Student Management System

A MINOR PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of Degree of Integrated Master of Computer Applications (IMCA)

Submitted to

RAJIV GANDHI PRODYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)

Submitted by:

Mr. Vatsalya Vyas
Enrollment No. 0827CA21DD62

Under the Supervision of

Dr. Nidhi Dahale / Prof. Smita Vijayvargiya

INTEGRATED MASTER OF COMPUTER APPLICATIONS

ACROPOLIS INSTITUTE OF TECHNOLOGY & RESEARCH, INDORE

SESSION May'2024

Faculty of Computer Applications

AITR, Indore

BONAFIDE CERTIFICATE

This is to certify that Minor Project entitled **“Student Management System”** being submitted by **Mr. Vatsalya Vyas (0827CA21DD62)** for partial fulfillment of the requirement for the award of degree of Integrated **Master of Computer Applications** to **Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P)** during the academic year 2024 is a record of bonafide piece of work, carried out by the student under my supervision and guidance in the **Faculty of Computer Applications, AITR, Indore.**

(Supervisor)	**(Prof. Geeta Santhosh)**
Designation, AITR, Indore	Professor & HOD, MCA

(Internal Examiner)	**(External Examiner)**
Date: 06/07/2024	**Date: 06/07/2024**

ACROPOLIS INSTITUTE OF TECHNOLOGY & RESEARCH, INDORE

(Faculty of Computer Applications)

ABSTRACT

The **Student Management System** is a centralized platform designed to streamline the management of students, faculty, classes, and subjects. This system provides specialized functionalities and access permissions tailored to the roles of various users, ensuring an efficient workflow and seamless collaboration. By integrating essential management tasks into a single platform, the system enhances overall administrative efficiency and supports the academic institution in maintaining accurate and up-to-date records. This project aims to facilitate the management processes within educational institutions, providing a user-friendly interface and robust functionalities to meet the diverse needs of its users.

ACROPOLIS INSTITUTE OF TECHNOLOGY & RESEARCH, INDORE

(Faculty of Computer Applications)

ACKNOWLEDGEMENT

I would like to use this opportunity to extend my heartfelt gratitude to our guide and supervisor of this project, **Dr. Nidhi Dahale / Prof. Smita Vijayvargiya** who is responsible for letting us explore this idea.

I am grateful to **Prof. Geeta Santosh**, Head of the Department of Faculty of Computer Applications, for her cooperation and support in completing this work. I would also like to express my gratitude to **Dr. S.C. Sharma**, Director of AITR Indore, for providing necessary facilities. Additionally, I would like to convey my thanks to all the staff members of the Faculty of Computer Applications for their help and support**.**

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations or Nomenclature

This is a list of Abbreviations, or Nomenclature used in the Student Management System (SMS) Project

Abbreviations

- **SMS** : Student Management System
- **MERN** : MongoDB, Express.js, React.js, Node.js
- **UI** : User Interface
- **UX** : User Experience
- **API** : Application Programming Interface
- **CRUD** : Create, Read, Update, Delete
- **DB** : Database
- **MVC** : Model-View-Controller
- **JWT** : JSON Web Token
- **REST** : Representational State Transfer
- **HTTP** : HyperText Transfer Protocol
- **HTTPS** : HyperText Transfer Protocol Secure

Nomenclature

- **Admin** : User with highest level of control, responsible for managing users and system settings.
- **Teacher** : User responsible for managing student records, attendance, and grades.
- **Student** : User who can view their own records, attendance, and academic performance.
- **Schema** : The structure of a database, described in a formal language.
- **Middleware** : Software that acts as a bridge between an operating system or database and applications, especially on a network.
- **Frontend** : The part of a website or web application that users interact with directly.
- **Backend** : The server-side of a web application responsible for business logic and database interactions.

(Chapter 1)

INTRODUCTION

In the digital age, educational institutions are increasingly seeking efficient and effective ways to manage their operations and enhance communication among stakeholders. The Student Management System (SMS) represents a robust solution tailored to meet these needs by streamlining student, faculty, class, and subject management. Developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), this web-based application leverages modern web technologies to offer a comprehensive suite of features aimed at optimizing administrative tasks and promoting academic excellence.

The primary objective of the Student Management System is to provide a centralized platform that facilitates efficient management and seamless collaboration within educational institutions. By offering specialized functionalities and access permissions tailored to the roles of admins, teachers, and students, the system ensures a smooth workflow and effective communication.

1.1 Brief Description

The Student Management System is a comprehensive web-based application designed to optimize the management of educational institutions. Utilizing the MERN stack (MongoDB, Express.js, React.js, Node.js), this system provides a centralized platform for managing students, faculty, classes, and subjects efficiently. It offers specialized functionalities and access permissions tailored to different user roles, including Admin, Teacher, and Student, ensuring a smooth workflow and seamless collaboration.

Key features include a robust admin dashboard for comprehensive management, attendance tracking, data visualization tools for academic performance insights, and a dedicated section for notices and complaints to enhance communication between stakeholders. The system aims to streamline administrative tasks, foster effective communication, and support academic excellence through its user-friendly interface and powerful features.

By integrating modern web technologies and adhering to best practices in software development, the Student Management System delivers a scalable and efficient solution for educational institutions, promoting operational efficiency and academic success.

1.2 Objective

The primary objectives of the Student Management System are to streamline educational institution operations, optimize student management, and facilitate effective communication among stakeholders. Specifically, the system aims to:

i. **Streamline Administrative Tasks:**

- Simplify the management of students, faculty, classes, and subjects through a centralized platform.
- Provide a comprehensive admin dashboard for efficient handling of administrative duties.

ii. **Enhance Communication:**

- Facilitate effective communication between students, teachers, and administrators.
- Offer dedicated sections for posting notices and handling complaints.

iii. **Improve Academic Tracking:**

- Enable teachers to mark and view attendance history easily.
- Provide interactive data visualization tools to help students gain insights into their academic performance and encourage self-awareness and continuous improvement.

iv. **Ensure Role-Based Access:**

- Implement secure and appropriate access permissions based on user roles (Admin, Teacher, Student) to ensure data integrity and privacy.
- Empower users with specialized functionalities tailored to their specific needs and responsibilities.

v. **Leverage Modern Technologies:**

- Utilize the MERN stack (MongoDB, Express.js, React.js, Node.js) to build a scalable, efficient, and user-friendly web application.
- Ensure the system is built using best practices in software development for robustness and maintainability.

By achieving these objectives, the Student Management System aims to enhance the overall efficiency of educational institutions, support academic excellence, and promote a collaborative and organized environment for all stakeholders.

1.3 Scope

The scope of the Student Management System encompasses the development and deployment of a comprehensive web-based application designed to address the administrative and academic needs of educational institutions. The system is intended to provide a centralized platform for managing students, faculty, classes, and subjects efficiently while facilitating effective communication among all stakeholders. The key areas covered by the scope include:

i. **User Management:**

- Admin: Manage student and faculty records, class schedules, and subject assignments. Oversee the overall system operations and ensure data integrity.
- Teacher: Access class schedules, manage student attendance, and view performance analytics. Communicate with students and administrators through the notice and complaints section.
- Student: View personal academic records, attendance, and performance analytics. Engage in effective communication with teachers and administrators.

ii. **Core Features:**

- Dashboard: A user-friendly interface providing quick access to key functionalities and information relevant to each user role.
- Attendance Tracking: Functionality for teachers to mark attendance and review attendance history.
- Data Visualization: Interactive charts and graphs to help students track their academic performance.
- Notices and Complaints: A dedicated section for posting notices and lodging complaints, ensuring streamlined communication between students, teachers, and administrators.

iii. **Technological Implementation:**

- Frontend: Developed using React.js and Material UI for a responsive and interactive user interface.
- Backend: Built with Node.js and Express.js to handle server-side logic and database interactions.
- Database: MongoDB for efficient and scalable data storage.

iv. **Installation and Deployment:**

- Detailed steps for setting up the system locally, including prerequisites, repository cloning, backend and frontend setup, and environment variable configuration.
- Instructions for starting the backend and frontend servers to run the application.

