Week 1 Assignment- SRS Document Development

Alex Nieves  
CST499: Capstone for Computer Software Technology  
Instructor: Amr Elchouemi  
10/03/2022

Table of Contents

Table of Contents ii

Revision History ii

1. Software Requirements Specification 1

1.1 Introduction 1

1.2 Overall Description 2

1.3 External Interface Requirements 2

1.4 System Features 3

1.5 Other Nonfunctional Requirements 3

2. Test Management Strategy 4

2.1 Test Teams 4

2.2 Test Roles 4

2.3 Exit Criteria 5

2.4 Test Estimated Effort 5

2.5 Test and Risk 5

2.6 Incident Reporting 6

2.7 Defect Classification 6

2.8 Configuration Management 7

3. Test Tools 7

3.1 Introduction 7

3.2 Test Management and Control Tools 7

3.3 Test Specification Tools 8

3.4 Static Testing Tools 8

3.5 Dynamic Testing Tools 9

3.6 Non-Functional Testing Tools 9

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Software Requirements Specification

**1.1 Introduction**

## 1.1.1 Purpose

The specifications for the Course Enrollment System build are laid out in this software requirements specification (SRS) document. The Course Enrollment System's main goal is to give students access to a straightforward self-service website where they can manage their academic course schedule. The Course Enrollment System is described in this document as version 1.0. The Course Enrollment System's initial construction, which includes the fundamental elements mentioned in Section 4 as well as other non-functional needs mentioned in Section 5, is entirely covered by the scope of this SRS.

### Document Conventions

The introduction, overall description, external interface requirements, system features, and other non-functional requirements are the headings for the five sections that make up this paper. Subheaders give specific information about the contents of certain sections.

### Intended Audience and Reading Suggestions

This SRS is meant for the architects, developers, project managers, testers, UI/UX designers, software designers, architects, and other stakeholders involved in the project's implementation. It is advised that all audience members read the entire text in order to fully comprehend the goals and specifications of the Course Enrollment System.

### Product Scope

The Course Enrollment System's objective is to make it simple and quick for users to manage their academic course schedule. A user can sign up and make an account on the website, log in, examine their course schedule, look for open courses, register for a course, drop a course, and be put on a waitlist if a course is filled. The creation of this course enrolment system is in line with the company's goals to boost enrollment, keep students, enhance the overall student experience, and cut costs and overhead related to the manual course registration procedure that is now in use.

## 1.2 Overall Description

**1.2.1 Product Perspective**

The Course Enrollment System build is a brand-new, standalone product.

**1.2.2 Product Functions**

A user should be able to sign up, make an account, log in, view their schedule, look for open courses, register for them, drop them, and be put on a waitlist if a course is filled. A course may only have a certain number of students, so the website should note this. If a course is full, the system should add the student to the waitlist. The computer system will alert the student when a space in a full course becomes available.

## 1.3 External Interface Requirements

**1.3.1 User Interfaces**

For consistency, the header, footer, and navigation bar should be the same on every page of the Course Enrollment System. The Font icon set should be used for all icons. A user should be able to go back to the previous page from any page without losing their session data. Users should be able to tab through a website and all images should have alt tags. Additionally, fonts should be readable and colors should be chosen for their usability and aesthetic appeal. Ads that take away from the content shouldn't be there, and the overall page style should be consistent. Every page should have a clear indication of the home button. Additionally, forms should give consumers clear feedback that explains how to fix any mistakes they may have made.

**1.4 System Features**

**1.4.1 System Features**

* New User Account Creation
* Returning User Login
* Search for Available Courses
* View Course Schedule
* Course Registration
* Course Drop
* Course Availability Notification
* User Can Logout

**1.5 Other Nonfunctional Requirements**

**1.5.1 Performance Requirements**

It should take no more than 60 seconds for new users to register. In less than 3 seconds after conducting a course search, a user ought to be presented with a list of search results. Under 3 seconds should pass once a user clicks the "submit" button to register for or drop a course before getting a confirmation message. Speed is crucial since delays could lead to a bad user experience, discourage users from visiting the site again, and increase the workload at the registrar's office for course enrolment.

