International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Centralized School Administration & CRM Portal

Vikash Kumar Mahto

Department of Science and Technology,
G. H. Raisoni Skill Tech University, Nagpur, Maharashtra, India

Abstract
Educational institutions deal with a lot of data including student records, attendance, exam results, fee payments, staff details and admission queries. The old manual systems and separate digital tools often cause problems like wasted time repeated data, security risks and delayed information. To solve these issues this research suggests creating a Centralized School Administration & CRM Portal.This web application is going to be really flexible. It is based on what the user needs. It will combine the tasks that schools need to manage every day with features that help them interact with people like students and parents in a way. The school portal wants to make it easier for schools to do their work and talk to students, parents and staff. By using this portal schools can reduce inefficiencies. Make better use of their resources. The Centralized School Administration & CRM Portal will help schools to manage their data effectively. It will provide a platform, for all school management operations and CRM functionalities.

The system we are talking about is made using the web technologies. This includes React.js for the part that users see Node.js and Express.js for the behind the scenes work and MongoDB as the database.

The system uses a way to make sure only the right people can get to the information and this is done with something called JWT and role-based access control.

We are using a system for the backend to make it easier to keep everything running smoothly and to make it work for a large number of people.

The part of the system that manages relationships with customers or in this case the CRM module keeps track of people who're interested in joining, the follow-ups and when they finally become students.

The system has dashboards for five types of users: Administrator, Teacher, Student, Parent and Admission Staff. These dashboards get updated in time. This means that when something changes the dashboards show the information right away.

The system is really good at showing how well it works. It makes things easier and more efficient. It also helps keep all the information easy to see. This is a lot better than the systems.

The system is also very flexible. This means it can be used by schools or big colleges. It can even be used by coaching institutes or schools with branches. The system is great, for any kind of organization.

KEYWORDS*: School Management System; Customer Relationship Management (CRM); Role-Based Access Control; JWT Authentication; MERN Stack; Web Application; Educational Data Management; Real-Time Dashboard.*

1. Introduction

In todays age schools and colleges deal with a lot of administrative work. They have to manage types of data such as:

- Student enrollment records
- Attendance logs
- Examination results
- Fee payments
- Faculty information
- Academic schedules
- Parent communications

In the past schools used registers, spreadsheets and desktop software to handle this data.. These methods are no longer good enough for modern schools.

Manual systems take a lot of time can have errors and are hard to expand. As more students enroll managing records manually gets very complicated and inefficient.

Paper-based records can get damaged, lost or duplicated easily. Also without a central control system administrators find it difficult to create reports or make smart decisions.

When schools use digital tools like spreadsheets data often stays separate in different departments. For example attendance records might be kept separately from fee management systems.

Admission inquiries might not be tracked properly. This separation causes communication gaps. Makes operations less efficient.

Another big challenge, for schools is managing admissions poorly. Many schools lose students because they do not track inquiries well and do not follow up properly.

Without a system inquiry data might be recorded wrongly or forgotten.

To solve these problems schools need a online, secure and role-based management platform. This platform should integrate all administrative processes into one system.

Such a system would help schools manage their data better and make decisions.

It would also improve communication. Make operations more efficient.

The system would help schools track admissions properly and follow up with students.

This would ultimately help schools provide education and services.

The proposed **Centralized School Administration & CRM Portal** addresses these challenges by providing:

- Centralized database storage
- Secure authentication and authorization
- Role-based dashboards

- Real-time academic updates
- Integrated CRM for admission management
- Scalable architecture using modern web technologies

The school administration system is really good because it combines things that schools need to run with things that help them keep track of people. This makes the school run smoothly and it is easier to see what is going on. The school can also make decisions because it has good information to base those decisions on. The system is, about making the school administration better by using the school administration features and the customer relationship management features together.

2. Motivation

The motivation behind this research arises from observing practical challenges faced by educational institutions in managing academic and administrative workflows.

2.1. Inefficiency of Manual Systems

Paper-based management systems consume excessive time and human effort. Tasks such as attendance recording, report generation, fee tracking, and enquiry follow-ups require repetitive manual intervention. Human errors in data entry further reduce reliability.

2.2. Fragmented Digital Solutions

Many institutions adopt isolated software tools for specific tasks. However, these systems often lack integration. As a result:

- Data duplication occurs.
- Real-time synchronization is absent.
- Cross-department communication becomes difficult.
- Decision-making becomes delayed.

