

Tribhuvan University

Faculty of Humanities and Social Sciences

“Academia-Plus” Academic Management System(AMS)

A PROJECT REPORT

Submitted to

Department of Computer Application

Lumbini City College

*In partial fulfillment of the requirements for the Bachelors in Computer Application*

Submitted by

Suman Khatri

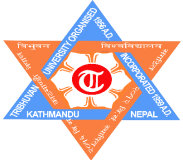
(T.U.Reg.No: 6-2-1134-60-2020)

Chitra Bahadur Thapa

(T.U.Reg.No: 6-2-1134-42-2020)

Under the Supervision of

**Dinesh Neupane**



**Tribhuvan University**

**Faculty of Humanities and**

**Social Sciences**

**Lumbini City College**

**Supervisor’s Recommendation**

I hereby recommend that this project prepared under my supervision by Suman Khatri and Chitra Bahadur Thapa entitled “**AcademiaPlus**” in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

**SIGNATURE**

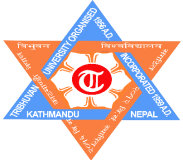
**Dinesh Neupane**

**SUPERVISOR**

Faculty of Humanities and Social Sciences

Lumbini City College

Tilottama-4, Drivertole, Rupandehi



**Tribhuvan University**

**Faculty of Humanities and**

**Social Sciences**

**Lumbini City College**

**LETTER OF APPROVAL**

This is to certify that this project prepared by Suman Khatri and Chitra Bahadur Thapa entitled “**AcademiaPlus”** in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| Mr. Dinesh Neupane  Supervisor  Lumbini City Collage | Mr. Bishal Kandel  Program Coordinator  Lumbini City Collage |
| Internal Examiner  Lumbini City Collage | External Examiner  FOHSS, Tribhuvan University |

# Abstract

The AcademiaPlus application is a comprehensive platform designed to streamline and enhance the management of educational institutions, including schools, colleges, and universities. By integrating essential functions such as student registration, class management, grading, attendance tracking, and student management, AcademiaPlus centralizes academic workflows into a unified system. This integration provides real-time access to academic records and fosters effective communication between students, faculty, and administrators. The system's ability to automate administrative tasks improves efficiency, reduces workload, and enables more accurate data-driven decision-making.

AcademiaPlus supports institutions by offering data analytics for performance tracking, which assists in monitoring academic progress and informs decision-making processes at various levels. Its scalability and adaptability make it a valuable solution for institutions of different sizes and complexities. By promoting a more organized and efficient academic environment, AcademiaPlus enhances both institutional governance and student outcomes.

As educational institutions evolve, the implementation of AcademiaPlus becomes increasingly critical in addressing the dynamic needs of modern academia. By automating routine processes and centralizing critical data, AcademiaPlus reduces administrative burdens, improves communication, and contributes to the growth and success of educational institutions.

Keyword: *grading, attendance tracking, data analytics, automation.*

# Acknowledgement

We are obliged to a number of people who helped organize this project, and we are thankful for their kind opinions, suggestions, and appropriate guidance. We have received endless support and encouragement in finalizing our project, and we would like to take this opportunity to thank them all.

First and foremost, we would like to express our special thanks of gratitude to our supervisor, Dinesh Neupane, for giving us the honor to work on this wonderful project on the topic of the Academic Management System Application, which has greatly enhanced our knowledge. We are also thankful for his continuous feedback and encouragement throughout the project. Without his guidance, this project would not have been accomplished. We have learned many new things while working on this project, and for that, we are deeply thankful to him.

Secondly, we are overwhelmed with all the support and guidance we received from our parents and friends. Their assistance and innovative ideas have been invaluable in helping us complete this project satisfactorily. We are truly grateful for their help in making this project a success.

Lastly, we would like to extend our gratitude to everyone who has encouraged and supported us through the challenges we faced during the completion of this project.

Suman Khatri and Chitra Bahadur Thapa

**Table of Contents**

[Abstract iii](#_Toc177321065)

[Acknowledgement iv](#_Toc177321066)

[Chapter 1: Introduction 1](#_Toc177321067)

[1.1 Introduction 1](#_Toc177321068)

[1.2 Problem Statement 1](#_Toc177321069)

[1.3 Objective 2](#_Toc177321070)

[1.4 Scope and Limitation 2](#_Toc177321071)

[1.5 Development Methodology 3](#_Toc177321072)

[1.6 Report Organization 4](#_Toc177321073)

[Chapter 2: Background Study and Literature Review 5](#_Toc177321074)

[2.1 Background Study: 5](#_Toc177321075)

[2.2 Literature Review: 5](#_Toc177321076)

[Chapter 3: System Analysis and Design 7](#_Toc177321077)

[3.1 System Analysis 7](#_Toc177321078)

[3.1.1 Requirement Analysis 7](#_Toc177321079)

[3.1.2 Feasibility Analysis 8](#_Toc177321080)

[3.1.3 Data Modelling: ER Diagram 9](#_Toc177321081)

[3.1.4 Process Modelling: DFD 10](#_Toc177321082)

[3.2 System Design 11](#_Toc177321083)

[3.2.1 Architectural Design 11](#_Toc177321084)

[3.2.2 Database Schema Design 12](#_Toc177321085)

[3.2.3 Interface Design (UI/UX) 12](#_Toc177321086)

[3.3 Algorithm details 14](#_Toc177321087)

[Chapter 4: Implementation and Testing 15](#_Toc177321088)

[4.1 Implementation 15](#_Toc177321089)

[4.1.1 Tools Used 15](#_Toc177321090)

[4.1.2 Implementation Details of Modules 15](#_Toc177321091)

[4.2 Testing 15](#_Toc177321092)

[4.2.1 Test Cases for Unit Testing 15](#_Toc177321093)

[4.2.2 Test Cases for System Testing 16](#_Toc177321094)

[Chapter 5: Conclusion and Future Recommendations 17](#_Toc177321095)

[5.1 Conclusion 17](#_Toc177321096)

[5.2 Future Recommendations 17](#_Toc177321097)

[References 18](#_Toc177321098)

[Appendices 19](#_Toc177321099)

**List of Abbreviations**

API – Application Programming Interface

CSS – Cascading Style Sheets

DBMS – Database Management System

HTML – Hypertext Markup Language

JS – JavaScript

JWT – JSON Web Token

SSL – Secure Socket

UI – User Interface

UX – User Experience

WS – WebSocket

**Table of Figures**

[Figure 1 Waterfall Methodology 4](#_Toc177298276)

[Figure 2 Use Case of Academic Management Systemting Application 7](file:////Users/mac/Desktop/final.docx#_Toc177298277)

[Figure 3 Gantt Chart 9](file:////Users/mac/Desktop/final.docx#_Toc177298278)

[Figure 4 ER-Diagram of Academic Management Systemting Application 9](file:////Users/mac/Desktop/final.docx#_Toc177298279)

[Figure 5 Zero Level DFD 10](file:////Users/mac/Desktop/final.docx#_Toc177298280)

[Figure 6 First Level DFD 10](file:////Users/mac/Desktop/final.docx#_Toc177298281)

[Figure 7 Second Level DFD 11](file:////Users/mac/Desktop/final.docx#_Toc177298282)

[Figure 8 Architecture Design 11](file:////Users/mac/Desktop/final.docx#_Toc177298283)

[Figure 9 Database Schema Design 12](file:////Users/mac/Desktop/final.docx#_Toc177298284)

[Figure 10 UI/UX of Index Page 13](file:////Users/mac/Desktop/final.docx#_Toc177298285)

[Figure 11 UI/UX of Register Page 13](file:////Users/mac/Desktop/final.docx#_Toc177298286)

[Figure 12 UI/UX of Login Page 14](file:////Users/mac/Desktop/final.docx#_Toc177298287)

**List of Tables**

[Table 1 Test Cases for Unit Testing 15](#_Toc177318297)

[Table 2 Test Cases for System Testing 16](#_Toc177318298)

# Chapter 1: Introduction

## **1.1 Introduction**

An Academic Management System is a digital platform designed to optimize the administrative and academic functions of educational institutions. With the increasing complexity of managing academic workflows in schools, colleges, and universities, AMS serves as a vital tool to efficient processes such as student enrollment, course management, attendance tracking, grading, faculty coordination, and exam scheduling. By consolidating these tasks into a centralized system, AMS enhances efficiency, reduces manual errors, and ensures that students, faculty, and administrators have real-time access to important academic information. As education continues to embrace digital transformation, AMS plays a crucial role in facilitating a more organized, data-driven, and collaborative academic environment. The development of these applications often leverages frameworks such as React, Material-UI for the frontend. Node.js and Express.js for backend while using databases like MongoDB to store histories and user data.

