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

1. 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.

2. 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.

3. 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.

2. Class testing:

2.1 Show Products

2.1.1 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

2.1.2 Unit Test

public > ProductTest.php

Click here to ask Blackbox to help you code faster

```php
<?php
require_once '../lib/products/general.php';

$tc1 = new General();
$tc2 = new General();

$tc1->setImageURL("/images/logo.jpg");
$tc1->setProductName("2 Piece");
$tc1->setPrice(99.99);
$tc1->setBrandName("Nike");
$tc1->setColors("Red");
$tc1->setSize("Large");

$tc2->setImageURL("/images/logo.jpg");
$tc2->setProductName("3 Piece");
$tc2->setPrice(199.99);
$tc2->setBrandName("Hugo Boss");
$tc2->setColors("Black");
$tc2->setSize("Medium");

$tc1->displaySuits();
$tc2->displaySuits();
```

Validation:
Image URL: /images/logo.jpg
Product Name: 2 Piece
Price: 99.99
Brand Name: Nike
Colors: Red
Size: Large

Validation:
Image URL: /images/logo.jpg
Product Name: 3 Piece
Price: 199.99
Brand Name: Hugo Boss
Colors: Black
Size: Medium

2.2 Orders

2.2.1 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

2.2.2 Unit Test

```
Click here to ask Blackbox to help you code faster
<?php
require_once '../lib/checkout.php';

$tc1 = new Checkout();
$tc2 = new Checkout();

$tc1->setTotalPrice(99.99);
$tc1->setDatePurchase(date("Y-m-d H:i:s"));
$tc1->setUserID(123);

$tc2->setTotalPrice(199.99);
$tc2->setDatePurchase(date("Y-m-d H:i:s"));
$tc2->setUserID(456);

$tc1->displayOrderDetails();
$tc2->displayOrderDetails();
```

Order Details:
Total Price: 99.99
Date of Purchase: 2024-04-28 15:30:13
User ID: 123

Order Details:
Total Price: 199.99
Date of Purchase: 2024-04-28 15:30:13
User ID: 456

2.3 Checkout

2.3.1 Blackbox Test

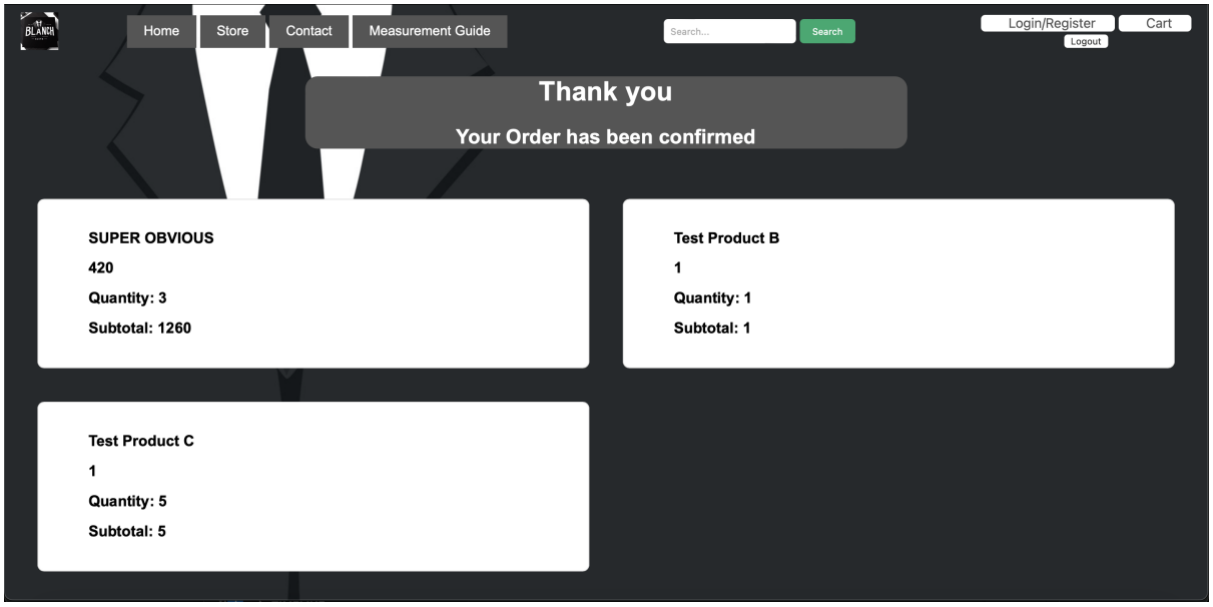
Users will be able to remove items they don't want from their cart in the checkout. This is where they confirm their order which gets sent to the DB.

