CSwap Test Plan Document

Apr 30, 2022

# Use-Case Testing:

Title: Sign Up

Actors: Unauthorized User

Requirements: The Unauthorized User has an email address, Google, or Facebook account, and is not already registered with CSwap

Main Scenario:

1. The system displays the login page
2. The unauthorized user clicks the “sign up” button
3. The user is directed to a sign-up page
4. The unauthorized user enters their email and password and clicks “sign up”
5. The system accordingly displays “Sign up successful” and send the user back to the sign in page

Alternatives:

1. The system displays the login page
2. The unauthorized user clicks the “sign up” button
3. The user is directed to a sign-up page

4a) The unauthorized user enters their email and password and clicks “sign up”

4b) The unauthorized user clicks “log in” under “already have an account?”

5a) The system accordingly displays “Sign up successful” and send the user back to the sign in page

5b) The system displays “Error email already in user”

5c) The system displays “Passwords should be at least 16 characters”

Test Situation:

1. User enters their email and password into the correct fields
2. User enters an invalid email into the email field
3. User enters a weak password into the password field
4. The User clicks “log in” on the signup page

Test coverage:

Base: 5

Test situations cover all 6 cases

100% test coverage

Title: Sign in with Google

Actors: Unauthorized User

Requirements: The user already created an account with google and the login page is displayed

Main Scenario:

1. The unauthorized user clicks the ReCAPTCHA button
2. The ReCAPTCHA button is verified
3. The Unauthorized user clicks the “login with google” button
4. The system displays the google sign in popup
5. The unauthorized enters their google account credentials
6. The system directs the user to the home page

Alternatives:

1. The unauthorized user clicks the reCAPTCHA button

2a) The reCAPTCHA is verified

2b) The reCAPTCHA is failed and must be attempted again

1. The unauthorized user clicks the “login with google” button
2. The system displays the sign in with google popup

4a) The unauthorized user enters their google account information

4b) The unauthorized user clicks on the google account of their choosing

4c) The unauthorized user closes the popup

5a) The system directs the user to the home page

5b) The system directs the user to the login page

Test Situation:

1. User completes the reCAPTCHA
2. The user fails the reCAPTCHA
3. The user clicks on the “login with google” button
4. The user enters their google account information
5. The user chooses a google account that is already signed in
6. The user closes the popup

Test coverage:

Base: 6

Test situation covers all 6 cases

100% test coverage

Title: Create Listing

Actor: Authorized User

Requirements: Authorized user has logged in

Main Scenario:

1. The user clicks on “Create Listing”
2. The system displays a list item form
3. The user enters price and a description of the item
4. The system adds the price and description to the listing
5. The user adds related tags based on the item and listing category
6. The system adds those tags to the listing
7. The user adds photos to the listing
8. The system adds those pictures to the listing
9. The user submits the form
10. The system publishes the item to the site

Alternatives:

1. The user clicks on the “create listing” button
2. The system displays a listing item form

3a) The user enters the price and a description of the item

3b) The user doesn’t enter a price or description

3c) The user closes the form

4a) The system adds the price and description to the listing

4b) The system prompts the user to enter a price and description

4c) The system redirects the user to the previous page

5a) The user adds related tags based on the item and the listing category

5b) The user doesn’t add any tags to the listing

6a) The system adds tags to listing

6b) The system doesn’t add any tags to listing

7a) The user uploads photos to the listing

7b) The user doesn’t upload any photos to the listing

8a) The system adds phots to the listing

8b) The system doesn’t add photos to the listing

9a) The user submits the form

9b) The user cancels the form

10a) The system publishes the form to the website

10b) The system redirects the user to the home page

Test Situation:

1. The user clicks on “Create listing”
2. The user enters a price and description
3. The user doesn’t enter a price and/or description
4. The user closes the form
5. The user adds tags related to the item’s category
6. The user doesn’t enter any tags
7. The user uploads a photo to the listing
8. The user doesn’t add any photos
9. The user submits the form
10. The user cancels the form

Test coverage:

Base 10

Test situations cover all 10 cases

100% test coverage

Title: Remove own account

Actor: Authorized user

Requirements: the user is logged in and is in the profile section with the account displayed

Main Scenario:

1. The user clicks on the “remove account” button
2. The system displays a popup with “are you sure you want to remove this account”
3. The authorized user clicks “yes”
4. The system removes the users account and redirects the user to the login page

Alternatives:

1. The user clicks on the “remove account” button
2. The system displays on a popup, “are you sure you want to remove this account”

3a) The authorized user clicks “Yes”

