**City University of Seattle**

**CS 504 - Software Engineering Summer 2021**

**Independent Project 3**

**Vaccine Scheduler**

**Anil Erturk**

**Introduction**

Since developers divide projects such as the vaccine scheduler into smaller modules and assign inputs and outputs to these modules, developers can develop test cases to see how the software handles correct and incorrect inputs. Depending on the resources spent on testing, this will give the developers a certain level of understanding and confidence in the software.

The goal of testing is never to make software bug-free. Software projects have gotten too big, and the platforms they interact with have gotten so diverse that, thinking about every single possibility from the start is very hard.

Instead, the goal of testing changed to putting the software through a systematic check to ensure a certain level of quality. Different levels of tests test the software at different stages. Unit tests enable testing to start immediately, and without all the infinite variables and environments that need to be dealt with. This test gives the developers a certain level of understanding of the capabilities of the units. If the developers require higher-level confidence in their software, they can spend more resources on testing.

Later, integration tests enable testing all the units together. This gives the developers a certain level of understanding of how the entire software works together.

And lastly, regression testing allows the developers to ensure that the quality of their software is still at least at a certain level after making changes to it. This is important because software is prone to creating new bugs when changes are made.

**Test Plan**

**Test objectives**

The vaccine scheduler web app is a dynamic website. The common attacks when it comes to web applications are known and the defenses against them are also in place. The test plan is to make sure the components of the web application have their security and safety problems covered to make it robust against external and internal problems.

The software needs to be capable of progressing the user to the next stage when the input is correct as well as warning the user and not letting the user to the next stage when the input is incorrect. The testing will reveal whether the software is capable of handling the correct and incorrect inputs in the test cases.

Testing will check the functional and nonfunctional requirements of the project. Functional requirements will commonly include user stories. These are the services the user needs to be served. Nonfunctional requirements commonly include protecting from cyberattacks such as SQL and Javascript injecting (OWASP, n.d. c).

**Test Methodology**

Since the project is going to be divided first into three by the MVC architecture, and later into further files, classes, and methods, the unit testing can test each module to see whether they can handle their own small workload in the first place. For example, the controllers will need to have methods that validate user input to see if the values are acceptable or not.

After the modules are developed, the entire project will need to be tested for requirements using system tests.

Regression testing will allow developers to run their tests after making changes to the software to ensure the quality stays above a certain level.

**Test environment**

Test classes containing test cases (test methods) will be prepared. Before the implementation of test methods, the structure and requirements for the software need to be decided.

IDEs such as Netbeans allow for creating of web applications and test classes for the controller classes of the project. The developers can add an arbitrary number of test methods to the methods of classes of the project.

The entire goal of testing is to ensure a certain level of quality by looking for bugs in the software. So, when the bugs are found, the developers are going to need to debug the software in order to understand the exact problem. Netbeans IDE allows debugging of web applications just like console applications. This allows the developers to understand and fix the problem.

**Test Cases**

One set of test cases will need to execute user stories with correct inputs to see whether functional requirements are met. These test cases will include correct input values and choosing to interact with the correct sequences of buttons and other components. Test tools such as Selenium help with automating web application tests (BrowserStack, 2021). One scenario can be COVID-19 vaccine, correct zip code of the user, selection of the vaccine type the user wants, selection of the store, confirming that the user is above 12 years old and then proceeding, selecting the correct dose, selecting the date and time the user will show up, entering correct user information such as name and address with no malicious inputs.

Another set of test cases will need to execute user stories with incorrect inputs to see whether the nonfunctional requirements are met. Handling of incorrect inputs allows for a more secure and robust system that can recover from external and internal issues. One test case will be timing out users after certain minutes. Another test case will be verifying user inputs such as birth date. Another test case will test SQL, and Javascript injection attacks. Another test case will validate user inputs for names to be in a certain format and length.

Since this is a dynamic web application with text inputs from the user, the inputs need to be escaped in order to prevent special characters from changing our SQL statement.

Prepared statements is another security feature that forces the entire user input string to be taken literally. So if the user enters “tom' or '1'='1”, this whole string will be treated as the username (OWASP, n.d. a).

When showing data inputted by the user on the website, it needs to be sanitized to prevent the user from adding javascript scripts to the website. This attack can be used to steal information of users just like SQL injection. (OWASP, n.d. b).

**References**

OWASP. (n.d. a) SQL Injection Prevention Cheat Sheet.

https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.html

OWASP. (n.d. b) Cross Site Scripting Prevention Cheat Sheet.

https://cheatsheetseries.owasp.org/cheatsheets/Cross\_Site\_Scripting\_Prevention\_Cheat\_Sheet.html

OWASP. (n.d. c) Injection Prevention Cheat Sheet in Java.

https://cheatsheetseries.owasp.org/cheatsheets/Injection\_Prevention\_in\_Java\_Cheat\_Sheet.html#injection-prevention-cheat-sheet-in-java

BrowserStack. (2021). Selenium with Java : Getting Started to Run Automated Tests.

https://www.browserstack.com/guide/selenium-with-java-for-automated-test