. The browser's address bar shows 'localhost:3000/whishoo/req'. The form contains the following fields: 'State', 'Choose File' (a file selection button), 'Mobile Number', 'Email', 'Create a Password', 'Confirm Password', and a 'Show Password' checkbox. Below these fields is a 'Submit' button. At the bottom of the form, there is a link that says 'Or login here' and a row of four icons: a house, a person, a group of people, and a person with a checkmark.

## **TESTING**



## 6.1 TESTING

Testing the iSchool system involves a thorough and systematic approach to ensure that it meets its functional requirements, performs reliably, and delivers a positive user experience. Below are various types of tests that can be conducted for the iSchool system:

- **Unit Testing:**

Objective: Verify the correctness of individual units or components.

Examples: Test individual functions, methods, or classes responsible for tasks like student registration, attendance tracking, and announcement management.

- **Integration Testing:**

Objective: Ensure seamless interaction between different components of the system.

Examples: Test interactions between modules, databases, and third-party integrations to ensure proper communication and data flow.

- **Functional Testing:**

Objective: Validate that the system functions as specified in the requirements.

Examples: Test features such as student enrollment, course management, grade calculation, and administrative tasks.

- **Usability Testing:**

Objective: Evaluate the user interface for user-friendliness.

Examples: Assess navigation, form layouts, clarity of information, and accessibility for users with disabilities.

- **Performance Testing:**

Objective: Assess system responsiveness and scalability.

Examples: Conduct load testing to evaluate how the system handles simultaneous user interactions, especially during peak usage periods.

- **Security Testing:**

Objective: Identify and mitigate security vulnerabilities.

Examples: Test for encryption and secure storage of sensitive data, including user credentials and personal information using bcrypt.

- **Compatibility Testing:**

Objective: Confirm that the system works across various platforms and browsers.

Examples: Test the system on different web browsers and devices to ensure consistent functionality and user experience.

- **Regression Testing:**

Objective: Ensure that new changes do not negatively impact existing functionalities.

Examples: Re-run previous test cases after each code change to verify the stability and integrity of the system.

- **User Acceptance Testing (UAT):**

Objective: Validate the system against user expectations and requirements.

Examples: Involve actual users or stakeholders to test the system's usability, functionality, and overall satisfaction.

- **API Testing:**

Objective: Verify the functionality of APIs if the system interacts with external services.

Examples: Test API endpoints for data exchange, error handling, and authentication.

Testing should be conducted iteratively throughout the development lifecycle, with a focus on ensuring the quality, reliability, and security of the iSchool system. Continuous integration and continuous testing practices can help maintain a high level of software quality as the system evolves.

## DEVELOPMENT CHALLENGES

During the development of iSchool, several challenges were encountered, each requiring unique insights and solutions:

### Integration Issues:

- Challenge: Integrating third-party services, such as payment gateways, posed challenges due to version compatibility and API changes.
- Insights: Continuous monitoring of third-party documentation and staying updated on version changes is crucial for smooth integration.

### Performance Bottlenecks:

- Challenge: Managing performance, especially during peak usage times, revealed bottlenecks in database queries and resource-intensive operations.
- Insights: Utilizing caching mechanisms and optimizing database queries significantly improved system responsiveness.

### User Experience Complexity:

- Challenge: Balancing feature richness with a seamless user experience resulted in occasional confusion among users during the event management process.
- Insights: Conducting usability testing with real users provided valuable insights, leading to iterative improvements in user interface design and information flow.

## SOLUTIONS:

### Integration Issues:

- Solution: Maintained close monitoring of documentation, actively participated in developer communities, and regularly tested the system with sandbox environments.
- Insights: Proactive communication with third-party service providers and community engagement are essential for adapting to changes and resolving integration issues promptly.

**Performance Bottlenecks:**

- Solution: Employed caching systems, optimized database queries, and utilized profiling tools.
- Insights: Regular profiling and optimization cycles during development are critical for identifying and resolving performance bottlenecks, ensuring a smooth user experience.

**User Experience Complexity:**

- Solution: Conducted user testing sessions to gather feedback, simplified workflows, and introduced instructional elements.
- Insights: Iterative design improvements based on user feedback are vital for creating an intuitive and user-friendly experience.

**Key Takeaways:**

- Continuous Learning and Adaptation: Staying informed about updates from third-party services and frameworks is crucial for maintaining a secure and functional system.
- User-Centric Development: Prioritizing user experience through iterative testing and improvements is fundamental for creating a successful application.
- Scalability Considerations: Anticipating and addressing scalability challenges during development are essential for long-term system stability.
- Proactive Issue Resolution: Proactively addressing challenges fosters a culture of continuous improvement and adaptability, ensuring a smoother launch and long-term success of the system.

