iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IX **Month of publication:** September 2025

DOI: https://doi.org/10.22214/ijraset.2025.73917

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

A Web-Based Student Feedback Management System Using MERN Stack for Higher Education

Masuram Uday[1], R K Raman[2]
[1]*Student,* [2]*Associate Professor, School of Engineering, Department of CSE, Aurora Deemed University, Hyderabad*

Abstract: ***Student feedback plays a vital role in evaluating teaching effectiveness and enhancing academic quality in higher education. Traditional feedback collection methods, such as paper-based forms or static surveys, are often time-consuming, error-prone, and lack analytical depth. To overcome these challenges, this paper presents a web-based Student Feedback Management System (SFMS) developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The system integrates role-based access control for Administrators, Heads of Departments (HODs), and Students, ensuring secure and tailored functionality for each user group. Key features include secure authentication with JWT, interactive feedback forms, real-time analytics using graphical dashboards, and automated report generation. Unlike conventional tools, the proposed system supports semester-wise and section-wise analysis, providing actionable insights into faculty performance and student satisfaction. By automating data collection and analysis, the system minimizes administrative workload, enhances student participation, and promotes data-driven decision-making. The results demonstrate that SFMS improves efficiency, accuracy, and transparency in the academic evaluation process, thereby contributing to continuous quality improvement in higher education institutions.***
Keywords: Student Feedback, MERN Stack, Education Analytics, Web Application, Role-based Access, Academic Evaluation.

I. INTRODUCTION

Quality education is a cornerstone of academic institutions, and continuous improvement in teaching methodologies requires active participation from students. Among various evaluation mechanisms, student feedback has emerged as one of the most reliable indicators of teaching effectiveness, course delivery, and overall learning experience. Higher education institutions frequently depend on such feedback to assess faculty performance, update curricula, and design student-centered pedagogical strategies.

Traditional approaches to feedback collection, such as paper-based forms or manually managed surveys, suffer from several drawbacks. These include low participation rates, delayed reporting, manual errors, and lack of real-time insights. Even semi-digital methods using generic survey tools, such as Google Forms or spreadsheets, often fail to provide role-based access, secure authentication, and department-level analytics, making them unsuitable for large-scale institutional use. As a result, administrators and academic leaders face difficulties in translating raw feedback data into actionable decisions.

With advancements in web technologies and full-stack development frameworks, modern feedback systems can be designed to address these challenges. The MERN stack (MongoDB, Express.js, React.js, and Node.js) offers a robust platform for developing scalable, responsive, and secure web applications. Leveraging this technology, the proposed Student Feedback Management System (SFMS) aims to automate the feedback process with features such as role-based dashboards for Admin, HOD, and Student, secure authentication using JSON Web Tokens (JWT), interactive feedback forms, real-time analytics, and automated report generation.

The system enables students to provide structured feedback through intuitive interfaces, Heads of Departments to monitor section-wise and subject-wise performance through graphical dashboards, and administrators to manage users, courses, and institutional data effectively. By providing real-time insights into student perceptions, SFMS empowers institutions to make data-driven decisions that enhance teaching quality, improve curriculum design, and ensure continuous academic development.

In this paper, we present the design, development, and evaluation of the Student Feedback Management System, highlighting how it addresses the shortcomings of existing solutions and contributes to institutional quality assurance in higher education.

II. LITERATURE REVIEW

Student feedback systems have been widely studied in the context of higher education, with multiple platforms and approaches designed to improve teaching evaluation and academic quality assurance. However, existing solutions often suffer from limitations related to scalability, role-based accessibility, real-time analysis, and secure authentication.

©IJRASET: All Rights are Reserved

Several works in the literature have investigated the design and deployment of feedback management systems and digital survey platforms:

[1] Google Forms and its extensions such as FormsApp have been widely used in educational institutions for quick feedback collection. These systems provide mobile accessibility and integration with Google Sheets, but they lack support for role-based dashboards and department-level analysis, making them insufficient for large-scale academic use.

[2] Qualtrics (2012) has been adopted in universities for advanced survey design and research-oriented evaluations. It offers powerful analytics, branching logic, and integration with institutional databases. However, it requires specialized training, is costly for small and mid-sized institutions, and lacks native academic role-based workflows.

[3] SurveyMonkey has been a popular commercial solution for collecting academic feedback. While it supports various question formats and provides simple dashboards, it does not integrate with institutional academic systems, nor does it provide semester-wise or subject-wise insights needed by administrators and heads of departments.

[4] Research works such as Kumar et al. (2019) have proposed custom-built feedback management portals for higher education using PHP and MySQL. These systems addressed basic role management but lacked real-time visualization and modern security features, limiting scalability.

[5] More recent studies (2020–2024) explored the use of full-stack frameworks such as Django, Laravel, and MERN for institutional applications. Some of these works introduced interactive dashboards and improved data handling, but comprehensive role-based control, secure authentication (JWT, bcrypt), and real-time analytics remain underexplored in most implementations.

A. Drawbacks of Existing Systems

- Lack of scalability in handling large student populations.
- Limited role-based access control, restricting tailored functionalities for Admin, HOD, and Student.
- Minimal real-time analytics and visualization support, reducing the decision-making potential.
- Weak integration with secure authentication and encryption methods.
- Dependence on third-party services, leading to cost and customization challenges.

