

San Francisco State University
SW Engineering CSC648 / 848 Spring 2020

“STOCK UP”

Team 103

Siddhita Chimote schimote@mail.sfsu.edu

Kevin Dizon-Cua

Calvin Tam

Sara Tama

Corey Russ

Matthew Ibrahim

Cassidy Mcskimming

Jose Pascual

Milestone 4

05/09/2020

Revisions

1. 05/09/2020

- First submission

Product Summary

Product name

“STOCK UP”

List of major committed functions:

1. User can register fridge
2. User can sign in
3. User can sign out
4. Ability to add a user
5. Ability to remove a user
6. User can view all their Inventory items
7. User can search for Items in their inventory
8. User can add an item to their inventory manually
9. User can remove an item from their inventory
10. User can add items to inventory by scanning receipt
11. User will be notified when an item expires (displayed on inventory screen)
12. User can view recipes
13. User can search recipes
14. User can view meal plan options
15. User can edit meal plans
16. User can create their own recipe
17. User can edit/create their own shopping list (Cart)
18. User can view refrigerator consumption report

Unique feature

Stock Up provides an application that makes organizing your refrigerator much easier. It is easy to lose track of items in your fridge that are old or have been forgotten. While there are other competitive applications available that offer similar features, Stock up uniquely provides the functionality of users identifying with multiple refrigerator ids. This applies to users that may have refrigerators in use at multiple locations such as at their house, at work, at multiple houses, etc. With this feature, users will be able to more accurately record all of their intake, as well as keep track of what items they have currently in stock at multiple locations.

URL: <http://team103.ddns.net/horizontal-prototype/>

QA Test Plan

Unit tests

Setup

For front and back ends, we are using AVA for unit testing on Node.js code. Ideally, Enzyme is used with AVA to test on React code, but we have not implemented testing with Enzyme.

Front-end

URL:

<https://github.com/CSC-648-SFSU/csc648-03-sp20-team103/tree/application-feature-product-prototype/source/application/tests/horizontal-prototype>

Test case	Scenario	Pass
/ GET 200 (Splash)	Valid Domain	✓
/users GET 200	Login	✓
/inventory GET 200	Click Inventory	✓
/inventory/view GET 200	Click View Button on Inventory	✓
/inventory/search GET 200	Enter Keywords to Search Inventory	✓
/inventory/add/receipt GET 200	Click Receipt Button then Add Button	✓
/inventory/add/barcode GET 200	Click Barcode Button	✓
/inventory/add GET 200	Click Add Floating Button on Inventory	✓
/carts GET 200	Click Carts	✓
/carts/view GET 200	Click View Button on Cart	✓
/meal-plans GET 200	Click Meal Plans	✓
/meal-plans/view GET 200	Click View Button on Meal Plans	✓
/recipes GET 200	Click Recipes	✓
/recipes/view GET 200	Click View Button on Recipes	✓
/recipes/create GET 200	Click Create Button on Recipes	✓
/recipes/search GET 200	Enter Keywords and Select Filters to Search Recipes	✓

/consumption GET 200	Click Consumption	✓
/consumption/view GET 200	Click View Button on Consumption	✓

Back-end

URL:

<https://github.com/CSC-648-SFSU/csc648-03-sp20-team103/tree/api-feature-product-prototype/source/api/tests/v3>

