

Assignment #2 – Template

Software Requirements Specification

Revision History

[illegible]

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):

2.3.1 The enrollment system must have an authentication, authorization and account creation system that differentiates between school admins and students based on their student or admin existing IDs.

2.3.2 Students will be able to access their online accounts, and input their name, email address, major and school year. Similarly, admins will be able to access their online accounts, and input their name, email address.

2.3.3 Students must be able to view college courses attributes such as; class IDs, names, times, days, instructor names, number of units, credit/no credit, number of available seats, and available waitlists .

2.3.4 School admin users must be able to view all class info, class availability status and student info.

2.3.5 The system must be large enough to handle and be able to store student account data, the list of all active courses, and the list of all waitlisted courses.

2.3.6 The system must allow admins to store and update a running list of courses with all courses attributes.

2.3.7 The system must allow students to view courses, must check if students are eligible to enroll, drop, waitlist, and/or withdraw from courses. System must not give any students access to other student's accounts.

2.3.8 The system must allow admins only to add/edit and delete courses and drop students if needed.

2.3.9 The system must be available to all users (students and admins) during course enrollment periods, and take into account earlier registration for students who have higher priority.

2.3.10 The system must be able to update courses info as well as students records in real time. Admins must also be able to see the up-to-date students and classes records.

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

Wail:

2.5.1. All users will have network access to connect to the system.

2.5.2. It is assumed that there will be two user roles within the system. The school administrator and the student role.

2.5.3. It is assumed that school administrators will maintain course addition, deletion, and maintain prerequisite requirements for courses that need it.

2.5.4. It is assumed that students are associated with one university.

2.5.5. It is assumed that each school has unique courses.

2.5.6 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. __The Student __ 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. __The School Administrator __ 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. __Course Planner/Reporting __ 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

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 interface must be clear, intuitive and provide a consistent user experience.

3.2.3 The system must provide error messages when invalid actions take place.

3.3. Internal Interface Requirements - Wail

- 3.3.1 The system must use a client-server communication via TCP/IP network.
- 3.3.2 The system must process a data-feed from the enrollment system such that the student updated course records are stored in the form of a comma-separated interface file that is exported from the enrollment system upon user's demand.
- 3.3.3 The data feed from the enrollment system must have the following fields included in the file ; student name, student id, major, school name, name and number of all courses enrolled and withdrawn.

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.