****

**Spring, 2022-2023**

**SE2226 - Software Quality Assurance and Testing**

**Term Project Report**

|  |  |
| --- | --- |
| **Project Name:** | Vivense Website Testing |
| **Student Name:** | Doğa TAN  İrem GÖKCEN |
| **Student ID:** | 20070006002  20070006015 |
| **Department:** | **Software Engineering** |
| **Course Section No:** | **1** |

1. **INTRODUCTION**

Vivense is an online shopping site that offers modern furniture designs and decoration products. Founded in 2015, the company provides services in Europe, the Middle East and North Africa as well as Turkey.

Vivense has reached more than 2 million users to date and is increasing the number of users every day. The products on the site are in different categories such as furniture, lighting, decoration, kitchen and bathroom. Within each category, there are many options to suit different styles and needs.

Vivense adopts customer satisfaction as its top priority. With its user-friendly interface and detailed product descriptions, the site enables customers to easily find products suitable for their needs. Products are produced in accordance with high quality standards and offered to customers at the most affordable prices.

In our project, we tested the Vivense website ( <https://www.vivense.com/> ) using blackbox and whitebox testing techniques.

1. **TESTING**
   1. **BLACKBOX TESTING:**

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed.( <https://www.javatpoint.com/black-box-testing> )

In our Project, the blackbox testing techniques we used are:

* Equivalence Partitioning Testing
* Boundary Value Analysis Testing
* Decision Table Testing
* State Transition Testing
* Use Case Testing

**2.1.1 DECISION TABLE TESTING FOR LOGIN**

We used the decision table to determine the results depending on whether the information entered on the login screen is valid or invalid.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** |
| **Conditions** |  |  |  |  |
| Valid e-mail | **T** | **T** | **F** | **F** |
| Valid password | **T** | **F** | **T** | **F** |
| **Actions** |  |  |  |  |
| Successful Login | **Y** | **N** | **N** | **N** |
| Error Message | **N** | **Y** | **Y** | **Y** |

Tablo 1-Decision Table for LOGIN

**2.1.2 STATE TRANSITION TESTİNG FOR LOGİN**

In the login section, we used the state transition testing method to test the transitions between states and whether the system responds correctly to events.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Enter Website | Enter Password & e-Mail | Valid pasword & e-Mail | Invalid password & e-Mail |
| S1- Start State | S2 | - | - | - |
| S2- Login Page | - | S3 | - | - |
| S3- Check e-Mail and Password | - | - | S4 | S2 |
| S4- Access to Account | - | - | ? | ? |

Tablo 2-State Table for LOGIN

State Diagram-1 for LOGIN


Şekil 1-State Diagram for LOGIN

**2.1.3 DECISION TABLE TESTING FOR REGISTRATION**

While testing the registration part, we used the decision table to identify and analyze all conceivable combinations of requirements.

**metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Tablo 3-Decision Table for REGISTRATION

**2.1.4 EQUIVALENCE PARTITIONING AND BOUNDARY VALUE ANALYSIS TESTING FOR PASSWORD CHARACTER LIMIT ON REGISTRATION**

Vivense website imposes some restrictions on the password to be set in the registration section. It states that the entered password must contain a minimum of 5 characters and a maximum of 120 characters. In this part, we used equivalence partitioning and boundary analysis testing techniques to determine the test values.

|  |  |  |
| --- | --- | --- |
| **Equivalence Class** | **Test Input (Number of characters)** | **Expected Output (Valid/Invalid)** |
| **E1 5<= password <= 120** | **57** | **Valid Password** |
| **U1 password < 5** | **2** | **Invalid Password** |
| **U2 password > 120** | **123** | **Invalid Password** |

1. **EQUIVALENCE PARTITIONING**

Tablo 4-Equivalence Partitioning Table for PASSWORD

1. **BOUNDARY VALUE ANALYSIS**

|  |  |  |
| --- | --- | --- |
| **Equivalence Class** | **Test Input** | **Expected Output** |
| **E1 5<= password <= 120** | **5** | **Valid Password** |
|  | **6** | **Valid Password** |
|  | **119** | **Valid Password** |
|  | **120** | **Valid Password** |
| **U1 password < 5** | **4** | **Invalid Password** |
| **U2 password > 120** | **121** | **Invalid Password** |

Tablo 5-Boundary Value Analysis Table for PASSWORD

**2.1.5 USE CASE TESTING FOR ORDER PRODUCT**

In this section, we used use case testing to explain the product ordering process step-by-step from start to finish and define test cases covering the entire system.

