SW Engineering CSC648/848 2019 – Team 102

PROJECT – DIGITAL INVENTORY RECOLLECTION TERMINAL

COLLABORATORS

Team Leader
Sanchit Joshi (sjoshi2@mail.sfsu.edu

SCRUM Master
Aurpon Gupta

Back End Leader
Zhifan Cai

Front End Leader
Anthony Wong

Git Master
Matthew Davis

Database Developer
Daniel Godfrey

Server Developer
Christian Melendez

Implementation Developer
Dmitry Polozov

REVISIONS HISTORY

Version	Summary
1	Initial Document Creation
2	

PRODUCT SUMMARY
D.I.R.T.(Digital Inventory Revolution Terminal)

Major Functionalities of the Application-

1. Login/Register to the Application-
   - User should be authenticated while login
   - User can be added by registering to the application

2. Manage Inventory
   - User can see the items in the refrigerator
   - User can add items to the refrigerator inventory
   - User can delete items from refrigerator inventory
   - User can update quantity of items in the refrigerator inventory

3. Scan Receipts functionality
   - User can upload a CostCo Wholesale receipt to add multiple items to the refrigerator inventory in one go

4. Manage Shopping lists
   - User can add items to Shopping List
   - User can delete items from Shopping List
   - User can move an item from Shopping List to Inventory

5. User consumption Statistics
   - User can see the nutrition consumption reports on daily basis.
   - User can see calories, proteins

6. Meal Plan
   - User can add a meal plan,
   - User can search for a meal to add to the plan

7. Recipe Recommendation
   - User can see the recipes that can be prepared using the inventory
   - User can add a recipe of their own to the list

QA Test Plan

1. Unit Testing
   - Function level unit testing will be performed for the below features-
     - **San receipts**
     - **Inventory Management- add, delete, update**
     - **User Management**.

   - **HW Setup**– smartphone activated with an internet connection(Wi-Fi or Cellular)
   - **SW Setup-** Mocha dependencies, Jest and React testing Library dependencies.

   Unit Testing Framework-

   a) Mocha (Backend)
   Mocha is a JavaScript test runner that runs both on Node.js and in the browser. It provides functionality for testing both synchronous and asynchronous code with a very simple and similar interface.

   b) Jest- React Testing Library

   Jest is a JavaScript test runner that lets you access the DOM via jsdom. While jsdom is only an approximation of how the browser works, it is often good enough for testing React components. Jest provides a great iteration speed combined with powerful features like mocking modules and timers so you can have more control over how the code executes.

   React Testing Library is a set of helpers that let you test React components without relying on their implementation details. This approach makes refactoring a breeze and also nudges you towards best practices for accessibility. Although it doesn't provide a way to "shallowly" render a component without its children, a test runner like Jest lets you do this by mocking.

2. Integration Test Plan

   Integration testing will be performed on all the major functionalities listed as below

   - Login/Register to the Application-
   - Manage Inventory
   - Scan Receipts functionality
   - Manage Shopping lists
   - User consumption Statistics
   - Meal Plan
   - Recipe Recommendation

   Github Link for Test Cases-

3) Beta Testing Plan

- Test objectives:
    - Evaluate Customer Experience
    - Evaluate the Usability of the Product
    - Evaluate Scalability and performance of the product

- Test plan:
    - System setup
        - Users can access the web application through their smartphones a mobile phone with the latest browser support.
    - Main objective of the beta test (i.e, what you would like to learn out of it)
    - Intended users
        - The intended users for Beta testing will be the other teams in class or team's relatives or friends who wish to provide positive and critical feedback for the application usability
    - Collecting User feedback
        - Collecting feedback from users is the main reason for Beta Testing. We are planning to take feedback via email with the steps that the user performed and the actual result screenshots for the functionality. A developer will then check the functionality
    - Collect the behavior of your product (which features are used the most, how long user spent for each feature, and what is the demographics of your user, etc..)
        -

- **URL of the system to be tested -** https://dirt-275118.wl.r.appspot.com

(This is the current link. This might change in case of any issues with deployment)

4) Code Review

- **Coding Style**
  - We used the Javascript standard coding style which is defined within Visual Studio Code
  - Below files comply with the standard
    1. https://github.com/CSC-648-SFSU/csc648-03-sp20-team102/blob/Dev-0.1/application/team102/backend/server.js
    2. https://github.com/CSC-648-SFSU/csc648-03-sp20-team102/blob/Dev-0.1/application/team102/backend/visionServer.js
    3. https://github.com/CSC-648-SFSU/csc648-03-sp20-team102/blob/Dev-0.1/application/team102/UI/src/Component/inventory.js
    4. https://github.com/CSC-648-SFSU/csc648-03-sp20-team102/blob/Dev-0.1/application/team102/UI/src/Component/Home.js
    5.
  - Code Review Policy
    - We have a peer review for each module. The team member who commits a code appoints the enter team as a reviewer. Points are discussed in meetings and merge is pushed after that.
    - During peer review the functionality is demonstrated, each team member give their inputs. Merge is completed if the team is satisfied with the feature else rejected with update comments

5) Self-check: Adherence to original Non-functional specs :

Non Functional Requirement	Status
Application shall be developed, tested and deployed using tools and servers reviewed by Class TA in M0	Done
Application shall be optimized for mobile browsers.	Done
Data shall be stored in the team's chosen database technology on the team's deployment server.	Done
Application shall be very easy to use and intuitive especially to add items and remove them.	Done
The accuracy : the nutrition consumption should be accurate.	Done
Coverage : the product should handle the "Safeway" receipt as much as possible even when it contains non-edible items.	Done
The code should be well maintained so that new comers could easily read and add up their own functionalities.	In Progress