Project Technical Report

Food Scanning Application

by Blue Team

INSTRUCTOR: Professor Maherukh Akhtar

https://github.com/kevhunte/FoodScanningApp

Member: Kevin Hunte

NYIT ID: 1030750

Member: Fernanda Tovar

NYIT ID:1090913

Member:Akash Ravindran

NYIT ID: 0865256

Member: Nick Lombardi

NYIT ID: 1286497

New York Institute of Technology -Dept. of Computer Science

CSCI-665:Software Engineering-Spring 2020

Table of Contents

Abstract

The purpose of this project is to design a mobile application that encourages the user to have a healthier lifestyle. A healthy lifestyle includes a balanced diet and consistent exercising. We noticed the importance of eating healthy and the impact it has on one's lifestyle. There are a variety of food options and with that said it can be hard to track what one consumes or even find those given items. For this application, we focus on one's eating habits to maximize their healthy lifestyle. The way that our application works is that it easily scans food barcodes and QR codes. Scanning a food item can provide the nutritional value of that product.Once a user logs in, they have the ability to store their food information and view where they can purchase their scanned food items based on their GPS location. Overall, it's a system to scan and manage the information of food items, so the user can stay on top of their health goals.

Index Terms: barcode, QRCode, food, nutrition, health, GPS, store, mobile, app, track,website,android, ios.

1. Introduction

1.1. Existing Systems

We will create a new version of this system, but review other applications in industry for comparison.

One application on the market is called Fooducate. This general-purpose application serves as a nutrition and diet tracker. It allows users to scan items and retrieve nutritional information. It can track calories and nutrients. Fooducate categorizes food recommendations for different diets. Next, an application called Shopwell allows users to scan items and retrieve nutritional information. A smart feature is that it recommends similar products to what the user has scanned before. As well, it gives users a score on how well a product matches their current diet habits. Last but not least, we have Ultimate Food Diary Plus which also allows users to scan

items and retrieve nutritional information. This application creates a variety of nutritional plans for users. It includes a weight, body measurement and calorie tracker.

1.2. Proposed System

1.2.1. Advantage of Proposed System

By learning from the best comparable existing systems such as Fooducate, Shopwell, and Ultimate Food Diary Plus, we intend on creating a system that utilizes the best properties from all three, while also implementing some features of our own that we think will complement the application nicely. By starting from scratch, we will understand the logistics behind creating all these different features and will also be able to prioritize the most important ones given the time constraint that we face. By doing so, we provide our users with an application that will deliver on the base needs while also containing quality-of-life features that will improve their user experience as time goes on.

1.2.2. Proposed System Changes

Due to the time constraint, our main priority in terms of features will be to get the scanning, the login, and the location features up and running. Everything else will be classified as quality-of-life-changes that will be implemented at a later stage, if the chance presents itself. These include favorites, nutrition tracker and followers. The “favorites” feature is a way to partition items. This means that they will have a maximum number of folders and can provide customized names to all of their folders depending on what they deem applicable. The nutrition tracker feature can be categorized as a food diary, where for every 2 weeks, a user can track their calories and other nutritional facts consumed from their scanned items. Lastly, we have a “followers” feature which allows users to be followed by their friends, nutritionist and even trainer to interact with their activity in order to encourage healthy eating.

1.3. Software Engineering Model

For this application, our team will be utilizing the Concurrent Model. We believe that the concurrent model fits our requirements best for a few reasons. The concurrent model allows for multiple objectives to be worked on simultaneously. This way, we're able to split our backend and frontend responsibilities and the other factor is due to the fact that we're running short on time because of our team's late start, we're in need of all the time and productivity that we can get in the short span that we have to remain. We'll also be implementing Feature-driven development in order to make sure we hit all our most important requirements (Barcode scan, Location, log in, etc.).

1.4. Purpose

To increase the healthy lifestyle of our users through the use of tracking one's food intake and locate those foods for easier consumption.

1.5. Project objective/Goals and Scope

The objective is to develop a fast and easy application for users to continuously view their food intake to achieve their health goals. The main goal is to create a cross-platform application that can work for android and ios users. It's important to incorporate more users, so we have a bigger audience to impact. Another goal is to design an app that is easy to use, which means the app design is simple, but yet appeals to the user's eye. With that mentioned, the app's layout consists of a few bottom tabs and a pull-out menu to navigate easily through the app. An important bottom tab was the scanning and this opens the user's camera to scan the given barcode. As well, the color palette is smooth and modern to create a pleasant and peaceful user experience. Another main goal was to make sure that the user can securely login in order to make sure they can see their scanned history information.

1.6. Technologies & Tools

The application consisted of a compilation of many different applications, languages, APIS, and frameworks to make it possible to build an application of this magnitude. The main focus first was to make sure that we had a way to share the code/ files in an organized way. This

was done with Github. Github has a feature where one can create a project and with that then start a kanban board where they can keep track of the to dos. Then the to dos were converted into issues so that they can be assigned to members accordingly. Another feature that our team took advantage of was “pull request”. It allowed us to request a review on the changes made in the local branches before it is merged into the main branch. Our team found this extremely helpful when we had to keep the updates of the code organized in case we had to go back to previous versions when an issue occurred. Lastly, this tool offers a space where we can keep track of documentation and other references.