Test case	Description	Scenario	Pass
/register POST 200	Registering New User	Enter Serial Number (S/N); Enter PIN	✓
/login POST 200	Login Registered User	Enter Registered S/N and Registered PIN	✓
/login POST 406	Login Unregistered User	Enter Unregistered S/N or Unregistered PIN	✓
/login POST 400	Login with Invalid Login	Enter Invalid S/N; or Enter Invalid PIN	✓
/logout POST 200	Login Successful	Click on Logout Button	✓
/logout POST 400	Logout Invalid	Click on Logout Button	✓
/logout POST 406	Logout No Result	Click on Logout Button	✓
/users GET 200	Get Users List	Click Users Button in Menu	✓
/users GET 400	Getting Invalid User List	Click Users Button in Menu.	✓
/users GET 401	No Permission to Get User List	Click Users Button in Menu	✓
/users GET 406	User List No Result	Click Users Button in Menu	✓
/users POST 200	Create User Successful	Click Create New User, Enter Name, Role, Intolerances.	✓
/users POST 400	Create User Invalid	Click Create New User. Enter Invalid Name, Role, Intolerances.	✓
/users POST 401	No Create User Permission	Click Create New User Enter Name, Role, Intolerances.	✓
/users POST 406	Cannot Update Due To Restraints	Click Create New User. Fill Empty Name, Role, Intolerances.	✓
/users DELETE 200	Delete User Successful	Click Delete User in Users Menu	✓
/users DELETE 400	Delete User Invalid	Click Delete User in Users Menu	✓
/users DELETE 401	No Delete User Permissions	Click Delete User in Users Menu	✓

/users DELETE 406	Cannot Update Due To Restraints	Click Delete User in Users Menu	✓
/inventory/list/all GET 200	Get All Inventory List	Click on Inventory Button in Menu	✓
/inventory/list/all GET 400	Cannot Get All Inventory (Invalid)	Click on Inventory Button in Menu	✓
/inventory/list/all GET 401	No Permission to Get All Inventory	Click on Inventory Button in Menu	✓
/inventory/list/all GET 406	No Result Get Inventory	Click on Inventory Button in Menu	✓
/inventory/list/stored GET 200	Get Stored Inventory List	Click on Stored Inventory Button	✓
/inventory/list/stored GET 400	Invalid get stored inventory	Click on Stored Inventory Button	✓
/inventory/list/stored GET 401	No Permission to Get Stored Inventory	Click on Stored Inventory Button	✓
/inventory/list/stored GET 406	Cannot Insert Due To Restraints	Click on Stored Inventory Button	✓
/inventory/list/expired GET 200	Get Expired Inventory	Click on Stored Inventory Button	✓
/inventory/list/expired GET 400	Can't Get Expired Inventory	Click on Expired Inventory Button	✓
/inventory/list/expired GET 401	No Permission to Get Expired Inventory List	Click on Expired Inventory Button	✓
/inventory/list/expired GET 406	Cannot Insert Due To Restraints	Click on Expired Inventory Button	✓
/inventory/consume POST 200	Consume Storable in Inventory	Click on Consume Button of Storable Card	✓
/inventory/consume POST 400	Cannot Consume Storable in Inventory (Invalid)	Click on Consume Button of Storable Card	✓
/inventory/consume POST 401	No Permission to Consume	Click on Consume Button of Storable Card	✓
/inventory/consume POST 406	Cannot Insert Due To Restraints	Click on Consume Button of Storable Card	✓
/inventory/discard POST 200	Discard Storable in Inventory	Click on Discard Button of Storable Card	✓
/inventory/discard POST 400	Cannot Discard Storable in Inventory (Invalid)	Click on Discard Button of Storable Card	✓
/inventory/discard POST 401	No Permission to Discard	Click on Discard Button of Storable Card	✓
/inventory/discard POST 406	No Result For Discarded Storable	Click on Discard Button of Storable Card	✓
/inventory/add/manual POST 200	Add Inventory Items Manually	Click on Add Button in Receipt Page	✓

