Software Requirement Specification

for

Engineering College Admission Management System (ECAMS)

Version 1.0.0

Prepared by

Group No.: 14

Omkar Kasture	**22210269**	**omkar.22210269@viit.ac.in**
Krishna Magar	**22210186**	**krishna.22210186@viit.ac.in**
Ayush Mahandule	**22211418**	**ayush.22211418@viit.ac.in**
Anushka Bachhav	**22320176**	**anushka.22320176@viit.ac.in**

Instructor	**Dr. Sachin S. Sakhare Sir**
Course	**Software Engineering and Project Management**
Date	**02/12/24**

1 Introduction

The Engineering College Admission Management System is a web-based application designed to streamline and automate the admission process for engineering colleges. This system aims to simplify the tasks of managing student applications, admissions, and enrollment, providing an efficient and transparent solution for administrators, applicants, and faculty members. By digitizing the admission process, the system minimizes manual effort, reduces errors, and ensures better data management.

Overview of This Section:

In this section, readers will find an in-depth look at the system's features, including applicant registration, document submission, merit list generation, and communication of admission status. It will also outline the key modules of the project, such as the admin dashboard, student portal, and reporting tools. Additionally, this section will discuss the system's technical architecture, implementation details, and the benefits it offers to colleges and applicants alike.

1.1 Document Purpose

This Software Requirements Specification (SRS) document defines the requirements for the Engineering College Admission Management System (ECAMS), Version 1.0. The primary purpose of this document is to provide a comprehensive overview of the functionalities, constraints, and specifications of the system, serving as a reference for stakeholders, including developers, administrators, and end-users. It outlines the software's objectives, scope, and detailed requirements, ensuring clarity and alignment among all project participants.

The scope of this document focuses on the entire admission management system, encompassing core modules such as applicant registration, merit list generation, admission tracking, and fee payment integration. It also includes subsystems for managing student records, generating admission reports, and facilitating communication between applicants and the college administration. This SRS ensures that the design, development, and implementation of ECAMS meet the specific needs of engineering colleges while maintaining scalability and adaptability for future enhancements.

1.2 Product Scope

The Engineering College Admission Management System (ECAMS) is a comprehensive web-based solution designed to streamline the admission process for engineering colleges. This system automates key functions such as application submission, document verification, merit list generation, fee payment, and enrollment. By digitizing these workflows, ECAMS eliminates the inefficiencies of manual processes, reducing errors, improving transparency, and saving time for both applicants and administrative staff.

The primary goal of ECAMS is to create a seamless and user-friendly platform that benefits all stakeholders. For students, it simplifies the application process with clear instructions, status tracking, and online payment options. For administrators, it provides robust tools for managing applications, generating reports, and ensuring compliance with admission guidelines. With its scalable design, the system can accommodate growing applicant numbers, making it suitable for both small and large institutions. Ultimately, ECAMS enhances the overall admission experience, fostering trust and efficiency in the college enrollment process.

1.3 Intended Audience and Document Overview

This document is intended for the following readers:

1. **Clients (College Administration):** To provide an understanding of the system's capabilities and ensure alignment with their requirements for the admission process.
2. **Professors:** To review and assess the system's educational context and functionality, ensuring its alignment with institutional objectives.
3. **Developers:** To serve as a detailed guide for implementing the specified requirements, ensuring all features and functionalities are built as described.
4. **Testers:** To identify and verify test cases based on the defined requirements, ensuring the system operates as intended.
5. **Project Managers:** To track the project's progress, ensuring timelines and objectives are met.
6. **Documentation Writers:** To ensure accurate user manuals, help guides, and other supporting documents are created based on the specifications in this document.

The rest of this Software Requirements Specification (SRS) document is organized as follows:

1. **Introduction:** Provides an overview of the system, its purpose, and the scope of this document.
2. **Overall Description:** Outlines the system's general functionality, user needs, and high-level constraints.
3. **Specific Requirements:** Details the functional and non-functional requirements, user interactions, and system interfaces.
4. **System Features:** Describes individual features and their specifications.
5. **Other Non-Functional Requirements:** Covers performance, security, and usability considerations.
6. **Appendices:** Includes additional references, terminology, and supporting material.

For a logical reading sequence, clients and professors should begin with the **Introduction** and **Overall Description** sections to understand the system's purpose and scope. Developers and testers should focus on the **Specific Requirements** and **System Features** sections to gather technical details. Documentation writers should review the entire document to create comprehensive user guides.

1.4 Definitions, Acronyms and Abbreviations

Below is a list of abbreviations and acronyms used in this document, sorted alphabetically:

1. **API** - Application Programming Interface
2. **CRM** - Customer Relationship Management
3. **DBMS** - Database Management System
4. **ECAMS** - Engineering College Admission Management System
5. **HTTP** - HyperText Transfer Protocol
6. **JSON** - JavaScript Object Notation
7. **SRS** - Software Requirements Specification
8. **UI** - User Interface
9. **URL** - Uniform Resource Locator
10. **XML** - Extensible Markup Language

1.5 Document Conventions

This Software Requirements Specification (SRS) document adheres to the following standards and conventions to ensure consistency and readability:

1. Formatting Conventions:

- **Font:** Times New Roman, size 11 or 12, is used for all text.
- **Spacing:** All document text is single-spaced with 1-inch margins on all sides.
- **Titles:** Section titles are formatted in bold and numbered (e.g., **1. Introduction**), while subsection titles are numbered hierarchically and use bold text (e.g., **1.1 Document Purpose**).
- **Emphasis:** Italics are used to indicate comments or placeholders, while bold text is used for headings and important keywords.
- **Lists:** Bulleted or numbered lists are used for clarity and organization where applicable.

2. Naming Conventions:

- **Acronyms:** Acronyms are defined in the **Definitions, Acronyms, and Abbreviations** section and are capitalized throughout the document.
- **Variables/Identifiers:** Any variables or identifiers referenced in the document follow a camelCase or snake_case naming convention to align with programming best practices.
- **File Names:** File names mentioned in the document use a lowercase naming convention with underscores for separation (e.g., admission_form_template).

