NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing and Cyber Systems

Software Testing Plan

4/5/2024

Team IQ

Project Sponsor: Erika Konrad

Faculty Mentor: Igor Steinmacher

Version: 1.1

Team Members:

Logan Samstag (Team Lead)

Nicholas Persley

Kristiana Kirk

Elian Zamora-Rivera

Robin Pace

NAU NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing and Cyber Systems

Table of Contents

Introduction

The project that our team has been working on is a product known as RedPen, which is a web based application that allows for aspiring journalists on NAU's campus to take quizzes that test their knowledge about concepts covered in their journalism courses. Our primary goal is to assist our client in making sure that her students understand the concepts that are being covered in the course, as the current testing materials that she is using have many problems, one of which being that the content that is covered in the assessments do not fully represent what is being taught in the class.

Before we discuss our plan for testing, it is important to cover what testing is. Software testing is the act of testing the product that you are developing in a number of ways. This could be testing individual functions, how the functions work together, or how certain pieces of the end product work from the user point of view. This is important as it allows for certain edge cases to be detected and allows for these edge cases to be addressed during development. Overall, the way that Software Testing works is by identifying certain cases that could be a problem, and then using some sort of framework or library in order to test if that case is a problem. For example, if you anticipate that a function is not going to work with a certain input, you write a test that tests that input against the output that you expect.

Our plan for testing our product is to have each team member test a different portion of the code than what they developed initially throughout the project. For example, someone who worked on the quiz display for the student will instead test something on the professor side of RedPen. In terms of the amount of testing that we will end up doing for RedPen, we plan on testing around 50% of our functions on the backend side of things, as well as using libraries to test integration and usability. We decided to do

50% testing of our functions as not all of them have an easy way that we have found to test them. For example, our function that renders the string for the user to click on does not have a very good way to unit test it. Thus, we will unit test the pieces around it to make sure that the function works, and we believe that testing 50% of functions is enough to make sure all of the functions of RedPen are working. Additionally, we will focus less on testing integration and usability wise as with our system design the modules should fit together nicely, and thus they require less testing to make sure they are functional. Thus, in the following sections, we will discuss more in detail our Unit Testing, Integration Testing, and Usability Testing.

Unit Testing

Background

Before discussing how unit tests will work for RedPen, we must first introduce what the idea is, which is in essence testing individual functions/methods/procedures (units) in which the program is made up of, all to make sure they function correctly under typical and, less common, atypical conditions, overall confirming that the program at the lowest level works.

Details

For unit testing, the two frameworks that have been chosen are Jest and PHPUnit, where Jest is to test JavaScript and PHPUnit is to test PHP. It is worth mentioning that the other consideration from Jest was Mocha. The reason to which Mocha was not chosen boiled down to the ease of setup/use, given that overall, they are extremely similar in their abilities when it comes to unit testing, plus Mocha requires

Node.js to run. Our target testing metrics are two different types, test coverage and the percentage of passed tests. Where the goal test coverage will be **50%** of the code depending on how difficult testing certain units end up being. The reasons why we will test 50% of the codebase is development time and complexity of testing certain bits of code. For example, to test some JavaScript functions that use ajax from jQuery, we must set up an environment through a browser and test that ajax pulls correct information. Moving onto percentage of passed tests, the goal here is to have **100%** of the tests pass no matter how niche a scenario may be, since the idea of testing is to make sure nothing will break when in use. Most tests are to be run on GitHub Actions to automate the process of running tests on a designated environment. If any tests cannot be run on GitHub Actions due to something like the need to pull data from a database, the test may be run locally/on a browser (i.e., Safari, FireFox, Chrome, etc.) to allow for dynamic actions/data. Moving on to the details of the unit tests, the modules that we plan on testing are as follows

- student.js
- student_result_page.js
- Any PHP file that is not directly called by the browser, i.e., only scripts that are called by JavaScript