/inventory/add/manual POST 400	Add Inventory Items Manually (Invalid)	Click on Add Button in Receipt Page	✓
/inventory/add/manual POST 401	Add Inventory Items Manually (Unauthorized)	Click on Add Button in Receipt Page	✓
/inventory/add/manual POST 406	Cannot Insert Due To Restraints	Click on Add Button in Receipt Page	✓
/ingredients POST 200	Add Ingredients	Create Recipe and Add Ingredients Button	✓
/ingredients POST 400	Add Ingredients (Invalid)	Create Recipe and Add Ingredients Button	✓
/ingredients POST 401	Add Ingredients (Unauthorized)	Create Recipe and Add Ingredients Button	✓
/ingredients POST 406	Cannot Insert Due To Restraints	Create Recipe and Add Ingredients Button	✓
/ingredients/search GET 200	Ingredient Searched	Enter Keywords on Search Bar in Ingredient Page	✓
/ingredients/search GET 400	Search Invalid	Enter Keywords on Search Bar in Ingredient Page	✓
/ingredients/search GET 401	Unauthorized Search	Enter Keywords on Search Bar in Ingredient Page	✓
/ingredients/search GET 406	Cannot Insert Due To Restraints	Enter Keywords on Search Bar in Ingredient Page	✓

URL for tracking bugs

<https://github.com/CSC-648-SFSU/csc648-03-sp20-team103/pulls?q=is%3Apr+is%3Aclosed+label%3Abug>

Test coverage

All endpoints return different status code on different scenarios, such as invalid parameters (400), unauthorized (401), no result / duplicate key (406), and internal server error (500). By running tests with different test data, we test if the endpoints are working as expected.

Integration tests

Priority 1 features

1. Fridge Management
 - a. Registration
 - b. Login
 - c. Logout
2. User Management
 - a. Creation
 - b. Deletion
 - c. Selection
 - d. Intolerances
3. Inventory Management
 - a. Viewing items in inventory
 - b. Searching for ingredients (user intolerances respected)
 - c. Manually adding items
 - d. Manually removing items
 - e. Expiration
4. Recipes
 - a. Viewing recipes favorites
 - b. Searching for recipes (user intolerances respected)
 - c. Favoriting recipes

System setup

Setup ID	Device	Operation system	Browser
1	Desktop	Windows 10	Google Chrome
2	Desktop	Windows 10	Mozilla Firefox
3	Apple MacBook Pro	macOS Catalina	Safari
4	Apple MacBook Pro	macOS Catalina	Mozilla Firefox
5	Apple iPhone 8	iOS 13	Safari
6	Samsung Galaxy S9+	Android 10	Google Chrome

Integration Test Cases

Test Case ID	Test case	Scenario	Test Results
1	Register a Fridge	Serial number and Pin auto generated	✓
2	Login	Enter Serial Number and Pin to Login	✓
3	Select User	Select an existing user	✓
4	Create user	Create a new user	✓
5	List of Inventory	User can view Inventory List	✓
6	Search Inventory	User can search an inventory	✓
7	Add Item to Inventory	User can add item to the inventory	✓
8	Consume inventory	User can consume item from the inventory	✓
9	Discard Inventory	User can discard item from an inventory	✓

Beta Test Plan

The objective of our beta testing is to ensure compatibility of our major features across different users and browsers. By having major features incorporated into beta testing, users can expect a final product similar to the one they are testing. We will collect feedback from the users including, but not limited to, ease of access to features, simplicity and ease of overall use, visual appearance of the application, and performance of the application. Our beta testers will include anyone unfamiliar with the design of our application, such as family members, close friends etc. The opinion of users unrelated to the project is valuable since they provide an unbiased and honest opinion of their use. We will collect all of the feedback from users via email. This will ensure proper documentation of experiences rather than remembering feedback from word of mouth.

System setup

Setup ID	Device	Operation system	Browser
1	Desktop	Windows 10	Google Chrome
2	Desktop	Windows 10	Mozilla Firefox
3	Apple MacBook Pro	macOS Catalina	Safari
4	Apple MacBook Pro	macOS Catalina	Mozilla Firefox
5	Apple iPhone 8	iOS 13	Safari
6	Samsung Galaxy S9+	Android 10	Google Chrome

