Academic Management Platform

[1]Mrs Sonia K P, [2]Prithesh, [3]Dhanush S U, [4]Shriman Jain, [5]Mohammed Hafiz
[1] *Assistant Professor, Department of CSE (IoT & Cyber Security with Blockchain Technology), Mangalore Institute of Technology and Engineering, India.*
[2, 3, 4, 5] *Student, Department of CSE (IoT & Cyber Security with Blockchain Technology), Mangalore Institute of Technology and Engineering, India.*

Abstract: ***The Academic Management Platform (AMP) is a comprehensive web-based solution designed to enhance and streamline the management of academic activities. Initially focused on attendance and marks management, AMP has been enhanced with features such as resource sharing, chat rooms, and gamified leader boards to foster better collaboration and engagement. Built using the MERN stack, AMP addresses critical challenges such as fragmented systems, manual workflows, and scalability issues in traditional academic environments. This paper discusses the platform's architecture, key features, and the impact of its implementation on academic workflows. AMP provides a centralized system that enhances resource accessibility, promotes effective communication, and ensures real-time data updates, significantly improving the academic experience.***

Keywords: Academic Management, MERN Stack, Resource Sharing, Gamification, Real-Time Data.

1. INTRODUCTION

Efficient academic management is essential in modern educational institutions to address challenges like fragmented systems, manual workflows, and communication gaps. Traditional tools often fail to integrate functionalities such as attendance tracking, resource sharing, and performance evaluation. The Academic Management Platform (AMP) was developed to create a centralized, scalable, and secure academic management solution. This paper provides an overview of AMP's architecture and features, showcasing its potential to transform academic workflows. By leveraging the MERN stack (MongoDB, Express.js, React.js, Node.js), AMP ensures scalability, adaptability, and user engagement through tools like chat rooms and gamified leaderboards. Unlike fragmented systems or physical records, these platforms offer advanced security measures like encryption and role-based access, ensuring the safety and integrity of sensitive academic data.

2. LITERATURE SURVEY

2.1 Existing Systems

2.1.1 Existing web-based systems:

Falebita (2022) developed a Secure Web-Based Student Information Management System, addressing inefficiencies in traditional academic management processes. The system implemented features such as role-based user management, secure data handling through encryption, and protection against SQL injection and XSS vulnerabilities using Laravel. Despite its security robustness, the study identified the need for mobile integration and enhanced data synchronization for broader accessibility [1].

Walia and Gill (2014) proposed a framework for web-based student record management systems using PHP, emphasizing simplicity, modularity, and secure data transfer mechanisms. However, they noted limitations in scalability and performance when handling large datasets, particularly in multi-institutional environments. These challenges underscore the importance of modern frameworks for large-scale deployments [2].

Sun and Chen (2016) explored the design of a university student management system based on a web platform. Their system streamlined workflows with centralized data management and responsive user interfaces, enabling faster information retrieval for students and administrators. However, the system lacked predictive analytics and data-driven decision-making capabilities, which are increasingly critical in modern academic platforms [3].

2.1.2 Mobile and Multi-Platform Innovations:

Kadam et al. (2017) introduced a mobile web-based Android application for college management systems, addressing the growing demand for mobile-first solutions. The application allowed students to access academic resources, attendance, and grades in

real time. Challenges included data synchronization across devices and the lack of offline accessibility, indicating the need for cloud-based architectures in future systems [4]. Gautam and Shrestha (2012) proposed Web-Enabled Campus-SIA, an integrated system automating campus workflows such as attendance tracking and performance reporting. While it effectively centralized administrative processes, the solution faced limitations in scalability and adaptability to larger institutions with diverse user needs [5].

2.1.3 Big Data and Analytics in Academic Systems:

Brozina et al (2019) demonstrated the transformative role of analytics in learning management systems, particularly for understanding student engagement and predicting academic outcomes. Using big data techniques, the study tracked attendance, grades, and learning behaviours to provide actionable insights for educators and administrators. However, integrating such tools into traditional systems remains a complex challenge, requiring robust infrastructure and seamless data synchronization [6].

2.1.4 Security and Privacy Concerns:

Falebita (2022) emphasized the critical importance of security in academic management systems. The study implemented encrypted storage, hashed passwords, and role-based access control to protect sensitive data. However, it recommended enhancements like multi-factor authentication and advanced logging mechanisms to further strengthen data integrity [1]. Walia and Gill (2014) also discussed security concerns, particularly the vulnerabilities in PHP-based systems. They stressed the importance of securing data transfers and user authentication mechanisms to prevent breaches, a critical consideration in modern academic platforms [2].