For testing the Student.js file, we plan on testing every function that has some sort of return value that we are able to test. For example, we plan on testing the *updateGiveUpAttempt* function by checking the local copy of the attempts that the code writes to in order to render the correct attempt number live, as well as checking the back-end side of things on the database by checking the value in the database against the value we expect to be in the database. For example, we plan on having a test case where one of us has our ID temporarily used for testing purposes, and we will run the test on one of the questions on one of the quizzes to see if the database is updated to a negative value and the array on the Javascript side is also updated to reflect the change. For the equivalence partition of this function, any value that is less than 0 is acceptable, as that implies that the database and local copy have both received the value that signifies a give up attempt, while any value that is above 0 is not acceptable, as that implies that the database and local copy have not received the sign that the user has given up, which will cause errors for both the Javascript side, as the user can keep inputting answers, and the database side, as the database will think the user has not completed the question. Another function that we plan on testing is the *isCorrectAnswer* function, and we plan on testing it by giving the function a mock question and a mock answer position and having the function report back to us if the word is the correct answer. We plan on writing 2 tests for this function, one test that should return that the word clicked was the right answer, while the second function should return that the answer was the correct answer. Any other results for these 2 tests aside from the desired result is an invalid result. For testing with erroneous inputs, we plan on testing functions mainly with parameters outside of their bounds. For example, we plan on testing the *isCorrectAnswer* function with a question index that is outside the bounds of the question to see if the code still functions with a bad input.

For testing the *student_result_page.js*, we plan on testing the results by giving the function to pull the data from the database a test user, making sure that test user has data for the desired quiz that we wish to pull data from, and then checking the results once the data has been given back to the Javascript file. The only results that we will accept are the exact results that we have in the database, as any other results imply an error within the code.

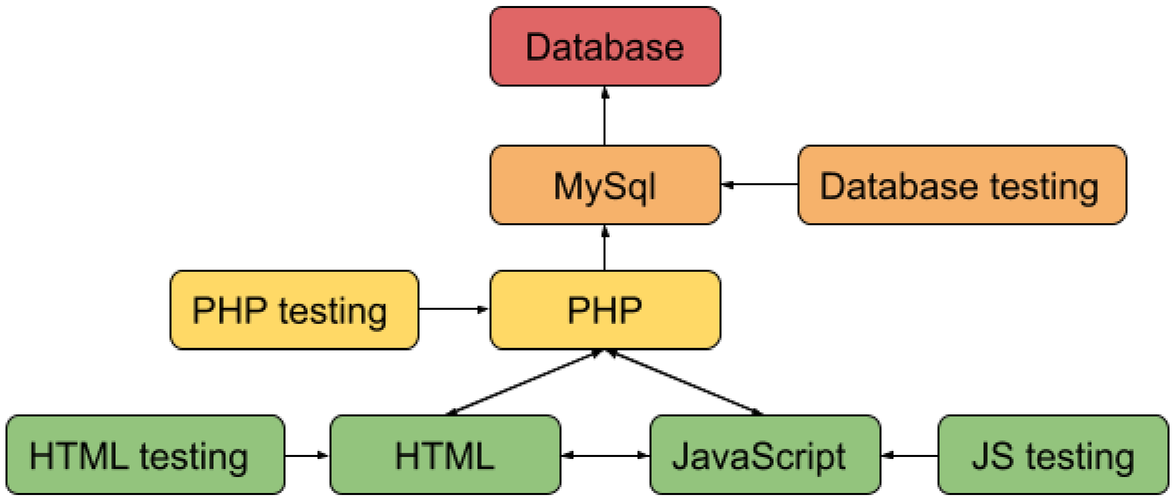
For testing all of our php files, we plan on once again testing all of the php files that have functions that give some sort of output. A lot of the php files that our team currently has on our website are files that simply display pages, such as *professor.php*. There is no unit for us to specifically test in this file, so these files will be excluded. The files that we mainly want to make sure work are all of the files that read or write to the database, as if they do not read or write correctly the entire site will not be functional. As an example of the functions/scripts that we are going to test, we can look at the *student_quiz.php* file, which reads all of the values that we need for a student to take a quiz after it is selected on the homepage. The file reads in the students' attempts, the questions that have been completed, the positions of the answers, and the attempt number of the quiz the student is on (e.g., the student attempted the quiz 4 times so far, so they start a 5th). As a quick side note, these are ways of cutting down inefficiencies when it comes to querying the database many times. The way that we plan on unit testing this file is by doing a mock reading from the database and seeing if the results are what we expect them to be. For example, one of us will use our student id for temporary testing, we will call the file to read the values from the DB, and we will then call PHPUnit to test to see if the values that we got from the DB were what we expected to get. This same procedure will occur for our writing operations as well, with the test calling the write, doing a read from the DB, and using PHPUnit to see if the read returned the value that was expected. The only values that we are going to accept are values that entirely agree between the php and the database. There can be no error writing into the database or else nothing on RedPen will work. We plan to also attempt to write values that would not normally appear on RedPen, such as a very high integer for attempts, to make sure that the database does not break when written to with an erroneous input.

