SOFTWARE REQUIREMENT SPECIFICATION

SREEHARI S	B220548CS
JEEV JOE JAISON	B220335CS
CHRISS CHARLS	B220224CS

TABLE OF CONTENTS

1. INTRODUCTION---------- 2

1.1 Purpose---------- 2

1.2 Scope----------2

2.USE CASE DIAGRAM---------- 3

3. FUNCTIONAL REQUIREMENTS ---------- 4

3.1 User Authentication & Access Control----------4

3.2 Student View----------4

3.3 Faculty View----------4

3.4 Admin View----------5

4. NON-FUNCTIONAL REQUIREMENTS ----------6

4.1 Performance Requirements----------6

4.2 Safety Requirements----------6

4.3 Logging & Monitoring----------6

4.4 Maintainability & Scalability----------6

4.5 Compatibility & Portability----------6

4.6 Compliance & Legal Requirements----------7

4.7 Integration Requirements----------7

4.8 Reliability Requirements----------7

5. DATABASE DESIGN----------8

6. CLASS DIAGRAM----------9

1 Introduction

1.1 Purpose

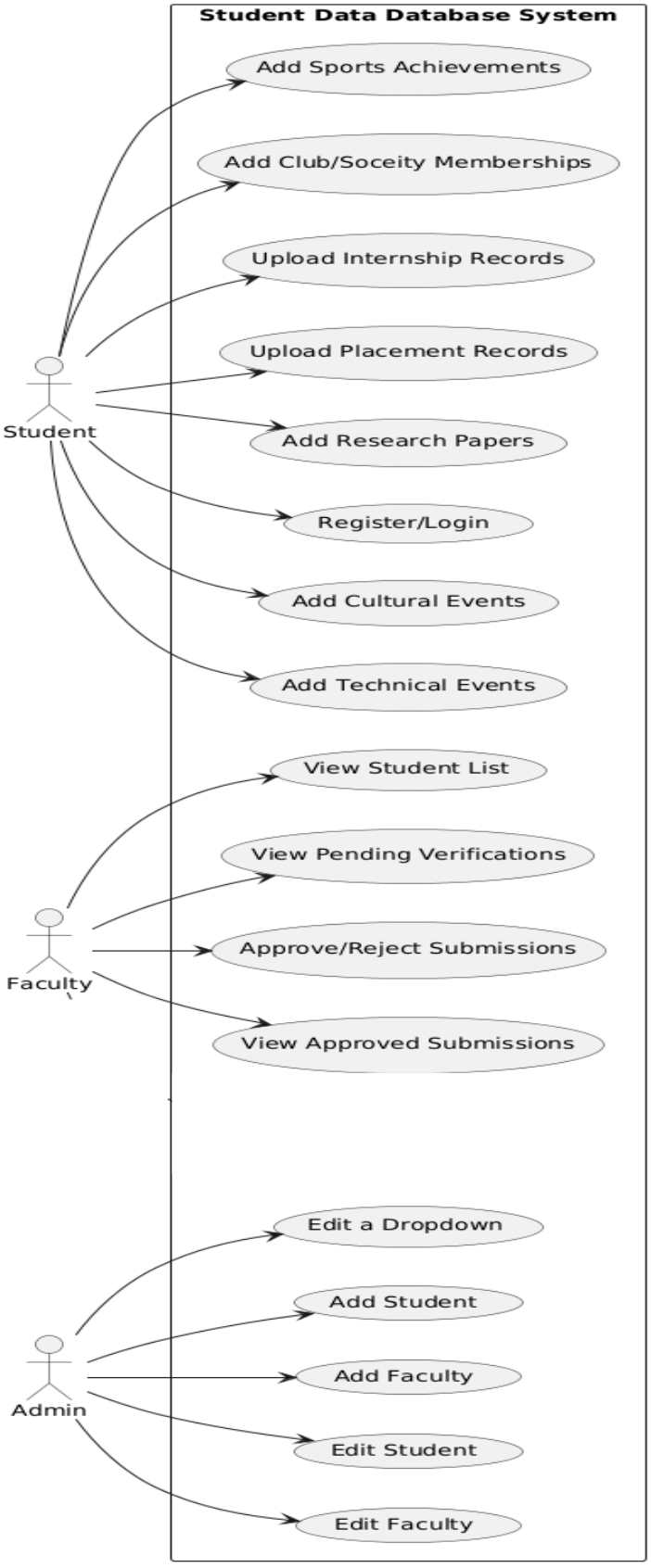
The purpose of this system is to provide a streamlined and secure platform for students, faculty, and administrators to manage academic achievements, student profiles, and research-related data. The system will facilitate easy access to research papers, achievement tracking, student profile management, and efficient filtering for faculty and admin users. By implementing role-based authentication, document handling, and search functionalities, the platform aims to enhance the overall academic management process.