In summary, overcoming development challenges in iSchool involves proactive communication, continuous learning, and a user-centric approach. Solutions implemented during the development phase contribute to creating a robust, scalable, and user-friendly system.

## **LIMITATIONS OF THE PROJECT**

While iSchool offers various benefits, it also faces certain limitations and challenges that need to be addressed:

### **Technical Challenges:**

- **Dependency on Internet Connectivity:** iSchool relies on internet connectivity, making it susceptible to performance issues in areas with slow internet or server downtimes.
- **Compatibility Concerns:** Users may experience compatibility issues with different web browsers, devices, or operating systems, affecting their experience with the system.

### **Security Risks:**

- **Data Security:** Handling sensitive information like student records and personal data requires robust security measures to prevent data breaches.
- **Cybersecurity Threats:** The system is vulnerable to cybersecurity threats such as hacking and data breaches, necessitating stringent security protocols.

### **User Adoption and Training:**

- **User Familiarity:** Users unfamiliar with online education platforms may face a learning curve, requiring comprehensive training and user-friendly interfaces.
- **Resistance to Change:** Some students and teachers may resist transitioning from traditional teaching methods to an online platform, impacting system adoption.

### **Customization and Scalability:**

- **Limited Customization:** iSchool may have constraints in terms of customization, limiting its adaptability to unique requirements of different educational institutions.
- **Scalability Issues:** As the user base and data volume increase, scalability becomes a concern, requiring continuous optimization to maintain performance.

**Dependency on Third-Party Integrations:**

- Reliability of Integrations: Integrations with third-party services for features like attendance tracking and grade management may affect system reliability, depending on the stability of these services.

**Costs and Budgeting:**

- Initial and Ongoing Costs: Implementation and maintenance costs, including software licensing and subscription fees, may pose financial challenges for educational institutions with limited budgets.

**User Experience:**

- Interface Design: Poorly designed interfaces or complex navigation may hinder user experience, requiring improvements to enhance usability.
- Accessibility: Ensuring accessibility for users with disabilities may require additional resources and effort to comply with accessibility standards.

**Regulatory Compliance:**

- Adherence to Regulations: iSchool must comply with data protection regulations such as FERPA and COPPA, requiring continuous updates and monitoring to ensure compliance.

**Complexity of Educational Events:**

- Handling Diverse Events: iSchool may encounter difficulties in managing complex educational events with diverse requirements, including multiple sessions and participant demographics.

**Customer Support:**

- Response Time: Dependence on customer support for issue resolution means that the system's reliability is contingent on the effectiveness and responsiveness of the support team.

Addressing these limitations through strategic planning, continuous improvement, and proactive measures will enhance the overall success and effectiveness of iSchool as an educational management system.

## **FUTURE ENHANCEMENTS**

As iSchool continues to evolve, several enhancements can be considered to enrich its functionality and user experience:

### **Integration with Learning Management Systems (LMS):**

- Objective: Integrate with popular LMS platforms like Moodle or Canvas to streamline course management and content delivery.
- Benefits: Enables seamless sharing of course materials, assignments, and grades between iSchool and existing LMS systems, providing a unified learning environment for students and educators.

### **AI-Powered Personalization:**

- Objective: Implement artificial intelligence algorithms to personalize learning experiences based on individual student preferences, learning styles, and performance.
- Benefits: Enhances student engagement and learning outcomes by delivering tailored content, recommendations, and adaptive assessments.

### **Collaborative Learning Spaces:**

- Objective: Introduce virtual classrooms and collaborative tools to facilitate real-time interactions, group projects, and peer-to-peer learning.
- Benefits: Fosters collaboration, teamwork, and knowledge sharing among students, promoting active learning and social engagement.

### **Gamification Elements:**

- Objective: Incorporate gamification features such as badges, leaderboards, and achievements to incentivize learning, encourage participation, and enhance motivation.
- Benefits: Makes learning more enjoyable and interactive, increasing student engagement and retention.

### **Expanded Content Repository:**

- Objective: Expand the content repository to include a wider range of multimedia resources, interactive simulations, and virtual labs across various subjects and disciplines.

- **Benefits:** Provides diverse learning materials catering to different learning styles and preferences, enriching the learning experience for students.

### **Virtual Reality (VR) Integration:**

- **Objective:** Integrate VR technology to create immersive learning experiences, virtual field trips, and simulations in subjects like science, history, and geography.
- **Benefits:** Enhances student comprehension and retention by offering realistic, hands-on learning experiences that transcend traditional classroom boundaries.

