**Development Project II (420-K50-HR)**

## Assignment 3 –System Test Preparation, Planning and Execution &

## Operational Documentation

## Table of Contents

[Assignment 3 –System Test Preparation, Planning and Execution & 1](#_Toc194912857)

[Operational Documentation 1](#_Toc194912858)

[Table of Contents 1](#_Toc194912859)

[Learning Objectives 2](#_Toc194912860)

[Section 1 (Testing) To Do: 3](#_Toc194912861)

[System Test Plan 3](#_Toc194912862)

[SECTION 1 Part B – System Test Setup 5](#_Toc194912863)

[SECTION 2 Part C – System Test Execution and reporting 5](#_Toc194912864)

[SECTION 1: To Submit 6](#_Toc194912865)

[SECTION 2: Part A - Operational Documentation - Runbook 7](#_Toc194912866)

[SECTION 2: Part A: Setup and Prep 7](#_Toc194912867)

[SECTION 2: Part B: System Design Documentation 8](#_Toc194912868)

[SECTION 2: To Submit 9](#_Toc194912869)

[Appendix I – System Test Process 10](#_Toc194912870)

[Appendix II – Installation Notes and Hints 11](#_Toc194912871)

[Appendix 2: Installation Overview 11](#_Toc194912872)

[Appendix II: Deployment of Database to a DB Server 11](#_Toc194912873)

[Appendix II: Deployment of Web Application to an Application Server 12](#_Toc194912874)

Date assigned: April 7, 2025

Date dues: See table 1, below.

Table 1: Assignment Due Dates

|  |  |  |
| --- | --- | --- |
| **Assignment Segment** | **Assignment Subsection** | **Due Date** |
| SECTION 1: Testing 1 | Section 1 Part A: Test Cases Written | Tuesday, Apr 8,  before midnight |
| SECTION 1: Testing 2 | Release Candidate Build 1 Build and Deployment Recipe | Thurs, Apr 10,  before midnight |
| SECTION 1: Testing 3 | RC Build 1 (vers 0.1) Setup and Test Cases executed, results reported  Section 1 Part B & Part C  (Individual, Dev environment) | Tues, Apr 15,  before midnight |
| SECTION 1: Testing 4 | RC Build 2 (vers 0.2) Setup and Test Cases executed  (Individual, Test environment) | Tuesday Apr 22,  before midnight |
| SECTION 1: Testing 5 | RC Build 2 Setup and Test Cases executed, results reported (Combined Team Results) | Weds Apr 23,  before midnight |
| SECTION 1: Testing 6 | Rerun Team Tests on RC Build 3: Release Candidate Build 3 (vers 0.3) in CSTEST, Production Readiness Spreadsheet  *(if applicable)* | *Date TBD*  (after 18th of April, if applicable) |
| SECTION 1: Testing 7 | Results reported, Public Testing or Production URL provided to user/customer  *(if applicable)* | Fri Apr 25 CS Prod, before midnight |
| SECTION 2: Operational and System docs 1 | Draft installation recipe | Thurs Apr 24,  before midnight |
| SECTION 2: Operational and System docs 2 | Runbook skeleton and tasks on CSAzure Devops | Fri Apr 25,  before midnight |
| SECTION 2: Operational and System docs 3 | Runbook Reviewed, Updated (based on feedback from peers) Submitted on Moodle | Monday April 28th,  before midnight |
| SECTION 2: Operational and System docs 4 | System Design Document Skeleton | Tuesday Apr 29th,  before midnight |
| SECTION 2: Operational and System docs 5 | System Design Document Reviewed, Updated  (based on feedback from peers) and Complete | Friday May 2nd,  before midnight |

## Learning Objectives

Upon successful completion of this assignment, the student will be able to:

* Develop a system test plan and system test cases
* Run a system test consisting of the above test cases
* Document defects found as a result of a comprehensive system test.
* Deploy a system to the development & test environments
* Deploy applications using the CI/CD Pipelines function in DevOps Azure.
* Deploy a new web application to a web server
* Secure a database connection on an IIS web server
* Update a Runbook for deploying a web application and a database
* Create a System document for future developers & the CS Maintenance Team

## BIG BLOCK LETTERS WARNINGS

* **This assignment is worth 30% of your course mark**
* **Students are required to pass this assignment in order to pass the course.**
* **Please note that if you do not get 80% or greater on your peer evaluation, you will fail the assignment, as well.**
* Students who complete and fail this assignment before late marks are deducted may correct and resubmit the assignment for a maximum mark of 60% less the late marks on the original assignment (I.E. 40% of the deductions applied still apply). Applies to SECTION 1.
* You may do research and investigation as a team, however the documentation and submissions for A3 SECTION 1 of this assignment is individual work only. You will combine your results, however, your mark is still individualized.
* A3 SECTION 2 is both individual and team work.

