Southern New Hampshire University

CS 320 Final Project Guidelines and Rubric

Overview

The final project for this course is the creation of a summary and reflections report.

While software quality assurance may not always elicit the same excitement as other aspects of the software development life cycle, it is nonetheless vital. A critical component of this process is software testing. In fact, it can be said that code should not be considered "clean" unless it has been tested. The ability to create unit tests that effectively uncover errors within software features is an essential skill for any software developer.

In this final project, you will execute a test plan provided in the Final Project Test Plan document for a given software application. The test plan will describe the requirements of the development project at a high level as well as indicate which features must be tested to ensure verification and validation. You will need to select, write, and run the specific unit tests. You should find **at least five** of the errors that exist within the application's code by applying techniques you have learned in the course. Based on your experiences, you will ultimately deliver a final summary and reflections report. In this report, you will document your decision making and results as well as reflect upon your lessons learned by drawing connections to the broader software quality assurance principles discussed in the course.

The project is divided into **two milestones**, which will be submitted at various points throughout the course to scaffold learning and ensure quality final submissions. These milestones will be submitted in **Modules Four and Five.** The final product will be submitted in **Module Seven.**

In this assignment, you will demonstrate your mastery of the following course outcomes:

- Evaluate various software testing techniques for their potential to meet the needs of different software development projects
- Identify strategic approaches for software unit testing based on applications' requirements
- Create technically sound unit tests using logical, efficient code that effectively uncovers errors within software
- Illustrate best practices for managing software testing to ensure continuous quality improvement over the software development life cycle
- Articulate the value of maintaining a disciplined, quality-focused mindset as a software engineering professional

Prompt

Your **summary and reflections report** should be a paper documenting your decision making and the results of your software testing as well as your reflections upon your lessons learned by drawing connections to the principles discussed in the course.

Specifically, the following **critical elements** must be addressed:

I. **Summary**

A. Describe your unit testing **approach** for each of the three features.

i. What was your **level** of testing (i.e., number of tests) for each of the features? Explain why your level of testing was appropriate based on what you have learned about testing best practices in the course.

ii. How did you **organize** your tests (e.g., grouping tests into test suites) for each of the features? Explain why your organizational strategy was appropriate based on what you have learned about testing best practices in the course.

iii. To what extent was your approach **aligned** to the software **requirements**? Support your claims with specific evidence from the test plan.

B. Describe your experience **writing the JUnit tests**.

i. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate.

ii. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.

C. Defend the overall quality of your JUnit tests. In other words, demonstrate that your **tests** were **effective** in finding the errors. Cite the specific number (at least five are required) and nature of the errors found.

II. **Reflection**

A. **Testing Techniques**

i. What were the software testing **techniques** that you **employed** in this project? Describe their characteristics using specific details.

ii. What are the **other** software testing **techniques** that you learned about in this course? Describe their characteristics using specific details.

iii. For each of the techniques you discussed, explain the practical **uses and implications** for different software development projects and situations.

B. **Test Management Practices**

i. Now that you have experienced the testing side of software development, assess the **importance** of properly managing software testing. What are the risks of not appropriately verifying and validating your code? Support your claims with specific evidence and principles discussed in the course.

ii. Analyze the major components of administrating testing that must be considered and implemented in order to ensure software quality. In other words, what are the **best practices** you have learned regarding testing strategy, planning, and monitoring? Illustrate your response with specific examples and principles discussed in the course.

iii. Finally, identify the types of **tools and technology** available for implementing testing strategies. For example, to what extent can automation be used to support quality software development? Illustrate your response with specific examples discussed in the course.

C. **Mindset**

i. Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ **caution**? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.

ii. Assess the ways you tried to limit **bias** in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.

iii. Finally, evaluate the importance of being **disciplined** in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.

Milestones

Milestone One: *JUnit Tests Code*

In **Module Four**, you will submit the code for your JUnit tests for the final project software application. **This milestone will be graded with the Milestone One Rubric.**

Milestone Two: *Summary of JUnit Testing*

In **Module Five**, you will draft the summary section of your final project documenting the results of your software testing in Milestone One. **This milestone will be graded with the Milestone Two Rubric.**

Final Submission: *Summary and Reflections Report*

In **Module Seven**, you will submit your final project. It should be a complete, polished artifact containing **all** of the critical elements of the prompt. It should reflect the incorporation of feedback gained throughout the course. **This submission will be graded with the Final Project Rubric.**

Final Project Rubric

Guidelines for Submission: Your summary and reflections paper should be 3 to 4 pages in length with double spacing and 12-point Times New Roman font. Any citations should be in APA format.

