Usability Test Plan

Andrea Lee
Team 6: Code Critters

Test Scenario 1: Coding Within Application

Test Goals

1. Test text input functionality of app interface
2. Test antipattern detection function
3. Feedback from critiquer should allow for errors in the code to be corrected

Required Software/Equipment

- Zoom video conference software (tester and test giver)
- desktop/laptop computer (tester and test giver)
- Keyboard and mouse
- Microphone
- webcam
- Internet access
- Code critiquer app
- Timer (only for test giver)

Description

The usability tester will be prompted to write their own program within the user interface. As testers are sampled from computer science classes, they will be expected to be capable of programming with java. Testers will be encouraged to make the program at least 10 lines long, excluding any blank lines. Allowing testers to independently generate code to be tested will allow for a wider variety of antipatterns to be detected within the same scenario. However, testers may still ask the attending consultant for possible code to test.

Once the program is complete the tester will be prompted to check the program for antipatterns. If any antipatterns are detected the tester must attempt to correct these errors and check the code again. This process will repeat until no errors are present.

Scenario Text

“This program uses two methods of input, file uploads and the text box in the center of the screen. We will be testing the text entry first. Imagine that you are just writing a program for yourself as practice. While the program can be simple, it should be at least 10 lines long. If you need help thinking of a program I can give some suggestions.Use the ‘check code’ function to find any errors in your code.”

Measurement List

- Time to complete tasks – Slow response times from testers may indicate problems with app navigation.

- Error detection – the code critiquer should automatically detect any errors within the code
- System understandability – users must understand the functions of each button within the app interface
- Error correction – users are capable of correcting their code when errors are detected

Potential Observations

- There is a chance that the code could be completed without errors on their first attempt, this may require prompting the tester to knowingly add an error to their code.
- When an antipattern is detected within the code, the tester's next actions will be informed by their interpretation of the error message. A misinterpretation will lead to a lack of error correction.
- An error may also be undetected by the critiquer itself, this will require a bug report.

Bug report form

See Bug report form

Post Test Questionnaire

See "post test questions" on test questionnaire form

Post Test Interview

1. Were there any elements in the interface you did not understand?
2. Did the format of the critique help you understand where and how an error occurred?
3. Were any of the antipattern messages you encountered unclear?
4. Were there any messages that you understood but did not know how to correct?

Test Setup

The tester will begin a zoom call with the consultant and any attending programmers. They will be given access to the Code Critiquer application as well as permission to share their screen. The tester's microphone will need to be active to maintain communication with the consultant and programmers.

Test Scenario 2: File Upload

Test Goals

1. Test file upload function of code critiquer
2. Test sign in function of code critiquer
3. Correct errors of an existing program through the code critiquer interface
4. Test the “view previous critiques” function of the code critiquer

Required Software/Equipment

- Zoom video conference software (tester and test giver)
- desktop/laptop computer (tester and test giver)
- Keyboard and mouse
- Microphone
- webcam
- Internet access
- Code critiquer app
- Timer (only for test giver)
- Sample program file

Description

The usability tester will be required to download a sample program file provided by the usability consultant. Once the file has been acquired, the tester will need to upload it into the application and check it for antipatterns. As this code was created by the researchers, it will have the same errors each time. Like in the previous scenario, the tester will need to correct and recheck the program until no errors are present.

Once no errors are present, the tester will be instructed to view the past critiques saved. The tester must use this function to view the first critique of the provided program file.

Scenario Text

“In addition to typing directly into the app, the code critiquer allows for files to be uploaded for critique as well. By logging in, you will also gain access to previous versions of your code. For this scenario, imagine that the file I gave to you is a homework assignment that you want to check for errors before turning in. Log into the Code Critiquer and correct any errors it finds. Once the code is free of errors, go through the past critiques to find the original version of the file to compare it to the final version.”

Measurement List

- Time to complete tasks – Slow response times from testers may indicate problems with app navigation.

- Error detection – the code critiquer should automatically detect any errors within the code
- System understandability – users must understand the functions of each button within the app interface
- Error correction – users are capable of correcting their code when errors are detected

Potential Observations

- Time spent searching for functions will need to be tracked by the consultant. As this scenario is more structured than scenario one, response times can be compared between participants.
- Misinterpretation of error messages will lead to their errors remaining uncorrected.
- Errors may also be uncorrected due to a lack of user knowledge.
- Repeated searches through past critiques would indicate that the naming scheme used by the program is unclear

Bug report form

See bug report form

Post Test Questionnaire

See “post test questions” on test questionnaire form

Post Test Interview

1. Were any functions difficult to locate?
2. Were the names of the previous versions of code easy to understand?
3. Did the error messages help you understand the flaws in the program?
4. What part of this scenario did you consider to be the most difficult?

Test Setup

Using the same zoom call from scenario one, the consultant will provide the tester with a .java file containing a sample program to test. As this scenario requires viewing past critiques, login info would need to be provided to the tester as well.