## **1.2 Problem Statement**

Educational institutions face numerous challenges in managing their academic processes efficiently. Traditional methods often rely on manual record-keeping and disjointed systems, leading to several key issues:

1. Inefficient Data Management: Manual entry and paper-based records are prone to errors, making it difficult to maintain accurate student information, course details, and academic performance metrics.
2. Limited Communication: Inadequate communication channels hinder interaction between students, faculty, and administrative staff, resulting in misunderstandings and delays in disseminating important information.
3. Time-Consuming Administrative Tasks: Faculty and administrative personnel spend excessive time on routine tasks such as attendance tracking, grading, and scheduling, detracting from their core responsibilities of teaching and supporting students.
4. Difficulty in Tracking Academic Progress: Students often struggle to access their academic performance data in a timely manner, making it challenging to identify areas for improvement and engage proactively with their education.
5. Scalability Issues: As institutions grow, the complexity of managing academic processes increases. Existing systems may not scale effectively, resulting in bottlenecks and decreased operational efficiency.

## **1.3 Objective**

The objectives of an Academic Management System are designed to enhance the efficiency and effectiveness of educational institutions.

1. Streamlined Data Management: To centralize and automate the collection, storage, and retrieval of student and academic data, ensuring accuracy and accessibility.
2. Enhanced Communication: To facilitate seamless communication between students, faculty, and administrative staff through integrated messaging systems, announcements, and forums.
3. Efficient Administrative Processes: To reduce the administrative burden on faculty and staff by automating routine tasks such as attendance tracking, grading, and scheduling, thereby allowing them to focus on teaching and student support.
4. Improved Academic Monitoring: To provide tools for tracking student performance and progress, enabling timely feedback and intervention to support academic success.
5. Flexible Course Management: To enable easy scheduling and management of courses, resources, and faculty assignments, minimizing conflicts and optimizing resource allocation.

By achieving these objectives, an Academic Management System aims to enhance the overall educational experience, improve institutional efficiency, and contribute to the academic success of students.

## **1.4 Scope and Limitation**

Scope of an Academic Management System

1. Student Management: AMS facilitates the management of student information, including personal details, enrollment, grades, attendance, and academic history.
2. Course and Curriculum Management: It allows administrators to create, modify, and manage course offerings, syllabi, and class schedules.
3. Faculty Management: AMS enables institutions to manage faculty profiles, assign teaching roles, track attendance, and monitor performance.
4. Examination and Grading: The system automates exam scheduling, grading, and result generation, reducing manual errors and delays.
5. Attendance Tracking: AMS tracks student attendance and provides automated reports to faculty and administrators.
6. Communication and Collaboration: It provides communication tools for students, faculty, and administrators to share announcements, assignments, and feedback.
7. Fee Management: The system helps track tuition and other fees, enabling efficient financial management and payment processing.
8. Analytics and Reporting: AMS generates real-time data analytics and reports to support decision-making and track institutional performance.

Limitations of an Academic Management System

1. Initial Setup Costs: Implementing an AMS can be expensive, requiring investments in software, hardware, and training.
2. Customization Complexity Institutions may have unique requirements, and the system may require extensive customization to meet those needs.
3. Data Security and Privacy: AMS handles sensitive academic and personal data, making it vulnerable to cyber threats if not properly secured.
4. Dependency on Internet Connectivity: Many AMS platforms are cloud-based, which may limit functionality in regions with poor internet access.
5. Technical Support: Continuous technical support is necessary for system maintenance, troubleshooting, and upgrades, which can be resource-intensive.
6. User Adaptability: Faculty, staff, and students may face a learning curve when transitioning to a new system, which could initially impact productivity.
7. Integration with Other Systems: Difficulty in integrating AMS with existing systems (such as Learning Management Systems, ERP software) may pose challenges for institutions with pre-established technology infrastructure.
8. The scope and limitations of AMS must be carefully considered to ensure it meets the specific needs of the institution while addressing any potential drawbacks.

## **1.5 Development Methodology**

We will use the Waterfall Methodology to build the Academic Management System application. This approach is structured and linear, meaning we complete each development phase before moving to the next. It involves detailed planning and documentation at each step, ensuring all requirements are clearly defined upfront. The Waterfall model includes sequential phases: requirements gathering, system design, implementation, integration and testing, deployment, and maintenance. This method helps keep the project organized and focused, with clear goals for each stage.

1. Requirement: In this phase, we will collect and document all the key requirements for the Academic Management System application. This includes features like real-time messaging, user authentication, and security. We will also analyze any specific user needs and define a clear project scope before proceeding to the next phase.
2. System Design: After finalizing the requirements, we move to system design. This involves creating detailed plans for the application’s architecture, database structure, user interfaces, and workflows. Key components like the messaging system, authentication, and user interface will be carefully designed to meet the project’s requirements.
3. Implementation: Once the design is finalized, developers start coding the Academic Management System application. They build the frontend and backend based on the design documents, focusing on creating the user interface and adding real-time messaging features.
4. Integration and Testing: After building the application, we'll integrate its components and conduct thorough testing for functionality, security, and performance. Any bugs or issues will be identified and fixed to ensure the application meets the defined requirements.
5. Deployment: After successful testing, the Academic Management System application will be deployed to the production environment. At this stage, the application is released to the users, and final configurations are made to ensure smooth operation.
6. Maintenance: After deployment, the application enters the maintenance phase. Any issues or bugs that arise will be fixed, and minor updates or enhancements can be made based on user feedback. Major changes usually require starting a new project cycle.