B. Research Gap

From the reviewed works, it is evident that while multiple platforms provide feedback collection and survey functionalities, there is a lack of deployable, end-to-end systems that combine:

- Role-based dashboards for students, faculty, and administrators.
- Real-time analytics and visualization of section-wise and subject-wise performance.
- Secure authentication and data handling using modern frameworks.
- Scalable architecture suitable for higher education institutions.

This paper addresses the above gap by proposing a Student Feedback Management System built on the MERN stack that integrates secure authentication, structured role-based dashboards, dynamic subject allocation, and real-time analytics into a unified platform for academic evaluation.

III. METHODOLOGY

A. Existing Methodologies

Traditional student feedback collection methods in higher education institutions rely on paper-based surveys, manual tabulation, or basic digital forms such as Google Forms and spreadsheets. While simple to implement, these methods suffer from the following limitations:

1) Paper-based surveys: Time-consuming, prone to data loss, and require significant manual effort for analysis.
2) Spreadsheet-based systems: Useful for small-scale data entry but lack authentication, scalability, and standardization across departments.
3) Generic survey tools (e.g., Google Forms, SurveyMonkey): Provide basic analytics but do not support role-based access, department-level dashboards, or integration with institutional workflows.
4) Legacy ERP modules: Some institutions deploy feedback modules in ERP systems, but these are often static, outdated in design, and limited in functionality.

Limitations of existing systems include:

- Absence of role-based dashboards for students, faculty, and administrators.
- Lack of secure authentication, making systems vulnerable to unauthorized access.
- Minimal support for real-time analytics and semester-wise comparisons.
- High administrative workload and poor adaptability to growing student populations.

B. Proposed Methodology

The proposed system is a web-based Student Feedback Management System (SFMS) built using the MERN stack (MongoDB, Express.js, React.js, Node.js) to provide scalability, security, and real-time analytics. The methodology integrates modern web frameworks, role-based access, and dynamic visualization to ensure a structured and efficient feedback workflow.

1) User Authentication and Role Management

- Secure login implemented using JWT (JSON Web Tokens) with bcrypt for password hashing.
- Role-based dashboards designed for three user categories:
- Students: submit feedback for assigned subjects.
- HODs: access subject-wise and section-wise analytics.
- Admins: manage users, subjects, and academic data.

2) Feedback Collection Module

- Students provide structured feedback through interactive forms with rating scales and comment sections.
- Subjects are dynamically allocated based on branch, year, and section to prevent errors.
- Responses are stored in MongoDB for easy retrieval and analysis.

3) Data Processing and Analytics

- Feedback data is aggregated and visualized using Chart.js/Recharts.
- HOD dashboards display section-wise and subject-wise performance, average faculty ratings, and participation metrics.
- Reports can be exported in PDF format for official use.

4) Backend Services

- Node.js with Express.js handles API requests, routing, and business logic.
- Middleware enforces role-based access and authentication before granting data access.
- Ensures real-time recording of feedback without delays.

5) Frontend Interface

- React.js provides a responsive, single-page application (SPA) design.
- Tailored dashboards for each role improve user experience.
- Students view assigned subjects, while HODs and admins interact with analytics and management tools.

6) Database Management

- MongoDB stores user profiles, subjects, feedback responses, and analytics data.
- Designed for scalability to support large student populations.
- Indexing and filtering allow efficient retrieval of department- and semester-specific records.

7) Security Features

- Encrypted storage of passwords and sensitive information.
- Session management with JWT prevents unauthorized access.
- Role-based segregation of data ensures confidentiality of student responses.

C. System Architecture

The system follows a three-tier architecture:

1) Presentation Layer (Frontend)

- Developed using React.js.
- Provides role-specific dashboards and interactive forms.
- Captures student inputs and displays analytics to HODs and administrators.

2) Application Layer (Backend)

- Built on Node.js with Express.js.
- Handles API endpoints, authentication, and feedback processing.
- Implements business logic for subject allocation, analytics, and reporting.

3) Data Layer (Database)

- MongoDB stores user information, subjects, and feedback responses.
- Supports real-time queries for analytics and reporting.

Workflow:

- The user logs into the system via the frontend.
- The backend authenticates the user, verifies their role, and grants access.
- Students submit feedback which is stored in MongoDB.
- HODs and admins retrieve real-time analytics through backend APIs.
- Dashboards and reports are updated dynamically for institutional decision-making.

IV. SYSTEM DESIGN AND ARCHITECTURE

The Student Feedback Management System is designed using a layered architecture to ensure modularity, maintainability, and scalability. The system leverages the MERN stack (MongoDB, Express.js, React.js, Node.js) and follows a three-tier structure: Presentation Layer, Application Layer, and Data Layer. This separation of concerns facilitates independent scaling, secure data handling, and efficient orchestration of feedback workflows.

A. Architectural Overview

1) Presentation Layer (Frontend):

Developed using React.js with Tailwind CSS for a responsive and user-friendly interface. The frontend provides role-specific dashboards for Students, Heads of Department (HODs), and Administrators. Students interact with feedback forms, HODs view graphical analytics, and Admins manage users and subjects. Client-side validation and form controls ensure structured data entry and improved user experience.