3. METHODOLOGY

3.1 Architecture

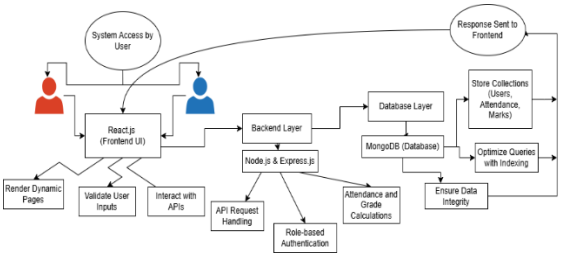

Fig 3.1 Architecture Overview

AMP follows a three-tier architecture:

i. Frontend: Built using React.js for dynamic and responsive user interfaces.
ii. Backend: Node.js and Express.js handle data processing, authentication, and API management.
iii. Database: MongoDB stores and retrieves academic data such as attendance, marks, and resources.

3.2 Features

i. Admin Tools: User and subject management, report generation.
ii. Faculty Tools: Attendance tracking, marks entry, resource uploads.
iii. Student Tools: Access to attendance and marks, resource downloads, gamified leaderboards.

3.3 Use Case Diagrams

The Use Case Diagram provides a high-level representation of user interactions with the system. It defines how the platform serves its users: Admins, Faculty, and Students.

3.3.1 Admins Use Cases:

i. Add/Remove Users.
ii. Add/Remove Subjects.

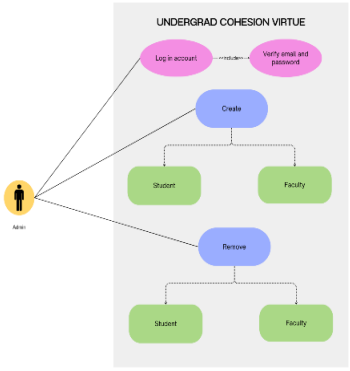

Fig 3.3.1 Admin Use Case Diagram

3.3.2 Faculty Use Cases:

i. Update Attendance
ii. Upload Marks
iii. Share Resources
iv. Chat Room

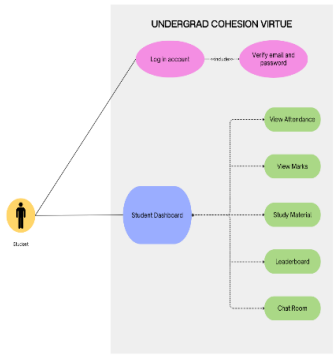

Fig 3.3.2 Faculty Use Case Diagram

3.3.2 Faculty Use Cases:

i. View Attendance and Marks
ii. Access Uploaded Resources.
iii. Communicate via Chat Room.

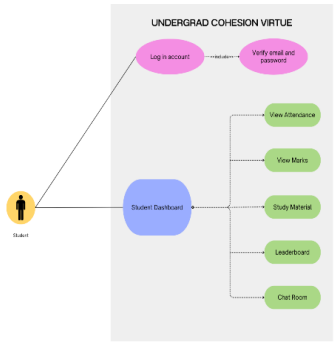

Fig 3.3.3 Student Use Case Diagram

4. IMPLEMENTATION

The implementation phase of the Academic Management Platform is built on the robust MERN (MongoDB, Express.js, React.js, Node.js) stack. This stack ensures high performance, scalability, and a seamless user experience. Below is an in-depth discussion of the technical stack, frontend, backend, and database aspects of AMP, elaborating on their roles in the system architecture and functionality.

4.1 Technical Stack: MERN

The MERN stack integrates four key technologies, each contributing uniquely to the platform's architecture and capabilities:

i. MongoDB, a NoSQL, document-based database system, stores data in BSON (Binary JSON) format, making it ideal for hierarchical and semi-structured data such as student records, attendance logs, and academic resources. Horizontal scalability through sharding allows the database to handle massive datasets efficiently. CRUD (Create, Read, Update, Delete) operations are executed via the backend using MongoDB's flexible schema, reducing the need for database migrations during iterative development.
ii. Express.js simplifies the creation of web applications and APIs. Modular middleware pipelines are implemented for request validation, error handling, and logging. RESTful endpoints handle requests from the front, such as /api/attendance or /api/marks. Middleware like Helmet.js secures HTTP headers, and Express Validator ensures input sanitization.
iii. React.js provides the user interface layer of AMP. Its component-based architecture ensures modularity and maintainability, while the virtual DOM improves rendering performance for smooth transitions and real-time data updates. Libraries like Redux or Context API manage application state, ensuring synchronization across components.
iv. Node.js provides a runtime for executing JavaScript code on the server side. The event-driven, non-blocking I/O model allows Node.js to handle multiple requests efficiently, making it suitable for high-traffic applications. Its npm (Node Package Manager) offers a vast library of modules, expediting the development process.

