

Assignments

Testing Documentation - In progress

Honor Pledge Accepted
Draft - In progress
Submitted
Returned

Assignment Details

Title
🧩 Testing Documentation
Due
Number of resubmissions allowed
Unlimited
Accept Resubmission Until
Apr 10, 2023 11:55 PM
Status
Honor Pledge Accepted
Grade Scale
Points (max 100.00)

Instructions

CS2212 Group Project - Testing Documentation
Winter Session 2023

Project Description

This assignment is part of the group project for CS2212. A full description of the project itself [can be found here](#).

Purpose of the Assignment

The general purpose of this assignment is to leverage the group project specification linked above, as well as your requirements and design documentation, and produce testing documentation useful in the verification and validation of your software implementation. In particular, this will give you experience in:

- Reading detailed software project specifications.
- Producing test cases from requirements and design documentation.
- Producing test cases from code.
- Writing test plans and testing documentation.

Due

Important: The due date for this component is based on the date of your Acceptance Testing meeting, it is NOT April 6th as listed on OWL!!

This assignment is due 24 hours after of your Acceptance Testing meeting with your team's assigned TA. These meetings will be tentatively scheduled to take place between April 3rd to 6th. Your team's meeting time will be assigned closer to these dates. For planning purposes you should assume your meeting will be on the earliest date (April 3rd).

No implementation will be accepted after April 10th at 11:55pm, even if you have late coupons remaining.

Late Penalty

Late assignments will be accepted for up to three days after the due date, so long as your team has late coupons remaining (every team starts the term with 6 late coupons). One late coupon is used for each day late (up to three days). After 3 days, or if your team has no late coupons remaining, you will receive a zero grade on this project component.

This assignment allows resubmissions. Only the most recent (the latest) submission will be considered and graded. This means a late penalty will be applied if you submit past the due date. The teaching assistants will only have access to your latest submission, so please ensure it is the correct one.

Group Effort

This stage of the project is expected to be a group effort, with each member of the group contributing equally in a reasonable fashion. Feel free to discuss ideas with other groups in the class; however, your submission must be the work of your own group. If it is determined that you are guilty of cheating on the assignment, you could receive a grade of zero with a notice of this offense submitted to the Dean of your home faculty for inclusion in your academic record.

What to Hand in

Your submission, as noted above, will be electronically through OWL. You are to complete your documentation using our [Confluence collaboration system](#) and submit an [exported PDF version](#) of this documentation as discussed below. Only one submission per group is necessary.

Further details on how to access and use our Confluence installation have been posted to OWL on the Project Software tab on OWL (nested under the Team Project tab).

In addition to the Testing Documentation, this compartment will also be assessed on the [JUnit](#) tests included in your [Bitbucket repository](#) zip archive that you submitted as part of the implementation project component. A passing grade on this project component will only be possible if both are completed (the testing document PDF submitted via OWL and your JUnit tests submitted as part of implementation project component).

Useful Resources

If you would like to get a start on this project component right away you may need to read ahead a bit in our textbook and review some of the resources below until we cover these topics in-class.

- [JUnit Website](#)
- [JUnit 5 User Guide](#)
- **Textbook Resources:**
 - Chapter 19 and 20

Assignment Task

In this assignment, you will craft testing documentation for the group project for this course. Given the project specification [that can be found here](#), the requirements, and design documentation your group prepared, and your implementation still in progress, you are to produce documentation that captures test cases that ensures that your software meets all project requirements and does so correctly.

As noted above, you are to prepare your testing documentation using our [Confluence installation](#). As you know from your earlier work, for this course, each group is given its own project space in this wiki. For this assignment, you are to create a page under this space titled "Testing Documentation" and you will proceed to fill the page with the various sections of content outlined below. As necessary, you can create sub-pages to provide a suitable structure to your documentation.

Your documentation must consist of the following sections and content:

1. Main Page

The main page for this assignment, Testing Documentation, should have the following content.

- The title for the project, and a sub-title indicating that this is the testing documentation for the project.
- A **tabular revision history** for your document. It should be updated as people add to and edit your testing documentation. It should be formatted roughly as follows:

Version	Date	Author(s)	Summary of Changes
---------	------	-----------	--------------------
- A **table of contents for the document**. This can be created and maintained automatically by using the "Table of Contents" or "Children Display" macros in Confluence, and will enable ready access to the other sections of the documentation.

2. Introduction

This sub-page provides an introduction to your testing documentation. As such, it should provide the following content. According to your preferences, this can either be delivered as sections on the introduction page, or as sub-pages.

- **Overview:** This section provides an overview of the problem being solved by this software and the requirements of the system. This should be a brief executive summary focused on testing.
- **Objectives:** This section outlines the general objectives of the project.
- **References:** A list of references to other documents that might provide context or otherwise assist in the understanding of this document.

Note that this section can be mostly copied from your requirements and design documentation, although it should be updated or adjusted as necessary for a focus on testing and incorporating any major changes since the last document. The project specification, requirements and documentation, and design documentation should be listed in your references as a minimum.

