iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78565

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 14 Issue III Mar 2026- Available at www.ijraset.com

A Web- Based Student Attendance Management System Using MERN Stack with Role-Based Access Control

Prof. Shital Gujar[1], Nitish Tarmale[2], Alok Singh[3], Sanket Pashte[4], Varad Chhallare[5]
[1]*Guide, Department of Computer Science and Engineering, Bharat College of Engineering, Badlapur, Maharashtra*
[2, 3, 4, 5] *Department of Computer Science and Engineering (Data Science), Bharat College of Engineering, Badlapur, Maharashtra*

Abstract: ***Many engineering colleges still use manual methods like paper registers and notice boards for managing attendance and results. This makes things slow and students often do not get information on time. In this paper we present a web- based Student Portal built during our Second Year Engineering project. The system supports four types of users - Student, Faculty, Admin, and Parent - each with their own login and dashboard. We used React.js for the frontend, Node.js and Express.js for the backend, and MongoDB Atlas to store data. Login is secured using JSON Web Tokens (JWT). Access to each dashboard is controlled based on the user role, both on the frontend and the backend. Role separation was confirmed - no user could access another role's data. The system is a simple but working solution to replace paper- based academic management.***
Keywords: React.js, Node.js, MongoDB, JWT Authentication, Role- Based Access Control, Web Application.

I. INTRODUCTION

Even today, many engineering colleges under University of Mumbai continue to use traditional methods like paper registers for attendance and notice boards for sharing results and announcements. While these methods have been used for years, they are not always efficient. Managing records manually can be time- consuming, and there are chances of errors or delays. Also, students and parents often do not get real- time updates, which creates a gap in communication.

One major issue is that there is no single platform where all users- students, faculty, parents, and administrators- can access the same information at the same time. For example, students may have to wait for notices to be displayed, and parents usually depend on students for updates about attendance and marks. Faculty members also have to handle a lot of paperwork, which becomes difficult when the number of students is large.

To solve these problems, we developed a web- based Student Portal System as part of our engineering mini- project. The main idea behind this project was to create a simple and practical system that can be used in real college environments. The portal provides separate login access for different users, and each user gets a dashboard based on their role.

Students can log in to check their attendance, marks, and important updates. Faculty members can mark attendance and upload results directly into the system, which reduces manual work. Parents can also log in and see their child's academic progress without depending on others. The admin has full control over the system, including managing users and maintaining records.

Another important part of this project is security. We made sure that each user can only access their own data. For example, a student cannot see another student's records, and parents can only view their child's details. This is done using proper login and access control mechanisms.

II. LITERATURE REVIEW

Earlier student information systems used PHP and MySQL [1]. These worked but had problems like full page reloads and no real-time updates. React.js fixes this using a virtual DOM that only updates the parts of the page that change [2].

Role-Based Access Control (RBAC) is a well-known security method where permissions are given based on a user's role, not to each user individually [3]. This is a good fit for our system because we have four clear user types.

JWT (JSON Web Token) is a way to handle login sessions without storing anything on the server. The token is created at login and sent with every API request. The server just checks the token's signature to verify the user [4]. We chose JWT over regular sessions because it works better with our separated frontend and backend.

©IJRASET: All Rights are Reserved

System	Technology Used	Features	Limitations
Manual System	Paper- based	Simple, no cost	Error- prone, slow
PHP- Based Systems	PHP, MySQL	Digital storage	No real- time updates
RFID System	RFID Hardware	Fast attendance	Expensive
Biometric System	Fingerprint/Face	High accuracy	Privacy issues
Proposed System	MERN Stack	Multi- role, real- time, secure	Basic UI

Table 1: Literature Survey Comparison

III. METHODOLOGY

The proposed Student Portal system was developed using a structured software development approach, beginning with requirement analysis to identify limitations in traditional academic management systems such as manual attendance tracking and delayed result dissemination. Based on these observations, a web-based solution was designed with four distinct user roles-Student, Faculty, Parent, and Admin- implemented using Role- Based Access Control (RBAC). The system follows a client-server architecture, where the frontend is developed using React.js, the backend is built with Node.js and Express.js, and MongoDB Atlas is used for cloud- based data storage. Communication between client and server is handled through RESTful APIs.

To ensure secure access and data protection, JSON Web Token (JWT) authentication was implemented. Upon successful login, a token is generated and stored on the client side, which is then included in all subsequent API requests. A two- layer security mechanism is employed: at the frontend level, protected routes validate user roles before granting access to specific dashboards, while at the backend level, middleware verifies the authenticity of the token and enforces authorization policies. This approach ensures that unauthorized access is prevented even in cases of direct API invocation.