For the compiler, the team used VisualCode by Microsoft to develop the javascript project. The project in VisualCode was set up into three main folders, which are frontend, backend, and Cassandra (the database folder). For the front end, we used React Native and Expo. React Native is a framework that builds a hierarchy of UI components to build the JavaScript code. It has a set of components for both iOS and Android platforms to build a mobile application with a native look and feel. Expo is a framework that is used to build React Native apps. It is basically a bundle with tools and services built for React Native, that will help us get started with building React Native apps with ease. Expo was mainly used to have the application open up on our phone through the Expo Go app. Expo also helped with giving us the ability to use the expo-barcode-scanner, which provides a React component that renders a viewfinder for the device's camera (either front or back) and will scan barcodes that show up in the frame. Also, we used expo-auth-session, which incorporates web browser based authentication to the app and works with auth0 authentication API.

Another important npm library that was used for the frontend was Axios. This is a promise-based HTTP client for the browser and Node.js. The purpose for this was to close the gap between the front end and the back end using a simple Axios request. In other words, it connects our client-side React application to our server-side Express application.

For the backend, we used Node and Express. Node (or more formally Node.js) is an open-source, cross-platform runtime environment that allows developers to create all kinds of server-side tools and applications in JavaScript. The runtime is intended for use outside of a

browser context (i.e. running directly on a computer or server OS). Express is the most popular Node web framework that allows developers to create and maintain servers. It is used to write server-side logic in Javascript for web and mobile applications.

For the database, we use NoSQL. Specifically, Cassandra is an open-source, distributed, and wide-column store designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. We run Cassandra on Docker. Docker is a set of the platform as service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels.

To help us with the login aspect of the application, we used Auth0, which is an Identity Service Provider. Auth0 is a solution to add authentication and authorization services to our applications. This helped us avoid the cost, time, and risk that comes with building our own solution to authenticate and authorize users.

Lastly, for APIS, we used two different ones. One of the APIs was from Nutrionix and it gave access to food item information such as GTIN/UPC number, ingredients, serving size etc. The other API we used was the Google Maps API, which was used for the location part. This Google API was managed and accessed with Google Cloud Platform

1.7. Users

This application is intended to be used for all users who are interested in their health. It will meet the requirements needed for users to be the healthiest they can be by tracking their nutrition and knowing where to buy wholesome foods

1.7.1. Access level

All users will have access to the same privileges at the moment. The only one that of course would be limited is the Admin user. We do not have a subscription system or any sort of payment system in place so as long as users are able to sign into their accounts, they will have access to all available features. Depending on the user, they will have a different login account that gives them access to different data as listed below:

User Access Level		
Action	User	Admin
View Scans	Allowed	Denied
View locations	Allowed	Denied
View folders/saved items	Allowed	Denied
login-Auth0	Allowed	Allowed

Figure - Table of User Access Level

1.8. Motivation

While there were many options for ideas to choose from, we decided to go with the food scanning app idea because it felt like something that - if done properly - could be beneficial to us in our everyday lives. Another reason that this project motivated us was due to the fact that we knew we would become better programmers. The skills that this project required would help us build our resume. Finally, this project was fun and that motivated us to challenge ourselves to accomplish this task. Not only that, we felt that if we were the ones to make the app, developing it would make the experience that much more enjoyable.It was exciting to see the application come to life and to see it work after so many hours on fixing bugs.

1.9. Timeline

Milestone	Tasks	Reporting	Hrs	Date
1 - Analysis				
1.1	Analysis and design stage, gather data and create system mockup	Detailed proposal	15	3/29/21
1.2	Diagrams include Context Level Diagram, Data Flow Diagram and Child Level Diagram	Deliverable 2	10	4/2/21

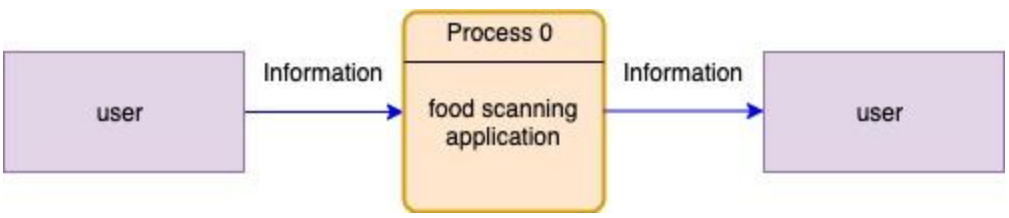
1.3	Design a data model, table schema, and a ER diagram.	Deliverable 2	10	4/2/21
2 - Development				
2.1	Design layout of app	None	2	4/6/21
2.2	Feature # 1: scanning	None	10	4/10/21
2.3	Feature # 2: location	None	15	4/17/21
2.4	Feature # 3 login/favorites	None	15	4/24/21
~~2.5~~	~~Feature # 4 food diary/calorie tracker~~	~~None~~	~~20~~	~~5/1/21~~
~~2.6~~	~~Feature # 5: followers~~	~~None~~	~~10~~	~~5/8/21~~
3 - Testing				
3.1	Alpha testing mobile application (Closed)	None	15	5/8/21
3.2	Adjust for improvements	None	5	5/9/21
3.3	Presentation	Powerpoint	5	5/10/21
3.4	Technical Report	Final Deliverable	5	5/17/21

