

CSE 565 - Software Verification Validation and Testing Project Portfolio

Kedar Sai Nadh Reddy Kanchi

School of Computing and Augmented Intelligence
Arizona State University, Tempe
kkanchi@asu.edu

Assignment 1

The software development process is incomplete without testing. Unit testing is a vital aspect of software testing that is used to ensure code quality. Unit testing frameworks are tools that help developers to test their code automatically and efficiently. The goal of this academic project is to provide an in-depth analysis and evaluation of unit testing frameworks in software development.

The project is divided into two parts. The first part of the project involves providing an overview of what a unit testing framework is, its scope, purpose, and an example of usage with a use case. The overview will also cover how a developer can use a framework and the benefits it provides. Furthermore, the overview will include a comparison of two frameworks in terms of common capabilities as well as any differences for a language such as Java, C++, Python, and JavaScript.

The second part of the project entails implementing the Heap-sort algorithm in Java, C++, Python, or JavaScript and testing the code with a unit testing framework of choice. The report will include a copy of the code, unit testing framework test cases, and an output report in the form of a screenshot from the testing tool.

Work Completed

To begin the project, I thoroughly investigated unit testing frameworks by performing a web search on "What is a unit testing framework?" and reading through multiple articles like "Unit testing" [1] & "Unit Test Frameworks by Paul Hamill" [2] to gain a comprehensive understanding of the topic. Through this research, I collected valuable information which allowed me to provide an in-depth analysis and evaluation of the scope, purpose, and usage of unit testing frameworks. I also provided an example of usage with a use case.

Continuing with the project, I conducted further research by performing a web search on "How would a developer utilize a unit testing framework?" and read through multiple articles like "What is Unit Testing and Why Developer Should Learn It"[3] & "Why Is Unit Testing Important in Software Development?" [4] to understand the Unit Testing Life Cycle to understand how developers utilize unit testing frameworks, and why they are widely used in software development. By singling out relevant information from the articles, I could comprehensively discuss how a developer

would utilize a unit testing framework and its benefits.

For the last part of the project's first section, I compared and contrasted two unit testing frameworks, JUnit and Jest, that are used extensively with Java and JavaScript. To accomplish this, I performed additional web searches for JUnit and Jest, analyzed various articles like "Jest Framework" [5], "JUnit Tutorial — Testing Framework for Java" [6] & "What are the differences between Jest and JUnit?" [7], and identified the required information. I then documented the similarities and differences between the two frameworks regarding their capabilities.

To begin the second part of the project, I chose JavaScript as the language to implement the Heap Sort Algorithm and began searching for the algorithm online. I found a JavaScript implementation of the Heap Sort Algorithm online[8] (code attached in the appendix) and used it to implement the algorithm for this project. I then tested the Heap Sort code with the Jest unit testing framework suitable for JavaScript.

Because I was unfamiliar with Jest, I relied on a YouTube tutorial[9] and the official Jest documentation[10] to gain a thorough understanding of creating test cases with this framework. Once I was confident in my ability to generate test cases, I proceeded to generate the necessary test cases (code attached in the appendix) to ensure the functionality of the Heap Sort code. I ran the test cases through Jest after creating them, and the output report confirmed the code's correct operation.

Results

```
PS-DIVUSI-M:CS-COURSE-CONTENT-AND-ASSESSMENTS-COURSES-CENTER-VAPRTE-2023CSE-565-Software-Veri-  
fication-Validation-and-Testing-ASSIGNMENTS-ASSIGNMENT1-Code rpx_jest --coverage  
FAIL ./heapsort-test.js  
  heapsort test case 1 (4 ms)  
  heapsort test case 2 (1 ms)  
  heapsort test case 3 (1 ms)  
  heapsort test case 4 (1 ms)  
  heapsort test case 5 (1 ms)  
  heapsort test case 6 (1 ms)  
  heapsort test case 7 (1 ms)  
  heapsort test case 8 (1 ms)  
  heapsort test case 9 (1 ms)  
  heapsort test case 10 - checking to fail (4 ms)  
  ● heapsort test case 10 - checking to fail (fail)  
    expect(received).toEqual(expected) // deep equality  
      + Expected: 2  
      + Received: 1  
        + array:  
          - 1  
          - 2  
          - 3  
          + 4  
          + 5  
          + 6  
          + 7  
          + 8  
          + 9  
        + length: 10  
        + type: object  
      - array:  
        + 1  
        + 2  
        + 3  
        + 4  
        + 5  
        + 6  
        + 7  
        + 8  
        + 9  
        + length: 10  
        + type: object  
      + value:  
        + array:  
          - 1  
          - 2  
          - 3  
          + 4  
          + 5  
          + 6  
          + 7  
          + 8  
          + 9  
        + length: 10  
        + type: object  
      - value:  
        + array:  
          - 1  
          - 2  
          - 3  
          + 4  
          + 5  
          + 6  
          + 7  
          + 8  
          + 9  
        + length: 10  
        + type: object  
      + type:  
        + value:  
          + array:  
            - 1  
            - 2  
            - 3  
            + 4  
            + 5  
            + 6  
            + 7  
            + 8  
            + 9  
          + length: 10  
          + type: object  
        - value:  
          + array:  
            - 1  
            - 2  
            - 3  
            + 4  
            + 5  
            + 6  
            + 7  
            + 8  
            + 9  
          + length: 10  
          + type: object  
        + type:  
          + value:  
            + array:  
              - 1  
              - 2  
              - 3  
              + 4  
              + 5  
              + 6  
              + 7  
              + 8  
              + 9  
            + length: 10  
            + type: object  
          - value:  
            + array:  
              - 1  
              - 2  
              - 3  
              + 4  
              + 5  
              + 6  
              + 7  
              + 8  
              + 9  
            + length: 10  
            + type: object  
          + type:  
            + value:  
              + array:  
                - 1  
                - 2  
                - 3  
                + 4  
                + 5  
                + 6  
                + 7  
                + 8  
                + 9  
              + length: 10  
              + type: object  
            - value:  
              + array:  
                - 1  
                - 2  
                - 3  
                + 4  
                + 5  
                + 6  
                + 7  
                + 8  
                + 9  
              + length: 10  
              + type: object  
            + type:  
              + value:  
                + array:  
                  - 1  
                  - 2  
                  - 3  
                  + 4  
                  + 5  
                  + 6  
                  + 7  
                  + 8  
                  + 9  
                + length: 10  
                + type: object  
              - value:  
                + array:  
                  - 1  
                  - 2  
                  - 3  
                  + 4  
                  + 5  
                  + 6  
                  + 7  
                  + 8  
                  + 9  
                + length: 10  
                + type: object  
              + type:  
                + value:  
                  + array:  
                    - 1  
                    - 2  
                    - 3  
                    + 4  
                    + 5  
                    + 6  
                    + 7  
                    + 8  
                    + 9  
                  + length: 10  
                  + type: object  
                - value:  
                  + array:  
                    - 1  
                    - 2  
                    - 3  
                    + 4  
                    + 5  
                    + 6  
                    + 7  
                    + 8  
                    + 9  
                  + length: 10  
                  + type: object  
                + type:  
                  + value:  
                    + array:  
                      - 1  
                      - 2  
                      - 3  
                      + 4  
                      + 5  
                      + 6  
                      + 7  
                      + 8  
                      + 9  
                    + length: 10  
                    + type: object  
                  - value:  
                    + array:  
                      - 1  
                      - 2  
                      - 3  
                      + 4  
                      + 5  
                      + 6  
                      + 7  
                      + 8  
                      + 9  
                    + length: 10  
                    + type: object  
                  + type:  
                    + value:  
                      + array:  
                        - 1  
                        - 2  
                        - 3  
                        + 4  
                        + 5  
                        + 6  
                        + 7  
                        + 8  
                        + 9  
                      + length: 10  
                      + type: object  
                    - value:  
                      + array:  
                        - 1  
                        - 2  
                        - 3  
                        + 4  
                        + 5  
                        + 6  
                        + 7  
                        + 8  
                        + 9  
                      + length: 10  
                      + type: object  
                    + type:  
                      + value:  
                        + array:  
                          - 1  
                          - 2  
                          - 3  
                          + 4  
                          + 5  
                          + 6  
                          + 7  
                          + 8  
                          + 9  
                        + length: 10  
                        + type: object  
                      - value:  
                        + array:  
                          - 1  
                          - 2  
                          - 3  
                          + 4  
                          + 5  
                          + 6  
                          + 7  
                          + 8  
                          + 9  
                        + length: 10  
                        + type: object  
                      + type:  
                        + value:  
                          + array:  
                            - 1  
                            - 2  
                            - 3  
                            + 4  
                            + 5  
                            + 6  
                            + 7  
                            + 8  
                            + 9  
                          + length: 10  
                          + type: object  
                        - value:  
                          + array:  
                            - 1  
                            - 2  
                            - 3  
                            + 4  
                            + 5  
                            + 6  
                            + 7  
                            + 8  
                            + 9  
                          + length: 10  
                          + type: object  
                        + type:  
                          + value:  
                            + array:  
                              - 1  
                              - 2  
                              - 3  
                              + 4  
                              + 5  
                              + 6  
                              + 7  
                              + 8  
                              + 9  
                            + length: 10  
                            + type: object  
                          - value:  
                            + array:  
                              - 1  
                              - 2  
                              - 3  
                              + 4  
                              + 5  
                              + 6  
                              + 7  
                              + 8  
                              + 9  
                            + length: 10  
                            + type: object  
                          + type:  
                            + value:  
                              + array:  
                                - 1  
                                - 2  
                                - 3  
                                + 4  
                                + 5  
                                + 6  
                                + 7  
                                + 8  
                                + 9  
                              + length: 10  
                              + type: object  
                            - value:  
                              + array:  
                                - 1  
                                - 2  
                                - 3  
                                + 4  
                                + 5  
                                + 6  
                                + 7  
                                + 8  
                                + 9  
                              + length: 10  
                              + type: object  
                            + type:  
                              + value:  
                                + array:  
                                  - 1  
                                  - 2  
                                  - 3  
                                  + 4  
                                  + 5  
                                  + 6  
                                  + 7  
                                  + 8  
                                  + 9  
                                + length: 10  
                                + type: object  
                              - value:  
                                + array:  
                                  - 1  
                                  - 2  
                                  - 3  
                                  + 4  
                                  + 5  
                                  + 6  
                                  + 7  
                                  + 8  
                                  + 9  
                                + length: 10  
                                + type: object  
                              + type:  
                                + value:  
                                  + array:  
                                    - 1  
                                    - 2  
                                    - 3  
                                    + 4  
                                    + 5  
                                    + 6  
                                    + 7  
                                    + 8  
                                    + 9  
                                  + length: 10  
                                  + type: object  
                                - value:  
                                  + array:  
                                    - 1  
                                    - 2  
                                    - 3  
                                    + 4  
                                    + 5  
                                    + 6  
                                    + 7  
                                    + 8  
                                    + 9  
                                  + length: 10  
                                  + type: object  
                                + type:  
                                  + value:  
                                    + array:  
                                      - 1  
                                      - 2  
                                      - 3  
                                      + 4  
                                      + 5  
                                      + 6  
                                      + 7  
                                      + 8  
                                      + 9  
                                    + length: 10  
                                    + type: object  
                                  - value:  
                                    + array:  
                                      - 1  
                                      - 2  
                                      - 3  
                                      + 4  
                                      + 5  
                                      + 6  
                                      + 7  
                                      + 8  
                                      + 9  
                                    + length: 10  
                                    + type: object  

```

Figure 1: Jest Testing Coverage of the Heap Sort Algorithm

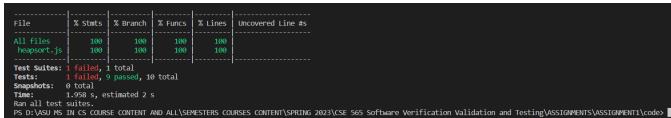


Figure 2: Total Jest Testing Coverage of the Heap Sort Algorithm

Lessons Learned

Throughout this project, I have comprehensively understood unit testing frameworks and their significance & purpose in software development. One important lesson I have learned is that unit testing frameworks can significantly enhance the software development process by automating the testing process, reducing the time and effort required to find and fix bugs, and increasing the overall quality of the code. Additionally, I have learned that understanding the unit testing life cycle is critical to utilizing unit testing frameworks effectively.

Furthermore, I have learned that it is vital to clearly understand the software development life cycle, specifically the unit testing life cycle, to utilize unit testing frameworks effectively. By understanding the unit testing life cycle, I could comprehend how developers utilize unit testing frameworks and their significance in software development.

I also learned that researching and comparing multiple frameworks can be time-consuming, but it is necessary to ensure that the best framework is selected for the project's requirements. In this project, I compared two-unit testing frameworks, JUnit and Jest, to determine their similarities and differences. This project has helped me understand the importance of selecting the appropriate testing framework for a specific programming language and project scope.

Lastly, generating test cases can be challenging, and it is essential to understand the framework's capabilities and features before beginning the testing process. This project has taught me to rely on documentation and tutorials to ensure that I thoroughly understand the testing framework before generating test cases.

Assignment 2

This project involves testing a database program that collects information and provides discounts for a product and its users based on various input factors. The project utilizes a pre-existing jar file containing the program with at least ten seeded defects. The goal of this project is to identify those ten seeded defects.

Testing aims to identify any flaws or bugs in the program and ensure that it performs its intended function correctly. Equivalence partitioning and boundary value analysis are commonly used in software testing to determine the most critical input parameter values. Furthermore, cause/effect analysis aids in the identification of relationships between inputs and outputs, allowing test cases to be refined further and the testing process to be more effective.

To achieve the objective of this project, we need to employ various software testing techniques such as equivalence partitioning, boundary value analysis, and cause/effect analy-

sis, which have been discussed in lectures. These techniques will help us develop a comprehensive set of test cases that can be used to identify the defects in the program. Once the test cases are developed, the tests will be executed according to the instructions to ensure that the program meets the required standards of functionality and reliability. By executing these tests, we aim to ensure that the program operates as intended and that all identified defects are effectively addressed.

Work Completed

In this project, I completed the first testing stage by using the Boundary Value Testing Technique to create test cases for three requirements: Username length, User Age, and Product Rating. This technique involves selecting test cases that explore the boundaries or limits of the input domain of a software application. The input domain refers to the range of valid and invalid input values the software application can accept. The objective of boundary value testing is to identify any defects or bugs that may exist at the boundaries of the input domain, as these are often the areas where defects are most likely to occur. To ensure that all possible scenarios were covered by using the Boundary Value Testing Technique exclusively, I created multiple test cases for each of the three requirements with different values that either met the requirement or were just above or below the required limits.

- **Requirement 1.1:** Username can be 5-10 characters long.
 - I created 4 different test cases with usernames of length 4, 5, 10, and 11.
- **Requirement 2.1:** User Age must be 18 years of age and above.
 - I created three different test cases with user ages of 16, 17 and 18.
- **Requirement 7.1:** Rating of Product must be between 1 – 10.
 - I created 4 different test cases with usernames of length 0, 1, 10, and 11.

I have completed the second testing stage using the Equivalence Partitioning Testing Technique. This technique involves dividing the input domain of a software application into groups of equivalent values called equivalence classes. This technique aims to reduce the number of test cases required to test the software application while ensuring that all possible scenarios are covered. Instead of testing each input value, only one representative value from each equivalence class is selected. For this project, I used the Equivalence Partitioning testing technique exclusively for the following requirements:

- **Requirement 1.2:** Username can only contain letters and no numbers.
 - I created two different test cases where one test case tests the username containing numbers, and another test case tests the username which does not contain numbers.

- **Requirement 1.3:** Username cannot contain a hyphen (-) or an underscore (_).
 - I created two test cases: one test case tests the username containing a hyphen (-), and the other tests the username containing an underscore (_).

So, till now, in this project, I have developed test cases using only the boundary value testing technique or the equivalence partitioning testing technique. However, software testing often uses these techniques to complement each other's strengths and weaknesses. This is because Boundary value testing effectively tests the boundaries of the input domain, while equivalence partitioning effectively reduces the number of test cases required by grouping inputs into equivalence classes. By using these techniques together, we can ensure that all input values within the input domain are tested, including those at the boundaries, and reduce the required test cases. So, following this learning, I used a combination of the Boundary Value and Equivalence Partitioning Testing Techniques to test three requirements: Username length, User Age, and Rating of the Product. By using these techniques together, I reduced the required test cases while ensuring that all input values within the input domain were tested.

- **Requirement 1.1:** Username can be 5-10 characters long.
 - I created 3 different test cases with usernames of length 4, 6, and 11.
- **Requirement 2.1:** User Age must be 18 years of age and above.
 - I created three different test cases with user ages of 16, 17 and 100.
- **Requirement 7.1:** Rating of Product must be between 1 – 10.
 - I created 2 different test cases with usernames of length 0 and 11.

In the final stage of this project, I utilized the Cause/Effect Analysis testing technique. Cause/effect analysis involves identifying potential causes of defects or problems by analyzing the relationships between different factors or variables in a system. This technique is based on the principle that every defect or problem in a system has underlying causes that can be identified by examining the relationships between the different factors contributing to the defect. In cause/effect analysis, the tester begins by identifying the defect or problem to be tested and then identifies the factors or variables that could potentially contribute to the defect. These factors include inputs, conditions, and other variables affecting the system's behavior. Therefore, as part of this project, I have exclusively used cause/effect analysis testing techniques for specific requirements best suited for this technique.

- **Requirement 9.1:** New users cannot be given any discounts regardless of reward status, product, and season.
 - I have identified and created 12 different test cases that could potentially contribute to the defect.

- **Requirement 9.2:** If product category is unknown, user gets no discount.
 - I have identified and created 12 different test cases that could potentially contribute to the defect.
- **For combination of Requirement 9.1 Requirement 9.2**
 - I have identified and created 12 different test cases that could potentially contribute to the defect.
- **Requirement 9.3.1:** If bought in Spring, then the returning users who are bronze members who bought Electronics get a 10% discount voucher.
- **Requirement 9.4.1:** If bought in Winter, then the returning users who are gold members who bought Electronics get a 25% discount voucher.
- **Requirement 9.5.1:** If bought in Summer, then the returning users who are gold members who bought Electronics get a 15% discount voucher.
- **Requirement 9.6.1:** If bought in Fall, then the returning users who are silver members who bought Electronics get a 15% discount voucher.
- **Requirement 9.7:** Any other combination of returning user, reward status, product, and season does not get a discount
 - I have identified and created 8 different test cases that could potentially contribute to the defect.

Results

Overall, I have developed a comprehensive set of 65 test cases using these testing techniques and identified the seeded defects. Combining these testing techniques helped me ensure that all possible scenarios were covered and reduced the required test cases.

The ten identified defects are as follows:

- The user name field is accepting a name which has more than 10 characters. This is invalid because as per the requirements the user name can only contain 5-10 characters.
 - **Requirement Mapping:** 1.1
 - **Observed Output:** The discount given is 15%
 - **Expected Output:** Username length is invalid
- The user name field is accepting the underscore character as a valid character. This should happen because as per the requirements the user name should not contain the underscore character.
 - **Requirement Mapping:** 1.3
 - **Observed Output:** The discount given is 15%
 - **Expected Output:** Username contains _
- The application gives an output even when the user enters the age as 17. This should not happen because as per the requirements the application should not calculate for the output i.e. in this case discount when the age of the user is less than 18.
 - **Requirement Mapping:** 2.1
 - **Observed Output:** The discount given is 15%

- **Expected Output:** You must be 18 or older to register in the database.
 - A rating of 0 which is less than 1 is accepted by the application and the discount is being calculated to be shown as an output. This is not the ideal behaviour because as per the requirements the range of values that should be accepted by the ratings field is between 1 and 10 including both the values.
 - **Requirement Mapping:** 7.1
 - **Observed Output:** The discount given is 15%
 - **Expected Output:** Rating must be between 1 and 10.
 - Even when the user status is set to New, the application is calculating for the discount and giving an output "The discount given is 10%". This is the wrong output as the ideal output should have been "No discount given" because as per the requirements New users cannot be given any discounts regardless of reward status, product, and season.
 - **Requirement Mapping:** 9.1
 - **Observed Output:** The discount given is 10%
 - **Expected Output:** No discount given.
 - Even when the product category is set to unknown - the combination of where the user status is set to Returning, the reward member status is set to Silver, and the season when the item was bought is set to Summer, the application gives an output "the discount given is 10%" instead of "No discount given" which is not the ideal case. This because as per the requirements, if product category is set to unknown, then the user gets no discount.
 - **Requirement Mapping:** 9.2
 - **Observed Output:** The discount given is 10%
 - **Expected Output:** No discount given
 - As per the requirements, if the item is bought in Spring, then the Returning users who are Bronze members who bought Electronics should get a 10% discount voucher. But when the same combination is entered into the application the output given is "the discount given is 5%" which is incorrect.
 - **Requirement Mapping:** 19.4.1
 - **Observed Output:** The discount given is 5%
 - **Expected Output:** The discount given is 10%
 - As per the requirements, if the item is bought in Winter, then the Returning users who are Gold members who bought Electronics should get a 25% discount voucher. But when the same combination is entered into the application the output given is "the discount given is 20%" which is incorrect.
 - **Requirement Mapping:** 9.5.1
 - **Observed Output:** The discount given is 20%
 - **Expected Output:** The discount given is 25%
 - As per the requirements, if the item is bought in Fall, then the Returning users who are Silver members who bought Electronics should get a 15% discount voucher. But when the same combination is entered into the application the

output given is "the discount given is 10%" which is incorrect.

- **Requirement Mapping:** 9.6.1
 - **Observed Output:** The discount given is 10%
 - **Expected Output:** The discount given is 15%

- As per the requirements, if any other combination of inputs(returning user, reward status, product, and season) set apart from the ones mentioned in 9.3.1, 9.4.1, 9.5.1, and 9.6.1 then the user will not get a discount. But in this case if the user status is set to Returning, the user member status is set to Gold, the season in which the user purchases the item is set to Fall and the product category is set to Electronics, then the output given by the application is "The discount given is 10%" instead of the output "No discount is given".

- **Requirement Mapping:** 9.7
 - **Observed Output:** The discount given is 10%
 - **Expected Output:** No discount given

Lessons Learned

This project provided valuable hands-on experience in software testing, highlighting the importance of rigorous testing of software applications to improve the ability to identify and mitigate defects in software programs. The project showed that different testing techniques could be used to ensure software program quality and performance. Specifically, the project demonstrated the benefits of the Boundary Value Testing Technique, Equivalence Partitioning Testing Technique, and Cause/Effect Analysis Testing Technique.

The Boundary Value Testing Technique proved to be a practical approach to identifying defects or bugs that may exist at the boundaries of the input domain, while the Equivalence Partitioning Testing Technique reduced the number of test cases required to test the software application. By combining these techniques, we could test all input values within the input domain, including those at the boundaries, and reduce the required test cases. Using equivalence partitioning in software testing can save time and effort while improving software application quality and reliability.

Moreover, the Cause/Effect Analysis Testing Technique proved to be a powerful method of identifying potential causes of defects or problems by analyzing the relationships between different factors or variables in a system. This technique helped identify underlying causes contributing to the defects, enabling efficient mitigation of these issues. The cause/effect analysis technique is beneficial for testing complex systems where the relationships between different variables are not always apparent. It can help testers to identify the root causes of defects or problems, rather than just the symptoms, and develop more targeted and effective test cases to address these underlying causes.

The project demonstrated the importance of utilizing different testing techniques to ensure software application quality and performance. Identifying and mitigating defects or bugs early in the development cycle can reduce the likelihood of encountering issues during production, thus increasing software programs' overall reliability and quality.

Assignment 3

In today's fast-paced mobile application industry, applications must be thoroughly tested to ensure a seamless user experience, quality, and reliability. Testing is crucial; it helps identify and fix bugs and other issues before deployment. However, testing all possible combinations of input variables can be a daunting task, mainly when many variables and combinations are involved.

One solution to this problem is using the Design of Experiments (DOE). This statistical method can help optimize testing efforts by creating test cases based on pairwise combinations of requirements, allowing the testers to test many variables and their combinations efficiently.

In this project, the testing team is tasked with utilizing a DOE tool to create test cases that cover all possible pairwise combinations of requirements for a newly developed mobile application. We will use a DOE tool to develop test cases based on the given specifications to achieve this.

The primary objective of this project was to conduct research and identify a tool that supports DOE that can support the creation of pairwise combination test cases for the mobile application and to demonstrate its application in creating test cases.

Additionally, the project required me to prepare a report that aims to provide an overview of the selected DOE tool, explain its scope and purpose, and provide an example of its usage through a use case.

Furthermore, we will provide details on the number of test cases created by the tool, along with high-resolution screenshots. Finally, we will evaluate the DOE tool based on its usability, coverage, and features and provide our assessment in my own words.

Work Completed

To start the project, I scoured the internet for articles and information about the Design of Experiments (DOE) and spent considerable time reading through multiple websites like “WHAT IS DESIGN OF EXPERIMENTS (DOE)?” [11] and watching lecture videos to form a solid understanding of what DOE is and when it is used. With this knowledge, I began my report by explaining DOE and its use.

After understanding DOE, I learned the basics of pairwise testing, which involves creating test cases that cover all possible pairwise combinations of requirements. I again took to the internet to learn about pairwise testing, spending a significant amount of time reading through multiple websites like “Pairwise Testing Or All-Pairs Testing Tutorial With Tools And Examples”[12] & “Pairwise Testing Guide: How To Perform Pairwise Testing”[13] and watching lecture videos to understand the objective, benefits, and limitations of pairwise testing. I then explained pairwise testing in my report.

Now, with the foundations required for the project formed, I moved on to the next thing, which was to research different DOE tools and identify a suitable tool.

Since we are creating test cases that cover all possible pairwise combinations, I scrounged through the net searching for dependable articles and information about different

Pairwise Testing tools available in the market. After reading through multiple articles and singling out helpful information, I understood that Pairwise testing tools are simple test case generators that allow testers to input and constrain a model of input parameters before generating test configurations based on the model. These tools are simple to use with many input parameters and can also add constraints to the input parameters and generate test configurations. To perform Pairwise testing, numerous tools available on the internet apply the all-pairs testing technique that facilitates us to effectively automate the Test Case Design process by generating a compact set of parameter value choices as the desired Test Cases. Some well-known tools from the industry are mentioned below.

- **PICT**: ‘Pairwise Independent Combinatorial Testing’, provided by Microsoft Corp.
- **IBM FoCuS** : ‘Functional Coverage Unified Solution’, provided by IBM.
- **ACTS**: Advanced Combinatorial Testing System’, provided by NIST, an agency of the US Government.
- **Hexawise**
- **Jenny**:
- **Pairwise** : by Inductive AS
- **VPTag** : free All-Pair Testing Tool

After going through the available tools listed above, the tool that I have used for the project is PICT (Pairwise independent combinatorial testing). The PICT tool from Microsoft is a well-documented pairwise test generator that also supports many advanced features (Microsoft, 2019[16]). PICT is an open-source and cross-platform program programmed in C++ and is single-threaded. The PICT tool is also compatible with various software systems, such as embedded devices, mobile apps, and web applications. It can be a versatile and adaptable testing tool for various software development projects. [14].