Remove from cart - Removes from cart in checkout page

Confirm Purchase - Sends order to the DB

2.3.2 Unit Test

```php
require_once '../lib/checkout.php';


class TestSendOrderToDB {
    public function testSendOrder() {
        // Simulate a POST request
        $_SERVER["REQUEST_METHOD"] = "POST";

        $checkout = new Checkout();
        $order = new Order();
        $order->manageCart();

        $_SESSION['username'] = 'john_doe';
        $_SESSION['cart'] = array(
            '1' => array(
                'productName' => 'SUPER OBVIOUS',
                'price' => 420.00,
                'quantity' => 3
            ),
            '2' => array(
                'productName' => 'Test Product B',
                'price' => 1.00,
                'quantity' => 1
            ),
            '3' => array(
                'productName' => 'Test Product C',
                'price' => 1.00,
                'quantity' => 5
            )
        );

        $result = $checkout->sendOrderToDB();

        // Verify the outcome
        if ($result === true) {
            echo "Order successfully sent to the database.";
        } else {
            echo "Error sending order to the database.";
        }
    }
}

// Create an instance of the test class and run it
$test = new TestSendOrderToDB();
$test->testSendOrder();
?>

```


Home
Store
Contact
Measurement Guide
Search...
Search
Login/Register
Logout
Cart
Thank you
Your Order has been confirmed
SUPER OBVIOUS
420
Quantity: 3
Subtotal: 1260
Test Product B
1
Quantity: 1
Subtotal: 1
Test Product C
1
Quantity: 5
Subtotal: 5

2.4 Signup

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

2.4.1 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

2.4.2 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)