**Context:**

During this assignment, you and the team will need to:

* Write a comprehensive system test plan
* Execute your test plan on a private copy (private app in IIS and DB on CSDEV)
* Triage, document and fix bugs found
* Republish, rerun the system tests to validate the environment and application are “sane” and,
* Test, and clean up the environments each time to ensure sanity between testing runs.
* Develop system documentation and incrementally add to that documentation

## SECTION 1 (Testing) To Do:

## System Test Plan

The purpose of the system test is to verify that the system functions according to the user stories by interacting with the application via the user interface and analyzing the output or results. In particular, the focus should be on the aspects defined in the System Test template document.

Create a new document named **YourUserName\_K50\_A03\_System\_Test** in your home drive based on the template provided.

see Moodle for Heritage–SystemTest-Template.docx

The document will contain a test plan for the system you have worked on. Fill out the document so that is consumable for the intended audiences.

Each test section should have a paragraph explaining the purpose for the tests. i.e. in the functional test section, define and describe the purpose of functional tests.

For functional test cases, use the following template and example as a guide to writing the test cases. Functional tests are written from a user's perspective. These tests confirm that the system does what users are expecting it to. Functional test cases should include a series of steps that are interrelated to ensure that they function as expected and should address how two parts of the system relate to each other. These tests are feature interaction tests as individual functions are covered by User Stories and their acceptance tests. The purpose of the system test is not to test all the error conditions. It is assumed that this was done during unit test. Include a unique system test case ID. Provide a **minimum** of 10 functional tests.

The prefix for functional tests is “FN” i.e. “FN-02”

**Example: Functional Test Case Table**

| **Test ID** | **Purpose** | **Expected Result** |
| --- | --- | --- |
| FN01 | Verify the link between the scheduling of an evaluation and the availability of the evaluation to the student. | Students cannot see the evaluation before the start date.  Students can see the evaluation between the start date and the end date.  Students cannot see the evaluation after the end date. |
| FN02 | … fill in here | … fill in, in detail, here |

Usability testing involves testing for consistency in the user interface, navigation, and ease of use. Include usability test cases for the system in a Usability Test Case Table, using a format similar to the Functional Test Case Table and prefixing the ID with US. Provide a minimum of 5 usability tests.

The prefix for these tests is “US” i.e. “US-5”

Compatibility testing involves testing using different browsers, devices, and operating systems. Include compatibility test cases for the system in a Compatibility Test Case Table, using a format similar to the Functional Test Case Table. Provide a minimum of 3 compatibility tests.

The prefix for these compatibility tests is “CO” i.e. “CO-1”

Security testing includes ensuring that only those actors specified to execute the user stories can do so and that the system is secure. Include security test cases for the system in a Security Test Case Table, using a format similar to the Functional Test Case Table. Provide a minimum of 5 security tests.

The prefix for security tests is “SE” i.e.. “SE-05”

Business cycle testing emulates the activities performed on the system over time. Transactions and activities that would occur during at least two consecutive semesters or years should be executed. Include business cycle test cases for the system in a Business Cycle Test Case Table, using a format similar to the Functional Test Case Table. Provide a minimum of 3 business cycle tests.

The prefix for business cycle tests is “BC” i.e. “BC-3”

Scale testing requires that you prove your system runs at the scales specified by the user. Capture the scale values and explain your plan for what testing will be covered at scale.

The prefix for scale testing tests is “ST” i.e. “ST-13”

Performance testing requires that you prove your system runs at the performance limits specified by the user. Capture the performance metrics and explain your plan for capturing the metrics and identifying any failures. Note, the performance standard for this course is that every page must be rendered within 2 seconds (worst-case time). This applies to all

The prefix for performance testing is “PERF” i.e. “PERF-3”

## SECTION 1 Part B – System Test Setup

Deploy your own private copy to the CSDEV application and database server. This includes your own database and the latest version of the application. Use the “CSDEV” CICD Pipeline on CSAzure for this.

(See “K50 CICD Pipelines Azure Graphics Package 2.zip” package on Moodle)

Deploy your own private copy to the CSTEST application and database server. This includes your own database and the latest version of the application. Use the “CSTEST” CICD Pipeline on CSAzure for this.

Publish to path Projects/test/projectName/uname

For 1. and 2. above: Note that you will need to manually attach this application to the AppPool in IIS. The Azure DevOps Pipelines deployment log has the deployment location on the file system.

