Software Requirements Specification - Done By LAZREK NASSIM

SAMS (Student Academic Management System)

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a complete description of the Student Academic Management System (SAMS). It details the functional and non-functional requirements for the system, which will serve students, faculty members, and administrators in managing academic operations efficiently. This document is intended for the development team, project stakeholders, and the main parties who will be involved in designing, implementing, and testing the system.

1.2 Scope

SAMS is a comprehensive academic management solution designed to streamline and enhance the educational experience for students, faculty, and administrators. The system provides a unified ecosystem for managing most aspects of academic life within an educational institution.

Product Name: SAMS (Student Academic Management System)

Key Benefits:

- Centralized academic information management
- Automated course registration
- Real-time grade and GPA tracking
- Integrated financial management with payment processing
- Enhanced communication between students, faculty, and administration
- Streamlined degree progress

Scope of Features:

The system encompasses three primary portals:

1. **Student Portal** - Enables students to register for courses, track academic progress, manage financial obligations, join study groups, and receive intelligent notifications.
2. **Faculty Portal** - Allows instructors to create course offerings, post materials, enter grades, and communicate with students.
3. **Administrator Portal** - Provides system administrators with tools to manage user accounts, validate payments, grant course access, and oversee academic operations.

SAMS will initially support up to 100 students with scalability considerations for future expansion. The system integrates course registration, grade management, financial transactions, and collaborative tools into a single, cohesive platform.

1.3 References

- IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications

- Spring Boot Framework Documentation
- Vue.js Framework Documentation
- PostgreSQL Database Documentation
- Spring Security Documentation

1.4 Overview

The remainder of this document provides a detailed description of SAMS. Section 2 presents an overall description of the system, including product perspective, functions, user characteristics, constraints, and assumptions. Section 3 specifies all functional and non-functional requirements in detail, along with use case diagrams and narratives. Section 4 contains supporting appendices, and Section 5 provides an index for quick reference.

2 Overall Description

2.1 Product Perspective

SAMS is a new, self-contained system that will operate as the primary academic management platform for an educational institution. The system is designed with a modern three-tier architecture.

System Context:

SAMS operates within the educational institution's technology ecosystem and interfaces with users through a web-based platform accessible via standard web browsers. The system is built on a contemporary technology stack that ensures reliability, scalability, and security.

Architecture Overview:

The system follows a layered architecture pattern:

- **Presentation Layer**: Vue.js frontend provides an interactive, responsive user interface that communicates with the backend through REST API calls
- **Application Layer**: Spring Boot controllers serve as the REST API layer, handling HTTP requests from the Vue.js frontend and converting them into service method calls
- **Business Logic Layer**: Spring Boot services contain the core business logic and processing rules.
- **Data Access Layer**: Spring Boot interfaces with PostgreSQL database for persistent data storage
- **Security Layer**: Spring Security handles authentication and authorization, integrated with Vue.js login flows to ensure secure access control

External Interfaces:

The system interfaces include:

- **User Interface**: Web-based interface accessible through modern browsers (Chrome, Firefox, Safari, Edge)
- **Payment Processing**: Integration points for credit card and bank transfer payment validation not initially but for future expansion
- **Database**: PostgreSQL relational database for persistent storage of all system data

The REST API architecture allows the Vue.js frontend to remain decoupled from the Spring Boot backend, enabling independent development and testing of each layer before integration.

2.2 Product Functions

SAMS provides comprehensive academic management capabilities organized into distinct functional areas:

2.2.1 Academic Management Features

Course Registration & Scheduling

Students can search for available courses, check prerequisites, view class schedules, and register for courses. The system prevents conflicts (time overlaps, prerequisite violations) and manages waitlists when courses are full.

Example: Sarah searches for "Data Structures," sees it's offered MWF 9:00-9:50 AM, checks that she's completed the prerequisite CS101, and registers with one click.

Real-time GPA & Transcript Management

The system automatically calculates and updates student GPAs as grades are entered. It generates official transcripts showing all completed courses, grades, and cumulative statistics.

Example: When Professor Martinez enters grades for CS301, the system immediately updates each student's semester and cumulative GPA, and students can instantly view their updated academic standing.

Automated Degree Progress Tracking

SAMS monitors student progress toward graduation requirements, showing completed courses, remaining requirements, and projected graduation date. It alerts students about missing prerequisites or requirements.