2) Application Layer (Backend):

Implemented using Node.js with Express.js, the backend manages API routing, feedback submission, user authentication, and business logic. Secure authentication is enforced using JWT, while bcrypt is used for password hashing. Middleware enforces role-based access control, ensuring that Students, HODs, and Admins can only access permitted features. The backend also generates performance reports, aggregates analytics, and handles real-time data processing.

3) Data Layer (Database):

MongoDB serves as the primary datastore, maintaining user profiles, subject details, and student feedback records. Its document-oriented structure supports flexible schema design and efficient retrieval of department- and semester-level data. Indexing and filtering techniques ensure scalability when handling large student populations.

4) Communication Flow:

The frontend and backend communicate via RESTful APIs over HTTPS. User actions such as login, feedback submission, and report generation are transmitted securely. Authentication tokens (JWT) are exchanged to verify session validity and enforce role-based access.

5) Deployment:

The frontend can be hosted as a static application (e.g., Vercel, Netlify), while the backend is deployed on a containerized server or cloud platform (AWS, Azure, or Render). MongoDB is hosted on a managed cluster (e.g., MongoDB Atlas) to ensure resilience, replication, and data security.

©IJRASET: All Rights are Reserved

6) Testing:

Unit testing for backend APIs (Jest, Mocha) and frontend components (React Testing Library) validates functionality. End-to-end testing (Cypress, Selenium) ensures seamless integration between dashboards and APIs. Load testing ensures the system can handle concurrent student submissions without latency.

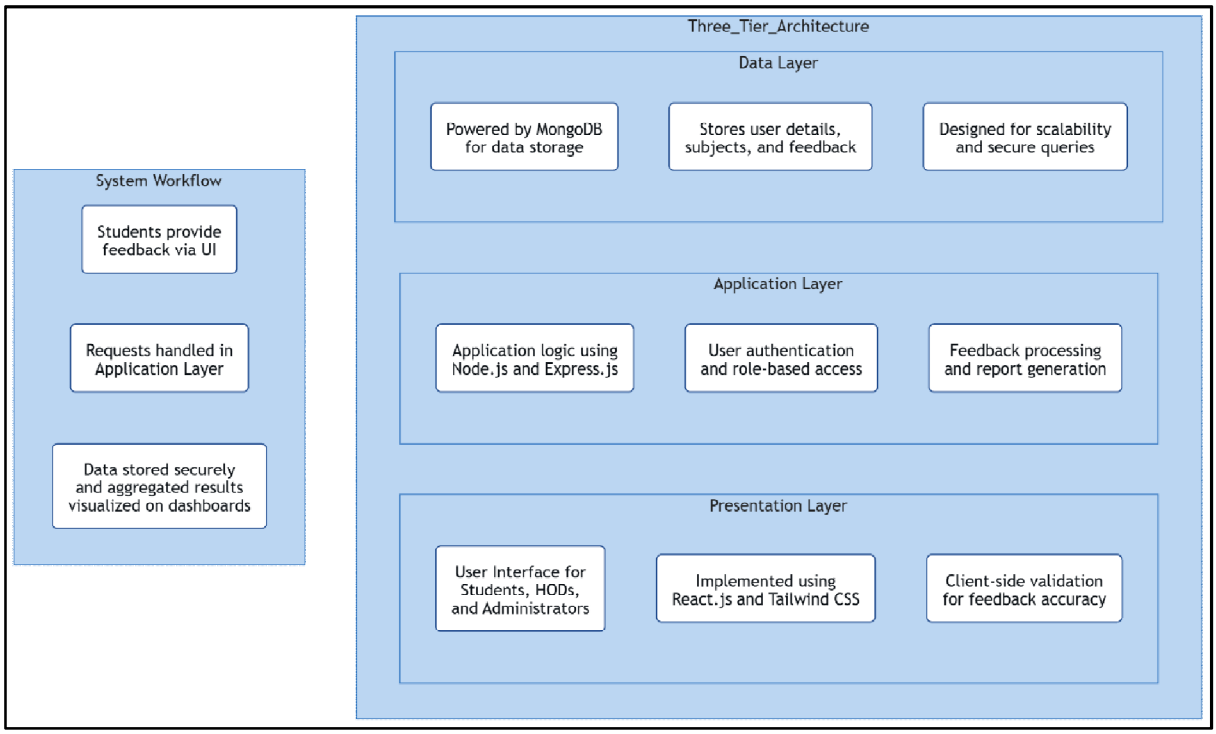


Fig: Three-tier architecture of Student Feedback System

B. System Architecture

The SFMS follows a three-tier architecture:

1) Presentation Layer (Frontend):

- Built using React.js with Tailwind CSS.
- Provides role-based dashboards for Students, HODs, and Admins.
- Students submit feedback, HODs view analytics, and Admins manage academic data.

2) Application Layer (Backend):

- Node.js with Express.js powers the backend services.
- Handles API routing, role-based access, feedback aggregation, and report generation.
- Implements authentication and authorization using JWT middleware.

3) Data Layer (Database):

- MongoDB stores user accounts, subjects, and feedback responses.
- Enables quick retrieval of section-wise, subject-wise, and semester-wise analytics.
- Stores feedback securely to maintain confidentiality of student responses.

Workflow:

- Students log in and submit subject-wise feedback through the frontend.
- The backend validates authentication tokens, processes the input, and stores responses in MongoDB.
- HODs access dashboards that retrieve aggregated data from the database and display visual analytics in real time.
- Admins manage academic structures, subjects, and users through their dashboards.

