Software Requirements

Specification

Revision History

Date	**Revision**	**Description**	**Author**
09/16/2024	1.0	Initial Version	Chen Li
09/23/2024	1.0	Adding Specific Requirements, External and Internel Interface Requirements	Chen Li
09/25/2024	1.0	Adding Use Cases	Chen Li

Table of Contents

1. Purpose

This document outlines the software requirements for a University Class Enrollment

System. The system aims to streamline the process of course registration, manage class schedules, and facilitate communication between students, faculty, and administrators.

1.1. Scope

The system will handle course catalog management, student registration, faculty course assignments, and basic reporting. It will integrate with existing university systems for student records and billing.

1.2. Definitions, Acronyms, Abbreviations

UI	User Interface
API	Application Programming Interface
DBMS	Database Management System

1.3. References

Use Case Specification Document – Step 2 in assignment description

UML Use Case Diagrams Document – Step 3 in assignment description

Class Diagrams – Step 5 in assignment description

Sequence Diagrams – Step 6 in assignment description

1.4. Overview

The University Class Enrollment System is a Java-based application designed to manage the course enrollment process for a university. It provides functionality for students to register for courses, faculty to manage their classes, and administrators to oversee the entire process. The system operates without web technologies or databases, instead utilizing file-based storage for all data persistence.

2. Overall Description

2.1. Product Perspective

The University Class Enrollment System will be a standalone Java application accessible to students, faculty, and administrators on their local machines. It will use file-based storage for all data.

2.2. Product Architecture

The system will be organized into 6 major modules: the Course catalog management module, the Student course registration and scheduling module, and the Faculty course assignment module, the Waitlist management module, the Report generation module and User authentication and authorization module

2.3. Product Functionality/Features

- Course catalog management
- Student course registration and scheduling
- Faculty course assignment

- Waitlist management
- Report generation
- User authentication and authorization

2.4. Constraints

Technology Constraints:

- The system must be developed entirely in Java.
- No web technologies or databases can be used.
- The application must run on JVM 8 or higher.

Data Storage Constraints:

- All data must be stored in local files.
- File formats must be designed to ensure data integrity and easy parsing.

Network Constraints:

- The system must operate within a local network environment.
- No internet connectivity should be required for core functionalities.

User Interface Constraints:

- The GUI must be developed using Java Swing or JavaFX.
- The interface must be accessible to users with basic computer skills.

2.5. Assumptions and Dependencies

2.5.1 Assumptions:

User Assumptions:

- All users have basic computer literacy and can operate a Java application.
- Users have appropriate permissions to run Java applications on their machines.

System Assumptions:

- The local network is reliable and has sufficient bandwidth to handle file operations.
- All user computers meet the minimum hardware requirements to run the Java application smoothly.

Data Assumptions:

- The initial course catalog and user data will be provided in a compatible format.
- Regular backups of data files will be maintained outside the system.

Operational Assumptions:

- The university has a process in place to distribute and update the application as needed.
- There is a designated system administrator to manage file permissions and backups.

2.5.2 Dependencies

Java Runtime Environment:

- The system is dependent on the availability of JRE 8 or higher on all user machines.

File System:

- The application depends on read and write access to specific directories for data storage.

Network File Sharing:

- Proper configuration of network file sharing is required for multi-user functionality.

Printing Services:

- Integration with local or network printers is necessary for report generation.

External Tools:

- The system may depend on external Java libraries for specific functionalities (e.g., PDF generation for reports).

University Policies:

- The system's operations are dependent on adherence to university enrollment policies and academic calendar.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1 Users should be allowed to log in using their issued id and pin, both of which are alphanumeric strings between 6 and 20 characters in length.

3.1.1.2 The system should provide Java swing or Java FX based help pages on each screen that describe the purpose of each function within the system.

3.1.1.3 All modules should implement proper error handling and display user-friendly error messages.

3.1.1.4 Each module should have a consistent look and feel using Java Swing or JavaFX components.

3.1.1.5 All data modifications should be immediately saved to the appropriate files to ensure data integrity.

3.1.2. Course Catalog Management Module Requirements:

3.1.2.1 The system shall allow administrators to add new courses to the catalog.

3.1.2.2 Administrators should be able to edit existing course information, including course code, name, description, credits, and prerequisites.

3.1.2.3 The system shall provide a search function to find courses based on various criteria (e.g., course code, name, department).

