**Software Requirements Specification (SRS)**

**Academic Module - ERP System**

**Date: August 2025**

**1. Introduction**

**1.1 Purpose**

This document specifies the requirements for the Academic Module of the Educational ERP system, focusing on course management, student performance tracking, examination scheduling, and academic analytics.

**1.2 Scope**

The Academic Module manages all educational activities including:

* Course and program curriculum management
* Student enrollment and academic records
* Grade management and transcript generation
* Attendance tracking (online and onsite)
* Examination scheduling and results
* Academic performance analytics
* Learning resources management

**2. Functional Requirements**

**2.1 Course and Program Management**

**Priority**: High

**Requirements:**

* **FR-A1.1**: System shall allow creation, modification, and deletion of academic programs
* **FR-A1.2**: System shall manage course catalogs with prerequisites
* **FR-A1.3**: System shall support course scheduling with room assignments
* **FR-A1.4**: System shall handle course capacity and waitlist management
* **FR-A1.5**: System shall track course versions and curriculum changes
* **FR-A1.6**: System shall support multi-semester course planning
* **FR-A1.7**: System shall manage academic calendar and important dates
* **FR-A1.8**: System shall support course equivalencies and transfers

**2.2 Student Performance Tracking**

**Priority**: High

**Requirements:**

* **FR-A2.1**: System shall record and calculate grades using various grading systems
* **FR-A2.2**: System shall track attendance for both online and onsite classes
* **FR-A2.3**: System shall generate real-time GPA calculations
* **FR-A2.4**: System shall support weighted grade categories
* **FR-A2.5**: System shall track student progress against degree requirements
* **FR-A2.6**: System shall identify at-risk students using predictive analytics
* **FR-A2.7**: System shall maintain complete academic history
* **FR-A2.8**: System shall support grade appeals and corrections

**2.3 Examination Management**

**Priority**: High

**Requirements:**

* **FR-A3.1**: System shall schedule examinations with conflict detection
* **FR-A3.2**: System shall manage exam venues and seating arrangements
* **FR-A3.3**: System shall support multiple exam formats (written, oral, practical)
* **FR-A3.4**: System shall handle make-up exam scheduling
* **FR-A3.5**: System shall publish results with security controls
* **FR-A3.6**: System shall generate examination reports and statistics
* **FR-A3.7**: System shall support online examination capabilities
* **FR-A3.8**: System shall manage proctoring assignments

**2.4 Academic Analytics and Reporting**

**Priority**: Medium

**Requirements:**

* **FR-A4.1**: System shall generate individual student transcripts
* **FR-A4.2**: System shall produce attendance summary reports
* **FR-A4.3**: System shall create course performance analytics
* **FR-A4.4**: System shall generate graduation tracking reports
* **FR-A4.5**: System shall provide enrollment statistics and trends
* **FR-A4.6**: System shall create faculty workload reports
* **FR-A4.7**: System shall support custom report generation
* **FR-A4.8**: System shall provide visual dashboards with charts and graphs

**2.5 Learning Management Integration**

**Priority**: Medium

**Requirements:**

* **FR-A5.1**: System shall integrate with Learning Management Systems
* **FR-A5.2**: System shall manage digital course materials
* **FR-A5.3**: System shall track assignment submissions and deadlines
* **FR-A5.4**: System shall support online discussion forums
* **FR-A5.5**: System shall manage digital library resources
* **FR-A5.6**: System shall support video conferencing integration
* **FR-A5.7**: System shall track student engagement metrics
* **FR-A5.8**: System shall support plagiarism detection

**3. Use Case Diagram - Academic Module**

Academic Module Use Cases

Academic Administrator -----> Create/Modify Programs

-----> Schedule Classes

-----> Manage Academic Calendar

-----> Generate Reports

-----> Track Enrollment

Faculty/Instructor ---------> Manage Courses

-----> Record Attendance

-----> Enter Grades

-----> Schedule Exams

-----> Upload Course Materials

-----> View Student Progress

Student --------------------> Enroll in Courses

-----> View Grades

-----> Check Attendance

-----> Access Course Materials

-----> View Schedule

-----> Request Transcripts

-----> Submit Assignments

Registrar ------------------> Manage Student Records

-----> Generate Transcripts

-----> Handle Transfers

-----> Verify Graduation Requirements

-----> Maintain Academic Policies

System Administrator -------> Configure Grading Systems

-----> Set Academic Parameters

-----> Manage User Permissions

-----> Monitor System Performance

**4. Class Diagram - Academic Module**

Student

├── studentId: String

├── firstName: String

├── lastName: String

├── email: String

├── dateOfBirth: Date

├── enrollmentDate: Date

├── studentStatus: StudentStatus

├── currentGPA: Double

├── totalCredits: Integer

└── methods: calculateGPA(), checkGraduation(), getTranscript()