2. Analysis of the System

2.1. Activity List

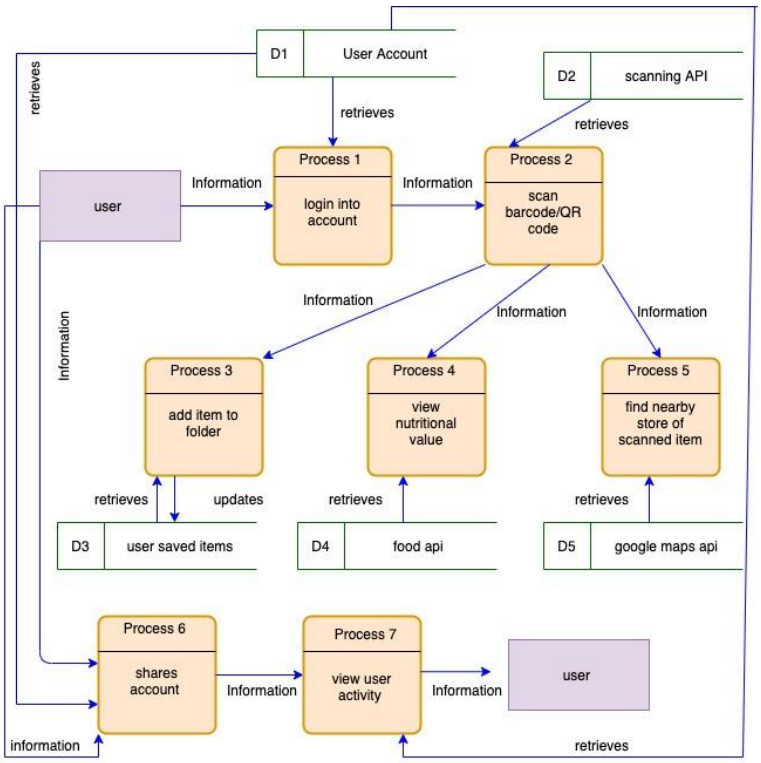
1. Login into account with a pre-existing account associated with Google
2. Scan QR code of any given food package
3. View the nutrition value of the scanned food item
4. View the nearest store that carries that food item

2.2. Context Diagram



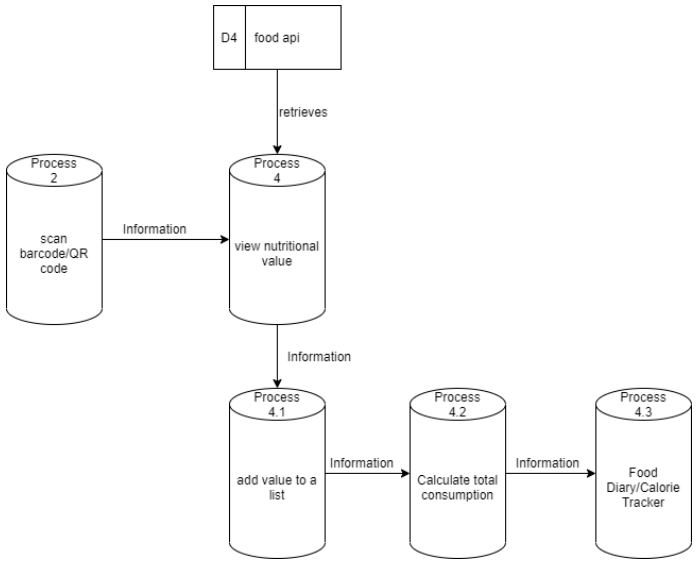
2.3. Data Flow Diagram

Process 6,7 are not included in the app.



2.4. Child Data Flow Diagram

Not able to implement this feature due to time constraints.



3. Database Design

The project was designed using a no SQL database. We choose Cassandra to set up the table schema. This set up helped us achieve a new skill that allows us to understand something other than SQL. It also allowed us to work on unstructured data, such as images for the scanned food item.