Example: The system shows Mike has completed 89/120 credits, needs 2 more science electives, and is on track to graduate Spring 2026.

Financial Aid & Billing Integration

The system displays tuition charges, financial aid awards, payment due dates, and account balances. Each course has its own fees, and students must pay the total amount for all enrolled courses before the semester starts. Payment can be made via credit card or bank transfer. Once payment is validated by system administrators, students receive access to course materials.

Study Group & Collaboration Tools

Students can create and join study groups for specific courses, view group member contact information, and send messages within the group.

Example: Sarah creates a "CS301 Study Group," invites classmates, and sends messages to coordinate study sessions. Members can share contact details.

Intelligent Notifications & Analytics

The system sends automated alerts for important deadlines, grade updates, registration periods, and academic milestones.

Example: Students receive notifications 24 hours before assignment deadlines, alerts when grades are posted, and reminders when registration opens.

2.2.2 User Authentication System

Secure login system with role-based access control. Users authenticate with username/password and are directed to the appropriate portal based on their role.

2.2.3 Multi-User System Architecture

Student Portal

Primary users: Enrolled students

Key functionalities:

- View personal academic dashboard with current courses and grades
- Register for courses and manage class schedules
- Track degree progress and graduation requirements
- Access financial aid information and billing statements
- Join study groups and communicate with classmates
- Receive personalized notifications and alerts

Faculty Portal

Primary users: Professors, instructors, teaching assistants

Key functionalities:

Course Management

- Create and modify course offerings (schedules, capacity, prerequisites)
- Manage course rosters and enrollment lists
- Post course materials, syllabi, and announcements

Grade Management

- Enter and update student grades for assignments, exams, and final grades
- Submit final grades at semester end

Student Interaction

- View enrolled student profiles and academic histories
- Communicate with students through integrated messaging
- Track student attendance and participation

Administrator Portal

Primary users: System administrators

Key functionalities:

User Management

- Create and manage student, faculty, and staff accounts
- Assign roles and permissions across the system
- Manage security settings and access controls
- Validate payment and grant course access

Academic Operations

- Oversee all information about student profiles and faculty staff profiles

2.3 User Characteristics

SAMS will serve three distinct user groups, each with different technical capabilities and system usage patterns:

Students

- **Technical Skill Level**: Basic to intermediate computer literacy
- **Expected Usage**: Daily access during academic semesters for course management, grade checking, and communication
- **Education Level**: Currently enrolled undergraduate or graduate students
- **System Familiarity**: Users will require minimal training; the interface should be intuitive and self-explanatory
- **Number of Users**: Initial deployment targets 100 students
- **Key Needs**: Quick access to grades, easy course registration, clear degree progress tracking, good payment processing

Faculty Members

- **Technical Skill Level**: Intermediate computer skills
- **Expected Usage**: Regular access for course management, grade entry, and student communication throughout the semester
- **Education Level**: Advanced degrees (Master's or Doctorate)
- **System Familiarity**: Willing to learn new systems but prefer straightforward interfaces

- **Key Needs**: Efficient grade entry, simple course material posting, streamlined communication tools

Administrators

- **Technical Skill Level**: Advanced computer skills with understanding of system operations
- **Expected Usage**: Frequent access for user management, payment validation, and system oversight
- **Education Level**: Bachelor's degree or higher in relevant fields
- **System Familiarity**: Expected to be power users who understand system architecture and data relationships
- **Key Needs**: Comprehensive user management tools, efficient payment processing workflows, detailed oversight capabilities, robust security controls

All user groups expect responsive performance, data security, and reliable system availability. The interface design should accommodate varying technical skill levels while maintaining efficiency for power users.