2.4.3 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

2.4.4 White Box Test

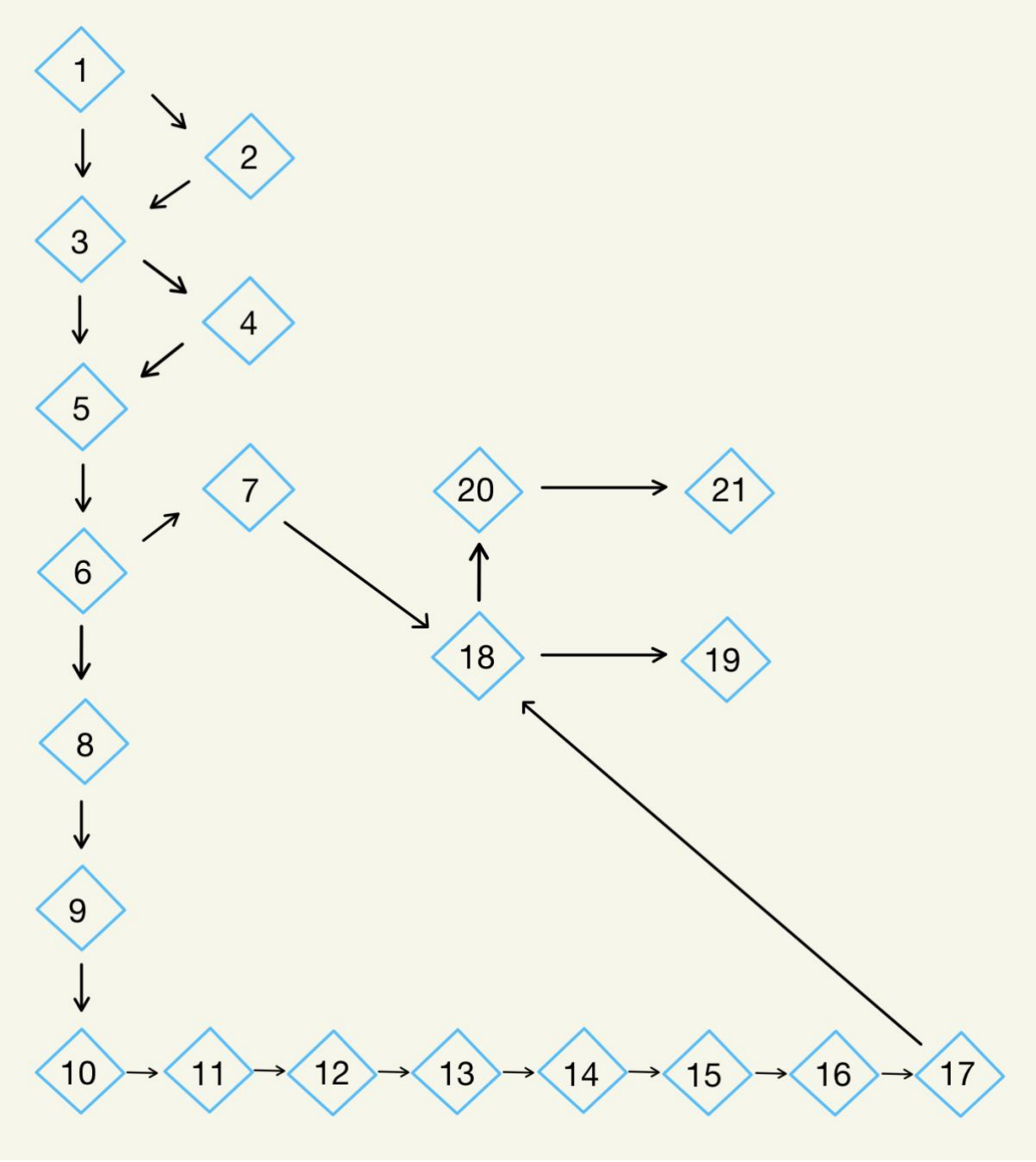
Flow graph

Function statements are numbers on the left side

```
3         function addUser($firstname, $lastname, $username, $email, $password, $address, $phone)
4           {
5               $pdo = get_connection();
6
7     --1--       if (!$this -> isValidPassword($password)) {
8     --2--           return "Password must be at least 8 characters long and contain at least one special character.";
      // Password does not meet requirements
9               }
10
11              // Check if the phone number matches the XXX-XXX-XXXX format
12    --3--       if (!$this->isValidNumber($phone)) {
13    --4--           return "Phone number must be in XXX-XXX-XXXX format";
14              }
15
16              // Remove hyphens from the phone number and count digits
17    --5--       $Digits = preg_replace('/[^0-9]/', '', $phone);
18
19              // Check if the phone number meets the length requirement
20    --6--       if (strlen($Digits) !== 10) {
21    --7--           return false; // Invalid phone number length
22              }
23
24    --8--       $encryptPass =  password_hash($password, PASSWORD_DEFAULT); //Reintech.io
      //PASSWORD_DEFAULT uses bcrypt
25
26              //insert user into database
27    --9--       $query = "INSERT INTO users (firstname, lastname, username, email, password, address, phone)
      VALUES (:firstname, :lastname, :username, :email, :password, :address, :phone)";
28    --10--      $stmt = $pdo->prepare($query);
29    --11--      $stmt->bindParam(':firstname', $firstname);
30    --12--      $stmt->bindParam(':lastname', $lastname);
31    --13--      $stmt->bindParam(':username', $username);
32    --14--      $stmt->bindParam(':email', $email);
33    --15--      $stmt->bindParam(':password', $encryptPass);
34    --16--      $stmt->bindParam(':address', $address);
35    --17--      $stmt->bindParam(':phone', $phone);
```

```
36
37    --18--        if ($stmt->execute()) {
38    --19--            return true; //user added successfully
39    --20--        } else {
40    --21--            return false; //error adding user
41                }
42            }
```

Path Flow



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)

Path 1

```
53    // Test valid user addition
54    $password = "Password1@";
55    $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
56    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Password1@", "123 Main St", "123-456-7890");
57    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Password1@, ", "address: 123 Main St, ", "number: 123-456-7890}";
58    echo "<p>Valid User Addition: " . ($result ? 'True' : 'False') . "</p>";
```