### **Enhanced Assessment and Feedback Tools:**

- **Objective:** Develop advanced assessment tools with automated grading, detailed analytics, and personalized feedback mechanisms.
- **Benefits:** Streamlines assessment processes, provides actionable insights into student performance, and fosters continuous improvement in teaching and learning practices.

### **Blockchain Credentialing:**

- **Objective:** Implement blockchain technology to issue and verify academic credentials, certificates, and badges securely.
- **Benefits:** Enhances the credibility and integrity of academic achievements, simplifies credential verification processes, and empowers students with ownership of their digital credentials.

### **Augmented Reality (AR) Learning Experiences:**

- **Objective:** Create AR-enhanced learning materials and experiences to visualize abstract concepts, historical events, and scientific phenomena.
- **Benefits:** Engages students through interactive, immersive learning experiences that bridge the gap between theory and practice.

### **Accessibility Enhancements:**

- **Objective:** Improve accessibility features to ensure inclusivity for students with disabilities, including screen readers, voice commands, and alternative input methods.
- **Benefits:** Ensures equitable access to educational resources and opportunities for all students, regardless of their physical or cognitive abilities.

These future enhancements aim to transform iSchool into a cutting-edge educational platform that embraces innovation, fosters collaboration, and empowers learners of all backgrounds. By



incorporating these advancements, iSchool can continue to lead the way in delivering high-quality, personalized education in the digital age. Regular feedback from users, educators, and stakeholders will be essential in guiding the prioritization and implementation of these enhancements effectively.

## CONCLUSION

The iSchool platform has achieved remarkable milestones, showcasing the successful integration of Node.js, Express.js, MongoDB, and other technologies. The project's success is evident in several key aspects:

### **Technological Integration:**

- **Node.js and Express.js Backend:** Leveraging Node.js's event-driven architecture and Express.js's robust features streamlined backend development, ensuring efficient request handling and data management.
- **MongoDB Database:** The utilization of MongoDB's flexible NoSQL database provided scalability and performance, allowing seamless integration with Node.js and Express.js.
- **Frontend Technologies:** The combination of HTML, CSS, Bootstrap, and JavaScript facilitated the creation of a responsive and visually appealing user interface, enhancing the overall user experience.

### **Seamless Functionality:**

- **User Authentication and Authorization:** Implementing secure authentication and authorization mechanisms ensured data privacy and user access control, enhancing system security.
- **Data Management:** MongoDB's document-oriented architecture allowed for efficient storage and retrieval of data, contributing to smooth system operations.
- **Real-time Communication:** Integration of WebSocket technology enabled real-time communication between users, facilitating instant messaging and collaboration.

### **User-Centric Design:**

- **Intuitive Interface:** The user interface design focused on simplicity and ease of use, providing a seamless navigation experience for both students and educators.
- **Personalized Learning:** Features such as adaptive content recommendations and progress tracking personalized the learning journey for each student, maximizing engagement and retention.

### **Efficient Learning Management:**

- **Course Management:** The platform's course management capabilities, including content creation, assignment submission, and grading, streamlined the teaching and learning process for educators and students.

- **Interactive Learning Tools:** Integration of interactive learning tools such as quizzes, forums, and virtual labs enriched the learning experience, fostering active participation and knowledge retention.

**Achieving Objectives:**

- **Centralized Learning Platform:** iSchool serves as a centralized platform for educational resources, facilitating collaboration, communication, and knowledge sharing among students and educators.
- **Effective Learning Outcomes:** The platform successfully meets its objectives of delivering quality education, fostering student engagement, and promoting academic excellence.

**Scalability and Adaptability:**

- **Scalable Architecture:** The project's architecture is designed with scalability in mind, allowing for future growth and expansion to accommodate increasing user demands.
- **Adaptability to Change:** Continuous updates and enhancements ensure that the platform remains relevant and adaptable to changes in technology and educational practices.

In conclusion, the successful integration of Node.js, Express.js, MongoDB, and other technologies has resulted in an innovative and efficient learning management platform in iSchool. By addressing the diverse needs of students and educators, iSchool provides a comprehensive solution for modern education, empowering learners and educators alike in their academic pursuits. The project's robust architecture, user-centric design, and seamless functionality contribute to its overall success and effectiveness in transforming the educational landscape.

## **BIBLIOGRAPHY**

- <https://www.w3schools.com/>
- <https://openai.com/>
- <https://github.com/>
- <https://gemini.google.com/>