4.2 Frontend

The frontend of AMP is developed using React.js in conjunction with Material-UI (MUI), a robust UI library for React. The focus of the frontend is on delivering a seamless and responsive user experience for desktop and tablet users, ensuring dynamic functionality through modular and reusable components. Although the frontend does not follow a mobile-first approach, it leverages Material-UI's Grid System and responsive utilities to ensure compatibility across desktop and tablet screens. The grid system divides the screen into columns, allowing flexible and precise layouts for different screen resolutions. Media queries built into Material-UI’s design ensure that layouts adjust to changes in window size. Navigation menus are tailored for desktop environments, with hover effects and expanded menus that are easy to use with a mouse pointer. Material UI's makestyle and sx props provide responsive styles for components, adjusting margins, paddings, and font sizes to suit larger screens.

React's component-based architecture ensures a modular design, where individual components encapsulate their own logic, styles, and interactions. Examples include LoginForm, AdminDashboard, FacultyAttendanceTable, and StudentLeaderboard. Each component is reusable, improving maintainability and scalability. Shared components like buttons, modals, and cards are styled with Material-UI's design tokens for consistency. Parent components pass data and functions to child components via props, maintaining loose coupling. For instance, the AdminDashboard passes user data to UserManagementTable for rendering, ensuring clean separation of concerns.

The frontend utilizes React Router for efficient and dynamic navigation across various sections of the platform. Each page corresponds to a route, such as /dashboard for dashboards, /attendance for attendance management, and /marks for grades and performance. Authentication and role-based access are implemented to restrict users from accessing unauthorized sections. Parameters like subjectId or studentId in the URL enable dynamic rendering of specific data, such as /faculty/subjects/1234/attendance displaying the attendance logs for a subject with ID 1234. The frontend communicates with the backend via RESTful APIs, handling both synchronous and asynchronous data exchange. The Axios library is used to send HTTP requests, ensuring seamless integration with backend endpoints. For instance, GET /api/attendance retrieves attendance data for display in the faculty dashboard, while POST /api/marks submits updated grades for a specific subject.

4.3 Backend

The backend of the Academic Management Platform (AMP) is the core layer connecting the frontend interface with the database, managing data flow, and enforcing business rules. It is developed using Node.js and Express.js, offering high performance and scalability while maintaining robust security through authentication and role-based access mechanisms.

The backend provides RESTful APIs that expose endpoints for the platform's various operations. These APIs enable seamless communication between the frontend and the database, ensuring that the system remains responsive and dynamic. For instance, the GET /api/attendance/:studentId API fetches attendance records for a specific student. Middleware ensures input validation, such as checking for valid student IDs, before processing requests. Token-based authentication using JSON Web Tokens (JWT) ensures secure access to protected resources. Each token includes user credentials and an expiration time, stored in HTTP-only cookies to prevent unauthorized access. Role-based access control (RBAC) ensures that users can only perform actions permitted for their role. The backend also implements centralized error handling to provide meaningful error messages to users while logging errors for developers.

4.3.1 Error Handling:

A robust error-handling mechanism ensures that the platform provides meaningful feedback to users and maintains consistency in API responses. It helps in identifying and resolving issues efficiently, minimizing the impact on user experience. Proper error logging enables developers to track and debug problems effectively, ensuring the system's reliability and stability over time. Express's error-handling middleware intercepts errors from any part of the application, categorizing them into client-side errors like 400 Bad Request or 401 Unauthorized, and server-side errors like 500 Internal Server Error. For example, if a faculty member sends an attendance update without a valid subjectId, the API responds with a descriptive error message such as "400 Bad Request: Missing or Invalid Subject ID." Similarly, a database failure during a MongoDB query results in a 500 Internal Server Error, with details logged for developer review. The error handler ensures consistent responses while keeping the system operational for unaffected users. This robust mechanism underpins the platform's ability to provide reliable and efficient operations, enhancing both user experience and system stability.

4.4 Database

The database layer of AMP is powered by MongoDB, a NoSQL database designed to handle large datasets with flexibility and scalability. The database schema is organized into multiple collections, each serving a distinct function. These collections ensure that AMP manages academic data efficiently and supports the platform's core functionalities, such as user management, attendance tracking, marks management, subject handling, and file storage.