1.2 Scope

This system is designed to support students, faculty, and administrators in a university environment. It provides functionalities such as:

- **Student Features:** Data management, document uploads, and research paper search.
- **Faculty Features:** Viewing assigned students, approving/rejecting student forms, and exporting student records.
- **Admin Features:** User management and dropdown customization for dynamic data entry.
- **Security & Performance:** Role-based authentication, encrypted data storage, and optimized backend performance using Spring Boot.
  The system will be developed using Spring Boot and will be scalable for future integrations with the college's central database.

2 Usecase Diagram



3 Functional Requirements

3.1 User Authentication & Access Control

F1: Role-Based Authentication:

- Implement authentication for **Students, Faculty, and Admins**.
- Assign specific roles and permissions to each category.

F2: Forgot Password & Email-Based Reset:

- Implement a password recovery system with **email verification**.

3.2 Student view

F3: Achievement Management

- Allow students to add or delete new achievements (e.g., certificates, publications, projects).

F4: Student Forms & Profile Management

- Enable students to upload PDFs, images, and certificates securely.

F5: Searching

- **Student Search for Research Papers:**
  - Students can search for faculty-published research papers by title, keywords, or author name.
  - Display results, and students can add relevant details to their forms.

3.3 Faculty View

F6: View Assigned Students:

- Faculty can access a list of assigned students based on their **department**.
- Search for specific students by **name, roll number, or department**.

F7:Export Student Details to Excel:

- Faculty can download student records in **Excel format** for offline use.

F8:Approve/Reject Student Forms:

- Faculty can review student-submitted **achievement forms**.
- Option to **approve, reject, or request modifications** with feedback

3.4 Admin View

F9:User Management:

- Create, modify, or delete **student and faculty accounts**.
- Reset passwords and update user details.

F10:Dropdown Customization:

- Manage dynamic dropdown values (e.g., adding new categories in achievements).

F11: Filtering:

- Admins can filter students based on their year and department and manage them accordingly.

4 Non Functional Requirements

4.1 Performance Requirements

- The system should handle at least 50 concurrent users without noticeable lag.
- The backend should be optimized using Spring Boot caching and indexing techniques to improve performance.
- The application should support asynchronous processing for background tasks (e.g., report generation).

4.2 Security Requirements

- Role-Based Access Control (RBAC) should be implemented to restrict access based on user roles (Student, Faculty, Admin).
- Passwords must be encrypted using a secure hashing algorithm like bcrypt or SHA-256.
- Sensitive fields (e.g., student personal details, academic records) should be encrypted before storage.
- Implement session timeouts after 10 minutes of inactivity to prevent unauthorized access.
- Prevent SQL Injection, XSS, and CSRF attacks using Spring Security best practices.
- Secure all API endpoints with JWT authentication.