3. Referencing Conventions:

2 **Section References:** References to other sections or subsections use the full numbering format (e.g., "See Section **2.2** for details").
3 **Figures and Tables:** Figures and tables are numbered sequentially and referenced by their numbers (e.g., "Figure 1: System Architecture").

1.6 References and Acknowledgments

References:

1. *IEEE Standard 830-1998: Recommended Practice for Software Requirements Specifications.*
2. *User Interface Style Guide - Version 2.0 (Applicable to the Engineering College Admission System).*
3. *Admission Process Guidelines Document - Provided by the client (Engineering College Administration).*
4. *Web Development Standards: https://www.w3.org/*
5. *Database Design Principles and Practices: https://dev.mysql.com/doc/*
6. *Node.js Documentation: https://nodejs.org/docs/latest/api/*

Acknowledgments:

We would like to acknowledge the following individuals and teams for their contributions:

- ***Engineering College Administration Team:*** *For providing detailed admission process insights and requirements.*
- ***Academic Supervisor/Professor:*** *For reviewing and guiding the project to meet educational standards.*
- ***Development Team Members:*** *For contributing to the design and prototyping of the system.*
- ***External Resources:*** *Authors of the referenced standards, guidelines, and documentation that informed the creation of this SRS.*

2 Overall Description

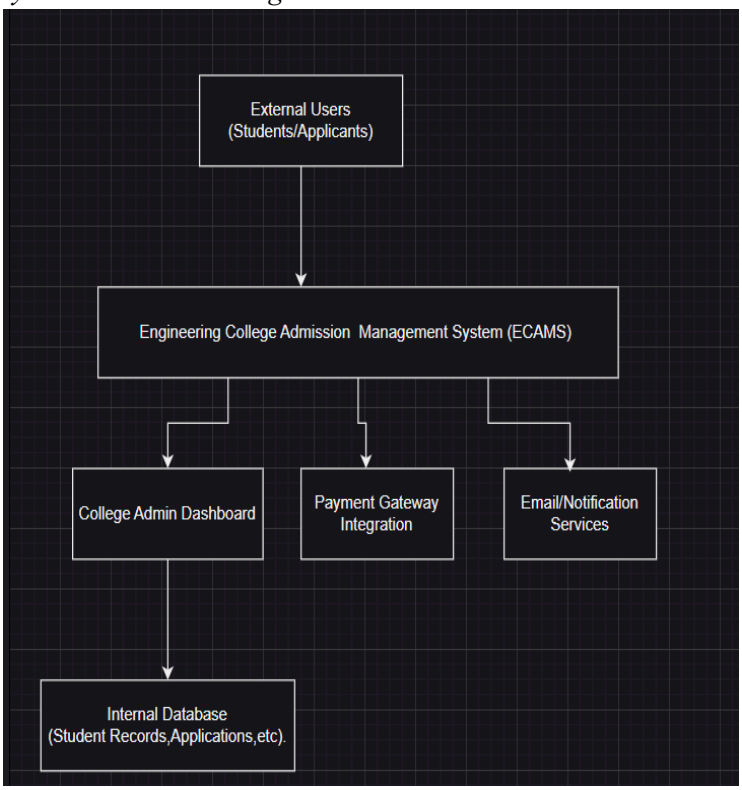
2.1 Product Overview

The Engineering College Admission Management System (ECAMS) is a standalone, web-based product designed to replace traditional, paper-based admission processes in engineering colleges. The system provides an integrated platform for students, administrators, and faculty members to handle the end-to-end admission workflow efficiently. This product is entirely self-contained and does not rely on legacy systems, although it can integrate with existing database management systems or ERP solutions for broader institutional use.

The system offers core functionalities such as online application submission, document upload and verification, merit list generation, payment processing, and student enrollment. It is designed to enhance accuracy, transparency, and user experience while ensuring compliance with admission guidelines and scalability for future growth.

Product Context:
ECAMS interacts with multiple external entities and systems. For example, it communicates with payment gateways for fee transactions, email services for notifications, and the college's internal database to store and retrieve student records. Additionally, it integrates with external regulatory systems to ensure adherence to government or institutional policies.

System Interaction Diagram:



2.2 Product Functionality

The Engineering College Admission Management System (ECAMS) is designed to perform several major functions to streamline and automate the admission process. Below is a high-level summary of the key functionalities:

- **User Registration and Authentication:**
- Allows students to create accounts, log in securely, and recover passwords if needed.
- **Online Application Submission:**
- Enables students to fill out admission forms, upload required documents, and submit applications digitally.
- **Document Verification:**
- Allows administrators to review, verify, and approve or reject uploaded documents.
- **Merit List Generation:**
- Automates the generation of merit lists based on predefined criteria, such as exam scores and reservation policies.
- **Fee Payment and Receipt Generation:**
- Integrates with payment gateways to facilitate secure online fee payments and provide receipts to students.
- **Admission Status Tracking:**
- Allows students to view the current status of their application and admission process in real-time.
- **Administrative Dashboard:**
- Provides college administrators with tools to manage applications, generate reports, and monitor system performance.
- **Notification System:**
- Sends email and SMS notifications to students about application updates, deadlines, and admission offers.
- **Data Management:**
- Maintains a centralized database to store student information, application records, and admission details securely.
- **Audit and Reporting:**
- Generates reports for auditing purposes and provides insights into the admission process for decision-making.

2.3 Design and Implementation Constraints

The design and implementation of the Engineering College Admission Management System (ECAMS) must adhere to the following constraints to ensure functionality, compatibility, and maintainability:

1. Methodology and Modeling Language

- **COMET Method:**

    - The development must follow the Component-Based Object Modeling and Execution Time (COMET) method for software design, which emphasizes object-oriented analysis and design for real-time and distributed systems. Reference: COMET Method Overview.
- **UML Modeling Language:**
    - Unified Modeling Language (UML) must be used for all design artifacts, including use case diagrams, class diagrams, and sequence diagrams, to ensure standardized and clear communication of system designs. Reference: UML Documentation.