4.4.1 Admin Collection:

The Admin collection stores information about the administrators who manage the platform. Admins typically have full access to features such as adding or removing users. Each admin record includes a unique identifier (adminId), their full name, email address, and a hashed password for secure storage using techniques like bcrypt. Additionally, the admin's role is fixed as "admin," and their permissions, such as adding or deleting users, are defined to specify the actions they can perform.

4.4.2 Student Collection:

The Student collection contains data about all students using the platform, enabling the system to manage student records, track their academic performance, and generate personalized views of attendance and marks. Each student record includes a unique identifier (studentId), the student's full name, email for login and notifications, and a securely hashed password. The collection also maintains references to the subjects the student is enrolled in, their attendance logs, and marks obtained in various assessments.

4.4.3 Attendance Collection:

The Attendance collection records the attendance of students for each subject, providing essential data for tracking academic progress. Each record is uniquely identified by an attendanceId and includes references to the studentId and subjectId. The collection also stores the date of attendance and the status, indicating whether the student was "Present" or "Absent."

4.4.4 Marks Collection:

The Marks collection stores the academic performance of students across various assessments such as quizzes, assignments, and exams. Each record has a unique identifier (marksId) and references to the studentId and subjectId. It also specifies the type of assessment (e.g., "Quiz," "Assignment") and the numeric score achieved by the student.

4.4.5 Subject Collection:

The Subject collection manages information about the courses or subjects offered by the institution. Each subject is identified by a unique subjectId and includes the name of the subject, a reference to the faculty member teaching it, and an array of studentIds representing the enrolled students.

4.4.6 File Collection:

The File collection stores academic resources uploaded by faculty members, such as reading materials. Each resource is identified by a unique resourceId and includes the file name, URL or file path (often on cloud storage), the faculty member who uploaded the resource, and the upload date. This collection ensures that students have easy access to relevant academic resources.

Each feature in the Academic Management Platform is designed to address the specific needs of its users, ensuring streamlined workflows for admins and faculty while empowering students to stay informed and engaged in their academic journey. This role-based approach enhances productivity and collaboration across the platform.

5. TESTING

The testing phase ensures the platform's reliability, functionality, and performance. Testing was conducted across all features to verify their accuracy and seamless integration, focusing on the platform's key components for admins, faculty, and students.

To ensure comprehensive validation of the platform, several testing methodologies were applied. Unit testing involved testing individual modules such as login validation, attendance updates, and mark entry in isolation. This ensured that each component functioned as expected before integration. Integration testing examined the interaction between the front end (React.js), back end (Node.js), and database (MongoDB). For instance, the communication between the attendance form and the database was verified to ensure proper data flow. System testing involved end-to-end testing to validate the overall functionality. Role-based workflows such as admin actions (adding students), faculty actions (marking attendance), and student actions (viewing marks) were thoroughly tested. Test cases were designed to verify the functionality and accuracy of key features in the Academic Management Platform, ensuring seamless performance across all user roles.

For admin features, the following scenarios were tested: adding students, which involved verifying that admins could add new student profiles with valid details and ensuring the records appeared in the

database and dashboards; removing students, which tested the deletion of student records and their removal from all associated dashboards; adding subjects, ensuring that new subjects were correctly added and visible in relevant dashboards; removing subjects, validating the deletion of subjects and clearing all references; and viewing student and subject lists, confirming that the displayed data matched the database entries with accurate filtering and sorting.

For faculty features, several functionalities were tested. Updating attendance was tested to ensure that faculty could mark students as "Present" or "Absent," save changes, and verify the updated records in the database and dashboards. Updating marks involved entering grades for students, saving the changes, and confirming their accuracy in the database and dashboards. The resource upload feature was tested by verifying that uploaded files appeared in the student dashboard, were downloadable, and maintained their integrity. Chatroom access was also tested for real-time communication, ensuring instant message delivery and accurate message logging.

For student features, key functionalities were validated. Viewing attendance involved ensuring that students could access accurate and real-time updates of their attendance records. Viewing marks was tested to confirm that students could view their grades without discrepancies, aligned with faculty inputs. Downloading resources was validated by confirming that files uploaded by faculty were easily accessible and intact. The leaderboard ranking feature was tested to ensure that rankings reflected real-time data and matched database entries. Chatroom access was also verified to ensure that students could send and receive messages instantly with consistent message history.