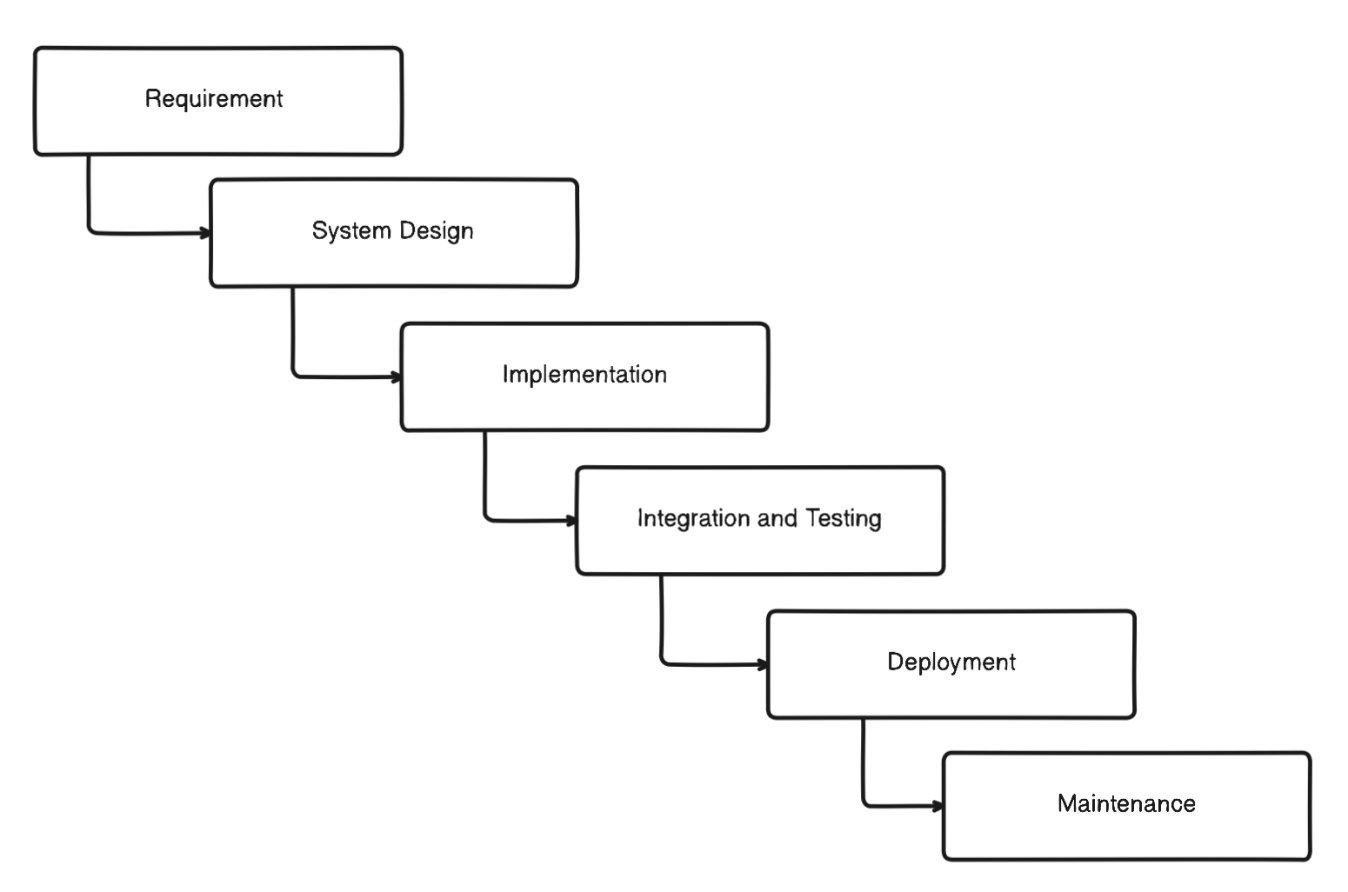


Figure 1 Waterfall Methodology

## **1.6 Report Organization**

This report is divided into five chapters, each with specific headings. The preliminary section includes general project information such as the abstract, table of contents, list of figures, and abbreviations.

* **Chapter 1** introduces the Academic Management System application, detailing the problem definition, objectives, scope, and limitations of the system.
* **Chapter 2** provides a background study and literature review, summarizing existing research related to Academic Management System applications and relevant technologies.
* **Chapter 3** covers the analysis and design of the system, including information about the current systems, data collection methods, system analysis, feasibility study, and system configuration. It also presents the overall system architecture, class diagrams, sequence diagrams, and database diagrams.
* **Chapter 4** details the development models, implementation techniques, tools used, and test cases for system testing.
* **Chapter 5** discusses the future scope of the project, recommendations for future improvements, and provides a conclusion.

# Chapter 2: Background Study and Literature Review

## **2.1 Background Study:**

Academic Management System is a software solution designed to streamline and automate the administration of educational institutions, encompassing key activities such as student enrollment, course management, faculty coordination, grading, and attendance tracking. Historically, academic institutions relied on manual processes for these operations, which were often time-consuming, error-prone, and inefficient. With the growing scale and complexity of modern educational systems, the need for a digital transformation became imperative. The development of an Academic Management System leveraging modern web technologies is a strategic approach to creating a robust, efficient, and scalable platform for educational institutions. In this system, React is used for the frontend, providing a dynamic and responsive user interface that enhances user experience through its component-based architecture. Material-UI, a popular React framework, is integrated for pre-built and customizable UI components, ensuring a visually appealing and consistent design.

On the backend, Node.js and Express form a powerful combination for handling server-side logic and APIs, offering fast, event-driven performance. This enables the AMS to process requests and manage operations such as student records, course data, and faculty management efficiently. The system's database is powered by MongoDB, a NoSQL database that offers flexibility in storing unstructured data, allowing the system to manage complex datasets like course structures, attendance logs, and grading records with ease.

For secure authentication, JSON Web Token (JWT) is implemented, ensuring that user sessions are safely managed and protected from unauthorized access. This is particularly crucial when handling sensitive information such as student data and academic results. Additionally, Axios is used for handling HTTP requests, enabling smooth communication between the frontend and backend, while Mongoose, an Object Data Modeling (ODM) library, simplifies interaction with MongoDB by providing schema-based models for data handling.

Together, these technologies create a comprehensive AMS that is scalable, secure, and responsive, meeting the growing needs of modern educational institutions.

## **2.2 Literature Review:**

This project is aimed at developing a College Management Information System (CMIS) that is of importance to either an educational institution or a college. It is difficult to prepare the manual work to store the information about the all students, teachers as well as about workers. This system can be used as a knowledge/information management system for the college. So this project helps to store those type of information using computerized system. The title of the project is “COLLEGE MANAGEMENT INFORMATION SYSTEM” (CMIS). CMIS is an Intranet based application that aims at providing information to all the levels of management within an organization. This system can be used as a information management system for the college. The front-end will be HTML pages with Java Script for client side validation where as all business logics will be in Java reside at middle layer. And these layers will interact with third layer of database, which will be MS-access database. The web server will be Glassfish. To start working on this project environment required is a server having Glassfish as web server, MS-access as database and Java Runtime Environment (JRE) as development environment. The project is divided into 6 scenarios; each scenario can be developed independently and knowledge of JSP with MS-access is desirable to execute this project [1]**.**

Introduction This project aims to deal with data of college students and also the fees that they have to pay in the college. 1.1 Purpose The purpose of the college management system is used to search and store students' data, find out what fees are required of them, etc. 1.2 Scope The scope of the college management system is as follows: This is a system that contains both a student data component and a student's paid or remaining fee component. 1.3 Technologies used • NetBeans • Java (as a programming language) • MySQL [2].

This paper is aimed at developing an Online College Management System that is of importance to the educational institute or college. This system is named College ERP using MERN stack. This system may be used to monitor college students and their various activities. This application is being developed for an engineering college to maintain and facilitate ease of access to information. For this the users must be registered with the system. College ERP is an Internet based application that aims at providing information to all levels of management within an organization. This system is used as an information management system for the college. For a given student and staff (technical and non-technical) can access the system to either upload and access some information from the database [3].

The Campus Management System (CMS) represents a pivotal advancement in educational technology, offering an integrated software solution tailored to the complex needs of modern educational institutions. This comprehensive system is designed to revolutionize administrative tasks, enhance efficiency, and foster a more organized campus environment. Through seamless automation of various processes, CMS empowers administrators, faculty, and students to navigate campus operations with ease. Utilizing Full Stack Development, CMS ensures seamless integration of front-end and back-end components, resulting in a user-friendly interface accessible to all stakeholders. This approach streamlines administrative tasks, simplifying processes such as student enrolment, course management, and resource allocation. Furthermore, CMS harnesses the power of Machine Learning (ML) algorithms to optimize key processes. These ML algorithms enable intelligent automation of tasks such as resume evaluation, eligibility criteria assessment, and course recommendations based on student performance. By analyzing vast amounts of data, CMS empowers institutions to make data-driven decisions, enhancing the overall educational experience for students. The integration of technology within educational institutions through CMS represents a significant step forward in campus management. By leveraging Full Stack Development and Machine Learning algorithms, CMS offers a robust solution to the challenges faced by modern educational institutions, ultimately paving the way for a more efficient, effective, and student-centric campus environment [4].

# Chapter 3: System Analysis and Design

## **3.1 System Analysis**

## **3.1.1 Requirement Analysis**

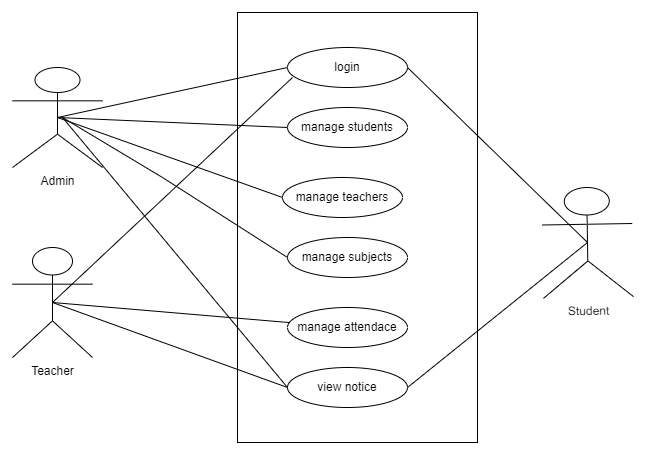
1. **Functional Requirements:**

Functional requirements for Academic Management System Application might include:

* User Management: Role based access for different users.
* Attendance Tracking: Monitoring and reporting attendance.
* Analytics and Reporting: Dashboards for performance tracking and customizable reports.
* Subject Management: Creation and management of subjects..