Critical Elements	Exemplary (100%)	Proficient (85%)	Needs Improvement (55%)	Not Evident (0%)	Value
Summary: Level Approach	Meets "Proficient" criteria and demonstrates shrewd ability to analyze requirements and make balanced software testing judgments	Defends the level of testing for each of the features using specific examples of testing best practices discussed in the course	Defends the level of testing, but fails to fully or appropriately explain the level using specific examples of testing best practices discussed in the course	Does not defend the level of testing for each of the features	6.5
Summary: Organizational Approach	Meets "Proficient" criteria and demonstrates shrewd ability to analyze requirements and make balanced software testing judgments	Defends the organization of tests for each of the features using specific examples of testing best practices discussed in the course	Defends the organization of tests, but fails to fully or appropriately use examples of testing best practices discussed in the course	Does not defend the organization of tests for each of the features	6.5
Summary: Alignment to Requirements	Meets "Proficient" criteria and demonstrates shrewd ability to analyze requirements and make balanced software testing judgments	Defends the alignment of the testing approach to the software requirements with specific examples from the test plan	Defends the alignment of the testing approach to the software requirements, but fails to fully or appropriately explain using specific examples from the test plan	Does not defend the alignment of the testing approach to the software requirements	6.5
Summary: Technically Sound Code	Meets "Proficient" criteria and demonstrates sophisticated programming abilities or develops particularly elegant code	Describes test writing experience, including strategies used to ensure technically sound code, and illustrates each with specific examples from the code	Describes test writing experience, but fails to include reasonable strategies for ensuring technically sound code, or does not illustrate each with specific examples from the code	Does not describe the test writing experience in terms of how the code was written to be technically sound	6.5
Summary: Efficient Code	Meets "Proficient" criteria and demonstrates sophisticated programming abilities or develops particularly elegant code	Describes test writing experience, including strategies used to make the code efficient, and illustrates each with specific examples from the code	Describes test writing experience, but fails to include reasonable strategies for making the code efficient, or does not illustrate each with specific examples from the code	Does not describe the test writing experience in terms of how the code was written to be efficient	6.5
Summary: Effective Tests	Meets "Proficient" criteria and demonstrates sophisticated programming abilities or develops particularly elegant code	Defends the effectiveness of the tests by citing at least five identified errors and describing the nature of each	Defends the effectiveness of the tests, but fails to cite at least five identified errors and describe the nature of each	Does not defend the effectiveness of the tests	6.5

Reflection: Techniques Employed	Meets "Proficient" criteria and demonstrates nuanced understanding of various software testing techniques	Describes the characteristics of the testing techniques employed in the project using specific details	Describes the characteristics of the testing techniques employed in the project, but fails to accurately address each technique using specific details	Does not describe the characteristics of the testing techniques employed in the project	5.5
Reflection: Other Techniques	Meets "Proficient" criteria and demonstrates nuanced understanding of various software testing techniques	Describes the characteristics of the other testing techniques discussed in the course using specific details	Describes the characteristics of the other testing techniques discussed in the course, but fails to accurately address each technique using specific details	Does not describe the characteristics of the other testing techniques discussed in the course	5.5
Reflection: Uses and Implications of Techniques	Meets "Proficient" criteria and demonstrates nuanced understanding of various software testing techniques	Explains the practical uses and implications of each of the techniques using specific examples of different software development projects and situations	Explains the practical uses and implications of the techniques, but fails to fully or accurately address each using specific examples of different software development projects and situations	Does not explain the practical uses and implications of the techniques	8.5
Reflection: Importance of Managing Software Testing	Meets "Proficient" criteria and demonstrates keen insight into best practices for managing software engineering projects to ensure continuous quality improvement over the life cycle	Assesses the importance of properly managing software testing by identifying the risks of not verifying and validating code using specific evidence and principles discussed in the course	Assesses the importance of properly managing software testing, but fails to fully or accurately identify the risks of not verifying and validating code using specific evidence and principles discussed in the course	Does not assess the importance of properly managing software testing	6.5
Reflection: Best Practices of Managing Software Testing	Meets "Proficient" criteria and demonstrates keen insight into best practices for managing software engineering projects to ensure continuous quality improvement over the life cycle	Analyzes the major components of administrating testing by citing specific best practices regarding testing strategy, planning, and monitoring	Analyzes the major components of administrating testing, but fails to fully or accurately detail each using specific best practices regarding testing strategy, planning, and monitoring	Does not analyze the major components of administrating testing	6.5

Reflection: Software Testing Tools and Technology	Meets "Proficient" criteria and demonstrates keen insight into best practices for managing software engineering projects to ensure continuous quality improvement over the life cycle	Identifies types of tools and technology available for implementing testing strategies and illustrates response with specific examples and principles discussed in the course	Identifies tools and technology available for implementing testing strategies, but fails to fully or accurately illustrate response with specific examples and principles discussed in the course	Does not identify tools and technology available for implementing testing strategies	6.5
Reflection: Caution	Meets "Proficient" criteria and demonstrates deep appreciation for the value of maintaining a disciplined, quality-focused mindset as a software engineering professional	Assesses the use and importance of employing caution when testing code using specific supporting examples	Assesses the use and importance of employing caution when testing code, but fails to fully or logically defend claims using specific supporting examples	Does not assess the use and importance of employing caution when testing code	6.5
Reflection: Bias	Meets "Proficient" criteria and demonstrates deep appreciation for the value of maintaining a disciplined, quality-focused mindset as a software engineering professional	Assesses the use and importance of limiting bias when testing code using specific supporting examples	Assesses the use and importance of limiting bias when testing code, but fails to fully or logically defend claims using specific supporting examples	Does not assess the use and importance of limiting bias when testing code	6.5
Reflection: Discipline	Meets "Proficient" criteria and demonstrates deep appreciation for the value of maintaining a disciplined, quality-focused mindset as a software engineering professional	Evaluates the importance of being disciplined when developing code using specific supporting examples	Evaluates the importance of being disciplined when developing code, but fails to fully or logically defend claims using specific supporting examples	Does not evaluate the importance of being disciplined when developing code	6.5
Articulation of Response	Submission is free of errors related to citations, grammar, spelling, syntax, and organization and is presented in a professional and easy-to-read format	Submission has no major errors related to citations, grammar, spelling, syntax, or organization	Submission has major errors related to citations, grammar, spelling, syntax, or organization that negatively impact readability and articulation of main ideas	Submission has critical errors related to citations, grammar, spelling, syntax, or organization that prevent understanding of ideas	2.5
				Total	**100%**