# Test Management Strategy

## Test Teams

It is crucial to take test teams and a methodology to each level of testing into consideration while developing the Course Enrollment System. Developers on the team should avoid testing their own code whenever possible. Software testing foundations: A study guide for the certified tester exam (4th ed.) (2014) states that "it is much more efficient to let different people perform testing and development and to organize testing as independently from development as possible" due to the tendency to be blind to our own errors. For this reason, component testing should be carried out by designated testers in the development team. Additionally, integration testing should be carried out by a defined testing team within the project. These testers could be from the IT or commercial sectors. Additionally, a specialized team with specialists should carry out system testing: "Especially in system testing, it is often required to enlarge the test team by adding IT professionals, at least temporarily, to carry out work for the test team" (Spillner, Linz, & Schaefer, 2014). This will make it possible to evaluate the system from many angles.

## Test Roles

In order to test the Course Enrollment System, people should be given the following test roles: test manager, test designer, test automator, test administrator, and tester (Spillner, Linz, & Schaefer, 2014). The test manager will need to have experience testing software, managing quality, running projects, and managing people. The test designer should be able to do testing, software engineering, and come up with test methods and specifications. A degree in Computer Science is also required. The test automator will need to have experience with testing, programming, scripting, test tools, and automation. The test administrator needs to know how to set up and support test environments, as well as how to run a system and connect to other computers. The tester should know how to follow instructions, run tests, report problems, and use test objects and testing tools (Spillner, Linz, & Schaefer, 2014).

## Exit Criteria

Three indicators will be utilized to determine whether testing of the course enrollment system is finished after test cases are prioritized and categorized depending on risk. The first is based on a time-series graph showing the number of flaws found. When testing first starts, there should be a high rate of defect discovery; however, this rate should decrease as the bulk of flaws are found and dealt with. Defect seeding and estimation will be the second technique used. Testing of the Course Enrollment System won't end until all flaws in high-risk test cases have been found, all flaws in medium-risk test cases have been found, and all flaws in low-risk test cases have been found. Finally, once 100% of the lines have been covered, testing for the Course Enrollment System should come to an end.

## Test Estimated Effort

Based on data from prior or related projects, test effort estimates for the Course Enrollment System will be made, as this approach yields the most accurate test effort estimates.

## Test and Risk

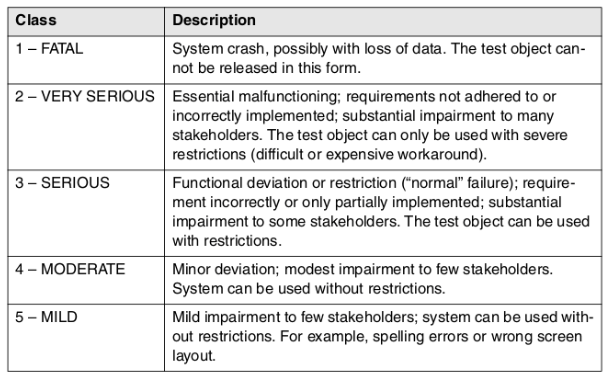
To make sure that the most important flaws in the Course Enrollment System are discovered as soon as feasible, risk-based prioritizing will be used. This will lessen the time and expense involved in handling major faults that reach production while preventing critical defects from having downstream effects.

## Incident Reporting

The Course Enrollment System will be developed and tested with incident reporting in place to record and manage incidents. Every problem that is found to be substantial and real (and not the result of a shoddy test) ought to be recorded. The incident report will include the following details: the program that was tested, the testing environment, the tester's identity, the class that contains the defect, the defect's priority, and details necessary for recreating and finding the fault (Spillner, Linz, & Schaefer, 2014).

## Assumptions and Dependencies

The Course Enrollment System will use defect classification to figure out which problems are most important. The level of damage caused by the defect will be shown by the severity level. There will be 5 different levels of how bad something is: 1-Deadly, 2-Very Serious, 3-Serious, 4-Moderate, and 5-Mild. The Figure (Spillner, Linz, & Schaefer, 2014) below shows the criteria for these levels.



## Configuration Management

During the development of the Course Enrollment System, configuration management will be used to keep track of the project's history of versions and to let multiple developers work on the project without interfering with each other.

# Test Tools

## Introduction

Software testing can be made easier with the help of test tools. Since the Course Enrollment System is new, test tools should be bought and used to make testing more efficient and reliable, cut down on manual testing, and do load and performance testing (Spillner, Linz, & Schaefer, 2014).

