# 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'])) {

    --4--    $query .= " AND colors = :colors";

    }

    --5--if (!empty($filters['size'])) {

    --6--    $query .= " AND size = :size";

    }

    --7--if (!empty($filters['tailored'])) {         *//Tailored suits not made yet*

    --8--    $query .= " AND tailored = :tailored";

    }

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

    --10--    $query .= " AND brandName = :brandName";

    }

    --11--$stmt = $pdo->prepare($query);

    --12--if (!empty($filters['colors'])) {

    --13--    $stmt->bindParam(':colors', $filters['colors']);

    }

    --14--if (!empty($filters['size'])) {

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

    }

    --15--if (!empty($filters['tailored'])) {

    --16--    $stmt->bindParam(':tailored', $filters['tailored']);

    }

    --17--if (!empty($filters['brandName'])) {

    --18--    $stmt->bindParam(':brandName', $filters['brandName']);

    }

    --19--$stmt->execute();

    --20--$suits = $stmt->fetchAll();

    --21--echo '<div class="product-grid">';

    --22--foreach ($suits as $suit) {

    --23--    $suits = $this->get\_suits();

    --24--    echo '<div class="filter-product">';

    --25--    echo '  <blockquote>';

    --26--    echo '  <img src="' . $suit['imageURL'] . '" alt="' . $suit['productName'] . '">';

    --27--    echo '    <h2>' . $suit['productName'] . '</h2>';

    --28--    echo '    <h2>' . $suit['price'] . '</h2>';

    --29--    echo '    <h2>' . $suit['brandName'] . '</h2>';

    --30--    echo '    <h2>' . $suit['colors'] . '</h2>';

    --31--    echo '    <h2>' . $suit['size'] . '</h2>';

    --32--    echo '    <a href="?action=addToCart&id=' . $suit['productID'] . '" class="cartB"><p>Add to Basket</p></a>';

    --33--    echo '    <a href="?action=rmCart&id=' . $suit['productID'] . '" class="cartB-rem"><p>Remove from Basket</p></a>';

    --34--    echo '  </blockquote>';

    --35--    echo '</div>';

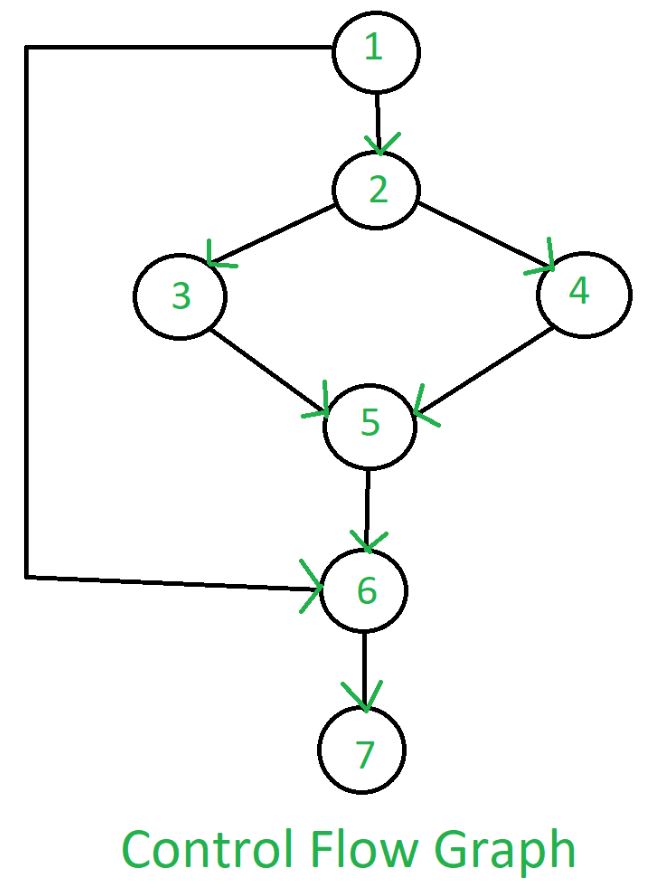
    --36--}

    --37--echo '</div>';

    --39--return $suits;

#### 2.2.3 Flow Graph:

\*\*\*THIS IS A RANDOM FLOW GRAPH – IM TIRED – SD \*\*\*\*\*



#### 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