2.4 Constraints

The development and operation of SAMS are subject to several constraints that will influence design decisions and implementation strategies:

Technical Constraints

- **Technology Stack**: The system must be built using Vue.js for the frontend, Spring Boot for the backend, and PostgreSQL for the database as specified in the architecture
- **Browser Compatibility**: The web application must function correctly on modern versions of Chrome, Firefox, Safari, and Edge browsers
- **Network Requirements**: The system requires stable internet connectivity for all operations; no offline functionality is supported
- **Database**: PostgreSQL is the mandated relational database management system
- **API Architecture**: The backend must implement RESTful API principles with Spring Boot handling all HTTP requests from Vue.js

Operational Constraints

- **Development Approach**: Backend-first development strategy is required - Spring Boot services must be built and tested with PostgreSQL before Vue.js frontend integration
- **Security Standards**: Spring Security must be implemented for all authentication and authorization operations
- **Payment Processing**: Initial implementation requires manual administrator validation of payments rather than automated payment gateway integration

- **Scalability**: Initial deployment must support 100 students with architecture allowing future expansion
- **Data Privacy**: Student academic and financial information must be protected according to educational data privacy standards

Implementation Constraints

- **Real-time Communication**: Decision on chat/forum implementation (WebSockets, Firebase, or simple forum) will be determined based on development timeline
- **File Upload**: Initial version will support simple text content upload; document upload features may be added if time permits
- **Development Timeline**: Features must be prioritized based on available development time
- **Testing Requirements**: Each architectural layer must be validated independently before integration

Resource Constraints

- **Development Team**: Limited to available team members and their skill sets
- **Infrastructure**: System must operate within available hosting and infrastructure resources
- **Memory and Performance**: System operations must be optimized for efficient resource utilization

These constraints define the boundaries within which the system must be designed and implemented, ensuring realistic expectations and achievable deliverables.

2.5 Assumptions and Dependencies

The successful implementation and operation of SAMS rely on several assumptions and external dependencies:

Assumptions

- **User Access**: All users (students, faculty, administrators) have access to computers or devices with modern web browsers and stable internet connections
- **User Accounts**: The institution will provide initial user data (names, email addresses, IDs) for account creation
- **Payment Methods**: Students have access to credit cards or bank accounts for tuition payment
- **Course Data**: The institution has defined course catalogs, prerequisites, and degree requirements that can be loaded into the system
- **Academic Calendar**: A clear academic calendar with registration periods, semester dates, and payment deadlines is established
- **Administrator Availability**: System administrators will be available to validate payments and grant course access in a timely manner during registration periods

- **Training**: Basic user training or orientation will be provided to all user groups before system launch

Dependencies

- **Technology Stack Components**:
  - Vue.js framework availability and stability
  - Spring Boot framework and its ecosystem libraries
  - PostgreSQL database system
  - Spring Security authentication/authorization framework
  - Compatible hosting environment for deployment
- **Browser Support**:
  - Continued support and updates for Chrome, Firefox, Edge browsers...
  - JavaScript enabled in user browsers
  - Cookie support for session management
- **Network Infrastructure**:
  - Reliable internet service provider
  - Adequate bandwidth for concurrent users
- **Future Enhancements** (if implemented):
  - Firebase availability for real-time messaging features
  - WebSocket support if implementing live chat
  - Document storage service for file upload capabilities

3 Specific Requirements

3.1 Functional Requirements

This section details all functional requirements for SAMS, organized by feature area and user role.

3.1.1 User Authentication and Authorization

FR-1.1: User Login

- The system shall provide a secure login interface requiring username and password
- The system shall authenticate users against stored credentials in the database
- The system shall implement Spring Security for authentication management
- The system shall create a secure session upon successful authentication
- The system shall display appropriate error messages for invalid credentials

FR-1.2: Role-Based Access Control

- The system shall support three distinct user roles: Student, Faculty, and Administrator
- The system shall redirect users to their respective portals after successful login
- The system shall restrict access to features based on user roles
- The system shall deny access to unauthorized functionality and display appropriate error messages

FR-1.3: Session Management

- The system shall maintain user sessions throughout their interaction with the system
- The system shall provide a logout function that terminates the user session
- The system shall automatically log out users after a period of inactivity for security

3.1.2 Student Portal Requirements

FR-2.1: Academic Dashboard

- The system shall display a personalized dashboard showing current enrolled courses
- The system shall display current grades for all courses on the dashboard
- The system shall show upcoming deadlines and important dates
- The system shall provide quick access to frequently used features

FR-2.2: Course Registration

- The system shall provide a course search function with filters (department, time, instructor)
- The system shall display course details including schedule, capacity, instructor, and prerequisites
- The system shall verify prerequisite completion before allowing course registration
- The system shall detect and prevent time conflicts when registering for courses
- The system shall add students to a waitlist when courses are at capacity
- The system shall allow students to drop courses before add/drop deadlines
- The system shall update available seats in real-time as students register or drop

