Senior Design

ENG EC 463

Memo

To: Professor Pisano
From: Trevor Chan, Jeffrey Chen, Andres Garcia, Cole Wentzel
Team: 10
Date: 2/29/2024
Subject: FridgeBuddy - Second Prototype Testing Plan

1.0 Required Materials

Hardware:

1. Mobile phone (or emulator)
2. Tablet

Software:

1. AWS Services
    a. IAM Users (allow code to access AWS services)
    b. Amplify (interface for connecting React Native to AWS services)
    c. DynamoDB (user and item information storage)
    d. Lambda (run Python scripts)
        i. Insert user item using 3rd-party UPC lookup API
    e. AppSync (define how React Native interacts with DynamoDB and Lambda)
2. React Native (User interface)
    a. Expo Go (development runtime for browser/mobile devices)
3. Third-party APIs
    a. Spoonacular Recipe API
    b. Clarifai API

2.0 Setup

The setup is divided into two components: the main FridgeBuddy React Native application running on a tablet and mobile device, and the Spoonacular recipe API calls for getting recipe information from item inputs.

The tests for the FridgeBuddy application are to demonstrate the base functionalities of the project that have been implemented. This includes adding items to a test user through manual entry, object recognition, and barcode scanning, as well as displaying, editing, and removing current items belonging to the user. The frontend user interface was developed using the React Native framework and TypeScript. It is run and deployed to the tablet / mobile device using the Expo platform, which utilizes Metro internally to bundle the application. The tests will show that the frontend application can display a list of items on the main screen with each item's name, expiration date, category, quantity, and calories (if the item has those fields), as well as showing that the item information is synchronized with the user's data stored in DynamoDB. The tests will also show the application being able to make changes to the user's items (manually create an item, edit existing items, and remove items), with the changes made on the application synchronizing to the DynamoDB data.

The user and item data are stored in the DynamoDB database backend. The React Native application can query and mutate the DynamoDB data because of a GraphQL schema that defines the DynamoDB entity data schema, as well as how the application can interact with DynamoDB.

Barcode scanning and object recognition tests will also be performed on the FridgeBuddy application to demonstrate the alternate, automated methods for adding item information to the user's data. The application's barcode scanner was implemented using the expo-camera library. When a barcode is detected within the camera view, its UPC is retrieved and sent to an AWS Lambda function that gathers item information using the FoodCentral API and then inserts the item into the user's data in DynamoDB. The smart scanner uses the Clarifai API and a food item recognition pre-trained model to classify food items in the camera view. Predictions are then used to retrieve item information so that it can be added to the user's inventory using the same Lambda function.

The tests for the Spoonacular API are to demonstrate the functionality of getting recipes using items the user has in their inventory, particularly a small list of items that may expire soon. Although this functionality has not been built into the app yet, the baseline functionality of being able to make requests for recipes based on ingredients and retrieve the data for any particular recipe is functional. A GET request is formatted using an array of items the user possesses. This returns tuples of recipe IDs and recipe names. Recipe IDs can further be used to get information about a particular recipe, such as its complete ingredient list and the steps needed to make the recipe.

3.0 Pre-Testing Setup Procedure

FridgeBuddy Application Tests:

1. Ensure that AWS services are up and React Native can connect to the AWS backend
2. Prepare test input barcodes and items
3. Start application with "npm start"
4. Connect Expo Go runtime on tablet to the host application via a QR code.
5. Connect Expo Go runtime on mobile device to the host application via a QR code.

Spoonacular Recipe API Tests:

1. Ensure Spoonacular services are operational
2. Prepare test inputs

4.0 Testing Procedure

FridgeBuddy Application Tests:

1. Test that application can query user items
2. Manually add items
    a. Input item fields for:
        i. Name
        ii. Expiration date
        iii. Category
        iv. Quantity
        v. Calories
    b. Manual add item date prompts with calendar pop-up
3. Edit existing item
    a. Name
    b. Quantity
    c. Expiration date (prompts with calendar pop-up)
4. Remove existing item
5. Add items using barcodes
    a. Strawberry filled crepes
    b. Bottle of milk
    c. Original kettle cooked potato chips
6. Add items using item recognition
    a. Banana
    b. Apple
    c. Pizza slice
7. Mobile Application tests
    a. Test that mobile application can query user items
    b. Test that mobile application can add item that shows up on tablet

5.0 Measurable Criteria

The criteria for successful running and output is as follows:

1. FridgeBuddy Tablet
    a. User is able to query correct item information from DynamoDB
    b. User is able to display correct item information from DynamoDB
    c. User is able to manually add items on application
    d. User is able to input expiration date using calendar prompt
    e. Manually added items and inputted fields show up on DynamoDB
    f. User is able to edit items on application
    g. Edited item fields show up on DynamoDB
        i. Name
        ii. Quantity
        iii. Expiration date
    h. User is able to remove items from user interface (X) button
    i. Removed items are deleted from DynamoDB
    j. User is able to scan barcodes using the tablet or mobile application's camera
        i. UPC pops up once scanned
        ii. Once it scans the UPC code, item information is added to user's items within 5 seconds
    k. User is able to scan objects using the tablet or mobile application's camera
        i. Possible items are logged to the console
        ii. Once item is recognized, item information is added to user's items within 5 seconds
    l. Mobile application displays changes made during the tablet tests
    m. User is able to add items to the mobile application
    n. Changes made to the mobile application appear on the tablet once refreshed (within < 2 seconds)