6. TESTING AND SNAPSHOTS

The testing phase validated the functionality and reliability of every feature in the Academic Management Platform. Each role-specific feature, from admin user management to faculty attendance updates and student leaderboard views, was thoroughly tested to ensure accuracy and seamless integration. With all identified issues resolved, the platform is now ready for deployment in a real-world academic environment. This chapter discusses the performance metrics and provides visual snapshots of the key dashboards in the Academic Management Platform. The performance metrics demonstrate how the platform meets its goals of efficiency, real-time data handling, and user-specific features. The snapshots showcase the Admin, Faculty, Student, and Home Page Dashboards, providing a clear picture of how AMP functions across various roles.

6.1 Performance Metrics

6.1.1 Response Time Under Two Seconds:

One of the primary design goals for AMP is to ensure that the platform responds to user actions (e.g., login, fetching data, updating attendance) within two seconds. This response time is considered optimal for maintaining a smooth user experience and keeping users engaged with the platform. A response time of over two seconds can lead to user frustration and abandonment of the platform. Research shows that users expect instant feedback, with three seconds being the upper threshold for web application performance.

6.1.2 Backend Optimization:

The platform leverages Node.js for fast asynchronous operations. All major backend requests are optimized using non-blocking I/O to ensure that multiple requests are handled simultaneously without slowing down the platform.

6.1.3 Database Queries:

MongoDB is optimized for high-performance queries, including indexing on fields like studentId, subjectId, and attendanceDate, which allows for quick retrieval of data even when the database scales up with more students and subjects.

6.1.4 Frontend Efficiency:

The use of React.js ensures that the interface remains highly responsive. React's virtual DOM and efficient re-rendering mechanics prevent unnecessary page reloads, maintaining a fast, interactive user experience.

6.1.5 Real-Time Updates for Data Handling:

The platform is designed to handle real-time data updates, meaning that actions performed by one user (e.g., faculty marking attendance or entering grades) are reflected instantly across the system and visible to other relevant users (e.g., students and admins). Real-time data handling is crucial for a dynamic

academic environment where instant updates to attendance or resource uploads are required. This feature eliminates delays and ensures that all users are immediately aware of changes.

6.1.6 WebSockets or Long Polling:

These technologies maintain an open connection between the frontend and backend, allowing real-time communication between the client and server. When a change occurs in the backend (e.g., faculty marks attendance), the change is pushed to the frontend instantly.

6.1.7 MongoDB Change Streams:

This feature allows AMP to monitor changes in the database (e.g., updates in attendance records or marks) and notify connected clients (students or faculty) of these changes in real-time.

Example: When a faculty member updates attendance in the system, the change is pushed to all relevant student dashboards immediately, reflecting their new attendance status without requiring a page reload.

6.2 Snapshots

6.2.1 Home Page Snapshot:

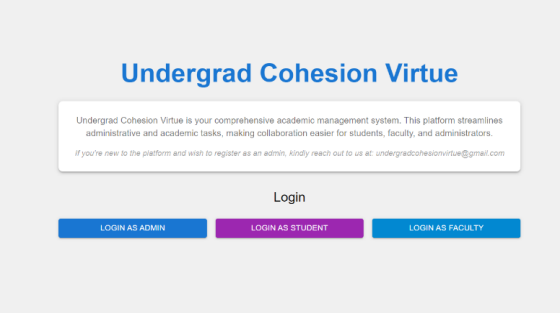

Fig 6.2.1 Home Page

The Home Page serves as the starting point for users, where they can select their role and log in to access the relevant dashboard. It provides clear directions to ensure users are guided through the login and role selection process.

Key Features:

i. Role Selection: Users are prompted to select their role (Faculty, Student) to ensure they are directed to the correct login screen and dashboard.
ii. Login/Registration: Provides login fields for returning users and registration options for new users, ensuring easy access to the platform.
iii. Navigation: Simple links guide users to their respective dashboard based on their role.
iv. Snapshot Features:
v. Prominent login and registration buttons for seamless access.
vi. A clean, intuitive layout that ensures users can quickly identify their role and proceed to the appropriate section.

6.2.2 Admin Dashboard Snapshot:

The Admin Dashboard offers admins a comprehensive view of platform management.

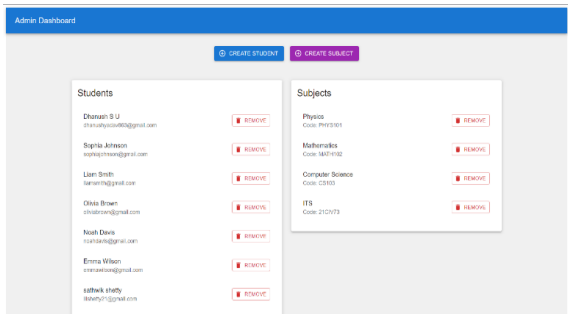

Fig 6.2.2 Admin Dashboard

Key Features:

i. Add/Remove Students: Admins can easily add new students by entering their details and assigning them to subjects. They can also remove students from the system as needed.
ii. Add/Remove Subjects: Admins have the ability to create new subjects, assign them to faculty, or remove them from the system entirely.
iii. Snapshot Features:
iv. A user-friendly dashboard layout with tables listing all students and subjects.
v. Action buttons for adding/removing students and subjects.