3.1. Table Schema

(no calorie tracker entity)

User (Table)

UserId: ID
FavoriteFolderNames: [{ID[],String}]
Followers: User[{ID, state}]
Following: User[{ID, state}]
CalorieTracker: CalorieTracker[]
State: Int
(TBD - data from source)

Food (Table)

FoodId: ID
TTL: Long
(TBD - data cached from source)

CalorieTracker (Entity)

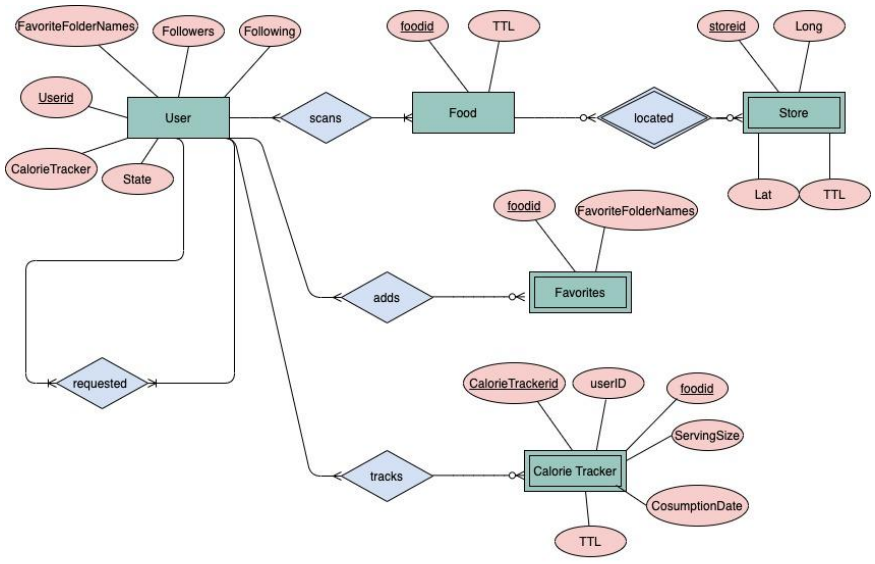
CalorieTrackerId: ID
Userid: ID
FoodId: ID
ServingSize: Float
ConsumptionDate: Datetime
TTL: Long

Store (Table)

StoreId: ID
Long: Long
Lat: Long
TTL: Long
Items: Food[]
(TBD - data cached from source)

3.2. Entity Relationship Diagram

(calorie tracker not implemented)



KEY

4. Functionality and Implementation

This project was designed using react native, which was mainly used to create a cross-platform application. The login was done with auth0. With this tool, we quickly added a solution to add authentication and authorization services to our applications. This helped us avoid the cost, time, and risk that comes with building our own solution to authenticate and authorize users. The user can login with their pre-made account through Google. This login personlized their account on our application by transferring their name and icon picture from Google to our application. Logging in gives them the capability to store their scanned history and come back to it when needed. For the location, we use Google Maps Place API, which gives us access to the latitude, longitude, and store availability of the store that contains that given item. This API made it quick and easy to implement a direction that the user can easily copy to their clipboard or click on it to open up their maps to direct them. Finally, for scanning we used a framework provided by Expo to access the camera in order to scan the barcode. This was then further integrated with axios to actually get a request to the data that was in Cassandra. That data was saved from the Nutrionix API . For example, the scanning would make a call to the database, where it would find the item based on the match GTIN/UPC number. Then it will send the nutritional value and photo.

4.1. Features

Login:

- An authentication mechanism for users of the application. This will allow for a secure and customized experience for users.
- These features will support social login for Google users.

Location:

- This is a core functionality of our application. Users will be able to find stores in their area that sell the item they are looking for.
- Location information will be based on the GPS coordinates.

Scanning:

- Another core feature: users will be able to scan the barcode of an item and view its nutritional information with it’s universal GTIN/UPC number.
- Device and operating system capabilities will determine what users can access this feature.

- The scanned items will be available for viewing after the fact and the app will hold onto this information for 30 days.

4.2. File Structure

4.2.1. UI

4.2.1.1. Code Snippet

```
const MainTabScreen = () => (
    <Tab.Navigator
      initialRouteName="Home"
      activeColor="#fff"
      barStyle={{ backgroundColor: '#000000' }}
    >
      <Tab.Screen
        name="Home"
        component={HomeStackScreen}
        options={{
          tabBarLabel: 'Home',
          tabBarColor: '#009387',
          tabBarIcon: ({ color }) => (
            <Icon name="ios-home" color={color} size={26} />
          ),
        }}
      />
      <Tab.Screen
        name="Scan"
        component={DetailsStackScreen}
        options={{
          tabBarLabel: 'Scan',
          tabBarColor: '#1f65ff',
```

```
          tabBarIcon: ({ color }) => (
            <Icon name="scan-circle-outline" color={color} size={26} />
          ),
        }}
      />
    </Tab.Navigator>
);

export default MainTabScreen;

const HomeStackScreen = ({navigation}) => (
    <HomeStack.Navigator screenOptions = {{
      headerStyle: {
        backgroundColor: '#000000'
      },
      headerTintColor: '#fff',
      headerTitleStyle: {
        fontWeight:'bold',
      }
    }}>
      <HomeStack.Screen name="Home" component={HomeScreen} options = {{
        title: 'Home',
        headerLeft: () => (
          <Icon.Button name= "ios-menu" size={25}
          backgroundColor= '#000000' onPress={() =>
navigation.openDrawer()}></Icon.Button>
        )
      }} />
```

```
    </HomeStack.Navigator>
  );


  const DetailsStackScreen = ({navigation}) => (
    <DetailsStack.Navigator screenOptions = {{
      headerStyle: {
        backgroundColor: '#1f65ff'
      },
      headerTintColor: '#fff',
      headerTitleStyle: {
        fontWeight:'bold'
      }
    }}>
      <DetailsStack.Screen name="Scan" component={ScanScreen} options = {{
         headerLeft: () => (
          <Icon.Button name= "ios-menu" size={25}
          backgroundColor= '#000000' onPress={() =>
navigation.openDrawer()}></Icon.Button>
        )
      }} />
    </DetailsStack.Navigator>
  );
```

