**Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Gopalganj-8100**

**Project Requirement Analysis on**

**“Institute Management System”**

**Course Code: CSE 310**

**SUBMITTED GROUP BY**

              M. A. Forhad Aziz, ID: 20CSE007

Mohammad Nimour Hossain, ID: 20CSE010

  Jahid Hasan,   ID: 20CSE037

Session: 2020-21

Department of Computer Science and Engineering

**SUBMITTED TO**

Dr. Syful Islam,

Assistant Professor.

Department of Computer Science and Engineering

**Table of Contents**

[**1. INTRODUCTION** 4](#_Toc185259594)

[***1.1 Purpose of the Institute Management System*** 4](#_Toc185259595)

[***1.2 Scope of the System*** 4](#_Toc185259596)

[***1.3 Goals of the Development Team*** 5](#_Toc185259597)

[***1.4 Development Process Model*** 5](#_Toc185259598)

[***1.5 Team Roles and Organization*** 6](#_Toc185259599)

[**2. RESEARCH** 7](#_Toc185259600)

[***2.1 Literature Survey and Technical Research*** 7](#_Toc185259601)

[2.1.1 User Interface Design: 7](#_Toc185259602)

[2.1.2 Data Management: 8](#_Toc185259603)

[2.1.3 Authentication and Security: 8](#_Toc185259604)

[2.1.4 Performance and Scalability: 9](#_Toc185259605)

[***2.2 Technology Stack:*** 10](#_Toc185259606)

[**3. DESCRIPTION** 11](#_Toc185259607)

[***3.1 System Modules*** 11](#_Toc185259608)

[3.1.1 Student Management Module 11](#_Toc185259609)

[3.1.2 Course Management Module 11](#_Toc185259610)

[3.1.3 Faculty Management Module 12](#_Toc185259611)

[3.1.4 Timetable Management Module 12](#_Toc185259612)

[3.1.5 Examination Management Module 12](#_Toc185259613)

[3.1.6 Fee Management Module 13](#_Toc185259614)

[***3.2 User Roles and Permissions*** 13](#_Toc185259615)

[3.2.1 Admin Role 13](#_Toc185259616)

[3.2.2 Faculty Role 13](#_Toc185259617)

[3.2.3 Student Role 14](#_Toc185259618)

[***3.3 System Workflow*** 14](#_Toc185259619)

[***3.4 System Security Features*** 14](#_Toc185259620)

[**4. REQUIREMENTS** 16](#_Toc185259621)

[***4.1 Functional Requirements*** 16](#_Toc185259622)

[4.1.1 Menu Requirements 16](#_Toc185259623)

[4.1.2 In-System Functions 17](#_Toc185259624)

[***4.2 Structural Requirements*** 18](#_Toc185259625)

[4.2.1 Database Structure 18](#_Toc185259626)

[4.2.2 User Interface (UI) 18](#_Toc185259627)

[4.2.3 Networking 18](#_Toc185259628)

[4.2.4 Security 19](#_Toc185259629)

[4.2.5 Integration 19](#_Toc185259630)

[***4.3 Performance Requirements*** 19](#_Toc185259631)

[***4.4 Software Requirements*** 19](#_Toc185259632)

[***4.5 Hardware Requirements*** 20](#_Toc185259633)

[***4.6 Non-Functional Requirements*** 20](#_Toc185259634)

[4.6.1 Security 20](#_Toc185259635)

[4.6.2 Maintainability 20](#_Toc185259636)

[4.6.3 Portability 20](#_Toc185259637)

[***4.7 Project Constraints*** 20](#_Toc185259638)

[**5. SYSTEM ANALYSIS AND MODELING** 21](#_Toc185259639)

[***5.1 Functional Modeling*** 21](#_Toc185259640)

[5.1.1 Level 0 of Data Flow Diagram 21](#_Toc185259641)

[5.1.2 Level 1 of Data Flow Diagram 21](#_Toc185259642)

[***5.2 Use Case Analysis*** 22](#_Toc185259643)

[5.2.1 Start Menu Use Case 22](#_Toc185259644)

[5.2.2 In-System Use Case 23](#_Toc185259645)

[**6.GANTT CHART** 25](#_Toc185259646)

# **1. INTRODUCTION**

The **Institute Management System (IMS)** is designed to improve the efficiency and functionality of administrative tasks in educational institutions. The system will streamline various operational processes such as managing student information, scheduling courses and examinations, tracking fee payments, and handling user roles within the institution. Built on the **MERN stack** (MongoDB, Express, React, and Node.js), the IMS will offer an intuitive, scalable, and secure solution for students, faculty, and administrators, significantly enhancing their interaction with the institution.