## Test Management and Control Tools

Testers and project managers can document, rank, list, and keep track of test cases with the help of tools that help with test management and control (Spillner, Linz, & Schaefer, 2014). Jira is a tool for managing tests that could be used to help build the Course Enrollment System. Tools for test management can be used to make sure there are test cases for each software requirement. A tool for testing that can do these things is called a requirements management tool. Test execution tools are another type of test management tool. These tools automatically run test scripts and record the results. Tools for incident management keep track of software bugs and how they are fixed. Tools for managing configuration keep track of the versions and builds that are going to be tested. Lastly, tool integration can be used to combine several test management tools into one (Spillner, Linz, & Schaefer, 2014). For the Course Enrollment System, a cost-benefit analysis should be used to figure out which tools will help the project the most.

## Test Specification Tools

For the Course Enrollment System, you should also think about tools for the specification phase of testing. There are four kinds of test generators: those that use a database, those that use code, those that use an interface, and those that use specifications. By filtering through databases, these tools help come up with test data. DatProf is an example of this kind of tool. It works with Windows and makes fake data using the most common database technologies (Software Testing Help, 2020). The source code is used by code-based test generators to make test data. Even though these tools can help, they can't find errors that are caused by missing code. Also, Spillner, Linz, and Schaefer (2014) say, "The use of code as a test basis for testing the code itself is generally a very poor foundation." Interface-based test generators find parameter domains, which can then be used for equivalence class partitioning and boundary value analysis to make test cases (Spillner, Linz, & Schaefer, 2014).

## Static Testing Tools

Tools for static testing should be used right on the Course Enrollment System's source code. They can help find bugs early in the development process, before the code is even run, because they can evaluate source code. Review tools are a type of static testing that help testers plan, run, and evaluate code reviews. They work like checklists (Spillner, Linz, & Schaefer, 2014). Polyspace is a tool that can be used for static analysis. The Polyspace Code Prover and the Polyspace Bug Finder are the two parts of this tool, which came out in 2014. "The first one uses all possible values of variables to find every code instruction, while the second one is used to do static analysis" (Khalid, 2017).

## Dynamic Testing Tools

When making the Course Enrollment System, using dynamic testing tools will help cut down on the mechanical work that comes with running tests by hand. These tools feed data into the test object and record the results. Debuggers, test drivers and test frameworks, simulators, test robots, comparators, dynamic analyzers, and tools for coverage analysis are all examples of dynamic testing tools (Spillner, Linz, & Schaefer, 2014). Popular text editors and IDEs have debuggers built in. Dynamic analyzers check how much memory is being used and where it is being put. They also look for "memory leaks, wrong pointer allocation, or pointer arithmetic problems" (Spillner, Linz, & Schaefer, 2014).

## Nonfunctional Testing Tools

Tools for non-functional testing should also be taken into account while evaluating the Course Enrollment System's quality characteristics. These characteristics are frequently challenging to assess alone manually. Other non-functional testing tools besides those for performance and load testing include those for evaluating data quality and testing security. Early in the development process, security flaws must be found. No matter how thorough their safeguards may be, developers cannot afford to think that their security requirements throughout development are flawless and impenetrable, according to Benchmark Requirements for Assessing Software Security Vulnerability Testing Tools (2018). Developers, testers, and project managers can create a strong and dependable course enrollment system by utilizing these testing tools.

**References**

Chikh, A., & Alajmi, H. (2014, January 17–19). Towards a dynamic software requirements specificationLinks to an external site. [Paper presentation]. 2014 World Congress on Computer Applications and Information Systems (WCCAIS), Hammamet, Tunisia. <https://doi.org/10.1109/WCCAIS.2014.6916656>

Khalid, R. (2017). Towards an automated tool for software testing and analysis. 2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Applied Sciences and Technology (IBCAST), 2017 14th International Bhurban Conference On, 461–465. https://doi- org.proxy-library.ashford.edu/10.1109/IBCAST.2017.7868094

Software Testing Help. (2020). Top 10 Best Test Data Generation Tools in 2020. Retrieved June 4, 2020 from https://www.softwaretestinghelp.com/test-data-generation-tools/.

Tsui, F., Karam, O., & Bernal, B. (2018). Essentials of software engineering (4th ed.). Jones & Bartlett Learning.

Wiegers, K. E. (1999). Software requirement specifications for <project>Links to an external site. [Template]. <https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc>

Link: <https://github.com/adanielnieves/CST499_Project>