# Online Suit Store Testing

## Signed:

Sean Doyle – B00156175 – 15/04/2024

Adam -

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.

# Testing:

## Black Box:

The function we are testing is “displayFilteredSuits” which is the filter option on the website.  
We will test this by going on the website and trying different filters.

We will use equivalence partitioning; the equivalence classes are as follows:

1. Colour Filter
2. Size Filter
3. Tailored Filter
4. Brand Filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Test Data | Expected Outcome | Classes Covered | Result |
| 1 | The red colour filter is applied | Only suits with red colour show | A | T |
| 2 | The ‘M’ size filter is applied | Only suits of ‘M’ size show | B | T |
| 3 | The Hugo Boss filter option is applied | Only Hugo boss suits show | D | F |
| 4 | The green colour filter and the tailored filter is applied | Only green tailored suits show | A, C | T |
| 5 | The red colour filter and the Armani filter is applied | Only red Armani suits show | A, D | F |

## White Box:

### 2.2.1 Basis Path Testing:

#### 2.2.2 Flow Graph-Code Snippet:

Function is displayed as a mapped version.

    public function displayFilteredSuits($filters) {

    --1--$pdo = get\_connection();

    --2--$query = "SELECT \* FROM products WHERE 1=1";

*/\*\*\*--1--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*// Append conditions for each filter*

    --3--if (!empty($filters['colors'])) {

       $query .= " AND colors = :colors";

    }

    --4--if (!empty($filters['size'])) {

    $query .= " AND size = :size";

    }

    --5--if (!empty($filters['tailored'])) {         *//Tailored suits not made yet*

    $query .= " AND tailored = :tailored";

    }

    --6--if (!empty($filters['brandName'])) {

    $query .= " AND brandName = :brandName";

    }

    --7--$stmt = $pdo->prepare($query);

    --8--if (!empty($filters['colors'])) {

    $stmt->bindParam(':colors', $filters['colors']);

    }

    --9--if (!empty($filters['size'])) {

    $stmt->bindParam(':size', $filters['size']);

    }

    --10--if (!empty($filters['tailored'])) {

    $stmt->bindParam(':tailored', $filters['tailored']);

    }

    --11--if (!empty($filters['brandName'])) {

    $stmt->bindParam(':brandName', $filters['brandName']);

    }

    --12--$stmt->execute();

    --13--$suits = $stmt->fetchAll();

    --14--echo '<div class="product-grid">';

    foreach ($suits as $suit) {

    $suits = $this->get\_suits();

    echo '<div class="filter-product">';

    echo '  <blockquote>';

    echo '  <img src="' . $suit['imageURL'] . '" alt="' . $suit['productName'] . '">';

    echo '    <h2>' . $suit['productName'] . '</h2>';

    echo '    <h2>' . $suit['price'] . '</h2>';

    echo '    <h2>' . $suit['brandName'] . '</h2>';

    echo '    <h2>' . $suit['colors'] . '</h2>';

    echo '    <h2>' . $suit['size'] . '</h2>';

    echo '    <a href="?action=addToCart&id=' . $suit['productID'] . '" class="cartB"><p>Add to Basket</p></a>';

    echo '    <a href="?action=rmCart&id=' . $suit['productID'] . '" class="cartB-rem"><p>Remove from Basket</p></a>';

    echo '  </blockquote>';

    echo '</div>';

   }

    echo '</div>';

    --15--return $suits;

#### 2.2.3 Flow Graph:

\*\*\*THIS IS A RANDOM FLOW GRAPH – IM TIRED – SD \*\*\*\*\* \*\*\*\*\*LOL NOT ANYMORE-ADC\*\*\*\*\*

A diagram of a diagram

Description automatically generated

#### 2.2.4 Cyclomatic complexity:

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

E – number of edges = 27

N – number of nodes = 15

Cyclomatic complexity = E – N + 2 = 27-15+2 = 15

We can now document the basis set for these paths:

#### 2.2.5 Paths:

\*\* Pathways within edges 8, 9, 10, 11 (\*\*\*) are interchangeable with each other

1. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 12 – 13 – 14 – 15
2. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 12 – 13 – 14 – 15
3. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 12 – 13 – 14 – 15
4. 1 – 2 – 3 – 4 – 5 – 6 – 7 -10 – 12 – 13 – 14 – 15
5. 1 – 2 – 3 – 4 – 5 – 6 – 7 -11 – 12 – 13 – 14 – 15
6. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
7. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
8. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
9. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 11 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
10. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 11 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
11. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
12. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 11 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
13. 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 11 – 12 – 13 – 14 – 15 // Paths \*\*\* are interchangeable
14. 1 – 2 - 3 - 4 - 5 - 6 - 7 – 10 – 11 - 12 - 13 – 14 - 15

#### 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 | What do I write here?  -ADC | Returns All | T |
| 2 |  | Returns chosen color | T |
| 3 |  | Returns chosen size | T |
| 4 |  | Returns chosen tailored suits | T + N/A |
| 5 |  | Returns chosen brand | F |
| 6 |  | Returns chosen size and color | T |
| 7 |  | Returns chosen size, color and tailor suits | T + N/A |
| 8 |  | Returns chosen size, color, tailor suits and brand | F |
| 9 |  | Returns chosen size, tailor suits and brand | F |
| 10 |  | Returns chosen color and brand | F |
| 11 |  | Returns chosen color and tailor suits | T + N/A |
| 12 |  | Returns chosen color, tailor suits, and brand | F |
| 13 |  | Returns chosen color and brand | F |
| 14 |  | Returns tailored suit and brand | F |

// N/A can also be null since the modules within the query selector works with the tailored suits even if the since tailored suits has not been made it will return

#### 2.2.7 Path 1:

#### A screenshot of a computer Description automatically generated

#### 2.2.8 Path 1:

#### A screenshot of a computer Description automatically generated

#### 2.2.9 Path 1:

#### A screenshot of a computer Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a black and white website

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a clothing store

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

#### 2.2.9 Path 1:

A screenshot of a computer

Description automatically generated

Another function we are testing is the “addProduct” which is the the way the admin can add new products to the store page.

We will use equivalence partitioning; the equivalence classes are as follows:

1. Product Name
2. Price
3. Brand Name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Test Data | Expected Outcome | Classes Covered | Result |
| 1 | A Product name is inserted only | Please input the two remaining fields | A | T |
| 2 | A price is inserted only | Please input the two remaining fields | B | T |
| 3 | A Brand name is inserted only | Please input the two remaining fields | C | T |
| 4 | A Product name and a Price is inserted | Please input the one remaining field | A, B | T |
| 5 | A Product name and a Brand name is inserted | Please input the one remaining field | A, C | T |
| 6 | A Price and a Brand name is inserted | Please input the one remaining field | B,C | T |
| 7 | All fields are inserted | The product will be added | A,B,C | T |