Course

├── courseId: String

├── courseCode: String

├── courseName: String

├── description: Text

├── credits: Integer

├── prerequisites: List<Course>

├── department: Department

├── isActive: Boolean

└── methods: addPrerequisite(), checkCapacity()

Enrollment

├── enrollmentId: String

├── studentId: String (FK)

├── sectionId: String (FK)

├── enrollmentDate: Date

├── status: EnrollmentStatus

├── finalGrade: String

├── gradePoints: Double

└── methods: calculateGrade(), checkAttendance()

Section

├── sectionId: String

├── courseId: String (FK)

├── instructorId: String (FK)

├── semester: String

├── year: Integer

├── schedule: Schedule

├── capacity: Integer

├── enrolledCount: Integer

├── venue: String

└── methods: checkAvailability(), addStudent()

Grade

├── gradeId: String

├── enrollmentId: String (FK)

├── assignmentType: GradeType

├── points: Double

├── maxPoints: Double

├── weight: Double

├── dateRecorded: Date

└── methods: calculatePercentage()

Attendance

├── attendanceId: String

├── enrollmentId: String (FK)

├── date: Date

├── status: AttendanceStatus

├── classType: ClassType (Online/Onsite)

├── duration: Integer

└── methods: markAttendance()

Exam

├── examId: String

├── sectionId: String (FK)

├── examType: ExamType

├── scheduledDate: Date

├── duration: Integer

├── venue: String

├── maxMarks: Double

└── methods: scheduleExam(), publishResults()

Program

├── programId: String

├── programName: String

├── degree: DegreeType

├── department: Department

├── totalCredits: Integer

├── duration: Integer

├── requirements: List<CourseRequirement>

└── methods: checkCompletion(), getRequiredCourses()

Instructor

├── instructorId: String

├── firstName: String

├── lastName: String

├── email: String

├── department: Department

├── qualifications: String

├── hireDate: Date

└── methods: assignSection(), recordGrade()

**5. Sequence Diagram - Grade Entry Process**

Instructor -> GradeEntryForm: Access grade entry

GradeEntryForm -> SectionService: getSectionStudents(sectionId)

SectionService -> Database: SELECT enrollments

Database -> SectionService: Student list

SectionService -> GradeEntryForm: Display students

Instructor -> GradeEntryForm: Enter grades

GradeEntryForm -> GradeService: validateGrades()

GradeService -> GradeEntryForm: Validation result

GradeEntryForm -> GradeService: saveGrades()

GradeService -> Database: INSERT/UPDATE grades

Database -> GradeService: Confirmation

GradeService -> GPAService: recalculateGPA()

GPAService -> Database: UPDATE student GPA

GradeService -> NotificationService: sendGradeNotification()

NotificationService -> Student: Grade notification

GradeService -> GradeEntryForm: Success confirmation

**6. Sequence Diagram - Course Enrollment Process**

Student -> EnrollmentPortal: Search courses

EnrollmentPortal -> CourseService: getAvailableCourses()

CourseService -> Database: SELECT available sections

Database -> CourseService: Course list

CourseService -> EnrollmentPortal: Display courses

Student -> EnrollmentPortal: Select course

EnrollmentPortal -> EnrollmentService: checkPrerequisites()

EnrollmentService -> Database: Verify prerequisites

Database -> EnrollmentService: Prerequisite status

EnrollmentService -> EnrollmentPortal: Prerequisite check result

Student -> EnrollmentPortal: Confirm enrollment

EnrollmentPortal -> EnrollmentService: enrollStudent()

EnrollmentService -> Database: INSERT enrollment

Database -> EnrollmentService: Enrollment confirmed

EnrollmentService -> NotificationService: sendEnrollmentConfirmation()

NotificationService -> Student: Enrollment confirmation

NotificationService -> Instructor: New student notification

**7. Activity Diagram - Attendance Tracking Process**

Start

↓

Instructor Opens Attendance Module

↓

Select Course Section

↓

Choose Class Date

↓

System Displays Student List

↓

For Each Student:

↓

Mark Attendance Status

(Present/Absent/Late/Excused)