FR-2.3: Class Schedule Management

- The system shall display the student's current semester schedule in calendar format
- The system shall show course meeting times, locations, and instructor information
- The system shall allow students to view schedules for upcoming semesters
- The system shall provide options to export or print schedules

FR-2.4: Degree Progress Tracking

- The system shall display total credits completed out of required credits for graduation
- The system shall show courses completed in each requirement category
- The system shall identify remaining requirements needed for graduation
- The system shall calculate and display projected graduation date
- The system shall alert students about missing prerequisites or unfulfilled requirements

FR-2.5: GPA and Transcript Access

- The system shall automatically calculate semester GPA when grades are posted
- The system shall automatically calculate cumulative GPA across all semesters
- The system shall display grade history for all completed courses
- The system shall generate official transcripts showing all courses, grades, and GPA statistics
- The system shall update GPA calculations immediately when grades are entered or modified

FR-2.6: Financial Information Management

- The system shall display current account balance and tuition charges
- The system shall show individual course fees for enrolled courses
- The system shall calculate total payment due for the semester
- The system shall display financial aid awards and their application to charges
- The system shall show payment history and transaction records
- The system shall display payment due dates prominently

FR-2.7: Payment Processing

- The system shall accept payment information for credit card and bank transfer methods
- The system shall require payment of total semester fees before granting course access
- The system shall prevent finalization of registration until payment is completed
- The system shall submit payment information to administrators for validation
- The system shall notify students when payment has been validated

FR-2.8: Study Group Collaboration

- The system shall allow students to create study groups for specific courses
- The system shall allow students to join existing study groups

- The system shall display study group member lists with contact information
- The system shall provide messaging capabilities within study groups
- The system shall allow group creators to invite other students to the group

FR-2.9: Notification System (if timeline permits)

- The system shall send notifications 24 hours before assignment deadlines
- The system shall alert students when grades are posted or updated
- The system shall send reminders when course registration periods open
- The system shall allow students to view notification history
- The system shall provide notification preferences for different alert types

3.1.3 Faculty Portal Requirements

FR-3.1: Course Offering Creation

- The system shall allow faculty to create new course offerings for upcoming semesters
- The system shall require specification of course schedule (days, times)
- The system shall require setting of maximum enrollment capacity
- The system shall allow definition of course prerequisites
- The system shall allow faculty to modify course details before registration opens

FR-3.2: Course Roster Management

- The system shall display enrolled student lists for each course
- The system shall show waitlisted students when courses are at capacity
- The system shall allow faculty to view student enrollment status
- The system shall update rosters in real-time as students register or drop
- The system shall display total enrollment count versus capacity

FR-3.3: Course Material Management

- The system shall allow faculty to post course syllabi
- The system shall allow faculty to post announcements to course students
- The system shall allow faculty to upload course materials as text content
- The system shall display posted materials to enrolled students only
- The system shall allow faculty to edit or remove previously posted materials

FR-3.4: Grade Entry and Management

- The system shall provide interfaces for entering assignment grades
- The system shall provide interfaces for entering exam grades
- The system shall allow faculty to enter final course grades

- The system shall allow faculty to update or modify grades when necessary
- The system shall validate grade entries to ensure they are within acceptable ranges
- The system shall trigger GPA recalculation immediately upon grade submission

FR-3.5: Final Grade Submission

- The system shall require faculty to submit final grades at semester end
- The system shall enforce grade submission deadlines
- The system shall confirm successful grade submission to faculty
- The system shall prevent modification of final grades after submission deadline without administrator approval

FR-3.6: Student Profile Access

- The system shall allow faculty to view enrolled student profiles
- The system shall display student academic histories to faculty
- The system shall show student contact information for enrolled courses
- The system shall maintain student privacy by restricting profile access to relevant courses

FR-3.7: Communication Tools

- The system shall provide messaging capabilities for faculty to contact students
- The system shall allow faculty to send messages to individual students
- The system shall allow faculty to send announcements to entire course rosters
- The system shall maintain message history for reference

FR-3.8: Attendance Tracking

- The system shall allow faculty to record student attendance for course sessions
- The system shall display attendance records and statistics
- The system shall allow faculty to track participation metrics
- The system shall provide attendance reports for each course