The system is modularly structured to handle functionalities such as attendance management, result processing, and user administration. Axios is used for efficient API communication, and the project is developed using Vite for improved performance and faster build times. The implemented system was tested across all user roles to validate functionality, security, and performance. The results demonstrate that the system provides efficient real- time data access while maintaining strict role- based access control and overall system reliability.

IV. SYSTEM ARCHITECTURE

The system uses a standard client- server structure. The React frontend talks to the Node.js/Express backend through REST API calls using Axios. All data is stored in MongoDB Atlas, which is a cloud database. Table 1 shows the full technology stack used.

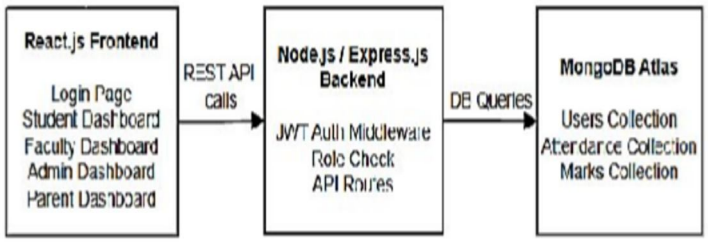


Figure 1: System Architecture Diagram

A. Security Approach

We used two layers of security to make sure users only see what they are allowed to see.

1) Frontend: Every dashboard route is wrapped in a Protected Route component. Before loading any page, it checks if the JWT in session Storage is valid and if the user's role matches the route. If not, the user is sent back to the login page.

2) Backend: Every API endpoint that returns protected data has a middleware function that reads and verifies the JWT from the request header. If the token is missing or wrong, the server returns a 403 error. This means even if someone skips the frontend and calls the API directly, they still cannot get the data.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

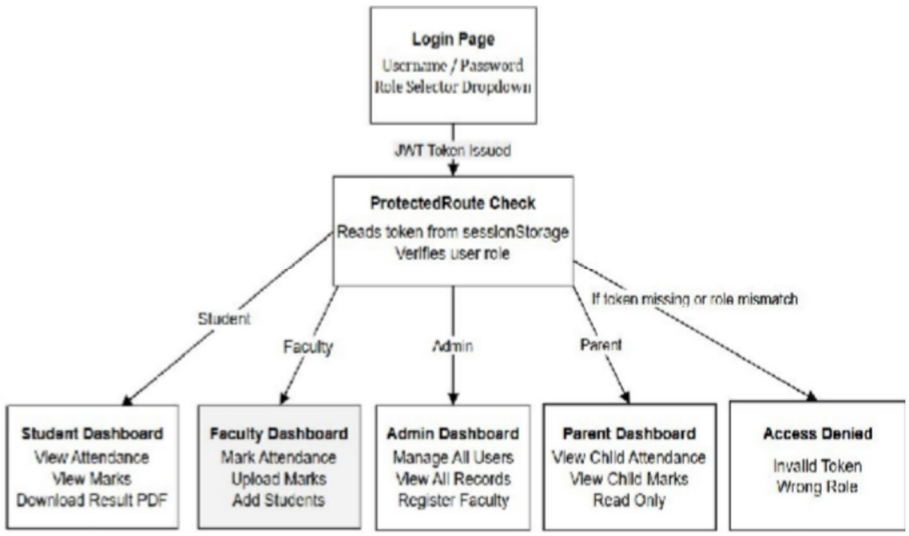


Figure 2: Role- Based Access Control Diagram

V. MODULE DESCRIPTION

The portal has four user roles. Table 2 shows what each role is allowed to do in the system.

Role	Read	Write	Modify	Delete	User Management
Student	Yes	No	No	No	No
Faculty	Yes	Yes	Yes	No	No
Parent	Yes	No	No	No	No
Admin	Yes	Yes	Yes	Yes	Yes

Table- 2: Role Permissions

1) Student: Can view their own attendance percentage, subject-wise marks, and download a result PDF.
2) Faculty: Can mark or update attendance for students, upload marks, and add new student records.
3) Parent: Read- only view of their child's attendance and marks. No editing allowed.
4) Admin: Full access to add students, register faculty, and manage all records in the system.

VI. IMPLEMENTATION

The project folder is organized into pages/, components/, and utils/ under the src/ directory. App.jsx handles all the routes. The Axios instance in api.js automatically adds the JWT to every outgoing request header so we do not have to do it manually in each file.