Now, as per the project's requirements, after identifying a suitable DOE tool for this project, I moved on to explain the PICT tool's scope and purpose. In simple terms, the scope of the PICT tool is that it supports higher interaction strength testing via a command line switch by allowing testers to generate more effective test cases and test configurations than manually generated tests in a fraction of the time that hands-on test case design requires. Again, in simple terms, the PICT tool's primary purpose is to generate a compact set of parameter value options that represent the test cases that the testers should use to achieve complete combinatorial coverage of project parameters. The PICT tool also includes constraints, value weights, and other advanced features described in Czerwonka (2006) [15]. Following these sections in the report, as per the project requirement, I have explained about usage of the PICT tool with a working example.

There are two different kinds of PICT tools available.

- **Online Version**: Pairwise Pict Online[17] - An online service that easily generates pair-wise test cases. It's powered by Microsoft Pict under the hood.

- **Offline Version:** PICT[16] - PICT runs as a command line tool

For this assignment I have used the offline version. So for this, I have visited the PICT releases web-page[16] and downloaded the executable (.exe) file and used this executable (.exe) file to generate pairwise combination test cases based upon the given requirements.

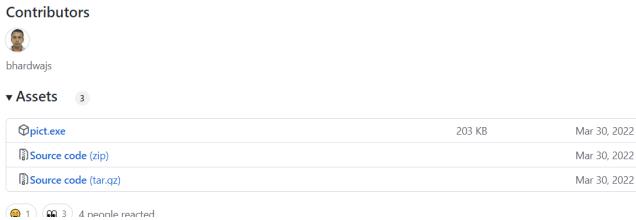


Figure 3: PICT offline latest version available releases

So, now with the DOE tool identified for the project, I moved on to creating the test cases. Taking the given requirements for the newly developed mobile application, the first step that I did was to develop the model file that would act as the input to the PICT tool. A model consists of the following sections: parameter definitions, [sub-model definitions], and the [constraint definitions] as you can see in the figure below[18].

Type of Phone					Parallel Tasks Running	Connectivity	Memory	Battery Level
iPhone X	Yes		Wireless	1 GB	< 20%			
iPhone 8	No		3G	2 GB	20 - 39%			
Samsung S9			4G	4 GB	40 - 59%			
Huawei Mate			Edge	6 GB	60 - 79%			
Google Pixel 3					80 - 100%			

Figure 4: Given requirements for the mobile application

parameter definitions
[sub-model definitions]
[constraint definitions]

Figure 5: Syntax to be followed for the input model file

Therefore, following the specified syntax, I created my

input model file (CSE565 Assignment 3 parameters) as a text file.

```
CSE565_Assignment3_params - Notepad
File Edit Format View Help
Type of Phone: iPhone X, iPhone 8, Samsung S9, Huawei Mate, Google Pixel 3
Parallel Tasks Running : Yes, No
Connectivity: Wireless, 3G, 4G, Edge
Memory: 1GB, 2GB, 4GB, 6GB
Battery Level: <20%, 20 - 39%, 40 - 59%, 60 - 79%, 80 - 100%
```

Figure 6: CSE565_Assignment3_params.txt file

Now, with the input model file created, I have opened my Command Prompt and navigated to the folder where I have my executable (.exe) file. After navigating the folder, I run the command ".pict CSE565 Assignment3_params.txt," and the test cases are generated.

With the tool identified and the test cases generated, to finish the report, I evaluated the usability, coverage, and features of the PICT tool. I scoured the internet for dependable articles and information like "Pairwise Independent Combinatorial Testing"[18] and formed a solid understanding of the tool's capabilities. Using the gathered information, I then completed the sections Assessment of the PICT tool in terms of usability, coverage, and features of the PICT tool and Assessment of the PICT tool in my own words in terms of usability, coverage, and features of the PICT tool, as per the project's requirements.

Results

So, to summarize, the number of test cases Generated by DOE tool i.e. the PICT tool are 25.

```
cmd Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

D:\setup files 1>.\pict CSE565_Assignment3_params.txt
Type of Phone Parallel Tasks Running Connectivity Memory Battery Level
Google Pixel 3 No Wireless 2GB 60 - 79%
Samsung S9 Yes 3G 4GB 80 - 100%
iPhone X No Edge 1GB 80 - 100%
Huawei Mate Yes 4G 6GB 40 - 59%
iPhone X No 3G 6GB 60 - 79%
iPhone X Yes 4G 2GB <20%
Samsung S9 Yes 4G 1GB 60 - 79%
iPhone 8 No Wireless 4GB <20%
Samsung S9 No Edge 2GB 40 - 59%
Google Pixel 3 Yes 4G 6GB 80 - 100%
iPhone 8 Yes Edge 1GB 60 - 79%
Huawei Mate No 3G 1GB <20%
Samsung S9 Yes Wireless 6GB <20%
Google Pixel 3 No Edge 4GB <20%
iPhone 8 No 4G 6GB 20 - 39%
Huawei Mate Yes Edge 4GB 20 - 39%
Samsung S9 No 3G 2GB 20 - 39%
Google Pixel 3 Yes Wireless 1GB 20 - 39%
iPhone X No Wireless 4GB 40 - 59%
Huawei Mate No Wireless 2GB 80 - 100%
Huawei Mate No 4G 4GB 60 - 79%
iPhone 8 No 3G 2GB 80 - 100%
iPhone X Yes Edge 6GB 20 - 39%
Google Pixel 3 Yes 3G 1GB 40 - 59%
iPhone 8 No Edge 2GB 40 - 59%
```

D:\setup files 1>.\pict CSE565_Assignment3_params.txt > GeneratedTestCases.txt
D:\setup files 1>.\pict CSE565_Assignment3_params.txt > GeneratedTestCases.xls
D:\setup files 1>

Figure 7: Generated Test Cases in Command Line Interface

Because of the readability issue of the output in the command line, I have saved the output in a different text file by running the command ".pict

CSE565_Assignment3_params.txt > GeneratedTestCases.xls" as you can see in the Figure above. And from the text file, the output has been moved into an excel file for better readability (which you can see at the end of this report) by running the command "./pict CSE565_Assignment3_params.txt > GeneratedTestCases.xls" as you can see in the figure above.

	A	B	C	D	E
1	Type of Phone	Parallel Tasks Running	Connectivity	Memory	Battery Level
2	Google Pixel 3	No	Wireless	2GB	60 - 79%
3	Samsung S9	Yes	3G	4GB	80 - 100%
4	iPhone X	No	Edge	1GB	80 - 100%
5	Huawei Mate	Yes	4G	6GB	40 - 59%
6	iPhone X	No	3G	6GB	60 - 79%
7	iPhone X	Yes	4G	2GB	<20%
8	Samsung S9	Yes	4G	1GB	60 - 79%
9	iPhone 8	No	Wireless	4GB	<20%
10	Samsung S9	No	Edge	2GB	40 - 59%
11	Google Pixel 3	Yes	4G	6GB	80 - 100%
12	iPhone 8	Yes	Edge	1GB	60 - 79%
13	Huawei Mate	No	3G	1GB	<20%
14	Samsung S9	Yes	Wireless	6GB	<20%
15	Google Pixel 3	No	Edge	4GB	<20%
16	iPhone 8	No	4G	6GB	20 - 39%
17	Huawei Mate	Yes	Edge	4GB	20 - 39%
18	Samsung S9	No	3G	2GB	20 - 39%
19	Google Pixel 3	Yes	Wireless	1GB	20 - 39%
20	iPhone X	No	Wireless	4GB	40 - 59%
21	Huawei Mate	No	Wireless	2GB	80 - 100%
22	Huawei Mate	No	4G	4GB	60 - 79%
23	iPhone 8	No	3G	2GB	80 - 100%
24	iPhone X	Yes	Edge	6GB	20 - 39%
25	Google Pixel 3	Yes	3G	1GB	40 - 59%
26	iPhone 8	No	Edge	2GB	40 - 59%

Figure 8: Generated Test Cases in Excel File

Lessons Learned

The project has provided several valuable lessons related to DOE and Pairwise Testing. One key takeaway is that DOE is a powerful and flexible data collection and analysis tool that can be applied to various experimental situations. Its ability to manipulate multiple input factors to determine their impact on a desired output. By adjusting multiple inputs simultaneously, DOE can identify significant interactions that may go unnoticed when experimenting with a single factor at a time. Furthermore, DOE can be employed to validate hypothesized input-output relationships and develop predictive models suitable for what-if analysis. It has demonstrated good code coverage, and many companies have successfully used it in software testing.

Another important lesson learned is that Pairwise Testing is a black-box testing that ensures full test coverage. This technique is beneficial for designing tests for applications with multiple parameters, as it covers all possible combinations of input parameters. Tests are designed to ensure that each pair of input parameters to a system has all possible discrete combinations of those parameters. Although the test suite does not cover all possible combinations, it is highly effective at detecting defects. PICT is a powerful tool that can help testers create test cases optimized for pairwise testing, covering a wide range of input parameters and dealing with constraints. Its usability, coverage, and features

make it an invaluable tool for software testing, particularly for projects with many input parameters. Software developers and testers can use PICT to ensure their applications are thoroughly tested and defects-free.

Using DOE and Pairwise Testing together can provide a robust testing solution. DOE can identify the input factors that have the most significant effect on the output, and Pairwise Testing can ensure that all combinations of those factors are tested. Using both techniques, testers can create a test suite optimized for maximum code coverage and defect detection. Using the appropriate testing tools and techniques for each project is essential. In this project, the use of DOE and Pairwise Testing was suitable due to the large number of input parameters that needed to be tested. Other testing techniques may be more appropriate for projects with fewer input parameters.

Overall, the project provided important insights into the capabilities of DOE and Pairwise testing and their usefulness in software testing. Both techniques are powerful tools that can help testers improve the efficiency, effectiveness, and flexibility of the testing process and enhance the quality of software products. Software developers and testers should consider incorporating these techniques into their testing processes to ensure comprehensive test coverage and detect defects early in the development cycle to optimize testing strategies and improve software quality.

Assignment 4

This project consists of four tasks.

In Task 1, the VendingMachine.java code is provided, and the goal is to use structural-based testing techniques to develop a set of test cases using JUnit for code coverage. The test cases should aim to reach 100% statement coverage and at least 90% decision coverage, except for the False decision for line 32 (input < 45).

Task 2 involves creating a report describing the tool used for Task 1, including its scope, purpose, and an example of usage. Additionally, the types of code coverage provided by the tool and the test cases' scope and coverage developed in Task 1 should be explained. Finally, the report should include a screenshot of the tool's report showing the coverage achieved by the test cases and an evaluation of the tool's usefulness in terms of usability, coverage, and tool features.

Task 3 requires selecting a static source code analysis tool and executing it on StaticAnalysis.java to identify the code's two different data flow anomalies. Task 4 involves continuing the report created in Task 2, which should include a description of the two data flow anomalies and why they are considered data anomalies. Furthermore, the report should include a high-resolution screenshot showing the analysis performed for detecting data anomalies and evaluating the tool's usefulness regarding usability, coverage, and tool features.

Work Completed

I began this project by selecting an appropriate Integrated Development Environment (IDE) for the task. After considering the available options, I used IntelliJ IDEA, commonly

paired with JaCoCo or Emma for high-level applications. However, since the task was to test the functionality of a single file, I opted to use the less resource-intensive IntelliJ Code Coverage Runner. IntelliJ IDEA is an Integrated Development Environment (IDE) for Java, Kotlin, and other programming languages, and IntelliJ Code Coverage Runner is one of IntelliJ's main features. It is an inbuilt code coverage tool that allows developers to measure how much their tests exercise their code. [19]

I then explained the scope and purpose of the IntelliJ Code Coverage Runner tool, drawing on information from dependable articles and online resources[19]. In simple terms, the scope of the IntelliJ Code Coverage Runner in IntelliJ IDEA is that the tool allows developers to measure code coverage for a wide range of programming languages and frameworks, with the ability to analyze code coverage for the unit, integration, and functional tests. At the same time, the purpose of the IntelliJ Code Coverage Runner tool is to analyze code and determine which statements or lines have been executed during tests, enabling developers to identify untested or under-tested areas of their code and create more effective tests to improve code quality. Following these two sections, I described a working example as a use case to showcase the usage of the IntelliJ Code Coverage Runner tool.

I then described the different types of code coverage the IntelliJ Code Coverage Runner tool provided, drawing on information from the lecture videos. The IntelliJ Code Coverage Runner tool in IntelliJ IDEA offers various types of code coverage to assist developers in determining how thoroughly a given test suite executes a program's source code. The following are the types of code coverage analysis provided by the IntelliJ IDEA code coverage runner tool: Statement/Line Coverage, Branch/Decision Coverage, Method Coverage, and Class Coverage.

Now, with the identified IDE and the tool for this project, I moved to develop a set of test cases using JUnit for code coverage. To summarize the VendingMachine.java code, there are 24 lines/statements and eight decisions/conditions in the dispenseItem function. The eight decisions/conditions result in 16 branches because each decision has two possible outcomes (true or false). The 8 decisions/conditions in the code are:

- **Decision 1:** item == "candy"
- **Decision 2:** item == "coke"
- **Decision 3:** item == "coffee"
- **Decision 4:** input > cost
- **Decision 5:** input == cost
- **Decision 6:** input < 45
- **Decision 7:** input < 25
- **Decision 8:** input < 20

So, I started creating test cases covering all 24 lines/statements and the 15 decision branches. After multiple attempts, I was finally able to create a set of 5 test cases that resulted in the required code coverage, which is 100% statement coverage and at least 90% decision coverage.

With the test cases developed, I moved on to the next part, describing each test case, its scope, and coverage in the report. Each test case was described in detail in the report, including the test case's description, scope, and coverage. The coverage of each test case was divided into two subsections, Statement/Line Coverage, and Decision/Branch Coverage. The Statement/Line Coverage subsection showcased which lines/statements of the code were covered by that particular test case and the percentage in total. The Decision/Branch Coverage subsection showcased which decision and which branch (True/False) of that decision were covered by that particular test case and the percentage of total decision coverage. Lastly, to complete Task 2, I had to include in the report the evaluation of the IntelliJ Code Coverage Runner tool in my own words. Using the tool firsthand and with some information from some dependable websites, I could efficiently complete this section, describing the tool's usability, coverage, and features.

Moving on to task 3, I had to select a static source code analysis tool. Before using which tool to be used, I wanted to learn what precisely static source code analysis is. So, for this, I scoured through the net, searching for dependable articles and information about static source code analysis. After reading multiple articles, I understood what static source code analysis is and what was expected from me in this task. So, I scoured through the net, searching for various static source code analysis tools available in the market. After going through all the available tools, I decided that the PMD(Programming Mistake Detector)[20] is suitable for this project. This is because PMD is a well-known open-source static source code analysis tool in the Java community for detecting potential bugs or issues in programming languages like Java, Kotlin, and others and is available as a plugin for several popular Java development environments, such as Eclipse and IntelliJ IDEA. Without running the program, PMD examines its source code for common programming errors, identifying coding issues such as unused variables, empty catch blocks, and inefficient code constructs and recommending improvements.

So, using this tool, I completed Task 3 to identify the two data flow anomalies in the StaticAnalysis.java code file. Lastly, to complete Task 4, I had to include in the report the evaluation of the PMD(Programming Mistake Detector) tool in my own words. Using the tool firsthand and with some information from some dependable websites, I could efficiently complete this section, describing the tool's usability, coverage, and features.

Results for Code Coverage

I created the minimum number of test cases which is 5 as shown in the attached image Figure 9, to complete the code coverage while satisfying the given condition mentioned in the project description - 100% statement coverage and at least 90% decision coverage. And, as shown in the attached images, Figures 12 and 13, the developed unit test cases provide the required coverage.

Total Statement Coverage

From Figure 10, you can see that we have a 100% statement coverage.

```

3 class VendingMachineTest {
4
5     @org.junit.jupiter.api.Test
6     void testCase1() {
7         VendingMachine testcase = new VendingMachine();
8         assertEquals("Item dispensed and change of 20 returned", testcase.dispenseItem( input: 40, item: "candy"));
9     }
10
11    @org.junit.jupiter.api.Test
12    void testCase2() {
13        VendingMachine testcase = new VendingMachine();
14        assertEquals("Item dispensed.", testcase.dispenseItem( input: 25, item: "coffee"));
15    }
16
17    @org.junit.jupiter.api.Test
18    void testCase3() {
19        VendingMachine testcase = new VendingMachine();
20        assertEquals("Item not dispensed, missing 5 cents. Can purchase candy or coke.", testcase.dispenseItem( input: 40, item: "coffee"));
21    }
22
23    @org.junit.jupiter.api.Test
24    void testCase4() {
25        VendingMachine testcase = new VendingMachine();
26        assertEquals("Item not dispensed, missing 3 cents. Can purchase candy.", testcase.dispenseItem( input: 22, item: "coke"));
27    }
28
29    @org.junit.jupiter.api.Test
30    void testCase5() {
31        VendingMachine testcase = new VendingMachine();
32        assertEquals("Item not dispensed, missing 2 cents. Cannot purchase item.", testcase.dispenseItem( input: 18, item: "candy"));
33    }
34

```

Figure 9: Generated Test Cases for the Vending Machine

Figure 10: The visual representation of the code coverage

- Line 9:** Covered by the Test Cases - 1, 2, 3, 4, and 5.
- Line 10:** Covered by the Test Cases - 1, 2, 3, 4, and 5.
- Line 11:** Covered by the Test Cases - 1, 2, 3, 4, and 5.
- Line 12:** Covered by the Test Cases - 1, 2, 3, 4, and 5.
- Line 13:** Covered by the Test Cases - 1, and 5.
- Line 14:** Covered by the Test Cases - 1, 2, 3, 4, and 5.
- Line 15:** Covered by the Test Cases - 2, and 4.
- Line 16:** Covered by the Test Cases - 1, 2, 3, 4, and 5.
- Line 17:** Covered by the Test Cases - 3.
- Line 19:** Covered by the Test Cases - 1, 2, 3, 4, and 5.
- Line 21:** Covered by the Test Cases - 1.
- Line 22:** Covered by the Test Cases - 1.
- Line 24:** Covered by the Test Cases - 2, 3, 4, and 5.
- Line 26:** Covered by the Test Cases - 2.
- Line 27:** Covered by the Test Cases - 2.
- Line 31:** Covered by the Test Cases - 3, 4, and 5.
- Line 32:** Covered by the Test Cases - 3, 4, and 5.
- Line 33:** Covered by the Test Cases - 3, 4, and 5.

	Class, %	Method, %	Line, %	Branch, %
Element				
all	66% (2/3)	66% (6/9)	70% (35/50)	75% (15/20)
StaticAnalysis	0% (0/1)	0% (0/3)	0% (0/15)	0% (0/4)
VendingMachine	100% (1/1)	100% (1/1)	100% (24/24)	93% (15/16)
VendingMachineTest	100% (1/1)	100% (5/5)	100% (11/11)	100% (0/0)

Figure 11: Terminal Output of the Total Code Coverage

Element	Class, %	Method, %	Line, %	Branch, %
all	66% (2/3)	66% (6/9)	70% (35/50)	75% (15/20)
StaticAnalysis	0% (0/1)	0% (0/3)	0% (0/15)	0% (0/4)
VendingMachine	100% (1/1)	100% (1/1)	100% (24/24)	93% (15/16)
VendingMachineTest	100% (1/1)	100% (5/5)	100% (11/11)	100% (0/0)

Figure 12: Total Code Coverage Statistics

Package	Class, %	Method, %	Branch, %	Line, %
<empty package>	66% (2/3)	72% (8/11)	77% (32/39)	70% (35/50)
Class ...	0% (0/1)	0% (0/3)	0% (0/4)	0% (0/15)
StaticAnalysis	0% (0/1)	0% (0/3)	0% (0/4)	0% (0/15)
VendingMachine	100% (1/1)	100% (1/1)	93.8% (15/16)	100% (24/24)
VendingMachineTest	100% (1/1)	100% (5/5)	100% (11/11)	100% (0/0)

Figure 13: Exported Coverage Report

- Line 34:** Covered by the Test Cases - 3, 4, and 5.
- Line 35:** Covered by the Test Cases - 4, and 5.
- Line 36:** Covered by the Test Cases - 3, 4, and 5.
- Line 37:** Covered by the Test Cases - 5.
- Line 40:** Covered by the Test Cases - 1, 2, 3, 4, and 5.

Total Decision Coverage

Figure 10 shows that we have a decision coverage of 93%. This is due to the fact that, as stated in the project description, all decisions have been covered with the exception of the False branch for the decision/condition on line 32 ($\text{input} < 45$). So, the code covers 15 of the 16 branches (8 conditions), for a total of 93.75% coverage.

- Decision 1:** `item == "candy"`
 - True Branch:** Covered by the Test Cases - 1 and 5.
 - False Branch:** Covered by the Test Cases - 2, 3, and 4.
- Decision 2:** `item == "coke"`
 - True Branch:** Covered by the Test Cases - 2 and 4.
 - False Branch:** Covered by the Test Cases - 1, 3, and 5.
- Decision 3:** `item == "coffee"`
 - True Branch:** Covered by the Test Cases - 3.
 - False Branch:** Covered by the Test Cases - 1, 2, 4, and 5.
- Decision 4:** `input > cost`
 - True Branch:** Covered by the Test Cases - 1.
 - False Branch:** Covered by the Test Cases - 2, 3, 4, and 5.
- Decision 5:** `input == cost`
 - True Branch:** Covered by the Test Cases - 2.
 - False Branch:** Covered by the Test Cases - 3, 4, and 5.
- Decision 6:** `input < 45`
 - True Branch:** Covered by the Test Cases - 3, 4, and 5.

- **False Branch:** NOT COVERED.
- **Decision 7:** input < 25
 - **True Branch:** Covered by the Test Cases - 4.
 - **False Branch:** Covered by the Test Cases - 3 and 5.
- **Decision 8:** input < 20
 - **True Branch:** Covered by the Test Cases - 5.
 - **False Branch:** Covered by the Test Cases - 3 and 4.

Results from the Static Analysis Tool

To perform its analysis, PMD employs a set of predefined rules or code guidelines that cover a wide range of coding issues, such as inefficient code, unused variables, and potential security vulnerabilities. It can also be customized with additional rules to meet the requirements of a specific project. These rules are based on best practices and common coding errors, and they can be tailored to the needs of a specific project. When PMD is run, it reports any violations of these rules, as well as where they occur in the source code[22].

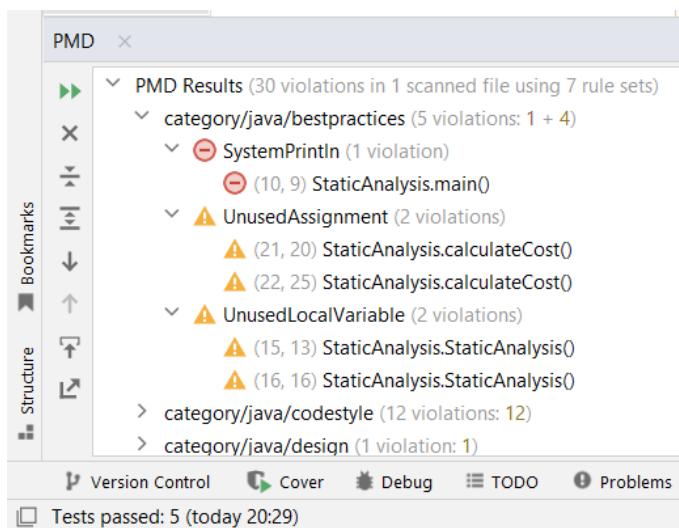


Figure 14: PMD RESULTS 1

Therefore, the StaticAnalysis.java file contains the following two different data flow anomalies:

- **UnusedAssignment – int cost = 0;** – Found 'DD'-anomaly for variable 'cost' (lines '21'-'26') in public static String calculateCost(). As the value of cost = 0 is never used and also it is later redefined on lines 26 and 30 depending on how the decision on line 24 evaluates, this error is thrown as Unused Assignment.
- **UnusedAssignment – String output = """;** – Found 'DD'-anomaly for variable 'output' (lines '22'-'34') in public static String calculateCost(). As the value of output ="" is never used and also it is later redefined to a different string value on lines 34 and 36 depending on how the decision on line 33 evaluates, this error is thrown as Unused Assignment.

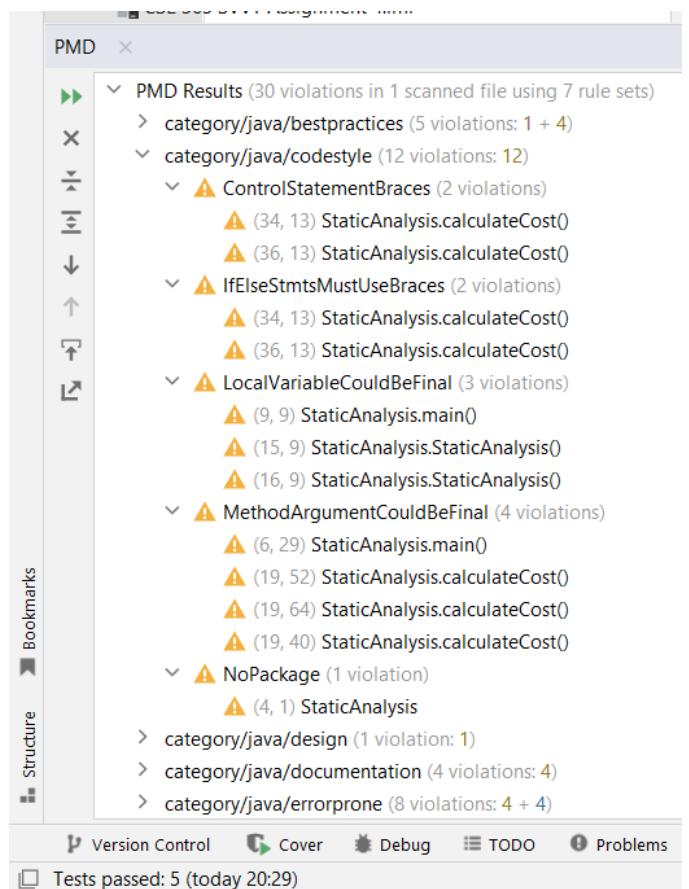


Figure 15: PMD RESULTS 2

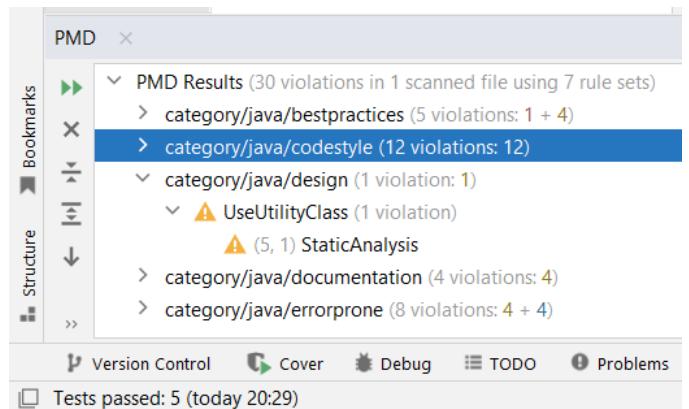


Figure 16: PMD RESULTS 3

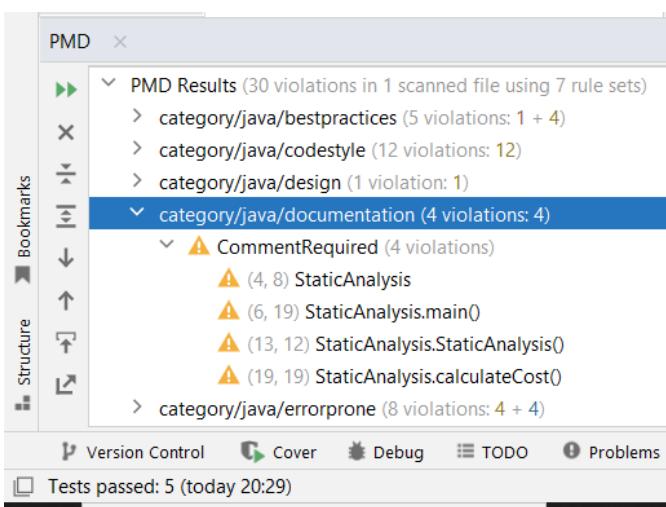


Figure 17: PMD RESULTS 4

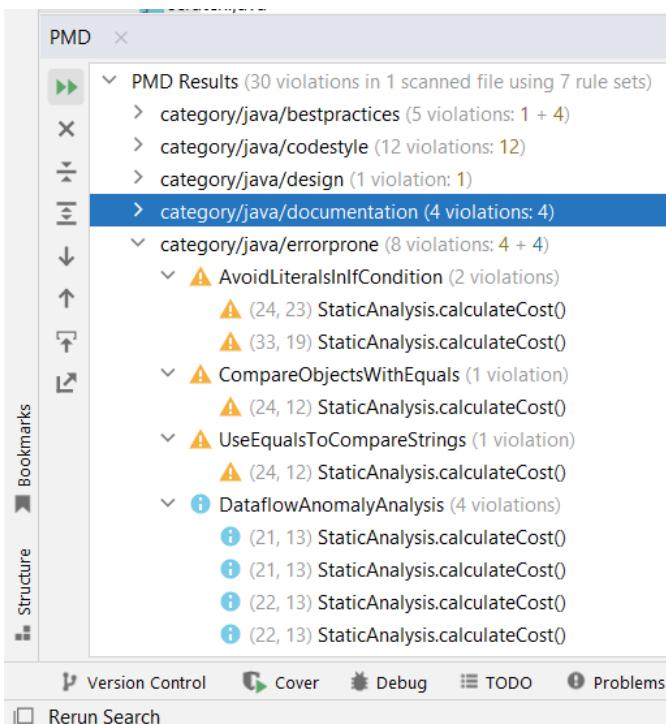


Figure 18: PMD RESULTS 5

The screenshot shows the IntelliJ IDEA code editor with the file 'VendingMachine.java' open. The code contains the following assignment:

```

public static String calculateCost(int weight, int length, String product)
{
    int cost = 0;
    String output = "";
    if(product == "Electronics")
        cost = weight * length * 2;
    else
        cost = weight * length;
}

```

The line 'int cost = 0;' is highlighted with a yellow arrow, indicating it is unused. The code coverage report on the right shows 100% coverage for all methods except for this specific assignment.

