CISC 365 A1: Front End Requirements

Question 1: List of test cases and what are they intended to test

1. **User Registration**
    a. Test objective: To verify that users can successfully register for an account
2. **User Login**
    a. Test objective: To verify that registered users can successfully log in
3. **Browse Menus**
    a. Test objective: To verify that users can view menus from local restaurants
4. **Place Orders**
    a. Test objective: To verify that users can successfully place an order
5. **Make Payments**
    a. Test objective: To verify that users can successfully make payments for their orders
6. **Delivery Process**
    a. Test objective: To verify that users receive their food orders

Question 2: Actual input and output for tests (represented as tables):

Test Case Name: User Registration

Objective: To verify that users can successfully register for an account

Arrange: Open the registration page

Act: Fill in valid user details (e.g., username, email, password) and click the "Register" button.

Assert: Check that the user is registered and redirected to the login page with a success message.

Preconditions

- The web application loads successfully
- User is on the registration page

Postconditions

- The user should be registered successfully and redirected to the login page with a success message
- Duplicate username and email should be appropriately handled with error messages.
- Invalid email formats, weak passwords, and missing password requirements should be validated and communicated to the user.

Requirements

- Username Requirements: Username must not be taken
- Email Requirements:
    - Email entered will have to be valid
    - Email must not already be in use for another user
- Password requirements:
    - Must contain at least one uppercase letter
    - Must contain at least one lowercase letter
    - Must contain at least one number

Input	Output (Expected Output)	Purpose

{
username: “testuser”
email: “testemail@example.com”
password: “Password123”
}
	{
	message: “User registered successfully”
	}
		Successful registration

{
username: “testuser1”
email: “testemail1@example.com”
password: “Password123”
}
	{
	message: “Username has been taken”
	}
		Duplicate username

{
username:””
email: “testemail1@example.com”
password: “Password123”
}
	{
	message: “Missing username”
	}
		Empty username field

{
username: “testuser2”
email: “testemail1@example.com”
password: “Password123”
}
	{
	message: “Email has been taken”
	}
		Duplicate email

{
username: “testuser3”
email: “testemailexample.com”
password: “Password123”
}
	{
	message: “Invalid email entered”
	}
		Invalid email format

{
username: “testuser3”
email: “”
password: “Password123”
}
	{
	message: “Missing email”
	}
		Empty email field

{	{	Missing password

username: “testuser4” email: “testemail4@example.com” password: “password123” }	message: “Password must contain at least one uppercase” }	requirements

{
username: “testuser5”
email: “testemail5@xample.com”
password: “PASSWORD123”
}
	{
	message: “Password must contain at least one lowercase”
	}
		Missing password requirements

{
username: “testuser6”
email: “testemail6@example.com”
password: “Password”
}
	{
	message: “Password must contain at least one number”
	}
		Missing password requirements

{
username: “testuser6”
email: “testemail6@example.com”
password: “”
}
	{
	message: “Missing password”
	}
		Empty password field

Test Case Name: User Login

Objective: To verify that registered users can successfully log in

Arrange: Open the login page

Act: Fill in valid user details (e.g., username, email, password) and click the "Login" button.

Assert: Check that the user details are correct and redirect to the homepage with a success message

Preconditions

- The web application loads successfully
- User has to have an existing account registered in the system

Postconditions

- The user should be logged in successfully and redirected to the homepage with a success message
- Incorrect usernames and passwords should be appropriately handled with error messages

Requirements

- Username Requirements: username must be a valid username which an account has been created under
- Password Requirements: password must be the correct password associated with its entered username

Input	Output (Expected Output)	Purpose

{
username: “testuser”
password: “Password123”
}
	{
	message: “Successful login”
	}
		Login with valid user credentials

{
username: “fakeuser”
password: “Password123”
}
	{
	message: “Username does not exist”
	}
		Login with non-existent username

{
username: “testuser”
password: “password321”
}
	{
	message: “Incorrect password”
	}
		Login with incorrect password

Test Case: Browse Menus

Objective: To verify that users can view menus from local restaurants

Arrange: Navigate to the platform’s main page.