Function Requirement can be expressed in Use Case for

Figure 2 UseCase



1. **Non-Functional Requirements:**

Non-functional requirements are the characteristics or qualities of a Academic Management System that cannot be directly observed by end-users but are still crucial for the application's overall performance and user experience. Here are some non-functional requirements for a Academic Management System application:

i. Security measures for data protection: Implement encryption and secure authentication protocols to protect user data and conversations from unauthorized access.

ii. User-friendly and responsive design: Ensure the academic management system is intuitive, easy to navigate, and responsive across various screen sizes and devices for a seamless user experience.

iii. Availability, usability, and scalability: The Academic management system must be available at all times, easy to use, and able to scale efficiently as the number of users grows.

iv. High-performance system with low latency: Messages should be delivered and received instantly with minimal delay, ensuring smooth and fast real-time communication.

v. Compatibility with different browsers and devices: The application should work consistently across major browsers (e.g., Chrome, Firefox) and on various devices, including desktops, tablets, and smartphones.

## **3.1.2 Feasibility Analysis**

i. Technical Feasibility:

Technical feasibility assesses whether the current infrastructure can support the academic management system application. It identifies the necessary software tools, procedures,

and inputs required for development. The academic management system is designed to be user-friendly and fast, reducing maintenance costs. With its efficient processing speed, the system is considered technically and operationally feasible.

1. Operational Feasibility: Operational feasibility for an Academic Management System refers to its ability to function effectively and efficiently in the current technological environment. A clear operational plan is essential for smooth operation and user satisfaction, ensuring the system meets user needs efficiently.
2. Economic Feasibility: Economic analysis is used to evaluate the effectiveness of the Academic Management System application by comparing its costs and benefits. The application is economically feasible as it doesn't require additional hardware resources and saves significant time, making it a cost-effective solution
3. Schedule Feasibility:

This includes the project schedule and all time allocated for their completion. The Gantt chart is as follow:

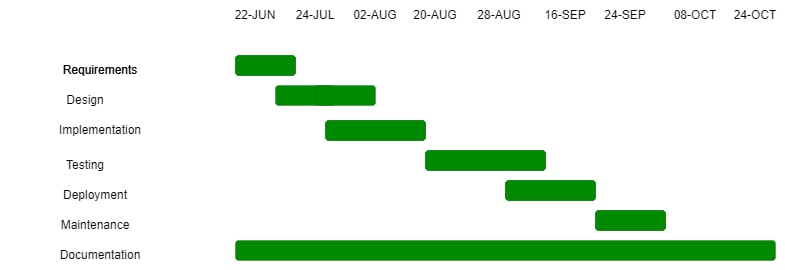


Figure 3 Gantt Chart

## **3.1.3 Data Modelling: ER Diagram**

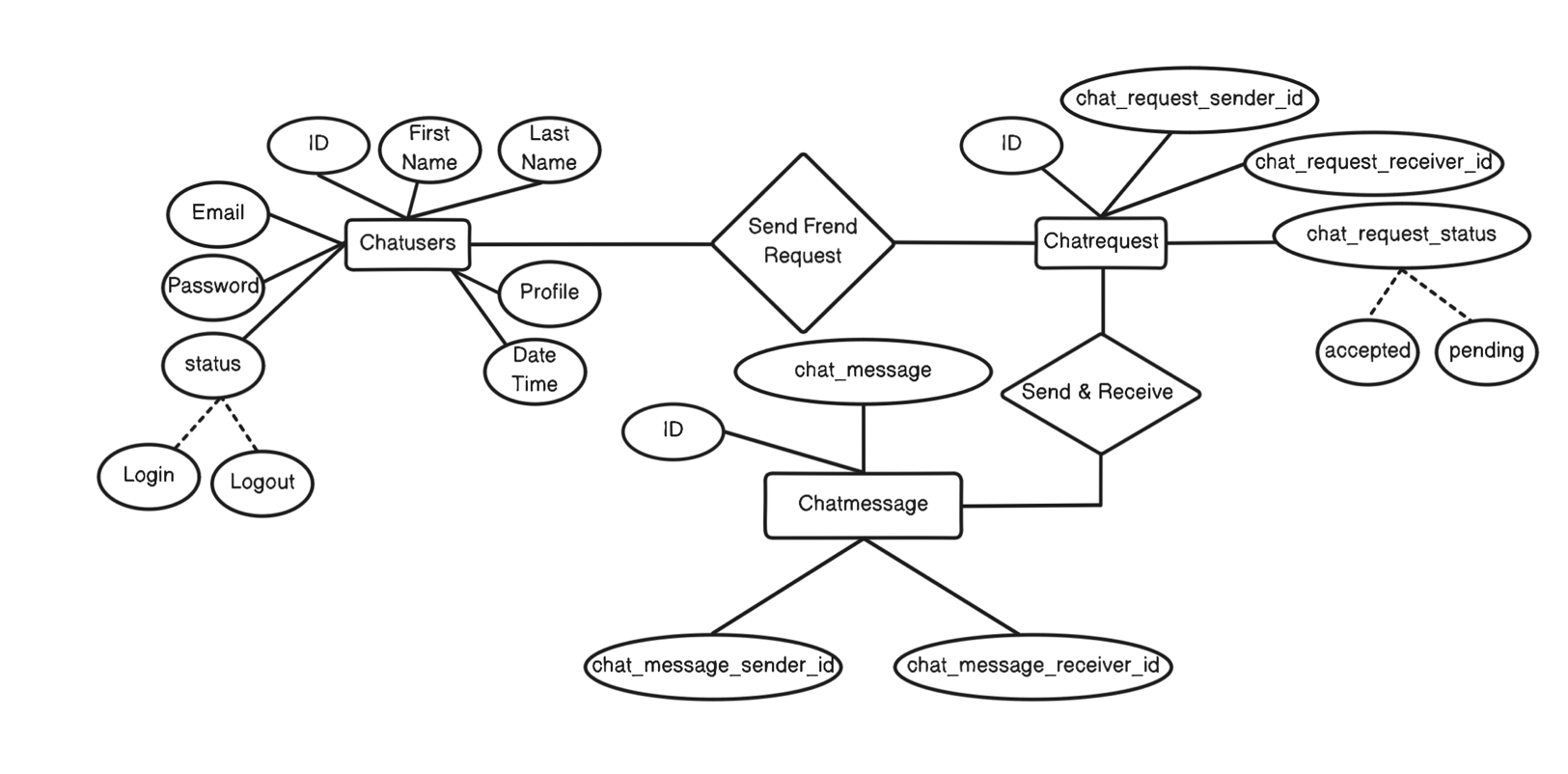
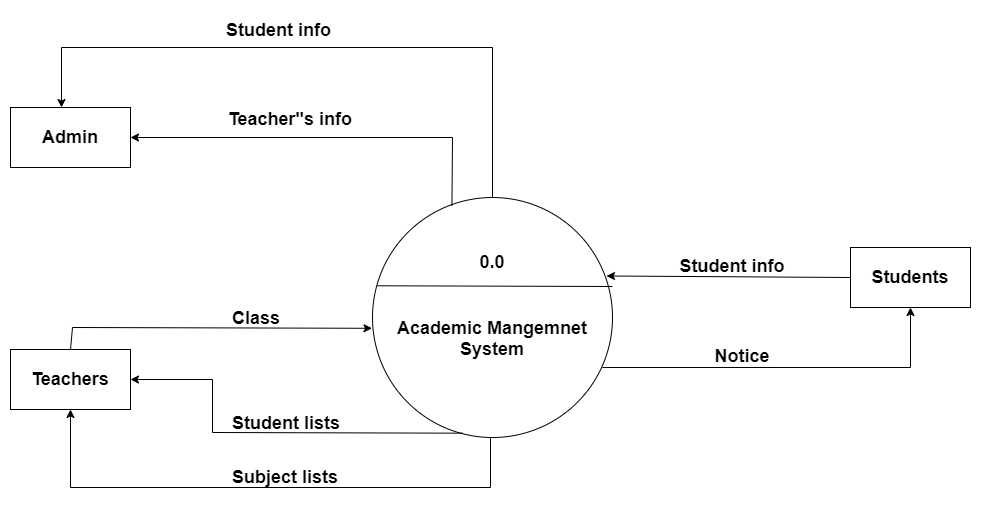
An ER diagram, also known as ER model, is a graphical representation of entities and their relationships to each other. Typically, it's used in computing regarding the organization of data within database or information system. The basic components of an ER diagram are entities, attributes, and relationship between and among those entities. The ER diagram for the AcademiaPlus is as follows.