3.1.4 Administrator Portal Requirements

FR-4.1: User Account Management

- The system shall allow administrators to create student accounts
- The system shall allow administrators to create faculty accounts
- The system shall allow administrators to create staff accounts
- The system shall require unique usernames for all accounts
- The system shall generate or allow setting of initial passwords

- The system shall allow administrators to edit user account information
- The system shall allow administrators to deactivate or delete accounts

FR-4.2: Role and Permission Assignment

- The system shall allow administrators to assign user roles (Student, Faculty, Administrator)
- The system shall allow administrators to modify user roles
- The system shall allow administrators to set specific permissions for user accounts
- The system shall enforce assigned permissions throughout the system

FR-4.3: Security Settings Management

- The system shall allow administrators to configure password policies
- The system shall allow administrators to manage session timeout settings
- The system shall allow administrators to configure access control rules
- The system shall provide security audit logs for administrator review

FR-4.4: Payment Validation

- The system shall display pending payment submissions from students
- The system shall show payment details (amount, method, student information)
- The system shall allow administrators to validate credit card payments
- The system shall allow administrators to validate bank transfer payments
- The system shall record payment validation timestamp and administrator ID
- The system shall prevent duplicate payment validation

FR-4.5: Course Access Management

- The system shall automatically grant course access upon payment validation
- The system shall allow administrators to manually grant course access in special circumstances
- The system shall allow administrators to revoke course access if necessary
- The system shall notify students when course access is granted
- The system shall prevent access to course materials until payment is validated

FR-4.6: Student Information Oversight

- The system shall allow administrators to view all student profile information
- The system shall allow administrators to view student academic records
- The system shall allow administrators to view student enrollment history
- The system shall allow administrators to view student financial information
- The system shall maintain audit logs of administrator access to student information

FR-4.7: Faculty Information Oversight

- The system shall allow administrators to view all faculty profile information
- The system shall allow administrators to view faculty course assignments
- The system shall allow administrators to view faculty workload and schedules
- The system shall maintain audit logs of administrator access to faculty information

FR-4.8: System Configuration

- The system shall allow administrators to configure system-wide settings
- The system shall allow administrators to set academic calendar dates
- The system shall allow administrators to configure registration periods
- The system shall allow administrators to set payment deadlines
- The system shall allow administrators to configure notification settings

3.1.5 Cross-Functional Workflows

FR-5.1: Course Registration Workflow

- Faculty creates course offerings for the semester with schedules, capacity, and prerequisites
- Students search and register for courses during the registration period
- System automatically manages waitlists and enforces enrollment limits
- Students complete payment for all enrolled courses
- Administrators validate payments and grant course access

FR-5.2: Grade Submission Workflow

- Faculty enters grades for assignments and exams throughout the semester
- System automatically calculates and updates student GPAs in real-time
- Students receive notifications when new grades are posted
- Faculty submits final grades at semester end
- System updates transcripts and degree progress tracking

FR-5.3: Payment and Access Workflow

- Students enroll in courses for the upcoming semester
- System calculates total fees based on enrolled courses
- Students submit payment via credit card or bank transfer before semester start
- Payment information is queued for administrator validation
- Administrators review and validate payment submissions
- System grants course access upon payment validation

- Students receive notification of course access approval
- Students can access course materials published by faculty

3.2 Non-Functional Requirements

This section specifies the non-functional requirements that define system quality attributes and operational characteristics.

3.2.1 Performance Requirements

NFR-1.1: Response Time

- The system shall load page content within 3 seconds under normal network conditions
- The system shall complete course registration transactions within 5 seconds
- The system shall display search results within 2 seconds
- The system shall update GPA calculations within 1 second of grade entry

NFR-1.2: Throughput

- The system shall support up to 100 concurrent users during peak registration periods
- The system shall process at least 50 course registrations per minute
- The system shall handle multiple simultaneous grade entries without performance degradation

NFR-1.3: Capacity

- The system shall support storage for up to 100 student records initially
- The system shall support unlimited course offerings across multiple semesters
- The system shall maintain complete academic history for all students

NFR-1.4: Scalability

- The system architecture shall accommodate future growth beyond 100 students
- The database design shall support efficient queries as data volume grows

3.2.2 Security Requirements

NFR-2.1: Authentication

- The system shall implement secure password storage using encryption (if time permits)
- The system shall enforce password complexity requirements (minimum 8 characters, combination of letters and numbers)
- The system shall lock user accounts after 5 consecutive failed login attempts (if time permits)