Start capturing the recipe on how to do this as you will need it to complete Assignment 5 – Operational and System Documentation (i.e. how to deploy your application). Capture this in a memo, entitled **YourUserName\_K50\_A03\_Test\_SetupRecipe.docx**

Setup the Test report based on the Test Results Template provided in Moodle. Call it **YourUserName\_K50\_A03\_TestResults.xlsm**

Ensure you have all the test scripts, sample data and inputs you may need to execute your test cases.

## SECTION 1 Part C – System Test Execution and reporting

Run the system test cases that were identified, using the deployed application, your individual databases, and **representative** data:

Record the actual results in the Test Results spreadsheet as Passed/Failed, including the error if the test case failed, and record any defects in DevOps Azure.

Do not look at bugs from other team members on Azure DevOps. We will remove the duplicate bugs at a later stage.

You will be evaluated based on the quantity and quality of bugs you find and on your description of the bugs, not the number of changes you make.

On Moodle: see the “Heritage\_Test\_Results\_Template.xlsm”.

Perform exploratory testing on the system and record any defects in Azure DevOps. Include the steps to recreate the bug in the description. List your exploratory bugs found in the test report.

Write a brief conclusion to the System Test Report, summarizing the number of defects found for each type of test and commenting on the severity of the defects and the overall quality of the system.

Update the table of contents in the test plan.

Resubmit the test plan and the test results into Moodle.

## SECTION 1: To Submit

**Submit portions to the appropriate links in Moodle. Note the Rubric also provided in Moodle on how this assignment will be assessed.**

Also please note that your primary evaluation is based on your individual submissions. WORK TOGETHER, SHARE IDEAS, but submit individually. This is different than the other assignments in this course.

## SECTION 2: Part A - Operational Documentation - Runbook

You will create a [Runbook](https://en.wikipedia.org/wiki/Runbook) that will capture your operational documentation. A template has been provided for you in Moodle.

You are free to modify the template, layout or organization as long as the required information is provided and well organized.

One of your goals is that with just the Runbook, access to your software (tools, scripts, repository, etc.) and access to the production environment, the system administrators are equipped to install, configure, run, monitor and maintain your system.

Recall, that a typical production environment may have 100’s or 1000’s of systems simultaneously running and that system administrators depend on well documented, well-packaged systems.

Save your Runbook as **YourUserName\_Teamxx\_K50\_A03\_RunBookSystemDoc v1.docx.** Include your (short) username and team number. You’ll add the system doc to this document, next.

## SECTION 2: Part B: Setup and Prep

Take a look at the Runbook template. Determine how to organize and capture all the relevant details of deployment in the Runbook. Add additional sections to the Runbook if needed.

Figure out how and where you will store your deployment and migration items (i.e. any scripts or tools that you will reference from the Runbook that your users will need).

Research and ensure that you know:

How to publish an application

How to copy and deploy a database

How to securely setup an application and a database (consider HTTPS, App Pools, no hard-coded usernames/passwords).

Terminology and Concepts

Target Server – A specific location or server that will host a service such as a database or application components.

Deployment instructions in your Runbook should be able to work on a variety of instances. For example, you shouldn’t hard-code in your instructions the names of target servers (i.e. which server the Database will run on, or what server the Application will run on). You’ll generally explain how to work on any Target Server as long as the Target Server meets your clear and precise requirements.

For example, in our environment, the System Administrator should be able to follow the document to install the application on CSTEST or CSPROD.

Installation

Describe the requirements for installation of your system.

Describe how the steps to install on a selected Target Server.

Make it more concrete by tying this into how this would work on installing it on a specific system. Select our CSTEST environment.

How to confirm that the installation is successful.

Hints:

You can highly reference the general section, and indicate the details (perhaps in a table format) to substitute in for the concrete example. i.e. a table with parameters, such as Target DB server, Target Application Server, and all the other selectables). Do not list all the steps and details again.

Read and Understand Appendix I

Upgrade/Downgrade

If your system supports in-place upgrades, with or without the persistence of users’ data, you need to explicitly describe what is possible.

Specifically, if there is already an instance of your application running on the Target servers and a new release is available.

Monitoring

Monitoring is understanding what is going on in the system.

It should cover:

How to determine if the application is functioning properly

How to determine if the application is performing properly

Maintenance

Maintenance includes any tasks that needs to be done for your system to keep it healthy.

Details for each task should include task purpose, steps and frequency (or what is the trigger to do this task).

Typical tasks include:

How to startup or shutdown the application (ideally, in a graceful way)

How to trim logs or other output that, left unchecked, might blow up a file system.

## SECTION 2: Part C: System Design Documentation

You’ll add this documentation to your existing documentation. And in Part B you’ll submit a second version (v2). More on V2 below.