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Password1@, address: 123 Main St, number: 123-456-7890}

Valid User Addition: True

Path 2

```
76    // Test valid user addition with invalid phone number (invalid format)
77    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Pass1@", "123 Main St", "123-4567890");
78    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123-4567890}";
79    echo "<p>Invalid User Addition (Invalid Phone Number Format): " . ($result === "Phone number must be in XXX-XXX-XXXX format" ? 'True' : 'False') . "</p>";
80
```

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123-4567890}

Invalid User Addition (Invalid Phone Number Format): True

Path 3

```
60    // Test invalid password
61    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Pass1@", "123 Main St", "123-456-7890");
62    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123-456-7890}";
63    echo "<p>Invalid Password: " . ($result === "Password must be at least 8 characters long and contain at least one special character." ? 'True' : 'False') . "</p>";
64    echo "<br>";
```

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123-456-7890}

Invalid Password: True

Path 4

```
70
71      // Test valid user addition with invalid phone number (less than 10 digits)
72      $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Password1@", "123 Main St", "1234567890");
73      echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123456789}";
74      echo "<p>Invalid User Addition (Invalid Phone Number - Length): " . ($result === "Phone number must be in XXX-XXX-XXXX format" ? 'True' : 'False') . "</p>";
75
```

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123456789}

Invalid User Addition (Invalid Phone Number - Length): True

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

Unit Testing

Password test

```php
<?php
require_once '../lib/user/user.php';

function testIsValidPassword()
{
    $user = new User(); // Instantiate your class

    // Test valid password
    $password = "Password1@";
    $isValid = $user->isValidPassword($password);
    echo "<p>Password: $password - IsValid: " . ($isValid ? 'True' : 'False') . "</p>";

    // Test invalid password (less than 8 characters)
    $password = "Pass1@";
    $isValid = $user->isValidPassword($password);
    echo "<p>Password: $password - IsValid: " . ($isValid ? 'True' : 'False') . "</p>";

    // Test invalid password (no special characters)
    $password = "Password1";
    $isValid = $user->isValidPassword($password);
    echo "<p>Password: $password - IsValid: " . ($isValid ? 'True' : 'False') . "</p>";
    echo "<br>";


}
```

Password: Password1@ - IsValid: True

Password: Pass1@ - IsValid: False

Password: Password1 - IsValid: False

Phone number Test

```php
function testIsValidNumber(){
    $user = new User();

    $phone = "123-456-7890";
    $phoneValid = $user->isValidNumber($phone);
    echo "<p>Phone: $phone - IsValid: " . ($phoneValid ? 'True' : 'False') . "</p>";

    $phone = "123456789";
    $phoneValid = $user->isValidNumber($phone);
    echo "<p>Phone: $phone - IsValid: " . ($phoneValid ? 'True' : 'False') . "</p>";

    $phone = "123 456 7890";
    $phoneValid = $user->isValidNumber($phone);
    echo "<p>Phone: $phone - IsValid: " . ($phoneValid ? 'True' : 'False') . "</p>";

    $phone = "123-4567890";
    $phoneValid = $user->isValidNumber($phone);
    echo "<p>Phone: $phone - IsValid: " . ($phoneValid ? 'True' : 'False') . "</p>";
    echo "<br>";

}
```

Phone: 123-456-7890 - IsValid: True

Phone: 123456789 - IsValid: False

Phone: 123 456 7890 - IsValid: False

Phone: 123-4567890 - IsValid: False

Add User (Password)

```php
function testAddUser()
{
    $user = new User(); // Instantiate your class

    // Test valid user addition
    $password = "Password1@";
    $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Password1@", "123 Main St", "123-456-7890");
    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Password1@, ", "address: 123 Main St, ", "number: 123-456-7890}";
    echo "<p>Valid User Addition: " . ($result ? 'True' : 'False') . "</p>";

    // Test invalid password
    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Pass1@", "123 Main St", "123-456-7890");
    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123-456-7890}";
    echo "<p>Invalid Password: " . ($result === "Password must be at least 8 characters long and contain at least one special character." ? 'True' : 'False') . "</p>";
    echo "<br>";
```