URL

<http://team103.ddns.net/horizontal-prototype/>

Code Review

The style guide of our project follows the style guide of Airbnb (<https://github.com/airbnb/javascript>). Modern JavaScript features are utilized (e.g. let for mutable variables and const for constant ones, spread operator, etc.), single quotes for strings, soft tabs (2 spaces) for indentation, etc. All changes to the product undergo mandatory code review by a team lead before being merged. A JavaScript linter with the Airbnb style guide in mind is recommended for everyone working on the project (<https://www.npmjs.com/package/eslint-config-airbnb>). Our code review process is generally done over discord due to how quickly feedback can be given and responded to, though an example can be seen here:

Fix parameter checking logic for users. #86

Merged Matthewl123 merged 2 commits into `api-feature-product-prototype` from `api-feature-product-prototype-users-fix` 5 days ago

Conversation 5 Commits 2 Checks 0 Files changed 1 +17 -16

Matthewl123 commented 5 days ago • edited

1. Ensure request's body size contains exactly 4 objects with the required fields.
2. Type-check the session, name, and role parameters.
3. Ensure each intolerance in the intolerances array is a known intolerance as per the spoonacular API.
4. The result set returned after an insertion is not an array of rows (see JSON Result-sets in <https://github.com/mariadb-corporation/mariadb-connector-nodejs/blob/master/documentation/promise-api.md>).

Fix parameter checking logic for users. Verified 7cdb3c6

Matthewl123 added the `bug` label 5 days ago

Matthewl123 requested a review from `ctam4` 5 days ago

ctam4 requested changes 5 days ago

source/api/routes/v3/users.js Outdated Show resolved

source/api/routes/v3/users.js Outdated Show resolved

source/api/routes/v3/users.js Outdated Show resolved

Matthewl123 requested a review from `ctam4` 5 days ago

ctam4 reviewed 5 days ago

source/api/routes/v3/users.js Outdated Show resolved

Reviewers

ctam4

Assignees

No one—assign yourself

Labels

bug

Projects

None yet

Milestone

No milestone

Linked issues

Successfully merging this pull request may close these issues.

None yet

Notifications

Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

2 participants

ctam4 Matthewl123

Lock conversation

<https://github.com/CSC-648-SFSU/csc648-03-sp20-team103/pull/86>.

A couple of issues corresponding to a particular endpoint have been fixed and the original author had been requested to review the changes. Additionally, all tests must pass when reaching the master branch. These tests are done with the ava testing framework and are checked on each commit and pull using GitHub Actions.

Self-check

Adherence to original Non-functional specs : Up to 1/3 page

Performance-- ON TRACK

1. Front-end renders React UI on the client side with minimum performance degradation.
2. Back-end responses to front-end requests on demands as it is running stateless containers.

Security-- DONE

3. All data sent and received are using HTTPS protocol via API, and are protected by SSL / TLS.
4. Every device has to be registered before they can access the application. (Similar to WhatsApp Web.)

Infrastructure-- ON TRACK

5. Data shall be stored in the team's chosen database technology, which is MariaDB, on the team's deployment server, which is Google Cloud.
6. Application is deployed on Google Cloud Run on a Google Kubernetes Engine cluster, which can automatically scale up when demands increase with load balancer if enabled.
7. Application is possible to run on multi-cloud using Kubernetes to avoid cloud service downtime.

UX / UI-- ON TRACK

8. Application UI shall be easy to use and intuitive, especially adding and removing items.
9. Front-end should not allow to send duplicated requests to the back-end while back-end can distinguish a duplicated request and not to process it.

Accuracy-- ON TRACK

- 10. The inventory should be accurate.
- 11. The expiration date of storables should be accurately monitored.
- 12. The (nutrition) consumption should be accurate.

Coverage-- ON TRACK

- 13. Application should be able to process "Safeway" receipt when it contains non-edible items

Project & code management-- DONE

- 14. The codebase should be well maintained so newcomers can easily follow up with the code and add up their own functionalities.
- 15. There is a code style guide for formatting code in a consistent and organized way.
- 16. There are primary (master, develop, api, application), dedicated (sub)feature and (sub)patch branches in the Git repo.
- 17. The codebase has continuous integration to run unit tests.
- 18. All branches in the Git repo are merged by pull requests only after passing all unit tests.