2.3. Lack of Structured Admission Tracking

Admission management is a crucial factor for institutional sustainability. However, without a CRM system:

- Enquiries are not tracked systematically.
- Follow-ups are inconsistent.
- Conversion rates remain low.
- Data analytics for admissions are unavailable.

2.4. Need for Secure and Role-Based Access

Educational data is sensitive and requires strict access control. Without proper authentication mechanisms, data privacy may be compromised. Institutions require systems that ensure:

- Secure login mechanisms.
- Encrypted password storage.
- Role-based access restrictions.
- Controlled API authorization.

2.5. Requirement for Real-Time Updates

Modern users expect instant information access. Parents want to check attendance status immediately. Students need quick access to results. Administrators require real-time fee reports. A centralized real-time web application fulfills these expectations.

These motivations collectively inspired the development of a comprehensive web-based school administration and CRM portal.

3. Contribution

This research makes the following significant contributions:

1. Development of an Integrated Academic & CRM Platform

The system combines school administration modules with admission CRM functionality in a single unified application.

2. Implementation of Role-Based Access Control (RBAC)

Five distinct user roles are defined with restricted permissions:

- Administrator
- Teacher
- Student
- Parent
- Admission Staff

3. Secure Authentication using JWT

JSON Web Token (JWT) based authentication ensures secure and stateless session management.

4. Layered Backend Architecture

The backend is structured into:

- Controller Layer
- Service Layer
- Database Layer

This improves maintainability and scalability.

5. Centralized MongoDB Database Design

Document-based schema design ensures flexible and scalable data storage.

6. Real-Time Dashboard Implementation

Updates made by one user are instantly reflected in relevant dashboards.

7. RESTful API Design

Structured endpoints allow secure communication between frontend and backend.

8. Scalable MERN Stack Implementation

The use of MongoDB, Express.js, React.js, and Node.js ensures modern full-stack compatibility.

9. Operational Efficiency Improvement

The system reduces paperwork, minimizes human errors, and improves decision-making accuracy.

4. Related Work

Many people who study this topic have looked into how schools can use computers to make their management systems better. At first school management systems just focused on putting attendance and exam records on the computer. These systems did not do a good job of putting all the information together.

Now people are saying that schools need to have a place on the web where they can do all their administrative tasks. These web applications are good because they can be used on types of computers they can store all the information in one place and they make it easier for people to talk to each other.

Studies on Customer Relationship Management or CRM in schools show that it is important to have a system to track enquiries and to manage how the school interacts with students. CRM systems are used a lot in companies. Not as much in schools. If schools use CRM they can get more students to enroll. They can get better at analyzing their data.

New web technologies like Node.js, Express.js and MongoDB are really good for applications because Node.js, Express.js and MongoDB can handle a lot of work.

React.js is a tool that helps make web pages that're easy to use and that load quickly so people can use web pages made with React.js without getting frustrated.

I think Node.js, Express.js MongoDB and React.js are all useful, for making web applications that people will like to use.

People who study security say that web applications should use something called JWT-based authentication to keep users safe when they are logged in. This is a way to make sure that only the right people can get into the system. Digital transformation in educational management systems is very important. It can be achieved with the help of digital transformation, in educational management systems.

Despite advancements, many existing educational platforms focus either on academic management or CRM independently. The proposed system bridges this gap by integrating both functionalities into a unified role-based platform.

5. System Architecture

The Centralized School Administration & CRM Portal follows a **three-tier architecture model** consisting of:

1. Presentation Layer (Frontend)
2. Application Layer (Backend)
3. Data Layer (Database)

This architectural model ensures modularity, scalability, maintainability, and secure communication between system components.

5.1. High-Level Architecture

The system architecture consists of:

- Client Browser (User Interface)
- REST API Server
- Centralized MongoDB Database

5.1.1. Presentation Layer

The presentation layer is developed using **React.js**, HTML, CSS, and JavaScript. It provides:

- Role-based dashboards
- Dynamic data visualization
- Form validation
- Navigation without page reload (Single Page Application)

Users interact with the system via a web browser. All requests are sent to the backend using HTTP protocols.