3.1.2.4 Administrators should be able to set the maximum enrollment capacity for each course.

3.1.2.5 The system shall allow administrators to mark courses as active or inactive for the current semester.

3.1.3. Student Course Registration and Scheduling Module Requirements:

3.1.3.1 Students should be able to view the course catalog and filter courses based on various criteria.

3.1.3.2 The system shall allow students to add courses to their schedule, provided they meet the prerequisites and there are available slots.

3.1.3.3 Students should be able to drop courses from their schedule within the allowed add/drop period.

3.1.3.4 The system shall prevent schedule conflicts when students attempt to register for courses.

3.1.3.5 Students should be able to view their current schedule, including course times and locations.

3.1.4. Faculty Course Assignment Module Requirements:

3.1.4.1 Administrators should be able to assign faculty members to specific courses.

3.1.4.2 Faculty members should be able to view the courses they are assigned to teach.

3.1.4.3 The system shall allow faculty to access and download class rosters for their assigned courses.

3.1.4.4 Faculty should be able to set or modify course-specific details such as syllabus and course materials.

3.1.4.5 The system shall prevent the assignment of faculty to courses with conflicting schedules.

3.1.5. Waitlist Management Module Requirements:

3.1.5.1 The system shall automatically add students to a waitlist when they attempt to register for a full course.

3.1.5.2 Students should be able to view their position on waitlists for courses they've attempted to join.

3.1.5.3 The system shall automatically notify the next student on the waitlist when a spot becomes available in a course.

3.1.5.4 Administrators should be able to view and manage waitlists for all courses.

3.1.5.5 The system shall allow administrators to manually add or remove students from waitlists.

3.1.6. Report Generation Module Requirements:

3.1.6.1 The system shall generate enrollment reports showing the number of students registered for each course.

3.1.6.2 Administrators should be able to generate reports on course waitlists, including lengths and movement.

3.1.6.3 The system shall provide functionality to export reports in various formats (e.g., CSV, plain text).

3.1.6.4 Faculty should be able to generate and print class rosters for their assigned courses.

3.1.6.5 The system shall allow administrators to generate and view historical enrollment data and trends.

3.1.7. User Authentication and Authorization Module Requirements:

3.1.7.1 The system shall securely store user credentials, with passwords being hashed and salted.

3.1.7.2 Users should be able to reset their PIN through a secure process that involves verification of their identity.

3.1.7.3 The system shall implement role-based access control, restricting access to functionalities based on user type (student, faculty, administrator).

3.1.7.4 The system shall log all login attempts, both successful and failed, for security auditing purposes.

3.1.7.5 After a configurable number of failed login attempts, the system shall temporarily lock the user account and notify the administrator.

3.2. External Interface Requirements

3.2.1 The system must provide an interface to the University billing system administered by the Bursar's office so that students can be automatically billed for the courses in which they have enrolled. The interface is to be in a comma-separated text file containing the following fields: student id, course id, term id, action. Where "action" is whether the student has added or dropped the course. The file will be exported nightly and will contain new transactions only.

3.2.2 The system must provide an interface to the University's student information system to retrieve and update student demographic information. The interface will be a comma-separated text file containing the following fields: student id, name, address, phone number, email. This file will be exported weekly and will contain all current student records.

3.2.3 The system must provide an interface to the University's course catalog system to retrieve updated course information. The interface will be a comma-separated text file containing the following fields: course id, course name, department, credits, description. This file will be imported at the beginning of each term and will contain all active courses for the upcoming term.

3.2.4 The system must provide an interface to the University's faculty management system to retrieve current faculty information. The interface will be a comma-separated text file containing the following fields: faculty id, name, department, office location, contact information. This file will be imported monthly and will contain all current faculty records.

3.3. Internal Interface Requirements

3.3.1 The system must process a data-feed from the grading system such that student grades are stored along with the historical student course enrollments. Data feed will be in the form of a comma-separated interface file that is exported from the grading system nightly.

3.3.2 The system must process a data-feed from the University billing system that contains new student records. The feed will be in the form of a comma-separated text file and will be exported from the billing system nightly with new student records. The fields included in the file are student name, student id, and student pin number.

3.3.3 The system must maintain an internal interface between the Course Catalog Management module and the Student Course Registration module to ensure that students can only register for courses that are currently active and have available slots.

3.3.4 The system must maintain an internal interface between the Waitlist Management module and the Student Course Registration module to automatically enroll students from the waitlist when spots become available in a course.

3.3.5 The system must maintain an internal interface between the User Authentication and Authorization module and all other modules to ensure that users can only access functionalities appropriate to their role (student, faculty, or administrator).