4.3 Logging & Monitoring

- Implement centralized logging using SLF4J and Logback in Spring Boot.
- Store logs in both console and file formats for debugging and tracking system activity.
- Maintain separate logs for errors, access logs, and security-related events.
- Log important events such as:
- User login/logout attempts
- Failed login attempts (to detect possible brute force attacks)
- System errors and exceptions

4.4 Maintainability & Scalability

- The codebase should follow modular principles (Controller-Service-Repository pattern) for easy updates and maintenance.
- The system should be scalable to support an increasing number of students, faculty, and admins.
- API and database queries should be optimized to handle large datasets efficiently.

4.5 Compatibility & Portability

- The system should be compatible with all modern browsers (Chrome, Firefox, Edge, Safari).
- It should be platform-independent, running smoothly on Windows, macOS, and Linux.

4.6 Compliance & Legal Requirements

- The system must comply with data protection laws.
- Student records should not be stored longer than necessary, following institutional policies.

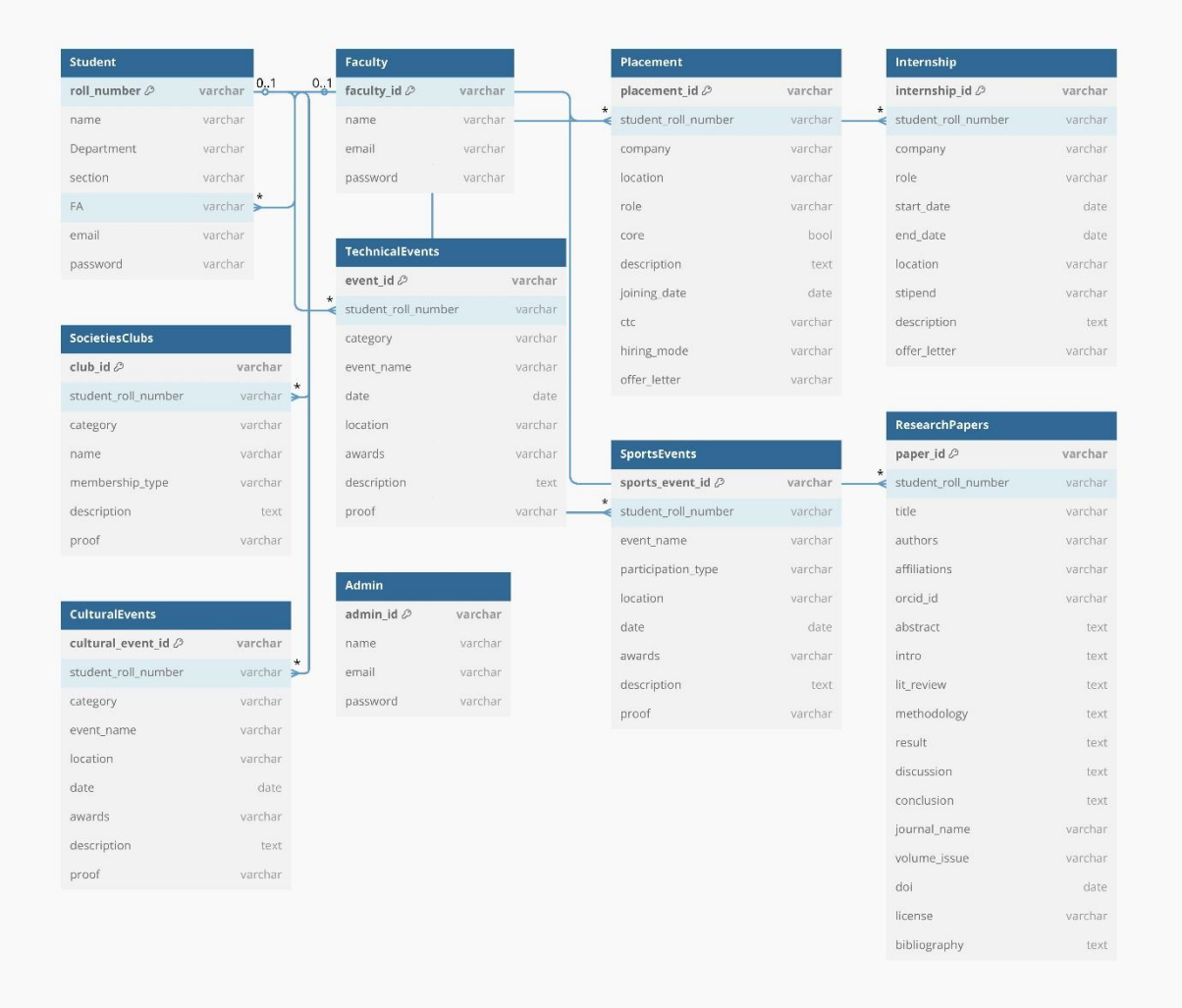
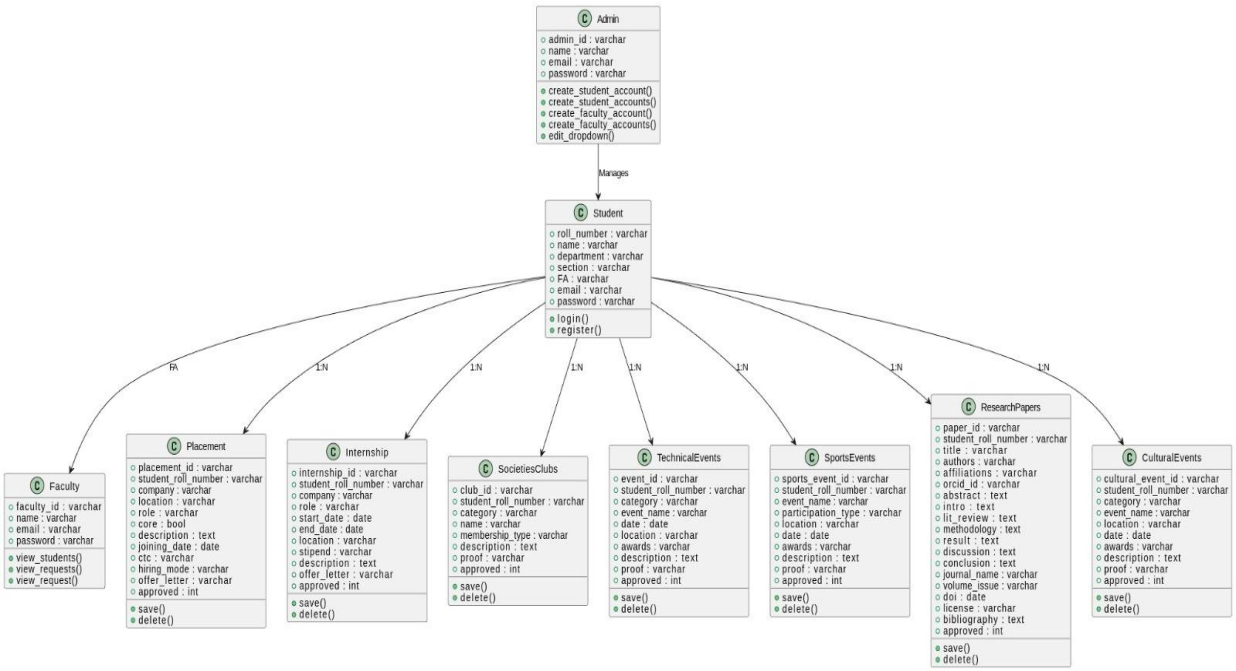
4.7 Integration Requirements

- The system should be designed for future integration with the college's central database.
- APIs should be built using RESTful principles for easy integration with external systems.
- Ensure standardized data formats (JSON/XML) to allow seamless communication with other services.

4.8 Reliability Requirements

- The system should automatically restart in case of a server crash using Docker.
- Logs should be stored securely so that debugging information is not lost after a restart.

5 Database Diagram



6 Class Diagram