# Online Suit Store Testing

## Signed:

Sean Doyle – B00156175 – 15/04/2024

Adam Ennis – B00152710 – 15/04/2024

Alex Anthony Dela Cruz – B00149504 – 15/04/2024

Table of Contents

[Online Suit Store Testing 1](#_Toc165218853)

[Signed: 1](#_Toc165218854)

[1. Introduction: 3](#_Toc165218855)

[2. Class testing: 3](#_Toc165218856)

[2.1 Show Products 3](#_Toc165218857)

[2.1.1 Black Box test 3](#_Toc165218858)

[2.1.2 Unit Test 4](#_Toc165218859)

[2.2 Orders 5](#_Toc165218860)

[2.2.1 Blackbox Test 5](#_Toc165218861)

[2.2.2 Unit Test 5](#_Toc165218862)

[2.3 Checkout 6](#_Toc165218863)

[2.3.1 Blackbox Test 6](#_Toc165218864)

[2.3.2 Unit Test 6](#_Toc165218865)

[2.4 Signup 8](#_Toc165218866)

[2.4.1 Black Box test 8](#_Toc165218867)

[2.4.2 Derived Equivalence classes 8](#_Toc165218868)

[2.4.3 Blackbox Test based on Equivalence 8](#_Toc165218869)

[2.4.4 White Box Test 9](#_Toc165218870)

[3. User Interface testing: 15](#_Toc165218871)

[3.1. Guidance 15](#_Toc165218872)

[3.2. Feedback 16](#_Toc165218877)

[3.3. Consistency 17](#_Toc165218883)

[3.4. Minimal clicks 18](#_Toc165218890)

[4 Requirements testing 18](#_Toc165218891)

# 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.

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.

# Class testing:

## Show Products

### Black Box test

displayFilteredSuits() allows for users to filter what suits they want to show up in the store:

* **colors -** Displays chosen color that is available
* **size -** Displays chosen size that is available
* **tailored -** Displays tailored suits if yes is chosen
* **brandName -** Displays chosen brand that is available

|  |  |  |  |
| --- | --- | --- | --- |
| # | 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 |

### Unit Test

A screen shot of a computer program

Description automatically generated

A screen shot of a product list

Description automatically generated

## Orders

### Blackbox Test

Orders reads from the products page and adds or removes those products into the cart

**BUY NOW –** Adds to the basket and transfers to the checkout page

**Add to Cart -** Adds to the cart checkout page

**Remove from cart -** Removes from cart in checkout page

### Unit Test

A screen shot of a computer program

Description automatically generated

A screenshot of a phone

Description automatically generated

## Checkout

### Blackbox Test

Users will be able to remove items they don’t want from their cart in the checkout. This is where they confirm their order which gets sent to the DB.

**Remove from cart -** Removes from cart in checkout page

**Confirm Purchase -** Sends order to the DB

### Unit Test

## Signup

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

### Black Box test

Method User() takes in 7 parameters:

* **firstname –** Pass in any character for first name
* **lastname –** Pass in any character for last name
* **username -** Pass in any character for username
* **email -** Pass in an email that needs a valid email format (HTML5)
* **password -** Password requires 8 characters and at least 1 special character
* **address -** Pass in any character for address
* **phone -** Requires phone number to be formatted with hyphens as XXX-XXX-XXXX

### Derived Equivalence 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. User passes a password with 8 or more characters with special character(s) (Valid)
5. User puts 0-9 digits for phone number (Invalid)
6. User puts 10 or more digits for phone number w/o format (Invalid)
7. User puts spaces for phone number (Invalid)
8. User puts special characters in the phone number (Invalid)
9. User puts 10 digits with proper formatting (Valid)
10. User puts 10 or more digits with hyphens (Invalid)
11. Positions of the phone and password are swapped (Invalid)

### Blackbox Test based on Equivalence

|  |  |  |  |
| --- | --- | --- | --- |
| # | Test Data | Expected outcome | Class covered |
| 1 | Password1@, 123-456-7890 | Object | 4, 9 |
| 2 | Pass1@, 123456789012 | Null | 1, 2, 6 |
| 3 | Password, | Null | 3, 5 |
| 4 | , 123-456-7890 | Null | 1, 9 |
| 5 | Password1@, 123-456-789 | Null | 4,5 |
| 6 | Password1@, 123\*456!7890 | Null | 4, 8 |
| 7 | Pass, 123 456 7890 | Null | 1, 7 |
| 8 | Password, 123-456-78901 | Null | 3, 10 |
| 9 | 123-456 7890, Password@! | Null | 4, 6, 7, 10, 11 |