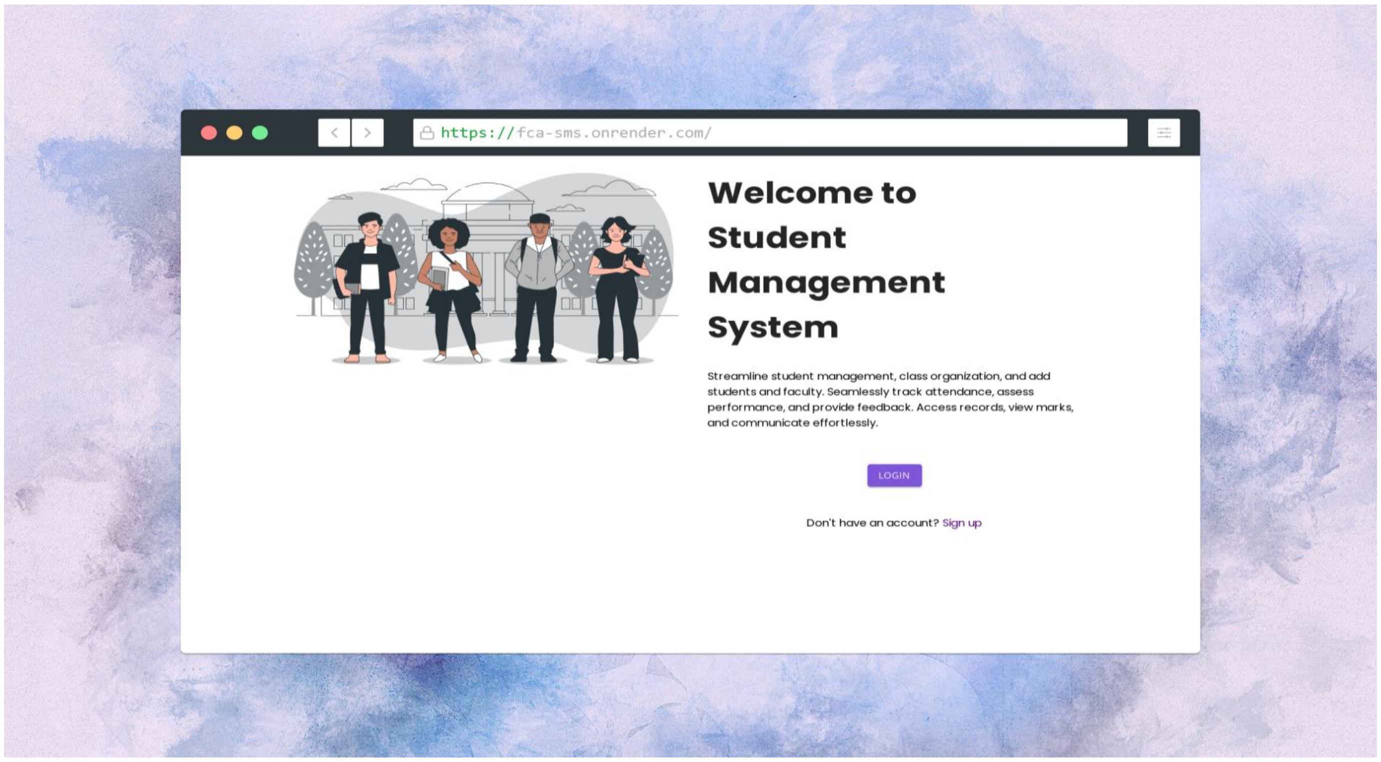
v. Security and Access Control:

- Implementation of role-based access control to ensure that users have appropriate permissions based on their roles (Admin, Teacher, Student).
- Secure handling of user data and authentication mechanisms to protect sensitive information.

vi. Maintenance and Scalability:

- Provision for ongoing maintenance and updates to ensure the system remains robust and functional.
- Scalability considerations to accommodate growing user numbers and additional features in the future.

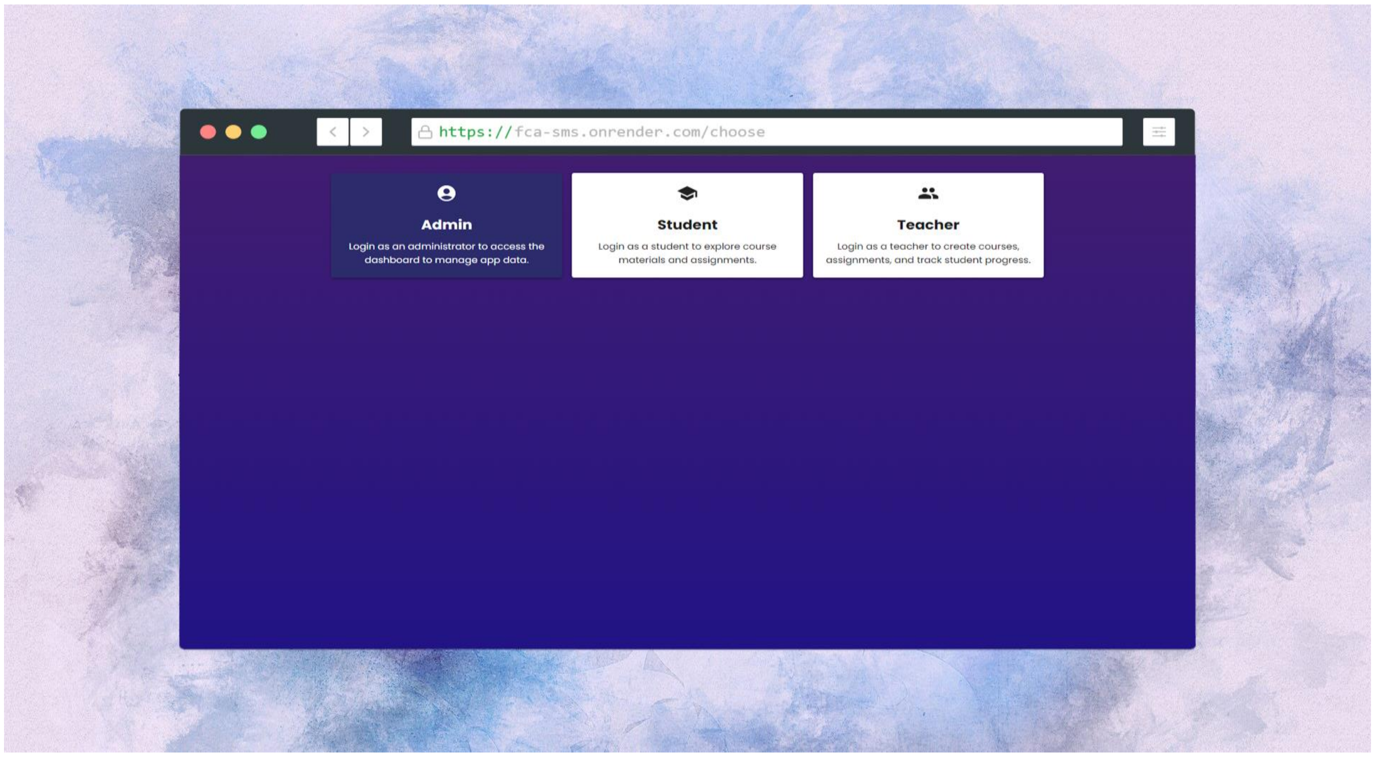
By defining these areas, the scope of the Student Management System ensures a comprehensive, user-friendly, and secure solution tailored to the needs of educational institutions, enhancing their operational efficiency and supporting academic excellence.



Welcome to Student Management System

Streamline student management, class organization, and add students and faculty. Seamlessly track attendance, assess performance, and provide feedback. Access records, view marks, and communicate effortlessly.

Home Page



Admin

Login as an administrator to access the dashboard to manage app data.

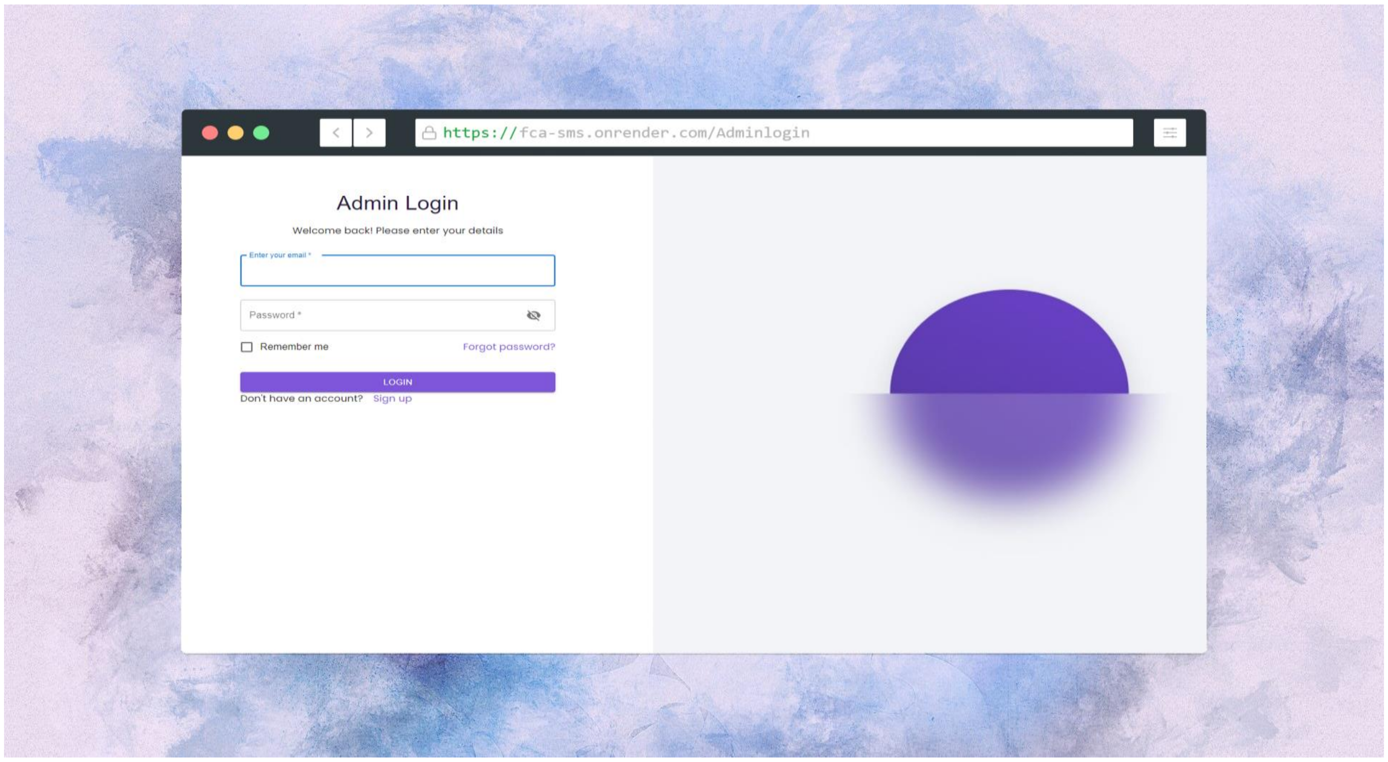
Student

Login as a student to explore course materials and assignments.

