

# ***Smart Grocery Coding Project Final Report***



***Prepared by  
Arijus Trakymas, Sharva Darpan Thakur, Russel Tjahjadi,  
Mohammad Zaid  
for use in CS 440  
at the  
University of Illinois Chicago***

**April 2023**

## Table of Contents

	List of Figures .....	4
	List of Tables .....	5
I	Project Description .....	6
1	Project Overview .....	6
2	Project Domain .....	6
3	Relationship to Other Documents .....	6
4	Naming Conventions and Definitions .....	6
4a	Definitions of Key Terms .....	6
4b	UML and Other Notation Used in This Document .....	7
4c	Data Dictionary for Any Included Models .....	7
II	Project Deliverables .....	8
5	First Release .....	8
6	Second Release .....	9
7	Comparison with Original Project Design Document .....	12
III	Testing .....	12
8	Items to be Tested .....	12
9	Test Specifications .....	12
10	Test Results .....	14
11	Regression Testing .....	15
IV	Inspection .....	15
12	Items to be Inspected .....	15
13	Inspection Procedures .....	16
14	Inspection Results .....	17
V	Recommendations and Conclusions .....	18
VI	Project Issues .....	18
15	Open Issues .....	18

16	Waiting Room.....	18
17	Ideas for Solutions.....	19
18	Project Retrospective.....	19
VII	Glossary .....	20
VIII	References / Bibliography .....	20
IX	Index .....	20

## List of Figures

No table of figures entries found.

## List of Tables

No table of figures entries found.

# **I Project Description**

## **1 Project Overview**

The Smart Grocery App is an app that aims to combine the common tasks of recipe searching, grocery store searching and restaurant searching in a one-stop-shop app. These ideas are based in part on the work that was done by a previous group in the Fall of 2022 [4].

## **2 Project Domain**

The domain for this project is food, restaurants, and grocery stores. These concepts are already familiar to most people, however there are additional details that relate to these domain areas that need to be defined.

In the food (“recipes”) domain area we reference the idea of preferences. Preferences are not normal preferences in the sense that the user can switch the app from light mode to dark mode but rather adjust the preferences of the food results they receive based on three different categories. The first category is cuisine, which just refers to a particular type of cuisine such as American, British, etc. The second category is diet which just refers to a specific diet type such as gluten free or vegetarian. Lastly, the third category is intolerances which can be thought of as allergies, examples include egg, dairy, etc.

In the places (“restaurants” and “grocery stores”) domain area, we refer to the idea of location. For the grocery stores, we have the name of the store, distance from the user’s current location, status (if the store is open or closed) and finally, the rating of the stores. In addition to grocery stores, we also have restaurants, which display the name of the restaurant, how relatively expensive it is (displayed by the ‘\$’ signs), the ratings and the description of the restaurant.

## **3 Relationship to Other Documents**

This document is related to the development project report that was prepared by another group in the Fall of 2022 [4]. It contains references to material covered in that report.

## **4 Naming Conventions and Definitions**

### **4a Definitions of Key Terms**

Recipe: A unique recipe found online that consists of a title, image, and an identifier.

API: Application programming interface. This is how we obtain our data ranging from recipes, nearby grocery stores, and restaurants.

API endpoint: A route within an API that allows developers to either send or receive data from.

API key: A key used in the app that authenticates requests that our project uses to handle requests made for the API services.

SDK: Software development kit. A set of tools for developers to get started quickly to build products using other software.

Return: A return statement that ends the execution of a function and returns a value to the calling function. API endpoints typically return a value, for example a recipe.

Spoonacular API: An API provided by company spoonacular that gives us access to recipes, restaurants, and other food related data.

Google Places API: An API provided by Google that takes in location data and returns data about places nearby such as grocery stores for example.

Firebase: Application development software built by Google that enables developers to build web apps.

VS code: Visual Studio Code. A code editor developed by Microsoft.

Flutter: A open-source user interface development kit.

Dart: A programming language that is used in Flutter.

URI: A uniform resource indicator that enables identification of obtaining a resource from the web.

URL: A uniform resource locator. Specifies the location of a resource on the Internet.

#### **4b UML and Other Notation Used in This Document**

This document makes use of the standards laid out in Version 2.0 OMG UML [5]. The references and bibliography section follows the IEEE citation style.

#### **4c Data Dictionary for Any Included Models**

Recipe ID = identifier returned by the Spoonacular API.

Recipe title = title returned by the Spoonacular API.

Recipe image URL = image URL returned by the Spoonacular API.

Recipe likes = aggregate value returned by the Spoonacular API.

Recipe source URL = recipe original source URL returned by the Spoonacular API.

Restaurant ID = identifier returned by the Spoonacular API.

Restaurant name = name returned by the Spoonacular API.

Restaurant description = description returned by the Spoonacular API.

Restaurant rating = rating value returned by the Spoonacular API.

Restaurant dollar signs = dollar signs value returned by the Spoonacular API \* '\$'.

Grocery store title = title returned by the Google Places API.

Grocery store opening hours = open/closed status returned by the Google Places API.

Grocery store distance = distance in miles returned by the Google Places API.

Grocery store image URL = image URL returned by the Google Places API.

Grocery store rating = rating returned by the Google Places API.