With all of this said, the final thing that should be discussed is how the database will be designed for tests. This will result in having a test table to contain mock student results. This will most likely be a fake semester table that will never be accessed by any student that is not in the table *students_Fall2023*, that will be a fake student (e.g., *eky0*). So when the tests are run, we should access this table instead of the current one. Then as for when we check for alumni, we can query the script (yet to be implemented) with the test username and return the results of if the user is an alumni to the class, i.e., we can pull the correct semester information.

Integration Testing

Integration testing tests the interactions between software modules to ensure they work as intended. Integration testing goes deeper than usability testing to ensure that while it may look like it works from a user perspective, we also ensure that it does what we're expecting from a backend perspective. On top of that, when integration testing, we can focus on subsystems that connect with one or multiple other systems. When testing this way, we can test that the module being tested works with all the connected systems at once, as well as with each part individually. This is particularly important for our project because we are using multiple languages that don't always interact in easy or expected ways. Being able to break the tests down enough to test both how the module works on its own and how it works with modules in a different language is essential. Integration testing is especially beneficial when multiple programmers are working together. This is because, while the style, structure, and format they use may be different, a lot of the modules still have to be able to work together. If there's ever any way that the two styles won't connect, integration testing can find it.

Our approach for integration testing will be to start at the bottom and work up, meaning checking that the lowest level system worked before checking the next level up, and so on. This was decidedly best for us because we have a lot of interconnecting lower subsystems, and very few high-level systems. If an error is found, we can reliably assume that the cause is from the level that is currently being tested, and when we move on we can be confident that the level is working as it should. In our case, this will start with checking the program's ability to send or retrieve data to and from the database. To do this, we will create test functions that test specific PHP operations and functions. These functions will often be hard coded to match the database so that the check can simply be true or false. Either it got the right data, or it didn't. This will check both values that may have been sent to the functions and the values that it would return. Once every result of this has been tested, we can then move on to the next level. In some cases, this will involve JavaScript operations, but in most, this involves HTML. In the cases of JavaScript operations, it will be fairly easy to test. First, those systems will be tested individually with simple and potential edge case data points, in some cases this will require a test table that can be freely altered with no issue. A function that passes in specific requirements will be made, and if there is a return, once again it will be hard-coded to check the desired results. For the HTML connections, we will be using Python to test the interactable elements. This sort of test will be fairly straightforward in most cases, as the element either exists or doesn't, and the element does what it's supposed to, or it doesn't. The HTML side of our website isn't overly complicated most of the time. The more complicated HTML comes from the student's results, and in those scenarios, the data comes from Javascript and PHP files that should be tested at the level before. On top of that, the HTML will be getting most of its checks from the usability and unit tests. Because of that, from the integration testing side of things, we should only have to worry about the functions they're connected to, sending off the right data.



If at any point errors or even concerning data is found, it will be marked thoroughly with important details and perhaps even screenshots to be checked later. Once all the other systems of that level are finished being tested, then the tester will return to the data that was marked for correction and further testing. Once the concerns have been resolved and the corrections made, the whole level is to be tested again to ensure that the changes didn't alter anything else. Once no errors are found the tester is allowed to proceed to the next level. This has been established this way to ensure confidence in the previous level tested. If we marked an error but didn't fix it until after everything had been tested, we would not be able to reliably evaluate the results of the level above and the following testing would end up taking more time. In the case that an error isn't found until testing the next level above, it is made clear that the tests for that level are not thorough and both the system and the test for the system should be altered immediately.