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Password1@, address: 123 Main St, number: 123-456-7890}

Valid User Addition: True

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123-456-7890}

Invalid Password: True

Add User (Phone Number)

```php
    // Test valid user addition with valid phone number
    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Password1@", "123 Main St", "123-456-7890");
    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123-456-7890}";
    echo "<p>Valid User Addition (Valid Phone Number): " . ($result ? 'True' : 'False') . "</p>";

    // Test valid user addition with invalid phone number (less than 10 digits)
    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Password1@", "123 Main St", "1234567890");
    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123456789}";
    echo "<p>Invalid User Addition (Invalid Phone Number - Length): " . ($result === "Phone number must be in XXX-XXX-XXXX format" ? 'True' : 'False') . "</p>";

    // Test valid user addition with invalid phone number (invalid format)
    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Password1@", "123 Main St", "123-4567890");
    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123-4567890}";
    echo "<p>Invalid User Addition (Invalid Phone Number Format): " . ($result === "Phone number must be in XXX-XXX-XXXX format" ? 'True' : 'False') . "</p>";

    // Test valid user addition with invalid phone number (invalid format (spaces))
    $result = $user->addUser("John", "Doe", "johndoe", "john@example.com", "Password1@", "123 Main St", "123 456 7890");
    echo "{Fname: John, ", "Lname: Doe, ", "username: johndoe, ", "email: john@example.com, ", "password: Pass1@, ", "address: 123 Main St, ", "number: 123 456 7890}";
    echo "<p>Invalid User Addition (Invalid Phone Number Format (spaces): " . ($result === "Phone number must be in XXX-XXX-XXXX format" ? 'True' : 'False') . "</p>";
}
```

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123-456-7890}

Valid User Addition (Valid Phone Number): True

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123456789}

Invalid User Addition (Invalid Phone Number - Length): True

{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123-4567890}

Invalid User Addition (Invalid Phone Number Format): True

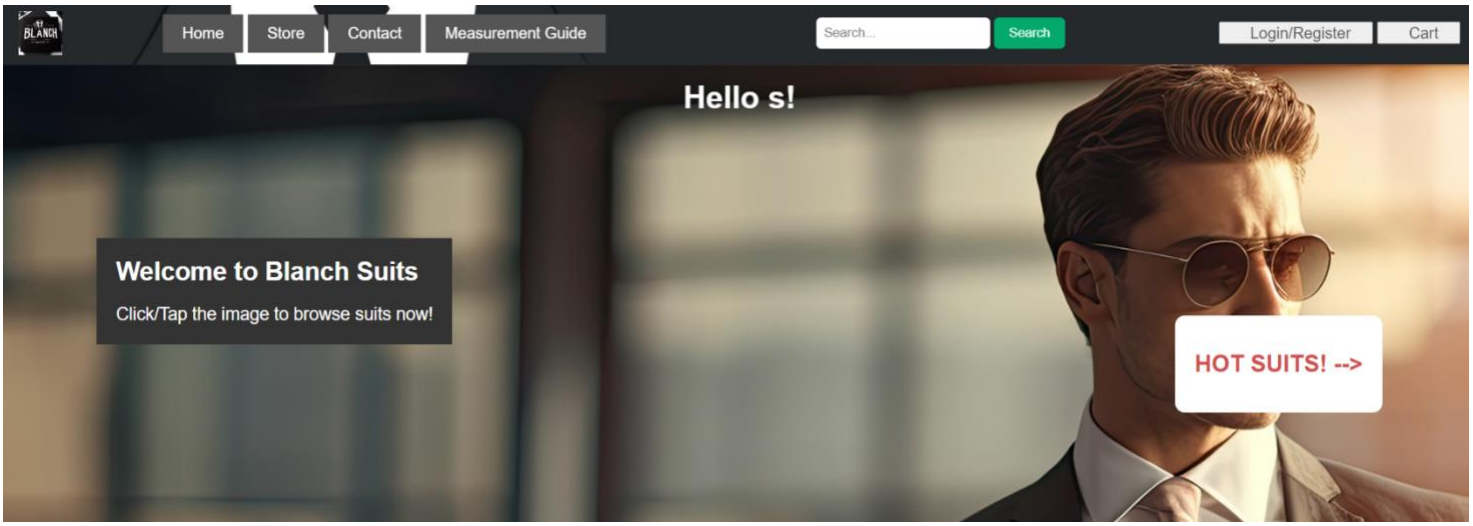
{Fname: John, Lname: Doe, username: johndoe, email: john@example.com, password: Pass1@, address: 123 Main St, number: 123 456 7890}