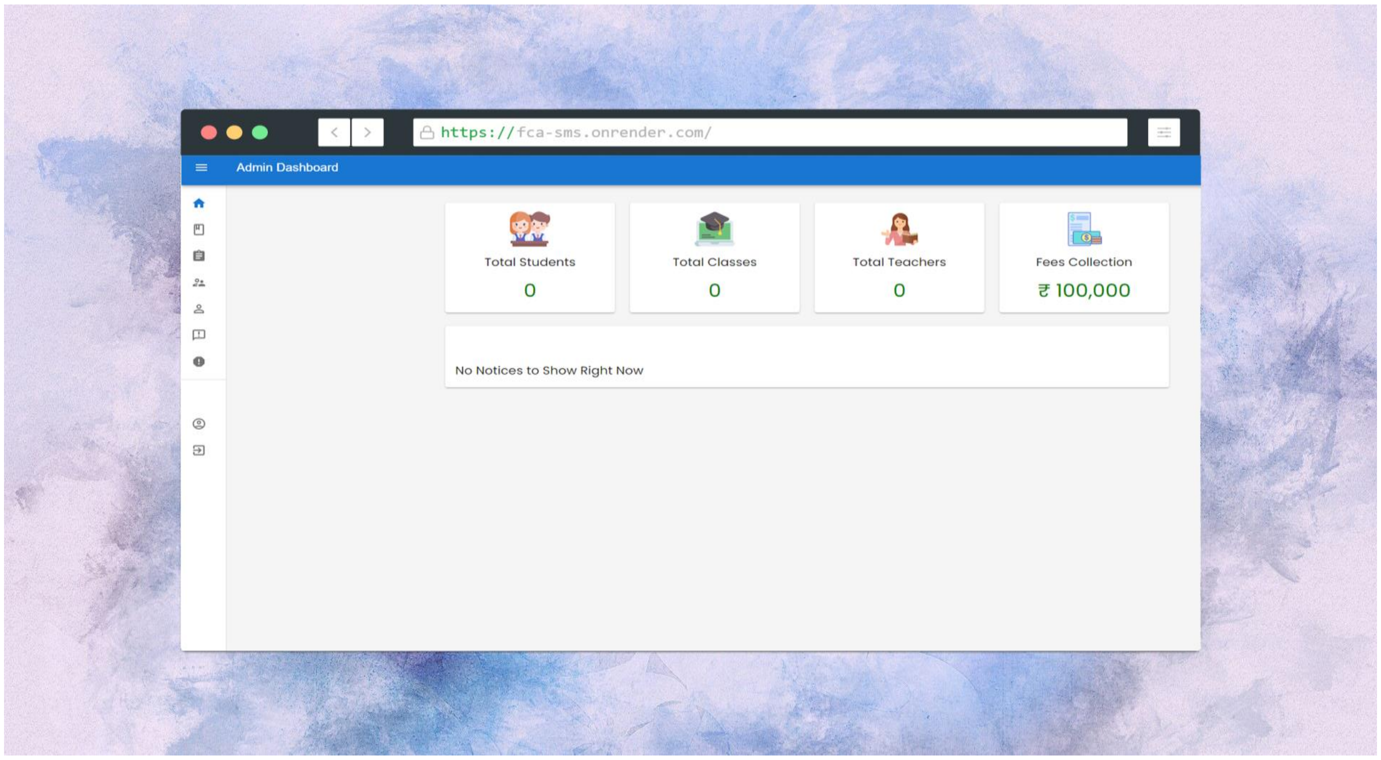
Teacher

Login as a teacher to create courses, assignments, and track student progress.

Login Page


https://fca-sms.onrender.com/Adminlogin
Admin Login
Welcome back! Please enter your details
Enter your email *
Password *
Remember me
Forgot password?
LOGIN
Don't have an account? Sign up

Admin Login



Admin Dashboard

(Chapter 2)

SYSTEM ANALYSIS

System analysis for the Student Management System involves a detailed examination of the system's requirements, its components, and the processes it supports. This analysis ensures that the system design aligns with the needs of its users and fulfills the project's objectives.

i. Requirements Analysis

User Requirements:

- Admin: Manage students, faculty, classes, and subjects. Monitor system operations and generate reports.
- Teacher: Mark and track attendance, manage class-related information, and communicate with students and administrators.
- Student: View personal academic records, track attendance and performance, and communicate with teachers and administrators.

Functional Requirements:
- User Authentication: Secure login and registration for Admin, Teacher, and Student roles.
- Dashboard: Customized dashboards for each user role with relevant information and functionalities.
- Student Management: Add, update, delete, and view student records.
- Faculty Management: Add, update, delete, and view faculty records.
- Class Management: Create and manage classes, assign subjects, and schedule classes.
- Attendance Tracking: Mark daily attendance and generate attendance reports.
- Data Visualization: Interactive charts and graphs to display academic performance and attendance statistics.
- Notice and Complaints: Post notices and handle complaints for effective communication.
- Reporting: Generate reports on various metrics such as attendance, performance, and class schedules.

Non-Functional Requirements:
- Performance: The system should handle multiple concurrent users without significant degradation in performance.
- Usability: The user interface should be intuitive and easy to navigate for all user roles.
- Security: Data encryption, secure authentication, and role-based access control must be implemented to protect sensitive information.

- Scalability: The system should be able to scale to accommodate an increasing number of users and data volume.
- Maintainability: The codebase should be modular and well-documented to facilitate maintenance and future enhancements.

ii. System Components

Frontend:
- React.js: Framework for building the user interface.
- Material UI: Library for implementing responsive and visually appealing design components.
- Redux: State management library to handle application state.

Backend:
- Node.js: JavaScript runtime for executing server-side code.
- Express.js: Web framework for building RESTful APIs.

Database:
- MongoDB: NoSQL database for storing and managing data.

Others:
- RESTful API: Interface for communication between frontend and backend.
- Environment Variables: Configuration settings for different environments (development, testing, production).

iii. Process Analysis

User Authentication:

a) User submits login credentials.
b) Backend verifies credentials against the database.
c) User is granted access and redirected to their respective dashboard based on their role.

Student Management:

a) Admin adds or updates student information through a form.
b) Data is sent to the backend and stored in the MongoDB database.
c) Admin can view and manage the list of students.

Class and Subject Management:

a) Admin creates classes and assigns subjects to them.
b) Class schedules are managed, and teachers are assigned to classes.
c) Information is updated in the database and reflected in the respective dashboards.

Attendance Tracking:

a) Teacher marks attendance for a class.

b) Attendance data is sent to the backend and stored in the database.
c) Attendance reports can be generated and viewed by Admins and Teachers.

Data Visualization:

a) Data is retrieved from the database and processed in the backend.
b) Charts and graphs are generated and displayed on the frontend using visualization libraries.

Notice and Complaints:

a) Users post notices or submit complaints through a form.
b) Data is stored in the database and displayed in the relevant sections.
c) Admins manage and respond to complaints.

iv. Data Flow Diagrams (DFDs)

Level 0 DFD (Context Diagram):
- Depicts the overall system and interactions with external entities (Admins, Teachers, Students).

Level 1 DFD:
- Breaks down the major processes such as User Authentication, Student Management, Class Management, Attendance Tracking, and Communication (Notices and Complaints).

v. Use Case Diagrams

Admin Use Cases:
- Manage Users (Students, Faculty)
- Manage Classes and Subjects
- Generate Reports
- Post Notices

Teacher Use Cases:
- Mark Attendance
- View Attendance Reports
- Post Complaints
- View Class Information

Student Use Cases:
- View Personal Information
- View Attendance and Performance
- Submit Complaints
- View Notices

vi. Entity-Relationship Diagram (ERD)

Entities:
- User: Attributes include UserID, Name, Role, Email, Password.
- Student: Attributes include StudentID, Name, ClassID, PerformanceData.
- Faculty: Attributes include FacultyID, Name, SubjectID, ClassID.
- Class: Attributes include ClassID, ClassName, SubjectID, FacultyID.
- Subject: Attributes include SubjectID, SubjectName.
- Attendance: Attributes include AttendanceID, Date, StudentID, Status.
- Notice: Attributes include NoticeID, Title, Description, Date.
- Complaint: Attributes include ComplaintID, UserID, Description, Status.

Relationships:
- One-to-many relationship between Class and Student.
- One-to-many relationship between Faculty and Class.
- One-to-many relationship between Subject and Class.
- One-to-many relationship between Class and Attendance.
- One-to-many relationship between User and Notice.
- One-to-many relationship between User and Complaint.

Conclusion

The system analysis provides a comprehensive understanding of the Student Management System's requirements, components, and processes. By addressing both functional and non-functional requirements and ensuring a well-structured design, the system aims to meet the needs of educational institutions effectively, enhancing administrative efficiency, communication, and academic performance tracking.

2.1 Feasibility Study

2.1.1 Technical Feasibility

The technical feasibility of the Student Management System involves evaluating whether the proposed technology and architecture can be implemented effectively with the available resources and technical expertise.

Technologies Used:
- Frontend: React.js, Material UI, Redux
- Backend: Node.js, Express.js
- Database: MongoDB

Technical Expertise:
- The development team has proficiency in the MERN stack, which is widely used for building scalable and efficient web applications.
- Availability of robust libraries and frameworks (e.g., Material UI for the frontend, Mongoose for MongoDB) that streamline development.

Infrastructure:
- Requires standard development environments with Node.js and MongoDB installed.
- The application can be hosted on cloud services such as AWS, Azure, or Heroku, which offer scalability and reliability.