Figure 19: Data Anomaly 1 - UnusedAssignment - int cost = 0;

The screenshot shows the IntelliJ IDEA code editor with the file 'VendingMachine.java' open. The code contains the following assignment:

```

public static String calculateCost(int weight, int length, String product)
{
    int cost = 0;
    String output = "";
    if(product == "Electronics")
        cost = weight * length * 2;
    else
        cost = weight * length;
}

```

The line 'String output = "";' is highlighted with a yellow arrow, indicating it is unused. The code coverage report on the right shows 100% coverage for all methods except for this specific assignment.

Figure 20: Data Anomaly 2 - UnusedAssignment - String output = "";

Lessons Learned

This project has provided an opportunity to explore and evaluate additional testing and analysis techniques and their associated tools for improving software development effectiveness and efficiency. From this project, several key lessons have been learned that can be beneficial in future software development projects.

Firstly, selecting an appropriate Integrated Development Environment (IDE) is crucial to achieving the desired testing and analysis outcomes. The choice of IDE should consider the task's scope, purpose, and available resources. In this project, IntelliJ IDEA and its built-in Code Coverage Runner are practical tools for measuring code coverage in a single file, with the latter being less resource-intensive. By utilizing online resources and dependable articles, I gained a deeper understanding of the tool's scope and purpose and its different types of code coverage analysis.

Secondly, I learned that developing a comprehensive set of test cases is a time-consuming task that requires careful consideration of the code's structure and logic. Creating a set of comprehensive test cases is essential to achieve the desired code coverage. During the project, I had to create test cases to cover all 24 lines/statements and the 15 decision branches of the VendingMachine.java code file. It took mul-

multiple attempts to develop a set of five test cases that achieved the desired code coverage of 100% statement coverage and at least 90% decision coverage. I also learned the importance of documenting each test case's scope and coverage in the report to demonstrate the effectiveness of the testing process.

Thirdly, I discovered the importance of static source code analysis in identifying potential bugs or issues in the code. I have learned that static source code analysis tools such as PMD can help identify coding issues that manual code reviews may miss. These tools can be helpful in improving code quality and reducing the number of bugs in software projects.

This project has provided a valuable learning experience and an opportunity to apply testing and analysis techniques to real-world code and gain practical experience with testing and analysis tools, creating comprehensive test cases, and evaluating tools' effectiveness and efficiency. The lessons learned from this project can benefit future software development projects, enabling developers to create high-quality software effectively and efficiently.

Assignment 5

This project aims to demonstrate the use of graphical user interface (GUI) testing techniques and tools. GUI testing is a software testing technique that focuses on the user interface of an application. In this project, we will select an application with a graphical user interface, make changes to the GUI, and use a GUI test automation tool to test the application. This project has two tasks.

For Task 1, this project uses a graphical user interface (GUI) testing technique and a GUI testing tool for a software application. The project requires creating any application with a GUI, which can be coded in any programming language and has at least three GUI pages with a flow from one page to another. Each page should have GUI elements such as images, buttons, labels, text boxes, and two of the following: scroll-bar, check-boxes, radio button, slider, list, and drawing elements. Two versions of the application should be created, where the first version will serve as a baseline for the second version, which will include at least three changes in the GUI, such as the size, location, and orientation of the GUI elements. Additionally, a change in the flow of pages, the link from one page to another, will also be included in the second version.

For Task 2, this project requires selecting a GUI test automation tool to record test cases and report the test results. Test cases should be developed to test the existence of GUI elements, the correctness of their location and size, the content of the GUI element, and the correctness of the link from one page to another. The test cases developed for the first version of the GUI will be executed for the second version, and the test results will be reported, highlighting the number of test cases that passed or failed.

As part of this project, a report will provide a detailed description of the application with screenshots of the GUI for version one and version two and a description of the selected GUI test automation tool, including its features, scope, and area of usage. The test cases developed for the two versions

of the GUI will be explained, highlighting the coverage of the test cases. The test results after the execution of the test cases will be explained, with references to screenshots in the report. Additionally, the report will include an assessment of the GUI testing tool in terms of its usability, coverage, and features. Overall, this project aims to showcase the importance of GUI testing and demonstrate the use of GUI testing tools for ensuring the quality and reliability of software applications with a graphical user interface.

Work Completed

I have made a web application using HTML and CSS for a user to register, log in, and give a review of my application. To begin with, I salvaged the underlying code framework for a login form from a GitHub repository [23] and then duplicated the same form to create the two additional pages for sign-up and review. Following this, I have changed each page to have GUI elements such as images, buttons, labels, text boxes, and two of the following: scroll-bar, check-boxes, radio button, slider, list, and drawing elements to meet the project requirements.

The first page in my application is the Sign Up form, where the user can register by providing specific information such as their username, email, phone number, gender, age, and password for the account. Per the project requirements, each page should have GUI elements such as images, buttons, labels, text boxes, and any of the following: scroll-bar, check-boxes, radio button, slider, list, and drawing elements. Therefore, the Sign-Up form has a small user avatar image at the top and the title "Create Account" just below the image. The sign-up form includes text input boxes for username, email, phone number, password, and confirm password fields. The form contains a group of radio buttons rendered as a list for the gender field, allowing the user to select their sex, and a slider for the age field, allowing the user to set their age by sliding the rounded element left and right. There are also two buttons on the sign-up form: The sign-Up Button and the Login Button. Now, to implement the navigation flow on this page, I have used the two created buttons mentioned previously. When the user clicks on the Sign-Up button, the application will create an account for the user and reroute them to the login page immediately so that the user can log in to the application and leave feedback on the review form page. Similarly, when the user clicks on the Login Button, the application will reroute them to the login page immediately so that the user can log in to the application and leave feedback on the review form page.

The second page of the application form is the Login Form, where the user can log in by entering their username, email, and password. Per the project requirements, each page should have GUI elements such as images, buttons, labels, text boxes, and any of the following: scroll-bar, check-boxes, radio button, slider, list, and drawing elements. Therefore, similar to the sign-up form, the login form has a small user avatar image at the top and a title "Login" just below the image. Apart from the image, title, and input fields for login credentials, the login form has four other components: Scroll Box with a Label, Radio Button with a Label, Sign Up Button, and the Login Button. The scroll box

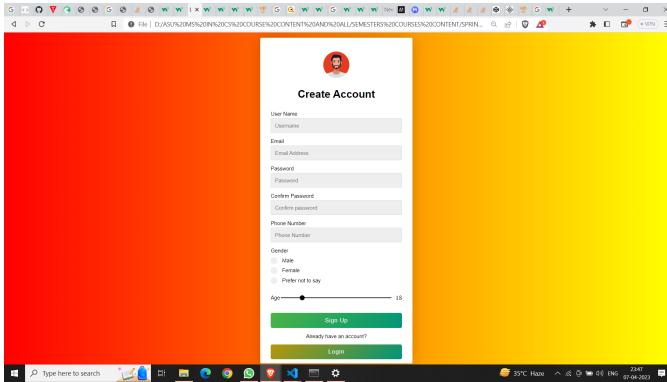


Figure 21: Form Page 1 : Registration / Sign Up Form

Figure 22: Code Part 1 for the Sign Up Form

```
secondpage.html  firstpage.html  testfile.py  # checkbox.css  # demo.css  # radioButton.css  # main.css  # reviewpage.css

151 <div>#pg2>
152   <div>#divContainer>
153     <div>#formCreateAccountForm>
154       <label id="signUp_user_phone_number_label" style="margin-bottom: 2px">
155         Phone Number
156       </label>
157       <span>
158         <input type="text" class="form_input" style="margin-top: 0.3rem; outline: none; border: 1px solid #ccc; width: 100%; height: 1.2em; padding: 0 0.2em;" placeholder="Phone Number" id="signupUser_phonenumber" required />
159       </span>
160       <div>#form_input_error_message>
161     </div>
162   </div>
163   <div>
164     <label id="signup_gender_label">Gender</label>
165     <div>
166       <label class="radio-container" id="male_radio_label">
167         Male
168         <input type="radio" name="radio" id="male_radio_label_button" required />
169         <span class="radio-checkmark"></span>
170       </label>
171     </div>
172     <div>
173       <label class="radio-container" id="female_radio_label">
174         Female
175         <input type="radio" name="radio" id="female_radio_label_button" required />
176         <span class="radio-checkmark"></span>
177       </label>
178     </div>
179     <div>
180       <label class="radio-container" id="other_gender_radio_label">
181         Prefer not to say
182         <input type="radio" name="radio" id="other_gender_radio_label_button" required />
183         <span class="radio-checkmark"></span>
184       </label>
185     </div>
186   </div>
187   <div class="slider">#slide_for_age>
188     <label id="slide_label_for_age">Age</label>
189     <input type="range" id="age_range_slider" min="0" max="100" value="18" oninput="rangeValue.innerText = this.value" />
190     <div id="rangeValue">18</div>
191   </div>
192   <div class="button">#button_signup>
193     <a href="#" onclick="return validateSignupForm('Page1', 'Page2');">Sign Up</a>
194     <input type="submit" value="Sign Up" id="user_signup_button" />
195   </div>

```

Figure 23: Code Part 2 for the Sign Up Form

Figure 24: Code Part 3 for the Sign Up Form

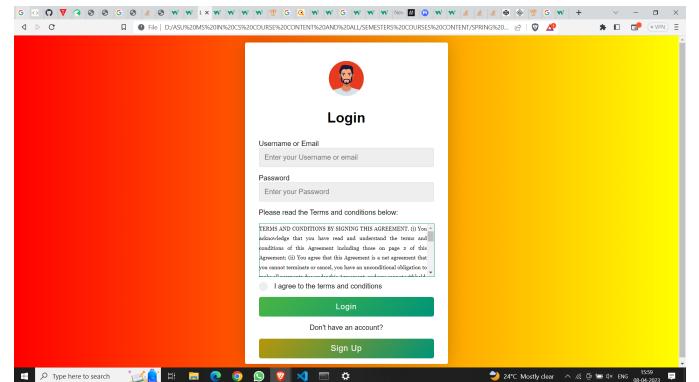


Figure 25: Form Page 2 : Login Form

is configured to match the form's width and a predefined height, allowing the user to quickly scroll through the terms and conditions before logging in to the application. A label "Please read the terms and conditions below" just above the scroll box makes the user aware of the information specified within the scroll area. This radio button is included in the form because the user can only log in if he or she agrees to the application's terms and conditions. To do so, the user must select the radio button with the label "I agree to the terms and conditions," which informs the application that the user has accepted the terms and conditions of the application. The Sign-Up button allows the user to navigate to the Sign-Up form page to create an account for themselves if they do not have an existing account. In contrast, if the user clicks the login button, the application validates the form to see if they have entered their credentials and checked the radio button, and if these validations pass, the application will proceed to the review form page, where the user can provide feedback.

The third page of this application is a review form. I created a review page after being inspired by the comments section of the ASU Canvas Grades Section. In this feedback form, I ask the user to rate this web application based on its UI, functionality implementation, and content. Per the project requirements, each page should have GUI elements such as images, buttons, labels, text boxes, and any of the following: scroll-bar, check-boxes, radio button, slider, list, and drawing elements. Therefore, similar to the first two forms, the feedback form has a relevant image at the top and the title "We appreciate your review!" just below the image. Nevertheless, the feedback form also has a subtitle, "Your review will help me to improve in my future assignments."

```

<div id="Page1" style="display: none;">
    <div class="container">
        <form class="form" id="login" action="#">
            <input type="text" class="form_input" placeholder="Enter your Username or email" id="username_email_input"/>
            <input type="password" class="form_input" placeholder="Enter your Password" id="password_input"/>
        </form>
        <p id="label_for_user_login_terms_and_conditions">
            TERMS AND CONDITIONS BY SIGNING THIS AGREEMENT. (i) You acknowledge that you have read and understand the terms and conditions of this Agreement including those on page 2 of this Agreement; (ii) You agree that this Agreement is a net agreement that you cannot terminate or cancel, you have an unconditional obligation to make all payments due under this Agreement, and you cannot withhold, reduce or cancel any payment for any reason; (iii) You agree the Products will be used for business purposes and in compliance with all applicable laws and regulations; (iv) You acknowledge that if this Agreement is replacing an existing agreement the new Installment Payment may include the balance of that existing agreement and result in a greater aggregate product cost to you; and (v) You agree that by providing a telephone number to a cellular or other wireless device, you are expressly consenting to receiving communications from us, our affiliates and agents (for non-marketing purposes) at that number, including, but not limited to, prerecorded and artificial voice messages, text messages, and calls from automated telephone dialing systems; these calls may incur fees from your cellular provider; and this consent applies to each such telephone number you provide to us now or in the future
        </p>
    </div>

```

Figure 26: Code Part 1 for the Login Form

```

<div class="radio-container" id="radio_label_for_user_login_terms_and_conditions">
    I agree to the terms and conditions
    <input type="radio" name="radio" id="Login_terms_and_conditions" required/>
    <span class="radio-checkmark"></span>
</div>

<div class="button" id="button_login">
    <a href="#" onclick="return validateForm('Page3','Page1', 'userNameEmail_input', 'password_input');">
        <input type="submit" value="Login" id="user_login_button"/>
    </a>
</div>

<div class="button" id="create">
    <p class="form_text" id="label_for_sign_up_on_login_page">
        Don't have an account?
    </p>
    <a href="#" onclick="return show('Page2','Page1');">
        <input type="submit" value="Sign Up" id="sign_up_button_on_login_page"/>
    </a>
</div>

```

Figure 27: Code Part 2 for the Login Form

ments,” written below the title to inform the user why this review or feedback is important to us. Furthermore, again, like the first two forms, this feedback form too has two input text boxes for the username and email fields. The form then has a kind of rating header label, “Please rate your service experience for the following parameters,” It asks the user to rate the applications for the following factors as they see fit. I have included 5 rating sections where each section has different GUI elements taking the rating from the user. The first two rating sections ask the user to rate the application for the following factors “Selection of the testing tool for this assignment?” and “Assessment of the submitted report in terms of the set of grammar or formatting errors, information about the selected testing tool and the description of the Test Cases executed as part of this assignment?”. Both these rating sections contain a star rating format, which corresponds to the usage of drawing elements requirement of the GUI. The third rating section asks the user to rate the application for the following factor ”How did you like the

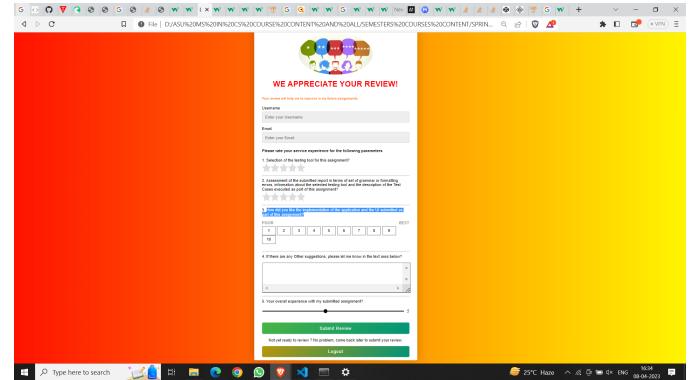


Figure 28: Form Page 3 : Feedback / Review Form

implementation of the application and the UI submitted as part of this assignment?”. Therefore, for this section, I have implemented a scale rating format that contains a group of radio buttons rendered side by side. The fourth rating section consists of a simple text area box in which the user can provide personal feedback on what they liked about the application and what could have been done better regarding the web application as a whole, the test cases developed, and the submitted report. Furthermore, for the last rating section, I have included a slider that asks the user to rate their overall experience with the web application, test cases developed, and report submitted on a scale of 1 to 5. Finally, to finish the Review form, I had to include it in the navigation flow to fulfill the GUI requirements. Therefore, I have added two buttons: Submit Review Button and the Logout Button, when the user can click the Submit Review button after providing all the feedback inputs. The review is submitted, the user is logged out when you click this button, and the application returns to the login page. Similarly, If the user still needs to be ready to leave feedback, he or she can click on the Logout button, redirecting the user to the login page, where he or she can log in once ready to leave feedback.

```

<input type="range" id="review_overall_rating_slider" min="1" max="10" value="5" oninput="review_overall_rating_slider_rangeValue.innerText = this.value" />
<p id="review_overall_rating_slider_rangeValue">5</p>

<div class="button" id="button_submit_review">
    <ul><li><span id="label_for_submit_review_button_on_review_page"> Note you will be logged out as soon as you submit the review. </span>
        <a href="#" onclick="return show('Page1','Page3');">
            <!-- validateForm('Page2','Page1', 'userNameEmail_input', 'password_input'); -->
            <input type="submit" value="Submit Review" id="user_submit_review_button"/>
        </a>
    </ul>
</div>

<div class="button" id="navigation_to_login_page_from_the_review_page">
    <p class="form_text" id="label_for_login_button_on_review_page"> Not yet ready to review? No problem, come back later to submit your review. </p>
    <a href="#" onclick="return show('Page1','Page3');">
        <input type="submit" value="Logout" id="logout_button"/>
    </a>
</div>

```

Figure 29: Code Part 4 for the Feedback Form

Moving on to the next phase of Task 1, using the first version of the application created above as a baseline for the second version, I have made at least three changes in the

```

  o secondpage.html   o firstpage.html   o TestFile3.html   # checkbox.css   # demo.css   # radiobutton.css   # main.css   # reviewpage.css
  o firstpage.html > body > div#page3 > div#review_container.container > form#loginform > div.form_input-group
  195   div id="Page3" style="display: none;">
  196     div class="container" id="review_container">
  197       form class="form" id="login" action="#">
  198         div class="form_group" id="review_homeimage">
  199           div class="form_group" id="review_page_title">
  200             WE APPRECIATE YOUR REVIEW!
  201           
```

Figure 30: Code Part 1 for the Feedback Form

```

  Go Run Terminal Help   • firstpage.html - login-signup-form-master - Visual Studio Code
  o secondpage.html   o firstpage.html > body > div#page3 > div#review_container.container > form#loginform > div.form_input-group > div.form_input-error-message
  240   div style="margin: 10px 0px 10px 0px;" id="review_assessment_2nd_question">
  241     Assessment of the
  242       submitted report in terms of set of grammar or formatting errors, information about the selected
  243       testing tool and the description of the Test Cases executed as part of this
  244       assignment!<br>
  245       .
  246       .
  247       span class="star-rating" id="review_assessment_2nd_question_star_rating">
  248         input type="radio" name="rating2" value="1"></input>
  249         input type="radio" name="rating2" value="2"></input>
  250         input type="radio" name="rating2" value="3"></input>
  251         input type="radio" name="rating2" value="4"></input>
  252         input type="radio" name="rating2" value="5"></input>
  253       
  254     div class="clear"></div>
  255     div class="survey-line" id="review_horizontal_line_two"></div>
  256     div class="survey-line" id="review_horizontal_line_three"></div>
  257     div class="survey-line" id="review_horizontal_line_four"></div>
  258   
```

Figure 31: Code Part 2 for the Feedback Form

```

  Go Run Terminal Help   • firstpage.html - login-signup-form-master - Visual Studio Code
  o secondpage.html   o firstpage.html > body > div#page3 > div#review_container.container > form#loginform > div.form_input-group > div.form_input-error-message
  260   div style="margin: 10px 0px 10px 0px;" id="review_implementation_3rd_question">
  261     How did you like the implementation of the
  262       application and UI submitted as part of this assignment?</label><br><br>
  263     div style="border: 1px solid black; padding: 5px; width: 100%; margin-bottom: 10px;">
  264       .
  265       .
  266       .
  267       .
  268       .
  269       .
  270       .
  271       .
  272       .
  273       .
  274       .
  275       .
  276       .
  277       .
  278       .
  279       .
  280       .
  281       .
  282       .
  283       .
  284       .
  285       .
  286       .
  287       .
  288       .
  289       .
  290       .
  291       .
  292       .
  293       .
  294       .
  295       .
  296       .
  297       .
  298       .
  299       .
  300       .
  301       .
  302       .
  303       .
  304       .
  305       .
  306       .
  307       .
  308       .
  309       .
  310       .
  311       .
  312       .
  313       .
  314     
```

Figure 32: Code Part 3 for the Feedback Form

GUI, such as the size, location, and orientation of the GUI elements. Additionally, I have changed the flow of pages, the link from one page to another. I have mentioned the changes I have made on each page in detail below.

The Sign Up form is the first page of the application where the user can register their account by providing specific information such as their username, email, phone number, gender, age, and password. The first changes made to this form are as follows:

- Image :** Initially, the sign-up form had the small user avatar image at the top and a title “Create Account” just below the image. But in this updated GUI, two changes have been made:

- Location of the Image :** The title “Create Account” has been pushed to the top, and the image has been pushed to be placed below the title.

- Size of the image :** The height and width CSS attributes have been updated to make the image more prominent, as shown in the image below.

- Orientation of the Input Text Box Fields :** All fields were initially implemented in a downward flow. However, four input fields, namely the username, email, password, and confirm password, have been placed in groups of two in the updated GUI.

- Width of the form:** The form’s width was set initially to 400px, but in the updated GUI, the width of the entire form is set to 600px, which is why the form’s width is visible in the image attached below.