Invalid User Addition (Invalid Phone Number Format (spaces): True

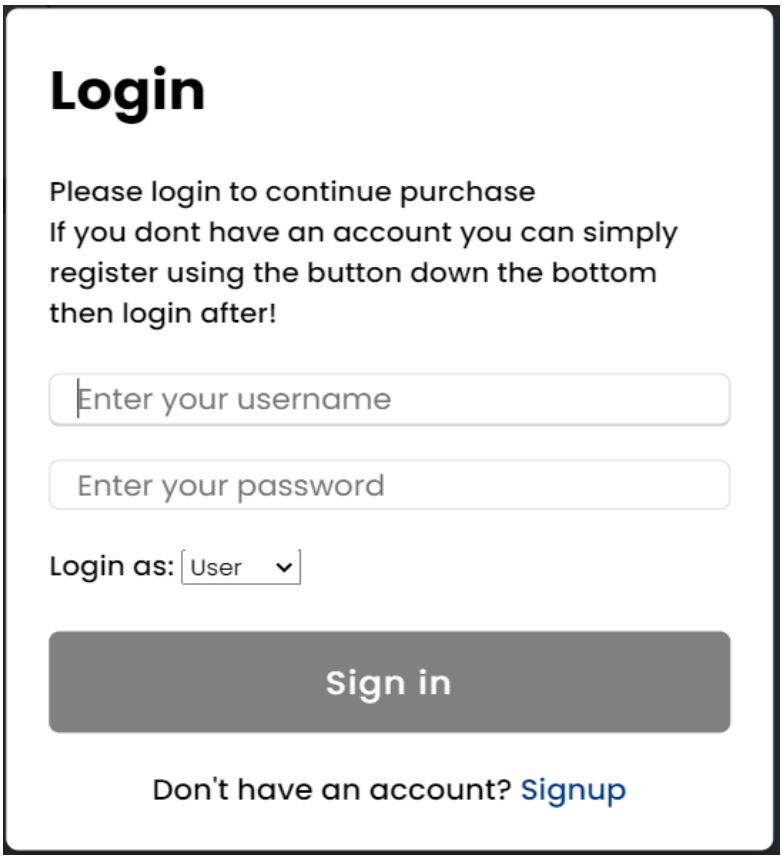
3. User Interface testing:

3.1. 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.