Risk Assessment:
- Potential technical challenges with integration and data synchronization between frontend and backend.
- Mitigation strategies include thorough testing, code reviews, and using established best practices in development.

2.1.2 Economic Feasibility

The economic feasibility assesses the cost implications of developing and maintaining the Student Management System compared to the benefits it provides.

Cost Analysis:
- Initial Costs: Development time, software licenses (if any), and hosting services.
- Ongoing Costs: Maintenance, updates, and potential scaling as user numbers increase.

Cost-Benefit Analysis:

- Benefits: Increased administrative efficiency, improved academic tracking, and enhanced communication, leading to better academic performance and resource management.
- Cost Savings: Reduction in manual administrative tasks, potentially lowering operational costs and reducing errors.

Funding:
- Funding options could include institutional budgets, grants, or external investments.
- Potential for cost-sharing among multiple institutions or monetizing through subscription models for other institutions.

2.1.3 Behavioral Feasibility

Behavioral feasibility examines how the proposed system will impact the users and whether it will be accepted by them.

User Acceptance:
- The system is designed to be user-friendly with intuitive interfaces for different user roles (Admin, Teacher, Student).
- Early user engagement and feedback during development to ensure the system meets user needs and preferences.

Training and Support:
- Minimal training required due to intuitive design and comprehensive documentation.
- Ongoing support and updates can be managed through a dedicated team or support system.

Change Management:
- Effective communication about the benefits and features of the system to ensure users are aware of its advantages.
- Gradual implementation and training sessions to help users adapt to the new system.

2.2 Drawback of Existing System

The existing systems in educational institutions often suffer from several drawbacks:

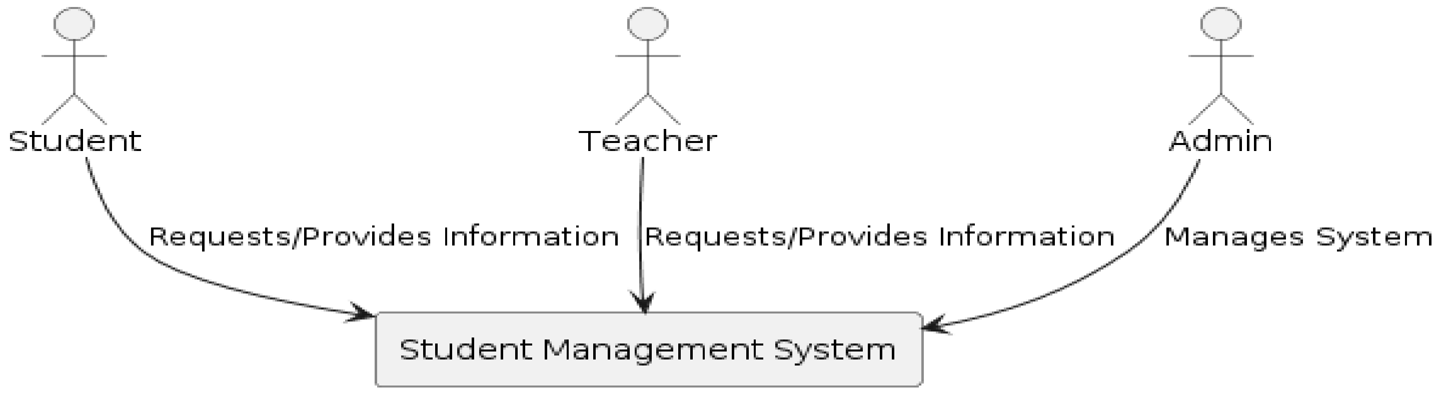
- Manual Processes: Reliance on manual processes for student and faculty management leads to inefficiencies and errors.
- Fragmented Data: Lack of centralized data storage results in fragmented information, making it difficult to access and manage data effectively.
- Poor Communication: Inadequate communication channels between students, teachers, and administrators hinder effective collaboration.
- Limited Tracking and Reporting: Existing systems may lack robust tools for tracking academic performance and generating insightful reports.

- Security Risks: Manual and outdated systems are more prone to data breaches and unauthorized access.

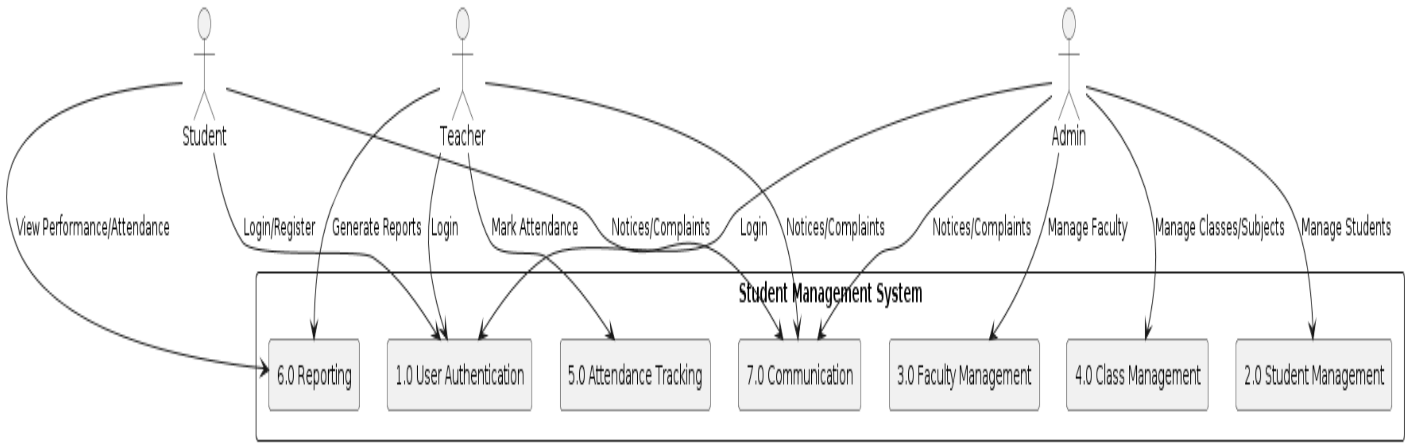
2.3 System Analysis

2.3.1 Data Flow Diagram (DFD)

Level 0 DFD (Context Diagram):
- Depicts the overall system and interactions with external entities (Admins, Teachers, Students).


Student
Teacher
Admin
Requests/Provides Information
Requests/Provides Information
Manages System
Student Management System

Level 1 DFD:
- Breaks down the major processes such as User Authentication, Student Management, Class Management, Attendance Tracking, and Communication (Notices and Complaints).



2.4 Proposed System

The proposed Student Management System addresses the drawbacks of the existing systems by providing a centralized, efficient, and user-friendly platform for managing educational institution operations.

Features:
- Centralized Management: Unified platform for managing students, faculty, classes, and subjects.
- Role-Based Access Control: Secure access tailored to user roles (Admin, Teacher, Student).
- Interactive Dashboards: User-specific dashboards displaying relevant information and functionalities.
- Attendance and Performance Tracking: Tools for marking attendance and visualizing academic performance.
- Communication Tools: Sections for posting notices and handling complaints to enhance communication.
- Scalability and Security: Built using the MERN stack, ensuring scalability and robust security measures.

