Software Requirements Specification (SRS)

Student Management System

Document Version: 1.0

Date: December 2025

---

1. Introduction

1.1 Purpose

This document specifies the requirements for a console-based Student Management System implemented in Java using Object-Oriented Programming principles.

1.2 Scope

The Student Management System is designed to manage student records, course assignments, and grade calculations through a command-line interface with administrative controls.

1.3 Definitions and Acronyms

- **CRUD**: Create, Read, Update, Delete
- **OOP**: Object-Oriented Programming
- **GPA**: Grade Point Average
- **CLI**: Command Line Interface
- **SDLC**: Software Development Life Cycle

---

2. Overall Description

2.1 Product Perspective

The system is a standalone console application that enables administrators to manage student information, course enrollments, and academic performance tracking.

2.2 Product Functions

- Student record management (CRUD operations)
- Unique ID generation for students
- Course assignment and management
- Automatic GPA calculation
- Search functionality
- Statistical reporting
- Admin authentication

2.3 User Classes

- **Administrator**: Primary user with full access to all system functions

2.4 Operating Environment

- Java Runtime Environment (JRE) 8 or higher
- Console/Terminal interface
- Cross-platform (Windows, Linux, macOS)

---

3. Functional Requirements

3.1 User Authentication

REQ-001: Admin Login

- The system shall provide a login interface requiring username and password
- Default credentials: username="admin", password="admin123"
- Maximum of 3 login attempts allowed
- System shall lock after failed attempts

3.2 Student Management

REQ-002: Create Student

- System shall generate unique student IDs automatically (format: STU1001, STU1002, etc.)
- Required fields: First Name, Last Name, Email, Age
- System shall validate all inputs before creation

REQ-003: Read Student Information

- System shall display all student records
- System shall display individual student details
- System shall show student ID, name, email, age, GPA, and enrolled courses

REQ-004: Update Student Information

- System shall allow selective field updates (first name, last name, email, age)
- System shall validate updated information
- System shall preserve student ID (immutable)

REQ-005: Delete Student

- System shall allow deletion by student ID
- System shall request confirmation before deletion
- System shall remove all associated data

3.3 Course Management

REQ-006: Assign Course

- System shall allow course assignment to students
- Required fields: Course Code, Course Name, Credits, Grade

- Course codes must follow format: 2-4 letters + 3 digits (e.g., CS101)
- Grades must be between 0-100

REQ-007: Remove Course

- System shall allow course removal from students
- System shall recalculate GPA after removal

REQ-008: Automatic GPA Calculation

- System shall calculate GPA automatically on a 4.0 scale
- Grading scale:
  - A (90-100): 4.0
  - B (80-89): 3.0
  - C (70-79): 2.0
  - D (60-69): 1.0
  - F (0-59): 0.0
- GPA = Sum(Grade Points × Credits) / Total Credits

3.4 Search and Query

REQ-009: Search Functionality

- System shall support search by student ID, name, or email
- Search shall be case-insensitive
- System shall return all matching records

REQ-010: Statistics

- System shall display total number of students
- System shall calculate and display average GPA
- System shall list top-performing students (GPA >= 3.0)

---

4. Non-Functional Requirements

4.1 Performance

- System shall respond to user inputs within 1 second
- System shall handle up to 1000 student records efficiently

4.2 Security

- Passwords shall be stored in plain text (demonstration purposes only)
- System shall validate all user inputs
- System shall prevent SQL injection and invalid data entry

4.3 Usability

- System shall provide clear menu navigation
- System shall display informative error messages

- System shall validate all inputs with appropriate feedback

4.4 Reliability

- System shall handle exceptions gracefully using try-catch blocks
- System shall not crash on invalid inputs
- System shall maintain data consistency

4.5 Maintainability

- Code shall follow OOP principles
- Classes shall be properly encapsulated
- System shall use appropriate design patterns

---

5. Data Requirements

5.1 Student Entity

- Student ID (String, unique, auto-generated)
- First Name (String, required, alphabetic)
- Last Name (String, required, alphabetic)
- Email (String, required, valid format)
- Age (Integer, range: 1-149)
- GPA (Double, calculated, range: 0.0-4.0)
- Courses (List of Course objects)

5.2 Course Entity

- Course Code (String, required, format: XX###)
- Course Name (String, required)
- Credits (Integer, range: 1-10)
- Grade (Double, range: 0-100)

---

6. System Constraints

6.1 Technical Constraints

- Must be implemented in Java
- Must use console-based interface (Scanner class)
- Must implement OOP principles
- Must use in-memory storage (no database required)

6.2 Business Constraints

- System is for educational demonstration purposes
- Single-user (admin) access only
- Data is not persisted between sessions

---

7. Input Validation Rules

7.1 Name Validation

- Must contain only letters and spaces
- Must not be empty

7.2 Email Validation

- Must follow format: username@domain.extension
- Must contain @ symbol and valid domain

7.3 Age Validation

- Must be numeric
- Must be between 1 and 149

7.4 Course Code Validation

- Must be 2-4 uppercase letters followed by 3 digits
- Example: CS101, MATH201

7.5 Grade Validation

- Must be numeric
- Must be between 0 and 100

7.6 Credits Validation

- Must be numeric
- Must be between 1 and 10

---

8. Error Handling

8.1 Input Errors

- System shall catch NumberFormatException for numeric inputs
- System shall catch IllegalArgumentException for invalid data
- System shall display user-friendly error messages

8.2 Not Found Errors

- System shall display "Student not found" for invalid IDs
- System shall handle empty search results gracefully

8.3 System Errors

- System shall catch general exceptions
- System shall log errors with descriptive messages
- System shall allow user to retry operations

9. User Interface Requirements

9.1 Menu System

- Clear menu options numbered 0-9
- Input prompt for user selection
- Return to main menu after each operation

9.2 Display Format

- Tabular format for student lists
- Detailed view for individual students
- ASCII art borders for visual appeal

9.3 Feedback Messages

- Success messages with ✓ symbol
- Error messages with ✗ symbol
- Confirmation prompts for destructive operations

---

10. Acceptance Criteria

1. System successfully authenticates admin user
2. System creates students with unique IDs
3. System performs all CRUD operations correctly
4. System validates all inputs according to rules
5. System calculates GPA automatically and accurately
6. System handles errors without crashing
7. System provides clear navigation and feedback
8. Documentation is complete and accurate

---

11. Future Enhancements

- Database persistence
- Multiple user roles (student, teacher, admin)
- Report generation (PDF/Excel)
- Grade analytics and trends
- Attendance tracking
- Parent portal access

Document Prepared By: Development Team
Approved By: Project Manager
Review Date: December 2025