- The system shall use Spring Security framework for authentication management

NFR-2.2: Authorization

- The system shall enforce role-based access control for all features
- The system shall verify user permissions before granting access to sensitive data
- The system shall prevent privilege escalation attempts (if time permits)
- The system shall log all authorization failures for security purpose

NFR-2.3: Data Protection

- The system shall encrypt all communication between client and server using HTTPS/TLS (if time permits)
- The system shall protect student academic records from unauthorized access
- The system shall protect financial information with appropriate security measures
- The system shall mask sensitive payment information in user interfaces

NFR-2.4: Session Security

- The system shall use secure session tokens
- The system shall invalidate sessions upon logout
- The system shall prevent session hijacking through secure cookie implementation

NFR-2.5: Audit Trail

- The system shall log all payment validation activities
- The system shall log all grade modifications with timestamp and user ID
- The system shall log administrative access to student and faculty information
- The system shall maintain logs for security analysis and compliance

3.2.3 Reliability Requirements

NFR-3.1: Availability

- The system shall maintain 99% uptime during academic semesters
- The system shall be available 24/7 except for scheduled maintenance windows
- Scheduled maintenance shall occur during low-usage periods with advance notice

NFR-3.2: Fault Tolerance

- The system shall handle database connection failures gracefully with appropriate error messages
- The system shall implement transaction rollback for failed operations
- The system shall prevent data corruption in case of system failures

NFR-3.3: Data Backup

- The system shall perform daily automated backups of the PostgreSQL database
- The system shall retain backup data for at least 30 days
- The system shall provide data recovery procedures for disaster scenarios

NFR-3.4: Error Handling

- The system shall display user-friendly error messages without exposing technical details
- The system shall log all errors with sufficient detail for troubleshooting
- The system shall recover gracefully from errors without data loss

3.2.4 Usability Requirements

NFR-4.1: User Interface

- The system shall provide an intuitive, easy-to-navigate interface for users with basic computer skills
- The system shall use consistent navigation patterns across all portals
- The system shall provide clear visual feedback for user actions
- The system shall use descriptive labels and instructions for all input fields

NFR-4.2: Learnability

- New students shall be able to complete course registration within 10 minutes after brief orientation
- Faculty members shall be able to enter grades without formal training
- The system shall provide contextual help and tooltips where appropriate

NFR-4.3: Accessibility

- The system shall comply with basic web accessibility standards
- The system shall support keyboard navigation for all essential functions
- The system shall use sufficient color contrast for readability
- The system shall provide text alternatives for non-text content

NFR-4.4: Responsiveness

- The system shall provide a responsive design that adapts to different screen sizes
- The system shall function correctly on tablets and mobile devices
- The system shall maintain usability across Chrome, Firefox, Safari, and Edge browsers

3.2.5 Maintainability Requirements

NFR-5.1: Code Quality

- The system shall follow Spring Boot and Vue.js best practices and coding standards

- The system code shall include comments explaining complex logic
- The system shall use meaningful variable and function names

NFR-5.2: Documentation

- The system shall include API documentation for all REST endpoints
- The system shall provide database schema documentation
- The system shall maintain user manuals for each portal type

NFR-5.3: Testability

- The system shall implement unit tests for critical business logic
- The system shall support automated testing of REST APIs
- The system shall allow independent testing of frontend and backend layers
- The system shall maintain test coverage for core functionalities

NFR-5.4: Modularity

- The system shall implement clear separation between Vue.js frontend and Spring Boot backend
- The system shall use REST API as the integration point between layers
- The system shall design database schema with normalized tables

3.2.6 Portability Requirements

NFR-6.1: Browser Compatibility

- The system shall function correctly on Chrome Firefox Edge

NFR-6.2: Platform Independence

- The system shall run on any platform supporting Java Runtime Environment (for Spring Boot)
- The system shall be deployable on standard web hosting environments
- The system shall be compatible with standard PostgreSQL installations