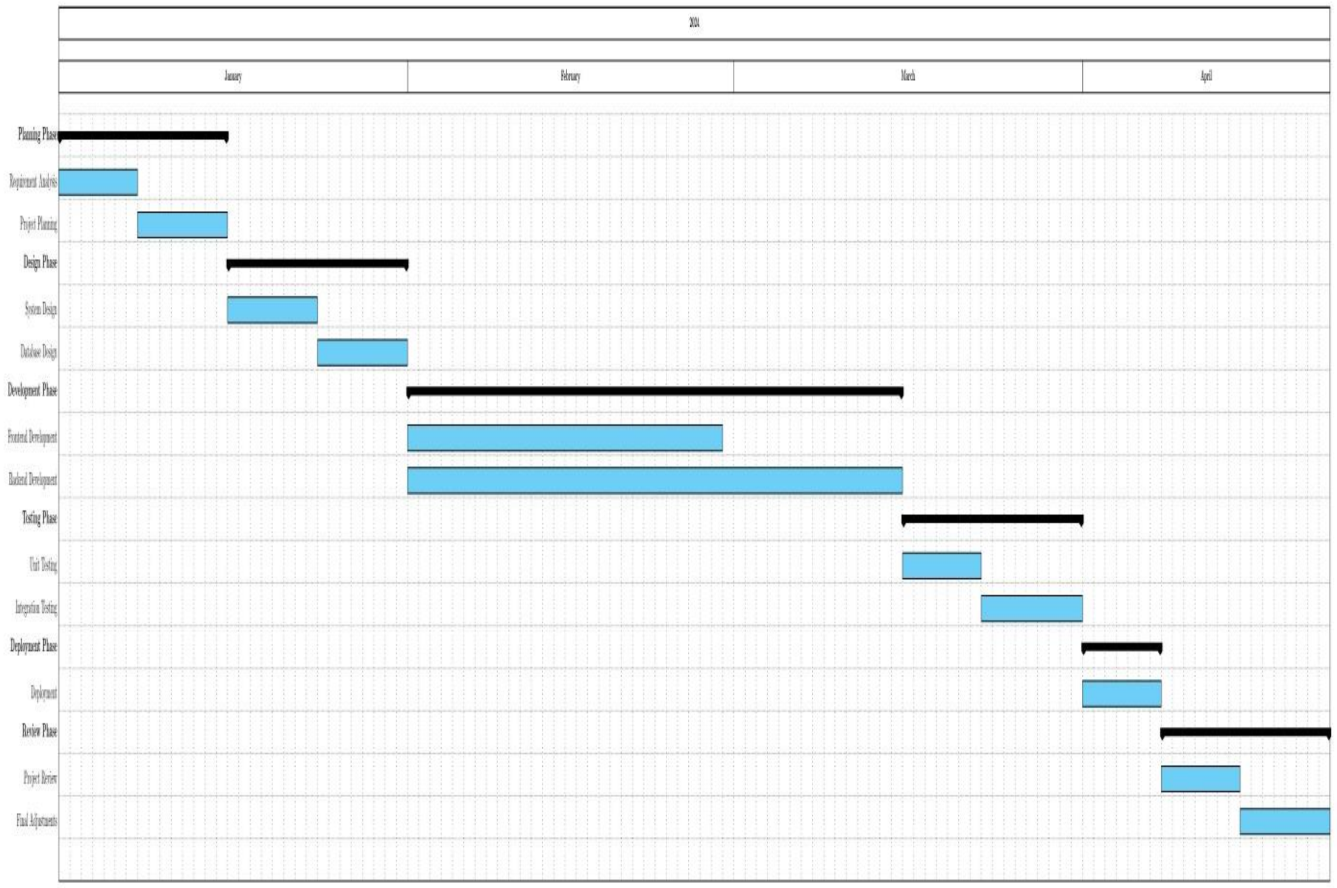
2.5 Project Plan

2.5.1 Gantt Chart

A Gantt chart is a visual project management tool that outlines the project schedule, showing the start and finish dates of the various elements of the project. Below is an example of the project plan for the Student Management System:

This Gantt chart illustrates the planned timeline for each phase of the project, ensuring that all tasks are completed on schedule. Adjustments can be made as needed based on project progress and feedback.

By following this structured project plan, the Student Management System can be developed and implemented efficiently, meeting the defined objectives and providing significant benefits to educational institutions.



i. **Planning Phase:** (January 1, 2024 - January 15, 2024)
   - Requirement Analysis (January 1, 2024 - January 7, 2024)
   - Project Planning (January 8, 2024 - January 15, 2024)

ii. **Design Phase:** (January 16, 2024 - January 31, 2024)
   - System Design (January 16, 2024 - January 23, 2024)
   - Database Design (January 24, 2024 - January 31, 2024)

iii. **Development Phase:** (February 1, 2024 - March 15, 2024)
   - Frontend Development (February 1, 2024 - February 28, 2024)
   - Backend Development (February 1, 2024 - March 15, 2024)

iv. **Testing Phase:** (March 16, 2024 - March 31, 2024)
   - Unit Testing (March 16, 2024 - March 22, 2024)
   - Integration Testing (March 23, 2024 - March 31, 2024)

v. **Deployment Phase:** (April 1, 2024 - April 7, 2024)
   - Deployment (April 1, 2024 - April 7, 2024)

vi. **Review Phase:** (April 8, 2024 - April 22, 2024)
   - Project Review (April 8, 2024 - April 14, 2024)
   - Final Adjustments (April 15, 2024 - April 22, 2024)

(Chapter 3)

DESIGN DESCRIPTION

The Student Management System (SMS) is designed to streamline the management of students, faculty, courses, and administrative tasks in educational institutions. The design of this system leverages the MERN stack (MongoDB, Express.js, React.js, Node.js) to provide a robust, scalable, and user-friendly web-based application. The system is divided into three primary user roles: Admin, Teacher, and Student, each with specific functionalities and permissions.

3.1 System Architecture

The architecture of the SMS is based on a three-tier architecture:

i. **Presentation Layer (Frontend):**

- Technologies Used: React.js, Material UI, Redux
- Responsibilities:
    - Provides an interactive user interface for the Admin, Teacher, and Student roles.
    - Manages state using Redux to ensure a smooth user experience.
    - Implements responsive design principles to ensure usability across various devices.
- Components:
    - Dashboard: Provides role-specific dashboards (Admin, Teacher, Student) with relevant information and actions.
    - Forms: For inputting data such as student details, attendance, grades, etc.
    - Tables and Lists: To display data such as student lists, course details, etc.
    - Graphs and Charts: For visualizing data such as attendance statistics and academic performance.

ii. **Application Layer (Backend):**

- Technologies Used: Node.js, Express.js
- Responsibilities:
    - Handles business logic and data processing.
    - Provides a RESTful API for communication between the frontend and the database.

    - Implements authentication and authorization to ensure secure access to resources.
- Endpoints:
    - /api/users: Handles user registration, login, and profile management.
    - /api/students: Manages student records, including CRUD operations.
    - /api/teachers: Manages teacher records, including CRUD operations.
    - /api/courses: Manages course information, including CRUD operations.
    - /api/attendance: Tracks and retrieves attendance data.
    - /api/grades: Records and retrieves student grades.

iii. Data Layer (Database):

- Technology Used: MongoDB
- Responsibilities:
    - Stores all persistent data for the application, including user information, student records, course details, attendance, and grades.
    - Ensures data integrity and provides efficient data retrieval.
- Collections:
    - Users: Stores user credentials and profile information for Admin, Teacher, and Student roles.
    - Students: Stores detailed information about students.
    - Teachers: Stores detailed information about teachers.
    - Courses: Stores information about courses and subjects.
    - Attendance: Stores attendance records for students.
    - Grades: Stores grades and academic performance data for students.

3.2 System Design Tool

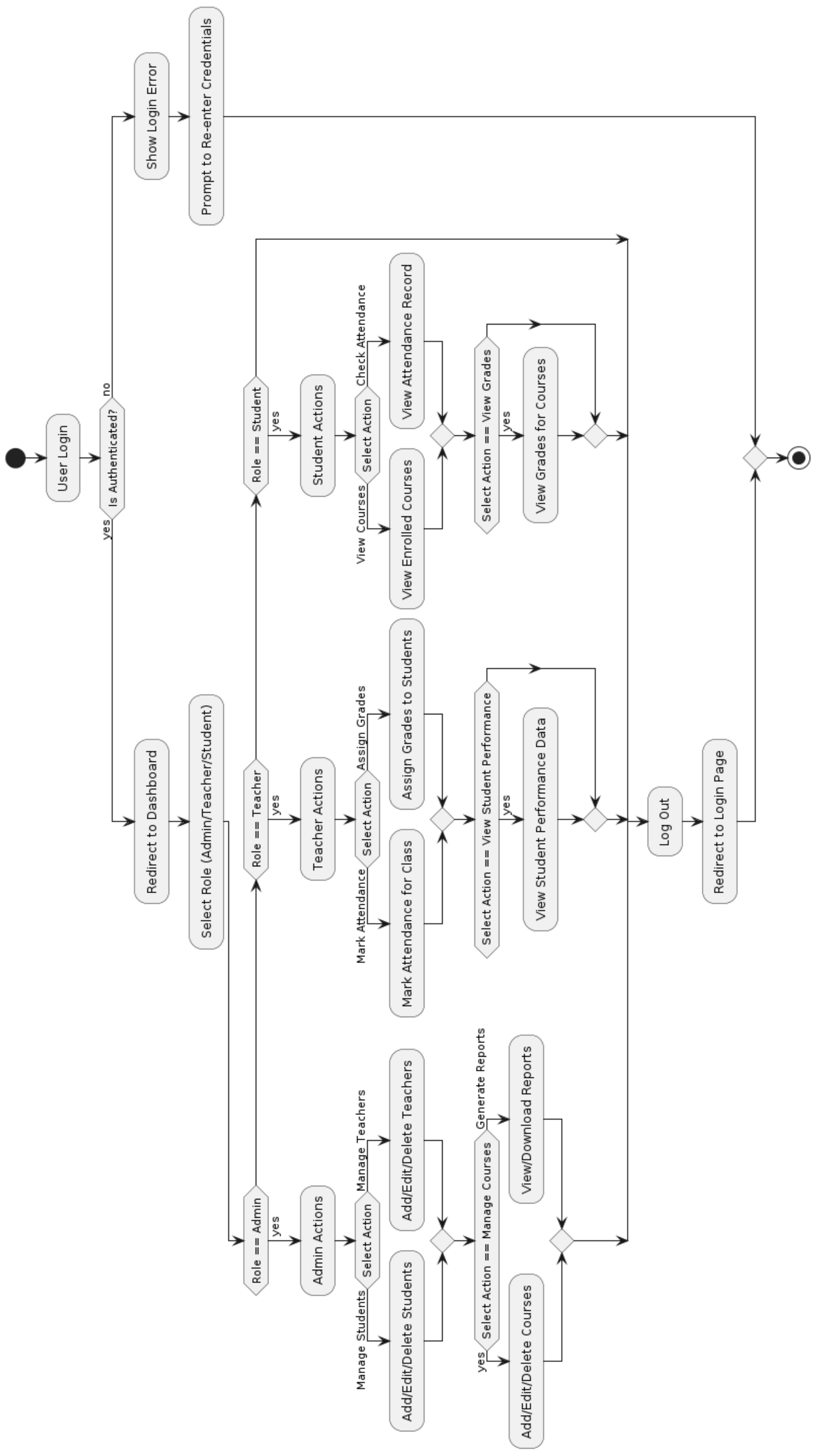
The design of the Student Management System (SMS) involves several key components, each focusing on different aspects of the system. This section outlines the various tools and diagrams used to design the system

3.2.1 System Flow Chart

A flowchart is a diagram that visually illustrates the steps, sequences, and decisions of a process or workflow.