3.3.6 The system must process an internal data-feed from the Student Course Registration module to the Report Generation module to provide up-to-date enrollment statistics for generating reports.

4. Non-Functional Requirements (Anthony)

4.1. Security and Privacy Requirements

Example:
4.1.1 The System must encrypt data being transmitted over the Internet.

4.2. Environmental Requirements

Example:
4.2.1 System cannot require that any software other than a web browser be installed on user computers.
4.2.2 System must make use of the University's existing Oracle 9i implementation for its database.
4.2.3 System must be deployed on existing Linux-based server infrastructure.

4.3. Performance Requirements

Example:

4.3.1 System must render all UI pages in no more than 9 seconds for dynamic pages. Static pages (HTML-only) must be rendered in less than 3 seconds.

5.Use Case Specification Document(Chen,Anthony, Connor, Andrew)

5.1 Student Course Registration

Use Case ID: UC-SCR-001 Use Case Name: Register for a Course Relevant Requirements: 3.1.3.1, 3.1.3.2, 3.1.3.4 Primary Actor: Student

Pre-conditions:

- Student is logged into the system
- Student has met all prerequisites for the desired course
- The course has available slots

Post-conditions:

- Student is enrolled in the course
- Course enrollment count is updated
- Student's schedule is updated

Basic Flow or Main Scenario:

1. Student navigates to the course registration page
2. Student searches for the desired course
3. System displays course details and availability
4. Student selects the course and clicks "Register"
5. System checks for schedule conflicts and prerequisites
6. System enrolls the student in the course
7. System displays confirmation message
8. System updates student's schedule and course enrollment count

Extensions or Alternate Flows:

- 4a. Course is full:
    1. System adds student to the course waitlist
    2. System displays waitlist confirmation
- 5a. Schedule conflict detected:
    1. System displays conflict warning
    2. Student can choose to cancel registration or proceed with conflict

Exceptions:

- Student does not meet course prerequisites
- System experiences a file I/O error during enrollment update

Related Use Cases:

- View Course Catalog
- Drop a Course

5.2 Faculty Grade Submission

Use Case ID: UC-FGS-001 Use Case Name: Submit Student Grades Relevant Requirements: 3.1.4.2, 3.1.4.3 Primary Actor: Faculty

Pre-conditions:

- Faculty is logged into the system
- Faculty is assigned to the course
- The grading period is open

Post-conditions:

- Student grades are recorded in the system
- Grade submission is marked as complete for the course

Basic Flow or Main Scenario:

1. Faculty navigates to the grade submission page
2. System displays list of assigned courses
3. Faculty selects a course
4. System displays the class roster with grade input fields
5. Faculty enters grades for each student
6. Faculty submits the completed grade sheet
7. System validates the entered grades
8. System saves the grades and marks submission as complete
9. System displays confirmation message

Extensions or Alternate Flows:

- 6a. Faculty saves grades as draft:
    1. System saves current progress without finalizing
    2. Faculty can return later to complete submission
- 7a. Invalid grade detected:
    1. System highlights invalid entries
    2. Faculty corrects errors and resubmits

Exceptions:

- System experiences a file I/O error during grade saving
- Grading period closes before submission is complete

Related Use Cases:

- View Class Roster
- Update Course Syllabus

5.3 Administrator Course Management

Use Case ID: UC-ACM-001 Use Case Name: Create New Course Relevant Requirements:

3.1.2.1, 3.1.2.2, 3.1.2.4 Primary Actor: Administrator

Pre-conditions:

- Administrator is logged into the system
- The course does not already exist in the catalog

Post-conditions:

- New course is added to the course catalog
- Course is available for student registration (if set as active)

Basic Flow or Main Scenario:

1. Administrator navigates to the course management page
2. Administrator selects "Create New Course" option
3. System displays course creation form
4. Administrator enters course details (code, name, description, credits, prerequisites)
5. Administrator sets maximum enrollment capacity
6. Administrator sets course status (active/inactive)
7. Administrator submits the new course information
8. System validates the entered information
9. System adds the new course to the catalog
10. System displays confirmation message

Extensions or Alternate Flows:

- 8a. Duplicate course code detected:
    1. System displays warning message
    2. Administrator can modify code or cancel creation
- 6a. Administrator sets course as inactive:
    1. Course is added to catalog but not available for registration

Exceptions:

- System experiences a file I/O error during course creation
- Invalid data format detected during validation

Related Use Cases:

- Edit Existing Course
- Set Course Prerequisites

6.UML Use Case Diagrams

7.Class Diagrams

8.Sequence Diagrams