2. Hardware and Software Limitations

- **Hardware Requirements:**
    - The system must be deployable on standard college-provided servers with a minimum of 8 GB RAM, 4-core processors, and 500 GB of storage.
    - Must support client devices with basic hardware specifications (e.g., smartphones, tablets, or PCs).
- **Performance Requirements:**
    - The system must handle up to 10,000 concurrent users during peak admission periods without significant degradation in performance.

3. Interfaces and Integration Constraints

- **Payment Gateway Integration:**
    - Must interface with at least two secure payment gateways (e.g., Razorpay and PayPal) for fee collection.
- **Regulatory System Compliance:**
    - Must ensure compatibility with external regulatory systems (e.g., AICTE/UGC guidelines).

4. Technology and Tools

- **Programming Languages:**
    - The backend must be implemented in Node.js, and the frontend in React.js.
- **Database Technology:**
    - MySQL is required for database management, ensuring relational data integrity and scalability.

5. Security Considerations

- **Data Protection:**
    - Must comply with GDPR and other relevant data protection laws to secure sensitive student and institutional information.
- **Authentication:**
    - Must use secure login methods with multi-factor authentication for administrators and strong password policies for users.

6. Development Standards

- **Coding Standards:**
  - Follow ESLint rules for JavaScript coding to maintain consistency and readability.
- **Documentation Standards:**
  - Use JSDoc for documenting the source code and provide developer-friendly guides.

7. Deployment Constraints

- The system must be deployable on a Linux-based web server.
- It must support a web-based interface accessible through all modern browsers (e.g., Chrome, Firefox, Edge).

8. Parallel Operations

- During the initial deployment, the system must coexist with legacy admission systems for at least one admission cycle to ensure a smooth transition.

2.4 Assumptions and Dependencies

The Engineering College Admission Management System (ECAMS) is based on several assumptions and dependencies, which could influence its requirements and design. These factors must be acknowledged to mitigate risks arising from changes or inaccuracies in these assumptions.

Assumptions

1. **User Internet Access:**
   - All users (students, administrators, and faculty) will have reliable internet access to interact with the web-based system.
2. **Device Compatibility:**
   - Students and administrators will access the system using devices compatible with modern web browsers (e.g., Chrome, Firefox, Safari).
3. **Scalability Needs:**
   - The system will initially handle a moderate number of applications (up to 10,000 concurrent users), with scalability required only in subsequent versions.
4. **Payment Gateway Availability:**
   - Payment gateway services (e.g., Razorpay or PayPal) will remain operational and meet compliance requirements for processing application fees.
5. **Institutional Policy Stability:**
   - Admission policies and guidelines will not undergo significant changes during the current development and deployment phases.
6. **Support from Regulatory Authorities:**
   - External regulatory bodies (e.g., AICTE, UGC) will provide required APIs and data-sharing mechanisms for compliance.

Dependencies

1. **Third-Party Services:**
    - The system depends on third-party APIs for payment processing, email notifications, and SMS alerts. Any disruption or changes in these services may impact system functionality.
2. **Database Technology:**
    - The project depends on MySQL for managing application and user data. Compatibility issues or limitations in database performance could affect system operations.
3. **Hosting Environment:**
    - The system requires a secure and reliable hosting platform (e.g., AWS, Azure, or institutional servers) for deployment.
4. **Development Tools:**
    - Dependencies on specific tools and frameworks (e.g., Node.js, React.js) necessitate that their support and updates remain consistent during the development lifecycle.
5. **Legacy System Coexistence:**
    - For the initial rollout, ECAMS must coexist with legacy admission systems, creating dependencies on their data exports and user transition processes.

Impact of Changes

Any deviations from these assumptions or disruptions in the listed dependencies could lead to delays, increased costs, or rework in the project. Proactive measures must be taken to address potential risks associated with these factors.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface of the Engineering College Admission Management System (ECAMS) is designed to be user-friendly, intuitive, and accessible across various devices. Users will primarily interact with the system through a web-based interface. Below are the key characteristics and interaction methods of the interface:

Graphic of the User Interface

While a visual representation of the thermostat interface (from the project description) isn't applicable for this system, the interface design includes a main **dashboard view** for students and administrators, with responsive menus for streamlined navigation.

High-Level Interface Description:

1. **Login/Registration Page:**
    - Users access the system via a login page with options for account creation, password recovery, and secure login.
2. **Student Dashboard:**
    - A clean, visually appealing layout displays pending actions (e.g., completing application forms, uploading documents).
    - Navigation options are displayed as tabs or menu items, enabling access to application status, notifications, and payment history.
3. **Admin Dashboard:**
    - Features a tabular interface for managing applications, merit list generation, and system monitoring.
    - Includes dropdown menus and action buttons for approving/rejecting documents, sending notifications, and generating reports.
4. **Touch Screen and Menu Interactions:**
    - **Desktop/Tablet Users:**
      Interaction through click-based navigation, drop-down menus, and modal dialogs for form submissions.
    - **Mobile Users:**
      Optimized for touch screen gestures, such as tapping, swiping, and scrolling, ensuring accessibility on smaller screens.
5. **Interactive Features:**
    - **Form Auto-Save:** Application forms auto-save progress to prevent data loss.
    - **Real-Time Notifications:** Pop-ups or badges alert users of updates or required actions.

User Interaction Workflow:

1. Students log in and navigate to the dashboard to view their tasks or updates.
2. They fill out and submit application forms through dynamic fields.
3. Administrators access their dashboard to process applications, verify documents, and generate merit lists.

3.1.2 Hardware Interfaces

The Engineering College Admission Management System (ECAMS) is a web-based application primarily interacting with hardware devices to facilitate user interactions and system operations. Below is a description of the hardware interfaces the system supports:

Supported Devices

1. **User Devices:**
    - **Desktop Computers and Laptops:**
        - Operating systems: Windows, macOS, Linux
        - Interaction through a web browser (Chrome, Firefox, Safari, etc.).
    - **Mobile Devices and Tablets:**
        - Operating systems: Android, iOS
        - Interaction through touch gestures on a responsive web interface.
2. **Server Hardware:**
    - High-performance web and database servers to host the ECAMS backend and handle user requests.
        - Minimum specifications:
            - Processor: Quad-core 2.5 GHz
            - RAM: 16 GB
            - Storage: SSD with at least 500 GB capacity
3. **Peripherals:**
    - **Printers:**
        - Used for generating hard copies of admission forms, merit lists, and fee receipts.
        - Interface: Standard USB or network printing protocols.
    - **Scanners:**
        - For digitizing hardcopy documents uploaded by administrators or scanned at the institute.
        - Interface: USB or wireless.
4. **Payment Terminals:**
    - Devices integrated with payment gateways for offline transactions and syncing with the system database.