3.3 Use Case Diagrams and Narratives

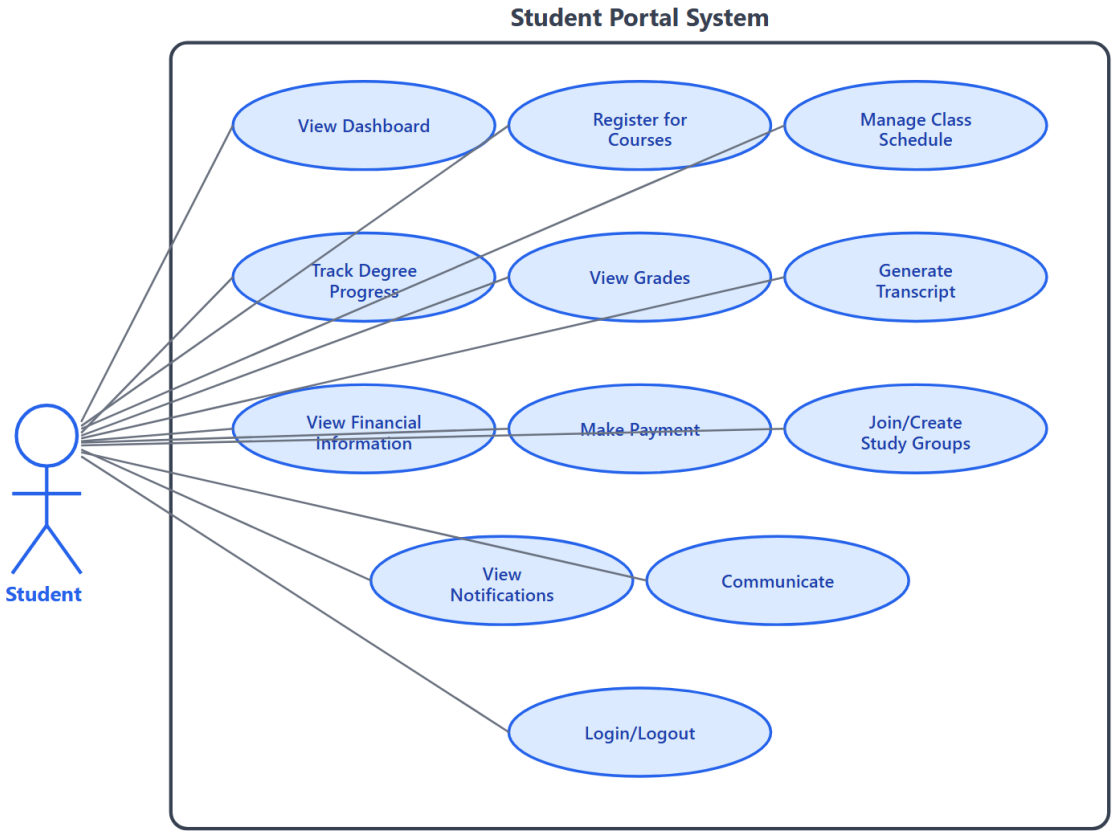

Figure 1: Student Portal Use Case Diagram

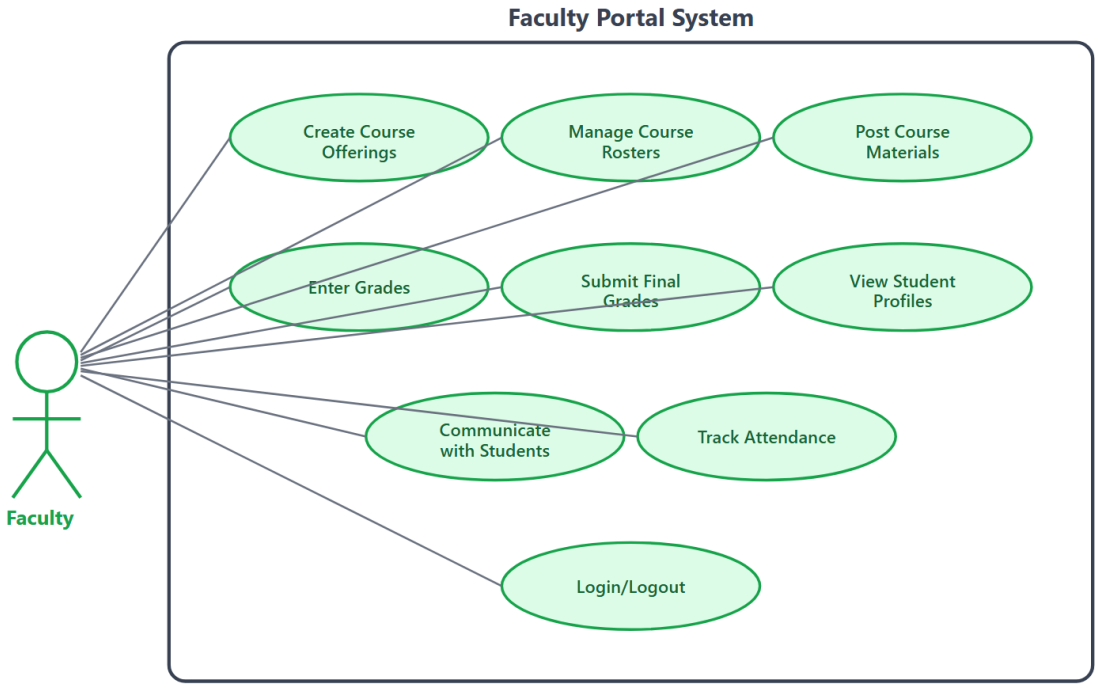

Figure 2: Faculty Portal Use Case Diagram

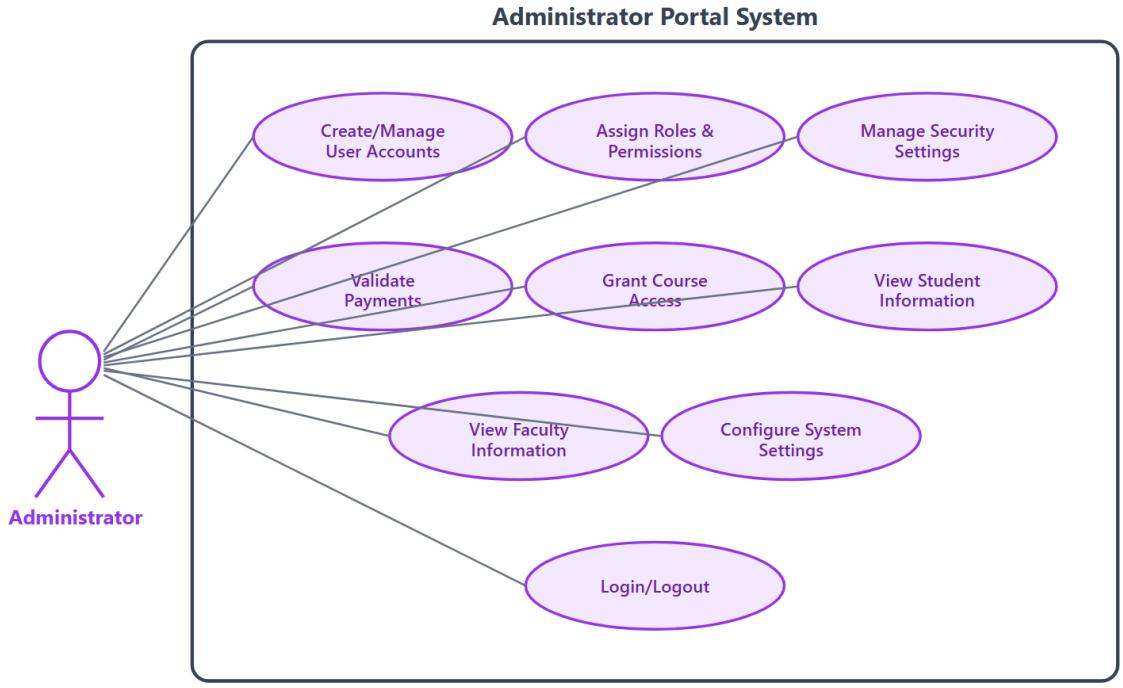

Figure 3: Administrator Portal Use Case Diagram

4 Appendices

Appendix: Technology Stack Details

Frontend:

- Vue.js framework for building interactive user interfaces
- Modern JavaScript (ES6+)
- HTML5 and CSS3
- Responsive design implementation

Backend:

- Spring Boot framework
- Spring Security for authentication and authorization
- REST API architecture using Spring Boot Controllers
- Java programming language

Database:

- PostgreSQL relational database management system
- SQL for data manipulation

Development Approach:

- Backend-first development methodology

- Layer-by-layer validation before integration
- RESTful API as integration contract