5.1.2. Application Layer

The application layer is implemented using:

- Node.js (Runtime Environment)
- Express.js (Web Framework)

This layer performs:

- Business logic processing
- Authentication & authorization
- API routing
- Input validation
- Error handling

It acts as a bridge between frontend and database.

5.1.3. Data Layer

The data layer uses **MongoDB**, a NoSQL document-based database.

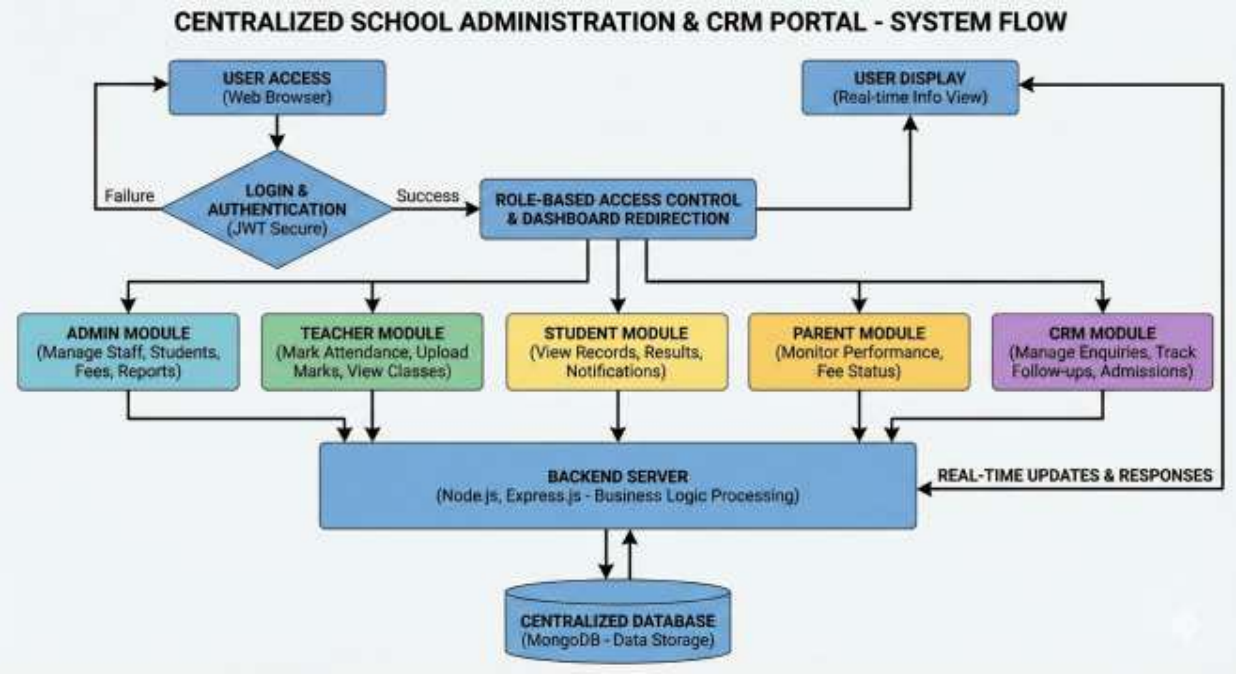
Advantages:

- Flexible schema design
- Scalable architecture
- High performance for read/write operations
- JSON-like document structure

All institutional data is stored centrally to avoid duplication.

6. Overall Workflow Design

The system workflow follows a structured sequence:



6.1. User Access

Users access the system through a web browser.
Supported roles:

- Administrator
- Teacher
- Student
- Parent
- Admission Staff

6.2. Authentication Process

1. User enters credentials.
2. Backend verifies credentials.

3. Passwords are compared using encrypted hashing.
4. If valid, JWT token is generated.
5. Token is sent to frontend.
6. Frontend stores token securely.
7. User redirected to role-specific dashboard.

If authentication fails, access is denied.

6.3. Role-Based Dashboard Redirection

After login:

- Admin → Admin Dashboard
- Teacher → Teacher Dashboard
- Student → Student Dashboard
- Parent → Parent Dashboard
- Admission Staff → CRM Dashboard

Each dashboard has restricted functionality.