6.2.3 Faculty Dashboard Snapshot:

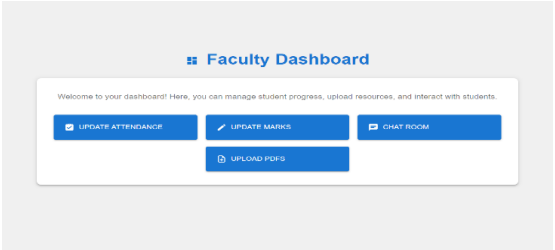

Fig 6.2.3 Faculty Dashboard

The Faculty Dashboard provides faculty members with the tools they need to manage their subjects, update attendance, enter grades, and share resources.

Key Features:

i. Attendance Management: Faculty can mark attendance for students in their assigned subjects, ensuring real-time updates on student presence.

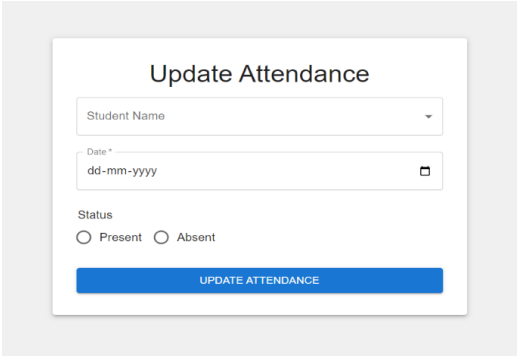

Fig 6.2.3 (i) Attendance Update Feature

ii. Marks Entry: Faculty can enter grades for various assessments, including quizzes, assignments, and exams. They can modify or update grades as needed.

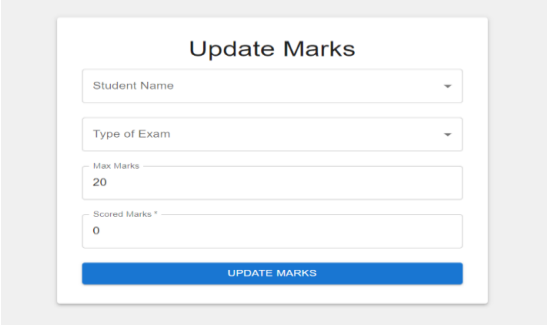

Fig 6.2.3 (ii) Marks Update Feature

iii. Resource Upload: Faculty can upload study materials (e.g., lecture slides, assignments) in PDF or other formats, which students can then download.

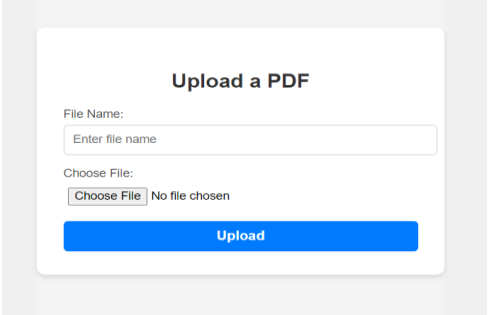

Fig 6.2.3 (iii) Resource Upload Feature

iv. Communication: Faculty members can communicate with students via integrated chat rooms.

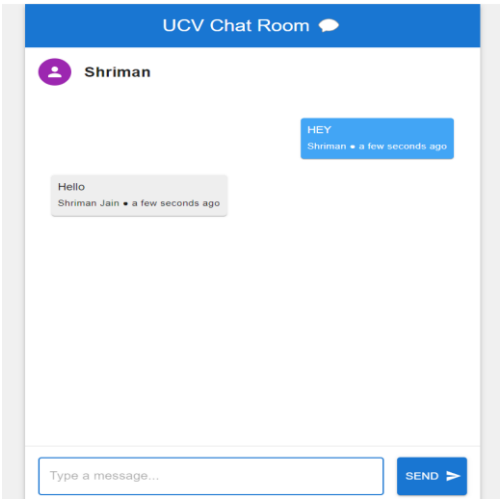

Fig 6.2.3 (iv) Chat Room Feature

Snapshot Features:

i. Displays a list of subjects with options for marking attendance and entering grades.
ii. A section for uploading academic resources, with drag-and-drop functionality for easy file uploads.
iii. Real-time updates show students' attendance and grade status immediately after being entered.