Data and Control Interactions

- **Device Input:**
    - Data is entered into the system via keyboards, touchpads, touchscreens, and external document uploads (via scanners).
- **Device Output:**

- Display on monitors or mobile screens, printed receipts from connected printers, and notifications sent via connected devices.

Hardware Integration:

All hardware components interact with the ECAMS via standard protocols. The system ensures seamless data flow between user devices, server hardware, and peripherals to support efficient admission processes.

3.1.3 Software Interfaces

The Engineering College Admission Management System (ECAMS) interacts with several software components to ensure seamless functionality and efficient communication. Below are the software interfaces integrated into the system:

Key Software Interfaces

1. **Web-Based User Interface:**
   - **Description:** The primary interface for students and administrators, accessible via web browsers.
   - **Interaction:**
     - Students submit applications, check status updates, and make payments.
     - Administrators manage admissions and send notifications through the dashboard.
   - **Technology:**
     - Frontend: React.js
     - Backend: Node.js with Express.js
2. **Mobile Application Interface:**
   - **Description:** A companion mobile app enabling students to send commands and receive notifications on the go.
   - **Interaction:**
     - Students can view application updates, submit documents, and make payments via the mobile app.
     - Push notifications inform users of status changes and deadlines.
   - **Technology:**
     - Backend API for app integration: RESTful API built with Express.js
     - Mobile Platforms: Android (Kotlin), iOS (Swift)
3. **Payment Gateway Integration:**
   - **Description:** Supports secure online payment for admission fees.
   - **Interaction:**
     - Students complete transactions via integrated payment services (e.g., Razorpay, Instamojo).
     - Payment status updates are synced with the application database.
4. **Database Management System:**
   - **Description:** Centralized storage for application data, user credentials, and payment records.
   - **Interaction:**

        - The application queries and updates data through an SQL database (MySQL).
    - **Technology:**
        - Database server hosted on the backend.
5. **Notification System:**
    - **Description:** Sends real-time alerts and updates via email or SMS.
    - **Interaction:**
        - Students and administrators receive notifications for application submissions, approvals, and fee payment deadlines.
    - **Technology:**
        - Email: Nodemailer
        - SMS: Twilio API

Interconnectivity and Commands:

The software components are interconnected through secure APIs to ensure consistent data flow and functionality. Commands initiated from the mobile app, such as submitting an application or checking updates, are processed by the backend and synchronized across all interfaces.

3.2 Functional Requirements

Functional requirements capture the specific behavior and functionalities the system must perform. Below is a detailed list of the functional requirements:

3.2.1 F1: Student Registration and Login

- **The system shall allow students to create an account by providing personal and academic details.**
- **The system shall validate user credentials during login and grant access to authenticated users.**

3.2.2 F2: Application Management

- The system shall enable students to fill out admission forms online.
- The system shall validate mandatory fields before allowing submission of the form.
- The system shall allow students to upload supporting documents in standard formats (PDF, JPG, PNG).

3.2.3 F3: Application Review and Status Tracking

- The system shall allow administrators to review and evaluate submitted applications.
- The system shall provide real-time updates to students about their application status (e.g., submitted, under review, approved, or rejected).

3.2.4 F4: Merit List and Seat Allocation

- The system shall generate merit lists based on predefined criteria, such as academic scores and entrance exam performance.
- The system shall allocate seats based on student preferences, merit rankings, and reservation policies.

3.2.5 F5: Fee Management

- The system shall enable students to pay fees securely through an integrated payment gateway.
- The system shall generate and store payment receipts for future reference.

3.2.6 F6: Notifications and Alerts

- The system shall send automated notifications (email and SMS) for critical updates, such as application status changes, payment reminders, and merit list announcements.
- The system shall allow administrators to send custom messages to specific users.

3.2.7 F7: Administrative Tools

- The system shall provide a dashboard for administrators to view statistics, manage applications, and track seat allocation.
- The system shall allow administrators to verify uploaded documents and mark them as valid or invalid.

3.2.8 F8: Security and Privacy

- The system shall encrypt sensitive user data, such as passwords and payment information.
- The system shall restrict access to certain features and data based on user roles (e.g., student, admin).

