# Online Suit Store Testing

## Signed:

Sean Doyle – B00156175 – 15/04/2024

Adam Ennis – B00152710 – 15/04/2024

Alex -

# TABLE OF CONTENTS

# Introduction:

The testing of our online suit store will be broken down into three categories.

1. Unit Testing:

Our first test will consist of unit testing. Five functions will be tested using unit testing and we will do them using the white and black box testing methods.

\*\*name the 5 functions and say which ones are going to be white box tested and which are going to be black box tested

1. Acceptance Test:

Our second test is an acceptance test, this will consist of going over are requirements document and looking at what we set out to do and compare it to what was actually done, we will look at any discrepancies and address them in our testing.

1. Usability Testing:

The final test we will do is usability testing this involves going through the website as a user would to look for any bugs that might occur under certain conditions. We will test, the login/register functionality, the cart functionality, the product filter functionality and all the basic functionality of the website.

# Unit Testing:

## Unit 1: Show Products

### *Black Box test*

We wanted to test our products to make sure that they all function correctly.

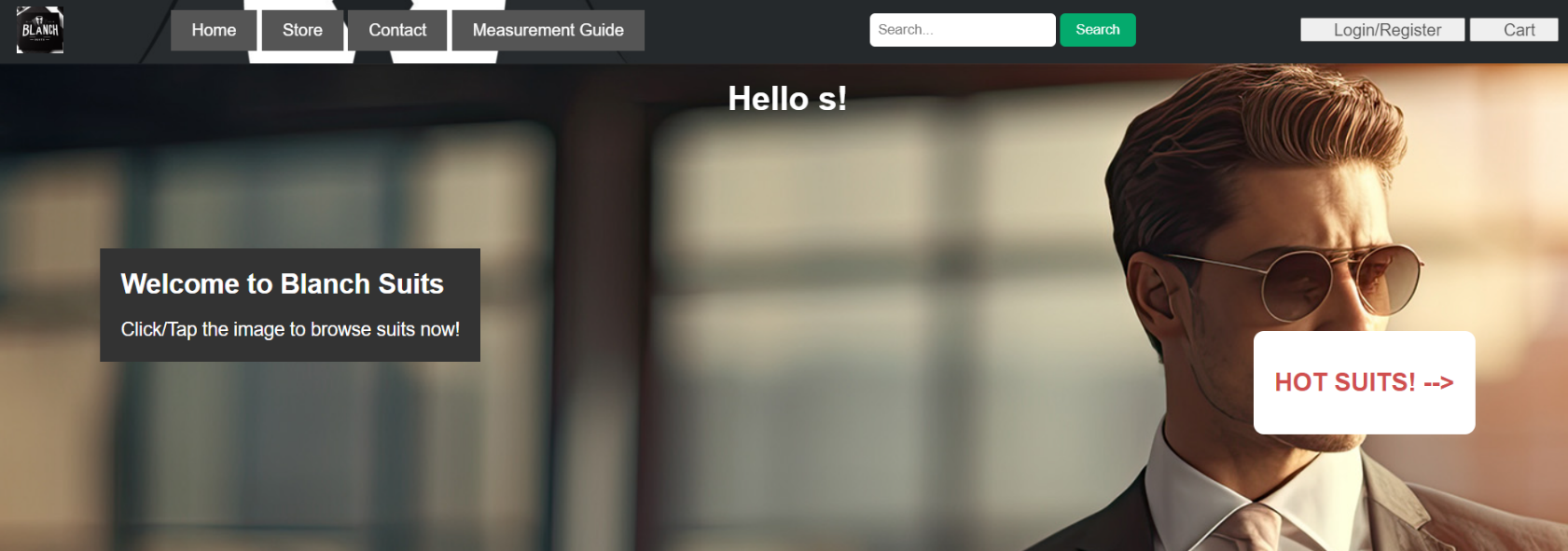
**Equivalence partitioning** will be used to test this function. **Equivalence classes** will be derived as the following.