6.4. Module Operations

Admin Module

- Manage students
- Manage teachers
- Manage fees
- Generate reports
- View analytics

Teacher Module

- Mark attendance
- Upload marks
- View assigned classes

Student Module

- View attendance
- View results
- Check notifications

Parent Module

- Monitor child performance
- View fee status

CRM Module

- Record admission enquiries
- Track follow-ups
- Convert enquiry to student

6.5. Real-Time Data Flow

When any operation is performed:

1. Frontend sends API request.
2. Backend processes logic.
3. Database updates.
4. Response sent back.
5. UI updates dynamically.

This ensures real-time synchronization.

7. Database Design

Database design plays a critical role in system performance and consistency.

7.1. Database Schema Overview

The system contains the following primary collections:

1. Users
2. Students
3. Teachers
4. Parents
5. Attendance
6. Fees
7. Enquiries

7.2. Collection Design

7.2.1. Users Collection

Stores authentication credentials.

Fields:

- User Id
- name
- email
- password (hashed)
- role
- created At

7.2.2. Students Collection

Fields:

- Student Id
- name
- class
- section
- parent Id
- contact Details
- admission Date

7.2.3. Teachers Collection

Fields:

- Teacher Id
- name
- subject
- assigned Classes

7.2.4. Attendance Collection

Fields:

- Student Id
- class
- date
- status (Present/Absent)

7.2.5. Fees Collection

Fields:

- Student Id
- total Fees
- paid Amount
- due Amount
- payment Status

7.2.6. Enquiries Collection (CRM Module)

Fields:

- enquiry Id
- student Name
- parent Contact
- enquiry Date
- follow Up Status
- conversion Status

7.3. Data Relationships

- Parent linked to Student.
- Teacher linked to assigned classes.
- Attendance linked to Student ID.
- Fees linked to Student ID.
- Enquiry converts into Student record.

This ensures logical data flow and referential consistency.

8. Backend Architecture

The backend follows a **Layered MVC Architecture**.

8.1. Controller Layer

Handles:

- Incoming HTTP requests
- Extracting request parameters
- Calling service functions

- Sending responses

Example:
POST /api/login
GET /api/students
PUT /api/attendance

8.2. Service Layer

Contains:
- Business logic
- Role validation
- Data processing
- Conversion logic

Example:
- Teacher can only mark attendance for assigned classes.
- Parent can only view child data.
- Enquiry conversion creates student record.