4.2.1.1.1. Code Explanation

The above code snippet is from the MainTabScreen.js file which is located in the screens folder of the frontend project. This portion of the code is responsible for creating both the bottom navigation bar that we see on the bottom of the app as well as the tabs that it holds. Tabs can be added and removed as needed but it would also require the creation of its counterpart StackScreen. One the bottom, we've created two StackNavigators - HomeStackScreen and

DetailsStackScreen. The first is for the Home tab that users will land on after signing up and the Details tab (aka Scan tab) is our main feature where users can go to Scan. These StackScreens also contain the styling for the tabs and their respective icons. The icons were primarily retrieved from ionicons and the style we used for the page was a chic black and white theme to match the logo that we had created.

4.2.2. DB

4.2.2.1. DB - Code Snippet

```
const cassandra = require('cassandra-driver')

global.db = new cassandra.Client({
  contactPoints: ['foodscanningdb'],
  localDataCenter: 'datacenter1',
  keyspace: ['foodscanning'],
  credentials: { username: "cassandra", password: "cassandra" },
})

module.exports = function initCassandra(delay) {
  console.log(`Waiting, ${delay} seconds to allow Cassandra container to complete setup`)
  setTimeout(function () {
    db.connect()
      .then(() => {
        console.log("Connected to Cassandra.")
      })
      .catch(e => {
      console.log(e)
      console.log("Could not connect to Cassandra, please make sure the database is running and restart the server.")
    })
  }, delay * 1000)
}
```

```
const getUserProfile = async (username) => {
  if (!!username) {
    const query = `SELECT * FROM users WHERE username = ?`

    try {
      const results = await db.execute(query, [ username ], {prepare: true})

      return results.first()
    } catch (e) {
      console.log(e)
    }
  }

  return false
}
const createUserProfile = async (user) => {
  if (!!user) {
    const query = `INSERT INTO users
                   (username, name, picture, state, timestamp) VALUES
                   (:username, :name, :picture, :state, :timestamp)`
    try {
      await db.execute(query, {...user}, {prepare: true})

      return true
    } catch (e) {
      console.log(e)
      return false
    }
  }

  return false
}
```

```
const getUserHistory = async (username) => {
  if (!!username) {
    const query = `SELECT history FROM users WHERE username = ?`

    try {
      const results = await db.execute(query, [ username ], {prepare: true})
      let history = results.first().history

      if (!!history) {
        for (let [index, foodId] of history.entries()) {
          foodId = foodId.toString()
          let food = await getFoodFromCassandra(foodId)

          // console.log(food)
          if (!food) {
            food = await getFoodFromNutritionix(foodId)
            // console.log(food)
            if (!!food) {
              saveFoodToCassandra(food)
            }
          }
          history[index] = food
        }
      }

      return history
    } catch (e) {
      console.log(e)
    }
  }

  return false
}
```

```
const addToUserHistory = async (username, foodId) => {
  if (!!username || !!foodId) {
    let history = await getUserHistory(username)

    let query = `UPDATE users SET history = history + ? WHERE username = ?`

    try {
      let results = await db.execute(query, [ [foodId], username ], {prepare: true})

      if (results) {
        // 60 * 60 * 24 * 30
        query = `UPDATE users USING TTL ${60 * 10} SET history[${!!history ? history.length : 0}] = ? WHERE username = ?`
        await db.execute(query, [ foodId, username ], {prepare: true})
      }

      history = await getUserHistory(username)
      return history
    } catch (e) {
      console.log(e)
    }
  }

  return false
}
```

```
const getFoodFromCassandra = async (foodId) => {
  const query = `SELECT * FROM foods WHERE "foodId" = ?`

  try {
    const results = await db.execute(query, [ foodId ], {prepare: true})

    return results.first()
  } catch (e) {
    console.log(e)
  }
}
```

```
const saveFoodToCassandra = async (food) => {
    if (food){
        /*Write to table*/

        const query = `INSERT INTO foods \
                       ("foodId", name, ingredients, serving_size, serving_size_unit, "labelNutrients", photo) VALUES \
                       (:foodId, :name, :ingredients, :servingSize, :servingSizeUnit, :labelNutrients, :photo) \
                       USING TTL ${60 * 60 * 24}`
        try {
          await db.execute(query, {...food}, {prepare: true})

          return true
        } catch (e) {
          console.log(e)
          return false
        }
    }
    else {
        return false
    }

}
```

4.2.2.1.1. DB - Code Explanation

The above code snippets are the functions used to manage the Cassandra database. While this code is responsible for the data the FoodScanningApp has access to, it is limited in how much logic is worked into the code. We decided to limit the logic used in the database code so that in the future if we decided we wanted to switch to a different database it would be easy to replace the old functionalities with the new ones. This modular design also helps with debugging what went wrong when we receive an error from the application.

