Senior Design

ENG EC 463

Memo

To: Professor Pisano
From: Trevor Chan, Jeffrey Chen, Andres Garcia, Cole Wentzel
Team: 10
Date: 11/16/2023
Subject: FridgeBuddy - First Prototype Testing Plan

1.0 Required Materials

Hardware:

1. Mobile phone (or emulator)

Software:

1. AWS Services
    a. IAM Users (allow code to access AWS services)
    b. Lambda (running Python scripts)
        i. Insert user item using 3rd-party UPC lookup API
        ii. Get all user items
    c. DynamoDB (user and item information storage)
2. React Native (User interface)

2.0 Setup

The equipment and setup are divided into two main components: the frontend React mobile application user interface and the AWS storage and computation back-end currently demonstrated as AWS Lambda functions running python scripts. For the mobile application interface tests, we will connect the application to a mobile Android device to demonstrate its functionality.

The frontend of the mobile application is developed using the React Native framework and TypeScript. It is run via the Expo platform, which utilizes Metro internally to bundle the application. Currently, the application is able to display a list of items on its main screen. Each item also displays a name and an optional expiration date and can individually be deleted. Users can currently add items by manually typing a name and optionally input a date, which prompts a calendar widget to appear on the screen. For development and testing, the application can be initiated on the host machine via the “expo start command”, which will display a QR code on the terminal. A mobile device can then scan the QR code to run the React Native application on the Expo Go runtime.

The AWS back-end comprises two main services: DynamoDB and Lambda connected using AWS IAM user role permissions. DynamoDB is a NoSQL (unstructured) database storage service that we’re using to store information about our users and items. Lambda is a computing platform that allows users to execute specified code as containerized workloads. AWS IAM user roles with the necessary permissions settings allow the Lambda functions to send read and write requests to the DynamoDB database. The back-end tests consist of two of the core user stories of our project as Lambda functions running python scripts: inserting an item for a user using a Universal Product Code (UPC), and querying all items belonging to an inputted user. The item insert test is performed by a Python script taking a UPC code and user id as input. The script sends an HTTP request to the United States Department of Agriculture’s FoodData Central API for item information using the UPC, and if the script gets a response it then inserts the item information (name, UPC, category, calorie count) into the DynamoDB database. The user item fetch test uses an inputted user id to run a DynamoDB query request for all rows with user id as its primary partition key, returning a response with the item information of all items belonging to the user.

3.0 Pre-Testing Setup Procedure

Phone Application:

1. Host mobile application with “expo start command”
2. Connect Expo Go runtime on mobile device to the host application via a QR code.

AWS Lambda Tests:

1. Ensure that the AWS services are up
2. Specify UCP and user id for each test sample:
    a. Test 1: Coconut Water Bars - Mango Flavored
    b. Test 2: Fairlife Vanilla Nutrition Shake
    c. Test 3: Good and Gather Green Beans
    d. Test 4: Lay’s Potato Chips
    e. Test 5: Diet Coke, 20 oz
    f. Test 6: Sargento Cheddar Snacks
    g. Test 7: Non-existing UPC code

4.0 Testing Procedure

Mobile Application Test

1. Test item name input
    a. Input item name and add item
    b. Add item without inputting name
2. Test item date input
    a. Press “Add Date” to show calendar pop-up
        i. Press “Add Date” again to cancel the calendar pop-up
        ii. Press a date on the calendar input it into the item
        iii. Add an item with a date should succeed
        iv. Add an item without a date should succeed
3. Delete items in arbitrary order

AWS Lambda Tests:

1. Query all information from the example user
2. Insert the specified 7 items into the database
3. Query all information from the user (items should show up)

<u>5.0 Measurable Criteria</u>

The criteria for successful running and output is as follows:

1. Mobile Application User Interface
    a. User is able to manually add items with a date
    b. User is able to manually add items without a date
    c. User is able to add items with a name
    d. User is not able to add items without a name
2. AWS Lambda Functions
    a. The insert item Lambda function should successfully gather the correct item information (item name, calories, and food category)
    b. The insert item Lambda function should successfully insert the item information into DynamoDB (shows up with the matching information under the correct user)
    c. The insert item Lambda function should run for $\leq$ 3s
    d. If there is no item matching the requested UPC, should return a 404 response error
    e. The query items Lambda function should successfully gather the up-to-date item information for all items associated with the user
    f. The query items Lambda function should run for $\leq$ 3s

<u>6.0 Score Sheet</u>

Mobile Application Tests		
Test #	**Functionality Working**	**Correct? (Y/N)**
1	User is able to manually add items with a date	Y
2	User is able to manually add items without a date	Y
3	User is able to add items with a name	Y
4	User is unable to add items without a name	Y
Result		100%

AWS Lambda Item Information Fetch Tests					
Test #	**UPC**	**Name**	**Category**	**Calories**	**Correct? (Y/N)**
1	853883003374	Coconut Water Bars - Mango Flavored	Ice Cream & Frozen Yogurt	112	Y
2	811620022149	Vanilla Nutrition Shake	Milk	44	Y
3	085239325742	Green Beans	Canned Vegetables	35	Y
4	028400421737	Lay's Potato Chips	Chips, Pretzels & Snacks	571	Y
5	00049000000450	Diet Coke, 20 oz	Non Alcoholic Beverages Ready to Drink	0	Y
6	046100339404	Sargento Cheddar Snacks	Cheese	395	Y
7	000000000000	N/A	N/A	N/A	Y
Result					100%

AWS Lambda Get User Item Tests					
Test #	**UPC**	**Name**	**Category**	**Calories**	**Correct? (Y/N)**
1	853883003374	Coconut Water Bars - Mango Flavored	Ice Cream & Frozen Yogurt	112	Y
2	811620022149	Vanilla Nutrition Shake	Milk	44	Y
3	085239325742	Green Beans	Canned Vegetables	35	Y
4	028400421737	Lay's Potato Chips	Chips, Pretzels & Snacks	571	Y
5	00049000000450	Diet Coke, 20 oz	Non Alcoholic Beverages Ready to Drink	0	Y
6	046100339404	Sargento Cheddar Snacks	Cheese	395	Y
7	000000000000	N/A	N/A	N/A	Y
Result					100%

7.0 Conclusion

Each of our tests ran successfully. The user interface of the React Native mobile application demonstrated all the current working functionalities (adding items, deleting items, inputting expiration dates, inputting item names) as expected. The AWS Lambda functions successfully fetched the FoodCentral API with the expected values and wrote the results into the DynamoDB database.

For the next steps of our project, we plan on working towards creating the complete mobile application by connecting the React Native front-end user interface with the AWS backend storage and computation using AWS Amplify. AWS Amplify provides an SDK for working with mobile applications for connecting to AWS services and requesting things like running Lambda functions. We are planning on connecting the DynamoDB backend to the user application using AWS Amplify DataStore, which allows the user to store and change their information from DynamoDB locally while handling data synchronization. This allows us to minimize the number of direct queries we have to run on the database, lowering costs of running the backend on AWS.

We also need to create the recipe suggestion function using user data stored in our DynamoDB backend. The API we use formats ingredients in a way that is easily searchable by the API, but we will need to implement additional functionality, such as prioritizing recipe suggestions with food that is expected to expire.