## ***1.1 Purpose of the Institute Management System***

The primary purpose of the **Institute Management System (IMS)** is to automate and simplify the administrative processes of an educational institution. It will act as a centralized platform that handles crucial operations in a seamless manner, from student enrollment to examination management. Key objectives include:

* **Student Management:** The IMS will store and manage student data such as enrollment details, attendance records, course registration, grades, and academic progress.
* **Course Management:** It will provide tools for creating and managing courses, assigning instructors, and scheduling lectures or classes for students.
* **Examination Management:** The system will manage examination schedules, monitor exam results, and track assignments, enabling faculty to evaluate students effectively.
* **Fee Management:** The IMS will automate fee collection, track pending fees, generate invoices, and maintain detailed records of all payments made.
* **Role-based User Access:** The system will provide distinct roles for administrators, faculty, and students, ensuring appropriate access levels for each user type.

The **MERN stack** will be employed to ensure that the system is not only secure but also highly scalable and capable of handling the growing demands of educational institutions in the future. The end goal is to create an efficient, user-friendly platform that simplifies institutional management and enhances communication among all stakeholders.

## ***1.2 Scope of the System***

The scope of the Institute Management System is broad, covering a wide range of features aimed at addressing various operational needs of an educational institution:

* **Student Management:** The system will allow the registration of students, maintaining their personal details, attendance records, academic history, and course registration. Students will also be able to track their performance, including grades and assignments.
* **Course Management:** The IMS will allow for the creation and management of courses, assigning instructors, and scheduling classes. It will track which courses students are enrolled in and allow faculty members to access relevant course details.
* **Examination Management:** The system will handle the scheduling of examinations and assignments, including results processing. It will store and display student performance records and help faculty evaluate their students effectively.
* **Fee Management:** A key feature of IMS is the ability to manage fee structures, keep track of payments, generate invoices, and provide students with reminders for pending fees. It will support multiple fee categories (tuition, lab fees, library fees, etc.) and allow admins to manage fee details.
* **User Roles and Permissions:** The system will include role-based access control to manage what information and actions are available to each user. Admins will have complete access, faculty will be able to manage courses and assessments, and students will have access to their academic details and results.
* **Security:** The system will implement security measures such as encryption and secure authentication to protect sensitive data and maintain privacy. Role-based access control ensures that each user can only access what is necessary for their role.
* **Real-Time Communication:** The system will provide real-time communication features such as notifications for assignment deadlines, examination schedules, fee reminders, and general announcements to ensure smooth interaction between students, faculty, and administration.
* **System Integration:** The system will enable online course registration, result viewing, and report generation, making it a comprehensive platform for academic management. It will also support the integration of third-party tools if needed for enhanced functionality.

## ***1.3 Goals of the Development Team***

The development team's primary goals for this project are:

* **Full-Stack Development Mastery:** The team aims to gain in-depth knowledge and hands-on experience in using the **MERN stack** (MongoDB, Express, React, and Node.js) to build a complete, functional system. Mastery of full-stack development is essential for delivering a seamless and efficient system that meets the needs of the educational institution.
* **Effective Documentation and Version Control:** The team will employ proper project documentation practices and use version control systems such as **Git** to ensure that the development process is well-managed and organized. This will help in tracking progress and handling collaborative changes effectively.
* **Efficient Time Management:** The team will focus on managing their time efficiently, dividing the project into well-defined phases and tasks, prioritizing key features, and ensuring timely completion of deliverables.
* **Teamwork and Collaboration:** The project will be a collaborative effort, with the team members sharing responsibilities and contributing equally to both front-end and back-end development. They will engage in regular communication and feedback sessions to ensure smooth project progression and problem-solving.
* **Scalability and Maintainability:** The final IMS should not only solve current problems but also be scalable and maintainable. The team will focus on writing clean, modular code that can be easily expanded with new features in the future.
* **Creating a Robust System:** The team will prioritize the security, performance, and user-friendliness of the IMS. They will ensure that it meets all functional requirements while providing a smooth experience for users.

## ***1.4 Development Process Model***

The project will be developed following the **Agile methodology**, which emphasizes iterative and incremental development. This model allows for continuous feedback and improvement, making it highly suitable for projects with evolving requirements. Specifically, the **Scrum framework** will be used to break the project into multiple **sprints**, each focusing on a specific feature or module of the IMS.

* **Sprint Planning:** At the beginning of each sprint, tasks will be identified, prioritized, and assigned based on their importance and complexity.
* **Daily Stand-ups:** The team will conduct short daily meetings to discuss progress, identify blockers, and plan the next steps.
* **Sprint Review and Retrospective:** At the end of each sprint, the team will present the completed work to stakeholders and review what went well and what could be improved in the next sprint.

This approach allows for greater flexibility and adaptability throughout the development process, ensuring that the final system is both functional and user-centric.

## ***1.5 Team Roles and Organization***

The IMS project is a collaborative effort among three team members who share responsibility for both front-end and back-end development, ensuring a balanced workload and efficient collaboration.