|  |  |
| --- | --- |
| **Use Case Name:** Order Product | **Primary Actor:** Customer |
| **Normal flow of events:**   1. The customer logs in to account with e-mail and password. 2. The system verifies the customer’s email and password. 3. The customer searches for the desired product by typing it in the search bar. 4. The system lists the possible results to the customer. 5. The customer selects the desired product and adds it to the shopping cart. 6. The customer chooses the quantity of the product they add to their shopping cart. 7. System confrims availability of the product (in the requested quantity) and displays the purchase summary. 8. The Customer selects the address. 9. Customer selects the method of payment and enters the corresponding account information. 10. System interacts with the payment authorization system to carry out the payment. 11. The system sends the customer a message that the payment has been confirmed. 12. Order is completed. | |
| **Alternate/exceptional flows:**  **1A1: The customer does not have an account.**   * The customer creates a new account by going to the registration page and entering the necessary information. * The use case continues from step one.   **2A1: The customer enters incorrect password**   * The system displays the error message and allows the customer to try to log in again. * Customer enters the correct password. * The use case continues**.**   **2A2: The customer enters invalid e-mail.**   * The system displays the error message and allows the customer to try to log in again. * Customer enters the valid e-mail. * The use case continues.   **4A1: Customer enters an invalid product name.**   * No products are listed and the customer is expected to search for products again. * The use case continues from step four.   **8A1: The customer does not have an address saved in the system.**   * The customer enters a new address from the add address section. * The use case continues.   **11A1: The payment information entered by the customer is incorrect.**   * The system sends a message to the customer that the entered information is incorrect. * The customer checks the payment information and enters it again. * The system validates the information and sends the payment confirmation message to the customer. | |

**metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Tablo 6-Use Case Testing Table for Order Product

* 1. **WHITEBOX TESTING:**

**White box testing** techniques analyze the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

White box testing is also known as structural testing or code-based testing, and it is used to test the software’s internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

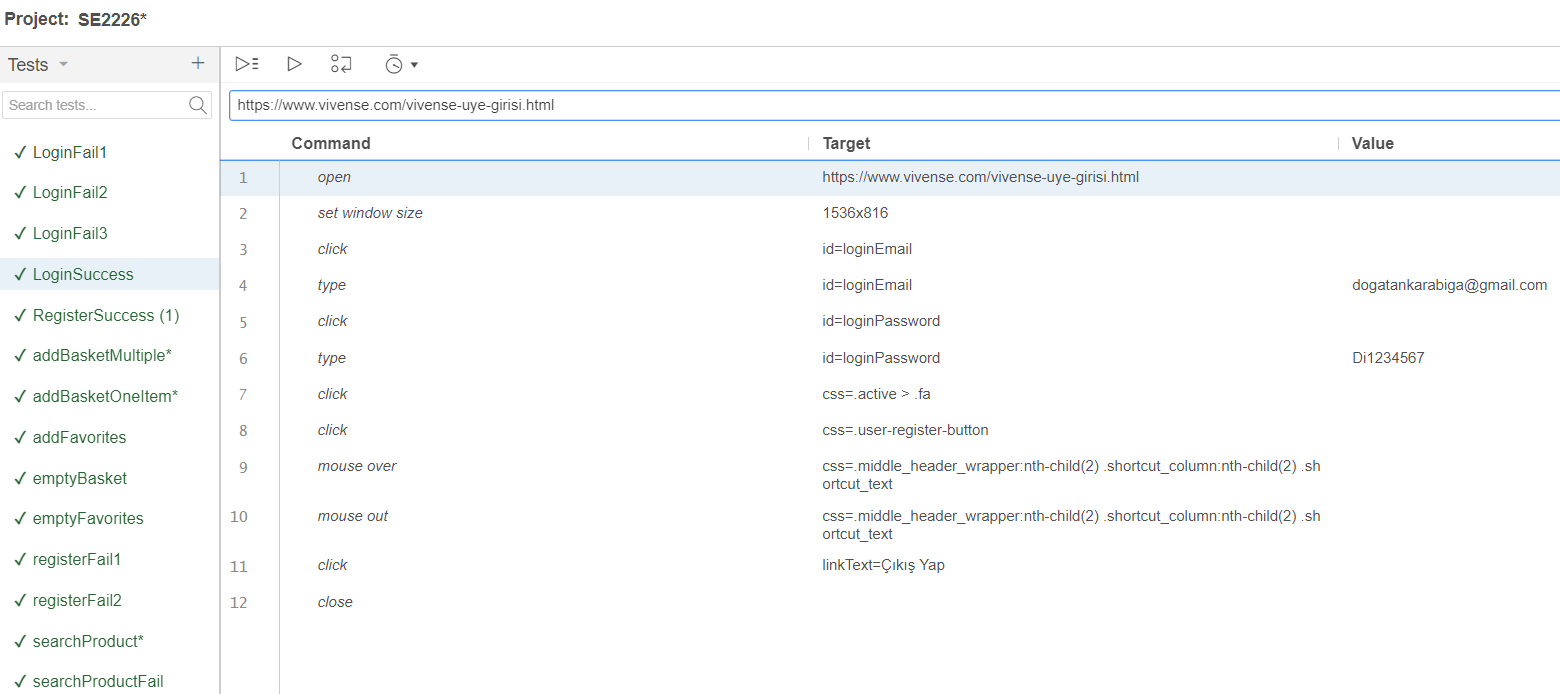
( <https://www.geeksforgeeks.org/software-engineering-white-box-testing/> )

* + 1. **SELENIUM**

In this part of the project, we tested the functions of the site such as login, registration, search, adding products to the cart with the selenium web driver. All test cases are automated by Selenium Web Driver successfully. You can view the details in the .side file.