### White Box Test

#### Flow graph

Function statements are numbers on the left side

1. function addUser($firstname, $lastname, $username, $email, $password, $address, $phone)
2. {
3. $pdo = get\_connection();
5. --1-- if (!$this -> isValidPassword($password)) {
6. --2-- return "Password must be at least 8 characters long and contain at least one special character."; // Password does not meet requirements
7. }
8. // Check if the phone number matches the XXX-XXX-XXXX format
9. --3-- if (!$this->isValidNumber($phone)) {
10. --4-- return "Phone number must be in XXX-XXX-XXXX format";
11. }
12. // Remove hyphens from the phone number and count digits
13. --5-- $Digits = preg\_replace('/[^0-9]/', '', $phone);
14. // Check if the phone number meets the length requirement
15. --6-- if (strlen($Digits) !== 10) {
16. --7-- return false; // Invalid phone number length
17. }
18. --8-- $encryptPass = password\_hash($password, PASSWORD\_DEFAULT); //Reintech.io //PASSWORD\_DEFAULT uses bcrypt
20. //insert user into database
21. --9-- $query = "INSERT INTO users (firstname, lastname, username, email, password, address, phone) VALUES (:firstname, :lastname, :username, :email, :password, :address, :phone)";
22. --10-- $stmt = $pdo->prepare($query);
23. --11-- $stmt->bindParam(':firstname', $firstname);
24. --12-- $stmt->bindParam(':lastname', $lastname);
25. --13-- $stmt->bindParam(':username', $username);
26. --14-- $stmt->bindParam(':email', $email);
27. --15-- $stmt->bindParam(':password', $encryptPass);
28. --16-- $stmt->bindParam(':address', $address);
29. --17-- $stmt->bindParam(':phone', $phone);
31. --18-- if ($stmt->execute()) {
32. --19-- return true; //user added successfully
33. --20-- } else {
34. --21-- return false; //error adding user
35. }
36. }

#### Path Flow

A diagram of a diagram

Description automatically generated

#### Cyclomatic complexity

Edges = 23

Nodes = 21

Cyclomatic complexity = 23 - 21 + 2 = 4

#### Paths

1 – 3 – 5 – 6 – 8 – 10 – 11 – 12 - 13 - 14 - 15 – 16 - 17 – 18 – 19

1 - 2 – 3 – 4 – 5 – 6 – 7 – 18 – 20 – 21

1 – 2 – 3 – 5 – 6 – 7 – 18 – 20 – 21

1 – 3 – 4 – 5 - 6 – 7 – 18 – 20 – 21

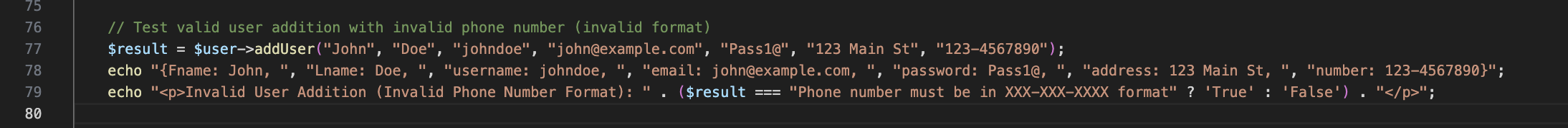
#### Unit Testing (With Path) \*\*\* NOT DRAWN YET -ADC