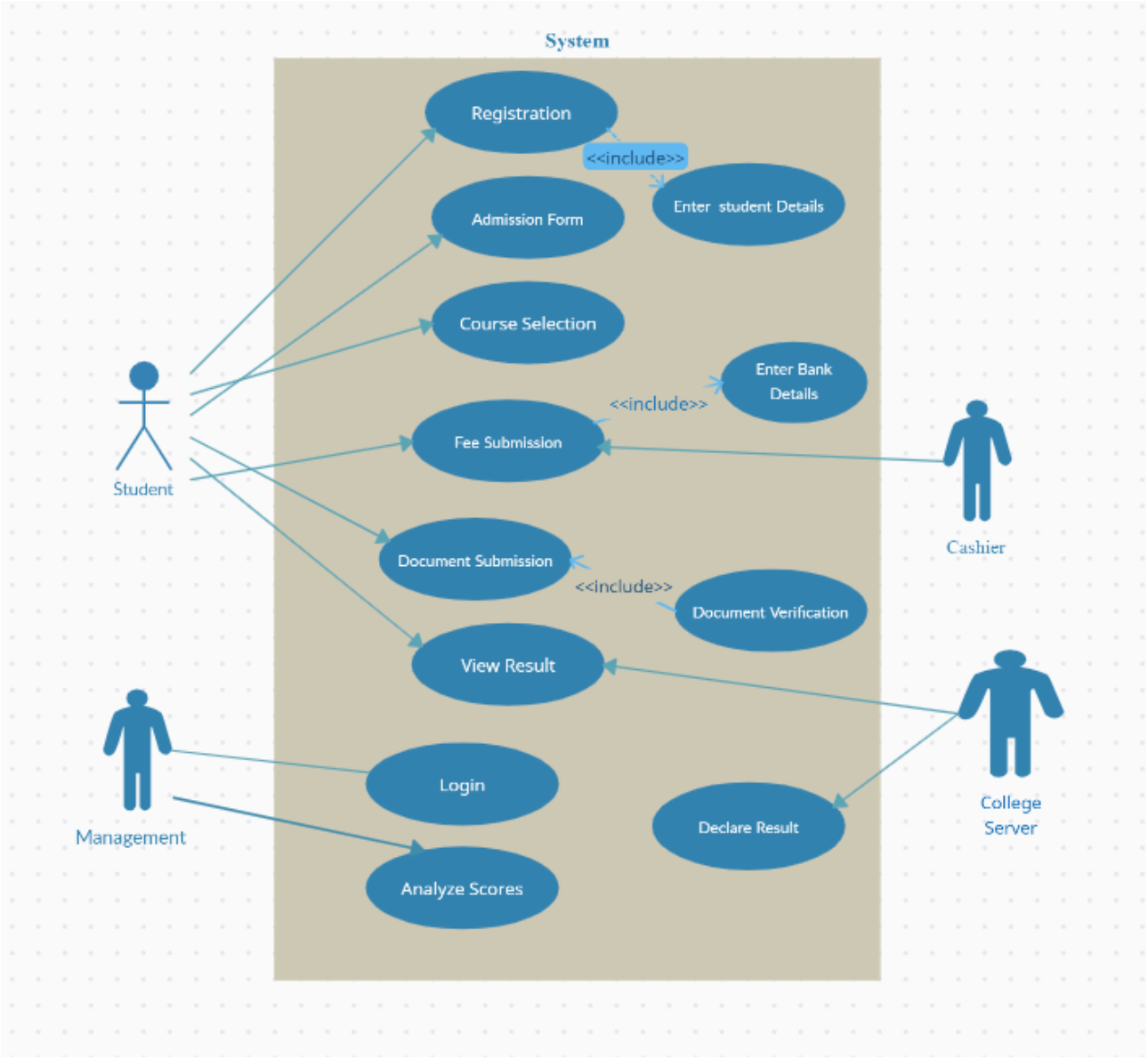
3.2.9 F9: Reporting and Analytics

- The system shall generate reports on application data, admission statistics, and financial transactions.
- The system shall provide filters for generating customized reports (e.g., by department, category, or admission year).

3.2.10 F10: Integration with Mobile Application

- The system shall synchronize data with the mobile application in real time.
- The system shall provide push notifications to mobile users for updates like merit list releases and payment deadlines.

3.3 Use Case Model



3.3.1 Use Case #1: Student Registration and Login (U1)

Author: Anushka Bachhav

Purpose:
Allow students to register and securely log into the system to access admission-related features.

Requirements Traceability:

- F1: Student Registration and Login

Priority: High

Preconditions:

- The student has access to the system's registration portal.
- The student has a valid email address and mobile number.

Postconditions:

- A new account is created, and login credentials are stored securely.
- The student is redirected to the dashboard after successful login.

Actors:

- Primary: Student
- Secondary: System Database

Extends: N/A

Flow of Events:
Basic Flow:

1. The student accesses the registration portal.
2. The system prompts for personal information (name, email, password, etc.).
3. The system validates the inputs and creates a new account.
4. The student logs in using the registered credentials.

Alternative Flow:

- If the email is already registered, the system prompts the student to log in or reset the password.

Exceptions:

- Invalid email format: The system displays an error message.
- Weak password: The system enforces password rules (e.g., minimum 8 characters).

Includes: N/A

Notes/Issues:

- Consider enabling third-party login options (e.g., Google).
- Include CAPTCHA to prevent bot registrations.

3.3.2 Use Case #2: Application Submission (U2)

Author: Krishna Magar

Purpose:
Allow students to complete and submit the admission application form.

Requirements Traceability:

- F2: Application Management

Priority: High

Preconditions:

- The student is logged into the system.
- The student has all required documents ready for upload.

Postconditions:

- The application form is successfully submitted and stored in the database.
- The system generates a confirmation receipt for the student.

Actors:

- **Primary:** Student
- **Secondary:** Administrator

Extends: N/A

Flow of Events:
Basic Flow:

1. The student accesses the application form from their dashboard.
2. The system displays the form fields for personal, academic, and preference details.
3. The student fills out the form and uploads the required documents.
4. The system validates the data and confirms submission.

Alternative Flow:

- If any mandatory field is left empty, the system highlights the field and prevents submission.

Exceptions:

- File upload exceeds size limit: The system displays an error message.
- Network failure: The system saves the progress and prompts the student to retry.

Includes:

- U1: Student Registration and Login

Notes/Issues:

- Ensure document uploads are virus scanned.

3.3.3 Use Case #3: Fee Payment (U3)

Author: Omkar Kasture

Purpose: Facilitate secure fee payment for the admission process.

Requirements Traceability:

- **F5: Fee Management**

Priority: Medium

Preconditions:

- The student has a submitted application.
- The payment gateway is configured and operational.

Postconditions:

- Payment is processed, and a receipt is generated.
- The system updates the payment status in the student's account.

Actors:

- Primary: Student
- Secondary: Payment Gateway

Extends: N/A

Flow of Events:
Basic Flow:

1. The student selects the "Pay Fees" option from the dashboard.
2. The system redirects to the payment gateway.
3. The student completes the payment and is redirected back.
4. The system confirms payment and generates a receipt.

Alternative Flow:

- If payment fails, the system displays an error and prompts for retry.

Exceptions:

- Payment gateway timeout: The system logs the error and notifies the student.