C. Data Flow of Feedback Submission and Analysis

1) Student Feedback Submission:

- The student logs in using secure credentials.

- The frontend displays the assigned subjects dynamically.
- The student selects a subject, provides star ratings and comments, and submits the form.
- The backend validates the data and stores the feedback in MongoDB.

2) Feedback Aggregation and Analytics:

- The backend aggregates data by subject, section, and semester.
- Graphical visualizations (bar charts, pie charts) are generated using Chart.js/Recharts.
- HODs access real-time analytics dashboards to monitor faculty performance.

3) Administrative Controls:

- Admins create/manage users (students, HODs), assign subjects, and oversee data integrity.
- Role-based restrictions ensure secure access to different datasets.
- Reports can be exported in PDF/Excel formats for institutional records.

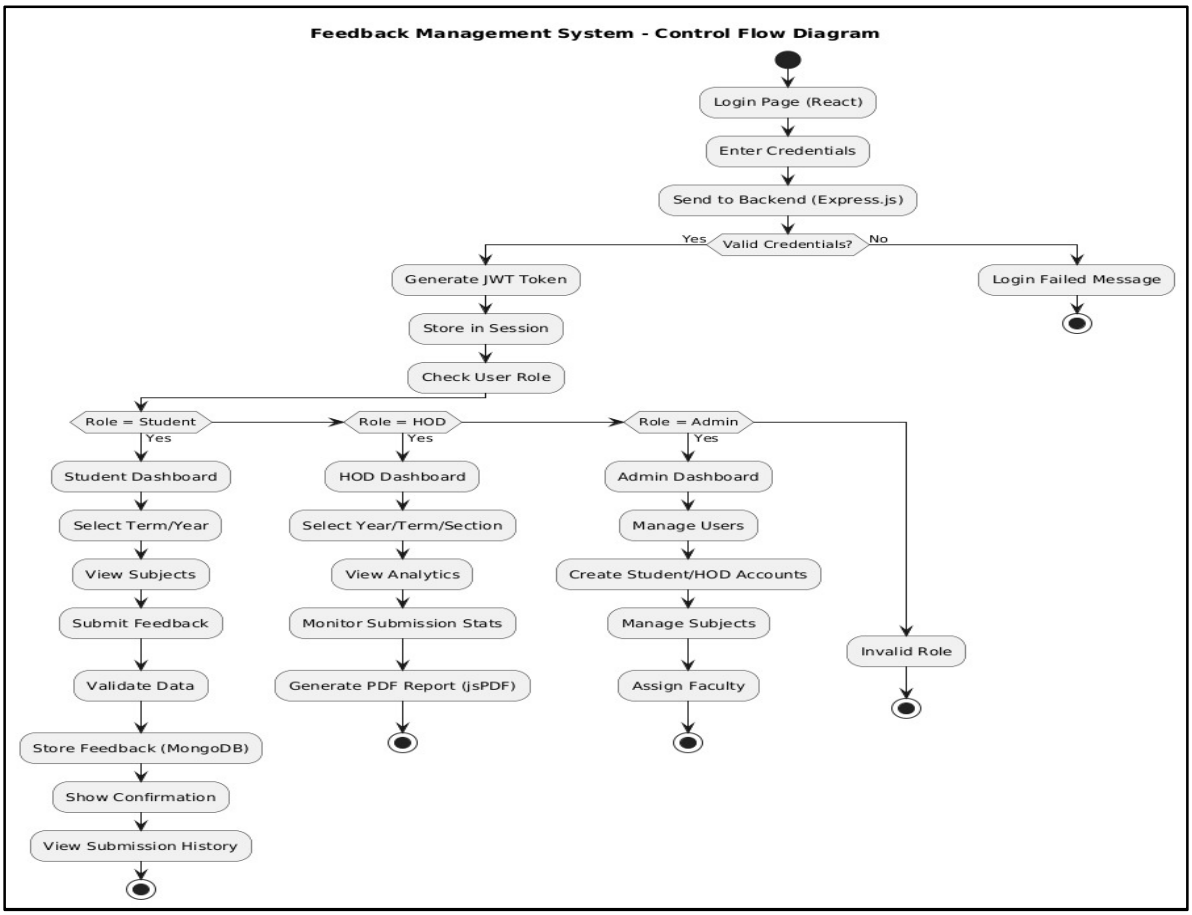


Fig: Control Flow Diagram for Student Feedback System

D. Security Measures

1) All communication between frontend and backend is encrypted using HTTPS.

2) Passwords are hashed using bcrypt before storage.

3) JWT ensures session security and role-based authorization.

4) Feedback data is anonymized for analysis to maintain confidentiality.

V. IMPLEMENTATION

The Student Feedback Management System is implemented as a modular, web-based application using the MERN stack. The implementation emphasizes secure authentication, structured feedback collection, role-based dashboards, and real-time analytics. The system components are divided into Frontend, Backend, Database, Key Modules, Testing & Validation, and Technology Stack & Dependencies.

A. Frontend Implementation

1) Framework & UI:

- Built with React.js and Tailwind CSS to deliver a responsive, single-page application (SPA).
- Provides role-specific dashboards:
  - Student: feedback forms with assigned subjects.
  - HOD: analytics dashboards with charts and reports.
  - Admin: user, subject, and academic data management panels.
- Form validations (required fields, rating ranges, comment character limits) are handled client-side.