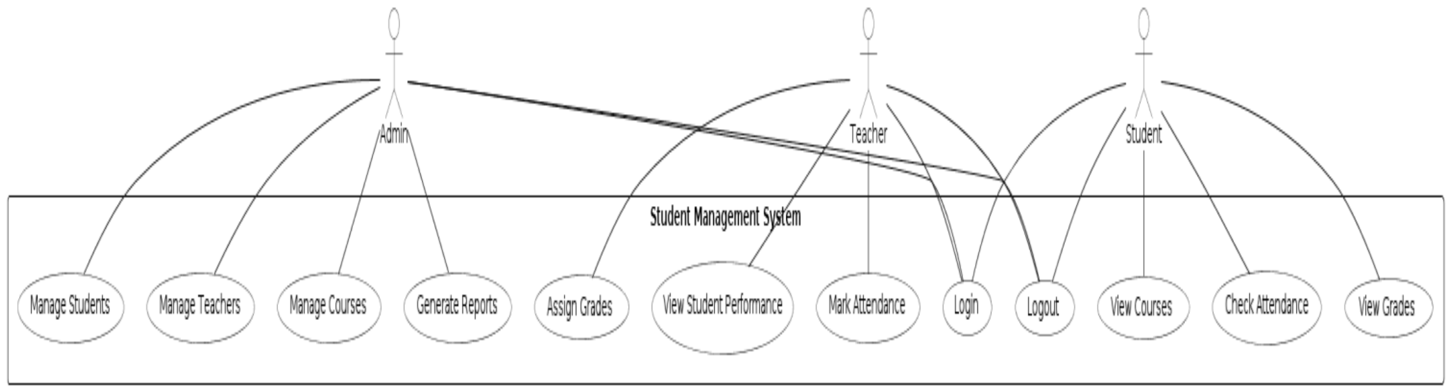
Example:

- **Processes:** Login, user authentication, data retrieval, attendance marking, grade input, etc.
- **Flows:** Data flows from the user interface to the server, and then to the database and back to the user interface.


User Login
Is Authenticated?
yes
no
Show Login Error
Prompt to Re-enter Credentials
Redirect to Dashboard
Select Role (Admin/Teacher/Student)
Role == Admin
yes
Admin Actions
Select Action
Manage Students
Manage Teachers
Add/Edit/Delete Students
Add/Edit/Delete Teachers
Select Action == Manage Courses
yes
Generate Reports
Add/Edit/Delete Courses
View/Download Reports
Role == Teacher
yes
Teacher Actions
Select Action
Mark Attendance
Assign Grades
Mark Attendance for Class
Assign Grades to Students
Select Action == View Student Performance
yes
View Student Performance Data
Role == Student
yes
Student Actions
Select Action
View Courses
Check Attendance
View Enrolled Courses
View Attendance Record
Select Action == View Grades
yes
View Grades for Courses
Log Out
Redirect to Login Page

3.2.2 Use Case Design

Use case design involves identifying the various users of the system and describing how they will interact with the system. Each use case represents a specific functionality that the system provides to its users.



Example Use Cases:

- **Admin:**
  - Manage Students
  - Manage Teachers
  - Manage Courses
  - Generate Reports

- **Teacher:**
  - Mark Attendance
  - Assign Grades
  - View Student Performance

- **Student:**
  - View Courses
  - Check Attendance
  - View Grades

3.2.3 Data Verification

Data verification ensures that the data entered into the system is accurate and consistent. This step involves implementing validation rules and checks within the system to prevent errors and inconsistencies.

Example Data Verification Methods:

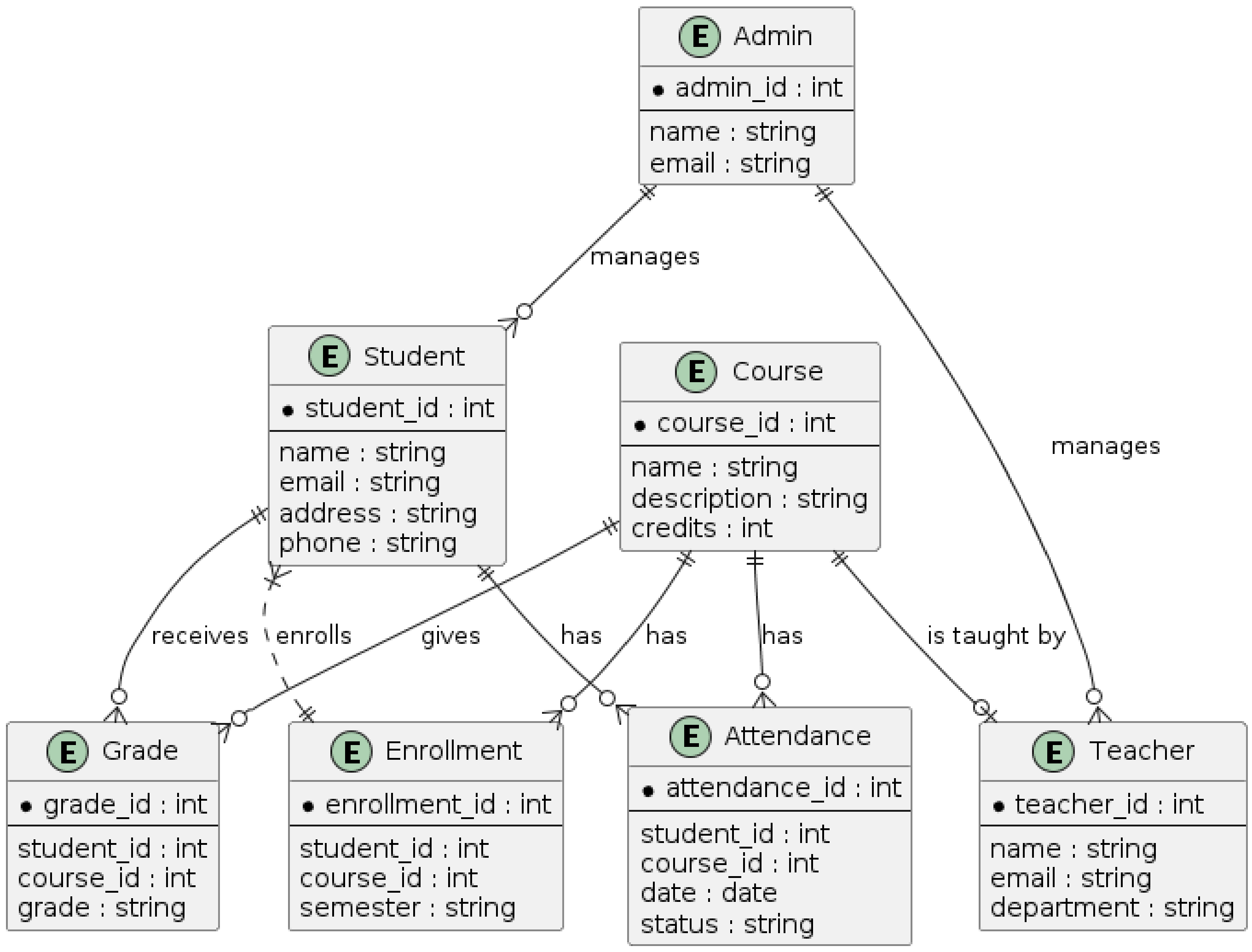
- **Input Validation:** Ensure all required fields are filled and data is in the correct format (e.g., email addresses, dates).
- **Data Integrity Checks:** Implement constraints in the database to maintain referential integrity (e.g., foreign key constraints).
- **Consistency Checks:** Validate that data follows business rules (e.g., a student's age should be within a certain range).

3.2.4 ER Diagram

An Entity-Relationship (ER) Diagram represents the data model of the system, showing how different entities (e.g., students, teachers, courses) are related to each other.

- **Entities**:
  - **Student**: Represents students with attributes like student_id, name, email, address, and phone.
  - **Teacher**: Represents teachers with attributes like teacher_id, name, email, and department.
  - **Course**: Represents courses with attributes like course_id, name, description, and credits.
  - **Enrollment**: Represents the enrollment of students in courses with attributes like enrollment_id, student_id, course_id, and semester.
  - **Attendance**: Represents attendance records with attributes like attendance_id, student_id, course_id, date, and status.
  - **Grade**: Represents grades awarded to students with attributes like grade_id, student_id, course_id, and grade.
  - **Admin**: Represents admins with attributes like admin_id, name, and email.

- **Relationships**:
  - A **student** enrolls in multiple **courses** through the **Enrollment** entity.
  - A **student** has multiple **attendance** records for each **course**.
  - A **student** receives multiple **grades** for different **courses**.
  - A **course** is taught by a **teacher**.
  - An **admin** manages multiple **teachers** and **students**.



