Christian Andrew Ortiz

Unit Testing

Student ID: 1593352
University of California, Santa Cruz

CMPS115 - Software Engineering Project I

Contents

SIGNIN.JS

SignIn.js has been tested in an un-automated environment several ways. The container holds several interactable fields.

- Two text fields for the user to enter their username and password (username, password)
- One Button
  - Submit

The two text fields are able to take in a string and then attempt to authenticate the username and password in the amazon database using cognito once the submit button is pressed. An empty string is invalid for either username or password. The submit button will be disabled until a non-empty username and password is inputted into the fields. If the user is authenticated they will be routed to **Artist Dashboard** if they are an authenticated artist or they will be routed to **Business Dashboard** if they are an authenticated business.

For the following un-automated tests, input will be referenced in the order presented in the itemized section above. (username, password)

Test 1:

Input: @Chris, @Chris123!
Expected Output: User had been routed to the artist dashboard

Upon entering the data in these fields, and hitting the **SUBMIT** button, the page redirected me to the artist dashboard as expected.

Actual Output: User had been routed to the artist dashboard

Test 1 was successful.

Test 2:

Input: @Chris, @abcd
Expected Output: An alert pops up stating "Incorrect username or password." and the user

stays in the sign in page

Upon entering the data in these fields, and hitting the **SUBMIT** button, an alert pops up stating that "Incorrect username or password." and the user stays in the sign in page

Actual Output: An alert pops up stating "Incorrect username or password." and the user stays in the sign in page

Test 2 was successful.

Test 3:

The parameter *null* represents a null or empty string
Input: @Chris, *null*
Expected Output: The user is not able to click the submit button and they are not able to sign in.

Upon entering the data in these fields, the **SUBMIT** button is disabled and the user is not able to sign in

Actual Output: The user is not able to click the submit button and they are not able to sign in.

Test 3 was successful.

Test 4:

The parameter *null* represents a null or empty string
Input: *null*, abcd
Expected Output: The user is not able to click the submit button and they are not able to sign in.

Upon entering the data in these fields, the **SUBMIT** button is disabled and the user is not able to sign in

Actual Output: The user is not able to click the submit button and they are not able to sign in.

Test 4 was successful.

Test 5:

Input: bChris, Chris123!
Expected Output: The user has been authenticated and has been routed to the business dashboard.

Upon entering the data in these fields, and hitting the **SUBMIT** button, the page will redirect the user to their business dashboard.

Actual Output: The user has been authenticated and has been routed to the business dashboard.

Test 5 was successful.

Test 6:

Input: chris111, Chris123!
Expected Output: An alert is popped up stating "User does not exist" and stays in the sign up page.

Upon entering the data in these fields, and hitting the **SUBMIT** button, an alert pops up stating, "User does not exist", clicking okay on the alert keep the user in the sign up page.

Actual Output: An alert is popped up stating "User does not exist" and stays in the sign up page.

Test 6 was successful.

ArtistSignUp.js

ArtistSignUp.js has been tested in an un-automated environment several ways. The container holds several interactable fields.

- Four text fields to create an artist account (username, email, password, password confirmation)
- One text field that takes in the confirmation code to confirm the artists account in the AWS server.
- Two Buttons
  - Sign Up
  - Confirmation

The four text fields are able to take in strings that will be used to create a new user on the AWS server. If a new user is created then the page will go to the confirmation code section. An email will be sent to the email that was entered in the email field containing a confirmation code. Entering the code into the text field will confirm the users information into the AWS server. If it was successful then the page will route to the artist home page. Usernames must be unique and contain no spaces, emails must follow be valid email format, passwords must be at least six characters long and include at least one uppercase character and at least one special character.

For the following un-automated tests, input will be referenced in the order presented in the itemized section above. (user name, email, password, password confirmation, confirmation code)

Test 1:

In this case it is assumed that the username and email have not been used to create an account. **Input:** "newChris", "chanorti@ucsc.edu", "Chris123!", "Chris123!", '123456'
Expected Output: The user has created a new artist and has been routed to the artist home page

Upon entering the data in these fields, and hitting the **Sign Up** button, the page loaded a confirmation field and a confirmation code has been sent to the specified email.

Actual Output: A new artist was created and the paged was routed to the artist home page.

Test 1 was successful.

Test 2:

In this case it is assumed that the username and email have not been used to create an account. **Input:** "newChris", "chanorti", "Chris123!", "Chris123!", ’123456’
Expected Output: An windows pops up stating "Please include an ’@’in the email address. ’newChris’ is missing an ’@’."

Upon entering the data in these fields, and hitting the **Sign Up** button, a window pops up stating that the email was not valid. The four fields stay the same.

Actual Output: An windows pops up stating "Please include an ’@’in the email address. ’newChris’ is missing an ’@’."

Test 2 was successful.
Test 1 was successful.

Test 3:

In this case it is assumed that the username ’chris’ has be used already **Input:** "chris", "chanorti@ucsc.edu", "Chris123!", "Chris123!", ’123456’
Expected Output: An windows pops up stating "Please include an ’@’in the email address. ’newChris’ is missing an ’@’."

Upon entering the data in these fields, and hitting the **Sign Up** button, a window pops up stating that the email was not valid. The four fields stay the same.

Actual Output: An windows pops up stating "Please include an ’@’in the email address. ’newChris’ is missing an ’@’."

Test 3 was successful.

REPLYSUBMISSION.JS

ReplySubmission.js has been tested in an un-automated environment several ways. The container holds several interactable fields.

- One text field that takes in a string that must be 50 characters long. One art piece
- Two Radio Buttons
  - Will submit art piece
  - Will not submit art piece

The text field is able to take in a string that must be 50 characters long to be used as the reply to the selected art piece. The art piece was chosen in the submissions page as the art piece the business wants to reply to. The two radio buttons serve as the information for business to choose whether or to they want to submit the art to their business page. Pressing the submit button sends the reply back to the user that asked for the art to be submitted.

For the following un-automated tests, input will be referenced in the order presented in the itemized section above. (art piece, reply, radio button)

Test 1:

Let *reply* be a string with than 51 characters. **Input:** *artPiece*, *reply*, "Will submit art piece"
Expected Output: The user has made a reply to the artwork and the page has been routed to the business home page.

Upon entering *reply* in the text fields and selecting the "Will submit art piece and clicking the **Submit** button, the page will send the reply, *artPiece* and the value of the radio button to the AWS server. The page will then be routed to the "Home" page.

Actual Output: The user has made a reply to the artwork and the page has been routed to the business home page.

Test 2 was successful.

Test 1:

Let *reply* be a string with than 50 characters. **Input:** *artPiece*, *reply*, "Will submit art piece"
Expected Output: The user has made a reply to the artwork and the page has been routed to the business home page.

Upon entering *reply* in the text fields and selecting the "Will submit art piece and clicking the **Submit** button, the page will send the reply, *artPiece* and the value of the radio button to the AWS server. The page will then be routed to the "Home" page.

Actual Output: The user has made a reply to the artwork and the page has been routed to the business home page.

Test 2 was successful.

Test 3:

Let *reply* be a string with than 49 characters. **Input:** *artPiece*, *reply*, "Will not submit art piece"
Expected Output: The user has not made a valid reply to the artwork and the page has remains on the same page.

Upon entering *reply* in the text fields and selecting the "Will not submit art piece" attempting to click the **Submit** button will do nothing. The **Submit** button will remain disabled until the *reply* is at least 50 characters long. The page will remain until a new action has been performed.

Actual Output: The user has not made a valid reply to the artwork and the page has remains on the same page.

Test 3 was successful.