* **Frontend Development:** One of the team members will focus on building the user interface using **React**. They will design a responsive and intuitive interface, ensuring that the system is user-friendly for all stakeholders (students, faculty, and administrators).
* **Backend Development:** Another team member will focus on server-side development, working with **Node.js** and **Express** to handle API development and database management. They will ensure the integrity of data and maintain smooth communication between the front-end and back-end.
* **Database Management and Integration:** The third team member will work on setting up and managing the **MongoDB** database, ensuring efficient data storage and retrieval. They will handle user authentication, data encryption, and implement role-based access control.

The team operates under a flat hierarchy, with all members having equal responsibility and authority over decisions related to the project. The team leader ensures that the project stays on track by managing timelines, delegating tasks, and keeping all members aligned on the overall project goals.

Regular meetings and brainstorming sessions will help the team identify any issues early on and make necessary adjustments. Each member will contribute their expertise, ensuring the successful delivery of the IMS.

# **2. RESEARCH**

The development of the **Institute Management System (IMS)** requires a solid foundation of research to identify suitable technologies, architectural approaches, and design patterns. This section explores the findings from literature surveys and technical research, highlighting the considerations for user interface design, data management, authentication, security, performance, and scalability. Additionally, it elaborates on the chosen technology stack and its relevance to the project.

## ***2.1 Literature Survey and Technical Research***

### 2.1.1 User Interface Design:

For a system like the **Institute Management System**, an intuitive, user-friendly, and responsive interface is essential. The UI must accommodate diverse users, including administrative staff, faculty, and students, each requiring tailored access and interaction points. Modern web development practices emphasize **dynamic and interactive interfaces**, making **React** (a core component of the MERN stack) the ideal choice for building the front-end.

Key considerations for UI design include:

1. **Framework Selection:**
   * **React:** React will enable the creation of reusable components, facilitating consistent design across all pages. Its virtual DOM ensures high performance, even for complex, data-heavy pages like student dashboards and fee management.
   * **Material-UI (MUI):** A React-based library offering pre-designed, customizable UI components such as buttons, tables, and form inputs. It will help maintain a professional, modern aesthetic.
   * **Bootstrap for React:** Bootstrap’s grid system and responsive design features will ensure the application works seamlessly on devices of all sizes, from desktops to smartphones.
2. **UI Features:**
   * **Clear Navigation Paths:** Navigation menus will guide users to features like managing student data, viewing courses, submitting assignments, and processing fee payments. Icons and tooltips will enhance usability.
   * **Real-Time Updates:** React’s state management will provide real-time feedback for actions like form submissions, course registrations, or attendance updates.
   * **Search and Filters:** Powerful search and filtering mechanisms will allow users to quickly access specific student records, courses, or fee payment histories.
   * **Form Validations:** The system will include client-side and server-side validation for all input fields, ensuring data integrity before submission.
   * **Custom Dashboards:** Tailored dashboards for admins, faculty, and students will highlight relevant information, such as course details, fee status, or announcements.

The UI design process will involve wireframing and prototyping to gather feedback and ensure the design aligns with user expectations before full-scale development.

### 2.1.2 Data Management:

Efficient data management is the backbone of the **Institute Management System**, as it involves handling extensive, diverse datasets. These include student records, faculty information, course schedules, fee transactions, and examination results. **MongoDB**, a NoSQL database, was selected for its flexibility and performance in managing large, dynamic datasets.

Key aspects of data management include:

1. **Database Structure:**
   * **Document-Oriented Model:** MongoDB’s JSON-like document structure makes it ideal for storing complex, hierarchical data such as student profiles with nested attributes (e.g., personal details, enrollment history, fee status).
   * **Dynamic Schema:** MongoDB supports schema-less design, allowing the system to adapt to evolving data requirements without major disruptions.
2. **Data Validation and Consistency:**
   * **Mongoose:** This ODM (Object Data Modeling) library for MongoDB will be used to define schemas, enforce data validation, and manage relationships between data entities (e.g., linking students to their enrolled courses).
3. **Sensitive Data Handling:**
   * **Encryption:** Sensitive information, such as passwords and financial records, will be encrypted both at rest and during transmission to prevent unauthorized access.
   * **Data Backup and Recovery:** Regular backups will ensure data safety, and recovery procedures will be established to prevent data loss in case of system failures.
4. **Data Querying and Performance:**
   * Indexing will be implemented to optimize search operations, ensuring that the system can quickly retrieve data, such as searching for a student by name or ID.
   * Aggregation pipelines will process complex queries, such as generating fee summaries for an academic year.

### 2.1.3 Authentication and Security:

Ensuring a secure and reliable authentication mechanism is critical for the **IMS**, as it deals with sensitive personal and institutional data. The system will use **JWT (JSON Web Tokens)** for authentication and session management, paired with robust security practices.