3.2.5 Database Design

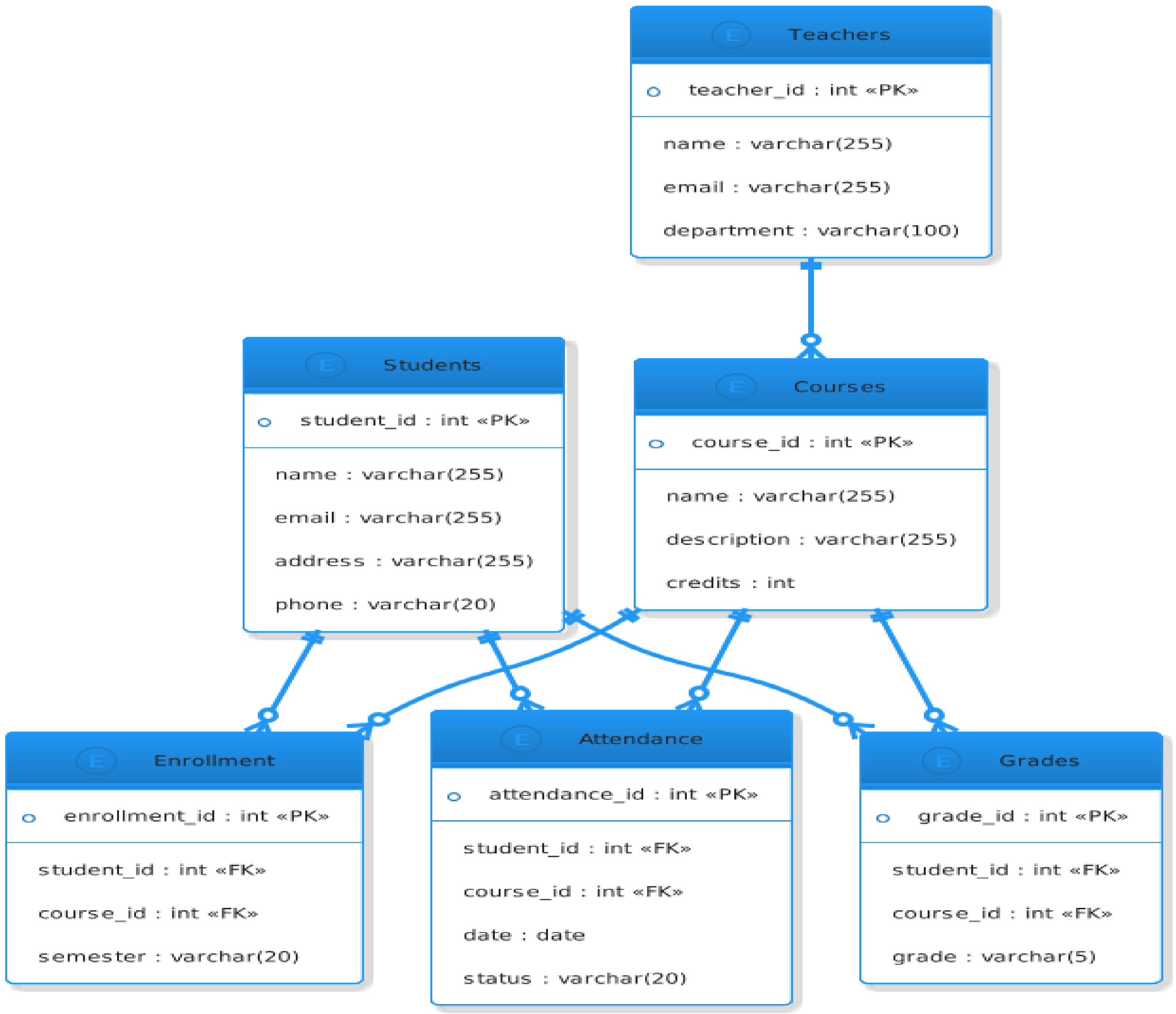
Database design involves creating a detailed schema that defines the structure of the database, including tables, columns, and relationships between tables

Explanation

- **Entities**:
  - **Students**: Represents students with attributes like student_id, name, email, address, and phone.
  - **Teachers**: Represents teachers with attributes like teacher_id, name, email, and department.

- **Courses**: Represents courses with attributes like course_id, name, description, and credits.
- **Enrollment**: Represents the enrollment of students in courses with attributes like enrollment_id, student_id, course_id, and semester.
- **Attendance**: Represents attendance records with attributes like attendance_id, student_id, course_id, date, and status.
- **Grades**: Represents grades awarded to students with attributes like grade_id, student_id, course_id, and grade.

- **Relationships**:
  - A **student** enrolls in multiple **courses** through the **Enrollment** entity.
  - A **course** has multiple **students** enrolled through the **Enrollment** entity.
  - A **student** has multiple **attendance** records for each **course**.
  - A **course** has multiple **attendance** records.
  - A **student** receives multiple **grades** for different **courses**.
  - A **course** has multiple **grades**.



3.3 Module Design

Module design is a design principle that breaks down a complex system into smaller parts, called modules, that can be independently created, modified, or replaced.

i. **Authentication Module**:
   - Responsible for user authentication and session management.
   - Handles login, logout, and session expiration.

ii. **User Management Module**:
   - Manages user accounts, including students, teachers, and administrators.
   - Provides functionalities for adding, editing, and deleting user accounts.
   - Handles user roles and permissions.

iii. **Student Management Module**:
   - Handles student-related functionalities, such as enrollment, attendance, and grades.
   - Manages student information, including personal details and academic records.

iv. **Teacher Management Module**:
   - Handles teacher-related functionalities, such as course assignment and grading.
   - Manages teacher information, including personal details and course assignments.

v. **Course Management Module**:
   - Manages course-related functionalities, such as course creation, scheduling, and curriculum management.
   - Handles course enrollment and registration.

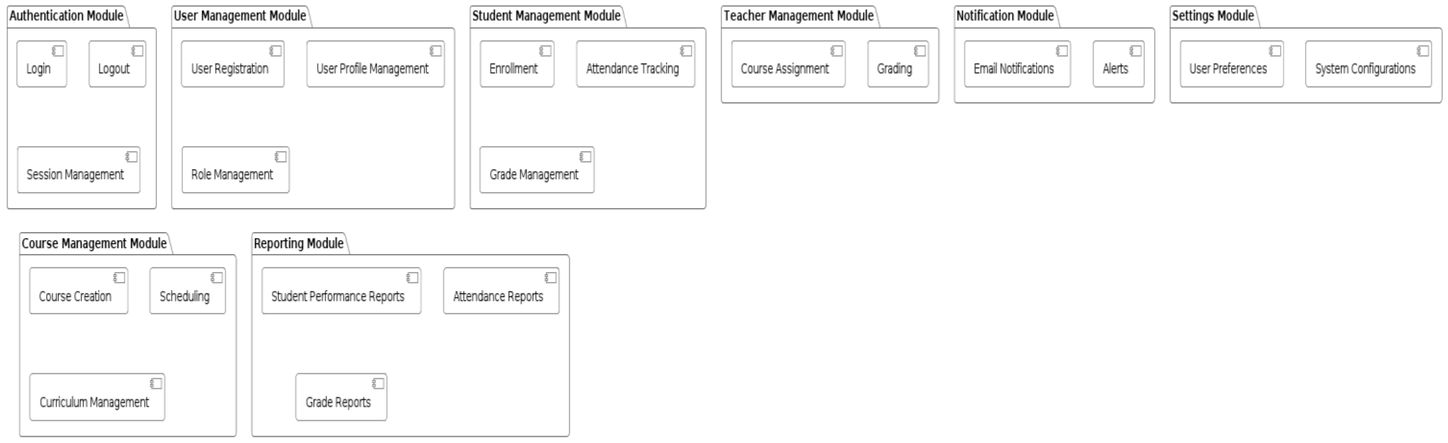
vi. **Attendance Tracking Module**:
   - Manages attendance tracking for classes and lectures.
   - Records student attendance and generates attendance reports.

vii. **Grade Management Module**:
   - Manages grading functionalities, including grade assignment, calculation, and recording.
   - Generates grade reports and transcripts.

viii. **Reporting Module**:
   - Provides reporting functionalities for administrators, teachers, and students.
   - Generates various reports such as student performance reports, attendance reports, and grade reports.

ix. **Notification Module**:
   - Sends notifications and alerts to users regarding important events or deadlines.
   - Notifies users about upcoming exams, assignment deadlines, or system updates.

x. **Settings Module**:
   - Allows users to customize their preferences and settings.
   - Manages system configurations and options.



3.4 User Interface Design

User Interface (UI) Design involves creating intuitive and visually appealing interfaces that enable users to interact effectively with the system. Here's a breakdown of UI design components for a Student Management System:

i. **Login Page:**
   - Username and password fields for authentication.
   - "Login" button to initiate the login process.
   - "Forgot Password" option for password recovery.

ii. **Dashboard:**
   - Overview of user-specific information (e.g., upcoming classes, pending tasks).
   - Navigation menu for accessing different modules and functionalities.
   - Quick links to commonly used features.

iii. **User Profile:**
   - Editable fields for updating personal information (e.g., name, email, contact).
   - Option to change password or update profile picture.

iv. **Student Management Interface:**
   - Interface for viewing and managing student records.
   - Search and filter options for finding specific students.
   - Buttons for adding, editing, or deleting student records.

v. **Teacher Management Interface:**
   - Interface for viewing and managing teacher records.
   - Similar functionalities as the student management interface.

vi. **Course Management Interface:**
   - Interface for managing courses, including creation, editing, and scheduling.
   - Options for assigning teachers to courses and managing course curriculum.

vii. **Attendance Interface:**
   - Interface for marking attendance and viewing attendance records.

- Dropdowns or calendars for selecting dates and courses.
- List of students with checkboxes for marking attendance.

viii. **Grades Interface:**
- Interface for entering and managing grades for students.
- Dropdowns for selecting courses and students.
- Table layout for displaying grades with options for editing.

ix. **Reports Interface:**
- Interface for generating and viewing various reports (e.g., student performance, attendance, grades).
- Options for selecting report parameters and exporting reports.

x. **Notification Interface:**
- Display notifications and alerts for important system events.
- Option to view or dismiss notifications.

xi. **Settings Interface:**
- Interface for configuring user preferences and system settings.
- Options for changing language, theme, notification preferences, etc.