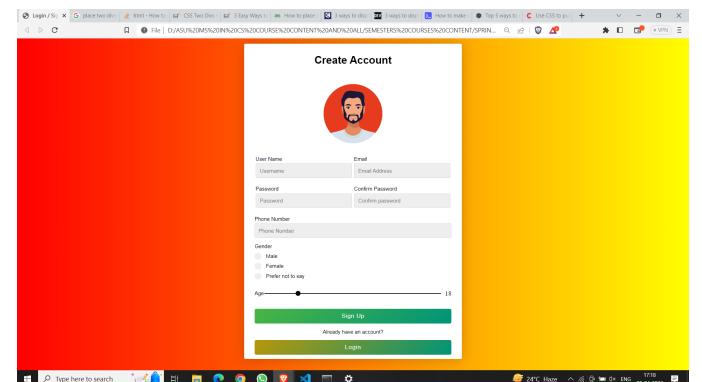


Figure 33: Updated Page 1 : Registration / Sign Up Form

The Login Form is the second page of the application form, where the user can log in by entering their username or email address and password. The changes made to this form are as follows:

- Image :** Initially, the login form had the small user avatar image at the top and a title “Login” just below the image. However, in this updated GUI, two changes have been made:

- Location of the Image :** The title “Login” has been pushed to the top and the image has been pushed to be placed below the title.

```

188 <div id="Page2">
189   <div class="container" id="sign_up_form_page_container">
190     <form class="form" id="createAccountForm">
191       
192       <h1 class="form_title" id="sign_up_create_account_header">
193         Create Account
194       </h1>
195       
196       <div class="form_message form_message_error"></div>
197       <div id="input_section_parent">
198         <div class="form_input_group" id="input_section_parent_child_left">
199           <span style="margin-bottom: 5px">
200             <label id="sign_up_username_label"> User Name </label>
201             </span>
202             <input type="text" id="signup_username" class="form_input" style="margin-top: 0.3rem" autofocus
203               placeholder="Username" required />
204             <div class="form_input_error_message"></div>
205           </div>
206           <div class="form_input_group" id="input_section_parent_child_right">
207             <span style="margin-bottom: 5px">
208               <label id="sign_up_email_label"> Email </label>
209             </span>
210             <input type="text" class="form_input" style="margin-top: 0.3rem" autofocus
211               placeholder="Email Address" id="sign_up_email" required />
212             <div class="form_input_error_message"></div>
213           </div>
214         </div>
215         <div id="input_section_parent">
216           <div class="form_input_group" id="input_section_parent_child_left">
217             <span style="margin-bottom: 5px">
218               <label id="sign_up_password_label"> Password </label>
219             </span>
220             <input type="password" class="form_input" style="margin-top: 0.3rem" autofocus
221               placeholder="Password" id="signup_password" required />
222             <div class="form_input_error_message"></div>
223           </div>
224           <div class="form_input_group" id="input_section_parent_child_right">
225             <span style="margin-bottom: 5px">
226               <label id="sign_up_confirm_password_label"> Confirm Password </label>
227             </span>
228             <input type="password" class="form_input" style="margin-top: 0.3rem" autofocus
229               placeholder="Confirm password" id="signup_confirm_password" required />
230           </div>
231         </div>
232       </div>
233       <div id="input_group" style="margin-top: 20px">
234         <span style="margin-bottom: 5px">
235           <label id="sign_up_gender_label"> Phone Number </label>
236         </span>
237         <input type="text" class="form_input" style="margin-top: 0.3rem" autofocus
238           placeholder="Phone Number" id="sign_up_phonenumber" required />
239         <div class="form_input_error_message"></div>
240       </div>
241       <div id="signup_gender_label" style="margin-top: 20px">
242         <label class="radio-container" id="male_radio_label">
243           Male
244           <input type="radio" name="radio" id="male_radio_label_button" required />
245           <span class="radio-checkmark"></span>
246         </label>
247         </div>
248         <div id="signup_gender_label" style="margin-top: 20px">
249           <label class="radio-container" id="female_radio_label">
250             Female
251             <input type="radio" name="radio" id="female_radio_label_button" required />
252             <span class="radio-checkmark"></span>
253           </label>
254         </div>
255         <div id="signup_gender_label" style="margin-top: 20px">
256           <label class="radio-container" id="other_gender_radio_label">
257             Prefer not to say
258             <input type="radio" name="radio" id="other_gender_radio_label_button" required />
259             <span class="radio-checkmark"></span>
260           </label>
261         </div>
262       </div>
263       <div class="slide" id="slide_range_for_gpa">
264         <label id="slide_label_for_gpa" for="gpa">GPA</label>
265         <input type="range" id="gpa_range_slider" min="0" max="100" value="18" />
266         <span id="slide_value_for_gpa" style="margin-left: 10px;">18
267       </div>
268     </div>
269   </div>
270 </div>

```

Figure 34: Code Part 1 for the Updated Sign Up Page

```

51       placeholder="Confirm password" id="signup_confirm_password" required />
52       <div class="form_input_error_message"></div>
53     </div>
54   </div>
55   <div class="form_input_group" style="margin-bottom: 25px">
56     <span style="margin-bottom: 5px">
57       <label id="sign_up_email_label" style="margin-bottom: 2px">
58         Phone Number
59       </label>
60     </span>
61     <input type="text" class="form_input" style="margin-top: 0.3rem" autofocus
62       placeholder="Phone Number" id="sign_up_phonenumber" required />
63     <div class="form_input_error_message"></div>
64   </div>
65   <div id="signup_gender_label" style="margin-top: 20px">
66     <label class="radio-container" id="male_radio_label">
67       Male
68       <input type="radio" name="radio" id="male_radio_label_button" required />
69       <span class="radio-checkmark"></span>
70     </label>
71   </div>
72   <div id="signup_gender_label" style="margin-top: 20px">
73     <label class="radio-container" id="female_radio_label">
74       Female
75       <input type="radio" name="radio" id="female_radio_label_button" required />
76       <span class="radio-checkmark"></span>
77     </label>
78   </div>
79   <div id="signup_gender_label" style="margin-top: 20px">
80     <label class="radio-container" id="other_gender_radio_label">
81       Prefer not to say
82       <input type="radio" name="radio" id="other_gender_radio_label_button" required />
83       <span class="radio-checkmark"></span>
84     </label>
85   </div>
86   <div class="slide" id="slide_range_for_gpa">
87     <label id="slide_label_for_gpa" for="gpa">GPA</label>
88     <input type="range" id="gpa_range_slider" min="0" max="100" value="18" />
89     <span id="slide_value_for_gpa" style="margin-left: 10px;">18
90   </div>
91 </div>

```

Figure 35: Code Part 2 for the Updated Sign Up Page

```

198   <p id="rangeValue">18</p>
199   </div>
200
201   <div class="button" id="button_signup">
202     <a href="#" onclick="return validateSignUpForm('Page1', 'Page2');">
203       <!-- validateForm('Page2', 'Page1', 'userNameOrEmail_input', 'password_input'); -->
204       <input type="submit" value="Sign Up" id="user_signup_button" />
205     </a>
206   </div>
207
208   <div class="button" id="navigation_to_login_page">
209     <p class="form_text" id="label_for_login_button_on_sign_up_page">
210       Already have an account?
211     </p>
212     <a href="#" onclick="return show('Page1', 'Page2');">
213       <input type="submit" value="Login" id="login_button_on_sign_up_page" />
214     </a>
215   </div>
216 </form>
217 </div>
218 </div>

```

Figure 36: Code Part 3 for the Updated Sign Up Page

- Size of the image :** As shown in the image below, the height and width CSS attributes have been updated to make the image larger.
- Spacing between the Labels and Input Text Boxes :** I have added a “margin-top” CSS property to the input field text boxes to add some spacing between them and their respective labels.
- Scroll Area Box :** The height of the Scroll Area Box has

increased compared to its initial height.

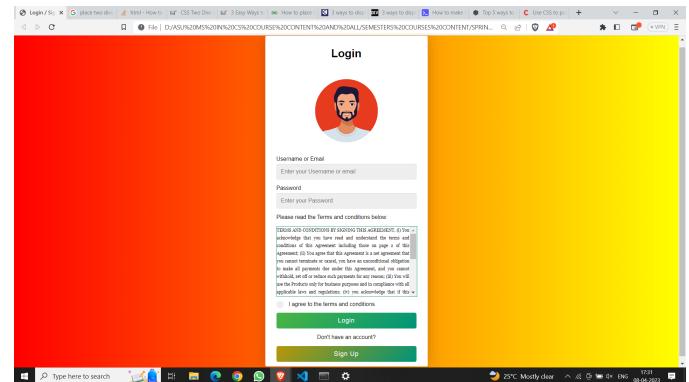


Figure 37: Updated Page 2 : Login Form

```

51       placeholder="Confirm password" id="signup_confirm_password" required />
52       <div class="form_input_error_message"></div>
53     </div>
54   </div>
55   <div class="form_input_group" style="margin-bottom: 25px">
56     <span style="margin-bottom: 5px">
57       <label id="sign_up_email_label" style="margin-bottom: 2px">
58         Phone Number
59       </label>
60     </span>
61     <input type="text" class="form_input" style="margin-top: 0.3rem" autofocus
62       placeholder="Phone Number" id="sign_up_phonenumber" required />
63     <div class="form_input_error_message"></div>
64   </div>
65   <div id="signup_gender_label" style="margin-top: 20px">
66     <label class="radio-container" id="male_radio_label">
67       Male
68       <input type="radio" name="radio" id="male_radio_label_button" required />
69       <span class="radio-checkmark"></span>
70     </label>
71   </div>
72   <div id="signup_gender_label" style="margin-top: 20px">
73     <label class="radio-container" id="female_radio_label">
74       Female
75       <input type="radio" name="radio" id="female_radio_label_button" required />
76       <span class="radio-checkmark"></span>
77     </label>
78   </div>
79   <div id="signup_gender_label" style="margin-top: 20px">
80     <label class="radio-container" id="other_gender_radio_label">
81       Prefer not to say
82       <input type="radio" name="radio" id="other_gender_radio_label_button" required />
83       <span class="radio-checkmark"></span>
84     </label>
85   </div>
86   <div class="slide" id="slide_range_for_gpa">
87     <label id="slide_label_for_gpa" for="gpa">GPA</label>
88     <input type="range" id="gpa_range_slider" min="0" max="100" value="18" />
89     <span id="slide_value_for_gpa" style="margin-left: 10px;">18
90   </div>
91 </div>

```

Figure 38: Code Part 1 for the Updated Login Page

```

64   <div id="Page3" style="display: none;">
65     <div class="container" id="login_form_container">
66       <form class="form" id="login_form" action="#">
67          -->
68         <h1 class="form_title" id="login_page_title">Login</h1>
69         
70         <div class="form_input_group">
71           <span style="margin-bottom: 5px">
72             <label id="user_nameor_email_label"> Username or Email </label>
73           </span>
74           <input type="text" class="form_input" style="margin-top: 0.3rem" autofocus
75             placeholder="Enter your Username or email" id="user_nameor_email_input" required />
76           <div class="form_input_error_message"></div>
77         </div>
78         <div class="form_input_group">
79           <span style="margin-bottom: 5px">
80             <label id="user_password_label"> Password </label>
81           </span>
82           <input type="password" class="form_input" style="margin-top: 0.3rem" autofocus
83             placeholder="Enter your Password" id="password_input" required />
84           <div class="form_input_error_message"></div>
85         </div>
86         <div id="label_for_login_terms_and_conditions">
87           <p>Please read the Terms and conditions below:</p>
88           <p>I agree to the terms and conditions of this Agreement including those on page 2 of this Agreement; (ii) You agree that this Agreement is a net agreement that you cannot terminate or cancel, you have an unconditional obligation to make all payments due under this Agreement, and you cannot withhold, set off or reduce such payments for any reason; (iii) You will use the Products only for business purposes and in compliance with all applicable laws and regulations; (iv) you acknowledge that this Agreement replaces an existing agreement, the new instrument, Payment may include the balance of that existing agreement and result in a greater aggregate product cost to you; and (v) You agree that by providing a telephone number to a cellular or other wireless device, you are expressly consenting to receiving communications from us, our affiliates and agents (for non-marketing purposes) at that number, including, but not limited to, pre-recorded, artificial, recorded, telemarketing, and calls from automated telephone dialing systems; these calls may incur fees from your cellular provider; and this consent applies to each such telephone number you provide to us now or in the future</p>
89       </div>
90     </div>
91   </div>

```

Figure 39: Code Part 2 for the Updated Login Page

The third page of this application is a review form. I created a review page after being inspired by the comments section of the ASU Canvas Grades Section. In this feedback form, I ask the user to rate this web application based on its UI, functionality implementation, and content. This form has undergone the following modifications:

- Third Rating Section:** I have done three changes in this section.
 - Interchanged the labels;** I have interchanged the positions of the POOR and the BEST labels.
 - Furthermore, to match the labels, I have also reordered the rating values from 10 - 1 instead of 1 - 10.
- Text Area Box :** I increased the number of rows for the text area box, which increased its height compared to its initial height.
- Flow Change:** Since it was stated in the assignment description that one flow change was required, instead of navigating to the Login page as was implemented initially, when the user clicks on the Logout button at the bottom, the application will now navigate to the Sign Up page.

The screenshot shows the updated review form. At the top, there is a logo with four stylized human figures and the text 'WE APPRECIATE YOUR REVIEW!'. Below this, there is a message: 'Your review will help us to improve in my future assignments.' There are fields for 'Username' and 'Email'. A large text area box is labeled 'Please rate your service experience for the following parameters:'. Below this, there are two sections of rating scales. The first section is for '1. Overall feel of the assignment?' with a scale from 1 (poor) to 10 (best). The second section is for '2. Assessment of the submitted report in terms of use of grammar or spelling, punctuation, and overall presentation?'. Both sections include a note: 'Grade included as part of the final grade for this assignment'. At the bottom, there is a text area box for '4. If there are any Other suggestions, please let me know in the text area below?' followed by a 'Submit Review' button and a 'Logout' link.

Figure 40: Updated Page 3 : Feedback / Review Form

```
secondpage.html ● TestFile.py 3 # checkbox.css # demo.css # radiobutton.css # main.css # reviewpage.css # scrollarea.css # style.css
secondpage.html > body > div#Page3 > div#review_container.container > form#loginForm > div.form_input-group
249 <div id="Page3" style="display: none;">
250   <div class="container" id="review_container">
251     <form class="form" id="review_form" method="post">
252       <input type="radio" name="rating" value="1"/>
253       <input type="radio" name="rating" value="2"/>
254       <input type="radio" name="rating" value="3"/>
255       <input type="radio" name="rating" value="4"/>
256       <input type="radio" name="rating" value="5"/>
257     </div>
258   </div>
259 </div>
260 <div class="clear"></div>
261 <div class="survey-hn" id="review_horizontal_line_one">
```

Figure 41: Code Part 1 for the Updated Review Form

```
secondpage.html ● TestFile.py 3 # checkbox.css # demo.css # radiobutton.css # main.css # reviewpage.css # scrollarea.css # style.css
secondpage.html > body > div#Page3 > div#review_container.container > form#loginForm > div.form_input-group
249 <div>
250   <label style="margin: 10px 0px 0px 0px;" id="review_assessment_2nd_question_star_rating">
251     Selection of the testing tool for this assignment?</label>
252   <input type="radio" name="rating" value="1"/>
253   <input type="radio" name="rating" value="2"/>
254   <input type="radio" name="rating" value="3"/>
255   <input type="radio" name="rating" value="4"/>
256   <input type="radio" name="rating" value="5"/>
257 </div>
258 <div class="clear"></div>
259 <div class="survey-hn" id="review_horizontal_line_two">
```

Figure 42: Code Part 2 for the Updated Review Form

```
secondpage.html ● TestFile.py 3 # checkbox.css # demo.css # radiobutton.css # main.css # reviewpage.css # scrollarea.css # style.css
secondpage.html > body > div#Page3 > div#review_container.container > form#loginForm > div.form_input-group
249 <div>
250   <label value="1">
251     <input type="radio" name="rating" value="10"/>
252     <label style="width:100%;"></label>
253   </label>
254   <label value="9">
255     <input type="radio" name="rating" value="9"/>
256     <label style="width:100%;"></label>
257   </label>
258   <label value="8">
259     <input type="radio" name="rating" value="8"/>
260     <label style="width:100%;"></label>
261   </label>
262   <label value="7">
263     <input type="radio" name="rating" value="7"/>
264     <label style="width:100%;"></label>
265   </label>
266 </div>
267 <div class="clear"></div>
268 <div class="survey-hn" id="review_horizontal_line_three">
```

Figure 43: Code Part 3 for the Updated Review Form

For Task 2 of the project, selecting a suitable GUI test automation tool was required to record test cases and report the test results. I have scoured the internet for articles [24, 25, 26, and 27] and information about the different GUI test automation tools available in the market. After extensive research on different GUI test automation tools available in the market, it was concluded that Selenium would be the best choice for this project.

Selenium is a popular open-source library for automating web browsers, including web application testing, scraping, and repetitive task automation. The Selenium tool provides a set of libraries and tools to simulate user interac-

```

 ① secondpage.html ② testfilepy ③ checkbox.cs ④ demo.css ⑤ radiobutton.css ⑥ main.css ⑦ reviewpage.css ⑧ scrollarea
⑨ secondpage.html > ⑩ body > ⑪ div#review_container.container > ⑫ form#login_form > ⑬ div.form_input_group
332   <div class="clear"></div>
333   <hr class="survey_hr" id="review_horizontal_line_four">
334   </div>
335
336   <div id="review_page_overall_experience">5. Your overall experience with my submitted assignment?</div>
337   <div class="slider"> ⑭ <input type="range" id="review_overall_rating_slider" min="1" max="10" value="5" ⑮
338   <input type="range" id="review_overall_rating_slider" value="Submit Review" id="user_submit_review_button" />
339   <p id="review_overall_rating_slider_rangeValue">5</p>
340   </div>
341
342   <div class="button" id="button_submit_review">
343     <input type="button" value="Submit Review" id="user_submit_review_button" />
344     <!--> you will be logged out as soon as you submit the review.
345     </div> -->
346     <a href="#" onclick="return show('Page1', 'Page3');">
347       <!-- validateForm('Page2', 'Page1', 'usernameOrEmail_input', 'password_input'); -->
348       <input type="submit" value="Submit Review" id="user_submit_review_button" />
349     </a>
350   </div>
351
352   <div class="button" id="navigation_to_login_page_from_the_review_page">
353     <input type="button" value="Logout" id="logout_button_on_review_page" />
354     Not yet ready to review? No problem, come back later to submit your review.
355   </div>
356   <a href="#" onclick="return show('Page1', 'Page3');">
357     <input type="submit" value="Logout" id="logout_button_on_review_page" />
358   </a>
359   </div>
360
361   </div>
362
363   </form>
364 </div>
365
366

```

Figure 44: Code Part 4 for the Updated Review Form

tions with web browsers, which can help test web applications and other tasks such as web scraping and web automation. The key features of Selenium, such as programming language support, browser support, fast test execution, parallelism, and record and playback features, were considered. However, the most attractive factor of Selenium was the Selenium Web-Driver. Selenium Web-Driver is a tool for automating web application testing using a programmatic interface to control a web browser. This means that developers and testers can use Selenium Web-Driver to create functional tests that automate browser actions like clicking buttons, filling out forms, and navigating between pages, as well as a set of APIs for interacting with web browsers like Chrome, Firefox, and Safari. Therefore, Selenium Web-Driver was an ideal tool for testing this GUI application as the GUI contains elements such as images, buttons, labels, text boxes, scroll-bar, check-boxes, radio buttons, slider, list, and drawing elements.

Selenium WebDriver Architecture

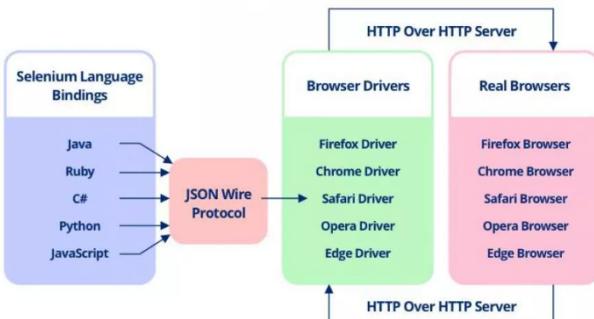


Figure 45: Selenium Web-Driven architecture

To fully understand the Selenium Web-Driven tool, the official documentation[28] and other reliable websites[29, 32, and 33] were consulted, and the scope, purpose, and us-

age of Selenium Web-Driven were studied. It was found that the scope of Selenium Web-Driven is to automate functional tests that verify the behavior of a web application, while the primary purpose of Selenium with Python is to automate web testing by controlling the browser and making assertions about the web application's behavior. Finally, after a thorough understanding of the tool, a working example of Selenium Web-Driven was included in the report with a detailed explanation of how the tool works and how it meets its scope and primary purpose. This example demonstrates how Selenium Web-Driven can automate the user login process with a web browser and verify the expected behavior of the web application. We can run this test by using Selenium Web-Driven to automate it repeatedly to ensure that the login functionality is working correctly without manual testing.

In order to perform UI testing using Selenium, it was necessary to set up the appropriate tools and dependencies. This involved installing Selenium, an IDE, and a browser driver. Firstly, I visited the official download page[30] for Selenium and selected Python as the appropriate language binding for this project. The reason for selecting Python was because it is a popular and influential programming language, and it has excellent support for Selenium testing. With Python selected, I downloaded the appropriate version of Selenium for my system.

Next, I ensured an IDE was installed for writing and executing the test cases. I used Visual Studio Code (VSCode) for this project, which I had already installed on my system. IDE was chosen because it is a robust and feature-rich text editor that offers a range of features to aid development, such as syntax highlighting, code completion, and debugging tools for various programming languages, including Python. It also provides an extensive range of plugins and extensions that can enhance the development process, such as auto-complete and debugging features.

Finally, I needed a browser driver to drive real browsers with automated UI testing scripts. As mentioned in the previous section, different browser drivers are specific to different browsers. However, for this project, I will only test this project in Chrome. The reason for selecting Chrome-Driven was that it is an open-source tool that provides excellent support for automating the testing of web applications and works seamlessly with Selenium Web-Driven. Therefore, I downloaded Chrome-Driven from the official website[31].

With all the dependencies installed, I created the Python file for testing. Following all the steps in detail mentioned in the official documentation[28] and other reliable websites[34, 35, and 36] to setup the project for testing, I first imported all the required dependencies within the Python file, including Selenium and Chrome-Driven. I then loaded the two HTML files containing the two versions of the GUI application we had created earlier. I could interact with the GUI elements by loading these files into the script and performing tests using the Selenium Web-Driven. With the initial setup completed, I was able to begin creating test cases following each every step mentioned in the official documentation[28] and other reliable websites[29, 34, 35, and 36] and to cover all aspects of the GUI application, which includes its correct location, the size of the element, the

content of the GUI element, and the link's correctness (flow from one page to another).

As a result of these guidelines, I primarily created three types of Test Cases.

- Category 1:** This category of test cases checks for the **existence, location and the size** of the element using either it's unique ID or CLASS_NAME.

- The command “**I = browser.find_element(By.ID, “signup_homeimage”)**” checks for the existence of the element.
- The commands “**location = I.location**”, “**x = location.get(“x”)**” and “**y = location.get(“y”)**” retrieve the x and y coordinates of the element.
- The commands “**size = I.size**”, and “**w, h = size[“width”], size[“height”]**” retrieve the width and the height of the element.

- Category 2:** This category of test cases checks for the **Content of the GUI element** using either it's unique ID or CLASS_NAME.

- This is possible with the help of the command “**s = I.text**”. This command is used to get the label content of the GUI element.

- Category 3:** This category of test cases checks for the **Flow of the GUI** from one link to another using either it's unique ID or CLASS_NAME.

- This is implemented by using the command “**browser.find_element(By.ID, “user_signup_button”).click()**”.
- The application is then navigated to the next page after clicking a button, so an additional code line is implemented to check for an element present on the next page to validate whether or not the application has indeed navigated.

```

28     def testcase100(self):
29         try:
30             I = browser.find_element(By.ID, "signup_homeimage") code line to find the existence of the element
31             s = I.text
32             location = I.location code line to find the location of the element
33             size = I.size code line to find the size of the element
34             w, h = I.size[“width”], I.size[“height”] code line to find the width and the height of the element
35             tw = 100
36             th = 100
37             xcov = 466
38             ycov = 48
39             x = location.get(“x”) code line to get the x-coordinate
40             y = location.get(“y”) and the y-coordinate of the element
41             # print(“width : ”, w)
42             # print(“height : ”, h)
43             # print(“xcov : ”, x)
44             # print(“ycov : ”, y)
45             self.assertEqual(w, tw)
46             self.assertEqual(s, value)
47             self.assertEqual((x, y), (xcov, ycov))
48             print(“Testcase 1 Passed”)
49         except NoSuchElementException:
50             print(“Element does not exist”)
51

```

Figure 46: Code Example of the Test Case Category 1

```

100    def testcase101(self):
101        try:
102            I = browser.find_element(By.ID, "signup_usernameEmail") code line to find the existence of the element in the GUI
103            s = I.text
104            value = “username”
105            self.assertEqual(s, value)
106            print(“Testcase 2 Passed”)
107        except NoSuchElementException:
108            print(“Element does not exist”)
109
110

```

Figure 47: Code Example of the Test Case Category 2

To end the report, as per the project requirements, I have provided a detailed evaluation of Selenium and the Selenium

```

834 ✓
835 
836     def testcase102(self):
837         try:
838             username = browser.find_element(By.ID, “signupUsernameEmail”)
839             username.clear()
840             username.send_keys(“entering my username here as a value”)
841
842             email = browser.find_element(By.ID, “signupUsernameEmail”)
843             email.clear()
844             email.send_keys(“entering my username here as a value”)
845
846             password = browser.find_element(By.ID, “signupPassword”)
847             password.clear()
848             password.send_keys(“entering my password here as a value”)
849
850             confirm_password = browser.find_element(By.ID, “signupConfirmPassword”)
851             confirm_password.clear()
852             confirm_password.send_keys(“entering my password here as a value”)
853
854             phone_number = browser.find_element(By.ID, “signupPhoneNumber”)
855             phone_number.clear()
856             phone_number.send_keys(“8135630372”)
857
858             browser.find_element(By.ID, “male_radio_label_button”).click()
859             browser.find_element(By.ID, “user_signup_button”).click() find the sign up button and click on it
860
861             i = 0
862             while i < 200:
863                 i = i + 1
864
865                 l = browser.find_element(By.ID, “homeimage”) check if the image on the login page which is the next
866                 s = “round”
867                 self.assertEqual(s, “round”)
868                 print(“Testcase 41 Passed”)
869
870             except:
871                 print(“Element does not exist”)
872

```

Figure 48: Code Example of the Test Case Category 3

Web-Driven tool in terms set of features and functionalities provided, type of coverage, reuse of test cases, test results produced, ease of usage, and type of GUI elements that can be tested. After working with Selenium and the Selenium Web-Driven tool firsthand for this project, I have gained a deeper understanding of the tool and its capabilities.

In terms of features and functionalities, Selenium provides a wide range of features for automating web application testing, such as browser automation, web element interaction, and various assertion methods for verifying expected behavior. The Selenium Web-Driven tool, in particular, provides a set of APIs for interacting with web browsers and simulating user interactions with web elements. Additionally, Selenium supports multiple programming languages, making it a versatile tool for test automation.

Regarding the type of coverage, Selenium Web-Driven is ideal for functional testing and can be used to verify the behavior of web applications. Selenium also supports unit testing frameworks, making it suitable for integration and regression testing.

One of the significant benefits of using Selenium Web-Driven is the ability to reuse test cases. As Selenium Web-Driven is a programmatic interface, test cases can be written in a modular and reusable way, allowing them to be run on multiple browsers and operating systems. This feature of Selenium makes it an efficient tool for web application testing, saving time and resources by reducing the effort required in creating new test cases for each browser.

Selenium produces comprehensive test results, including pass/fail status, screenshots of failures, and log files performance metrics. The test results produced by Selenium are highly detailed and informative, making it easy to interpret and analyze, as Selenium provides detailed logs and reporting capabilities. These results provide insight into the application's behavior and can be used to identify and diagnose issues in the web application. The test results are provided in various formats, such as XML, HTML, and JSON, making it easier for testers to analyze and interpret the results. Additionally, Selenium provides visual confirmation of test steps, making it easier to track test progress and locate errors.

In terms of ease of usage, Selenium Web-Driven has a rel-

atively steep learning curve, especially for beginners unfamiliar with programming languages, as it depends on the user's experience with programming languages. However, the official Selenium documentation provides a comprehensive guide on how to use the tool. Additionally, the Selenium community provides ample resources for troubleshooting and support. Therefore, once familiar with the tool, writing test cases can be straightforward and efficient.

Selenium Web-Driver can test various GUI elements, including images, buttons, labels, text boxes, scroll-bars, check-boxes, radio buttons, sliders, lists, and drawing elements. This makes it a versatile tool for testing various web applications.

Based on my experience with the tool, Selenium Web-Driver is a powerful and versatile tool for automating web application testing. Its extensive features and functionalities, type of coverage, ability to reuse test cases, complete test results, ease of usage, and type of GUI elements that can be tested make it an efficient and effective tool for web application testing.

Number of Developed Test Cases

For the whole application, I have developed a total of 109 test cases based on the initial GUI.