2. Spoonacular API
    a. User is able to query for recipes using an array of ingredients and receive 8-10 recipes that contain at least one of the ingredients specified, but will attempt to use more if possible
    b. The recipes will return with their name and a recipe ID
        i. Returned as an array of tuples
    c. The returned recipe IDs can be used to get
        i. A detailed list of steps
        ii. A list of ingredients needed for the recipe

6.0 Score Sheet

FridgeBuddy Tablet Tests		
Test #	**Functionality Working**	**Correct? (Y/N)**
1	User is able to query correct item information from DynamoDB	Y
2	User is able to display correct item information from DynamoDB	Y
3	User is able to manually add items on application using Add Items field and + button	Y
4	User is able to input expiration date using calendar prompt	Y
5	Manually added items and inputted fields show up on DynamoDB	Y
6	User is able to edit items on application using Edit button	Y
7	Edited item fields show up on DynamoDB (Name, quantity, and expiration date)	Y
8	User is able to remove items from user interface (X) button	Y
9	Removed items are deleted from DynamoDB	Y
10	Mobile application displays changes made during the tablet tests	Y
11	User is able to add items to the mobile application	Y
12	Changes made to the mobile application appear on the tablet once refreshed (within < 2 seconds)	Y
Result		12 / 12

FridgeBuddy Barcode Scanning Tests						
Test #	**UPC**	**Name**	**Category**	**Calories**	**Correct Info (Y/N)?**	**Appears in Database (Y/N)?**
1	852160006022	Strawberry filled crepes	Pancakes, waffles, french toast & crepes	375	Y	Y
2	811620020633	Complete protein 42g elite high protein milk shake, chocolate	Energy, protein & muscle recovery drinks	56	Y	Y
3	020685001642	Original kettle cooked potato chips	Chips, pretzels & snacks	500	Y	Y
Result					3 / 3	3 / 3

FridgeBuddy Food Item Recognition Tests					
Test #	**Name**	**Category**	**Calories**	**Correct Info (Y/N)?**	**Appears in Database (Y/N)?**
1	Banana, raw	Bananas	97.0	Y	Y
2	Apple, raw	Apples	61.0	Y	Y
3	Pizza	Pizza	199	Y	Y
Result				3 / 3	3 / 3

Spoonacular API Recipe Search			
Test #	**Ingredients**	**Returned Recipe IDs**	**Correct? (Y/N)**
1	apples, flour, sugar	673463, 633547, 663748, 715381, 639637, 635315, 1155776, 652952, 635778	Y
2	eggs, milk, lemon	641759, 649609, 991625, 665573, 635964, 633574, 633574, 652952, 665574, 638717, 643972	Y
3	butter, asparagus, strawberry, cheese	1697591, 633167, 633165, 645733, 642058, 661447, 1077392, 655477, 644958, 1697653	Y
Result			3 / 3

Spoonacular API Instructions Search			
Test #	**Recipe ID**	**Recipe Name**	**Steps Returned? (Y/N)**
1	635315	Blood Orange Margarita	Y
2	661447	Square Deviled Eggs	Y
3	641808	Easy 4 Ingredient Texas Toast	Y
Result			3 / 3

7.0 Conclusion

Each of our tests ran successfully. The React Native mobile application on both mobile devices and the tablet demonstrated the correct working functionalities connected to the AWS backend. This consisted of querying, editing, and deleting existing items and adding items manually, through barcode scanning or object recognition. Our Spoonacular API calls were also successfully able to gather recipe information using a list of inputted item names.

Based on observations we made while working on our prototype tests and instructor feedback, for the next steps of the project, we want to modify our queries to FoodCentral for gathering item information by object recognition, implement a way to perform object recognition on multiple food items, implement text recognition for food labels, and add local offline storage of user items using Amplify DataStore.

While working on the tests, we found that querying the FoodCentral database using item names will not always give us the correct item information. For example, querying the database with an item's name like "banana" returns information about "Banana Butter," rather than giving us information about a regular banana. For our demo, we were able to fix this problem specifically for produce by adding a button that would toggle a “+raw” tag to the item query (because produce like apples and bananas are named “<item>, raw” in FoodCentral). Although this has worked for the tests, we believe we will need to create a more complex system of adding tags so that FoodCentral will give us the proper item information when querying by item name.

During the demo, an instructor mentioned that we should try to enable the smart scanner to scan for multiple objects using the same image. When testing whether this would work, the tablet was successfully able to recognize both “apple” and “banana” as possible outputs. For the future, we want to add a prompt that lists the top 3 (or at least the most possible) item names returned by the Clarifai API so that users will be able to select one or more items scanned (as well as if the first option isn’t the correct one).

Based on feedback, another method for adding items we want to work on is text recognition of food labels. If we are able to complete the initial features and functionalities we planned on adding, we will work on implementing it as a reach goal.

A goal that we have had since before the prototype testing was to add local, offline storage to the device using AWS Amplify DataStore. DataStore would allow the device to locally store item information, and if any mutations or queries are made while the device is offline, they will be stored and queued for when the device reconnects to the internet. Our progress in setting up DataStore has been slowed down due to limitations in documentation (and because Amplify is set up so that the GraphQL schema in the Amplify project versus the AWS console cannot be synced). However, once DataStore is set up, this should make implementing auth roles easier to start implementing user authentication.