##### Path 1

##### 

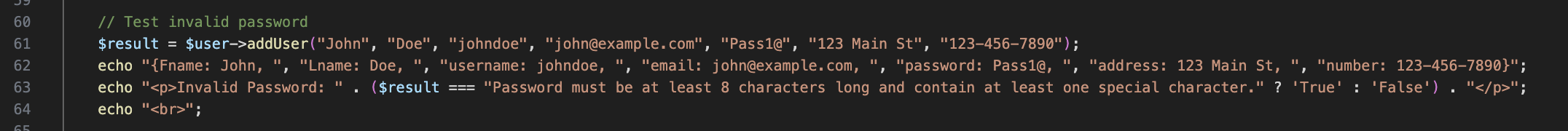
##### 

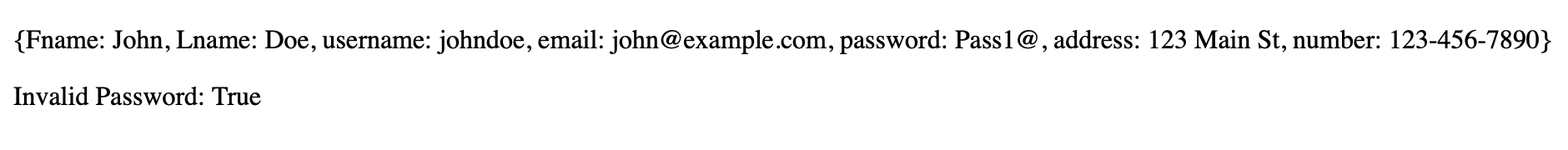
##### Path 2



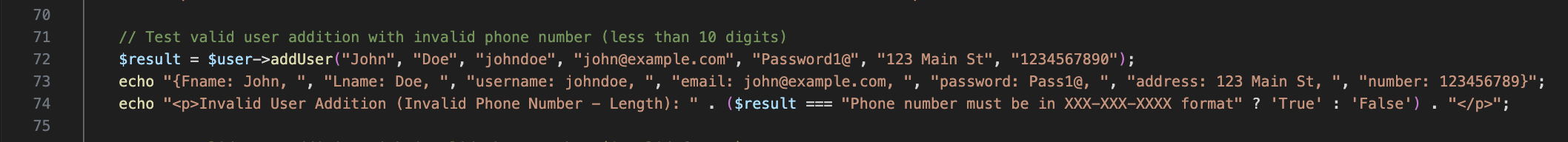


##### Path 3





##### Path 4





Mistake made in the echo. Otherwise all works and password is valid

#### Unit Testing

##### Password test

A computer screen shot of a program

Description automatically generated

A white background with black text

Description automatically generated

##### Phone number Test

##### A screen shot of a computer code Description automatically generated

##### A number and text on a white background Description automatically generated

##### Add User (Password)

##### A black background with text Description automatically generated

A close-up of a computer screen

Description automatically generated

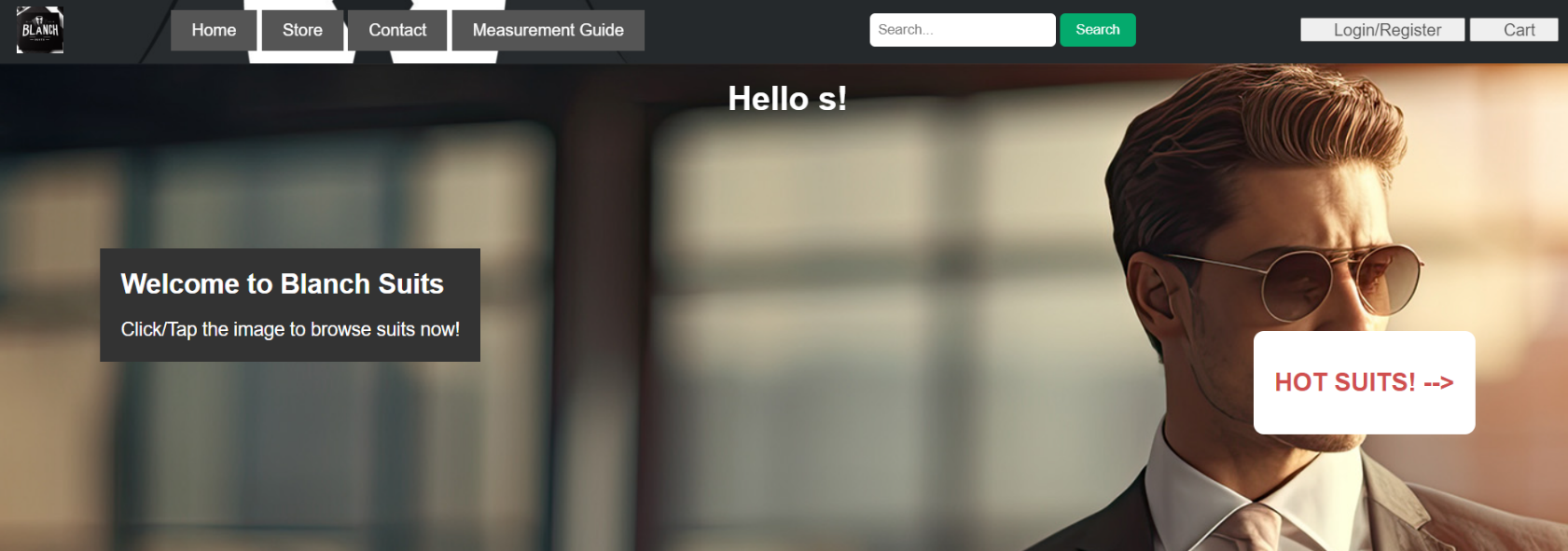
##### Add User (Phone Number)