Test Cases Results for the Version-1 GUI

All the 109 test cases have successfully passed after a successful execution of the test cases.

```
File: D:\USU\ME IN CS COURSE CONTENT AND ALL SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\ASSIGNMENT 5\login-signup-form-master.py
DevTools listening on ws://127.0.0.1:57832/devtools/browser/2f9bdad9-dba9-4e3b-aec1-53917732d0ed
Testsuite: All 109 tests in 4.1ms
Testcase 1 Passed
Testcase 2 Passed
Testcase 3 Passed
Testcase 4 Passed
Testcase 5 Passed
Testcase 6 Passed
Testcase 7 Passed
Testcase 8 Passed
Testcase 9 Passed
Testcase 10 Passed
Testcase 11 Passed
Testcase 12 Passed
Testcase 13 Passed
Testcase 14 Passed
Testcase 15 Passed
Testcase 16 Passed
Testcase 17 Passed
Testcase 18 Passed
Testcase 19 Passed
Testcase 20 Passed
Testcase 21 Passed
Testcase 22 Passed
Testcase 23 Passed
Testcase 24 Passed
Testcase 25 Passed
Testcase 26 Passed
Testcase 27 Passed
Testcase 28 Passed
Testcase 29 Passed
Testcase 30 Passed
Testcase 31 Passed
Testcase 32 Passed
Testcase 33 Passed
Testcase 34 Passed
Testcase 35 Passed
Testcase 36 Passed
Testcase 37 Passed
Testcase 38 Passed
Testcase 39 Passed
Testcase 40 Passed
Testcase 41 Passed
Testcase 42 Passed
Testcase 43 Passed
Testcase 44 Passed
Testcase 45 Passed
Testcase 46 Passed
Testcase 47 Passed
Testcase 48 Passed
Testcase 49 Passed
Testcase 50 Passed
Testcase 51 Passed
Testcase 52 Passed
Testcase 53 Passed
Testcase 54 Passed
Testcase 55 Passed
Testcase 56 Passed
Testcase 57 Passed
Testcase 58 Passed
Testcase 59 Passed
Testcase 60 Passed
Testcase 61 Passed
Testcase 62 Passed
Testcase 63 Passed
Testcase 64 Passed
Testcase 65 Passed
Testcase 66 Passed
Testcase 67 Passed
Testcase 68 Passed
Testcase 69 Passed
Testcase 70 Passed
Testcase 71 Passed
Testcase 72 Passed
Testcase 73 Passed
Testcase 74 Passed
Testcase 75 Passed
Testcase 76 Passed
Testcase 77 Passed
Testcase 78 Passed
Testcase 79 Passed
Testcase 80 Passed
Testcase 81 Passed
Testcase 82 Passed
Testcase 83 Passed
Testcase 84 Passed
Testcase 85 Passed
Testcase 86 Passed
Testcase 87 Passed
Testcase 88 Passed
Testcase 89 Passed
Testcase 90 Passed
Testcase 91 Passed
Testcase 92 Passed
Testcase 93 Passed
Testcase 94 Passed
Testcase 95 Passed
Testcase 96 Passed
Testcase 97 Passed
Testcase 98 Passed
Testcase 99 Passed
Testcase 100 Passed
109 tests in 4.1ms
```

Figure 49: Results of Test Cases on the Version-1 GUI - 1

Test Cases Results for the Version-2 GUI

Out of the developed 109 test cases, 55 test cases failed when executed on the updated GUI.

Understanding of the Results of the Test Cases for the Updated GUI

- Test Cases with respect to the Sign Up Page:** For the Sign-Up page, 41 test cases have been created. Because of the changes made to this page, such as changing the position of the image and title and changing the format of the form, all of the test cases about location and size, totaling 28 cases, have failed. Twelve of the thirteen test

```
File: D:\USU\ME IN CS COURSE CONTENT AND ALL SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\ASSIGNMENT 5\login-signup-form-master.py
DevTools listening on ws://127.0.0.1:63095/devtools/browser/598ba29a-caff-4ef5-a8f7-2e86c018870
Testsuite: All 109 tests in 4.1ms
Testcase 1 Passed
Testcase 2 Passed
Testcase 3 Passed
Testcase 4 Passed
Testcase 5 Passed
Testcase 6 Passed
Testcase 7 Passed
Testcase 8 Passed
Testcase 9 Passed
Testcase 10 Passed
Testcase 11 Passed
Testcase 12 Passed
Testcase 13 Passed
Testcase 14 Passed
Testcase 15 Passed
Testcase 16 Passed
Testcase 17 Passed
Testcase 18 Passed
Testcase 19 Passed
Testcase 20 Passed
Testcase 21 Passed
Testcase 22 Passed
Testcase 23 Passed
Testcase 24 Passed
Testcase 25 Passed
Testcase 26 Passed
Testcase 27 Passed
Testcase 28 Passed
Testcase 29 Passed
Testcase 30 Passed
Testcase 31 Passed
Testcase 32 Passed
Testcase 33 Passed
Testcase 34 Passed
Testcase 35 Passed
Testcase 36 Passed
Testcase 37 Passed
Testcase 38 Passed
Testcase 39 Passed
Testcase 40 Passed
Testcase 41 Passed
Testcase 42 Passed
Testcase 43 Passed
Testcase 44 Passed
Testcase 45 Passed
Testcase 46 Passed
Testcase 47 Passed
Testcase 48 Passed
Testcase 49 Passed
Testcase 50 Passed
Testcase 51 Passed
Testcase 52 Passed
Testcase 53 Passed
Testcase 54 Passed
Testcase 55 Passed
Testcase 56 Passed
Testcase 57 Passed
Testcase 58 Passed
Testcase 59 Passed
Testcase 60 Passed
Testcase 61 Passed
Testcase 62 Passed
Testcase 63 Passed
Testcase 64 Passed
Testcase 65 Passed
Testcase 66 Passed
Testcase 67 Passed
Testcase 68 Passed
Testcase 69 Passed
Testcase 70 Passed
Testcase 71 Passed
Testcase 72 Passed
Testcase 73 Passed
Testcase 74 Passed
Testcase 75 Passed
Testcase 76 Passed
Testcase 77 Passed
Testcase 78 Passed
Testcase 79 Passed
Testcase 80 Passed
Testcase 81 Passed
Testcase 82 Passed
Testcase 83 Passed
Testcase 84 Passed
Testcase 85 Passed
Testcase 86 Passed
Testcase 87 Passed
Testcase 88 Passed
Testcase 89 Passed
Testcase 90 Passed
Testcase 91 Passed
Testcase 92 Passed
Testcase 93 Passed
Testcase 94 Passed
Testcase 95 Passed
Testcase 96 Passed
Testcase 97 Passed
Testcase 98 Passed
Testcase 99 Passed
Testcase 100 Passed
109 tests in 4.1ms
```

Figure 50: Results of Test Cases on the Version-1 GUI - 2

```
File: D:\USU\ME IN CS COURSE CONTENT AND ALL SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\ASSIGNMENT 5\login-signup-form-master.py
DevTools listening on ws://127.0.0.1:63095/devtools/browser/598ba29a-caff-4ef5-a8f7-2e86c018870
Testsuite: All 109 tests in 4.1ms
Testcase 1 Passed
Testcase 2 Passed
Testcase 3 Passed
Testcase 4 Passed
Testcase 5 Passed
Testcase 6 Passed
Testcase 7 Passed
Testcase 8 Passed
Testcase 9 Passed
Testcase 10 Passed
Testcase 11 Passed
Testcase 12 Passed
Testcase 13 Passed
Testcase 14 Passed
Testcase 15 Passed
Testcase 16 Passed
Testcase 17 Passed
Testcase 18 Passed
Testcase 19 Passed
Testcase 20 Passed
Testcase 21 Passed
Testcase 22 Passed
Testcase 23 Passed
Testcase 24 Passed
Testcase 25 Passed
Testcase 26 Passed
Testcase 27 Passed
Testcase 28 Passed
Testcase 29 Passed
Testcase 30 Passed
Testcase 31 Passed
Testcase 32 Passed
Testcase 33 Passed
Testcase 34 Passed
Testcase 35 Passed
Testcase 36 Passed
Testcase 37 Passed
Testcase 38 Passed
Testcase 39 Passed
Testcase 40 Passed
Testcase 41 Passed
Testcase 42 Passed
Testcase 43 Passed
Testcase 44 Passed
Testcase 45 Passed
Testcase 46 Passed
Testcase 47 Passed
Testcase 48 Passed
Testcase 49 Passed
Testcase 50 Passed
Testcase 51 Passed
Testcase 52 Passed
Testcase 53 Passed
Testcase 54 Passed
Testcase 55 Passed
Testcase 56 Passed
Testcase 57 Passed
Testcase 58 Passed
Testcase 59 Passed
Testcase 60 Passed
Testcase 61 Passed
Testcase 62 Passed
Testcase 63 Passed
Testcase 64 Passed
Testcase 65 Passed
Testcase 66 Passed
Testcase 67 Passed
Testcase 68 Passed
Testcase 69 Passed
Testcase 70 Passed
Testcase 71 Passed
Testcase 72 Passed
Testcase 73 Passed
Testcase 74 Passed
Testcase 75 Passed
Testcase 76 Passed
Testcase 77 Passed
Testcase 78 Passed
Testcase 79 Passed
Testcase 80 Passed
Testcase 81 Passed
Testcase 82 Passed
Testcase 83 Passed
Testcase 84 Passed
Testcase 85 Passed
Testcase 86 Passed
Testcase 87 Passed
Testcase 88 Passed
Testcase 89 Passed
Testcase 90 Passed
Testcase 91 Passed
Testcase 92 Passed
Testcase 93 Passed
Testcase 94 Passed
Testcase 95 Passed
Testcase 96 Passed
Testcase 97 Passed
Testcase 98 Passed
Testcase 99 Passed
Testcase 100 Passed
109 tests in 4.1ms
```

Figure 51: Results of Test Cases on the Version-1 GUI - 3

```
File: D:\USU\ME IN CS COURSE CONTENT AND ALL SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\ASSIGNMENT 5\login-signup-form-master.py
DevTools listening on ws://127.0.0.1:63095/devtools/browser/598ba29a-caff-4ef5-a8f7-2e86c018870
Testsuite: All 109 tests in 4.1ms
Testcase 1 Passed
Testcase 2 Passed
Testcase 3 Passed
Testcase 4 Passed
Testcase 5 Passed
Testcase 6 Passed
Testcase 7 Passed
Testcase 8 Passed
Testcase 9 Passed
Testcase 10 Passed
Testcase 11 Passed
Testcase 12 Passed
Testcase 13 Passed
Testcase 14 Passed
Testcase 15 Passed
Testcase 16 Passed
Testcase 17 Passed
Testcase 18 Passed
Testcase 19 Passed
Testcase 20 Passed
Testcase 21 Passed
Testcase 22 Passed
Testcase 23 Passed
Testcase 24 Passed
Testcase 25 Passed
Testcase 26 Passed
Testcase 27 Passed
Testcase 28 Passed
Testcase 29 Passed
Testcase 30 Passed
Testcase 31 Passed
Testcase 32 Passed
Testcase 33 Passed
Testcase 34 Passed
Testcase 35 Passed
Testcase 36 Passed
Testcase 37 Passed
Testcase 38 Passed
Testcase 39 Passed
Testcase 40 Passed
Testcase 41 Passed
Testcase 42 Passed
Testcase 43 Passed
Testcase 44 Passed
Testcase 45 Passed
Testcase 46 Passed
Testcase 47 Passed
Testcase 48 Passed
Testcase 49 Passed
Testcase 50 Passed
Testcase 51 Passed
Testcase 52 Passed
Testcase 53 Passed
Testcase 54 Passed
Testcase 55 Passed
Testcase 56 Passed
Testcase 57 Passed
Testcase 58 Passed
Testcase 59 Passed
Testcase 60 Passed
Testcase 61 Passed
Testcase 62 Passed
Testcase 63 Passed
Testcase 64 Passed
Testcase 65 Passed
Testcase 66 Passed
Testcase 67 Passed
Testcase 68 Passed
Testcase 69 Passed
Testcase 70 Passed
Testcase 71 Passed
Testcase 72 Passed
Testcase 73 Passed
Testcase 74 Passed
Testcase 75 Passed
Testcase 76 Passed
Testcase 77 Passed
Testcase 78 Passed
Testcase 79 Passed
Testcase 80 Passed
Testcase 81 Passed
Testcase 82 Passed
Testcase 83 Passed
Testcase 84 Passed
Testcase 85 Passed
Testcase 86 Passed
Testcase 87 Passed
Testcase 88 Passed
Testcase 89 Passed
Testcase 90 Passed
Testcase 91 Passed
Testcase 92 Passed
Testcase 93 Passed
Testcase 94 Passed
Testcase 95 Passed
Testcase 96 Passed
Testcase 97 Passed
Testcase 98 Passed
Testcase 99 Passed
Testcase 100 Passed
109 tests in 4.1ms
```

Figure 52: Results of Test Cases on the Version-2 GUI - 1

cases passed because they were checking for the content of the GUI elements, and since I had not changed any of the element's content, all of them passed. Finally, the last test case that passed was about the GUI flow, and it passed because the flow from the Sign Up page to the Login page was not hampered or modified in this case.

- Test Cases with respect to the Login Page:** For the Login page, 20 test cases have been created because of the changes made to this page, such as changing the position of the image and title and adding some space

```

[1] 88 Passed
[1] 89 Passed
[1] 90 Passed
[1] 91 Passed
[1] 92 Passed
[1] 93 Passed
[1] 94 Passed
[1] 95 Passed
[1] 96 Passed
[1] 97 Passed
[1] 98 Passed
[1] 99 Passed
[1] 100 Passed
[1] 101 Passed
[1] 102 Passed
[1] 103 Passed
[1] 104 Passed
[1] 105 Passed
[1] 106 Passed
[1] 107 Passed
[1] 108 Passed
[1] 109 Passed

[1] 110 testcases00 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 100, in testcases00
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (224, 225) != (100, 100)
First differing element: 0
  100
  224
+ 100
  100

[1] 111 testcases01 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 101, in testcases01
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (224, 225) != (100, 100)
First differing element: 0
  100
  224
+ 100
  100

[1] 112 testcases02 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 102, in testcases02
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 113 testcases03 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 103, in testcases03
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 114 testcases04 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 104, in testcases04
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 115 testcases05 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 105, in testcases05
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

```

Figure 53: Results of Test Cases on the Version-2 GUI - 2

```

[1] 116 testcases07 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 107, in testcases07
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  100

[1] 117 testcases08 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 108, in testcases08
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  100

[1] 118 testcases09 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 109, in testcases09
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  100

[1] 119 testcases10 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 110, in testcases10
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  100

```

Figure 55: Results of Test Cases on the Version-2 GUI - 4

```

[1] 120 testcases12 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 112, in testcases12
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 121 testcases13 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 113, in testcases13
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 122 testcases14 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 114, in testcases14
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 123 testcases15 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 115, in testcases15
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 124 testcases16 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 116, in testcases16
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 125 testcases17 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 117, in testcases17
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

```

Figure 54: Results of Test Cases on the Version-2 GUI - 3

```

[1] 126 testcases18 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 118, in testcases18
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 127 testcases19 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 119, in testcases19
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 128 testcases20 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 120, in testcases20
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

```

Figure 56: Results of Test Cases on the Version-2 GUI - 5

between the labels and fields, all of the test cases about location and size, totaling 13 cases, have failed. Six of the seven test cases passed because they were checking for the content of the GUI elements, and since I had not changed any of the element's content, all of them passed. Finally, the last test case that passed was about the GUI flow, and it passed because the flow from the Login page to the Feedback form page was not hampered or modified in this case.

• Test Cases with respect to the Feedback Form Page: The Feedback Form page has 49 test cases. However, because I did not change the position of the image at the top of this page, all subsequent location and size test cases did not fail and instead passed. The failed test cases are associated with the “POOR” and “BEST” labels, which were relocated. Following these test cases, the text area box test case failed because its height was increased. As a result, some of the following 11 test cases concerning location and size failed since I have changed the flow where when the user submits the feedback form, the application is rerouted to the sign-up page instead of the login page per the initial implementation.

As a result, we have 55 failing test cases, most related to location and size, except for one that failed due to a change in the application’s flow. Only two of the 54 passed test cases check for the flow of the GUI application, while the majority

```

[1] 129 testcases21 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 121, in testcases21
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 130 testcases22 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 122, in testcases22
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 131 testcases23 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 123, in testcases23
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

[1] 132 testcases24 [__main___.TestStringMethods]
Traceback (most recent call last):
  File "testfile_py_1.py", line 124, in testcases24
    self.assertEqual((x,y), (100, 100))
AssertionError: tuples differ: (210, 212) != (100, 100)
First differing element: 0
  100
  210
+ 100
  212

```

Figure 57: Results of Test Cases on the Version-2 GUI - 6

of the 54 passed test cases check for the content of the GUI elements.

Figure 58: Results of Test Cases on the Version-2 GUI - 7

```
Aravind Pratap(Aravind)

=====
#file: testcase019 (__main__.TestStringMethods)

Traceback (most recent call last):
  File "TestFile.py", line 198, in testcase019
    self.assertEqual((w, h), (1m, 1n))
AssertionError: tuples differ: (100, 45) != (400, 45)
First differing element 0:
<100>
><400>

=====
#file: testcase020 (__main__.TestStringMethods)

Traceback (most recent call last):
  File "TestFile.py", line 200, in testcase020
    self.assertEqual((w, h), (1m, 1n))
AssertionError: tuples differ: (100, 45) != (400, 45)
First differing element 0:
<100>
><400>

=====
#file: testcase021 (__main__.TestStringMethods)

Traceback (most recent call last):
  File "TestFile.py", line 202, in testcase021
    self.assertEqual((w, h), (1m, 1n))
AssertionError: tuples differ: (100, 45) != (400, 45)
First differing element 0:
<100>
><400>

=====
#file: testcase022 (__main__.TestStringMethods)

Traceback (most recent call last):
  File "TestFile.py", line 204, in testcase022
    self.assertEqual((w, h), (1m, 1n))
AssertionError: tuples differ: (100, 45) != (400, 45)
First differing element 0:
<100>
><400>
```

Figure 61: Results of Test Cases on the Version-2 GUI - 10

```
Acquire:Priority(0x0000)

=====
full: testcases25 [__main__, TestStringMethods]
Traceback (most recent call last):
  File "Testfile.py", line 591, in testcases25
    assertEqual(x, y)
AssertionError: tuples differ: (400, 18) != (400, 18)
First differing element 0:
<400>
<400>
+ (400, 18)
+
- (400, 18)

=====
full: testcases27 [__main__, TestStringMethods]
Traceback (most recent call last):
  File "Testfile.py", line 603, in testcases27
    self.assertEqual(x, y)
AssertionError: tuples differ: (567, 748) != (467, 748)
First differing element 0:
<567>
<467>
+ (567, 748)
+
- (467, 748)

=====
full: testcases28 [__main__, TestStringMethods]
Traceback (most recent call last):
  File "Testfile.py", line 600, in testcases28
    assertEqual(x, y)
AssertionError: tuples differ: (400, 62) != (400, 62)
First differing element 0:
<400>
<400>
+ (400, 62)
```

Figure 59: Results of Test Cases on the Version-2 GUI - 8

```
File "Textfile.py", line 82, in testcase130
    self.assertEqual((w, h), (400, 16))
AssertionError: tuples differ: (400, 16) != (400, 45)
First differing element 0:
400
|
v
(400, 45)

File "Textfile.py", line 82, in testcase131
    self.assertEqual((w, h), (400, 16))
AssertionError: tuples differ: (400, 16) != (400, 45)
First differing element 0:
400
|
v
(400, 45)

File "Textfile.py", line 82, in testcase132
    self.assertEqual((w, h), (400, 45))
AssertionError: tuples differ: (400, 45) != (400, 18)
First differing element 0:
400
|
v
(400, 18)

File "Textfile.py", line 82, in testcase133
    self.assertEqual((w, h), (400, 45))
AssertionError: tuples differ: (400, 45) != (400, 18)
First differing element 0:
400
|
v
(400, 18)
```

Figure 62: Results of Test Cases on the Version-2 GUI -11

```
Acorns Prompt (Acorns)

full testcase29 [__main__.TestStringMethods]
-----
Traceback (most recent call last):
  File "TestFile.py", line 625, in testcase29
    self.assertEqual((x, y), (w, h))
AssertionError: tuples differ: (210, 765) != (310, 797)
First differing element 0:
210
310
+ (210, 765)
+ (310, 797)

full testcase31 [__main__.TestStringMethods]
-----
Traceback (most recent call last):
  File "TestFile.py", line 637, in testcase31
    self.assertEqual((x, h), (w, h))
AssertionError: tuples differ: (338, 2) != (338, 2)
First differing element 0:
338
338
+ (338, 2)
+
+ (338, 2)

-----
full testcase32 [__main__.TestStringMethods]
-----
AssertionError: tuples differ: (680, 795) != (680, 795)
First differing element 0:
680
680
+ (680, 795)
+ (680, 795)

-----
```

Figure 60: Results of Test Cases on the Version-2 GUI - 9

```
src/main.cpp:10:1: error: no matching function for call to 'main()'
main()
^~~~~~
```

Figure 63: Results of Test Cases on the Version-2 GUI - 12

```

[1] Anconda Prompt (Anconda)
>>> python TestFile.py
... (output truncated)

```

Figure 64: Results of Test Cases on the Version-2 GUI - 13

```

[1] Anconda Prompt (Anconda)
>>> python TestFile.py
... (output truncated)

```

Figure 67: Results of Test Cases on the Version-2 GUI - 16

```

[1] Anconda Prompt (Anconda)
>>> python TestFile.py
... (output truncated)

```

Figure 65: Results of Test Cases on the Version-2 GUI - 14

```

[1] Anconda Prompt (Anconda)
>>> python TestFile.py
... (output truncated)

```

Figure 68: Results of Test Cases on the Version-2 GUI - 17

```

[1] Anconda Prompt (Anconda)
>>> python TestFile.py
... (output truncated)

```

Figure 66: Results of Test Cases on the Version-2 GUI - 15

```

[1] Anconda Prompt (Anconda)
>>> python TestFile.py
... (output truncated)

```

Figure 69: Results of Test Cases on the Version-2 GUI - 18

References

- Unit testing
- Unit Test Frameworks by Paul Hamill”
- What is Unit Testing and Why Developer Should Learn It
- Why Is Unit Testing Important in Software Development?
- Jest Framework
- Unit Tutorial — Testing Framework for Java
- What are the differences between Jest and JUnit?
- Heap Sort Algorithm
- Jest Crash Course - Unit Testing in JavaScript
- Jest Tutorial: Complete Guide to Jest Testing
- WHAT IS DESIGN OF EXPERIMENTS (DOE)?
- Pairwise Testing Or All-Pairs Testing Tutorial With Tools And Examples
- Pairwise Testing Guide: How To Perform Pairwise Testing
- EXTENDED COMBINATORIAL TESTING USING GRAPH ALGORITHMS AND APACHE SPARK
- Czerwonka (2006)
- PICT tool from Microsoft
- Pairwise Pict Online
- Pairwise Independent Combinatorial Testing
- Code coverage
- PMD Source Code Analyzer Project
- Collecting Unit Test Coverage in IntelliJ IDEA
- How To Use PMD Code Analyzer With IntelliJ
- Sample Code for the assignment
- 40 Best UI Testing Tools And Techniques
- 11 Best Automated UI Testing Tools In 2022
- 19 BEST Automation Testing Tools (2023)
- 9 Best GUI Testing Tools in Software Testing
- WebDriver
- A Complete Guide to Selenium UI testing
- Selenium Official Downloads
- Chrome-Driver Official Downloads Selenium Webdriver
- Selenium Selenium & Selenium WebDriver
- Selenium WebDriver Tutorial - A Comprehensive Guide to WebDriver Automation Selenium with Python
- The Selenium Browser Automation Project
- Prof. James Collofello, CSE 565 Lecture Videos, 2023
- Prof. Ayca Tuzmen, CSE 565 Lectures, 2023
- CSE 565 - Software Verification Validation and Testing - Assignment 1 - Unit testing & Unit testing Frameworks by Kedar Sai Nadh Reddy Kanchi

```

[231] : (210, 1277)
[232] : (210, 1331)

=====
[233] : Anconda Prompt (Anconda)
[234] : 
[235] : File "testfile.py", line 201, in testcase200
[236] :     self.assertEqual(x, y, "Error: user")
[237] : AssertionError: Tuples differ: (210, 1276) != (210, 1176)
[238] : First differing element:
[239] :   1276
[240] :   1176
[241] : 
[242] : 
[243] : File "testfile.py", line 203, in testcase202
[244] :     self.assertEqual(x, y, "Error: user")
[245] : AssertionError: Tuples differ: (210, 1216) != (210, 1176)
[246] : First differing element:
[247] :   1216
[248] :   1176
[249] : 
[250] : 
[251] : File "testfile.py", line 210, in testcase203
[252] :     self.assertEqual(x, y, "Error: user")
[253] : AssertionError: Tuples differ: (210, 1216) != (210, 1176)
[254] : First differing element:
[255] :   1216
[256] :   1176
[257] : 
[258] : 
[259] : File "testfile.py", line 211, in testcase204
[260] :     self.assertEqual(x, y, "Error: user")
[261] : AssertionError: Tuples differ: (210, 1277) != (210, 1231)
[262] : First differing element:
[263] :   1277
[264] :   1231
[265] : 
[266] : 
[267] : File "testfile.py", line 215, in testcase205
[268] :     self.assertEqual(x, y, "Error: user")
[269] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[270] : First differing element:
[271] :   1311
[272] :   1266
[273] : 
[274] : 
[275] : Ran 100 tests in 0.339s
[276] : 
[277] : FAILED (failures=55)

```

Figure 70: Results of Test Cases on the Version-2 GUI - 19

```

[276] : Anconda Prompt (Anconda)
[277] : 
[278] : File "testfile.py", line 201, in testcase206
[279] :     self.assertEqual(x, y, "Error: user")
[280] : AssertionError: Tuples differ: (210, 1277) != (210, 1231)
[281] : First differing element:
[282] :   1277
[283] :   1231
[284] : 
[285] : 
[286] : File "testfile.py", line 202, in testcase207
[287] :     self.assertEqual(x, y, "Error: user")
[288] : AssertionError: Tuples differ: (210, 1216) != (210, 1231)
[289] : First differing element:
[290] :   1216
[291] :   1231
[292] : 
[293] : 
[294] : File "testfile.py", line 203, in testcase208
[295] :     self.assertEqual(x, y, "Error: user")
[296] : AssertionError: Tuples differ: (210, 1216) != (210, 1231)
[297] : First differing element:
[298] :   1216
[299] :   1231
[300] : 
[301] : 
[302] : File "testfile.py", line 204, in testcase209
[303] :     self.assertEqual(x, y, "Error: user")
[304] : AssertionError: Tuples differ: (210, 1277) != (210, 1231)
[305] : First differing element:
[306] :   1277
[307] :   1231
[308] : 
[309] : 
[310] : File "testfile.py", line 210, in testcase206
[311] :     self.assertEqual(x, y, "Error: user")
[312] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[313] : First differing element:
[314] :   1311
[315] :   1266
[316] : 
[317] : 
[318] : File "testfile.py", line 211, in testcase207
[319] :     self.assertEqual(x, y, "Error: user")
[320] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[321] : First differing element:
[322] :   1311
[323] :   1266
[324] : 
[325] : 
[326] : File "testfile.py", line 212, in testcase208
[327] :     self.assertEqual(x, y, "Error: user")
[328] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[329] : First differing element:
[330] :   1311
[331] :   1266
[332] : 
[333] : 
[334] : File "testfile.py", line 213, in testcase209
[335] :     self.assertEqual(x, y, "Error: user")
[336] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[337] : First differing element:
[338] :   1311
[339] :   1266
[340] : 
[341] : 
[342] : Ran 100 tests in 0.339s
[343] : 
[344] : FAILED (failures=55)