4.2.3. Services

4.2.3.1. Controllers- Code Snippet

```
food.get('/:foodId/', async (req, res) => {
   const foodId = req.params.foodId


   if (foodId) {
       let result = await getFoodFromCassandra(foodId)


       if (result) {
           console.log(`cache hit on ${foodId}`);
           res.status(200).json(result)
           return
       }


       console.log(`cache miss on ${foodId}`);


       result = await getFoodFromNutritionix(foodId)


       if (result) {


           // res.status(200).json(result)
           saveFoodToCassandra(result)
           res.status(200).json(result)
           return
       }
       else {
           res.status(404).json({
               ErrorMessage: "Item not found"
```

```
            })

            return

        }

    }

    else {

        res.status(400).json({

            ErrorMessage: "foodId is required"

        })

        return

    }

})
```

4.2.3.1.1. Controllers- Code Explanation

The above code snippet is within the controllers folder of the backend project. This portion of the code is solely responsible for logic related to requests and responses to clients. It also provides some level of logging by showing metrics of which UPC IDs users have scanned for, as well as the percentage of cache hits and misses to the Cassandra table for this endpoint. All functionality for data retrieval is abstracted to a different folder in the code. This allows for cleaner, isolated code, and allows for functions to be more testable. This also allows for the seamless implementation of different technologies for retrieving data. Switching different technologies is as straightforward as swapping functions within this file.

4.2.3.2. Scanning- Code Snippet

```
useEffect(() => {

  // handleBarCodeScanned();

  (async () => {

    try {

      const { status } = await BarCodeScanner.requestPermissionsAsync();

      setHasPermission(status === 'granted');

    } catch (error) {
```

```
      alert('Cannot access camera.')
      setHasPermission(false)
    }
  })();
}, []);
const handleBarCodeScanned = (barcode) => {
  setScanned(true);
  // console.log(barcode.data)
  axios.get(`http://${api}/food/${barcode.data}`, {
    headers: {
      'Authorization': `Bearer ${token}`
    }
  })
    .then(response =>{
      const allFoods= response.data;
      // console.log(allFoods)
      setFoods(allFoods);
    })
    .catch((error) => {
      console.log(error)
  })
}
if (hasPermission === null) {
  return <Text>Requesting for camera permission</Text>;
}
if (hasPermission === false) {
  return <Text>No access to camera</Text>;
}
```

```
return (
  <View
    style={{
      flex: 1,
      flexDirection: "column",
      justifyContent: "flex-end"
    }}
  >
    {scanned ? (
      <View
        style={{
          flex: 1,
          marginTop: 15
        }}
      >
        <DetailsScreen foods={foods}/>
        <Button title="Tap to Scan Again" onPress={() => setScanned(false)}/>
      </View>
    ) : (
      <BarCodeScanner
        onBarCodeScanned={handleBarCodeScanned}
        style={StyleSheet.absoluteFillObject}
      />
    )}

  </View>
);
}
```

4.2.3.2.1. Scanning- Code Explanation

The above code uses an import package called expo-barcode-scanner. This package first requests permission from the camera before being able to scan. This part of the code gives a pop up for the user to give access first, which is vital for their security. Once it scans, it does an axios call to the backend to get it for the appropriate GTIN/UPC barcode number. It gets the data and sets it as prop to be used in another page where the data shows up. After it scans, the user can click to scan again .

5. Testing

5.1. Objectives

This initial release focuses on the deployment of the minimum viable product, which consists of three features. These include the food scanner, store location, and login features.

5.2. Requirements

Acceptance criteria for each feature are defined in terms of expected behavior. This behavior-driven approach ensures that development, testing, and user-acceptance aligns with the best experience for the customer.

Food Scanner

As a user, I would like to scan the barcode of an item and receive the nutrition information about that item.What if the service is unable to return the nutrition information? The user should be informed appropriately of any errors

Store Location

As a user, I would like to know where I can buy a food scanned food item in my area.

- What if the item is not available in my area? The user should be notified that there are no stores nearby.

Login

As a user, I should be authenticated in order to use the Food Scanning Application.

Can I use the application without logging in? No, all transactions must happen after a user is authenticated

5.3. Test Cases

Food Scanner

Happy Path - Scan foods - iOS

Happy Path - Scan foods - Android

Happy Path - Return nutritional information - Services

Happy Path - Caching of returned information - Services / DB

Location

Happy Path - return a list of stores in an area - Services

Negative Path - return error when gps information is not provided - Services

Login

Happy Path - sign up by username and password

Happy Path - sign up by third party - google

Negative Path - invalid tokens return a 401 from the service layer

6. Conclusion

6.1. Limitations

There were a number of limitations that our team had to tackle in order to complete this project. The first and foremost being the late start that we unfortunately had to deal with. By no fault of the team or the professor, our group was forced to wait for about 2 weeks to hear back from the representative from NYIT in regards to the application that they originally wanted us to build. This 2-week setback put us far behind the rest of the teams and we essentially had to play

a game of catch up all while having to juggle work from our other classes and for some members, work from their full-time positions as well.