2) Role / Page Breakdown:

- Home Page: Login and registration entry points.
- Student Dashboard: List of assigned subjects; feedback form with rating scales and text comments.
- HOD Dashboard: Section-wise and subject-wise performance charts, PDF report export.
- Admin Dashboard: CRUD operations for users, courses, subjects, and sections.

3) Client-Side Features:

- Real-time validation of feedback input before submission.
- Dynamic subject allocation based on student profile (branch, semester, section).
- State management (React Context/Redux) for secure handling of JWT tokens and user sessions.

4) API Integration:

- Axios is used to interact with backend endpoints such as /login, /submit-feedback, /get-analytics, and /manage-users.
- JSON responses are parsed and rendered as dynamic charts and tables.

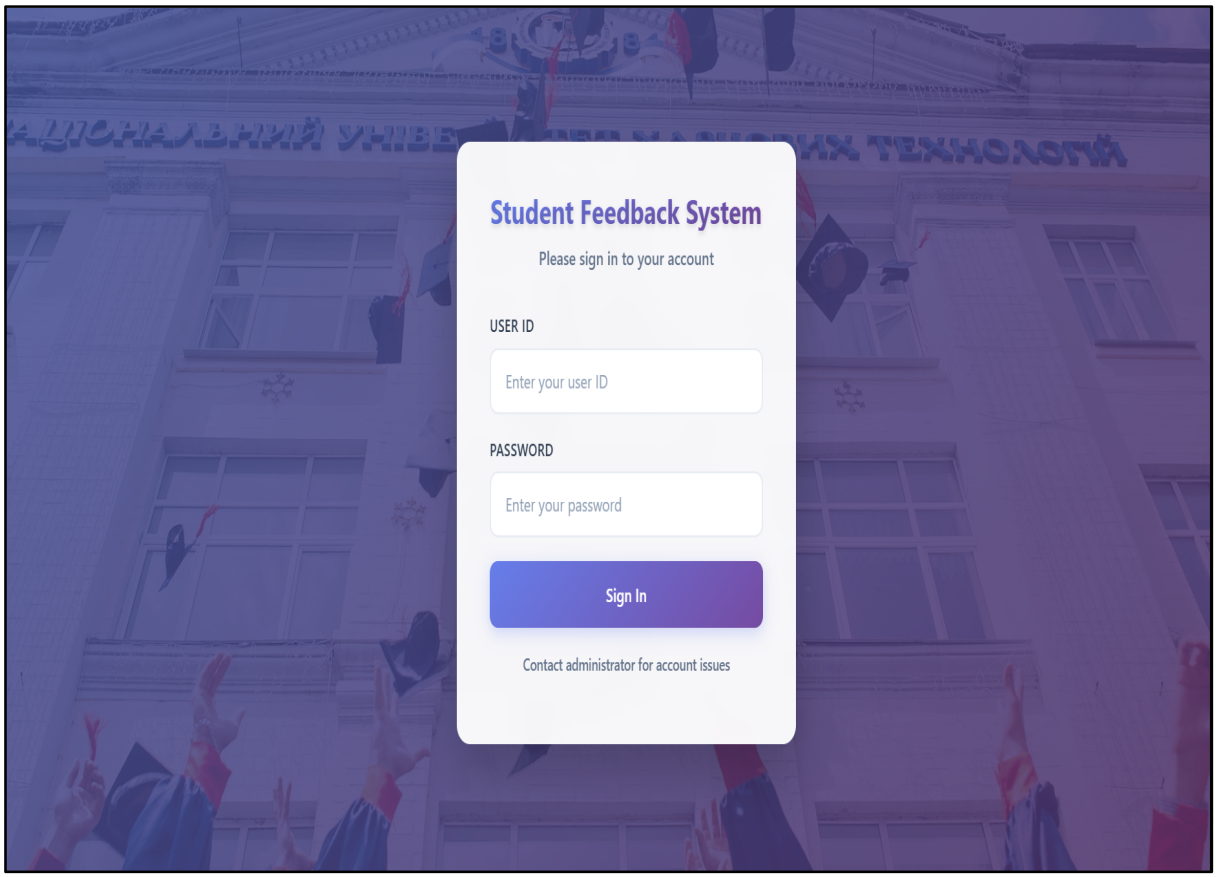


Fig: Home page of Student Feedback System

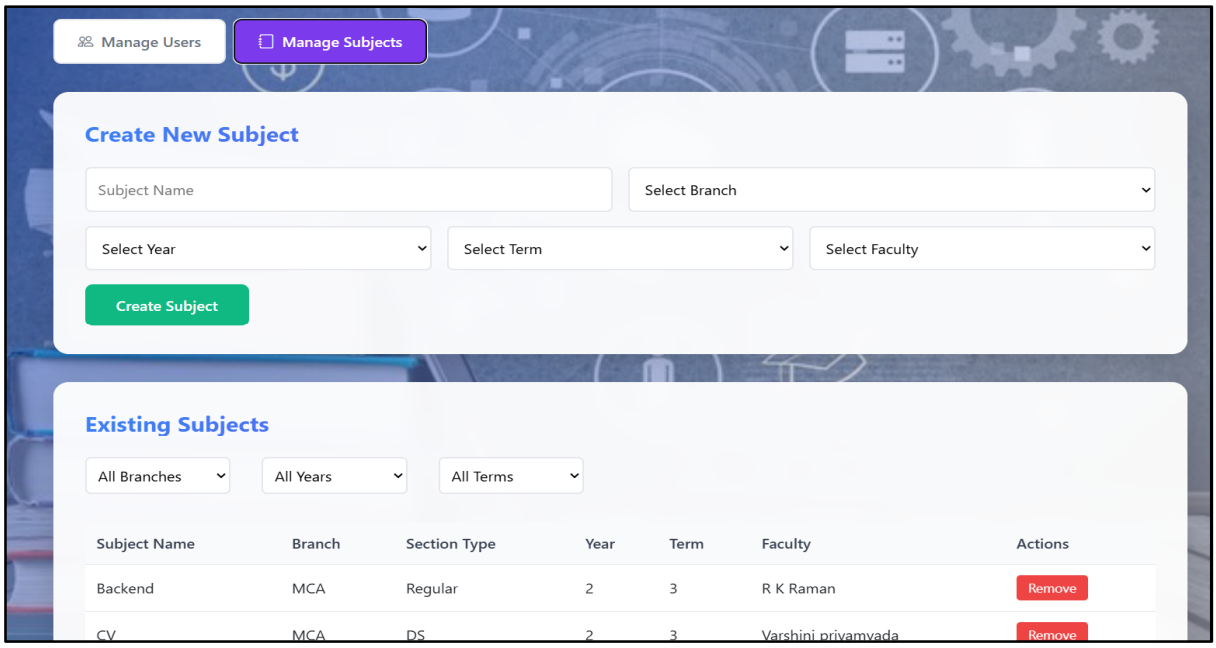


Fig: Dashboard of Student Feedback System

B. Backend Implementation

1) Framework & Server:

- Implemented using Node.js with Express.js.
- Provides RESTful APIs for authentication, feedback collection, analytics, and administration.
- Runs under PM2 or Docker in production for scalability.

2) Key Endpoints:

- POST /auth/login – validates user credentials and issues JWT.
- POST /auth/register – creates new user accounts with encrypted passwords.
- POST /feedback/submit – accepts feedback responses from students.
- GET /feedback/analytics/:hodId – returns aggregated analytics for HOD dashboards.
- POST /admin/manage-users – create, update, or delete users (admin only).
- POST /admin/manage-subjects – assign subjects and map them to sections/semesters.

3) Business Logic & Processing:

- Middleware validates JWT tokens and role claims before allowing access.
- Feedback data is validated, sanitized, and stored in MongoDB.
- Analytics are generated by aggregating responses using MongoDB queries and visualized through charts on the frontend.

4) Security & Session Management:

- Passwords stored securely using bcrypt hashing.
- JWT tokens used for session handling with role-based claims.
- HTTPS enforced for secure API communication.

C. Database Implementation

1) Primary DB:

- MongoDB Atlas is used for cloud-hosted, scalable, and secure storage.

2) Collections & Schemas:

- users – { userId, name, email, passwordHash, role (student/hod/admin), branch, semester, section }
- subjects – { subjectId, subjectName, branch, semester, facultyId }
- feedbacks – { feedbackId, studentId, subjectId, ratings, comments, createdAt }