```

Figure 71: Results of Test Cases on the Version-2 GUI - 20

- CSE 565 - Software Verification Validation and Testing - Assignment 3 - Design of Experiments & Pairwise Testing by Kedar Sai Nadh Reddy Kanchi
- CSE 565 - Software Verification Validation and Testing - Assignment 4 - Decision Code Coverage by Kedar Sai Nadh Reddy Kanchi
- CSE 565 - Software Verification Validation and Testing - Assignment 5 - Graphical User Interface Testing by Kedar Sai Nadh Reddy Kanchi

```

[343] : Anconda Prompt (Anconda)
[344] : 
[345] : File "testfile.py", line 201, in testcase206
[346] :     self.assertEqual(x, y, "Error: user")
[347] : AssertionError: Tuples differ: (210, 1277) != (210, 1231)
[348] : First differing element:
[349] :   1277
[350] :   1231
[351] : 
[352] : 
[353] : File "testfile.py", line 202, in testcase207
[354] :     self.assertEqual(x, y, "Error: user")
[355] : AssertionError: Tuples differ: (210, 1216) != (210, 1231)
[356] : First differing element:
[357] :   1216
[358] :   1231
[359] : 
[360] : 
[361] : File "testfile.py", line 203, in testcase208
[362] :     self.assertEqual(x, y, "Error: user")
[363] : AssertionError: Tuples differ: (210, 1216) != (210, 1231)
[364] : First differing element:
[365] :   1216
[366] :   1231
[367] : 
[368] : 
[369] : File "testfile.py", line 204, in testcase209
[370] :     self.assertEqual(x, y, "Error: user")
[371] : AssertionError: Tuples differ: (210, 1277) != (210, 1231)
[372] : First differing element:
[373] :   1277
[374] :   1231
[375] : 
[376] : 
[377] : File "testfile.py", line 210, in testcase206
[378] :     self.assertEqual(x, y, "Error: user")
[379] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[380] : First differing element:
[381] :   1311
[382] :   1266
[383] : 
[384] : 
[385] : File "testfile.py", line 211, in testcase207
[386] :     self.assertEqual(x, y, "Error: user")
[387] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[388] : First differing element:
[389] :   1311
[390] :   1266
[391] : 
[392] : 
[393] : File "testfile.py", line 212, in testcase208
[394] :     self.assertEqual(x, y, "Error: user")
[395] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[396] : First differing element:
[397] :   1311
[398] :   1266
[399] : 
[400] : 
[401] : File "testfile.py", line 213, in testcase209
[402] :     self.assertEqual(x, y, "Error: user")
[403] : AssertionError: Tuples differ: (210, 1311) != (210, 1266)
[404] : First differing element:
[405] :   1311
[406] :   1266
[407] : 
[408] : 
[409] : Ran 100 tests in 0.339s
[410] : 
[411] : FAILED (failures=55)

```

Figure 72: Results of Test Cases on the Version-2 GUI- 21

Implemented Heap Sort Algorithm For Assignment 1

1/21/23, 11:39 AM

heapsort.js

```
1 // JavaScript program for implementation of Heap Sort
2
3 const heapsort_algorithm = {
4
5     heapsort_code: function(arr) {
6         var N = arr.length;
7
8         // Build heap (rearrange array)
9         for (var i = Math.floor(N / 2) - 1; i >= 0; i--)
10            heapify(arr, N, i);
11
12        // One by one extract an element from heap
13        for (var i = N - 1; i > 0; i--) {
14            // Move current root to end
15            var temp = arr[0];
16            arr[0] = arr[i];
17            arr[i] = temp;
18
19            // call max heapify on the reduced heap
20            heapify(arr, i, 0);
21        }
22        return arr;
23    }
24
25}
26
27
28 // To heapify a subtree rooted with node i which is an index in arr[].
29 // n is size of heap
30 const heapify = (arr, N, i) => {
31
32     var largest = i; // Initialize largest as root
33     var l = 2 * i + 1; // left = 2*i + 1
34     var r = 2 * i + 2; // right = 2*i + 2
35
36     // If left child is larger than root
37     if (l < N && arr[l] > arr[largest])
38         largest = l;
39
40     // If right child is larger than largest so far
41     if (r < N && arr[r] > arr[largest])
42         largest = r;
43
44     // If largest is not root
45     if (largest != i) {
46         var swap = arr[i];
47         arr[i] = arr[largest];
48         arr[largest] = swap;
49
50         // Recursively heapify the affected sub-tree
51         heapify(arr, N, largest);
52     }
53 }
54
55 module.exports = heapsort_algorithm;
56
```

```
1 const heapsort = require("./heapsort");
2
3 test('heapsort test case 1', () => {
4     expect(heapsort.heapsort_code([12, 11, 13, 5, 6, 7])).toEqual([ 5, 6, 7, 11, 12,
13 ]);
5 });
6
7 test('heapsort test case 2', () => {
8     expect(heapsort.heapsort_code([4,1,3,9,7])).toEqual([ 1,3,4,7,9 ]);
9 });
10
11 test('heapsort test case 3', () => {
12     expect(heapsort.heapsort_code([10,9,8,7,6,5,4,3,2,1])).toEqual([ 1,
2,3,4,5,6,7,8,9,10 ]);
13 });
14
15 test('heapsort test case 4', () => {
16     expect(heapsort.heapsort_code([])).toEqual([]);
17 });
18
19 test('heapsort test case 5', () => {
20     expect(heapsort.heapsort_code(["banana", "apple", "mango", "pineapple",
"orange"])).toEqual(["apple", "banana", "mango", "orange", "pineapple"]);
21 });
22
23 test('heapsort test case 6', () => {
24     expect(heapsort.heapsort_code(["CAB", "ACB", "BCA", "ABC", "CBA",
"BAC"])).toEqual(["ABC", "ACB", "BAC", "BCA", "CAB", "CBA"]);
25 });
26
27 test('heapsort test case 7', () => {
28     expect(heapsort.heapsort_code([5,3,3,1,5,3,2])).toEqual([1,2,3,3,3,5,5]);
29 });
30
31 test('heapsort test case 8', () => {
32
    expect(heapsort.heapsort_code(['5','3','3','1','5','3','2'])).toEqual(['1','2','3',
'3','3','5','5']);
33 });
34
35 test('heapsort test case 9', () => {
36     expect(heapsort.heapsort_code(["M4dkbH6NjJ", "1XSRJpa60H", "hxWwGbcMJM",
"PjMgcuKxsO", "AWVQ1I3pZs", "6I4v0Dbmk8", "LjeMBnEqr1", "lnJnShEBE6", "RJzCi0UcRE",
"IOvB93CeP9"])).toEqual([
"1XSRJpa60H", "6I4v0Dbmk8", "AWVQ1I3pZs", "IOvB93CeP9", "LjeMBnEqr1",
"M4dkbH6NjJ", "PjMgcuKxsO", "RJzCi0UcRE", "hxWwGbcMJM", "lnJnShEBE6"]);
37 });
38
39 test('heapsort test case 10 - checking to fail', () => {
40     expect(heapsort.heapsort_code([4,1,3,9,7])).toEqual([ 1,7,4,3,9 ]);
41 });
42 })
```

Decision Code Coverage Summary for Vending Machine Code - Assignment 4

3/24/23, 8:56 PM

Coverage Report > <empty package>

Current scope: [all classes](#) | <empty package name>**Coverage Summary for Package: <empty package>**

Package	Class, %	Method, %	Branch, %	Line, %
<empty package>	66.7% (2/3)	72.7% (8/11)	75% (15/20)	70% (35/50)

Class ▲	Class, %	Method, %	Branch, %	Line, %
StaticAnalysis	0% (0/1)	0% (0/3)	0% (0/4)	0% (0/15)
VendingMachine	100% (1/1)	100% (2/2)	93.8% (15/16)	100% (24/24)
VendingMachineTest	100% (1/1)	100% (6/6)		100% (11/11)

generated on 2023-03-22 20:30

Static Source Code Analysis for the StaticAnalysis Code using the PMD tool - Assignment 4 PMD report

Problems found

#	File	Line	Problem
1	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	4	All classes, interfaces, enums and annotations must belong to a named package
2	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	4	Class comments are required
3	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	5	All methods are static. Consider using a utility class instead. Alternatively, you could add a private constructor or make the class abstract to silence this warning.
4	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	6	Parameter 'args' is not assigned and could be declared final
5	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	6	Public method and constructor comments are required
6	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	9	Local variable 'value' could be declared final
7	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	10	System.out.println is used
8	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	13	Public method and constructor comments are required
9	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	15	Avoid unused local variables such as 'weight'.
10	D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING	15	Local variable 'weight' could be declared final

2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

11 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

16 [Avoid unused local variables
such as 'length'.](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

12 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

16 [Local variable 'length' could be
declared final](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

13 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

19 [Parameter 'length' is not assigned
and could be declared final](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

14 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

19 [Parameter 'product' is not
assigned and could be declared
final](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

15 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

19 [Parameter 'weight' is not assigned
and could be declared final](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

16 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

19 [Public method and constructor
comments are required](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

17 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

21 [Found 'DD'-anomaly for variable
'cost' \(lines '21'-'26'\).](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

18 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

21 [Found 'DD'-anomaly for variable
'cost' \(lines '21'-'30'\).](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

19 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

21 [The initializer for variable 'cost'
is never used \(overwritten on
lines 26 and 30\).](#)

D:\ASU MS IN CS COURSE CONTENT AND
ALL\SEMESTERS COURSES CONTENT\SPRING

20 2023\CSE 565 Software Verification Validation and
Testing\ASSIGNMENTS\Assignment 4 - Decision Code
Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java

22 [Found 'DD'-anomaly for variable
'output' \(lines '22'-'34'\).](#)