The ProtectedRoute component checks two things before showing any dashboard page: first, whether a token exists in sessionStorage: second, whether the token's user role matches the required role for that page. If either check fails, it redirects to login.

On the backend, a checkAuth middleware function extracts the token from the Authorization header, verifies it with the server's secret key using the jsonwebtoken library, and attaches the user's role to the request object. Protected routes then check this role before returning any data.

We used Vite as the build tool instead of Create React App because it starts the development server much faster and applies code changes instantly using Hot Module Replacement (HMR).

©IJRASET: All Rights are Reserved

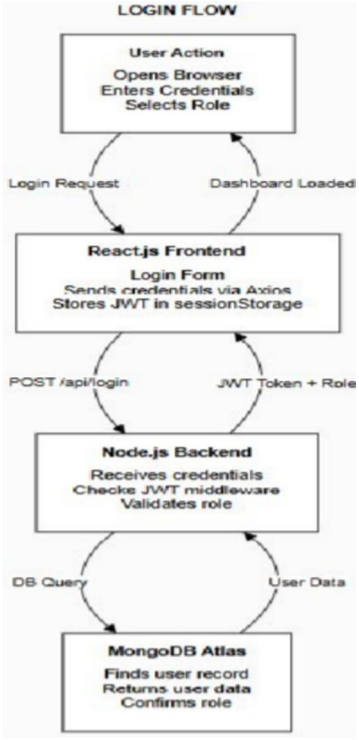


Figure 3: Data Flow Diagram

VII. RESULTS

We tested the system manually across all four roles to check whether security restrictions were working correctly. Here are the main observations:

1) The Vite development server started in about 0.8 seconds. This is much faster than Create React App which took around 4 seconds for the same project size.
2) When a faculty member marked attendance, the student dashboard showed the updated percentage instantly
3) All four role isolation tests passed. A student session could not open the admin or faculty dashboard even by typing the URL directly.
4) JWT tokens cleared automatically on browser tab close in Chrome, Firefox, and Edge. This confirms sessionStorage works as expected for session security.
5) Direct API calls without a valid token returned 403 Forbidden from the backend, confirming that backend protection works independently of the frontend.

Overall the system worked as expected. The two- layer security approach - ProtectedRoute on the frontend and JWT middleware on the backend - made sure that role boundaries were respected in all test cases.

VIII. FUTURE SCOPE

The proposed system can be further enhanced to improve usability, scalability, and functionality. While the current implementation provides a secure and efficient platform for attendance and academic management, future improvements can focus on real- time communication, better accessibility, and advanced features to make the system more robust and suitable for large- scale deployment.

A. Future Enhancements

1) Real- Time Updates: Implement WebSockets to provide instant updates for attendance and results.
2) Mobile Application: Develop a mobile app using React Native for better accessibility.
3) Notifications System: Add email/SMS alerts for attendance shortage and important updates.
4) Advanced Analytics: Include graphical dashboards for performance and attendance analysis.
5) Biometric Integration: Integrate fingerprint or face recognition for automated attendance marking.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

IX. CONCLUSION

In this project, we successfully designed and developed a Student Portal that replaces traditional paper- based academic processes with a structured and secure web-based system. The main objective was to create something practical that could actually be used in a real college environment, and not just a theoretical model. By introducing a centralized platform, we were able to simplify tasks like attendance tracking, result management, and communication between different users.

The system supports four different user roles-student, faculty, parent, and admin- each with its own login and dashboard. This role-based approach made the system more organized and ensured that users only interact with features relevant to them. We implemented JWT- based authentication to maintain security, which helped in protecting user data and preventing unauthorized access. During testing, role isolation worked correctly, meaning users could not access data outside their permissions.

Another important outcome of this project was the improvement in efficiency. Tasks that normally require manual effort, such as maintaining registers or updating notice boards, can now be done quickly through the portal. The system also reduces the chances of errors and provides faster access to information. From our testing, the performance of the system was smooth and responsive, even with multiple users interacting at the same time.

This project also gave us practical experience in full-stack development, including frontend design, backend logic, database handling, and authentication mechanisms. It helped us understand real-world challenges like handling user roles, ensuring data security, and building a user- friendly interface.