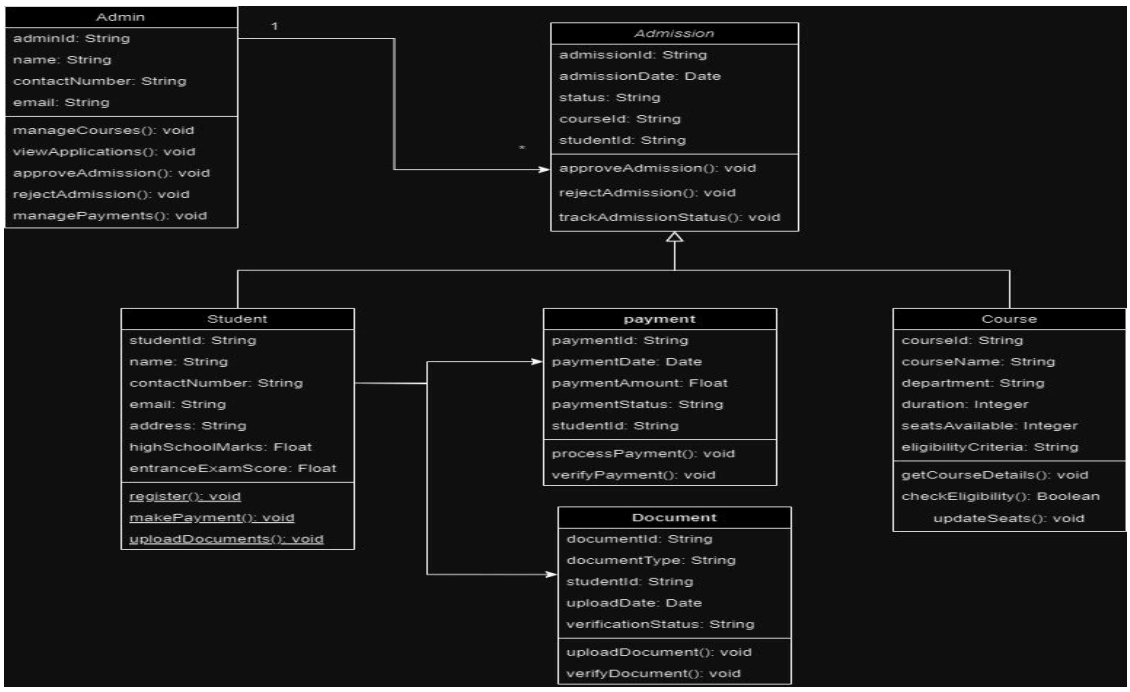
Includes:

- U2: Application Submission

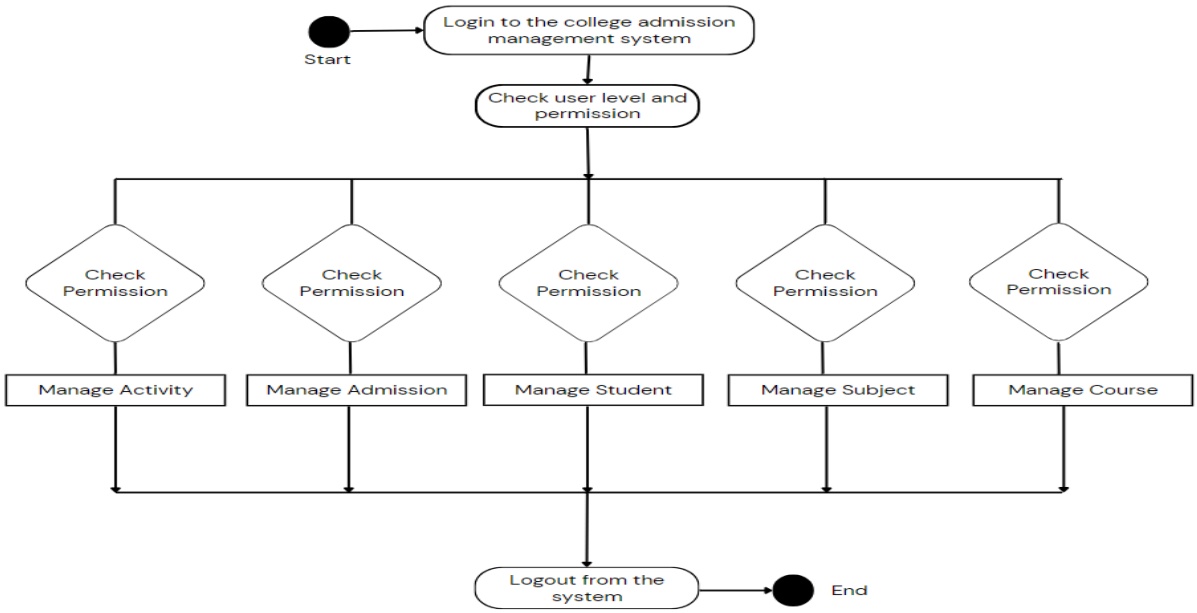
Notes/Issues:

- Provide support for multiple payment methods (credit card, UPI, net banking).

Class Diagram



Activity Diagram



4 Other Non-functional Requirements

4.1 Performance Requirements

P1: The system shall process user registration requests within **2 seconds** to ensure a smooth experience for applicants.

P2: The application form submission process shall be completed within **5 seconds**, including data validation and storage.

P3: The system shall retrieve applicant details for administrative review within **3 seconds** after a query is initiated.

P4: The allocation of seats for a selected course shall be performed within **10 seconds** after the final admission confirmation.

P5: The system shall support concurrent access for up to **500 users** during peak application periods without degradation in performance.

P6: Email notifications for successful form submission or status updates shall be triggered within **30 seconds** of the corresponding system event.

P7: The system shall generate reports (e.g., application statistics or seat allocation) within **15 seconds** for administrative use.

P8: The payment gateway shall process application fees and return transaction statuses within **10 seconds** to ensure smooth payment operations.

P9: The system shall handle data backups automatically without affecting ongoing operations, with a backup completion time of no more than **30 minutes** per session.

P10: The system dashboard shall load within **3 seconds**, displaying relevant metrics and information for administrators.

4.2 Safety and Security Requirements

Safety Requirements:

- **S1**: The system shall ensure that no user data is lost or corrupted during power failures or unexpected shutdowns by implementing automatic data backup and recovery mechanisms.
- **S2**: The system shall prevent data input errors (e.g., incorrect GPA values or invalid document formats) by validating all inputs before processing.
- **S3**: The system shall implement safeguards to avoid overwriting critical data, such as admission decisions, without explicit administrative approval.