3. Test Plan

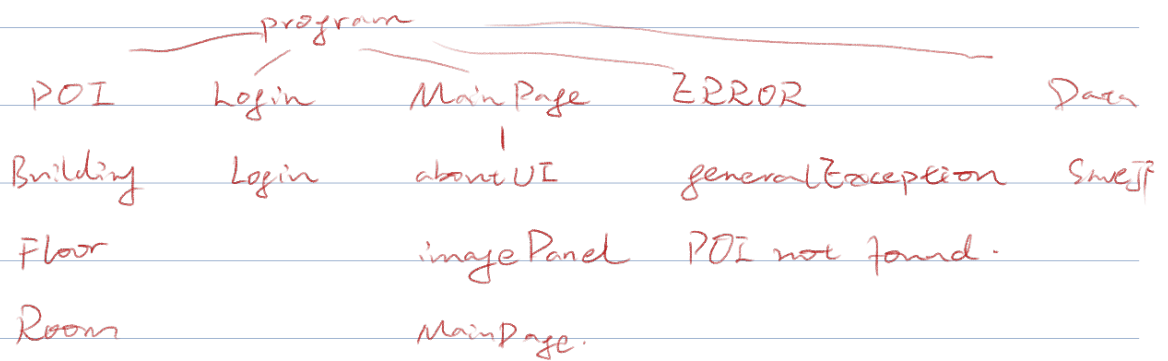
This sub-page discusses the detailed test plan for your software system. This plan should indicate both the test cases that you are using to assess the quality of your software, as well as a justification for their inclusion as part of your test plan. This discussion should indicate why your plan is both sufficient and complete. In delivering its detail, this plan should include a number of sub-pages, each covering one of the following:

- **Unit Testing:** Unit testing concentrates on each unit (for example, component, class, etc.) of the software as implemented in source code.
- **Integration Testing:** The focus of integration testing is on the coming together and connecting of the above units, as reflected in the software's overall design and architecture.
- **Validation Testing:** In this testing, requirements established as part of requirements modelling are validated against the software has been constructed.
- **System testing:** Here, the software and other system elements are tested as a whole.

To support the above testing, you will need to use a mixture of both white box or structural testing (focusing on control flow, decision paths, and boundary cases) and black box or functional testing (focusing on assessing whether the software meets all of its functional and non-functional requirements). Note that these approaches are complementary and will each uncover different classes of errors and issues with your implementation.

integration Testing:

Depth-first top-down incremental approach.



Where possible, particularly for your unit testing, you should make use of [JUnit](#) (documentation available [here](#)). It provides a useful testing infrastructure for your Java code. These tests should be included as part of your code in your [Bitbucket repository](#) and cover all major functions (units) of your software's code.

For unit testing, you can just give a general description of how unit testing will be done (as the actual tests will be included in your code). You should have unit tests for any important public method in your software. Automated Unit tests are not required for GUI components or generated code.

For integration, validation, and system testing you must detail the tests you will perform on your software using the template below:

Test Case Name:	<i>This field uniquely identifies a test case.</i>
Test Case Description:	<i>This field describes the test case objective.</i>
Test Steps:	<i>In this field, the exact steps are mentioned for performing the test case.</i>
Pre-Requisites:	<i>This field specifies the conditions or steps that must be followed before the test steps executions.</i>
Expected Results:	<i>Expected output from the test case data or the criteria for success.</i>
Test Category:	<i>The type or category of the test (e.g. unit test, integration test, validation test, system test, etc.).</i>
Requirement:	<i>Reference to the requirement being tested/evaluated.</i>
Automation:	<i>Whether this test case is automated (e.g. using JUnit) or not (e.g. manually run by a human).</i>
Date Run:	<i>The date/time this test was last run.</i>
Pass/Fail:	<i>If this test was passed or failed when last run.</i>
Test Results:	<i>The output or results of the test.</i>
Remarks:	<i>Remarks and comments regarding the last time the test was executed (e.g. explaining why the test failed).</i>

Additionally, for integration testing you should state if you are doing Top-Down Integration or Bottom-Up Integration and describe your integration process (e.g. are you making stubs or drivers for test?). This description should be consistent with the integration tests you list. For system testing, at least one of the tests you list should be ensuring your software works properly on Windows 10. You should provide enough tests to cover all major functions of your software.

4. Summary

Provide a brief summary wrapping up your document and summarizing it's contents. If you've introduced terms or notations in your document, you should add a table defining such things concisely for future reference.

Submission

You are representing the following group:

☐ Team 60Yulun Feng (yfeng445) , Hanley Han (hhan97) , Haoran Wang (hwan372) , Tianchen Wang (twang738) , Shunxi Yu (syu428)

 Your submission will be sent to Turnitin to be reviewed for plagiarism.

- Only the following file types will be accepted: PDF (.pdf), Word 97-2003 (.doc), PowerPoint (Presentation) (.ppt, .pptx), PowerPoint (Slideshow) (.pps, .ppsx, .pptx), Excel 97-2003 (.xls), Word 2007+ (.docx), Excel 2007+ (.xlsx), PostScript (.ps), rich text (.rtf), HTML (.htm, .html), WordPerfect (.wpd), OpenOffice (.odt), plain text (.txt)

Attachments

No attachments yet


Select a file from computer No file chosen

Proceed

Preview

Save Draft

Cancel

 Don't forget to save or proceed!