Other limitations include the fact that some of the core functionalities of our system - nutrition retrieval and store locations - all depend on 3rd parties. This limitation hinders our testing with these systems. In order to stay below rate limits (free tier) and in service agreements, automated functional tests and performance tests with services are limited. Testing with 3rd parties has been limited to manual regression when changes are made to its functionality. Not only that, when using 3rd party systems like the nutrition retrieval, we were oftentimes met with food items that had information that was missing whether that be the image of the food, certain nutritional values, etc. There isn't much we can do to remedy this at the moment because all of these values are being provided by NutritionIX and there's no way that we can alter their source code. The final limitation is the fact that in order to support a large number of users and requests, a free tier would not be sufficient. A monetization strategy would need to be developed at a level sufficient to support the application.

6.2. Future Enhancements/Recommendations

Some future enhancements could include implementing a settings page, food diary, follower system, and a Tier system/Subscription system. A settings page is always a requirement for any application. This contains any preference changes for the application' design and usage. A main feature that we would love to implement is the food diary. This food diary would have a huge calorie tracking limit. The user would scan the food and note it as consumed, so that it shows up in their food diary. The food diary would aggregate the nutritional value per day .This feature would go hand in hand, with the follower system. The idea for that would be to include different types of users such as a coach, trainer, nutritionist or even a friend that can access the user's food diary to help them keep track of their diet. When they sign up, they can add the user as a client and interact with their activity through likes and comments. The main purpose for the follower system would be to promote accountability and motivation by seeing if the user eats foods/calories out of their set goals. The last feature would be for sustainability of the application, if there would be an influx of users. This feature would be a subscription system,

where a user can pay to unlock special features. As well with more users that would mean we would have to invest in the next level of the tier system to allow more calls to the third party APIS.

6.3. Team Members

Nick Lombardi (Leader)

I have been working for NYIT College of Osteopathic Medicine as a Web Developer for about two years now. In my time working at NYIT I have developed multiple web applications using multiple technologies and programming languages including React, Node.js, Laravel, Docker, MariaDB (MYSQL), and MongoDB (NoSQL). With this experience I was able to contribute to both the backend and frontend of the FoodScanningApp. Even though I have had extensive experience with web development I was able to push myself to learn new technologies such as React Native and Cassandra to further improve my skills as a programmer.

Kevin Hunte (Co-Leader)

Since graduating undergrad in May of 2019, I have worked as a QA/SDET, mostly with platforms and services. I contributed to architecting and implementing backend systems, creating user stories, choosing 3rd party technologies, and managing the code repository. Most of my code contributions were backend related, along with assisting in implementing the client side code for authentication. I learned a lot about React.js, Express.js, and system architecture.

Fernanda Tovar

In my case, I have done a couple of websites using mainly HTML,CSS , mySQL, and PHP. Therefore, this mobile application that was done on React Native was entirely new to me. I had brief knowledge on Javascript, so it was a new skill I acquired. My main contribution to this project was to implement the barcode scanner and to integrate that with the backend to get the data associated with the food. I also helped with finding a new food API when it was discovered that the original one didn't provide a GTIN/UPC number. I also focused on setting up the template for the paper and presentation.

Akash Ravindran

I've worked on a few websites before - primarily using HTML and CSS - but this was my first time working on a mobile app that's gone past one or two weeks of development. Plus, a majority of the projects that I've worked on thus far have been solo, so working with a group was

a nice change of pace and great experience for the future as well. Since React and React Native were uncharted territory for me, I was excited to be tasked with developing the frontend design of our application. I researched frontend design etiquette to see what sort of practices were accepted, along with popular color schemes. Then I got to work on creating prototypes of what our app would look like and once we settled on a design, I developed the first iteration of our app using React Native. As the scope of our project changed and different functionalities had to be either added or removed, I ended up creating multiple iterations, finally settling on the one we have right now. I also assisted with the final report.

6.3.1. Learning Experience and Outcome

Nick Lombardi

Throughout my time working on this project I was able to learn two new technologies that I have been interested in working with for a long time. The first new technology was React Native, a open-source mobile application framework created by Facebook. This was an amazing learning experience as I have had projects from work that I have wanted to implement a mobile application in tandem with its web application and now I have the background experience to do this. The second technology was CassandraDB, an open-source NoSQL database used by multiple large tech companies such as Netflix, Spotify and Reddit. This was more of a personal interest to learn this database since I don't have any large scale applications that would need CassandraDB, however I always like to learn the technologies that are used by big tech so that I'm able to constantly develop projects at the highest level.

Kevin Hunte

I learned a lot about architecture, development, and version control. I am also very grateful to experience what it was like to take a senior role on a project. I have previously used many tools within the .NET ecosystem, such as Visual Studio, C#, Azure Cloud, and Team Foundation Version Control. Using tools such as Express, Cassandra, Google Cloud Platform, Auth0, and React Native were a great learning experience. From working in Quality Assurance, I often have tested and viewed systems from the outside, and wrote black-box tests. Getting to design, implement, and test a product from a developer point of view was a great experience. I now know that I enjoy development more than quality assurance, and am looking to stay on that path moving forward.