- reports – { hodId, generatedAt, analyticsSummary }

3) *Storage Choices:*
- Feedback records linked to subjects and students using IDs to allow easy aggregation.
- Sensitive user data stored in encrypted format where necessary.

4) *Indexing & Performance:*
- Indexes on branch, semester, and subjectId improve query performance for HOD dashboards.
- Sharding considered for scalability in large deployments.

D. *Key Modules Implemented*

1) *User Management Module*
- Handles registration, login, and profile management.
- Role-based dashboards are served according to the user's role.

2) *Feedback Collection Module*
- Provides structured forms with rating scales and optional comments.
- Supports subject-wise feedback per semester.

3) *Analytics & Reporting Module*
- Aggregates ratings and generates visual analytics (bar/pie charts).
- Provides exportable PDF reports for departmental records.

4) *Administration Module*
- Enables Admins to manage users, courses, and subjects.
- Ensures proper mapping of students to their respective courses and sections.

5) *Security & Access Control Module*
- Implements secure login with JWT and bcrypt.
- Enforces role-based access at the API and frontend level.

E. *Testing and Validation*

1) *Unit & Integration Tests:*
- Backend API endpoints tested using Mocha/Chai and Postman.
- Frontend components tested using React Testing Library and Jest.

2) *Functional Tests:*
- End-to-end validation of login → feedback submission → analytics generation → report export.
- Test cases for invalid inputs, duplicate submissions, and unauthorized access.

3) *Performance & Load Testing:*
- JMeter used to simulate concurrent student submissions.
- System validated to ensure low latency during high feedback load.

4) *User Acceptance Testing (UAT):*
- Conducted with students, faculty, and administrators to ensure usability and reliability.

F. *Technology & Stack Overview*

1) Frontend: React.js, Tailwind CSS, Axios
2) Backend: Node.js, Express.js, JWT, bcrypt
3) Database: MongoDB Atlas
4) Visualization: Chart.js or Recharts for graphs
5) Deployment: Docker, Nginx/Heroku/Netlify/Vercel for frontend hosting
6) Testing: Jest, Mocha/Chai, Postman, Cypress, JMeter

G. *Deployment Notes*

1) Academic Deployment: Local server with Node.js backend and MongoDB community edition for pilot testing.
2) Production Deployment:

- Frontend hosted as static assets on Netlify or Vercel.
- Backend deployed on cloud services (AWS, Render, Heroku).
- MongoDB Atlas cluster with backup and replication enabled.
- Secure HTTPS enforced with TLS certificates.