1. **Authentication Approach:**
   * **JWT:** JWTs will provide stateless authentication, allowing secure user sessions without overloading the server with session data. Tokens will be signed with secret keys and validated for every user request.
   * **Role-Based Access Control (RBAC):** Role-based permissions will ensure that each user (admin, faculty, or student) can access only the features and data relevant to their role.
2. **Password Security:**
   * **Bcrypt:** Passwords will be hashed using bcrypt before storage, adding a layer of security against potential data breaches.
3. **Server and Application Security:**
   * **Helmet.js:** This middleware will set secure HTTP headers, protecting the system against common web vulnerabilities like cross-site scripting (XSS) and clickjacking.
   * **Rate Limiting:** To prevent brute force attacks, rate limiting will restrict the number of login attempts from a single IP address.
4. **Audits and Testing:**
   * Regular security audits will identify and mitigate vulnerabilities.
   * Penetration testing will simulate real-world attacks to ensure the system’s defenses are robust.

### 2.1.4 Performance and Scalability:

Performance optimization and scalability are vital for a system expected to handle growing datasets and concurrent user interactions. The IMS will use technologies and practices that ensure smooth performance during peak loads and allow for future growth.

1. **Backend Performance:**
   * **Node.js:** Node.js’s non-blocking, event-driven architecture allows the backend to handle multiple requests concurrently, reducing latency and improving user experience.
   * **Asynchronous Programming:** Asynchronous operations, such as database queries and API calls, will ensure efficient use of resources.
2. **Database Scalability:**
   * **Horizontal Scaling:** MongoDB’s ability to distribute data across multiple servers will allow the system to grow as the institution expands.
   * **Sharding:** Data sharding will distribute datasets across different servers, improving query performance and reducing load on individual nodes.
3. **System Scalability:**
   * **Load Balancing:** Load balancers will distribute traffic evenly across servers, ensuring consistent performance during high-demand periods.
   * **Microservices Architecture:** The system may be modularized into independent microservices (e.g., authentication, fee management) to simplify scaling specific features as needed.

## ***2.2 Technology Stack:***

The **MERN stack** was selected for the IMS due to its comprehensive, JavaScript-based ecosystem, which ensures seamless communication between the front-end and back-end. Each component of the stack contributes to the overall system’s functionality:

1. **MongoDB:**
   * A NoSQL database designed for high performance and scalability.
   * Used for storing all system data, including student records, course schedules, examination results, and fee transactions.
   * Provides flexibility in handling structured and unstructured data.
2. **Express:**
   * A minimalist web application framework for Node.js that simplifies routing and middleware management.
   * Handles backend API development, managing communication between the database and front-end.
3. **React:**
   * A JavaScript library for building the user interface.
   * Enables the creation of reusable, interactive components and ensures a responsive, dynamic front-end.
4. **Node.js:**
   * A runtime environment for executing server-side JavaScript.
   * Manages API requests, authentication, and real-time data exchange between the client and server.

The seamless integration of these technologies ensures the development of a scalable, secure, and user-friendly Institute Management System.

# **3. DESCRIPTION**

The **Institute Management System (IMS)** is a comprehensive platform designed to streamline the administrative, academic, and financial operations of an educational institution. This section provides an in-depth description of the system’s primary modules, the roles and permissions granted to users, the overall workflow of the system, and its security features.

## ***3.1 System Modules***

The IMS consists of several modules, each focusing on a specific aspect of institutional management. These modules are designed to work cohesively, ensuring an efficient and user-friendly system.

### 3.1.1 ****Student Management Module****

This module handles all student-related functionalities, including registration, profile management, academic tracking, and attendance monitoring.

**Key Features:**

* **Student Registration:** Enables administrators to register new students, capturing details such as personal information, contact details, academic history, and enrollment information.
* **Profile Management:** Allows students to view and update their personal details, ensuring their information is accurate and up-to-date.
* **Attendance Tracking:** Facilitates attendance recording for each student, helping faculty monitor class participation and identify irregularities.
* **Progress Tracking:** Provides detailed reports of student performance, including grades, attendance, and participation.
* **Transcripts and Reports:** Automates the generation of academic transcripts and progress reports, which can be downloaded by students or administrators.

### 3.1.2 ****Course Management Module****

This module facilitates the creation, organization, and monitoring of academic courses offered by the institution.

**Key Features:**

* **Course Creation and Updates:** Administrators can create, update, or delete courses, including details like course codes, descriptions, and credits.
* **Instructor Assignment:** Assign faculty members to specific courses based on their expertise and availability.
* **Student Enrollment:** Tracks which students are enrolled in which courses, allowing real-time updates of class rosters.
* **Course Material Distribution:** Faculty can upload course materials, such as lecture notes and assignments, which students can download or access directly from the system.