D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
21 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	22	Found 'DD'-anomaly for variable 'output' (lines '22'-'36').
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
22 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	22	The initializer for variable 'output' is never used (overwritten on lines 34 and 36).
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
23 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	24	Avoid using Literals in Conditional Statements
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
24 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	24	Use equals() to compare object references.
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
25 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	24	Use equals() to compare strings instead of '==' or '!='
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
26 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	33	Avoid using Literals in Conditional Statements
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
27 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	34	Avoid using if...else statements without curly braces
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
28 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	34	This statement should have braces
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
29 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	36	Avoid using if...else statements without curly braces
D:\ASU MS IN CS COURSE CONTENT AND ALL\SEMESTERS COURSES CONTENT\SPRING		
30 2023\CSE 565 Software Verification Validation and Testing\ASSIGNMENTS\Assignment 4 - Decision Code Coverage\CSE 565 SVVT Assignment 4\src\StaticAnalysis.java	36	This statement should have braces

```
1 import time
2 import unittest
3
4 from selenium import webdriver
5 from selenium.common import NoSuchElementException
6 from selenium.webdriver.common.by import By
7
8 # from pytime import time
9
10
11 browser = webdriver.Chrome()
12
13 # Please update this to the file location on your System
14 # browser.get(
15 #     "file:///D://ASU MS IN CS COURSE CONTENT AND ALL//SEMESTERS COURSES
16 #     CONTENT//SPRING 2023//CSE 565 Software Verification Validation and
17 #     Testing//ASSIGNMENTS//ASSIGNMENT 5//login-signup-form-master//firstpage.html"
18 # )
19
20
21
22
23 class TestStringMethods(unittest.TestCase):
24     maxDiff = None
25
26     # test cases for the registration page
27
28     def testcase100(self):
29         try:
30             l = browser.find_element(By.ID, "signup_homeimage")
31             s = l.text
32             location = l.location
33             size = l.size
34             w, h = size["width"], size["height"]
35             tw = 100
36             th = 100
37             xcor = 460
38             ycor = 48
39             x = location.get("x")
40             y = location.get("y")
41             # print("width - ", w)
42             # print("height - ", h)
43             # print("xcor - ", x)
44             # print("ycor - ", y)
45             self.assertEqual((w, h), (tw, th))
46             self.assertEqual((x, y), (xcor, ycor))
47             print("Testcase 1 Passed")
48
49         except NoSuchElementException:
50             print("Element does not exist Test Case 1 Failed")
51
52     def testcase101(self):
53         try:
54             l = browser.find_element(By.ID, "signup_create_account_header")
```

```
55         s = l.text
56         location = l.location
57         size = l.size
58         w, h = size["width"], size["height"]
59         tw = 400
60         th = 36
61         xcor = 310
62         ycor = 169
63         x = location.get("x")
64         y = location.get("y")
65         # print("width - ", w)
66         # print("height - ", h)
67         # print("xcor - ", x)
68         # print("ycor - ", y)
69         self.assertEqual((w, h), (tw, th))
70         self.assertEqual((x, y), (xcor, ycor))
71         print("Testcase 2 Passed")
72
73     except NoSuchElementException:
74         print("Element does not exist Test Case 2 Failed")
75
76 def testcase102(self):
77     try:
78         l = browser.find_element(By.ID, "signup_userNameorEmail")
79         s = l.text
80         location = l.location
81         size = l.size
82         w, h = size["width"], size["height"]
83         tw = 81
84         th = 18
85         xcor = 310
86         ycor = 237
87         x = location.get("x")
88         y = location.get("y")
89         # print("width - ", w)
90         # print("height - ", h)
91         # print("xcor - ", x)
92         # print("ycor - ", y)
93         self.assertEqual((w, h), (tw, th))
94         self.assertEqual((x, y), (xcor, ycor))
95         print("Testcase 3 Passed")
96
97     except NoSuchElementException:
98         print("Element does not exist Test Case 3 Failed")
99
100    def testcase103(self):
101        try:
102            l = browser.find_element(By.ID, "signup_userNameorEmail")
103            s = l.text
104            value = "User Name"
105            self.assertEqual(s, value)
106            print("Testcase 4 Passed")
107
108        except NoSuchElementException:
109            print("Element does not exist Test Case 4 Failed")
110
111    def testcase104(self):
112        try:
113            l = browser.find_element(By.ID, "signupUsername")
114            s = l.text
```

```
115     location = l.location
116     size = l.size
117     w, h = size["width"], size["height"]
118     tw = 400
119     th = 44
120     xcor = 310
121     ycor = 261
122     x = location.get("x")
123     y = location.get("y")
124     # print("width - ", w)
125     # print("height - ", h)
126     # print("xcor - ", x)
127     # print("ycor - ", y)
128     self.assertEqual((w, h), (tw, th))
129     self.assertEqual((x, y), (xcor, ycor))
130     print("Testcase 5 Passed")
131
132 except NoSuchElementException:
133     print("Element does not exist Test Case 5 Failed")
134
135 def testcase105(self):
136     try:
137         l = browser.find_element(By.ID, "signup_user_Email_label")
138         s = l.text
139         location = l.location
140         size = l.size
141         w, h = size["width"], size["height"]
142         tw = 40
143         th = 18
144         xcor = 310
145         ycor = 321
146         x = location.get("x")
147         y = location.get("y")
148         # print("width - ", w)
149         # print("height - ", h)
150         # print("xcor - ", x)
151         # print("ycor - ", y)
152         self.assertEqual((w, h), (tw, th))
153         self.assertEqual((x, y), (xcor, ycor))
154         print("Testcase 6 Passed")
155
156     except NoSuchElementException:
157         print("Element does not exist Test Case 6 Failed")
158
159 def testcase106(self):
160     try:
161         l = browser.find_element(By.ID, "signup_user_Email_label")
162         s = l.text
163         value = "Email"
164         self.assertEqual(s, value)
165         print("Testcase 7 Passed")
166
167     except NoSuchElementException:
168         print("Element does not exist Test Case 7 Failed")
169
170 def testcase107(self):
171     try:
172         l = browser.find_element(By.ID, "signupUsername_email")
173         s = l.text
174         location = l.location
```

```
175     size = l.size
176     w, h = size["width"], size["height"]
177     tw = 400
178     th = 44
179     xcor = 310
180     ycor = 344
181     x = location.get("x")
182     y = location.get("y")
183     # print("width - ", w)
184     # print("height - ", h)
185     # print("xcor - ", x)
186     # print("ycor - ", y)
187     self.assertEqual((w, h), (tw, th))
188     self.assertEqual((x, y), (xcor, ycor))
189     print("Testcase 8 Passed")
190
191 except NoSuchElementException:
192     print("Element does not exist Test Case 8 Failed")
193
194 def testcase108(self):
195     try:
196         l = browser.find_element(By.ID, "signup_user_password_label")
197         s = l.text
198         location = l.location
199         size = l.size
200         w, h = size["width"], size["height"]
201         tw = 70
202         th = 18
203         xcor = 310
204         ycor = 404
205         x = location.get("x")
206         y = location.get("y")
207         # print("width - ", w)
208         # print("height - ", h)
209         # print("xcor - ", x)
210         # print("ycor - ", y)
211         self.assertEqual((w, h), (tw, th))
212         self.assertEqual((x, y), (xcor, ycor))
213         print("Testcase 9 Passed")
214
215     except NoSuchElementException:
216         print("Element does not exist Test Case 9 Failed")
217
218 def testcase109(self):
219     try:
220         l = browser.find_element(By.ID, "signup_user_password_label")
221         s = l.text
222         value = "Password"
223         self.assertEqual(s, value)
224         print("Testcase 10 Passed")
225
226     except NoSuchElementException:
227         print("Element does not exist Test Case 10 Failed")
228
229 def testcase110(self):
230     try:
231         l = browser.find_element(By.ID, "signup_password")
232         s = l.text
233         location = l.location
234         size = l.size
```

```
235     w, h = size["width"], size["height"]
236     tw = 400
237     th = 44
238     xcor = 310
239     ycor = 427
240     x = location.get("x")
241     y = location.get("y")
242     # print("width - ", w)
243     # print("height - ", h)
244     # print("xcor - ", x)
245     # print("ycor - ", y)
246     self.assertEqual((w, h), (tw, th))
247     self.assertEqual((x, y), (xcor, ycor))
248     print("Testcase 11 Passed")
249
250 except NoSuchElementException:
251     print("Element does not exist Test Case 11 Failed")
252
253 def testcase111(self):
254     try:
255         l = browser.find_element(By.ID, "signup_user_confirm_password_label")
256         s = l.text
257         location = l.location
258         size = l.size
259         w, h = size["width"], size["height"]
260         tw = 131
261         th = 18
262         xcor = 310
263         ycor = 487
264         x = location.get("x")
265         y = location.get("y")
266         # print("width - ", w)
267         # print("height - ", h)
268         # print("xcor - ", x)
269         # print("ycor - ", y)
270         self.assertEqual((w, h), (tw, th))
271         self.assertEqual((x, y), (xcor, ycor))
272         print("Testcase 12 Passed")
273
274     except NoSuchElementException:
275         print("Element does not exist Test Case 12 Failed")
276
277 def testcase112(self):
278     try:
279         l = browser.find_element(By.ID, "signup_user_confirm_password_label")
280         s = l.text
281         value = "Confirm Password"
282         self.assertEqual(s, value)
283         print("Testcase 13 Passed")
284
285     except NoSuchElementException:
286         print("Element does not exist Test Case 13 Failed")
287
288 def testcase113(self):
289     try:
290         l = browser.find_element(By.ID, "signup_confirm_password")
291         s = l.text
292         location = l.location
293         size = l.size
294         w, h = size["width"], size["height"]
```

```
295         tw = 400
296         th = 44
297         xcor = 310
298         ycor = 510
299         x = location.get("x")
300         y = location.get("y")
301         # print("width - ", w)
302         # print("height - ", h)
303         # print("xcor - ", x)
304         # print("ycor - ", y)
305         self.assertEqual((w, h), (tw, th))
306         self.assertEqual((x, y), (xcor, ycor))
307         print("Testcase 14 Passed")
308
309     except NoSuchElementException:
310         print("Element does not exist Test Case 14 Failed")
311
312 def testcase114(self):
313     try:
314         l = browser.find_element(By.ID, "signup_user_phone_number_label")
315         s = l.text
316         location = l.location
317         size = l.size
318         w, h = size["width"], size["height"]
319         tw = 108
320         th = 18
321         xcor = 310
322         ycor = 570
323         x = location.get("x")
324         y = location.get("y")
325         # print("width - ", w)
326         # print("height - ", h)
327         # print("xcor - ", x)
328         # print("ycor - ", y)
329         self.assertEqual((w, h), (tw, th))
330         self.assertEqual((x, y), (xcor, ycor))
331         print("Testcase 15 Passed")
332
333     except NoSuchElementException:
334         print("Element does not exist Test Case 15 Failed")
335
336 def testcase115(self):
337     try:
338         l = browser.find_element(By.ID, "signup_user_phone_number_label")
339         s = l.text
340         value = "Phone Number"
341         self.assertEqual(s, value)
342         print("Testcase 16 Passed")
343
344     except NoSuchElementException:
345         print("Element does not exist Test Case 16 Failed")
346
347 def testcase116(self):
348     try:
349         l = browser.find_element(By.ID, "signupUser_phonenumber")
350         s = l.text
351         location = l.location
352         size = l.size
353         w, h = size["width"], size["height"]
354         tw = 400
```

```
355     th = 44
356     xcor = 310
357     ycor = 593
358     x = location.get("x")
359     y = location.get("y")
360     # print("width - ", w)
361     # print("height - ", h)
362     # print("xcor - ", x)
363     # print("ycor - ", y)
364     self.assertEqual((w, h), (tw, th))
365     self.assertEqual((x, y), (xcor, ycor))
366     print("Testcase 17 Passed")
367
368     except NoSuchElementException:
369         print("Element does not exist Test Case 17 Failed")
370
371 def testcase117(self):
372     try:
373         l = browser.find_element(By.ID, "signup_gender_label")
374         s = l.text
375         location = l.location
376         size = l.size
377         w, h = size["width"], size["height"]
378         tw = 53
379         th = 18
380         xcor = 310
381         ycor = 653
382         x = location.get("x")
383         y = location.get("y")
384         # print("width - ", w)
385         # print("height - ", h)
386         # print("xcor - ", x)
387         # print("ycor - ", y)
388         self.assertEqual((w, h), (tw, th))
389         self.assertEqual((x, y), (xcor, ycor))
390         print("Testcase 18 Passed")
391
392     except NoSuchElementException:
393         print("Element does not exist Test Case 18 Failed")
394
395 def testcase118(self):
396     try:
397         l = browser.find_element(By.ID, "signup_gender_label")
398         s = l.text
399         value = "Gender"
400         self.assertEqual(s, value)
401         print("Testcase 19 Passed")
402
403     except NoSuchElementException:
404         print("Element does not exist Test Case 19 Failed")
405
406 def testcase119(self):
407     try:
408         l = browser.find_element(By.ID, "male_radio_label")
409         s = l.text
410         location = l.location
411         size = l.size
412         w, h = size["width"], size["height"]
413         tw = 400
414         th = 18
```

```
415         xcor = 310
416         ycor = 684
417         x = location.get("x")
418         y = location.get("y")
419         # print("width - ", w)
420         # print("height - ", h)
421         # print("xcor - ", x)
422         # print("ycor - ", y)
423         self.assertEqual((w, h), (tw, th))
424         self.assertEqual((x, y), (xcor, ycor))
425         print("Testcase 20 Passed")
426
427     except NoSuchElementException:
428         print("Element does not exist Test Case 20 Failed")
429
430 def testcase120(self):
431     try:
432         l = browser.find_element(By.ID, "male_radio_label")
433         s = l.text
434         value = "Male"
435         self.assertEqual(s, value)
436         print("Testcase 21 Passed")
437
438     except NoSuchElementException:
439         print("Element does not exist Test Case 21 Failed")
440
441 def testcase121(self):
442     try:
443         l = browser.find_element(By.ID, "male_radio_label_button")
444         s = l.text
445         location = l.location
446         size = l.size
447         w, h = size["width"], size["height"]
448         tw = 13
449         th = 13
450         xcor = 384
451         ycor = 687
452         x = location.get("x")
453         y = location.get("y")
454         # print("width - ", w)
455         # print("height - ", h)
456         # print("xcor - ", x)
457         # print("ycor - ", y)
458         self.assertEqual((w, h), (tw, th))
459         self.assertEqual((x, y), (xcor, ycor))
460         print("Testcase 22 Passed")
461
462     except NoSuchElementException:
463         print("Element does not exist Test Case 22 Failed")
464
465 def testcase122(self):
466     try:
467         l = browser.find_element(By.ID, "female_radio_label")
468         s = l.text
469         location = l.location
470         size = l.size
471         w, h = size["width"], size["height"]
472         tw = 400
473         th = 18
474         xcor = 310
```

```
475         ycor = 714
476         x = location.get("x")
477         y = location.get("y")
478         # print("width - ", w)
479         # print("height - ", h)
480         # print("xcor - ", x)
481         # print("ycor - ", y)
482         self.assertEqual((w, h), (tw, th))
483         self.assertEqual((x, y), (xcor, ycor))
484         print("Testcase 23 Passed")
485
486     except NoSuchElementException:
487         print("Element does not exist Test Case 23 Failed")
488
489 def testcase123(self):
490     try:
491         l = browser.find_element(By.ID, "female_radio_label")
492         s = l.text
493         value = "Female"
494         self.assertEqual(s, value)
495         print("Testcase 24 Passed")
496
497     except NoSuchElementException:
498         print("Element does not exist Test Case 24 Failed")
499
500 def testcase124(self):
501     try:
502         l = browser.find_element(By.ID, "female_radio_label_button")
503         s = l.text
504         location = l.location
505         size = l.size
506         w, h = size["width"], size["height"]
507         tw = 13
508         th = 13
509         xcor = 403
510         ycor = 717
511         x = location.get("x")
512         y = location.get("y")
513         # print("width - ", w)
514         # print("height - ", h)
515         # print("xcor - ", x)
516         # print("ycor - ", y)
517         self.assertEqual((w, h), (tw, th))
518         self.assertEqual((x, y), (xcor, ycor))
519         print("Testcase 25 Passed")
520
521     except NoSuchElementException:
522         print("Element does not exist Test Case 25 Failed")
523
524 def testcase125(self):
525     try:
526         l = browser.find_element(By.ID, "other_gender_radio_label")
527         s = l.text
528         location = l.location
529         size = l.size
530         w, h = size["width"], size["height"]
531         tw = 400
532         th = 18
533         xcor = 310
534         ycor = 745
```

```
535         x = location.get("x")
536         y = location.get("y")
537         # print("width - ", w)
538         # print("height - ", h)
539         # print("xcor - ", x)
540         # print("ycor - ", y)
541         self.assertEqual((w, h), (tw, th))
542         self.assertEqual((x, y), (xcor, ycor))
543         print("Testcase 26 Passed")
544
545     except NoSuchElementException:
546         print("Element does not exist Test Case 26 Failed")
547
548 def testcase126(self):
549     try:
550         l = browser.find_element(By.ID, "other_gender_radio_label")
551         s = l.text
552         value = "Prefer not to say"
553         self.assertEqual(s, value)
554         print("Testcase 27 Passed")
555
556     except NoSuchElementException:
557         print("Element does not exist Test Case 27 Failed")
558
559 def testcase127(self):
560     try:
561         l = browser.find_element(By.ID, "other_gender_radio_label_button")
562         s = l.text
563         location = l.location
564         size = l.size
565         w, h = size["width"], size["height"]
566         tw = 13
567         th = 13
568         xcor = 467
569         ycor = 748
570         x = location.get("x")
571         y = location.get("y")
572         # print("width - ", w)
573         # print("height - ", h)
574         # print("xcor - ", x)
575         # print("ycor - ", y)
576         self.assertEqual((w, h), (tw, th))
577         self.assertEqual((x, y), (xcor, ycor))
578         print("Testcase 28 Passed")
579
580     except NoSuchElementException:
581         print("Element does not exist Test Case 28 Failed")
582
583 def testcase128(self):
584     try:
585         l = browser.find_element(By.ID, "slider_range_for_age")
586         s = l.text
587         location = l.location
588         size = l.size
589         w, h = size["width"], size["height"]
590         tw = 400
591         th = 62
592         xcor = 310
593         ycor = 775
594         x = location.get("x")
```

```
595     y = location.get("y")
596     # print("width - ", w)
597     # print("height - ", h)
598     # print("xcor - ", x)
599     # print("ycor - ", y)
600     self.assertEqual((w, h), (tw, th))
601     self.assertEqual((x, y), (xcor, ycor))
602     print("Testcase 29 Passed")
603
604 except NoSuchElementException:
605     print("Element does not exist Test Case 29 Failed")
606
607 def testcase129(self):
608     try:
609         l = browser.find_element(By.ID, "slider_label_for_age")
610         s = l.text
611         location = l.location
612         size = l.size
613         w, h = size["width"], size["height"]
614         tw = 28
615         th = 18
616         xcor = 310
617         ycor = 797
618         x = location.get("x")
619         y = location.get("y")
620         # print("width - ", w)
621         # print("height - ", h)
622         # print("xcor - ", x)
623         # print("ycor - ", y)
624         self.assertEqual((w, h), (tw, th))
625         self.assertEqual((x, y), (xcor, ycor))
626         print("Testcase 30 Passed")
627
628     except NoSuchElementException:
629         print("Element does not exist Test Case 30 Failed")
630
631 def testcase130(self):
632     try:
633         l = browser.find_element(By.ID, "slider_label_for_age")
634         s = l.text
635         value = "Age"
636         self.assertEqual(s, value)
637         print("Testcase 31 Passed")
638
639     except NoSuchElementException:
640         print("Element does not exist Test Case 31 Failed")
641
642 def testcase131(self):
643     try:
644         l = browser.find_element(By.ID, "age_range_slider")
645         s = l.text
646         location = l.location
647         size = l.size
648         w, h = size["width"], size["height"]
649         tw = 338
650         th = 2
651         xcor = 340
652         ycor = 805
653         x = location.get("x")
654         y = location.get("y")
```

```
655     # print("width - ", w)
656     # print("height - ", h)
657     # print("xcor - ", x)
658     # print("ycor - ", y)
659     self.assertEqual((w, h), (tw, th))
660     self.assertEqual((x, y), (xcor, ycor))
661     print("Testcase 32 Passed")
662
663 except NoSuchElementException:
664     print("Element does not exist Test Case 32 Failed")
665
666 def testcase132(self):
667     try:
668         l = browser.find_element(By.ID, "rangeValue")
669         s = l.text
670         location = l.location
671         size = l.size
672         w, h = size["width"], size["height"]
673         tw = 30
674         th = 22
675         xcor = 680
676         ycor = 795
677         x = location.get("x")
678         y = location.get("y")
679         # print("width - ", w)
680         # print("height - ", h)
681         # print("xcor - ", x)
682         # print("ycor - ", y)
683         self.assertEqual((w, h), (tw, th))
684         self.assertEqual((x, y), (xcor, ycor))
685         print("Testcase 33 Passed")
686
687     except NoSuchElementException:
688         print("Element does not exist Test Case 33 Failed")
689
690 def testcase133(self):
691     try:
692         l = browser.find_element(By.ID, "rangeValue")
693         s = l.text
694         value = browser.find_element(By.ID, "age_range_slider").get_attribute(
695             "value"
696         )
697         self.assertEqual(s, value)
698         print("Testcase 34 Passed")
699
700     except NoSuchElementException:
701         print("Element does not exist Test Case 34 Failed")
702
703 def testcase134(self):
704     try:
705         l = browser.find_element(By.ID, "button_signup")
706         s = l.text
707         location = l.location
708         size = l.size
709         w, h = size["width"], size["height"]
710         tw = 400
711         th = 45
712         xcor = 310
713         ycor = 852
714         x = location.get("x")
```

```
715     y = location.get("y")
716     # print("width - ", w)
717     # print("height - ", h)
718     # print("xcor - ", x)
719     # print("ycor - ", y)
720     self.assertEqual((w, h), (tw, th))
721     self.assertEqual((x, y), (xcor, ycor))
722     print("Testcase 35 Passed")
723
724 except NoSuchElementException:
725     print("Element does not exist Test Case 35 Failed")
726
727 def testcase135(self):
728     try:
729         l = browser.find_element(By.ID, "user_signup_button")
730         s = l.text
731         location = l.location
732         size = l.size
733         w, h = size["width"], size["height"]
734         tw = 400
735         th = 45
736         xcor = 310
737         ycor = 852
738         x = location.get("x")
739         y = location.get("y")
740         # print("width - ", w)
741         # print("height - ", h)
742         # print("xcor - ", x)
743         # print("ycor - ", y)
744         self.assertEqual((w, h), (tw, th))
745         self.assertEqual((x, y), (xcor, ycor))
746         print("Testcase 36 Passed")
747
748     except NoSuchElementException:
749         print("Element does not exist Test Case 36 Failed")
750
751 def testcase136(self):
752     try:
753         l = browser.find_element(By.ID, "navigation_to_login_page")
754         s = l.text
755         location = l.location
756         size = l.size
757         w, h = size["width"], size["height"]
758         tw = 400
759         th = 45
760         xcor = 310
761         ycor = 913
762         x = location.get("x")
763         y = location.get("y")
764         # print("width - ", w)
765         # print("height - ", h)
766         # print("xcor - ", x)
767         # print("ycor - ", y)
768         self.assertEqual((w, h), (tw, th))
769         self.assertEqual((x, y), (xcor, ycor))
770         print("Testcase 37 Passed")
771
772     except NoSuchElementException:
773         print("Element does not exist Test Case 37 Failed")
774
```

```
775     def testcase137(self):
776         try:
777             l = browser.find_element(By.ID,
778 "label_for_login_button_on_sign_up_page")
779             s = l.text
780             location = l.location
781             size = l.size
782             w, h = size["width"], size["height"]
783             tw = 400
784             th = 18
785             xcor = 310
786             ycor = 913
787             x = location.get("x")
788             y = location.get("y")
789             # print("width - ", w)
790             # print("height - ", h)
791             # print("xcor - ", x)
792             # print("ycor - ", y)
793             self.assertEqual((w, h), (tw, th))
794             self.assertEqual((x, y), (xcor, ycor))
795             print("Testcase 38 Passed")
796
797         except NoSuchElementException:
798             print("Element does not exist Test Case 38 Failed")
799
800     def testcase138(self):
801         try:
802             l = browser.find_element(By.ID,
803 "label_for_login_button_on_sign_up_page")
804             s = l.text
805             value = "Already have an account?"
806             self.assertEqual(s, value)
807             print("Testcase 39 Passed")
808
809         except NoSuchElementException:
810             print("Element does not exist Test Case 39 Failed")
811
812     def testcase139(self):
813         try:
814             l = browser.find_element(By.ID, "login_button_on_sign_up_page")
815             s = l.text
816             location = l.location
817             size = l.size
818             w, h = size["width"], size["height"]
819             tw = 400
820             th = 45
821             xcor = 310
822             ycor = 948
823             x = location.get("x")
824             y = location.get("y")
825             # print("width - ", w)
826             # print("height - ", h)
827             # print("xcor - ", x)
828             # print("ycor - ", y)
829             self.assertEqual((w, h), (tw, th))
830             self.assertEqual((x, y), (xcor, ycor))
831             print("Testcase 40 Passed")
832
833         except NoSuchElementException:
834             print("Element does not exist Test Case 40 Failed")
```

```
833
834     def testcase140(self):
835         try:
836             username = browser.find_element(By.ID, "signupUsername")
837             username.clear()
838             username.send_keys("entering my username here as a value")
839
840             email = browser.find_element(By.ID, "signupUsername_email")
841             email.clear()
842             email.send_keys("entering my username here as a value")
843
844             password = browser.find_element(By.ID, "signup_password")
845             password.clear()
846             password.send_keys("entering my password here as a value")
847
848             cofirm_password = browser.find_element(By.ID,
849             "signup_confirm_password")
850             cofirm_password.clear()
851             cofirm_password.send_keys("entering my password here as a value")
852
853             phone_number = browser.find_element(By.ID, "signupUser_phonenumber")
854             phone_number.clear()
855             phone_number.send_keys("8135630372")
856
857             browser.find_element(By.ID, "male_radio_label_button").click()
858             browser.find_element(By.ID, "user_signup_button").click()
859
860             i = 0
861             while i < 20:
862                 i = i + 1
863
864             l = browser.find_element(By.ID, "homeimage")
865             s = "Found"
866             self.assertEqual(s, "Found")
867             print("Testcase 41 Passed")
868
869         except:
870             print("Element does not exist Test Case 41 Failed")
871
872         # test cases for the login page
873
874     def testcase141(self):
875         try:
876             l = browser.find_element(By.ID, "homeimage")
877             s = l.text
878             location = l.location
879             size = l.size
880             w, h = size["width"], size["height"]
881             tw = 100
882             th = 100
883             xcor = 460
884             ycor = 48
885             x = location.get("x")
886             y = location.get("y")
887             # print("width - ", w)
888             # print("height - ", h)
889             # print("xcor - ", x)
890             # print("ycor - ", y)
891             self.assertEqual((w, h), (tw, th))
892             self.assertEqual((x, y), (xcor, ycor))
```

```
892         print("Testcase 42 Passed")
893
894     except NoSuchElementException:
895         print("Element does not exist Test Case 42 Failed")
896
897 def testcase142(self):
898     try:
899         l = browser.find_element(By.ID, "login_page_title")
900         s = l.text
901         location = l.location
902         size = l.size
903         w, h = size["width"], size["height"]
904         tw = 400
905         th = 36
906         xcor = 310
907         ycor = 169
908         x = location.get("x")
909         y = location.get("y")
910         # print("width - ", w)
911         # print("height - ", h)
912         # print("xcor - ", x)
913         # print("ycor - ", y)
914         self.assertEqual((w, h), (tw, th))
915         self.assertEqual((x, y), (xcor, ycor))
916         print("Testcase 43 Passed")
917
918     except NoSuchElementException:
919         print("Element does not exist Test Case 43 Failed")
920
921 def testcase143(self):
922     try:
923         l = browser.find_element(By.ID, "userNameorEmail")
924         s = l.text
925         location = l.location
926         size = l.size
927         w, h = size["width"], size["height"]
928         x = location.get("x")
929         y = location.get("y")
930         tw = 137
931         th = 18
932         xcor = 310
933         ycor = 237
934         # print("width - ", w)
935         # print("height - ", h)
936         # print("xcor - ", x)
937         # print("ycor - ", y)
938         self.assertEqual((w, h), (tw, th))
939         self.assertEqual((x, y), (xcor, ycor))
940         print("Testcase 44 Passed")
941
942     except NoSuchElementException:
943         print("Element does not exist Test Case 44 Failed")
944
945 def testcase144(self):
946     try:
947         l = browser.find_element(By.ID, "userNameorEmail")
948         s = l.text
949         value = "Username or Email"
950         self.assertEqual(s, value)
951         print("Testcase 45 Passed")
```

```
952
953     except NoSuchElementException:
954         print("Element does not exist Test Case 45 Failed")
955
956 def testcase145(self):
957     try:
958         l = browser.find_element(By.ID, "userNameorEmail_input")
959         s = l.text
960         location = l.location
961         size = l.size
962         w, h = size["width"], size["height"]
963         tw = 400
964         th = 44
965         xcor = 310
966         ycor = 256
967         x = location.get("x")
968         y = location.get("y")
969         # print("width - ", w)
970         # print("height - ", h)
971         # print("xcor - ", x)
972         # print("ycor - ", y)
973         self.assertEqual((w, h), (tw, th))
974         self.assertEqual((x, y), (xcor, ycor))
975         print("Testcase 46 Passed")
976
977     except NoSuchElementException:
978         print("Element does not exist Test Case 46 Failed")
979
980 def testcase146(self):
981     try:
982         l = browser.find_element(By.ID, "user_password")
983         s = l.text
984         value = "Password"
985         self.assertEqual(s, value)
986         print("Testcase 47 Passed")
987
988     except NoSuchElementException:
989         print("Element does not exist Test Case 47 Failed")
990
991 def testcase147(self):
992     try:
993         l = browser.find_element(By.ID, "password_input")
994         s = l.text
995         location = l.location
996         size = l.size
997         w, h = size["width"], size["height"]
998         tw = 400
999         th = 44
1000        xcor = 310
1001        ycor = 334
1002        x = location.get("x")
1003        y = location.get("y")
1004        # print("width - ", w)
1005        # print("height - ", h)
1006        # print("xcor - ", x)
1007        # print("ycor - ", y)
1008        self.assertEqual((w, h), (tw, th))
1009        self.assertEqual((x, y), (xcor, ycor))
1010        print("Testcase 48 Passed")
1011
```

```
1012         except NoSuchElementException:
1013             print("Element does not exist Test Case 48 Failed")
1014
1015     def testcase148(self):
1016         try:
1017             l = browser.find_element(By.ID,
1018 "label_for_user_login_terms_and_conditions")
1019             s = l.text
1020             value = "Please read the Terms and conditions below:"
1021             self.assertEqual(s, value)
1022             print("Testcase 49 Passed")
1023         except NoSuchElementException:
1024             print("Element does not exist Test Case 49 Failed")
1025
1026     def testcase149(self):
1027         try:
1028             l = browser.find_element(By.ID,
1029 "label_for_user_login_terms_and_conditions")
1030             s = l.text
1031             location = l.location
1032             size = l.size
1033             w, h = size["width"], size["height"]
1034             tw = 400
1035             th = 18
1036             xcor = 310
1037             ycor = 394
1038             x = location.get("x")
1039             y = location.get("y")
1040             # print("width - ", w)
1041             # print("height - ", h)
1042             # print("xcor - ", x)
1043             # print("ycor - ", y)
1044             self.assertEqual((w, h), (tw, th))
1045             self.assertEqual((x, y), (xcor, ycor))
1046             print("Testcase 50 Passed")
1047
1048         except NoSuchElementException:
1049             print("Element does not exist Test Case 50 Failed")
1050
1051     def testcase150(self):
1052         try:
1053             l = browser.find_element(By.ID, "login_scroll_area")
1054             s = l.text
1055             location = l.location
1056             size = l.size
1057             w, h = size["width"], size["height"]
1058             tw = 402
1059             th = 122
1060             xcor = 310
1061             ycor = 429
1062             x = location.get("x")
1063             y = location.get("y")
1064             # print("width - ", w)
1065             # print("height - ", h)
1066             # print("xcor - ", x)
1067             # print("ycor - ", y)
1068             self.assertEqual((w, h), (tw, th))
1069             self.assertEqual((x, y), (xcor, ycor))
1070             print("Testcase 51 Passed")
```

```
1070     except NoSuchElementException:
1071         print("Element does not exist Test Case 51 Failed")
1072
1073     def testcase151(self):
1074         try:
1075             l = browser.find_element(By.ID, "login_scroll_area")
1076             s = l.text
1077             value = "TERMS AND CONDITIONS BY SIGNING THIS AGREEMENT. (i) You
acknowledge that you have read and understand the terms and conditions of this
Agreement including those on page 2 of this Agreement; (ii) You agree that this
Agreement is a net agreement that you cannot terminate or cancel, you have an
unconditional obligation to make all payments due under this Agreement, and you
cannot withhold, set off or reduce such payments for any reason; (iii) You will use
the Products only for business purposes and in compliance with all applicable laws
and regulations; (iv) you acknowledge that if this Agreement is replacing an
existing agreement the new Installment Payment may include the balance of that
existing agreement and result in a greater aggregate product cost to you; and (v)
You agree that by providing a telephone number to a cellular or other wireless
device, you are expressly consenting to receiving communications from us, our
affiliates and agents (for non-marketing purposes) at that number, including, but
not limited to, prerecorded and artificial voice messages, text messages, and calls
from automated telephone dialing systems; these calls may incur fees from your
cellular provider; and this consent applies to each such telephone number you
provide to us now or in the future"
1078             self.assertEqual(s.strip(), value.strip())
1079             print("Testcase 52 Passed")
1080         except NoSuchElementException:
1081             print("Element does not exist Test Case 52 Failed")
1082
1083     def testcase152(self):
1084         try:
1085             l = browser.find_element(
1086                 By.ID, "radio_label_for_user_login_terms_and_conditions"
1087             )
1088             s = l.text
1089             value = "I agree to the terms and conditions"
1090             self.assertEqual(s.strip(), value.strip())
1091             print("Testcase 53 Passed")
1092         except NoSuchElementException:
1093             print("Element does not exist Test Case 53 Failed")
1094
1095     def testcase153(self):
1096         try:
1097             l = browser.find_element(
1098                 By.ID, "radio_label_for_user_login_terms_and_conditions"
1099             )
1100             s = l.text
1101             location = l.location
1102             size = l.size
1103             w, h = size["width"], size["height"]
1104             tw = 400
1105             th = 18
1106             xcor = 310
1107             ycor = 562
1108             x = location.get("x")
1109             y = location.get("y")
1110             # print("width - ", w)
1111             # print("height - ", h)
1112             # print("xcor - ", x)
1113             # print("ycor - ", y)
```

```
1114     self.assertEqual((w, h), (tw, th))
1115     self.assertEqual((x, y), (xcor, ycor))
1116     print("Testcase 54 Passed")
1117
1118     except NoSuchElementException:
1119         print("Element does not exist Test Case 54 Failed")