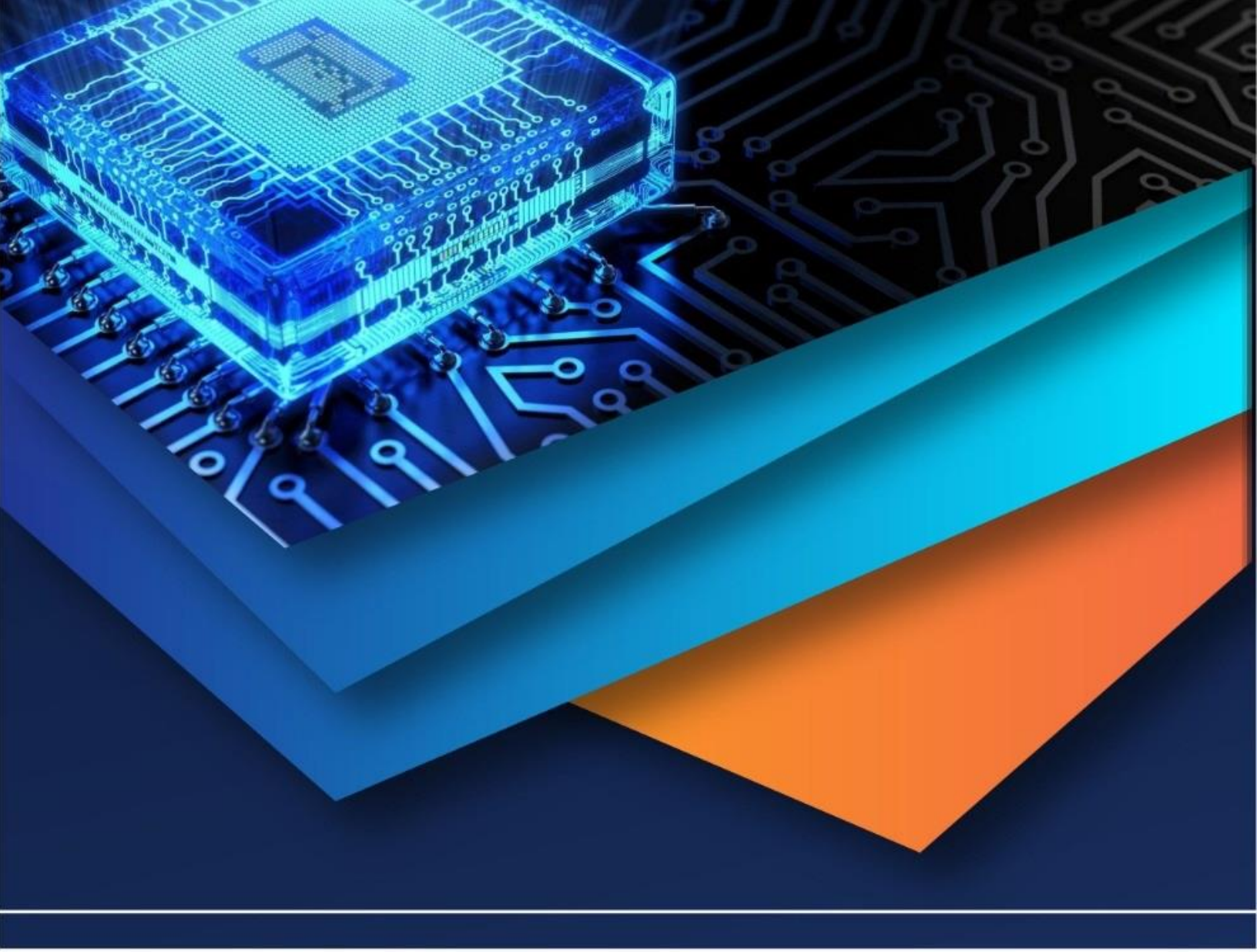
X. ACKNOWLEDGEMENT

We would like to thank our project guide and the Department of Computer Engineering for their support and guidance throughout this project.

REFERENCES

[1] S. Chopade, R. Patil, "Web Based Student Information System," International Journal of Computer Applications, vol. 975, pp. 1–5, 2014.
[2] React Official Documentation, "React - A JavaScript Library for Building User Interfaces," https://react.dev, 2023.
[3] D. F. Ferraiolo, D. R. Kuhn, Role- Based Access Control, Artech House, 2nd ed., 2007.
[4] M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)," RFC 7519, IETF, May 2015.
[5] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," https://owasp.org/Top10, 2021.
[6] React Router Documentation, "React Router DOM v6," https://reactrouter.com, 2021.
[7] Vite Official Documentation, "Vite - Next Generation Frontend Tooling," https://vitejs.dev, 2021.
[8] Axios Documentation, "Axios - Promise Based HTTP Client," https://axios- http.com, 2020.

©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)