Consider maintenance teams that will inherit the wonders of your creation. This is the audience for the System Design documentation. It must cover details on how the system is implemented, including (but not limited to):

* Coding and organizational standards (where to find what)
* Architecture (i.e. layers and blocks), high level design of the system, class diagram
* Data Model
* How to find the code, build, run, revise and any other common tasks that they will need to do.

This does not have to be very large, but sufficient details to give the maintenance team context so that they can work with the project.

You may freely copy and update from your past design documents.

Once again, your audience is 3rd yr Computer Science student that has no concept of what your project does, how it’s made or how to run it.

Present this in a report format (Title page, TOC, Introduction, References, Body, ….)

**Save this as YourUserName\_TeamXX\_K50\_A03\_RunbookSystemDoc v2.docx.** Submit on moodle, per instructions, below.

## SECTION 2: To Submit

**Submit portions to the appropriate folders in Moodle. Note the Rubric also provided in Moodle on how this assignment will be assessed.**

## Appendix I – System Test Process

You will be doing 2 passes of your system test plan.

Results of each pass of your test plan will be captured in your Test Results Spreadsheet. There is a separate column for each pass and it produces summary statistics.

Once the first pass of the system test is done, each person submits:

Test Results Spreadsheet

Enters Azure DevOps bugs with the “Test Bucket” target. We will triage this Test Bucket later. Tag all entered bugs – as a group, use a standard that you agree to i.e. “FUNCTIONAL” tagged bugs, etc. Tag any adhoc bugs with “adhoc”. Adhoc bugs that are not of the other types described above (i.e. Functional, Scale)

**Post first pass results.**

Enter any user found bugs. Tag with “user”.

Pairwise review for severity and proper target.

Score.

Group deduplication.

Triage.

Plan.

Prioritize bugs, outstanding, user Q&A and explanation of the current bugs.

As course time allows, we will do a debugging dev sprint for a few days. The build is to be called RC3 or Release Candidate 3 (version 0.3), and it will be pushed to the CSTEST Application Server and MSSQL Database.

## Appendix II – Installation Notes and Hints

## Appendix II: Installation Overview

This section involves writing an installation guide.

Installation instructions should include the following details:

Instructions on how to open the application from source control in Visual Studio, including steps on connecting to the DevOps server.

Instructions on how to deploy the application.

Instructions on how to add the application pool if it doesn’t exist.

Instructions on how to set up the security on the databases in SQL Server.

Instructions on how to verify that the application has been correctly deployed, including a link to the deployed application.

Include any other instructions required.

An appropriate number of screen shots which would enhance the instructions. You must use your own screen shots. Refer to the following source on how to take proper screen shots: <http://www.hanselman.com/blog/TakingProperScreenshotsInWindowsForBlogsOrTutorials.aspx>

**You must properly reference any citations or sources used. Put a Reference section at the bottom of your document and preferably follow APA 7 notation. Or whatever citation style you’re comfortable with.**

## Appendix II: Deployment of Database to a DB Server

This section involves how to create a deployment of your database environment in the development server.

Deploy the skeleton schema to the Target DB, using pre-generated scripts (via Entity Framework or other)

Deploy the installation data scripts (seed data) for basic application functioning

Deploy any test data required

Ensure data “sanity” by examining all table data

Complete the “Deployment of Web Application Server”

Point application to Target DB via configuration

Run application to ensure sanity

Examine machine-level SQL logs to ensure that no issues remain

## Appendix II: Deployment of Web Application to an Application Server

This section involves deploying the application to the development server. You may need an application pool created has been created (by a prof or system administrator) which will be used to run the application on the web server and a corresponding login has been created on the database server.

I’ve put in some details for getting this on CSDEV, but you need to abstract so that this works on other target servers.

Make sure that you have the latest version of the application and ensure that the web.config file is using the individual database created in Part B and your individual application database.

Change all the connection strings in your config files use Windows integrated authentication instead of a username and password.

Ensure there are no hardcoded details that wouldn’t survive an environmental change. This includes people. For example, support emails and other contact information should not be hardcoded by be configuration file or database driven.

Build the application.

Deploy the application to a site in the **Projects** folder with your uName/projectName on the CSDev web server (<http://csdev.cegep-heritage.qc.ca>).

Login to the CSDev development web server using Remote Desktop Connection.

Start Internet Information Services Manager (IIS).

In IIS, change the application to use the default AppPool, which is the identity of the application pool under which the application will run on the web server, and make sure that the Login page is the first item in the list of default documents.

In SQL Server, change the security on your databases to assign the required rights for the default AppPool login.

Test the installed application to make sure it runs.

**Be sure to logoff or sign out of the server when you are done using the connection, but do NOT shut down the server**.