**Group 3 – Enrollment**

Software Requirements Specification

Revision History

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 09/08/2025 | 1.0 | Initial Version: add functional requirements | Stephen |
| 09/08/2025 | 1.0 | Environmental and Performance requirements, External Interface Requirements | Harry |
| 9/17/2025 | 1.0 | Class Diagrams | Stephen |
| 9/22/2025 | 1.0 | Use Cases Document | Harry |
| 9/29/2025 | 1.0 | Use Case UML Diagram | Nathan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**1.** **Purpose 4**

1.1. Scope 4

1.2. Definitions, Acronyms, Abbreviations 4

1.3. References 4

1.4. Overview 4

**2.** **Overall Description 5**

2.1. Product Perspective 5

2.2. Product Architecture 5

2.3. Product Functionality/Features 5

2.4. Constraints 5

2.5. Assumptions and Dependencies 5

**3.** **Specific Requirements 6**

3.1. Functional Requirements 6

3.2. External Interface Requirements 6

3.3. Internal Interface Requirements 7

**4.** **Non-Functional Requirements 8**

4.1. Security and Privacy Requirements 8

4.2. Environmental Requirements 8

4.3. Performance Requirements 8

# Purpose

This document outlines the requirements for the CourseTrack College Enrollment System.

## Scope

This document will catalog the user, system, and hardware requirements for the CourseTrack system. It will not, however, document how these requirements will be implemented.

## Definitions, Acronyms, Abbreviations

CT: Course Track

Admin: Administrator

## References

Use Case Specification Document – Step 2 in assignment description  
<https://docs.google.com/document/d/1BtYeeRTmzw9WinrGeynCEGzKPkW1_f7GgNwaScKp0sU/edit?usp=sharing>

UML Use Case Diagrams Document – Step 3 in assignment description

Class Diagrams – Step 5 in assignment description

Sequence Diagrams – Step 6 in assignment description

## Overview

The CourseTrack College Enrollment System is designed to enable administrators and students to manage and enroll in college courses. Administrators and students across multiple university systems can register and enroll in many classes, implementing basic functionality of a registrar, checking prerequisites and registering the enrollment of every student. Provides basic information and audit reports about university enrollment data.

# Overall Description

## Product Perspective

CourseTrack is a Java-based system with a client and server application that lets students and administrators manage enrollment.

## Product Architecture

The system will be organized into \_\_\_ major modules: the \_\_\_ module, the \_\_\_ module, and the \_\_\_\_\_ module.

Note: System architecture should follow standard OO design practices.

## Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

## Constraints

List appropriate constraints.

Constraint example: Since users may use any web browser to access the system, no browser-specific code is to be used in the system.

## Assumptions and Dependencies

List appropriate assumptions

Assumption Example: It is assumed that the maximum number of users at a given time is 15,000.

# Specific Requirements

## Functional **Requirements**

**3.1.1. Users and Accounts**

3.1.1.1 User can log in with username and password

3.1.1.2 User can change password and info

3.1.1.3 User accounts with different roles (administrator, student)

**3.1.2. Administrator Functions**

3.1.2.1 Administrator can create/read/update/delete user accounts

3.1.2.2 Administrator can create/read/update/delete universities/campuses

3.1.2.3 Administrator can create/read/update/delete new term schedules of classes

3.1.2.4 Administrator can create/read/update/delete classes in term

3.1.2.5 Administrator can add schedule data by loading a formatted data file

**3.1.3. Student Functions**

3.1.3.1 Student can search classes available at specific term at university/campus

3.1.3.2 Student can enroll in classes

3.1.3.3 Student can drop enrolled classes

3.1.3.4 Student can view enrolled class schedule

3.1.3.5 Student can automatically generate viable schedules using interval-overlap detection between class meeting times.

**3.1.4. Class Data**

3.1.4.1 Class university

3.1.4.2 Department

3.1.4.3 Professor

3.1.4.4 Course number

3.1.4.5 Course name

3.1.4.6 Class maximum size

3.1.4.7 Prerequisite classes

3.1.4.8 Time and meet dates

3.1.4.9 (Semester vs quarters?)

3.1.4.10 Online vs in person

3.1.4.11 Synchronous vs asynchronous

Credits earned?

**3.1.5. Student Data**

3.1.5.1 Unique ID

3.1.5.2 Name

3.1.5.3 Password

3.1.5.4 Major

3.1.5.5 Classes taken

3.1.5.6 Classes scheduled/in progress

3.1.5.7 Total credits?

**3.1.6. Enrollment Validation**

3.1.5.1 Students may only enroll in courses that have fulfilled prerequisites.

3.1.5.2 Students are notified when they are not allowed to register a class yet.

3.1.5.3 Student may not enroll in two classes with conflicting meet times.

3.1.5.4 Student may only enroll in (max) credits per term

**3.1.7. Waitlisting**

3.1.6.1 Students registering for a full class will be placed on waitlist

3.1.6.2 Student can view their place in waitlist queue

3.1.6.3 Student is automatically enrolled if students drop the class

**3.1.8. Reporting: Basic Statistics**

3.1.7.1 Student can view degree progress audit

3.1.7.2 Administrator can audit class schedules for report on class size and capacity,

3.1.7.3 Which classes are most popular and least popular

**3.1.9. Reporting: Time-Based Analysis**

3.1.7.1 Accessing historical data to create audits and reports of

Allows administrators to view reports of enrollment trends in classes and departments to make administrative decisions.

3.1.7.2 Also allows students to view the most popular classes.

**3.1.10. User Notifications**

User has a collapsible notifications window.

User is shown notifications while logged in.

Notifications alert users to important events such as a student being enrolled in a course.

**3.1.11. Client-Server Architecture**

Client program must allow users to log in and access server remotely via remote API

All events and interactions must occur between client and server

User login and sessions and timeout

## External Interface Requirements

3.2.1 The client application must provide a text-based or GUI interface where students and administrators can log in, view courses, enroll, and run commands.

3.2.2 The course codes must be in correct format

3.2.3 Any kind of error messages must have meaningful messages to the current user like if classes are full.

3.2.4 The client will log errors into a log.txt file.

3.2.5 The client should kick out the user after X mins of inactivity.  
3.2.6 The client should have the option to log out.

## Internal Interface Requirements

3.3.1 The server will authenticate the username and password received from the client.

3.3.2 The server will log authentication attempts (success/failure) timestamped into a text file.

3.3.3 The server will store usernames and passwords in a csv file.

3.3.4 The server must save all course and student enrollment data to files so nothing will be lost after shutdown

3.3.5 The server must verify that students cannot enroll in courses with conflicting meeting times

# Non-Functional Requirements

## Security and Privacy Requirements

4.1.1 The client shall not display the password in plaintext (masking \*\*\*\*)   
 4.1.2 The client will terminate the session after inactivity

4.1.3 A user cannot access data above their privilege level  
 4.1.4 The client will not reveal sensitive data when an incorrect password is used (invalid username/password rather than invalid password)

## Environmental Requirements

Example:

4.2.1 System must run on Java 21 or higher

4.2.2 System must make use TCP/IP sockets for data transfer between the client and server application

4.2.3 Client and the Server must run on Windows, macOS, or Linux

4.2.4 Minimal use of external libraries

## Performance Requirements

Example:

4.3.1 System must render all courses within 5 seconds

4.3.2 Data must be saved immediately to files to prevent loss in case of crashes

4.3.3 Enrollment or drop requests must be processed in under 2 seconds