3.5 Report Design

In a Student Management System, the generation of comprehensive and informative reports is paramount for monitoring student progress, analyzing performance trends, and facilitating informed decision-making by administrators, faculty, and students. Effective report design ensures that data is presented in a clear, concise, and visually appealing manner, enhancing the usability and utility of the system.

i. Title Page
ii. Bonafide Certificate
iii. Certificate from the Company
iv. Abstract
v. Acknowledgement
vi. Table of Contents
vii. List of Tables
viii. List of Figures
ix. List of Symbols, Abbreviations or Nomenclature (Optional)
x. Chapters
xi. Appendices
xii. References

3.5.1 Format of Report

i. For the format of the report, the following guidelines are adhered to:

ii. All fonts are set to Times New Roman.

iii. Font size for the chapter is set to 12.

iv. Figure captions and table names are formatted in Title Case with a font size of 10 Middle.

v. Chapter headings are formatted in Uppercase with a font size of 24 Middle.

vi. Line spacing is set to 1.5.

vii. Space between paragraphs is added using the "Add space After Paragraph" option.

viii. Page size is adjusted to have left margin of 1.5”, right margin of 1.0”, top margin of 1.0”, bottom margin of 0.5”, and gutter of 0” with gutter position set to Left.

ix. Main headings are formatted in Upper case with a font size of 12 and bold.

x. Subheadings are formatted in Title Case with a font size of 12 and bold. Subheadings are numbered according to the hierarchical structure provided.

3.5.2 Frequency of Report

The frequency of generating reports from the Student Management System (SMS) is determined based on the needs and requirements of stakeholders. Reports may be generated periodically or on an ad-hoc basis to provide timely and relevant information for decision-making purposes.

i. **Administrative Needs**: Administrative reports, such as enrollment statistics, financial summaries, and resource allocation, may be generated on a monthly or quarterly basis to track performance indicators and monitor budgetary allocations.
ii. **Academic Requirements**: Academic reports, including student progress reports, grade summaries, and course evaluations, are typically generated at the end of each semester or academic year to assess student performance and academic outcomes.
iii. **Operational Efficiency**: Reports that support day-to-day operations, such as attendance records, timetable management, and inventory tracking, may be generated on a weekly or daily basis to ensure smooth functioning of the institution.
iv. **Ad Hoc Requests**: In addition to scheduled reports, ad hoc reports may be generated in response to specific queries or requests from stakeholders. These reports provide timely insights into emerging issues or areas of interest and support data-driven decision-making.

(Chapter 4)

IMPLEMENTATION AND TESTING

The implementation and testing phase is a critical stage in the development lifecycle of the Student Management System (SMS). This phase involves translating the design specifications into a functional system and rigorously testing its performance to ensure reliability, accuracy, and user satisfaction.

4.1 Implementation Constraints

During the implementation phase, various constraints may impact the development process and influence the final outcome of the SMS. These constraints include:

- **Technical Constraints**: Limitations in hardware, software, or network infrastructure may affect the implementation of certain features or functionalities.
- **Resource Constraints**: Constraints related to budget, time, and human resources may impact the scope and scale of the implementation effort.
- **Regulatory Constraints**: Compliance with regulatory standards and legal requirements may impose constraints on data security, privacy, and accessibility.
- **Integration Constraints**: Integration with existing systems or third-party services may pose challenges in terms of compatibility, interoperability, and data migration.

4.2 Testing

The testing phase of the SMS project involves comprehensive evaluation of the system's functionality, performance, and usability. Key aspects of testing include:

4.2.1 Testing Methodology

A structured testing methodology is employed to systematically identify and address defects, errors, and anomalies in the SMS. Testing methodologies include:

- **Unit Testing**: Individual components and modules of the SMS are tested in isolation to verify their correctness and functionality.

- **Integration Testing**: Integration of multiple components and modules is tested to ensure proper interaction and interoperability.
- **System Testing**: The integrated system is tested as a whole to validate its compliance with functional and non-functional requirements.
- **User Acceptance Testing (UAT)**: End users participate in UAT to evaluate the system's usability, accessibility, and alignment with their needs and expectations.

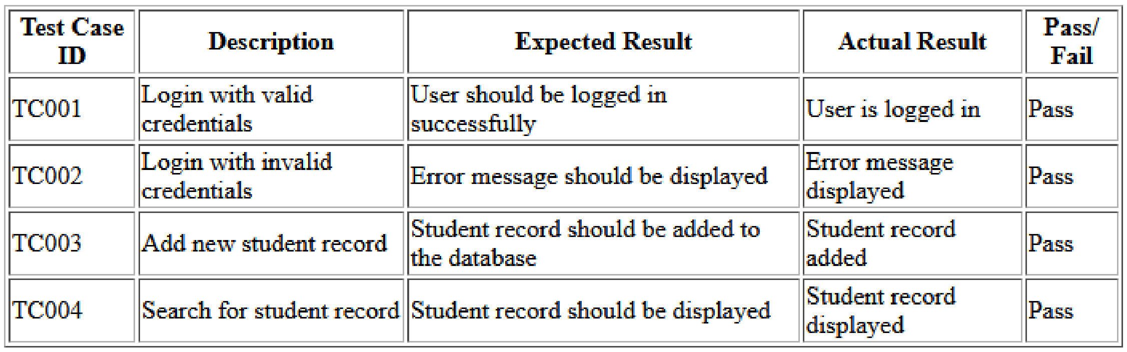
4.2.2 Test Cases

Test cases are systematically designed and executed to validate the behavior and performance of the SMS. Test cases include:

- **Functional Test Cases**: Test cases designed to verify the fulfillment of functional requirements specified for the SMS.
- **Performance Test Cases**: Test cases designed to evaluate the system's responsiveness, scalability, and reliability under varying load conditions.
- **Security Test Cases**: Test cases designed to assess the system's resistance to security threats, vulnerabilities, and attacks.
- **Usability Test Cases**: Test cases designed to evaluate the system's ease of use, intuitiveness, and user experience.

Conclusion:

The implementation and testing phase of the SMS project are essential for ensuring the successful deployment and operation of the system. By addressing implementation constraints and conducting thorough testing, the SMS is prepared to deliver reliable, efficient, and user-friendly functionality to its stakeholders.



Test Case ID	Description	Expected Result	Actual Result	Pass/ Fail
TC001	Login with valid credentials	User should be logged in successfully	User is logged in	Pass
TC002	Login with invalid credentials	Error message should be displayed	Error message displayed	Pass
TC003	Add new student record	Student record should be added to the database	Student record added	Pass
TC004	Search for student record	Student record should be displayed	Student record displayed	Pass

(Chapter 5)

CONCLUSION & FUTURE ENHANCEMENT

In conclusion, the development of the Student Management System (SMS) has been a significant endeavor aimed at enhancing the efficiency, transparency, and effectiveness of educational institution operations. Through the implementation of robust features and functionalities, the SMS has successfully addressed key challenges in student management, faculty coordination, and administrative oversight.

Key highlights of the SMS project include:

- **Streamlined Processes:** Automation of routine tasks such as enrollment, attendance tracking, and grade management has significantly reduced administrative burden and improved operational efficiency.

- **Enhanced Communication:** The SMS provides a centralized platform for seamless communication and collaboration among students, faculty, and administrators, fostering a conducive learning environment.

- **Data-driven Insights:** Advanced reporting and analytics capabilities enable stakeholders to gain valuable insights into student performance, academic trends, and institutional metrics, facilitating informed decision-making.

While the current iteration of the SMS offers comprehensive functionality and usability, there is always room for improvement and innovation. Future enhancements to the SMS may include:

- **Mobile Accessibility:** Developing mobile applications for iOS and Android platforms to enable anytime, anywhere access to SMS features for students, faculty, and administrators.

- **Personalization Features:** Implementing personalized dashboards, notifications, and recommendations tailored to individual user preferences and learning objectives.

- **Integration with Learning Management Systems (LMS):** Integrating the SMS with popular LMS platforms to streamline course management, content delivery, and assessment processes.

- **Enhanced Data Security:** Strengthening data security measures through encryption, access controls, and regular security audits to safeguard sensitive information and ensure compliance with data protection regulations.

Appendices

In the appendices section of the project report, additional supplementary information relevant to the project is provided. This section includes links to the relevant documentation, resources, and project website.

A. Project Repository

The source code, technical documentation, user manuals, code samples, test data, and glossary for the Student Management System (SMS) project are available in the following GitHub repository:

Student Management System GitHub Repository

Feel free to explore the repository for detailed technical documentation, user manuals, code samples, test data, and glossary related to the project.

B. Project Website

The official website for the Student Management System (SMS) project provides additional information, updates, and resources. You can visit the project website at the following URL:

Student Management System Project Website

Explore the website for more information about the SMS project, its features, and how to get involved.

References

MERN stack and web development resources, books, and tutorials used in this project

- **MongoDB**
  - MongoDB Documentation: https://docs.mongodb.com/
  - MongoDB University: https://university.mongodb.com/
- **Express.js**
  - Express.js Documentation: https://expressjs.com/
  - Learn Express: https://learnexpressjs.com/
- **React.js**
  - React.js Documentation: https://reactjs.org/docs/getting-started.html
  - React Tutorial for Beginners: https://www.youtube.com/watch?v=Ke90Tje7VS0
- **Node.js**
  - Node.js Documentation: https://nodejs.org/en/docs/
  - Node.js Tutorial for Beginners: https://www.youtube.com/watch?v=U8XF6AFGqlc
- **MERN Stack Tutorials**
  - MERN Stack Tutorial for Beginners: https://www.youtube.com/watch?v=7CqJlxBYj-M
  - Build a MERN Stack App: https://www.youtube.com/watch?v=ngc9gnGgUdA
- **Additional Resources**
  - Full Stack Open: https://fullstackopen.com/en/
  - The Net Ninja - React, Node & MongoDB Full Stack: https://www.thenetninja.co.uk/courses/mern-stack-tutorial