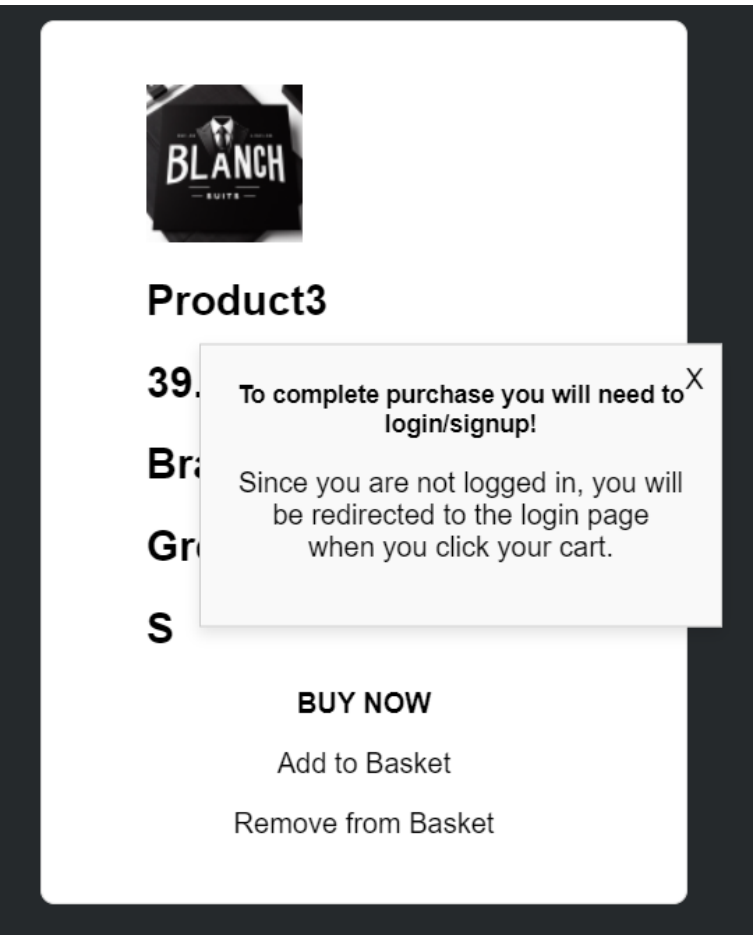


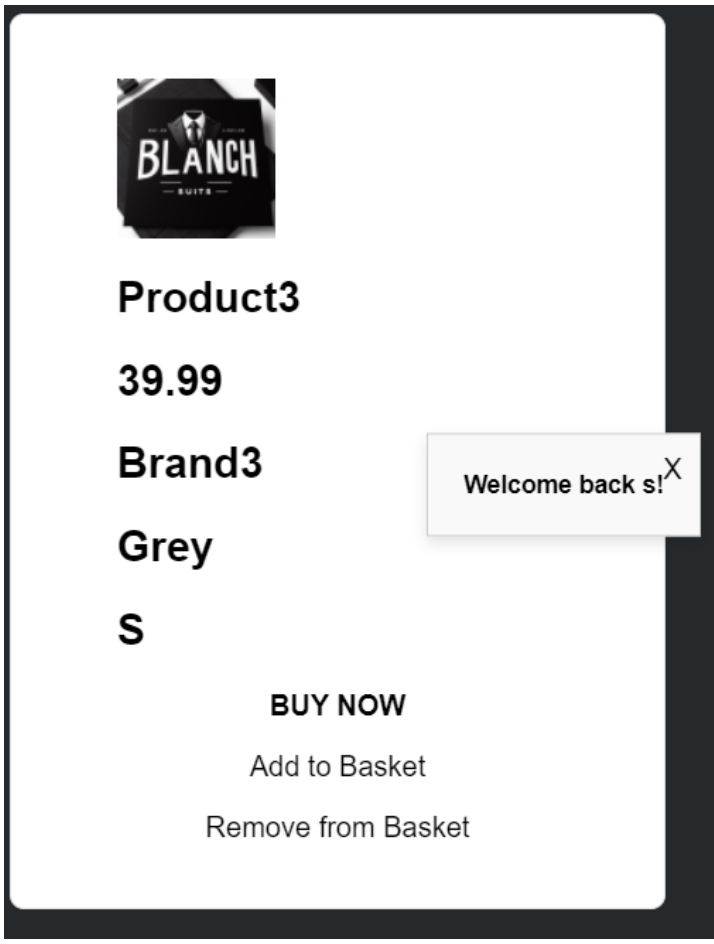
On our login page it tells the customer exactly what to do to sign in or to sign up as shown in the image below