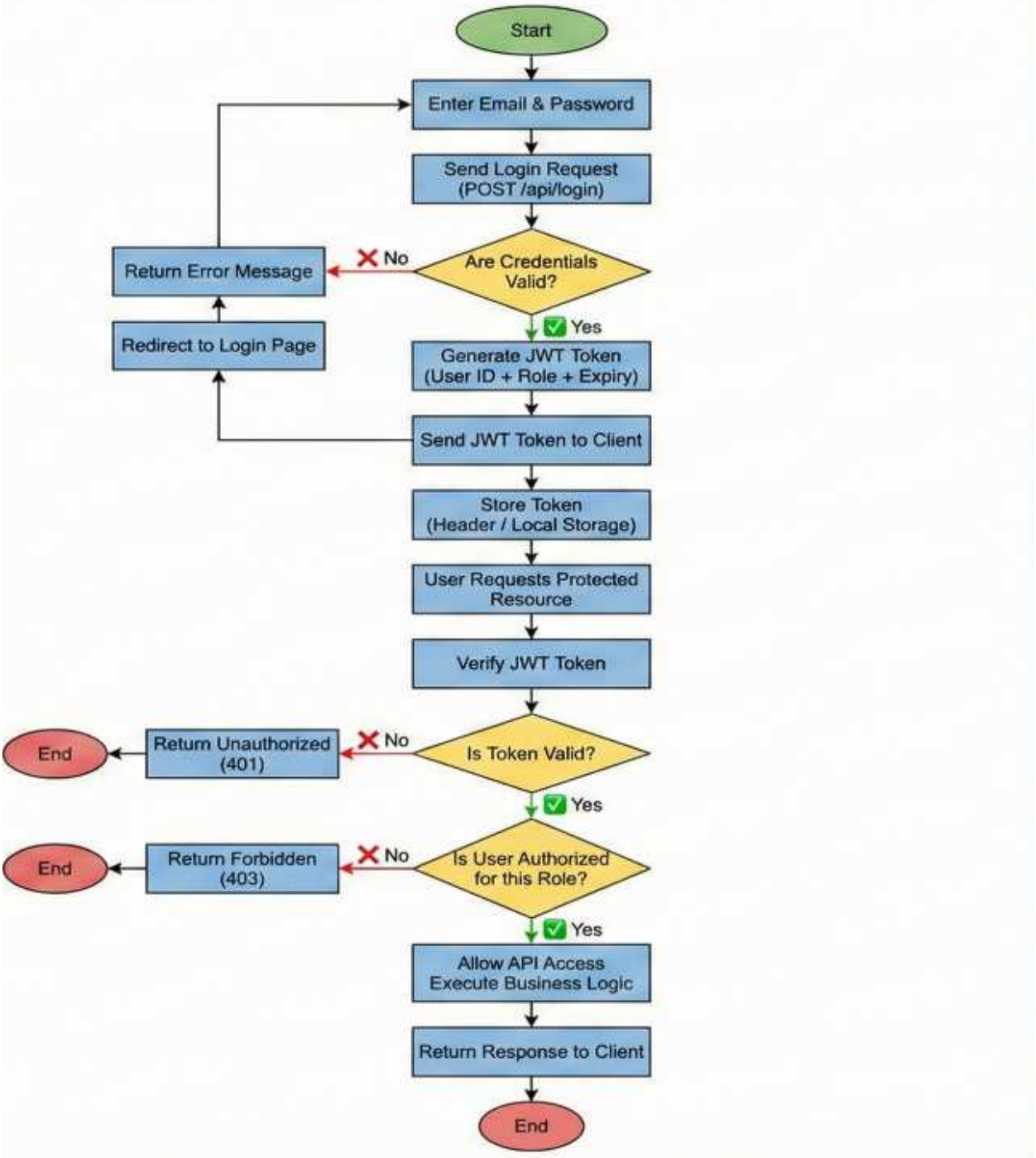
8.3. Database Layer

Uses Mongoose models to interact with MongoDB.
Responsibilities:
- Schema validation
- CRUD operations
- Query optimization

9. Authentication and Security Model

Security is a major component of the system.

Figure X: JWT-Based Authentication and Role-Based Authorization Flowchart



9.1. Password Encryption
- Passwords stored using hashing algorithm.
- Plain text passwords are never stored.
- Protects against data breaches.

9.2. JWT Authentication
Process:
1. Login success → Generate token.
2. Token contains:
- User Id
- role
- expiration time
3. Token sent in request headers.
4. Middleware verifies token before API access.

9.3. Role-Based Authorization
Middleware checks:
- Token validity.
- User role.
- Access permissions.

Unauthorized requests are rejected.

9.4. Input Validation
- Required field checks.
- Data type validation.
- Prevents injection attacks.

9.5. Error Handling
- Structured error responses.
- Prevents system crash.
- Logs unauthorized attempts.

10. API Structure
The system uses RESTful APIs.

10.1. Authentication APIs
- POST /api/login
- POST /api/register

10.2. Student Management APIs
- GET /api/students
- POST /api/students
- PUT /api/students/:id
- DELETE /api/students/:id

10.3. Attendance APIs
- POST /api/attendance
- GET /api/attendance/: studentId

10.4. Fee APIs
- GET /api/fees
- POST /api/fees/payment

10.5. CRM APIs
- POST /api/enquiry
- PUT /api/enquiry/:id
- POST /api/enquiry/convert

Each API follows:
- Proper HTTP method usage
- JWT validation
- Structured JSON response

11. Frontend Design and Implementation
The frontend of the Centralized School Administration & CRM Portal is designed to provide a responsive, intuitive, and role-specific user interface. It is developed using **React.js**, which enables the creation of a Single Page Application (SPA).

11.1. Frontend Technology Stack
- HTML5 – Structure of web pages
- CSS3 / Tailwind CSS – Styling and responsive design
- JavaScript (ES6+) – Dynamic functionality
- React.js – Component-based architecture
- Axios / Fetch API – Backend communication
- React Router – Client-side routing

11.2. Single Page Application (SPA) Approach
React.js allows:
- Faster navigation without full page reload.
- Efficient DOM updates through Virtual DOM.
- Component reusability.
- Better user experience.

