**Assignment #2 – Template**

*Software Requirements Specification*Revision History

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 09/17/2025 | 1.0 | Initial Version | Wail Mohammed |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

**1.** **Purpose**

This document outlines the requirements for the College Course Enrollment System Project.

**1.1.** **Scope**

This document will catalog the user, system, and hardware requirements for the CCES (College Course Enrollment) system. It will not, however, document how these requirements will be implemented.

**1.2.** **Definitions, Acronyms, Abbreviations**

CCES: College Course Enrollment System

**1.3.** **References**

Use Case Specification Document

UML Use Case Diagrams Document

Class Diagrams

Sequence Diagrams

**1.4.** **Overview**

The CCES (College Course Enrollment System) allows for the creation of college course schedules

by administrators and allows students to enroll in these courses. The system will support class sizes,

waiting lists, prerequisites, and reports. This system supports a network of universities, students, and

Administrators. This is a Java application with a GUI that operates over TCP/IP. This system requires

a server application and a client application. There is no web or HTML component.

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

**2.1.** **Product Perspective**

The CCES system is a platform designed for university students and administrators. Administrators can control the number of courses, class size, waiting list size, prerequisites lists, and issue reports. Students can enroll in courses, drop courses, and view schedules,

**2.2.** **Product Architecture**

The system will be organized into three major modules: the User module, the Course Scheduler module, and the Reporting (log) module.

**2.3.** **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):

**2.4.** **Constraints - Yesenia**

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.

**2.5.** **Assumptions and Dependencies—shichang wang and Wail**

List appropriate assumptions

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

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

**3.1.** **Functional Requirements**

**3.1.1.** **Common Requirements:**

Provide requirements that apply to all components as appropriate.

Example:

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 HTML-based help pages on each screen that describe the purpose of each function within the system.

**3.1.2.** **\_\_\_\_\_ Module Requirements:-Emmanuel**

Provide module specific requirements as appropriate.

Example:

3.1.2.1 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.

**3.1.3.** **\_\_\_\_\_ Module Requirements: - Yesenia**

Provide module specific requirements as appropriate.

Example:

3.1.2.1 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.

**3.1.4.** **\_\_\_\_\_ Module Requirements: -Shichang Wang(**Reporting (log)**)**

Provide module specific requirements as appropriate.

Example:

3.1.2.1 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.

**3.2.** **External Interface Requirements-Wail**

Provide module specific requirements as appropriate.

Example:

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.

**3.3.** **Internal Interface Requirements - Wail**

Provide module specific requirements as appropriate.

Example:

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 enrolments. 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.

**4.** **Non-Functional Requirements - ALL**

**4.1.** **Security and Privacy Requirements - Yesenia**

Example:

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

**4.2.** **Environmental Requirements- shichang wang**

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 - Emmanuel**

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.