6.2.4 Student Dashboard Snapshot:

The Student Dashboard provides students with a centralized place to track their academic progress, view attendance, check marks, access resources, participate in leaderboard rankings, and communicate through chat rooms. It is designed to offer a seamless and intuitive interface, ensuring students can easily navigate between different features. The dashboard visually represents attendance and marks using charts and graphs, allowing students to quickly assess their performance. Resources uploaded by faculty, such as lecture notes and assignments, are readily accessible through a dedicated section, ensuring students have the tools they need for academic success.

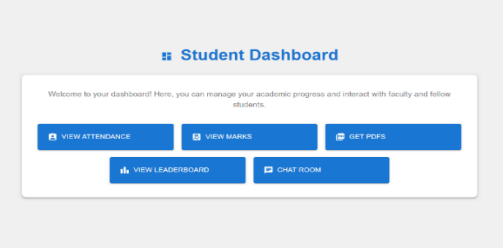

Fig 6.2.4 Student Dashboard

Key Features:

i. View Attendance: Students can check their attendance records for each subject and monitor their overall attendance percentage.

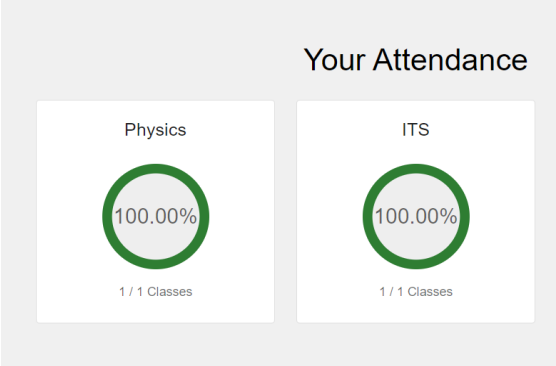

Fig 6.2.4 (i) View Attendance Feature

ii. View Marks: Students can view their grades for quizzes, assignments, and exams in real-time.

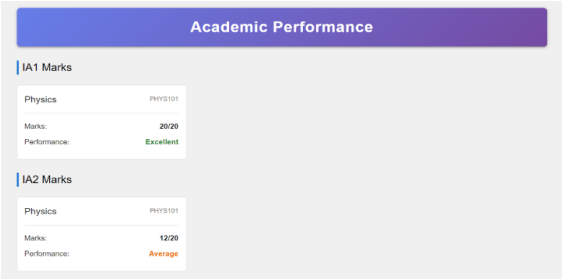

Fig 6.2.4 (ii) View Marks Feature

iii. Get PDFs: Students have access to all uploaded resources (lecture slides, assignments, etc.) and can download them for offline viewing.

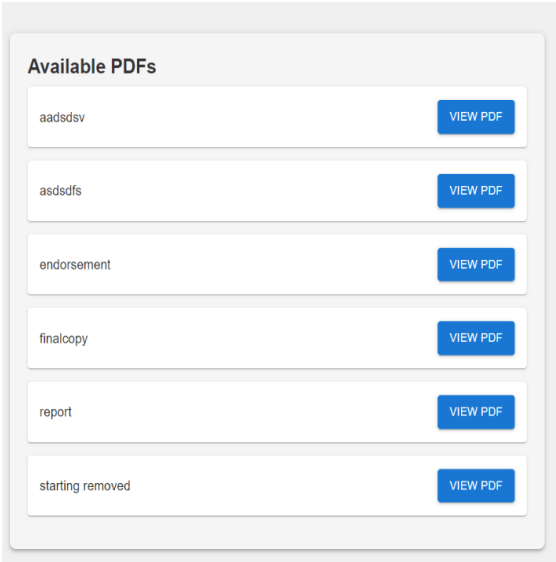

Fig 6.2.4 (iii) View Resource Feature

iv. Leaderboard: A ranking system allows students to see how they are performing in comparison to their peers.