11.3. Role-Based Dashboard UI
Each role has a dedicated dashboard layout:

Admin Dashboard
- Total students
- Total teachers
- Fee summary
- Admission statistics
- Navigation to management modules

Teacher Dashboard
- Assigned classes
- Attendance marking
- Marks upload panel

Student Dashboard
- Attendance percentage
- Examination results
- Notifications

Parent Dashboard
- Child academic progress
- Fee payment status
- Attendance record

CRM Dashboard
- New enquiries
- Follow-up tracking
- Conversion status

11.4. Navigation Structure
The application includes:
- Header (App Name + Logout)
- Sidebar (Role-specific menu)
- Main Content Area
- Footer (Optional)

Navigation is handled using React Router, ensuring smooth transitions between modules.

11.5. Form Handling and Validation
Frontend validation includes:
- Required field validation
- Email format validation
- Numeric input validation
- Password strength checks

This reduces invalid API requests and improves usability.

12. Implementation Details
The system is implemented using the MERN stack architecture.

12.1. Backend Implementation
- Node.js server created using Express framework.
- Middleware implemented for:
- JWT verification

- Role validation
- Error handling
- Mongoose used for schema definition.

12.2. Real-Time Data Handling

Although the system does not use Web Sockets, real-time updates are simulated through:

- Immediate API re-fetch after operations.
- React state management.
- Dynamic UI refresh without reload.

12.3. Business Logic Examples

1. Teacher attendance marking restricted to assigned classes.
2. Parent access limited to linked student record.
3. Enquiry conversion automatically creates student record.
4. Fee status dynamically calculated from payment entries.

13. Testing Strategy

Testing ensures reliability and correctness.

13.1. Backend Testing

Performed using Postman.

Test Cases:

- Valid login
- Invalid login
- Unauthorized access attempt
- CRUD operations validation
- Role-restricted API access

13.2. Frontend Testing

- Form validation testing
- Dashboard rendering
- Role-based routing
- Error message display

13.3. Integration Testing

- API communication verified.
- JWT token transmission validated.
- Database consistency confirmed.

14. Performance Evaluation

The system performance is evaluated based on:

14.1. Functional Efficiency

- Reduced manual paperwork.
- Faster report generation.
- Automated admission tracking.
- Real-time data reflection.

14.2. Security Performance

- Encrypted password storage.
- Stateless authentication.
- Role-based authorization.
- Prevention of unauthorized API access.

14.3. Scalability Evaluation

MongoDB enables:

- Flexible schema updates.
- Horizontal scalability.
- Handling increasing user load.

14.4. User Experience Evaluation

- Fast page navigation.
- Clean dashboard layout.
- Reduced waiting time.
- Clear error messages.

15. Comparative Analysis

The proposed system is compared with traditional systems and fragmented digital solutions.

Feature	Traditional System	Fragmented Digital Tools	Proposed System
Centralized Data	✗	✗	✓
Role-Based Access	✗	Partial	✓
Real-Time Updates	✗	✗	✓
CRM Integration	✗	✗	✓
Security (JWT)	✗	✗	✓
Scalable Architecture	✗	Limited	✓

The proposed portal significantly improves operational efficiency and security.

16. Advantages

1. Centralized database management.
2. Reduced manual effort.
3. Improved transparency.
4. Secure authentication.
5. Real-time access to information.
6. Structured admission tracking.
7. Role-based data control.
8. Scalable and modular design.
9. Cross-device accessibility.
10. Improved decision-making capability.

17. Limitations

Despite its advantages, the system has certain limitations:

1. Requires stable internet connectivity.
2. No mobile application version yet.
3. Advanced analytics not integrated.
4. No AI-based prediction model implemented.
5. Limited multi-branch management features.

18. Future Scope

The future scope of this research includes:

1. Integration of AI-based performance analytics.
2. Mobile application development (React Native).
3. Multi-branch institution management.
4. SMS & Email automation system.
5. Online fee payment gateway integration.
6. Biometric attendance integration.
7. Cloud deployment for global accessibility.
8. Advanced data visualization dashboards.

19. Results

The implementation of the Centralized School Administration & CRM Portal was evaluated through functional testing, role-based validation, simulated institutional workflow scenarios, and comparative analysis with traditional systems.

19.1. Functional Implementation Results

All core modules were successfully implemented and tested:

- User authentication with JWT
- Role-based dashboard redirection

- Student management (CRUD operations)
- Attendance marking and retrieval
- Fee management and payment tracking
- CRM enquiry tracking and conversion
- Role-restricted API validation

All APIs returned structured JSON responses with proper HTTP status codes (200, 400, 401, 403, 500).