- **S4**: An automatic session timeout shall be implemented for inactive users, logging them out after **10 minutes** of inactivity to prevent unauthorized access.

Security Requirements:

- **SE1**: The system shall enforce **role-based access control (RBAC)** to restrict access to sensitive modules (e.g., administrative dashboards, applicant personal data) based on user roles.
- **SE2**: All user connections to the system, including mobile and web access, shall be secured with **HTTPS** to encrypt data in transit.
- **SE3**: The system shall require multi-factor authentication (MFA) for administrative users to access the system.
- **SE4**: Passwords shall be stored securely using **hashing algorithms (e.g., bcrypt)**, ensuring that no plaintext passwords are stored.
- **SE5**: The system shall comply with applicable privacy regulations such as **GDPR** or **India's IT Act**, ensuring protection and confidentiality of user data.
- **SE6**: The database shall be configured with **encryption at rest** to protect stored applicant and administrative data.
- **SE7**: Unauthorized attempts to access restricted areas of the system shall trigger alerts to administrators and log the event for auditing.
- **SE8**: Payment transactions shall adhere to **PCI DSS** standards, ensuring secure processing and storage of payment information.
- **SE9**: The system shall provide an option for applicants to request account deletion, ensuring compliance with privacy laws.

Compliance and Certification:

- The system must adhere to institutional policies for data security and privacy.
- Security testing and certification shall be performed periodically to identify vulnerabilities and ensure compliance with security standards.
- The product must be designed to comply with safety standards to protect against loss or unauthorized disclosure of user data.

4.3 Software Quality Attributes

4.3.1 Reliability

The ECAMS shall ensure dependable operation to support critical admission processes, including application submission, evaluation, and decision-making.

- **Requirements:**
- **High Uptime:** The system shall maintain an uptime of **99.9%**, particularly during peak admission periods.

- **Data Integrity:** Admission data (student profiles, documents, payment records) shall be protected against loss or corruption, ensuring a recovery point objective (RPO) of 15 minutes.
- **Error Handling:** The system shall identify and log errors in real-time, with user-friendly error messages to guide users through resolution steps.
- **Redundancy:** Backup mechanisms for databases and servers shall ensure minimal downtime in case of hardware or software failures.

- **Implementation Steps:**
    - Employ load balancing and server replication to distribute workloads evenly and ensure redundancy.
    - Schedule regular database integrity checks and system audits to ensure data consistency.
    - Integrate automated monitoring tools like **New Relic** or **Prometheus** to track system health.

4.3.2 Adaptability

The system shall support customization and scalability to accommodate future requirements, such as policy changes or increased application volumes.

- **Requirements:**
- **Modular Architecture:** ECAMS shall be designed with modular components (e.g., separate modules for admissions, payments, and reporting), enabling the addition or replacement of functionalities without affecting existing operations.
- **Scalability:** The system shall scale horizontally and vertically to handle surges in user traffic, supporting up to **10,000 concurrent users** during peak periods.
- **Flexible Configuration:** All dynamic system settings, such as admission deadlines or fee structures, shall be stored in configuration files or a database for easy updates.
- **Integration Readiness:** ECAMS shall allow integration with external APIs, such as third-party verification services, without extensive reconfiguration.

- **Implementation Steps:**
    - Use cloud-based infrastructure (e.g., AWS, Azure) to enable elastic scaling during high-demand periods.
    - Define clear APIs for third-party integrations to simplify future extensions.
    - Incorporate configuration management tools like **Ansible** or **Terraform** to automate system updates.

4.3.3 Usability

ECAMS shall prioritize user satisfaction by ensuring ease of access and efficient interaction for applicants, administrators, and evaluators.

- **Requirements:**
- **User-Centered Design:** The interface shall be intuitive, reducing the learning curve for new users.
- **Accessibility Standards:** The system shall comply with **WCAG 2.1 Level AA** to ensure usability for users with disabilities.

- **Responsive Design:** ECAMS shall adapt to various device screens, including desktops, tablets, and smartphones.
- **Interactive Tutorials:** The system shall provide onboarding tutorials and tooltips for new users.

- **Implementation Steps:**
    - Conduct usability testing with target users to identify and address pain points.
    - Develop a responsive front-end using frameworks like **Bootstrap** or **Tailwind CSS**.
    - Implement accessibility best practices, such as ARIA roles and keyboard navigation.

4.3.4 Maintainability

The system shall be designed for efficient maintenance and continuous improvement to accommodate future requirements and updates.

- **Requirements:**
- **Version Control:** All code and configurations shall be managed using a system like **Git**, ensuring clear documentation of changes.
- **Code Modularity:** Modular coding practices shall ensure individual components can be updated or replaced independently.
- **Documentation:** Comprehensive system and API documentation shall be maintained to support developers during troubleshooting and upgrades.
- **Regular Updates:** ECAMS shall undergo regular security and feature updates to remain compatible with new technologies.

- **Implementation Steps:**
    - Schedule regular codebase reviews to identify areas for improvement.
    - Use automated deployment pipelines (e.g., CI/CD) to streamline updates and minimize downtime.
    - Maintain a backlog for feature requests and prioritize updates based on user feedback.

5 Other Requirements

5.1 Database Requirements

- **Relational Database:**
  The system shall use a **relational database management system (RDBMS)** like MySQL or PostgreSQL for structured storage and retrieval of admission data.
- **Data Backup and Recovery:**
  Regular automated backups shall be configured, with a retention period of **6 months**, to prevent data loss and enable recovery during system failures.
- **Data Encryption:**
  Sensitive data, such as applicant personal details and payment records, shall be encrypted at rest and in transit using **AES-256** encryption for compliance and security.
- **Database Scalability:**
  The database schema shall support scaling to handle up to **1 million records**, accommodating data from multiple academic years.