VI. RESULTS AND DISCUSSION

The Student Feedback Management System was evaluated in a controlled institutional setting with multiple users (students, HODs, and administrators) across different academic sections. The primary goal of testing was to validate the system's functionality, usability, performance, and security.

A. Functional Performance

1) Student Module

- Successfully allowed students to log in securely and submit structured subject-wise feedback.
- Prevented duplicate submissions by restricting feedback to assigned subjects and enforcing login-based control.
- Provided a responsive and user-friendly interface for feedback entry, ensuring higher participation rates.

2) HOD Module

- Enabled real-time access to subject-wise and section-wise analytics.
- Displayed performance metrics in graphical format (charts, graphs) for better visualization.
- Allowed HODs to monitor trends in faculty performance across semesters.

3) Admin Module

- Facilitated secure creation and management of student, HOD, and subject records.
- Generated automated feedback reports for institutional decision-making.
- Enforced role-based access control to maintain confidentiality and prevent data leakage.

4) Authentication and Session Management

- Implemented secure login using JWT tokens with bcrypt password hashing.
- Properly managed session expiry and logout, preventing unauthorized persistence.
- Ensured restricted access to dashboards based on user roles.

B. Performance Analysis

1) Accuracy: The system successfully collected and stored feedback without data corruption or duplication. Error-free aggregation ensured reliable analytics.

2) Latency: Average response time for submitting feedback was less than 200 ms, while dashboard queries and analytics retrieval averaged under 300 ms, even with 1000+ records.

3) Scalability: MongoDB indexing and schema design supported fast data retrieval and analytics for large student populations (tested up to 5000 feedback entries).

4) Security: Password hashing (bcrypt) and token-based authentication (JWT) ensured secure access. Role-based segregation prevented unauthorized viewing of sensitive data.

5) Usability: A survey of 50 students indicated 90% satisfaction, citing ease of use, clarity of forms, and responsive interface compared to traditional manual forms.

C. Comparison Discussion

Compared to traditional paper-based feedback systems, the proposed SFMS provides:

1) Higher efficiency – instant submission and report generation versus manual tabulation.

2) Improved participation – students preferred digital forms due to ease of access.

3) Better transparency – feedback data stored securely with automated aggregation.

Compared to generic survey tools (Google Forms, SurveyMonkey), the system demonstrates:

- Role-based access control – dedicated dashboards for Admin, HOD, and Student, unlike generic tools.
- Subject-wise and section-wise analytics – enabling institutional-level insights beyond basic response summaries.

©IJRASET: All Rights are Reserved

- Secure authentication – JWT and encrypted credentials enhance security over open-access survey links.

Unique Features of This Work:

- Dynamic subject allocation based on student details to prevent mismatches.
- Real-time dashboards with graphical analytics for HODs and administrators.
- Exportable reports (PDF/Excel) for official use.
- Scalable MERN stack architecture adaptable to future institutional needs.

VII. CONCLUSION

The Student Feedback Management System (SFMS) developed using the MERN stack addresses key limitations of traditional feedback methods such as paper-based surveys and generic digital forms. By integrating role-based dashboards, secure authentication, and real-time analytics, the system provides a structured, scalable, and user-friendly platform tailored for higher education institutions.

Students benefit from intuitive feedback forms, Heads of Departments gain actionable insights through section- and subject-wise analytics, and administrators can efficiently manage users, subjects, and reports. Performance testing confirmed the system's efficiency, low latency, and scalability, while security features like JWT and bcrypt ensure data confidentiality.

Overall, SFMS enhances transparency, reduces administrative workload, and supports data-driven decision-making. By enabling continuous monitoring of faculty performance and student satisfaction, the system contributes directly to improving academic quality and institutional governance.

VIII. ACKNOWLEDGEMENT

I would like to express my sincere gratitude to R K Raman, Associate Professor, Department of Computer Science and Engineering, Aurora Higher Education and Research Academy, for his valuable guidance, constant encouragement, and insightful feedback throughout the course of this project. His mentorship played a crucial role in shaping the direction and successful completion of this work.