Fernanda Tovar

My learning experience was great due to my team members. Kevin and Nick were great teachers and helpers. Learning all these new tools, technologies and language was very

overwhelming, but they helped me when I ran into problems. I learn a lot of new skills that I can add to my resume. I have now experienced noSQL, which was something I had on my list. I also got more comfortable with Javascript and the frameworks that come with it. It was also challenging trying to finish the project since we had little time to learn and then implement a whole application. Overall, I learned the general steps to create a cross-platform application, which has overall helped me to improve my programming skills.

Akash Ravindran

I learned a great deal from this project. I came in not knowing a whole lot about the technologies that we were going to be implementing in this project but Nick and Kevin were very patient mentors so I did my best to absorb as much knowledge from them as I could. I ended up learning about React, React Native, Expo CLI, Docker, Auth0, and a few other tools that I'll now be able to utilize in my future projects. Nick also showed me my new favorite code editor - Visual Studio Code- solely because it has git integrated into its system already and it makes my life that much easier since I was always a bit nervous using git. I wish we had some more time to work on this project because the learning curve for me was pretty steep and there's a lot more that I would've liked to have experimented with but all in all, I ended up learning a lot about full-stack development and I plan on using all of it post-graduation.

7. Reference

https://github.com/kevhunte/FoodScanningApp

https://github.com/kevhunte/FoodScanningApp/wiki

https://github.com/kevhunte/FoodScanningApp/wiki/Backend-Docs

https://github.com/kevhunte/FoodScanningApp/wiki/Database

https://github.com/kevhunte/FoodScanningApp/wiki/Frontend-Docs

https://github.com/kevhunte/FoodScanningApp/wiki/Backend-Docs

https://docs.expo.io/versions/latest/sdk/bar-code-scanner/

https://docs.expo.io/versions/latest/sdk/permissions/

https://stackoverflow.com/questions/59865290/not-returning-jsx-from-function-call-in-expo-app

https://medium.com/@goodpic/expo-barcode-scanner-with-react-navigation-aafc2556e6f3

https://stackoverflow.com/questions/59484149/how-to-access-camera-in-react-native-expo

https://medium.com/codespace69/react-typescript-parameter-props-implicitly-has-an-any-type-error-653f224f6b9d

https://www.youtube.com/watch?v=OT7Lprq-xPE

https://vtechguys.medium.com/expo-barcode-scanner-69b4f4122da2

https://github.com/vtechguys/medium

https://dev.to/wbali/comment/14894

https://github.com/axios/axios#example

https://sunnychopper.medium.com/how-to-use-axios-to-quickly-connect-to-an-api-in-your-react-native-application-a69c1c048f8e

https://www.digitalocean.com/community/tutorials/react-axios-react

https://stackoverflow.com/questions/58139644/axios-didnt-return-data-with-matching-query-string-of-parameter-object-but-retu

https://betterprogramming.pub/connect-your-express-and-react-applications-using-axios-c35723b6d667

https://www.positronx.io/react-axios-tutorial-make-http-get-post-requests/

https://blog.logrocket.com/how-to-make-http-requests-like-a-pro-with-axios/

https://www.npmjs.com/package/axios

https://www.youtube.com/watch?v=rpg1jOvGCtQ

https://rapidapi.com/blog/react-hooks-fetch-data-api/

https://scotch.io/courses/10-react-challenges-beginner/fetch-and-display-from-an-api

https://blog.bitsrc.io/things-you-should-know-when-fetching-data-for-react-components-39d61602feda

https://levelup.gitconnected.com/fetch-api-data-with-axios-and-display-it-in-a-react-app-with-hooks-3f9c8fa89e7b

https://css-tricks.com/using-data-in-react-with-the-fetch-api-and-axios/

https://www.codegrepper.com/code-examples/javascript/how+can+we+fetch+data+from+api+in+using+axios+react

https://www.codegrepper.com/code-examples/javascript/post+axios+react+app.post+

https://javascript.plainenglish.io/what-is-axios-and-how-to-use-it-with-react-1470d19e1b83

https://rapidapi.com/blog/axios-react-api-tutorial/

https://www.youtube.com/playlist?list=PLQWFhX-gwJbmmqcP-9zMXBaxQbGKfIJY2

https://www.youtube.com/watch?v=WnS7dcY5Hys

https://shreyasnisal.medium.com/multiple-navigators-react-native-b6545c8ba16

https://www.youtube.com/watch?v=vOPf0QL2mWs

https://stackoverflow.com/questions/43313178/react-native-error-yarn-is-not-recognized-as-an-internal-or-external-command

https://reactnative.dev/docs/getting-started

https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

https://developer.apple.com/design/tips/

https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/color/

https://material.io/design/color/applying-color-to-ui.html#top-and-bottom-app-bars

https://ionic.io/ionicons

https://docs.expo.io/