↓

[Online Class?] → Yes → Record Session Duration

↓

Record IP Address

↓

[Onsite Class?] → Yes → Record Physical Presence

↓

Save Attendance Data

↓

Calculate Attendance Percentage

↓

Check Minimum Attendance Requirement

↓

[Below Threshold?] → Yes → Generate Warning Alert

↓

Send Notification to Student

↓

Notify Academic Advisor

↓

Update Student Dashboard

↓

Generate Attendance Report

↓

End

**8. Activity Diagram - Exam Scheduling Process**

Start

↓

Academic Admin Initiates Exam Scheduling

↓

Select Academic Period

↓

Retrieve Course Sections

↓

For Each Course Section:

↓

Check Instructor Availability

↓

Check Venue Availability

↓

Check Student Schedule Conflicts

↓

[Conflicts Found?] → Yes → Find Alternative Slot

↓

Update Schedule

↓

Assign Exam Slot

↓

Assign Venue

↓

Assign Proctors

↓

Validate Complete Schedule

↓

[Schedule Valid?] → No → Resolve Conflicts

↓

Return to Scheduling

↓

Publish Exam Schedule

↓

Send Notifications to Students

↓

Send Notifications to Instructors

↓

Send Notifications to Proctors

↓

Generate Exam Reports

↓

End

**9. Non-Functional Requirements**

**9.1 Performance Requirements**

* **NFR-A1**: Grade calculations shall complete within 1 second
* **NFR-A2**: Attendance recording shall support 200 concurrent users
* **NFR-A3**: Report generation shall complete within 30 seconds
* **NFR-A4**: Course search shall return results within 2 seconds

**9.2 Security Requirements**

* **NFR-A5**: Grade data shall be encrypted at rest and in transit
* **NFR-A6**: Access to student records shall be role-based and audited
* **NFR-A7**: Exam schedules shall be protected from unauthorized modifications
* **NFR-A8**: Academic records shall maintain complete audit trails

**9.3 Usability Requirements**

* **NFR-A9**: Grade entry interface shall support bulk operations
* **NFR-A10**: Attendance marking shall be mobile-friendly
* **NFR-A11**: Reports shall be exportable in multiple formats (PDF, Excel, CSV)
* **NFR-A12**: Dashboard shall provide real-time academic metrics

**10. Database Design - Key Tables**

**10.1 Students Table**

CREATE TABLE students (

student\_id VARCHAR(20) PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

date\_of\_birth DATE NOT NULL,

enrollment\_date DATE NOT NULL,

student\_status ENUM('Active', 'Inactive', 'Graduated', 'Suspended'),

current\_gpa DECIMAL(3,2),

total\_credits INTEGER DEFAULT 0,

program\_id VARCHAR(20),

FOREIGN KEY (program\_id) REFERENCES programs(program\_id)

);

**10.2 Courses Table**

CREATE TABLE courses (

course\_id VARCHAR(20) PRIMARY KEY,

course\_code VARCHAR(10) UNIQUE NOT NULL,

course\_name VARCHAR(200) NOT NULL,

description TEXT,

credits INTEGER NOT NULL,

department VARCHAR(50),

is\_active BOOLEAN DEFAULT TRUE

);

**10.3 Enrollments Table**

CREATE TABLE enrollments (

enrollment\_id VARCHAR(20) PRIMARY KEY,

student\_id VARCHAR(20) NOT NULL,

section\_id VARCHAR(20) NOT NULL,

enrollment\_date DATE NOT NULL,

status ENUM('Enrolled', 'Withdrawn', 'Completed'),

final\_grade VARCHAR(5),

grade\_points DECIMAL(3,2),

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (section\_id) REFERENCES sections(section\_id)

);

**11. Integration Points**

**11.1 Internal Integrations**

* Finance Module: Student fee status for enrollment eligibility
* HR Module: Instructor information and assignments
* Main System: User authentication and notifications

**11.2 External Integrations**

* Learning Management Systems (Moodle, Canvas, Blackboard)
* Student Information Systems
* Academic plagiarism detection services
* Video conferencing platforms (Zoom, Teams)
* Digital library systems

**12. Acceptance Criteria**

**12.1 Course Management**

* Courses can be created, modified, and deactivated successfully
* Prerequisites are enforced during enrollment
* Course schedules display without conflicts
* Capacity management prevents over-enrollment

**12.2 Grade Management**

* Grades are calculated accurately using defined weights
* GPA calculations are real-time and accurate
* Transcripts generate with complete academic history
* Grade changes maintain audit trail

**12.3 Attendance Tracking**

* Attendance can be marked for both online and onsite classes
* Attendance percentages calculate correctly
* Low attendance triggers appropriate warnings
* Attendance reports are accurate and timely

**12.4 Examination System**

* Exam schedules generate without conflicts
* Results are published securely to authorized users
* Exam statistics and analytics are accurate
* Make-up exams are scheduled appropriately