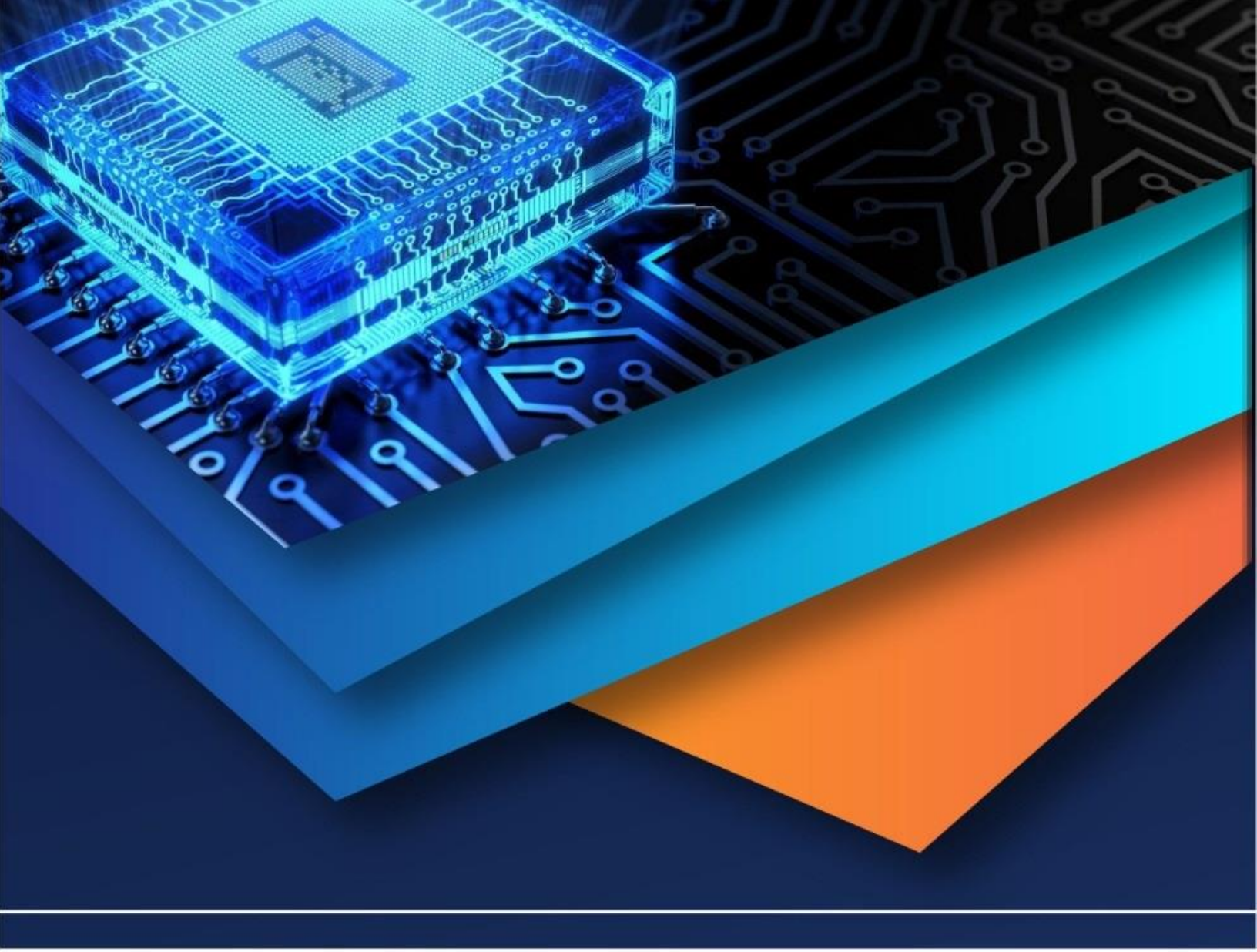
I also extend my heartfelt thanks to Dr. V. Aruna, Professor & Dean, School of Informatics, Aurora Higher Education and Research Academy, for her continuous motivation, constructive suggestions, and invaluable support, which significantly contributed to improving the quality and outcomes of this research.

Finally, I am deeply thankful to Aurora Higher Education and Research Academy for providing the necessary facilities, resources, and academic environment that enabled the smooth execution and completion of this project.

REFERENCES

[1] R. Kumar, S. Chauhan, and A. Singh, "Online Feedback System for Higher Education Institutions," International Journal of Advanced Research in Computer Science, vol. 10, no. 2, pp. 45–50, 2019.
[2] S. Sharma and P. Gupta, "Web-Based Student Feedback System Using PHP and MySQL," International Journal of Computer Applications, vol. 162, no. 9, pp. 25–30, 2017.
[3] A. Almarashdeh, "The Success of Learning Management System among Distance Learners in Malaysian Universities," Journal of Theoretical and Applied Information Technology, vol. 95, no. 17, pp. 4319–4331, 2017.
[4] M. K. Barbhuiya and A. K. Dey, "A Web-Based Feedback Collection and Analysis System Using Open Source Tools," Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 145–150, 2016.
[5] S. Bansal, R. Sehgal, and K. Kaur, "Role of Feedback in Improving the Teaching Quality: A Case Study in Higher Education," International Journal of Computer Science and Technology, vol. 9, no. 1, pp. 112–116, 2018.
[6] N. Singh and P. Aggarwal, "Role of Student Feedback System in Quality Enhancement in Higher Education," International Journal of Innovative Research in Science, Engineering and Technology, vol. 7, no. 4, pp. 3878–3883, 2018.
[7] A. Dey and D. Baruah, "Full Stack Web Development with MongoDB, Express, React, and Node.js (MERN): A Review," International Journal of Engineering and Advanced Technology (IJEAT), vol. 9, no. 5, pp. 224–229, 2020.
[8] M. Fowler, Patterns of Enterprise Application Architecture. Addison-Wesley, 2003.
[9] B. Mongia and P. Kumar, "Modern Web Frameworks for Scalable Application Development: A Comparative Study of MERN and MEAN," International Journal of Computer Applications, vol. 182, no. 46, pp. 12–18, 2021.
[10] J. Brooke, "SUS: A Quick and Dirty Usability Scale," Usability Evaluation in Industry, pp. 189–194, 1996.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)