5.2 Internationalization Requirements

- **Language Support:**
  ECAMS shall support multiple languages (e.g., English, Hindi, Marathi) to cater to diverse user demographics.
- **Date and Time Formats:**
  The system shall display dates and times according to the **Indian Standard Time (IST)** format, with options to localize for international users if required.
- **Currency Support:**
  All financial transactions shall be displayed in **Indian Rupees (INR)**, with provisions to include international payment gateways in the future.

5.3 Legal and Compliance Requirements

- **Data Privacy:**
  ECAMS shall comply with the **Indian IT Act, 2000** and any applicable privacy regulations (e.g., GDPR if handling international applicants).
- **Audit Logs:**
  The system shall maintain an audit trail of all user interactions and administrative actions for at least **one year**, ensuring accountability and compliance with institutional policies.
- **Accessibility Compliance:**
  The system shall adhere to **WCAG 2.1 Level AA** to ensure it is accessible to users with disabilities.

5.4 Reuse Objectives

- **Code Reusability:**
  The project shall employ a modular design to enable the reuse of components like authentication, payment processing, and notification services across other educational platforms.

- **API Reusability:**
  The APIs developed for ECAMS shall follow RESTful principles and include clear documentation, making them reusable for other systems within the institution.

5.5 Environmental Requirements

- **Cloud Deployment:**
  ECAMS shall be hosted on a cloud infrastructure, such as AWS or Azure, to ensure scalability, high availability, and environmental sustainability.
- **Power Efficiency:**
  The system shall use energy-efficient computing practices, including optimized queries and server configurations, to minimize its carbon footprint.

5.6 Integration Requirements

- **Third-Party Systems:**
  ECAMS shall integrate with external systems, including:
  - **Government APIs** for student verification (e.g., Aadhaar, Digilocker).
  - **Payment Gateways** for online fee collection.
  - **Email/SMS Services** for notifications to applicants.

5.7 Disaster Recovery Requirements

- **Failover Systems:**
  ECAMS shall implement a failover mechanism to ensure uninterrupted service in case of primary server failure.
- **Recovery Time Objective (RTO):**
  System downtime during a disaster shall not exceed **15 minutes**.
- **Recovery Point Objective (RPO):**
  The maximum acceptable data loss shall be limited to **5 minutes**.

Appendix A – Data Dictionary

Type	Variable Name	Description	Possible Values/States	Operations /Functional Requirements	Related Requirements
Constant	MAX_APPLICATIONS	Maximum number of applications the system can handle simultaneously.	Integer (e.g., 10,000)	Used for load balancing and system scalability checks.	Scalability requirements
Input	applicant_name	Full name of the applicant.	String	Collected during application submission.	F1: Application submission
Input	date_of_birth	Applicant's date of birth.	YYYY-MM-DD	Validated to ensure the applicant meets age criteria.	F3: Validation checks
Input	email_address	Applicant's email address.	String(valid email format)	Used for sending confirmation emails and notifications.	F2: Communication with applicants
State Variable	application_status	Tracks the status of an application.	Pending, Under Review, Approved, Rejected	Updated based on admission process steps.	F7: Application tracking
State Variable	payment_status	Tracks whether the admission fee has been paid.	Pending, Paid, Failed	Updated after transaction response from payment gateway.	F6: Payment processing
Output	error_message	Error message displayed for invalid inputs or system errors .	String (e.g., "Invalid email address")	Shown to users when validation checks fail.	F3: Validation checks

Appendix B - Group Log

Group Meeting Logs

Date	Time	Duration	Attendees	Agenda	Outcome
11/05/2024	10:00 AM	1 hour	Ayush, Krishna, Anushka, Omkar	Initial discussion on project scope and understanding client requirements.	Agreed on the project scope and identified key functional and non-functional requirements.
11/10/2024	03:00 PM	1.5 hours	Ayush, Krishna, Anushka, Omkar	Divided responsibilities for document sections and scheduled future meetings.	Delegated tasks for SRS document preparation and set deadlines for completion.
11/15/2024	11:30 AM	2 hours	Ayush, Krishna, Omkar	Reviewed functional requirements and use case diagrams.	Finalized functional requirements and completed initial use case diagrams.
11/20/2024	04:00 PM	1 hour	Krishna, Anushka, Omkar	Worked on performance requirements and discussed quality attributes.	Drafted initial content for performance and quality attribute sections.
11/25/2024	02:00 PM	2 hours	Ayush, Krishna, Anushka, Omkar	Proofread and reviewed the document for consistency and formatting.	Incorporated final edits and ensured alignment with project objectives.
12/01/2024	05:00 PM	1.5 hours	Ayush, Krishna, Anushka, Omkar	Final review meeting to consolidate feedback and finalize the SRS document.	Ensured the document met client expectations and prepared for submission.

Group Activities

Activity	Date	Participants	Details
Requirement Gathering	11/05/2024	Ayush, Krishna, Anushka, Omkar	Conducted client interviews to understand system requirements.
Document Drafting	11/10/2024	Ayush, Krishna	Prepared initial drafts for functional requirements and system description sections.
Use Case Diagram Creation	11/15/2024	Krishna, Omkar	Developed and reviewed use case diagrams for clarity and completeness.
Performance Requirements Review	11/20/2024	Krishna, Anushka	Collaboratively wrote and revised performance requirements for the system.
Team Collaboration Workshop	11/25/2024	Ayush, Krishna, Anushka, Omkar	Discussed challenges, shared updates on progress, and resolved any blockers.
Final Document Assembly	12/01/2024	Anushka	Compiled all sections into a single cohesive document, ensuring formatting consistency.

Challenges and Resolutions

Challenge	Date	Description	Resolution
Misalignment in functional requirements	11/10/2024	Discrepancies in understanding some client requirements.	Scheduled an additional client interview to clarify expectations.
Delayed submissions from team members	11/20/2024	Some members struggled to meet deadlines due to workload.	Reassessed workload distribution and adjusted deadlines accordingly.
Diagram inconsistencies	11/25/2024	Initial diagrams lacked uniformity in formatting and symbols.	Standardized diagram templates and reviewed all diagrams for consistency.

Effort Summary

- **Total Hours Spent:** ~25 hours
- **Number of Meetings:** 6
- **Collaborative Tools Used:** Google Docs, Slack, Trello