1120
1121 def testcase154(self):
1122     try:
1123         l = browser.find_element(By.ID, "login_terms_and_conditions")
1124         s = l.text
1125         location = l.location
1126         size = l.size
1127         w, h = size["width"], size["height"]
1128         tw = 13
1129         th = 13
1130         xcor = 596
1131         ycor = 565
1132         x = location.get("x")
1133         y = location.get("y")
1134         # print("width - ", w)
1135         # print("height - ", h)
1136         # print("xcor - ", x)
1137         # print("ycor - ", y)
1138         self.assertEqual((w, h), (tw, th))
1139         self.assertEqual((x, y), (xcor, ycor))
1140         print("Testcase 55 Passed")
1141
1142     except NoSuchElementException:
1143         print("Element does not exist Test Case 55 Failed")
1144
1145 def testcase155(self):
1146     try:
1147         l = browser.find_element(By.ID, "button_login")
1148         s = l.text
1149         location = l.location
1150         size = l.size
1151         w, h = size["width"], size["height"]
1152         tw = 400
1153         th = 45
1154         xcor = 310
1155         ycor = 596
1156         x = location.get("x")
1157         y = location.get("y")
1158         # print("width - ", w)
1159         # print("height - ", h)
1160         # print("xcor - ", x)
1161         # print("ycor - ", y)
1162         self.assertEqual((w, h), (tw, th))
1163         self.assertEqual((x, y), (xcor, ycor))
1164         print("Testcase 56 Passed")
1165
1166     except NoSuchElementException:
1167         print("Element does not exist Test Case 56 Failed")
1168
1169 def testcase156(self):
1170     try:
1171         l = browser.find_element(By.ID, "user_login_button")
1172         s = l.text
1173         location = l.location
```

```
1174     size = l.size
1175     w, h = size["width"], size["height"]
1176     tw = 400
1177     th = 45
1178     xcor = 310
1179     ycor = 596
1180     x = location.get("x")
1181     y = location.get("y")
1182     # print("width - ", w)
1183     # print("height - ", h)
1184     # print("xcor - ", x)
1185     # print("ycor - ", y)
1186     self.assertEqual((w, h), (tw, th))
1187     self.assertEqual((x, y), (xcor, ycor))
1188     print("Testcase 57 Passed")
1189
1190 except NoSuchElementException:
1191     print("Element does not exist Test Case 57 Failed")
1192
1193 def testcase157(self):
1194     try:
1195         l = browser.find_element(By.ID, "label_for_sign_up_on_login_page")
1196         s = l.text
1197         location = l.location
1198         size = l.size
1199         w, h = size["width"], size["height"]
1200         tw = 400
1201         th = 18
1202         xcor = 310
1203         ycor = 657
1204         x = location.get("x")
1205         y = location.get("y")
1206         # print("width - ", w)
1207         # print("height - ", h)
1208         # print("xcor - ", x)
1209         # print("ycor - ", y)
1210         self.assertEqual((w, h), (tw, th))
1211         self.assertEqual((x, y), (xcor, ycor))
1212         print("Testcase 58 Passed")
1213
1214     except NoSuchElementException:
1215         print("Element does not exist Test Case 58 Failed")
1216
1217
1218 def testcase158(self):
1219     try:
1220         l = browser.find_element(By.ID, "label_for_sign_up_on_login_page")
1221         s = l.text
1222         value = "Don't have an account?"
1223         self.assertEqual(s.strip(), value.strip())
1224         print("Testcase 59 Passed")
1225     except NoSuchElementException:
1226         print("Element does not exist Test Case 59 Failed")
1227
1228 def testcase159(self):
1229     try:
1230         l = browser.find_element(By.ID, "sign_up_button_on_login_page")
1231         s = l.text
1232         location = l.location
1233         size = l.size
```

```
1234     w, h = size["width"], size["height"]
1235     tw = 400
1236     th = 45
1237     xcor = 310
1238     ycor = 691
1239     x = location.get("x")
1240     y = location.get("y")
1241     # print("width - ", w)
1242     # print("height - ", h)
1243     # print("xcor - ", x)
1244     # print("ycor - ", y)
1245     self.assertEqual((w, h), (tw, th))
1246     self.assertEqual((x, y), (xcor, ycor))
1247     print("Testcase 60 Passed")
1248
1249 except NoSuchElementException:
1250     print("Element does not exist Test Case 60 Failed")
1251
1252 def testcase160(self):
1253     try:
1254         username = browser.find_element(By.ID, "userNameorEmail_input")
1255         username.clear()
1256         username.send_keys("entering my username here as a value")
1257
1258         password = browser.find_element(By.ID, "password_input")
1259         password.clear()
1260         password.send_keys("entering my password here as a value")
1261
1262         browser.find_element(By.ID, "login_terms_and_conditions").click()
1263         browser.find_element(By.ID, "user_login_button").click()
1264
1265         l = browser.find_element(By.ID, "review_homeimage")
1266         s = "Found"
1267         self.assertEqual(s, "Found")
1268         print("Testcase 61 Passed")
1269
1270     except:
1271         print("Element does not exist Test Case 61 Failed")
1272
1273 def testcase161(self):
1274     try:
1275         l = browser.find_element(By.ID, "review_homeimage")
1276         s = l.text
1277         location = l.location
1278         size = l.size
1279         w, h = size["width"], size["height"]
1280         tw = 301
1281         th = 168
1282         xcor = 359
1283         ycor = 48
1284         x = location.get("x")
1285         y = location.get("y")
1286         # print("width - ", w)
1287         # print("height - ", h)
1288         # print("xcor - ", x)
1289         # print("ycor - ", y)
1290         self.assertEqual((w, h), (tw, th))
1291         self.assertEqual((x, y), (xcor, ycor))
1292         print("Testcase 62 Passed")
1293
```

```
1294     except NoSuchElementException:
1295         print("Element does not exist Test Case 62 Failed")
1296
1297 def testcase162(self):
1298     try:
1299         l = browser.find_element(By.ID, "review_page_title")
1300         s = l.text
1301         location = l.location
1302         size = l.size
1303         w, h = size["width"], size["height"]
1304         tw = 600
1305         th = 36
1306         xcor = 210
1307         ycor = 237
1308         x = location.get("x")
1309         y = location.get("y")
1310         # print("width - ", w)
1311         # print("height - ", h)
1312         # print("xcor - ", x)
1313         # print("ycor - ", y)
1314         self.assertEqual((w, h), (tw, th))
1315         self.assertEqual((x, y), (xcor, ycor))
1316         print("Testcase 63 Passed")
1317
1318     except NoSuchElementException:
1319         print("Element does not exist Test Case 63 Failed")
1320
1321 def testcase163(self):
1322     try:
1323         l = browser.find_element(By.ID, "review_page_title")
1324         s = l.text
1325         value = "WE APPRECIATE YOUR REVIEW!"
1326         self.assertEqual(s.strip(), value.strip())
1327         print("Testcase 64 Passed")
1328     except NoSuchElementException:
1329         print("Element does not exist Test Case 64 Failed")
1330
1331 def testcase164(self):
1332     try:
1333         l = browser.find_element(By.ID, "review_title_subtet")
1334         s = l.text
1335         location = l.location
1336         size = l.size
1337         w, h = size["width"], size["height"]
1338         tw = 600
1339         th = 16
1340         xcor = 210
1341         ycor = 305
1342         x = location.get("x")
1343         y = location.get("y")
1344         # print("width - ", w)
1345         # print("height - ", h)
1346         # print("xcor - ", x)
1347         # print("ycor - ", y)
1348         self.assertEqual((w, h), (tw, th))
1349         self.assertEqual((x, y), (xcor, ycor))
1350         print("Testcase 65 Passed")
1351
1352     except NoSuchElementException:
1353         print("Element does not exist Test Case 65 Failed")
```

```
1354  
1355     def testcase165(self):  
1356         try:  
1357             l = browser.find_element(By.ID, "review_title_subtet")  
1358             s = l.text  
1359             value = "Your review will help me to improve in my future assignments."  
1360             self.assertEqual(s.strip(), value.strip())  
1361             print("Testcase 66 Passed")  
1362         except NoSuchElementException:  
1363             print("Element does not exist Test Case 66 Failed")  
1364  
1365     def testcase166(self):  
1366         try:  
1367             l = browser.find_element(By.ID, "review_username_label")  
1368             s = l.text  
1369             location = l.location  
1370             size = l.size  
1371             w, h = size["width"], size["height"]  
1372             tw = 74  
1373             th = 18  
1374             xcor = 210  
1375             ycor = 344  
1376             x = location.get("x")  
1377             y = location.get("y")  
1378             # print("width - ", w)  
1379             # print("height - ", h)  
1380             # print("xcor - ", x)  
1381             # print("ycor - ", y)  
1382             self.assertEqual((w, h), (tw, th))  
1383             self.assertEqual((x, y), (xcor, ycor))  
1384             print("Testcase 67 Passed")  
1385  
1386         except NoSuchElementException:  
1387             print("Element does not exist Test Case 67 Failed")  
1388  
1389     def testcase167(self):  
1390         try:  
1391             l = browser.find_element(By.ID, "review_username_label")  
1392             s = l.text  
1393             value = "Username"  
1394             self.assertEqual(s.strip(), value.strip())  
1395             print("Testcase 68 Passed")  
1396         except NoSuchElementException:  
1397             print("Element does not exist Test Case 68 Failed")  
1398  
1399     def testcase168(self):  
1400         try:  
1401             l = browser.find_element(By.ID, "review_username_input")  
1402             s = l.text  
1403             location = l.location  
1404             size = l.size  
1405             w, h = size["width"], size["height"]  
1406             tw = 600  
1407             th = 44  
1408             xcor = 210  
1409             ycor = 367  
1410             x = location.get("x")  
1411             y = location.get("y")  
1412             # print("width - ", w)  
1413             # print("height - ", h)
```

```
1414     # print("xcor - ", x)
1415     # print("ycor - ", y)
1416     self.assertEqual((w, h), (tw, th))
1417     self.assertEqual((x, y), (xcor, ycor))
1418     print("Testcase 69 Passed")
1419
1420     except NoSuchElementException:
1421         print("Element does not exist Test Case 69 Failed")
1422
1423 def testcase169(self):
1424     try:
1425         l = browser.find_element(By.ID, "review_user_email")
1426         s = l.text
1427         location = l.location
1428         size = l.size
1429         w, h = size["width"], size["height"]
1430         tw = 40
1431         th = 18
1432         xcor = 210
1433         ycor = 427
1434         x = location.get("x")
1435         y = location.get("y")
1436         # print("width - ", w)
1437         # print("height - ", h)
1438         # print("xcor - ", x)
1439         # print("ycor - ", y)
1440         self.assertEqual((w, h), (tw, th))
1441         self.assertEqual((x, y), (xcor, ycor))
1442         print("Testcase 70 Passed")
1443
1444     except NoSuchElementException:
1445         print("Element does not exist Test Case 70 Failed")
1446
1447 def testcase170(self):
1448     try:
1449         l = browser.find_element(By.ID, "review_user_email")
1450         s = l.text
1451         value = "Email"
1452         self.assertEqual(s.strip(), value.strip())
1453         print("Testcase 71 Passed")
1454     except NoSuchElementException:
1455         print("Element does not exist Test Case 71 Failed")
1456
1457 def testcase171(self):
1458     try:
1459         l = browser.find_element(By.ID, "review_user_email_input")
1460         s = l.text
1461         location = l.location
1462         size = l.size
1463         w, h = size["width"], size["height"]
1464         tw = 600
1465         th = 44
1466         xcor = 210
1467         ycor = 450
1468         x = location.get("x")
1469         y = location.get("y")
1470         # print("width - ", w)
1471         # print("height - ", h)
1472         # print("xcor - ", x)
1473         # print("ycor - ", y)
```

```
1474     self.assertEqual((w, h), (tw, th))
1475     self.assertEqual((x, y), (xcor, ycor))
1476     print("Testcase 72 Passed")
1477
1478     except NoSuchElementException:
1479         print("Element does not exist Test Case 72 Failed")
1480
1481 def testcase172(self):
1482     try:
1483         l = browser.find_element(By.ID, "rate_your_service")
1484         s = l.text
1485         location = l.location
1486         size = l.size
1487         w, h = size["width"], size["height"]
1488         tw = 600
1489         th = 18
1490         xcor = 210
1491         ycor = 515
1492         x = location.get("x")
1493         y = location.get("y")
1494         # print("width - ", w)
1495         # print("height - ", h)
1496         # print("xcor - ", x)
1497         # print("ycor - ", y)
1498         self.assertEqual((w, h), (tw, th))
1499         self.assertEqual((x, y), (xcor, ycor))
1500         print("Testcase 73 Passed")
1501
1502     except NoSuchElementException:
1503         print("Element does not exist Test Case 73 Failed")
1504
1505 def testcase173(self):
1506     try:
1507         l = browser.find_element(By.ID, "rate_your_service")
1508         s = l.text
1509         value = "Please rate your service experience for the following
parameters"
1510         self.assertEqual(s.strip(), value.strip())
1511         print("Testcase 74 Passed")
1512     except NoSuchElementException:
1513         print("Element does not exist Test Case 74 Failed")
1514
1515 def testcase174(self):
1516     try:
1517         l = browser.find_element(By.ID, "review_selection_tool_rating_label")
1518         s = l.text
1519         location = l.location
1520         size = l.size
1521         w, h = size["width"], size["height"]
1522         tw = 358
1523         th = 18
1524         xcor = 210
1525         ycor = 555
1526         x = location.get("x")
1527         y = location.get("y")
1528         # print("width - ", w)
1529         # print("height - ", h)
1530         # print("xcor - ", x)
1531         # print("ycor - ", y)
1532         self.assertEqual((w, h), (tw, th))
```

```
1533     self.assertEqual((x, y), (xcor, ycor))
1534     print("Testcase 75 Passed")
1535 
1536     except NoSuchElementException:
1537         print("Element does not exist Test Case 75 Failed")
1538 
1539 def testcase175(self):
1540     try:
1541         l = browser.find_element(By.ID, "review_selection_tool_rating_label")
1542         s = l.text
1543         value = "1. Selection of the testing tool for this assignment?"
1544         self.assertEqual(s.strip(), value.strip())
1545         print("Testcase 76 Passed")
1546     except NoSuchElementException:
1547         print("Element does not exist Test Case 76 Failed")
1548 
1549 def testcase176(self):
1550     try:
1551         l = browser.find_element(
1552             By.ID, "review_page_overall_experience_star_rating"
1553         )
1554         s = l.text
1555         location = l.location
1556         size = l.size
1557         w, h = size["width"], size["height"]
1558         tw = 175
1559         th = 35
1560         xcor = 210
1561         ycor = 578
1562         x = location.get("x")
1563         y = location.get("y")
1564         # print("width - ", w)
1565         # print("height - ", h)
1566         # print("xcor - ", x)
1567         # print("ycor - ", y)
1568         self.assertEqual((w, h), (tw, th))
1569         self.assertEqual((x, y), (xcor, ycor))
1570         print("Testcase 77 Passed")
1571 
1572     except NoSuchElementException:
1573         print("Element does not exist Test Case 77 Failed")
1574 
1575 def testcase177(self):
1576     try:
1577         l = browser.find_element(By.ID, "review_horizontal_line_one")
1578         s = l.text
1579         location = l.location
1580         size = l.size
1581         w, h = size["width"], size["height"]
1582         tw = 600
1583         th = 2
1584         xcor = 210
1585         ycor = 627
1586         x = location.get("x")
1587         y = location.get("y")
1588         # print("width - ", w)
1589         # print("height - ", h)
1590         # print("xcor - ", x)
1591         # print("ycor - ", y)
1592         self.assertEqual((w, h), (tw, th))
```

```
1593     self.assertEqual((x, y), (xcor, ycor))
1594     print("Testcase 78 Passed")
1595
1596 except NoSuchElementException:
1597     print("Element does not exist Test Case 78 Failed")
1598
1599 def testcase178(self):
1600     try:
1601         l = browser.find_element(By.ID, "review_assesment_2nd_question")
1602         s = l.text
1603         location = l.location
1604         size = l.size
1605         w, h = size["width"], size["height"]
1606         tw = 564
1607         th = 54
1608         xcor = 210
1609         ycor = 639
1610         x = location.get("x")
1611         y = location.get("y")
1612         # print("width - ", w)
1613         # print("height - ", h)
1614         # print("xcor - ", x)
1615         # print("ycor - ", y)
1616         self.assertEqual((w, h), (tw, th))
1617         self.assertEqual((x, y), (xcor, ycor))
1618         print("Testcase 79 Passed")
1619
1620     except NoSuchElementException:
1621         print("Element does not exist Test Case 79 Failed")
1622
1623 def testcase179(self):
1624     try:
1625         l = browser.find_element(By.ID, "review_assesment_2nd_question")
1626         s = l.text
1627         value = "2. Assessment of the submitted report in terms of set of grammar or formatting errors, information about the selected testing tool and the description of the Test Cases executed as part of this assignment?"
1628         self.assertEqual(s.strip(), value.strip())
1629         print("Testcase 80 Passed")
1630     except NoSuchElementException:
1631         print("Element does not exist Test Case 80 Failed")
1632
1633 def testcase180(self):
1634     try:
1635         l = browser.find_element(
1636             By.ID, "review_page_assesment_2nd_question_star_rating"
1637         )
1638         s = l.text
1639         location = l.location
1640         size = l.size
1641         w, h = size["width"], size["height"]
1642         tw = 175
1643         th = 35
1644         xcor = 210
1645         ycor = 699
1646         x = location.get("x")
1647         y = location.get("y")
1648         # print("width - ", w)
1649         # print("height - ", h)
1650         # print("xcor - ", x)
```

```
1651     # print("ycor - ", y)
1652     self.assertEqual((w, h), (tw, th))
1653     self.assertEqual((x, y), (xcor, ycor))
1654     print("Testcase 81 Passed")
1655
1656     except NoSuchElementException:
1657         print("Element does not exist Test Case 81 Failed")
1658
1659 def testcase181(self):
1660     try:
1661         l = browser.find_element(By.ID, "review_horizontal_line_two")
1662         s = l.text
1663         location = l.location
1664         size = l.size
1665         w, h = size["width"], size["height"]
1666         tw = 600
1667         th = 2
1668         xcor = 210
1669         ycor = 748
1670         x = location.get("x")
1671         y = location.get("y")
1672         # print("width - ", w)
1673         # print("height - ", h)
1674         # print("xcor - ", x)
1675         # print("ycor - ", y)
1676         self.assertEqual((w, h), (tw, th))
1677         self.assertEqual((x, y), (xcor, ycor))
1678         print("Testcase 82 Passed")
1679
1680     except NoSuchElementException:
1681         print("Element does not exist Test Case 82 Failed")
1682
1683 def testcase182(self):
1684     try:
1685         l = browser.find_element(By.ID,
1686 "review_implementation_3rd_question_label")
1687         s = l.text
1688         location = l.location
1689         size = l.size
1690         w, h = size["width"], size["height"]
1691         tw = 573
1692         th = 36
1693         xcor = 210
1694         ycor = 760
1695         x = location.get("x")
1696         y = location.get("y")
1697         # print("width - ", w)
1698         # print("height - ", h)
1699         # print("xcor - ", x)
1700         # print("ycor - ", y)
1701         self.assertEqual((w, h), (tw, th))
1702         self.assertEqual((x, y), (xcor, ycor))
1703         print("Testcase 83 Passed")
1704
1705     except NoSuchElementException:
1706         print("Element does not exist Test Case 83 Failed")
1707
1708 def testcase183(self):
1709     try:
```

```
1709     l = browser.find_element(By.ID,
1710 "review_implementation_3rd_question_label")
1711     s = l.text
1712     value = "3. How did you like the implementation of the application and
1713     the UI submitted as part of this assignment?"
1714     self.assertEqual(s.strip(), value.strip())
1715     print("Testcase 84 Passed")
1716 except NoSuchElementException:
1717     print("Element does not exist Test Case 84 Failed")
1718
1719 def testcase184(self):
1720     try:
1721         l = browser.find_element(
1722             By.ID, "review_implementation_3rd_question_scale_rating"
1723         )
1724         s = l.text
1725         location = l.location
1726         size = l.size
1727         w, h = size["width"], size["height"]
1728         tw = 600
1729         th = 35
1730         xcor = 210
1731         ycor = 838
1732         x = location.get("x")
1733         y = location.get("y")
1734         # print("width - ", w)
1735         # print("height - ", h)
1736         # print("xcor - ", x)
1737         # print("ycor - ", y)
1738         self.assertEqual((w, h), (tw, th))
1739         self.assertEqual((x, y), (xcor, ycor))
1740         print("Testcase 85 Passed")
1741     except NoSuchElementException:
1742         print("Element does not exist Test Case 85 Failed")
1743
1744 def testcase185(self):
1745     try:
1746         l = browser.find_element(By.ID, "review_horizontal_line_three")
1747         s = l.text
1748         location = l.location
1749         size = l.size
1750         w, h = size["width"], size["height"]
1751         tw = 600
1752         th = 2
1753         xcor = 210
1754         ycor = 902
1755         x = location.get("x")
1756         y = location.get("y")
1757         # print("width - ", w)
1758         # print("height - ", h)
1759         # print("xcor - ", x)
1760         # print("ycor - ", y)
1761         self.assertEqual((w, h), (tw, th))
1762         self.assertEqual((x, y), (xcor, ycor))
1763         print("Testcase 86 Passed")
1764     except NoSuchElementException:
1765         print("Element does not exist Test Case 86 Failed")
1766
```

```
1767     def testcase186(self):
1768         try:
1769             l = browser.find_element(By.ID,
1770             "3rd_question_review_scale_rating_labels")
1771             s = l.text
1772             location = l.location
1773             size = l.size
1774             w, h = size["width"], size["height"]
1775             tw = 600
1776             th = 0
1777             xcor = 210
1778             ycor = 815
1779             x = location.get("x")
1780             y = location.get("y")
1781             # print("width - ", w)
1782             # print("height - ", h)
1783             # print("xcor - ", x)
1784             # print("ycor - ", y)
1785             self.assertEqual((w, h), (tw, th))
1786             self.assertEqual((x, y), (xcor, ycor))
1787             print("Testcase 87 Passed")
1788
1789         except NoSuchElementException:
1790             print("Element does not exist Test Case 87 Failed")
1791
1792     def testcase187(self):
1793         try:
1794             l = browser.find_element(
1795                 By.ID, "3rd_question_review_scale_rating_labels_for_poor"
1796             )
1797             s = l.text
1798             location = l.location
1799             size = l.size
1800             w, h = size["width"], size["height"]
1801             tw = 47
1802             th = 18
1803             xcor = 210
1804             ycor = 815
1805             x = location.get("x")
1806             y = location.get("y")
1807             # print("width - ", w)
1808             # print("height - ", h)
1809             # print("xcor - ", x)
1810             # print("ycor - ", y)
1811             self.assertEqual((w, h), (tw, th))
1812             self.assertEqual((x, y), (xcor, ycor))
1813             print("Testcase 88 Passed")
1814
1815         except NoSuchElementException:
1816             print("Element does not exist Test Case 88 Failed")
1817
1818     def testcase188(self):
1819         try:
1820             l = browser.find_element(
1821                 By.ID, "3rd_question_review_scale_rating_labels_for_best"
1822             )
1823             s = l.text
1824             location = l.location
1825             size = l.size
1826             w, h = size["width"], size["height"]
```

```
1826         tw = 42
1827         th = 18
1828         xcor = 768
1829         ycor = 815
1830         x = location.get("x")
1831         y = location.get("y")
1832         # print("width - ", w)
1833         # print("height - ", h)
1834         # print("xcor - ", x)
1835         # print("ycor - ", y)
1836         self.assertEqual((w, h), (tw, th))
1837         self.assertEqual((x, y), (xcor, ycor))
1838         print("Testcase 89 Passed")
1839
1840     except NoSuchElementException:
1841         print("Element does not exist Test Case 89 Failed")
1842
1843 def testcase189(self):
1844     try:
1845         l = browser.find_element(
1846             By.ID, "3rd_question_review_scale_rating_labels_for_poor"
1847         )
1848         s = l.text
1849         value = "POOR"
1850         self.assertEqual(s.strip(), value.strip())
1851         print("Testcase 90 Passed")
1852     except NoSuchElementException:
1853         print("Element does not exist Test Case 90 Failed")
1854
1855 def testcase190(self):
1856     try:
1857         l = browser.find_element(
1858             By.ID, "3rd_question_review_scale_rating_labels_for_best"
1859         )
1860         s = l.text
1861         value = "BEST"
1862         self.assertEqual(s.strip(), value.strip())
1863         print("Testcase 91 Passed")
1864     except NoSuchElementException:
1865         print("Element does not exist Test Case 91 Failed")
1866
1867 def testcase191(self):
1868     try:
1869         l = browser.find_element(By.ID, "custom_review_text_area_box")
1870         s = l.text
1871         location = l.location
1872         size = l.size
1873         w, h = size["width"], size["height"]
1874         tw = 600
1875         th = 151
1876         xcor = 210
1877         ycor = 914
1878         x = location.get("x")
1879         y = location.get("y")
1880         # print("width - ", w)
1881         # print("height - ", h)
1882         # print("xcor - ", x)
1883         # print("ycor - ", y)
1884         self.assertEqual((w, h), (tw, th))
1885         self.assertEqual((x, y), (xcor, ycor))
```

```
1886         print("Testcase 92 Passed")
1887
1888     except NoSuchElementException:
1889         print("Element does not exist Test Case 92 Failed")
1890
1891 def testcase192(self):
1892     try:
1893         l = browser.find_element(By.ID, "custom_review_text_area_box_labek")
1894         s = l.text
1895         location = l.location
1896         size = l.size
1897         w, h = size["width"], size["height"]
1898         tw = 571
1899         th = 18
1900         xcor = 210
1901         ycor = 914
1902         x = location.get("x")
1903         y = location.get("y")
1904         # print("width - ", w)
1905         # print("height - ", h)
1906         # print("xcor - ", x)
1907         # print("ycor - ", y)
1908         self.assertEqual((w, h), (tw, th))
1909         self.assertEqual((x, y), (xcor, ycor))
1910         print("Testcase 93 Passed")
1911
1912     except NoSuchElementException:
1913         print("Element does not exist Test Case 93 Failed")
1914
1915 def testcase193(self):
1916     try:
1917         l = browser.find_element(By.ID, "custom_review_text_area_box_labek")
1918         s = l.text
1919         value = "4. If there are any Other suggestions, please let me know in
the text area below?"
1920         self.assertEqual(s.strip(), value.strip())
1921         print("Testcase 94 Passed")
1922     except NoSuchElementException:
1923         print("Element does not exist Test Case 94 Failed")
1924
1925 def testcase194(self):
1926     try:
1927         l = browser.find_element(By.ID, "custom_review_text_area_box_ui_html")
1928         s = l.text
1929         location = l.location
1930         size = l.size
1931         w, h = size["width"], size["height"]
1932         tw = 606
1933         th = 98
1934         xcor = 210
1935         ycor = 950
1936         x = location.get("x")
1937         y = location.get("y")
1938         # print("width - ", w)
1939         # print("height - ", h)
1940         # print("xcor - ", x)
1941         # print("ycor - ", y)
1942         self.assertEqual((w, h), (tw, th))
1943         self.assertEqual((x, y), (xcor, ycor))
1944         print("Testcase 95 Passed")
```

```
1945
1946     except NoSuchElementException:
1947         print("Element does not exist Test Case 95 Failed")
1948
1949 def testcase195(self):
1950     try:
1951         l = browser.find_element(By.ID, "review_horizontal_line_four")
1952         s = l.text
1953         location = l.location
1954         size = l.size
1955         w, h = size["width"], size["height"]
1956         tw = 600
1957         th = 2
1958         xcor = 210
1959         ycor = 1063
1960         x = location.get("x")
1961         y = location.get("y")
1962         # print("width - ", w)
1963         # print("height - ", h)
1964         # print("xcor - ", x)
1965         # print("ycor - ", y)
1966         self.assertEqual((w, h), (tw, th))
1967         self.assertEqual((x, y), (xcor, ycor))
1968         print("Testcase 96 Passed")
1969
1970     except NoSuchElementException:
1971         print("Element does not exist Test Case 96 Failed")
1972
1973 def testcase196(self):
1974     try:
1975         l = browser.find_element(By.ID, "review_page_overall_experience")
1976         s = l.text
1977         location = l.location
1978         size = l.size
1979         w, h = size["width"], size["height"]
1980         tw = 600
1981         th = 18
1982         xcor = 210
1983         ycor = 1074
1984         x = location.get("x")
1985         y = location.get("y")
1986         # print("width - ", w)
1987         # print("height - ", h)
1988         # print("xcor - ", x)
1989         # print("ycor - ", y)
1990         self.assertEqual((w, h), (tw, th))
1991         self.assertEqual((x, y), (xcor, ycor))
1992         print("Testcase 97 Passed")
1993
1994     except NoSuchElementException:
1995         print("Element does not exist Test Case 97 Failed")
1996
1997 def testcase197(self):
1998     try:
1999         l = browser.find_element(By.ID, "review_page_overall_experience")
2000         s = l.text
2001         value = "5. Your overall experience with my submitted assignment?"
2002         self.assertEqual(s.strip(), value.strip())
2003         print("Testcase 98 Passed")
2004     except NoSuchElementException:
```

```
2005     print("Element does not exist Test Case 98 Failed")
2006
2007     def testcase198(self):
2008         try:
2009             l = browser.find_element(By.ID, "slider_range_for_overall_rating")
2010             s = l.text
2011             location = l.location
2012             size = l.size
2013             w, h = size["width"], size["height"]
2014             tw = 600
2015             th = 62
2016             xcor = 210
2017             ycor = 1093
2018             x = location.get("x")
2019             y = location.get("y")
2020             # print("width - ", w)
2021             # print("height - ", h)
2022             # print("xcor - ", x)
2023             # print("ycor - ", y)
2024             self.assertEqual((w, h), (tw, th))
2025             self.assertEqual((x, y), (xcor, ycor))
2026             print("Testcase 99 Passed")
2027
2028     except NoSuchElementException:
2029         print("Element does not exist Test Case 99 Failed")
2030
2031     def testcase199(self):
2032         try:
2033             l = browser.find_element(By.ID, "review_overall_rating_slider")
2034             s = l.text
2035             location = l.location
2036             size = l.size
2037             w, h = size["width"], size["height"]
2038             tw = 576
2039             th = 2
2040             xcor = 212
2041             ycor = 1123
2042             x = location.get("x")
2043             y = location.get("y")
2044             # print("width - ", w)
2045             # print("height - ", h)
2046             # print("xcor - ", x)
2047             # print("ycor - ", y)
2048             self.assertEqual((w, h), (tw, th))
2049             self.assertEqual((x, y), (xcor, ycor))
2050             print("Testcase 100 Passed")
2051
2052     except NoSuchElementException:
2053         print("Element does not exist Test Case 100 Failed")
2054
2055     def testcase200(self):
2056         try:
2057             l = browser.find_element(By.ID,
2058             "review_overall_rating_slider_rangeValue")
2059             s = l.text
2060             location = l.location
2061             size = l.size
2062             w, h = size["width"], size["height"]
2063             tw = 20
2064             th = 22
```

```
2064         xcor = 790
2065         ycor = 1113
2066         x = location.get("x")
2067         y = location.get("y")
2068         # print("width - ", w)
2069         # print("height - ", h)
2070         # print("xcor - ", x)
2071         # print("ycor - ", y)
2072         self.assertEqual((w, h), (tw, th))
2073         self.assertEqual((x, y), (xcor, ycor))
2074         print("Testcase 101 Passed")
2075
2076     except NoSuchElementException:
2077         print("Element does not exist Test Case 101 Failed")
2078
2079 def testcase201(self):
2080     try:
2081         l = browser.find_element(By.ID,
2082 "review_overall_rating_slider_rangeValue")
2083         s = l.text
2084         value = "5"
2085         self.assertEqual(s.strip(), value.strip())
2086         print("Testcase 102 Passed")
2087     except NoSuchElementException:
2088         print("Element does not exist Test Case 102 Failed")
2089
2090 def testcase202(self):
2091     try:
2092         l = browser.find_element(By.ID, "button_submit_review")
2093         s = l.text
2094         location = l.location
2095         size = l.size
2096         w, h = size["width"], size["height"]
2097         tw = 600
2098         th = 45
2099         xcor = 210
2100         ycor = 1170
2101         x = location.get("x")
2102         y = location.get("y")
2103         # print("width - ", w)
2104         # print("height - ", h)
2105         # print("xcor - ", x)
2106         # print("ycor - ", y)
2107         self.assertEqual((w, h), (tw, th))
2108         self.assertEqual((x, y), (xcor, ycor))
2109         print("Testcase 103 Passed")
2110
2111     except NoSuchElementException:
2112         print("Element does not exist Test Case 103 Failed")
2113
2114 def testcase203(self):
2115     try:
2116         l = browser.find_element(By.ID, "user_submit_review_button")
2117         s = l.text
2118         location = l.location
2119         size = l.size
2120         w, h = size["width"], size["height"]
2121         tw = 600
2122         th = 45
2123         xcor = 210
```

```
2123     ycor = 1170
2124     x = location.get("x")
2125     y = location.get("y")
2126     # print("width - ", w)
2127     # print("height - ", h)
2128     # print("xcor - ", x)
2129     # print("ycor - ", y)
2130     self.assertEqual((w, h), (tw, th))
2131     self.assertEqual((x, y), (xcor, ycor))
2132     print("Testcase 104 Passed")
2133
2134 except NoSuchElementException:
2135     print("Element does not exist Test Case 104 Failed")
2136
2137 def testcase204(self):
2138     try:
2139         l = browser.find_element(
2140             By.ID, "navigation_to_login_pagefrom_the_review_page"
2141         )
2142         s = l.text
2143         location = l.location
2144         size = l.size
2145         w, h = size["width"], size["height"]
2146         tw = 600
2147         th = 45
2148         xcor = 210
2149         ycor = 1231
2150         x = location.get("x")
2151         y = location.get("y")
2152         # print("width - ", w)
2153         # print("height - ", h)
2154         # print("xcor - ", x)
2155         # print("ycor - ", y)
2156         self.assertEqual((w, h), (tw, th))
2157         self.assertEqual((x, y), (xcor, ycor))
2158         print("Testcase 105 Passed")
2159
2160     except NoSuchElementException:
2161         print("Element does not exist Test Case 105 Failed")
2162
2163 def testcase205(self):
2164     try:
2165         l = browser.find_element(By.ID,
2166         "label_for_login_button_on_review_page")
2167         s = l.text
2168         location = l.location
2169         size = l.size
2170         w, h = size["width"], size["height"]
2171         tw = 600
2172         th = 18
2173         xcor = 210
2174         ycor = 1231
2175         x = location.get("x")
2176         y = location.get("y")
2177         # print("width - ", w)
2178         # print("height - ", h)
2179         # print("xcor - ", x)
2180         # print("ycor - ", y)
2181         self.assertEqual((w, h), (tw, th))
2182         self.assertEqual((x, y), (xcor, ycor))
```

```
2182         print("Testcase 106 Passed")
2183
2184     except NoSuchElementException:
2185         print("Element does not exist Test Case 106 Failed")
2186
2187     def testcase206(self):
2188         try:
2189             l = browser.find_element(By.ID,
2190 "label_for_login_button_on_review_page")
2191             s = l.text
2192             value = "Not yet ready to review ? No problem, come back later to
2193 submit your review."
2194             self.assertEqual(s.strip(), value.strip())
2195             print("Testcase 107 Passed")
2196         except NoSuchElementException:
2197             print("Element does not exist Test Case 107 Failed")
2198
2199     def testcase207(self):
2200         try:
2201             l = browser.find_element(By.ID, "logout_button_on_review_page")
2202             s = l.text
2203             location = l.location
2204             size = l.size
2205             w, h = size["width"], size["height"]
2206             tw = 600
2207             th = 45
2208             xcor = 210
2209             ycor = 1266
2210             x = location.get("x")
2211             y = location.get("y")
2212             # print("width - ", w)
2213             # print("height - ", h)
2214             # print("xcor - ", x)
2215             # print("ycor - ", y)
2216             self.assertEqual((w, h), (tw, th))
2217             self.assertEqual((x, y), (xcor, ycor))
2218             print("Testcase 108 Passed")
2219
2220         except NoSuchElementException:
2221             print("Element does not exist Test Case 108 Failed")
2222
2223     def testcase208(self):
2224         try:
2225             username = browser.find_element(By.ID, "review_username_input")
2226             username.clear()
2227             username.send_keys("entering my username here as a value")
2228
2229             email = browser.find_element(By.ID, "review_user_email_input")
2230             email.clear()
2231             email.send_keys("entering my email here as a value")
2232
2233             textarea = browser.find_element(
2234                 By.ID, "custom_review_text_area_box_ui_html"
2235             )
2236             textarea.clear()
2237             textarea.send_keys("entering my suggestions here as a value")
2238
2239             browser.find_element(By.ID, "user_submit_review_button").click()
2240
2241             l = browser.find_element(By.ID, "homeimage")
```

```
2240     s = "Found"
2241     self.assertEqual(s, "Found")
2242     print("Testcase 109 Passed")
2243
2244 except:
2245     print("Element does not exist Test Case 109 Failed")
2246
2247
2248
2249 if __name__ == "__main__":
2250     unittest.main()
2251
```