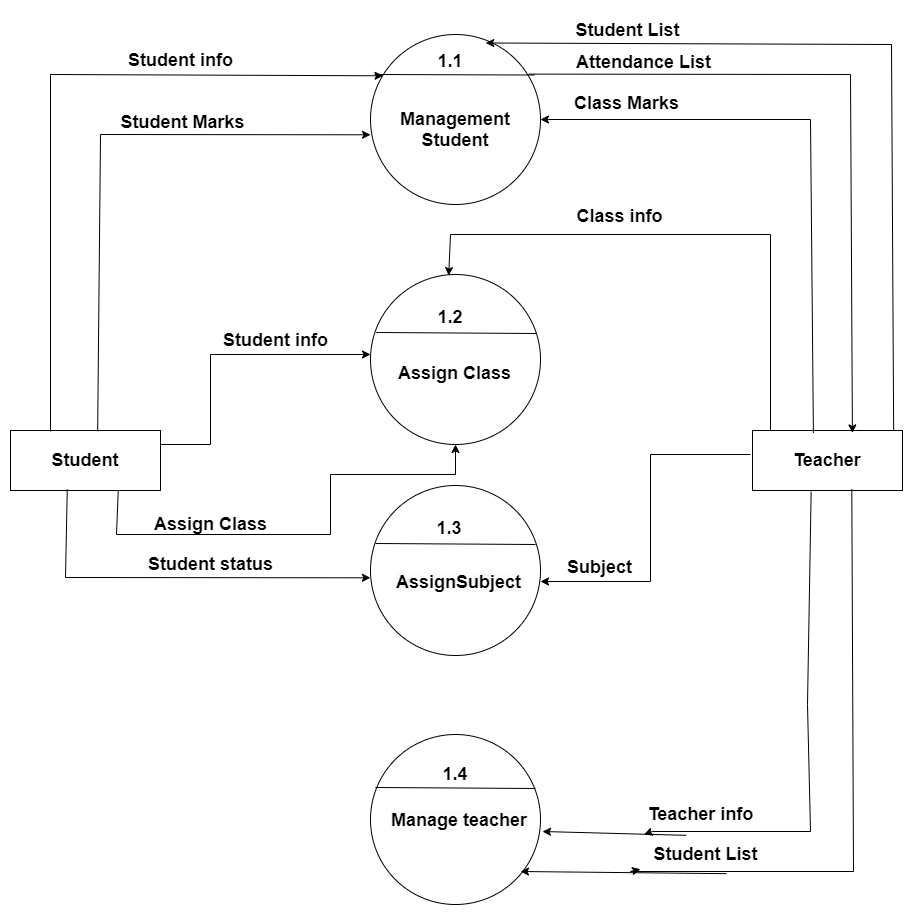
Figure 4 ER-Diagram of Academic Management System Application