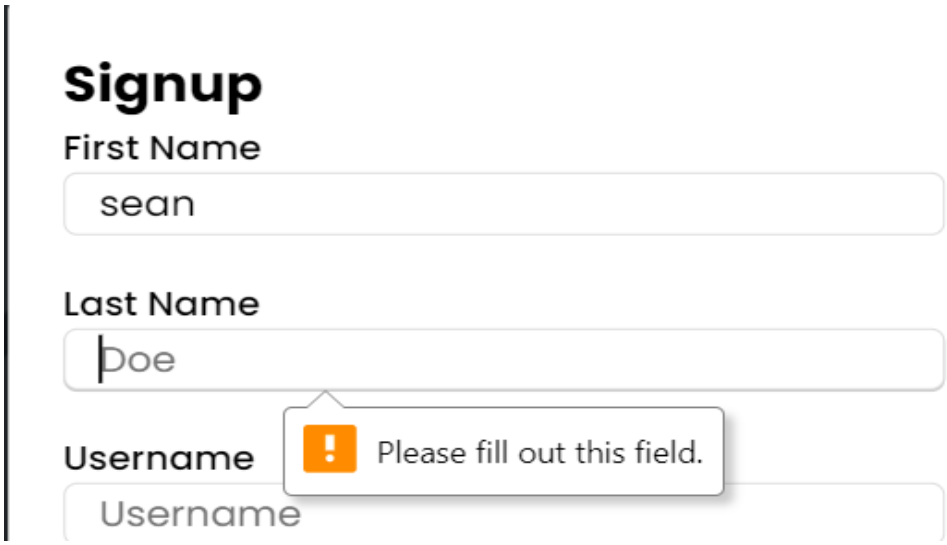
3.2. Feedback

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)"





When someone is registering and inserts just the name and presses register it says

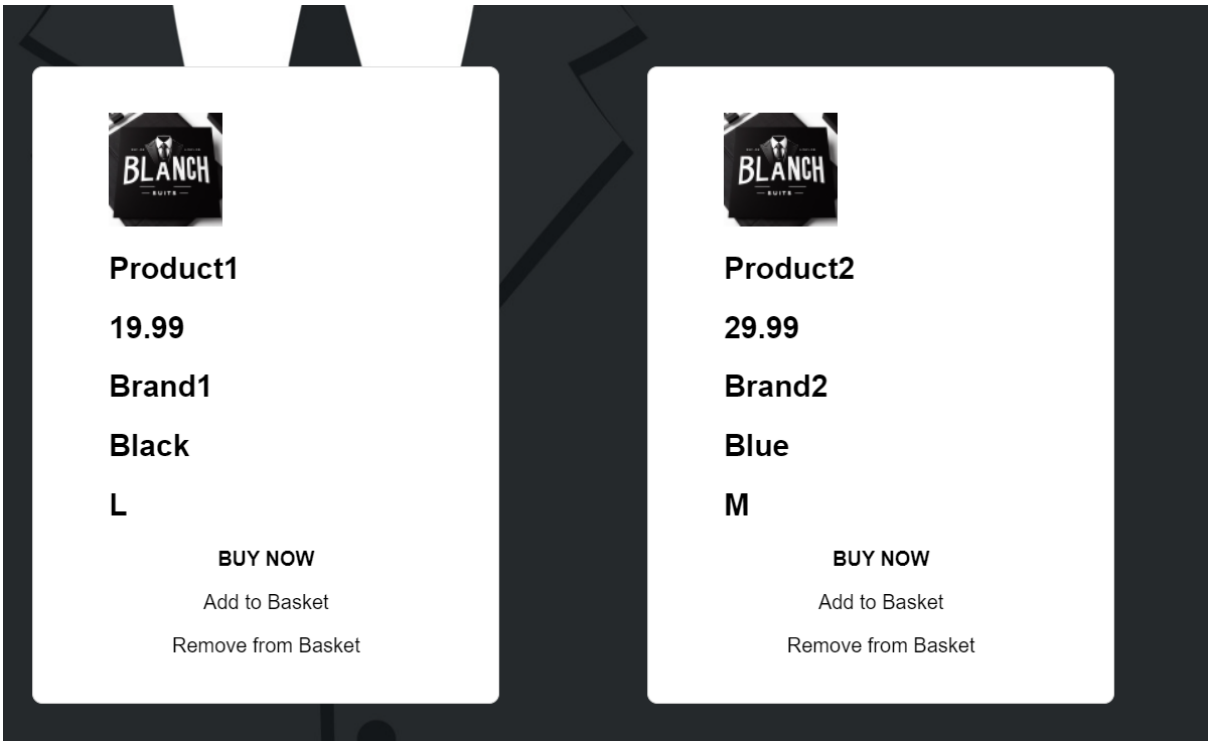


We give feedback on nearly all steps of the login and register.

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

3.3. 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.



3.4. 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.

4 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