3b) The authorized user clicks “No”

4a) The system removes the authorized user’s account and sends the user to the login page

4b) The system returns the user to the profile page

Testing situation:

1. The user clicks on the “remove account button”
2. The user clicks on the yes button
3. The user clicks on the no button

Test coverage:

Base 4

Tests cover all 4 cases

100% test converge

Title: Login

Actor: Unauthorized User

Requirements: The unauthorized user has previously created an account is the login page is displayed

Main Scenario:

1. The unauthorized user clicks the reCAPTCHA button
2. The reCAPTCHA is verified
3. The unauthorized user enters their credentials
4. The system checks that email and password could be valid
5. The unauthorized user clicks login
6. The system redirects the user to the home page

Alternatives:

1. The unauthorized user clicks the reCAPTCHA button

2a) The reCAPTCHA is verified

2b) The reCAPTCHA fails

3a) The unauthorized user enters correct credentials

3b) The unauthorized user enters invalid credentials

4a) The system verifies the email and password could be valid

4b) The system verifies the email and password are invalid

1. The unauthorized user clicks login

6a) The system redirects the user to the home page

6b) The system displays “Invalid Credentials”

Test situation:

1. The unauthorized user clicks the reCAPTCHA button
2. The unauthorized user fails the reCAPTCHA
3. The user enters the correct email and password
4. The user enters invalid email or invalid password
5. The user enters a wrong email or password
6. The user clicks login

Test Coverage:

Base: 6

The tests cover all 6 cases

100% test coverage

# Graphical user interface, text, application, chat or text message Description automatically generatedUnit-Testing:

Text

Description automatically generatedOur unit testing consisted of a total of ten different tests across our application. The first area to be tested was the home page to make sure that the page was rendering correctly. That test is visible to the right in *figure 1.* The purpose of this first unit test was to check to see if the home page was rendering correctly. This test first renders the home element and assigns constants from the rendered page. The Home page is supposed to render the text “Textbooks”, “Apartments”, “Electronics”, “Furniture”, and “Appliances” so the test assigns constants to that text. The test than checks to see if those are rendered. This test fails due to undefined variables on the page which only becomes defined once a user logs into the application. One of these variables is logout which is tested to be undefined, and that test passes. Currently, there hasn’t been a workaround found to define the required variables for the test. One idea is to manually define a user, but since the authentication which defines the variable is found in a separate file, that idea may not work. The next four tests involve testing the Login page. These tests are visible in *figure 2* on the left. The first three tests in the image are meant to test the three different login functionalities. The first test is written to test the handleSubmit function which is called when a user logs in with their email and password. The test is mean to render the page and the submit function and then submit the form. The test is then to expect that the method to have been called. The next two tests for sign-in-with-Google and sign-in-with-Facebook are written similarly. Due to the previously described issue with the undefined variables, those three tests fail. The fourth test checks if the variable is undefined, which that test passes. *Figure 4* has two tests. The first test is like the one in *figure 1* which is meant to test whether the page is rendering correctly. The test renders the element, assigns constants to text rendered on the screen, and expects that text to be on screen. Since the page that is being tested also has variables that are undefined until a user logs in to the application, the first test fails. The second test checks one of the undefined variables. The last three tests are to test the methods involving the Textbooks page. These three tests can be seen in *figure 3* below. The first test is meant to test the method that is responsible for getting textbooks from the database and expecting the constant products to be of length greater than zero. The problem with that test is due to the page also having undefined variables, it’s unclear whether the correct method is being tested or tested correctly. The second test is Text

Description automatically generatedmeant to check that the page is rendering correctly, and the third test is there to check one of the undefined variables. The next thing to discuss are the errors that are visible in *figure 5* and *figure 6.* As mentioned previously, most of the unit testing failed due to the presence of undefined variables. These variables are found on almost every page and are not defined until a user logs in to the application. As previously mentioned, a workaround hasn’t been discovered, but one idea is to manually define a user which would define the required Text

Description automatically generatedvariables. The image in *figure 5* demonstrates the main error that is thrown by most of the tests. The logout and user constants are both undefined and would remain undefined until a user logs into the application. Logout and user become defined by the userAuthContext file which is active once a user logs in or signs up. *Figure 6* demonstrates another error from a different undefined variable, the signup variable. This variable also becomes defined through the same process and same file.

Figure : Login page testing

Figure : Home page testing

Figure : Signup page testing

Figure : Textbook page testing

# Graphical user interface, text Description automatically generated

Figure : Textbook testing error

Graphical user interface, text

Description automatically generated

Figure : Signup page testing error