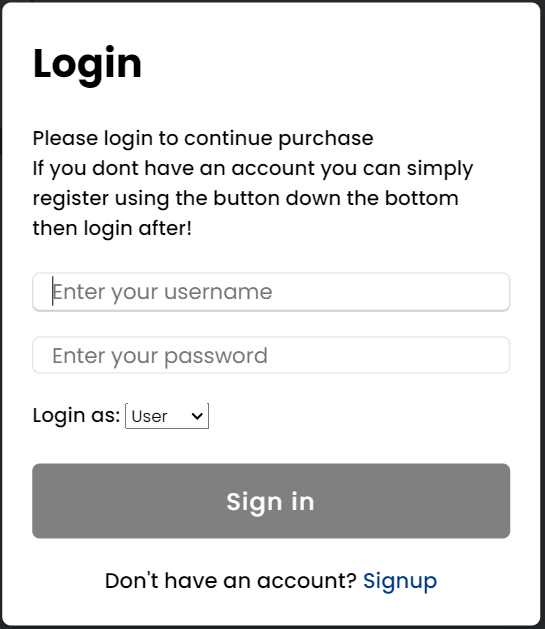
1. V
2. V
3. V
4. V

|  |  |  |  |
| --- | --- | --- | --- |
| # | Test Data | Expected Outcome | Result |
| 1 | The image shows | The image will show | F |
| 2 | The name of the product will show | The name shows | T |
| 3 | The price of the product will show | The price shows | T |
| 4 | The Brand name of the product will show | The brand name shows | T |
| 5 | The Size of the product will show | The size shows | T |
| 6 | The Colour of the image will show | The colour shows | T |

# User Interface testing:

## Guidance

In our website we have quite a few ways that we guide the user to do an action on the website. For example, on our home page we have a note that says, “click the image to browse the store”. We added this feature so as soon as customers go onto our website, they know exactly what to do. We also have a button on the website that says, “Hot suits!” this when clicked directs you to the suits page.

On our login page it tells the customer exactly what to do to sign in or to sign up as shown in the image below

## Feedback

In our website when someone is trying to log in or register on the website, we give the customer feedback. You can see in the image below when you are browsing the store if not logged in, we give guidance and let the user know they will need to login to view the cart, but when logged in it says:” Welcome back (username)”

A screenshot of a phone

Description automatically generated

A screenshot of a product

Description automatically generated

A screenshot of a computer

Description automatically generatedWhen someone is registering and inserts just the name and presses register it says

We give feedback on nearly all steps of the login and register.

We use feedback in our website to make the website easy to use for our customers.

## Consistency

In the project we use the Sorry lads I haven’t a clue what to say for this

## Minimal clicks

In our website we have a feature that supports the use of minimal clicks. On our home page you can click the image which will take you to the store then on the store we have a buy now feature that lets you buy the suit straight away instead of adding it to the cart first and then you having to access the cart and then buying. The use of minimal clicks in a website is very important as it allows the customer to save time when buying products.

#### 2.2.4 Cyclomatic complexity:

From the flow graph we can discover how many independent paths are in the function.

E – number of edges = \*?\*

N – number of nodes = \*?\*

Cyclomatic complexity = E - N + 2.

CC = \*?\* - \*?\* + 2

We can now document the basis set for these paths:

#### 2.2.5 Paths:

1. 1 - 2 - 3 - 4 - 5 - 6 - 7 -10
2. 1 - 2 - 3 - 4 - 5 - 6 - 7 -10
3. 1 - 2 - 3 - 4 - 5 - 6 - 7 -10
4. 1 - 2 - 3 - 4 - 5 - 6 - 7 -10
5. 1 - 2 - 3 - 4 - 5 - 6 - 7 -10

#### 2.2.6 Test cases for execution of each path:

Here we will test the path case and the functionality of each path.

|  |  |  |  |
| --- | --- | --- | --- |
| Path | Test Attempted | Expected Outcome | Result |
| 1 |  |  |  |
| 2 | SAMPLE | SAMPLE | SAMPLE |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

#### 2.2.7 Path 1:

INSERT IMAGE HERE

#### 2.2.8 Path 1:

INSERT IMAGE HERE

#### 2.2.9 Path 1:

INSERT IMAGE HERE

## User(Signup)

We want this to accept certain passwords only and the right format for a number

### *Black Box test (signUp)*

Method User() accepts password String parameters

* **isValidPassword –** Represents a ruleset for password parameters
* **phoneNo -** //I have YET to implement this – ADC

### Derived Equiv classes

1. User passes a password with a string length of 0-7 (Invalid)
2. User passes a password with a string length of 0-7 including special character (Invalid)
3. User passes a password with 8 or more characters without special character (Invalid)
4. Password left blank (Invalid)
5. User passes a password with 8 or more characters with special character (Valid)
6. User tries to input a SQL injection to bypass passwords and fails (Valid?)
7. Us

## Login

I secured the login from SQL injections (hopefully); and paired the user’s password input to the hashed password in the databased.

### *Black Box test (Login)*

* **username –** Takes in any String of characters, CLEANS UP FROM SQL INJECTIONS
* **password –** Take in any String of characters, CLEANS UP FROM SQL INJECTIONS

## Requirements testing

|  |  |  |
| --- | --- | --- |
| # | Requirements | Pass/Fails |
| 1 | Users can login to their session | P |
| 2 | Users can sign up for an account | P |
| 3 | Users sign ups are UNIQUE | F |
| 4 | Users |  |