## **3.1.4 Process Modelling: DFD**

1. **Zero level DFD (Context Diagram)**



1. **Level 1 Data Flow Diagram**



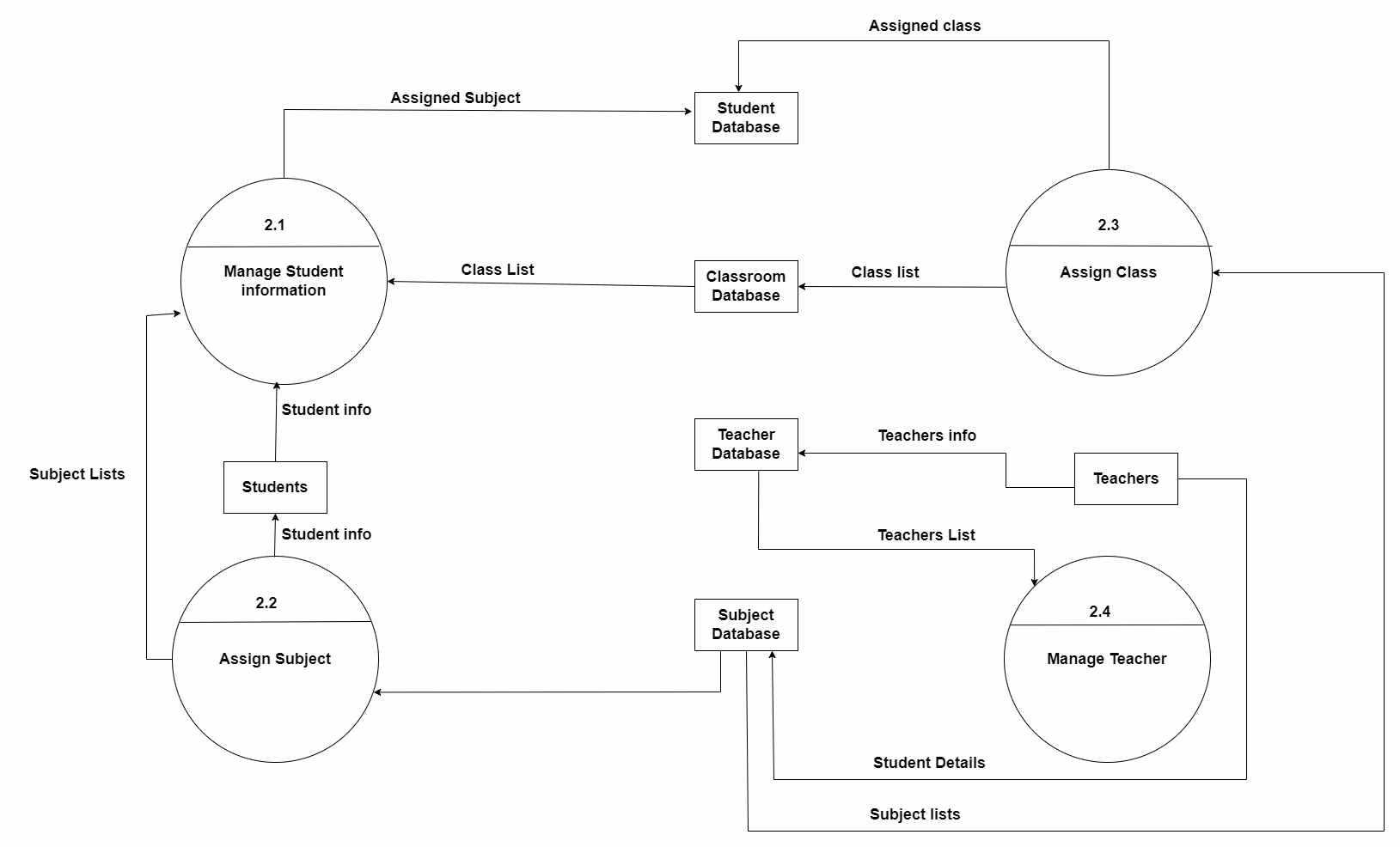
1. **Level 2 Data Flow Diagram**

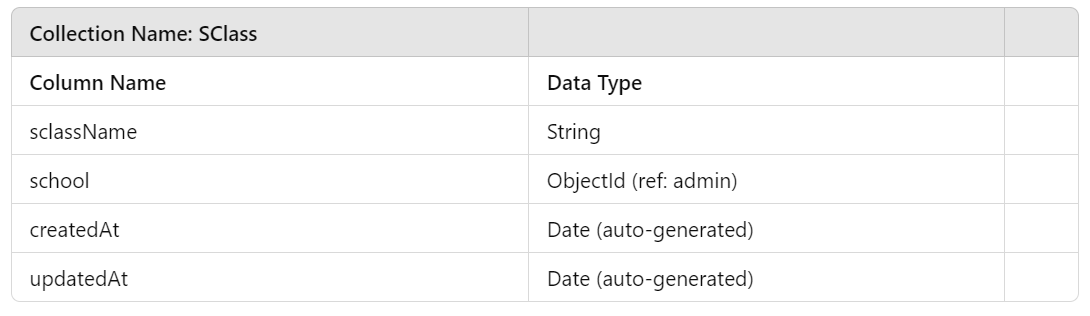
Figure 7 Second Level DFD

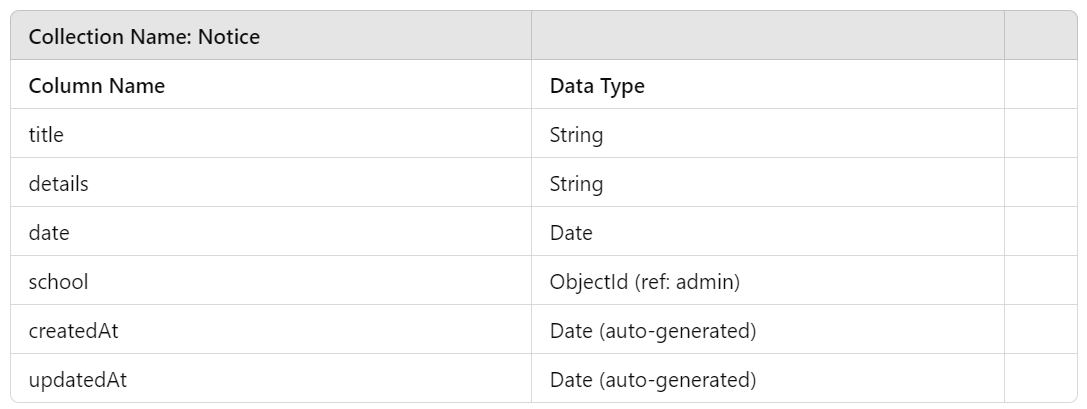
## **3.2 System Design**

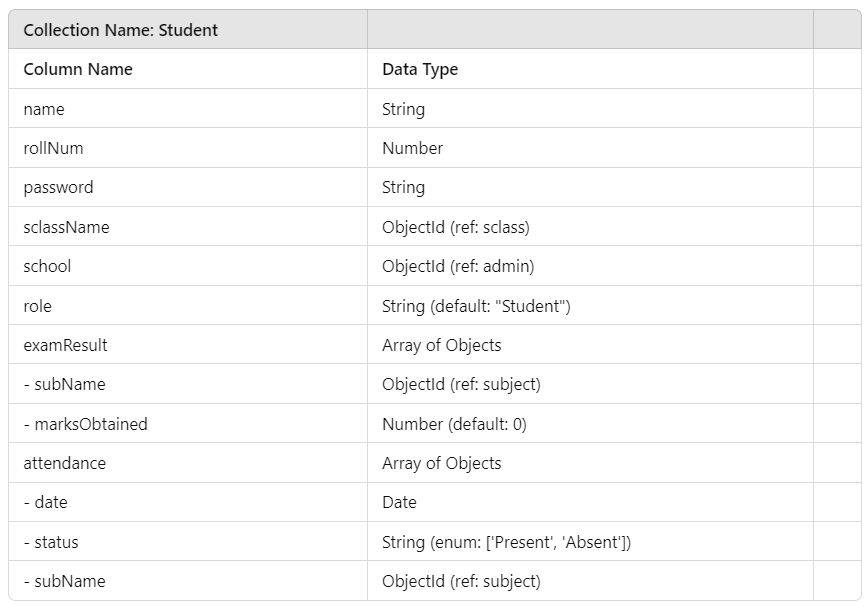
## **3.2.1 Architectural Design**

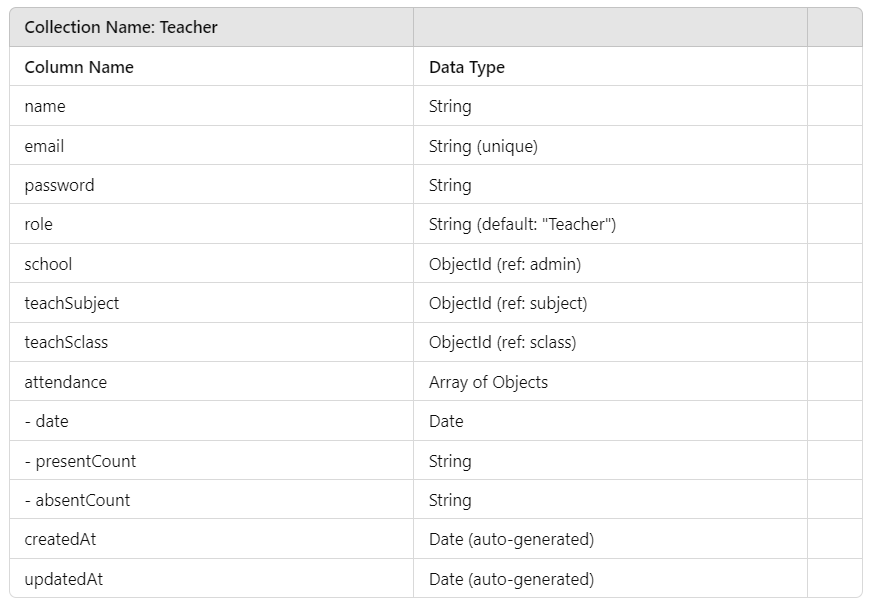
Figure 8 Architecture Design

## **3.2.2 Database Schema Design**









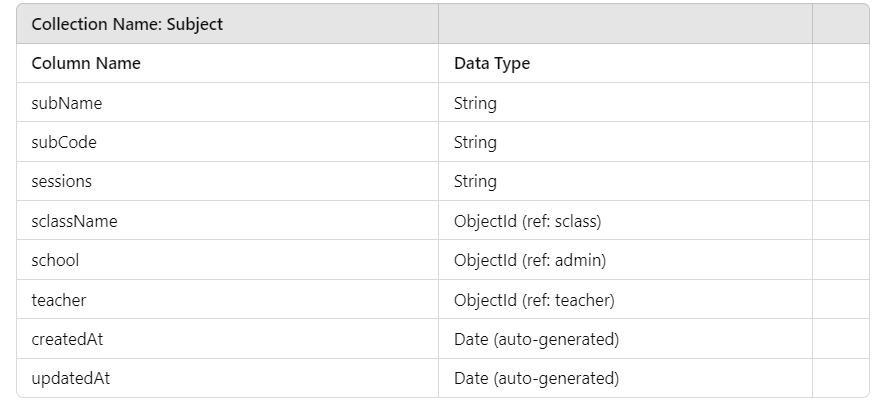


Figure 9 Database Schema Design

## **3.3 Algorithm details**

1. **Student Registration & Management:**

* Student input personal details, course enrollment information.
* Validate the inputs, check for availability in the chosen course, assign a unique student ID, and save data to the database (e.g., MongoDB)
* Success or failure message are shown along with student ID.

1. **Course & Curriculum Management:**

* Academic admin input course details, course schedules, faculty assignments.
* Verify that the course details adhere to the academic guidelines (e.g., credits, prerequisites). Assign instructors to the courses and update the schedule accordingly.
* Update course schedule and instructor assignments

1. Server Receives Message (Node.js/Express)

* The server receives the message from the client through Socket.io.
* The message is processed and broadcasted to all other connected clients in the chat room.
* The message, along with the sender’s details and timestamp, is stored in MongoDB.

1. Display Message (Client-Side JavaScript)

* Once the server broadcasts the message, all clients receive the message.
* JavaScript on the client side updates the chat window by appending the new message in real-time.

# Chapter 4: Implementation and Testing

## **4.1 Implementation**

## **4.1.1 Tools Used**

* IDE: Visual Studio Code (VS Code) – A user-friendly integrated development environment (IDE) used for coding and development tasks.
* Documentation: MS Office – Tools for creating and editing project-related documents and presentations.
* Front-end: HTML, CSS, JavaScript – Languages used for the development of the graphical user interface of the Academic Management System application.
* Back-end: Node.js, Express.js – Node.js is a JavaScript runtime used for backend operations, while Express.js is a web framework to handle server-side logic and manage communication between the front-end and database.
* Database Management: MongoDB – A database used to store user information and chat messages in a flexible, document-oriented format.
* Localhost: Node.js with Express.js – A web server framework that runs the application locally during development and testing.
* Diagramming: Eraser.io – An online diagramming tool used for creating visual representations, such as system architecture diagrams.

## **4.1.2 Implementation Details of Modules**

1. User Registration and Authentication: This module allows users to register, log in, and log out of the chat application. It securely stores user credentials, enabling each user to access their chat history and profile settings.
2. Search Functionality: The search module enables users to quickly find specific messages or conversations by searching keywords within their chat history.
3. Profile Management: Users can update their personal information, such as username and profile picture, through this module. It allows users to customize their profile and control their visibility in the chat.
4. Private Messaging (Direct Messages): This module allows users to initiate one-on-one private conversations with specific users, ensuring that messages are visible only to the participants involved.