Usability Testing

Usability testing is a crucial phase in the development of any software or website, including our online quiz editing website, “Red Pen”. It focuses on evaluating the user experience by observing how users interact with the system. The primary goals of usability testing are to identify usability issues, gather feedback on the user interface, and ensure that the system meets the needs and expectations of the target audience. Given the nature of our product, which is an online quiz editing website designed for higher-level English classes, several considerations have influenced our approach to usability testing:

Target Audience: Our online quiz editing website is specifically designed for a niche audience comprising online professional writing graduate students and technical writing undergraduate students at NAU. These individuals are characterized by their commitment to honing their writing and editing skills to excel in their academic and professional pursuits. As such, our platform serves a dual purpose: to assess their existing knowledge and to impart new insights and techniques relevant to the field of professional writing and editing. Given the specialized nature of our target audience, it's crucial that our website caters to their unique needs and expectations. These students are likely to possess a strong foundation in writing and editing principles, but they may vary in their familiarity with digital learning tools. Therefore, the website must strike a balance between sophistication and accessibility, offering advanced features while ensuring ease of use for all users. Furthermore, our target audience consists of individuals who are preparing for careers as professional editors and writers. As such, the quizzes on our platform are not merely assessments but also learning opportunities. Each quiz is designed to impart practical skills and knowledge that are directly applicable to the real-world tasks and challenges encountered by professional editors on a daily basis.

Consequences of Bad Design: The implications of poor design in our online quiz editing website extend beyond inconvenience; they can hinder students' ability to effectively practice and apply the skills they are learning. Inaccurate feedback or confusing instructions may lead to misunderstandings and frustration, undermining the educational value of the quizzes. For students aspiring to become professional editors, the consequences of bad design are particularly significant. A poorly designed interface can erode confidence in the accuracy and reliability of the platform, undermining its credibility as a tool for skill development and practice. Moreover, it may discourage students from fully engaging with the material, limiting their opportunities for growth and improvement.

Novelty of Product: Our online quiz editing website takes a fresh approach to learning and skill-building in the professional writing and editing realm. Instead of just testing what you know, our quizzes also teach you new concepts and techniques. This makes it a one-of-a-kind tool for students who want to boost their editing skills. Unlike typical quiz platforms, our website is made specifically for aspiring professional editors and writers. Each quiz is designed to mimic real editing situations, giving students a chance to practice skills they'll actually use in their future careers. This helps bridge the gap between theory and real-world application, giving students the practical know-how they need to stand out in the competitive world of professional writing and editing. In short, we've tailored our platform to meet the unique needs of our users, ensuring it's both user-friendly and packed with valuable learning opportunities. With our innovative approach, we're set to revolutionize the way students learn and grow as editors and writers.

Testing Plan: The primary objective for this usability testing plan is to have a thorough evaluation of the user experience with Red Pen, we want to make sure that it meets the needs of our sponsor as well as the students using the online application.

We have decided to ask Dr Konrad and possibly some of her students to test the pages that we have created and possibly give us some feedback on what they like and what improvements could be made with these pages. We have decided that Dr. Konrad would test out the admin side of our website as she will most likely be the only one using the admin pages and because no one else would have access to those pages except herself. Some of her students are going to be testing out the student side of the website and we will ask them to give us as much feedback as possible about what they like and the things our site could improve on. If Dr. Konrad is not able to get any of her students for testing. We will be asking some of our fellow classmates if they can take some time to help us test the site and give us feedback. The main goal for this testing is to ensure everything is working as expected. The person testing for the professor side would want to make sure that they're able to successfully create quizzes and publish them for students to see, as well edit them and delete them. The tester should also be able to view the stats for the students when a test has been completed as well as view a graph that summarizes some of that information. On the student side, We want to be able to take the quizzes created by the professor, view progress of the quiz and see the quiz scores as well. To finalize our testing. We want to make sure that all of the pages can be used efficiently. We want to make sure each button on all pages does what it's supposed to do and everything is displayed in a properly formatted way. We also want to make sure we can navigate to any page in the website without any problems and there are no incidents where the site could possibly crash.

Conclusion

After discussing all of our plans for testing, including how we plan to do Unit Tests, Integration Tests, and Usability Testing, we believe that we will be able to make RedPen into a product with minimal bugs. First, with our Unit Testing, we will be able to validate the correctness of many of the units that comprise RedPen and let it function. Second, with our Integration Testing, we will make sure that all of our individual modules work together and give the results that are expected. Finally, with our Usability Testing, we will make sure that RedPen is the best product that it can be for the users. That is our plan for how we plan to test RedPen at every level.