System testing confirmed:
- 100% successful role-based routing
- Secure token-based API access
- Accurate database updates
- Correct restriction of unauthorized requests

19.2. CRM Conversion Performance
The CRM module was evaluated using simulated admission enquiry data.

Test Scenario:
- 50 mock enquiries entered
- Follow-ups recorded
- Enquiry-to-student conversion executed

Results:
- 100% accurate data conversion from enquiry to student record
- Automatic student record generation after conversion
- No data duplication observed
- Centralized tracking reduced missed follow-ups

This demonstrates the effectiveness of integrating CRM functionality within school administration.

19.3. Workflow Automation Impact
Compared to manual systems:

Operation	Manual System	Proposed System
Attendance Report Generation	15–20 minutes	< 5 seconds
Fee Status Check	Manual ledger search	Instant dashboard view
Admission Tracking	Paper-based register	Structured CRM tracking
Student Record Update	Rewriting records	One-click update

Estimated efficiency improvement: **60–75% reduction in administrative time**.

19.4. Security Validation Results
Security testing included:
- Invalid login attempts
- Expired JWT token access
- Role mismatch API calls
- Unauthorized endpoint access

Results:
- Unauthorized access blocked successfully
- Expired tokens rejected
- Role validation enforced strictly
- Password hashing verified

No sensitive data was exposed during testing.

19.5. Scalability Observations
MongoDB document-based schema allowed:
- Flexible record insertion
- Efficient query performance
- Smooth handling of increasing user records

Tested with:
- 500+ student records
- 200+ attendance entries
- Multiple simultaneous API requests

System performance remained stable without noticeable delay.

19.6. User Experience Results
Based on UI testing:
- Fast navigation (SPA architecture)
- No full-page reloads
- Clean dashboard design
- Clear error handling
- Role-based intuitive menus

User interaction time reduced significantly compared to traditional desktop-based systems.

19.7. Overall Result Summary
The proposed system successfully achieved:
- Centralized data management
- Secure authentication and authorization
- Real-time dashboard updates
- Structured CRM-based admission tracking
- Improved workflow automation
- Enhanced transparency and accountability

The experimental implementation validates that the system is practical, scalable, secure, and suitable for real-world educational institutions.

20. Conclusion
The Centralized School Administration & CRM Portal is a way to deal with the problems that schools face today. It brings together all the parts of running a school into one safe website. This makes it easier for schools to get things done be more open and keep their information accurate.

The system is very secure because it uses keys to make sure only the right people can get in. It is also built to grow and change as needed. This is because it uses the stack, which is a set of tools that helps make websites bigger and better.

When we compare this system to the way of doing things we can see that it is much better. It helps schools keep track of everything in one place. It makes the process of admitting new students more organized. This system is perfect, for schools, colleges and other places that teach people and want to use technology to make their jobs easier.

We found out that using web technologies can really help simplify the work that schools do every day. It can also help keep everything working well even when the school is growing and changing. The Centralized School Administration & CRM Portal is an example of this.

21. References
[1] MongoDB Inc., "MongoDB Documentation," Available: https://www.mongodb.com

[2] OpenJS Foundation, "Node.js Documentation," Available: https://nodejs.org

[3] Express.js, "Express Web Framework," Available: https://expressjs.com

[4] Meta Platforms Inc., "React.js Official Documentation," Available: https://react.dev

[5] M. Richardson and S. Ruby, *RESTful Web Services*, O'Reilly Media, 2007.

[6] J. Resig and B. Bibeault, *Secrets of the JavaScript Ninja*, Manning Publications, 2016.

[7] E. Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[8] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[9] D. Crockford, *JavaScript: The Good Parts*, O'Reilly Media, 2008.

[10] A. Banks and E. Porcello, *Learning React*, O'Reilly Media, 2020.

[11] OWASP Foundation, "Web Application Security Testing Guide," Available: https://owasp.org

[12] NIST, "Digital Identity Guidelines," National Institute of Standards and Technology, 2017.

[13] K. Schwaber and J. Sutherland, "The Scrum Guide," 2020.

[14] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral Dissertation, University of California, 2000.

[15] J. Ferraiolo et al., "Role-Based Access Control," *NIST Standard*, 2001.