## **4.2 Testing**

## **4.2.1 Test Cases for Unit Testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No. | Module | Test Description | Steps | Expected  Result | Result |
| TC\_01 | User  Authentication | Verify valid  user allows  successful login | Input valid  user details | User is  successfully  login | Pass |
| TC\_02 | Profile Update | Update User profile | Fill in  relevant  details and  Save  profile | Profile is  update | Pass |
| TC\_03 | Sending Messages | Verify that a user can send a message in the chat. | User types a valid message and clicks "Send." | Message appears in the chat window for all participants. | Fail |
| TC\_04 | Receiving Messages | Check if a user receives a message from another user in real time. | Another user sends a message in the chat. | Unread messages are highlighted for the user upon logging in. | Pass |

Table 1 Test Cases for Unit Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test  No | Module | Test  Description | Steps | Expected  Result | Result |
| TC\_05 | Push Notifications | Verify that users receive push notifications when they receive new messages, even when they are not actively using the app. | Send a message to a user while they are offline or not focused on the chat window. | A notification appears to alert the user of the new message. | Pass |
| TC\_06 | User Status | Verify that user’s online/offline status is updated correctly. | User logs in and logs out. | users can see the correct online/offline status of that user | Pass |
| TC\_07 | Cross-Browser Compatibility | Ensure the Academic Management System application works correctly across different web browsers (e.g., Chrome, Firefox, Safari, Edge). | Open the chat application in multiple browsers and test all major functionalities. | The application functions consistently across all browsers without UI glitches. | Pass |
| TC\_07 | Offline Messaging and Synchronization | Test if messages sent while a user is offline are delivered when they come back online. | User A sends messages while User B is offline. | User B receives all missed messages once they log back into the application. | Pass |

## **4.2.2 Test Cases for System Testing**

Table 2 Test Cases for System Testing

# Chapter 5: Conclusion and Future Recommendations

## **5.1 Conclusion**

The development of an Academic Management System using modern technologies like React, Node.js, Express, MongoDB, and JWT authentication provides an efficient, scalable, and secure solution for managing the various aspects of academic institutions. By leveraging React and Material-UI on the frontend, the system offers an intuitive and responsive user interface, ensuring an enhanced user experience for students, faculty, and administrators. The backend, built with Node.js and Express, facilitates smooth handling of data and operations through well-defined APIs, while MongoDB offers a flexible and scalable database solution to manage diverse datasets such as student records, courses, and grading. JWT authentication ensures secure access to sensitive data, enhancing the system's overall security. Together, these technologies streamline the administration of academic operations, reducing paperwork, minimizing human error, and improving efficiency.

## **5.2 Future Recommendations**

This project is currently limited to basic management of academic system. The following modules are recommended to be included in the Academic Management System in the future:

1. AI and Machine Learning Integration: Future versions of the AMS could incorporate AI-powered analytics to track student performance trends, predict outcomes, and offer personalized learning recommendations.

2. Cloud Integration: To enhance scalability and accessibility, the AMS could be deployed in a cloud environment, offering institutions the flexibility to manage resources efficiently and provide access from anywhere.

3. Mobile App Development: Developing a mobile application alongside the web platform would improve accessibility for students and faculty, allowing them to engage with the system from any device.

4. Enhanced Security Features: In addition to JWT, implementing features like multi-factor authentication (MFA) and end-to-end encryption would strengthen the system’s security, especially for handling sensitive academic data.

5. Role-based Access and Permissions: A more granular role-based access control mechanism could be developed to allow different levels of permission based on roles (e.g., administrators, teachers, students, and parents) to manage the scope of data access and operations.

By adopting these recommendations, the AMS can evolve into a more sophisticated, accessible, and future-proof system, continuing to meet the evolving demands of modern educational institutions.

# References

|  |  |
| --- | --- |
| [1] | K. Acharya, "College information management system," Tribhuvan University, March 2024. |
| [2] | N. A. A. Almotawkel , "College Management System (CMS)," March 2024. |
| [3] | S. Patil, "College ERP Using MERN Stack," May 2021. |
| [4] | K. S. Kumari, "Campus Management System," April 2024. |

# Appendices

* Dashboard (Admin)