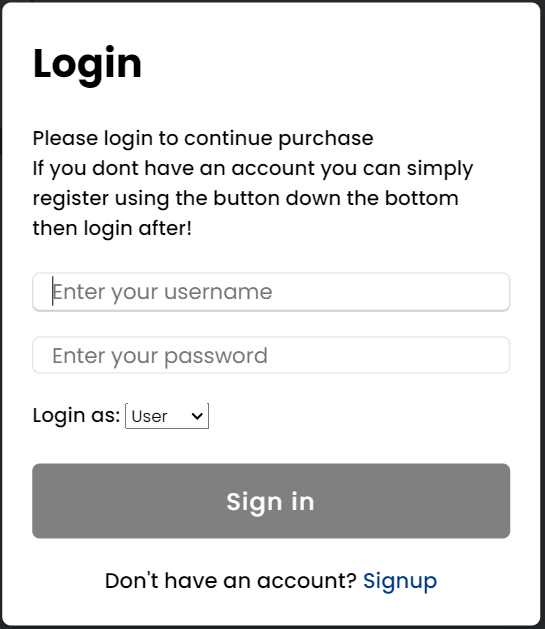
##### A screenshot of a computer program Description automatically generated

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

# 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

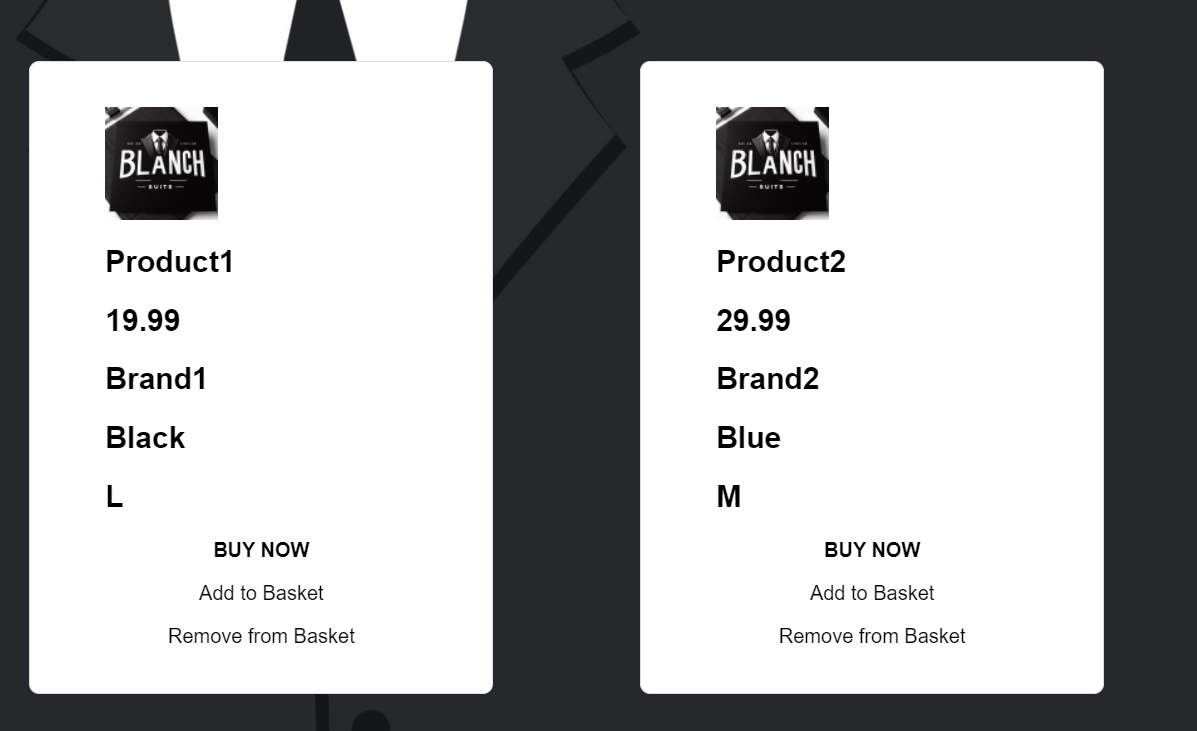
Description automatically generatedWhen someone is registering and inserts just the name and presses register it says

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 consistency through the header and footer which are the same on every page, we also made sure that our login and register forms are consistent. Consistency was used throughout our store, wherever a product appears on the webpage, whether it be on the store page or the shipping page or even the confirmation page, the product will have a consistent look throughout.



## 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.

## Requirements testing

|  |  |  |
| --- | --- | --- |
| # | Requirements | Pass/Fails |
| 1 | Users can login to their session | P |
| 2 | Users can sign up for an account | P |
| 3 | User’s signups are UNIQUE | F |
| 4 | User’s data stored in DB (purchase history and account details) | P |
| 5 | Basket | P |
| 6 | Picture of item shown in basket | P |
| 7 | Ability to delete item from basket | P |
| 8 | Saved data from previous purchases | F |
| 9 | Tailored page | F |
| 10 | Store page | P |
| 11 | Store filter/refine option | P |
| 12 | Customer can select size | P |
| 13 | Sizes are S M L | P |
| 14 | Confirmation of order | P |