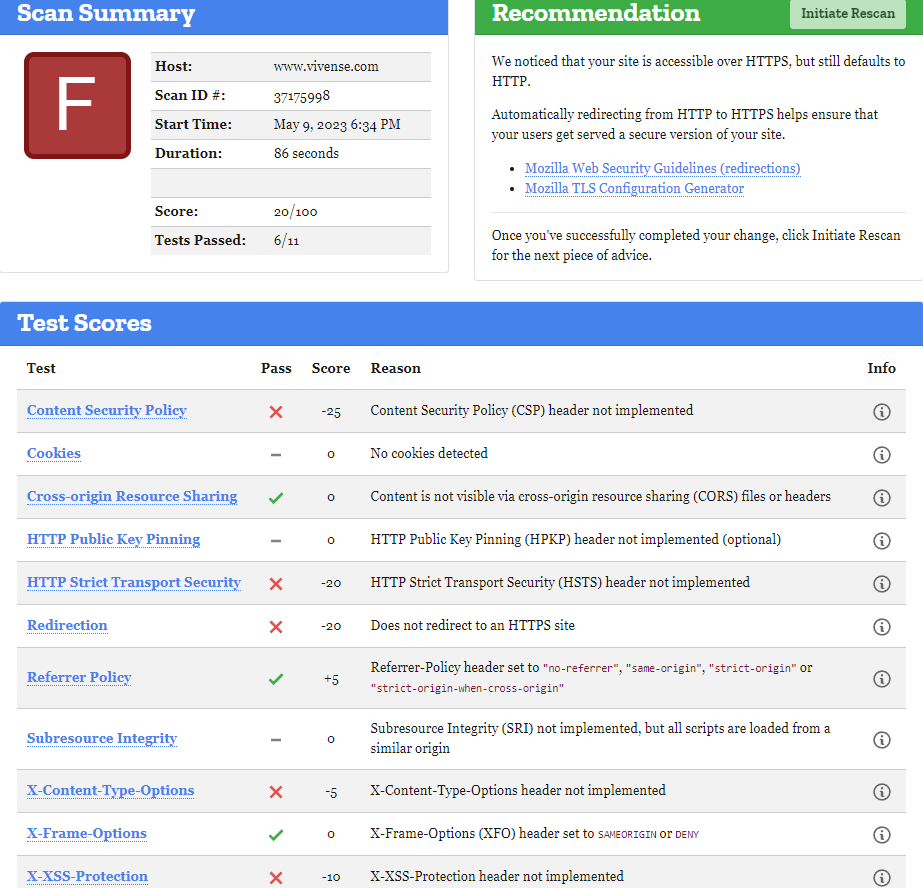
NOTE:

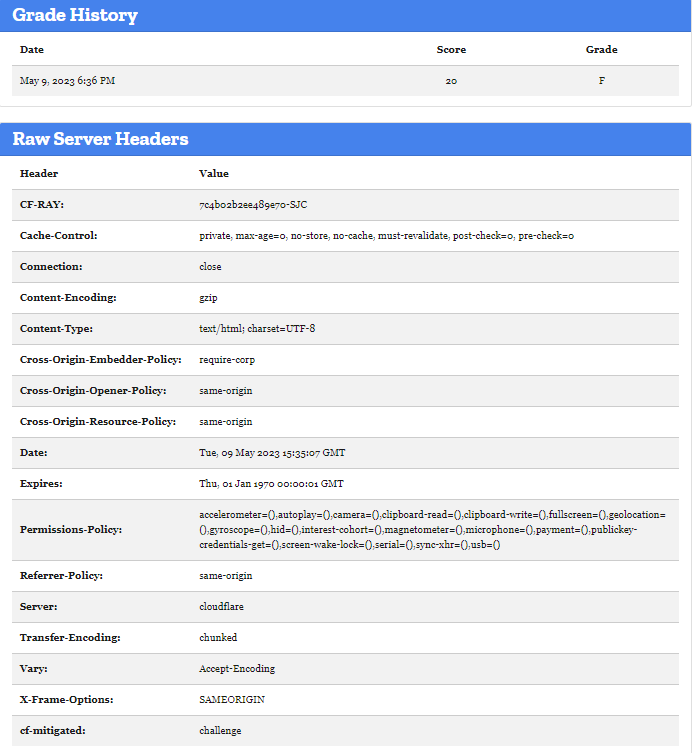
* Tests need to be initialized to run properly. For example, if a product was added to the basket in the previous test case, the basket should be emptied and the other case should continue.
* If the test case remains logged on the login screen, it must be logged out and run again.

****

* + 1. **SECURITY TESTING**

For security testing, Mozilla Observatory (https://observatory.mozilla.org/) is used.

****

****

* + 1. **USABILITY TESTING**

We have tested this site in the following web browsers and observed that the site can work successfully.

- Google Chrome

- Safari

- Opera

- Mozilla Firefox

- Microsoft Edge