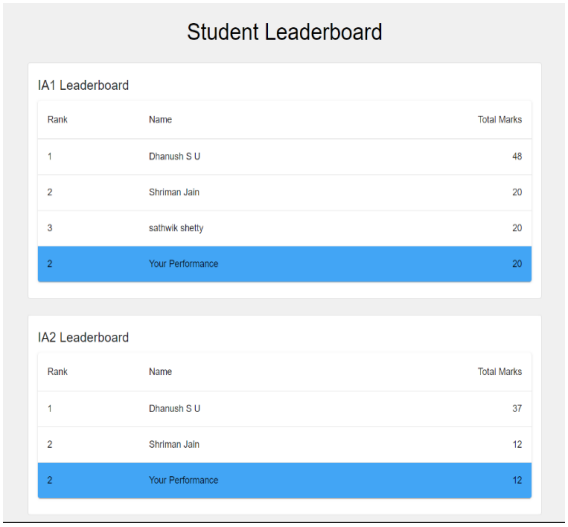

Fig 6.2.4 (iv) Leaderboard Feature

v. Chatroom Access: Students can engage in real-time communication with faculty and peers.

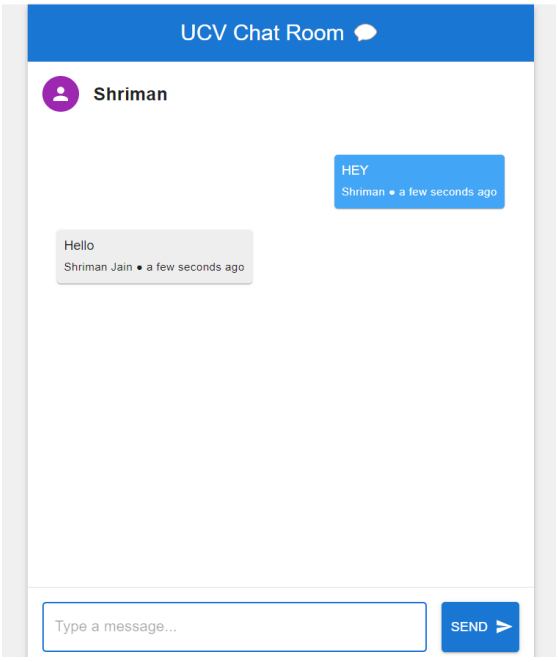

Fig 6.2.4 (v) Chat Room Feature

Snapshot Features:

i. The dashboard includes attendance progress bars and marks charts for easy tracking.

ii. A Leaderboard that shows student rankings based on marks, encouraging competition and motivation.

iii. Quick links to download resources, with a scrollable list for easy navigation.

7. CONCLUSION

The Academic Management Platform (AMP) addresses core challenges faced by educational institutions. By centralizing academic data and fostering communication, it enhances institutional efficiency. Its role-based features ensure the needs of admins, faculty, and students are met seamlessly.

For admins, AMP offers a simplified workflow for managing students and subjects, reducing administrative burden and improving accuracy in data handling. The ability to view detailed reports and perform key actions like adding or removing users and subjects ensures a high degree of control and adaptability.

For faculty, AMP provides an efficient mechanism to track student attendance, update grades, share resources, and engage in real-time communication with students. These tools enable faculty to focus on teaching quality while maintaining administrative records.

Students, the primary beneficiaries of AMP, gain access to comprehensive academic insights, including attendance and performance data, which empower them to track their progress. Features like resource access, leaderboards, and chatrooms foster collaboration and active learning.

The platform's implementation using the MERN stack ensures scalability, responsiveness, and customization. MongoDB for data storage, React.js for a dynamic frontend, and Express.js with Node.js for backend processing create a robust and adaptable system. In conclusion, AMP has transformed academic management systems and set the stage for further technological advancements, offering high adaptability and performance in modern education.

REFERENCES

[1] O. S. Falebita, "Secure Web-Based Student Information Management System," *arXiv preprint arXiv:2211.00072*, 2022.

[2] S. Walia and S. K. Gill, "A framework for web based student record management system using PHP," *Int. J. Comput. Sci. Mobile Comput.*, vol. 3, no. 8, pp. 24-33, 2014.

[3] L. Sun and X. Chen, "Design and Application of University Students Management System based on web Platform," *Rev. Fac. Ing.*, vol. 31, no. 8, 2016.

[4] A. J. Kadam et al., "Mobile Web Based Android Application for College Management System," *Int. J. Eng. Comput. Sci.*, vol. 6, no. 2, pp. 20206-20209, 2017.

[5] B. P. Gautam and S. K. Shrestha, "Effective Campus Management through Web Enabled Campus-SIA (Student Information Application)," in *Proc. Int. Multi-Conf. Eng. Comput. Sci.*, vol. 1, 2012.

[6] C. Brozina, D. B. Knight, T. Kinoshita, and A. Johri, "Engaged to Succeed: Understanding First-Year Engineering Students' Course Engagement and Performance Through Analytics," *IEEE Access*, vol. 7, pp. 163686-163699, 2019, doi: 10.1109/ACCESS.2019.2945873.