Design Inspection

|  |  |
| --- | --- |
| Product | Mixbook |
| Date | 02/10/17 |
| Author | Alex Coll, Nick Zenobi, Tyler Preston, Wyatt Dahlenburg |

|  |  |  |  |
| --- | --- | --- | --- |
| Defect # | Description | Severity | How it was Corrected |
| 1. | There is no way to add a review for a recipe | 2 | It was corrected by adding a link to the page on the recipe page so that users can seamlessly traverse to that page when viewing a recipe. |
| 2. | User should be the legal drinking age to use the app even if they do not have an account | 2 | When continuing without an account the user will have to input his/her age and it must be over 21 to continue using the application. |
| 3. | Even when a user has an account his/her recipes will only be saved locally instead of on the server | 2 | Have the application interact with the web server to save the user’s recipes to the database |
| 4. | When adding a review there is no limit to how long the description can be | 2 | Create a verification on the app side that will only accept descriptions under a certain limit. |
| 5. | When a user attempts to edit a recipe and removes all of the alcoholic ingredients it should show an error alert | 2 | Do a validation check on alcoholic section of the edit page when the user submits the edited recipe |
| 6. | When the user attempts to edit a recipe and removes all of the no alcoholic ingredients it should show an error alert | 2 | Do a validation check on the no alcoholic section of the edit page when the user submits the edited recipe |
| 7. | On the edit ingredient page when a user changes the name of an ingredient it should be under 255 characters | 2 | Do a validation check on the name field on the ingredient edit page when the user selects the button to submit the edited ingredient. |
| 8. | If a user edits his/her email address on the edit account page his/her email should match the correct format | 3 | Add a regex validation check on the email field of the account page |
| 9. | Even when a user has an account their ingredients will only be stored locally | 2 | Have the mobile application interact with the web server to sync the logged in user’s ingredients to the database |

Code Inspection

|  |
| --- |
| In the mydrinks component on one of the tabs instead of pulling from the database the list is currently hard coded instead of being pulled from the database in the case of having an account or locally if the user does not have an account |
| Different components that have been written by different people will sometimes have the styling on the same page and sometimes will have it on another page. Although this has yet to cause a functional error for consistency purposes it should be the same |
| When searching for drinks the query is passed on to the database without being properly validated first. This can cause serious security issues. We fixed this promptly in the application and added additional sanitation on the web server. |

Unit Testing

1. On the back end of our application we continued to use automated testing. Our strategy from sprint one was successful so we continued in a similar fashion. We created URL queries to test the different components of the server and then recorded our expected result. Then we used a bash script to examine any differences in the output. We did not automate testing for the front end.

2. Front End

* Registering for an account
  + Valid Account
    - Input: {First name, Last name, email, password, password}
    - Output: Account is created and the user can now log in
  + Invalid Account
    - Input: {Missing information from above}
    - Output: User is notified that respective field can not be empty
    - Input: {passwords do not match}
    - Output: User is notified that the passwords do not match and can not proceed
    - Input: {email is missing '@' and '.'}
    - Output: User is notified that the email is not valid
* Logging In
  + Valid login
    - Input: {username, password}
    - Output: User is loaded into the app
  + Invalid login
    - Input: {username}
    - Output: Alert saying password can not be empty
    - Input: {password}
    - Output: Alert saying username can not be empty
    - Input: {}

Output: Alert saying username can not be empty

* + - Input: {Incorrect username, Incorrect password}
    - Output: User gets an alert saying invalid credentials
* User adds ingredient to Inventory
  + Valid add
    - Input: {type, brand, flavor, proof}
    - Output: The ingredient shows up in either the mixers or alcohol tab
  + Invalid add
    - Input: {missing field}
    - Output: User is notified that more information is needed
    - Input: {proof is outside of [0,200] range}
    - Output: User is notified that alcohol can only have a range of 0-200
* User removes ingredient from Inventory
  + Valid remove
    - Input: User taps remove
    - Output: Item is removed from inventory
* User adds a recipe
  + Valid add
    - Input: {name, alcohol, mixers, instructions}
    - Output: User is notified that more information is needed
  + Invalid add
    - Input: {missing field}

Output: The recipe is saved to the user’s recipes and other users can now search for it.

* + - Input: {Instruction field is to long}

Output: The user is notified that their instruction field is too long and needs to be shorter

* User deletes a recipe
  + Valid delete
    - Input: User taps remove
    - Output: Item is removed from the database and cannot be searched
* User edits a recipe
  + Valid edit
    - Input: User edits a field correctly
    - Output: Item can be accessed and edited information can be seem
  + Invalid Edit
    - Input: User leaves a field blank
    - Output: Alert is shown to the user to fill in the missing fielding
    - Input: User adds to much text to the instruction section
    - Output: An alert is thrown telling the user to shorten his/her instructions
* User reviews a recipe
  + Valid review
    - Input: User reviews a recipe
    - Output: Review can be seen on recipe and rating is updated
  + Invalid Edit
    - Input: User’s review is to long
    - Output: Alert is shown to the user to make the review shorter
* User updates contact information
  + Valid update
    - Input: {First Name} or {Last Name} or {Email} or {Password}
    - Output: The user is notified that the content is updated correctly
  + Invalid update
    - Input: {The old password is incorrect}
    - Output: The user is notified that the old password is not correct
    - Input: {The email address does not contain a '@' or '.'}

Output: The user is notified that the email is not valid

Back End

* Create an account
  + Created a valid account
    - Input: {username, password, firstName, lastName, email}
    - Output: json response with OK response
  + Created an account with empty fields (Error found and fixed)
    - Input: {username, blank password, firstName, lastName, blank email}
    - Output: json response with FAILED response
  + Created an account with fields that are too long (Error found and fixed)
    - Input: {username with length over 255, password, firstName, lastName, email}
    - Output: json response with FAILED response
  + Create an account with an already existing user
    - Input: {username, password, firstName, lastName, email}
    - Output: json response with FAILED response
  + Create an invalid account
    - Input: {username, lastName, email}
    - Output: json response with FAILED response
* Login to an account
  + Log into a valid account
    - Input: {username, password}
    - Output: json response with token
  + Log into an invalid account
    - Input: {username, password}
    - Output: HTTP response with 500 error
* Edit a user's information
  + Edit with valid information
    - Input: {token, username}
    - Output: json response with OK
  + Edit with empty information
    - Input: {token, empty parameter}
    - Output: json response with FAILED
  + Edit with information that is too long (Greater than 255 characters)
    - Input: {token, parameter over 255 characters}
    - Output: json response with FAILED
* Delete a user
  + Try to delete an invalid token
    - Input: {token}
    - Output: json response with FAILED
  + Try to delete a valid token
    - Input: {token}

Output: json response with OK