Act: Click on a restaurant to view its menu

Assert: Check that you have successfully entered the restaurant page (console message will be displayed)

Preconditions

- The user is logged in
- The user is on the platforms homepage displaying all available restaurants

Postconditions

- The user should be on the selected restaurants page
- A success message confirming entry into the restaurant page should be visible in the console.

Input	Output (Expected Output)	Purpose

{
route: “Pizza Pizza”
}
	{
	message: “You have successfully entered Pizza Pizza Page”
	}
		Successful routing

Test Case: Place Orders

Objective: To verify that users can successfully place an order

Arrange: Navigate to the order placement page
Act: Select items from from the restaurant's menu (adding them to cart)
Assert: On checkout, selected menu items are successfully checked out

Preconditions

- The user is logged in
- The user is on the selected restaurants page where they can select items for their order

Postconditions

- The user should receive a confirmation message indicating that the order was placed successfully
- The order details should be recorded in the system

Input	Output (Expected Output)	Purpose

{
userSelections:
{
Pepperoni Pizza:
{
price: 10.00,
quantity: 2
}
}
}
	{
	message: "Order placed successfully"
	}
		Successful checkout

{
userSelections:
{ }
}
	{
	message: "No item has been selected"
	}
		Empty checkout

Test Case: Make Payments

Objective: To verify that users can successfully make payments for their orders

Arrange: Proceed to the checkout page with items in the cart

Act: Select a payment method and enter the necessary details

Assert: Check that the payment is processed (using Stripe API) and payment confirmation is displayed

Preconditions

- The user is logged in
- The user has items in the cart and is on the "Make Payment" page

Postconditions

- The user should receive a confirmation message indicating that the payment was made successfully
- The order details should be recorded be recorded in the system

Requirements

- Payment Requirements**:** If the payment method is “Credit Card” the card number, expiry date, and cvv must all be valid. User must be logged in.

Input	Output (Expected Output)	Purpose

{
paymentMethod: “Credit Card”
cardNumber: 1234567890123456
expiryDate: “12/24”
CVC: 123
}
	{
	message: “Payment information valid”
	}
		Valid payment details

{
paymentMethod : “Credit Card”
cardNumber : 1234567890123450
expiryDate: “12/24”
CVC: 123
}
	{
	message : “Your card number is invalid”
	}
		Invalid card number

{
paymentMethod: “Credit Card”
cardNumber: 1234567890123450
expiryDate: “12/22”
CVC: 123
}
	{
	message: “Your card’s expiration date is invalid”
	}
		Invalid card expiration date

{
paymentMethod: “Credit Card”
cardNumber: 1234567890123450
expiryDate: “12/24”
CVC: 123
}
	{
	message: “Your card’s security code is invalid”
	}
		Invalid card security code

{
paymentMethod: “Credit Card”
cardNumber: null
expiryDate: “12/24”
CVC: 123
}
	{
	message: “Card number has not been entered”
	}
		Missing card details

{
paymentMethod: “Credit Card”
cardNumber: 1234567890123456
expiryDate: “”
	{
	message: “Expiry date has not been entered”
	}
		Missing card details

CVC: 123 }		

{
paymentMethod: “Credit Card”
cardNumber: 1234567890123456
expiryDate: “12/24”
CVC: null
}
	{
	message: “CVC has not been entered”
	}
		Missing card details

Test Case: Delivery Process

Objective: To verify that users receive their food orders

Arrange: Open the tracking delivery page

Act: User waits patiently and is able to track the estimated time required for their food to be delivered successfully

Assert: Check that the estimated time is updated accurately and that the appropriate status message is displayed

Preconditions

- The user is logged in
- The user has made a successful payment for the order
- The user is on the “Track Delivery” page

Postconditions

- When the estimated delivery time reaches 0, the food item(s) should be marked as delivered successfully
- The user should receive a message indicating that the food has been delivered

Input	Output (Expected Output)	

{
deliveryTime: 10
}
	{
	message: “Delivery on its way!”
	}
		Ongoing delivery

{
deliveryTime: 0
}
	{
	message: “Food delivered!”
	}
		Successful delivery