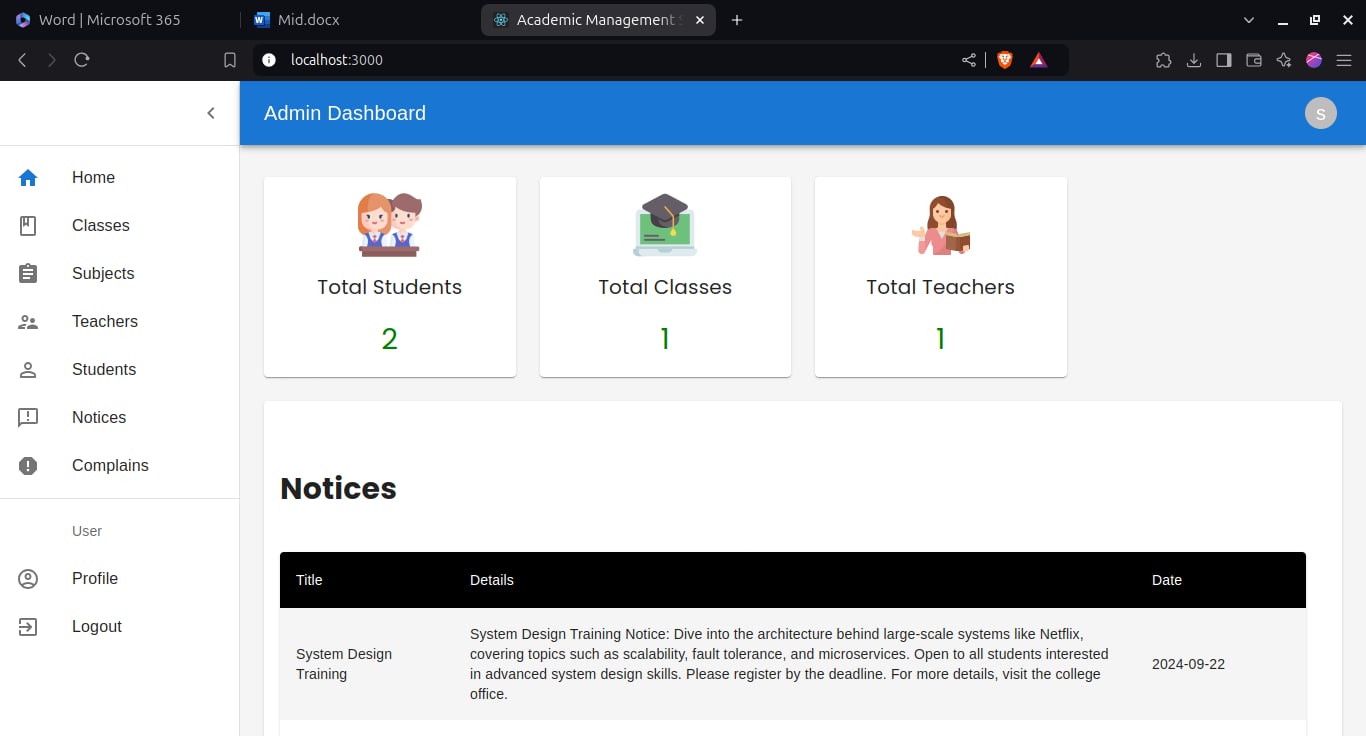


Figure 13 Index Page

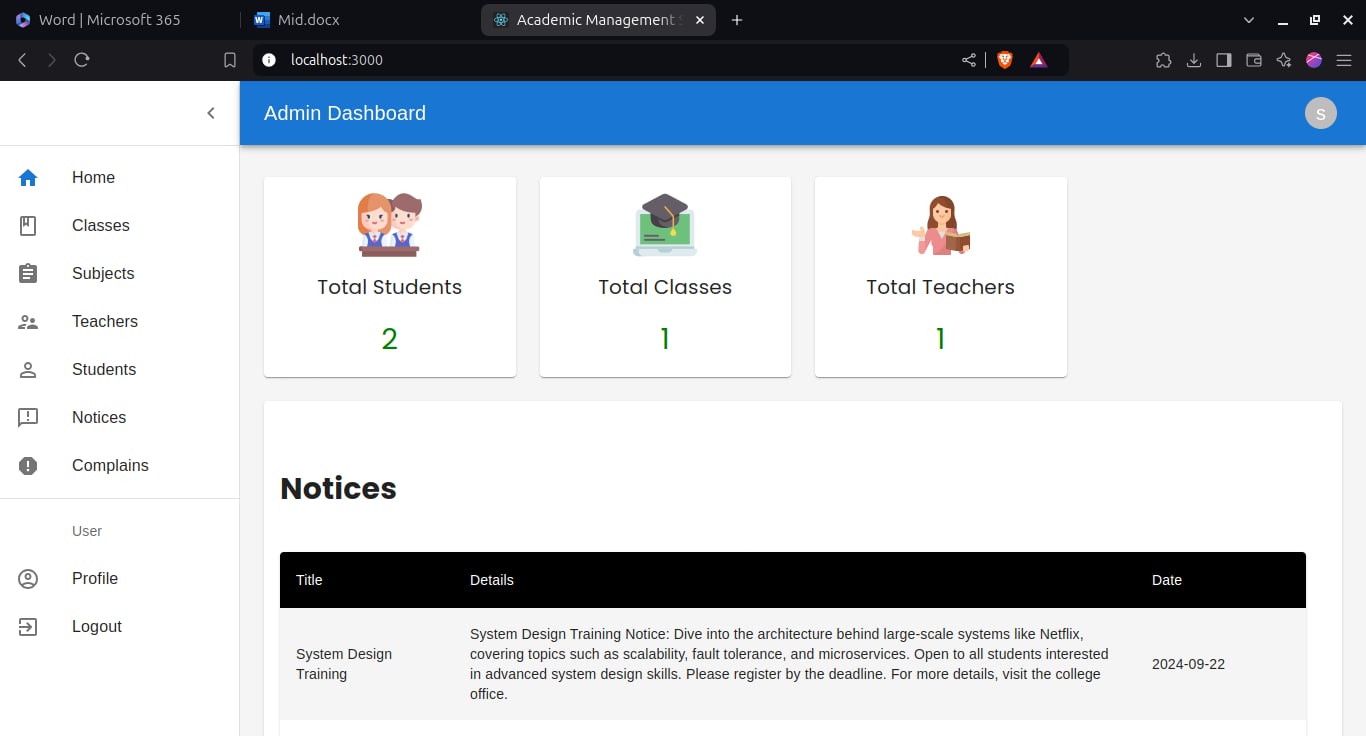
* Dashboard (Student)

Figure 14 Profile Update

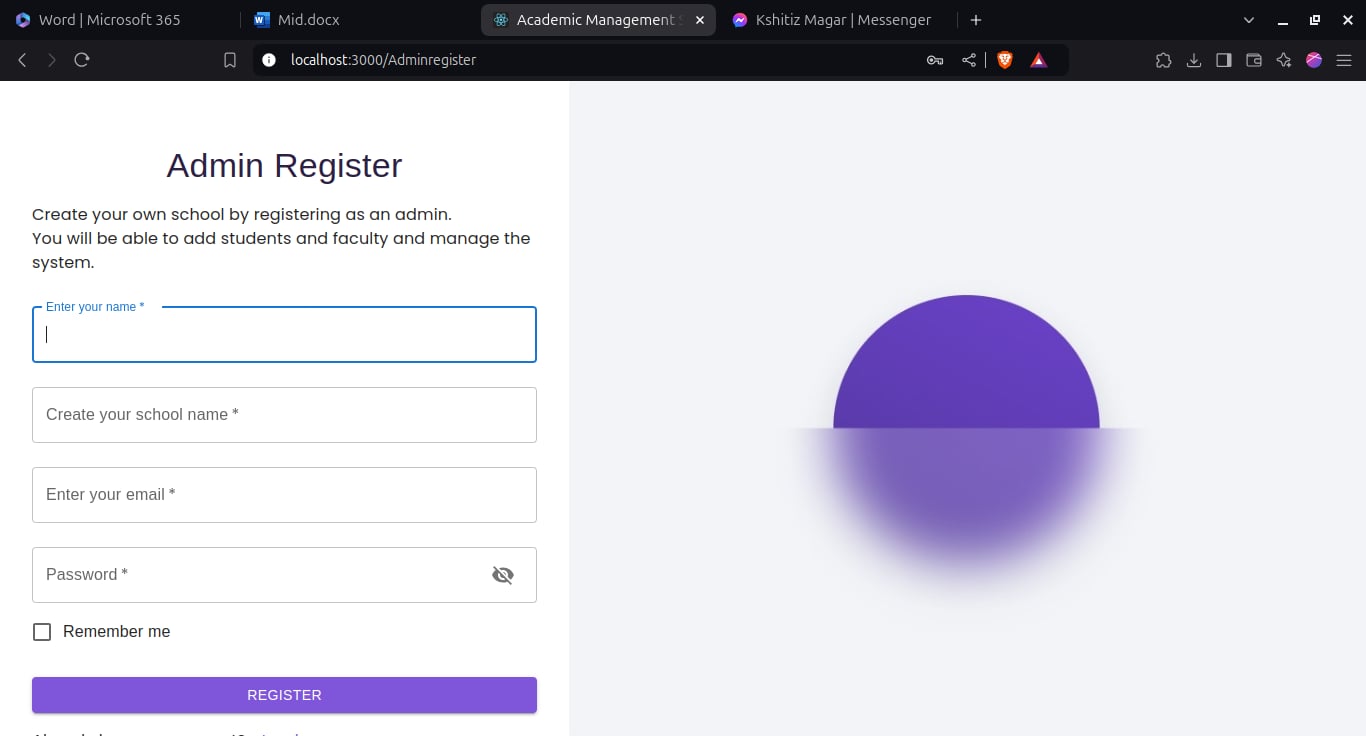
* Register

Figure 15 Register

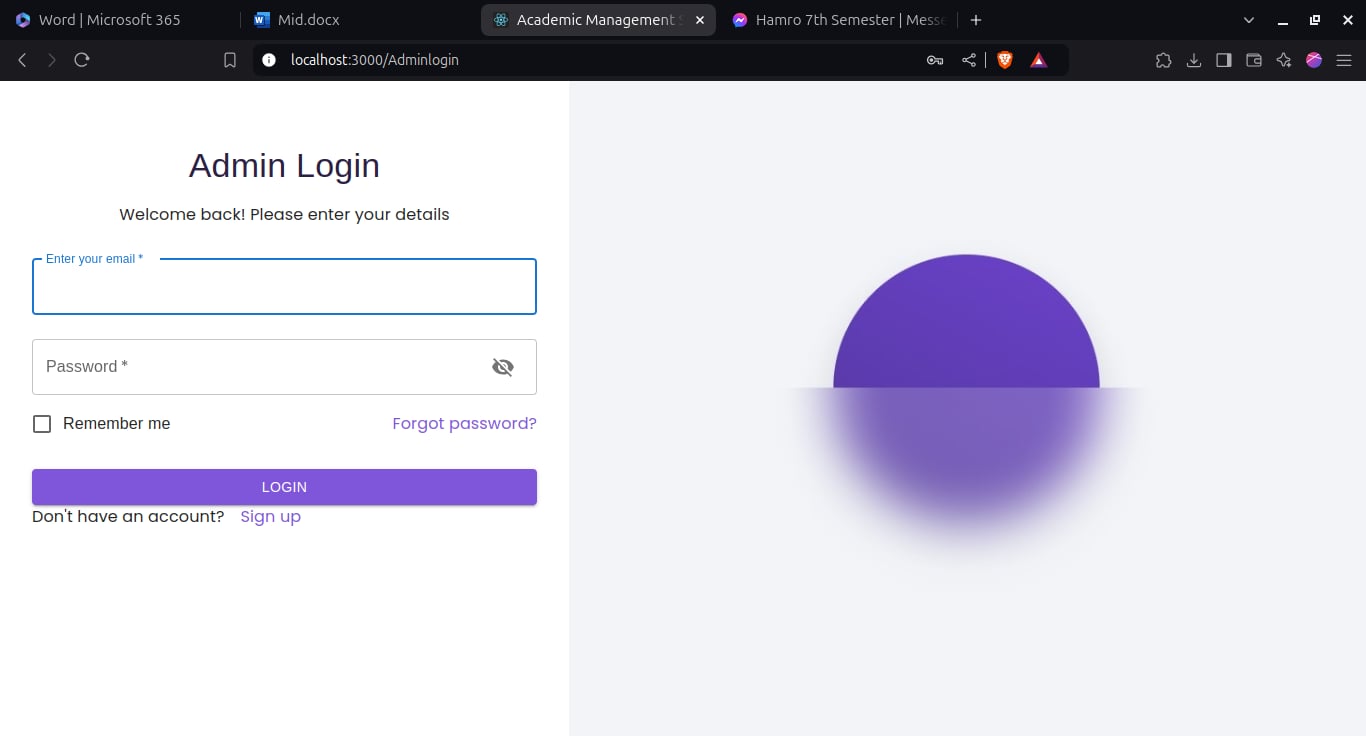
* Login

Figure 16 Login