## **II Project Deliverables**

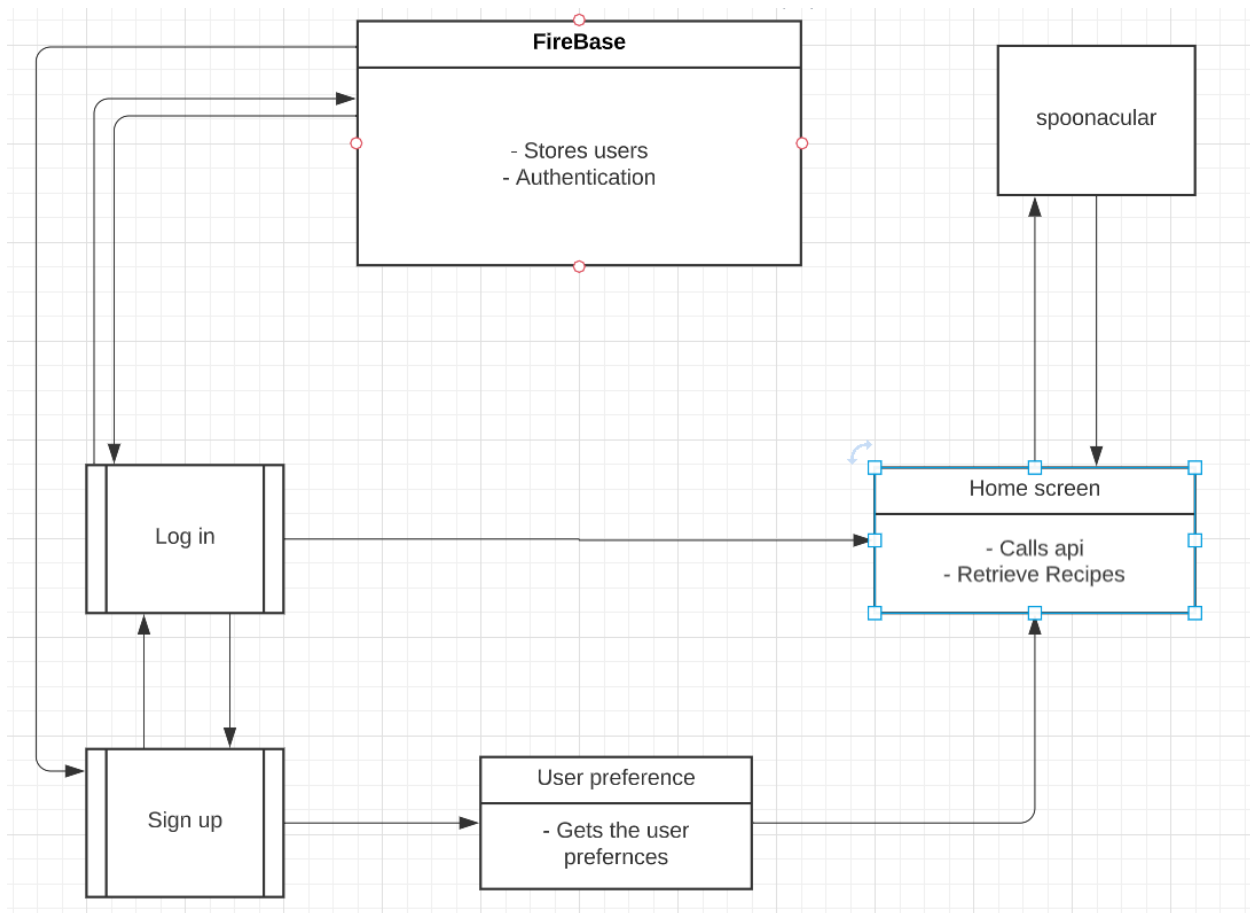
Our team has implemented the features which include getting recipes based on user preferences, restaurants, and grocery stores nearby, login screen, signup screen and a profile screen.

### **5 First Release**

The first release of the project was released on 24<sup>th</sup> of February. This release consisted of a login screen which had authentication built into it, the signup screen also has authentication built which checks if the account exists and if the password is strong enough.

There is a preference screen which lets the user input their preferences and their allergies. The Home screen uses a spoonacular API to retrieve the recipes according to the user preferences.

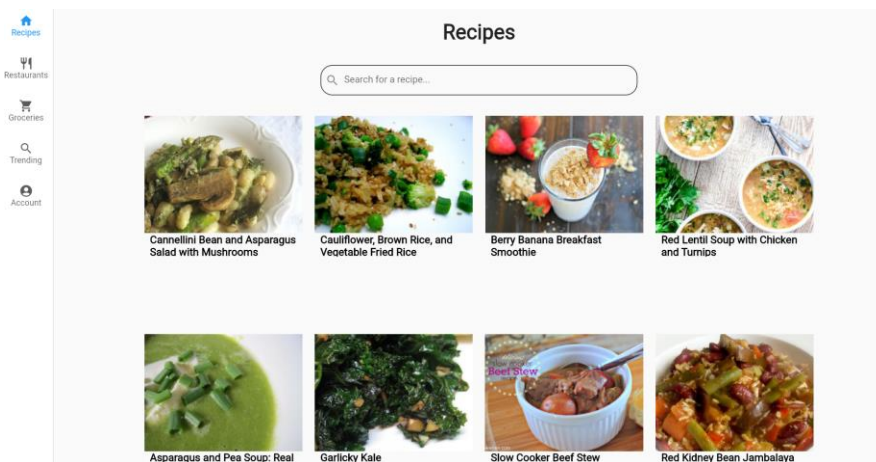
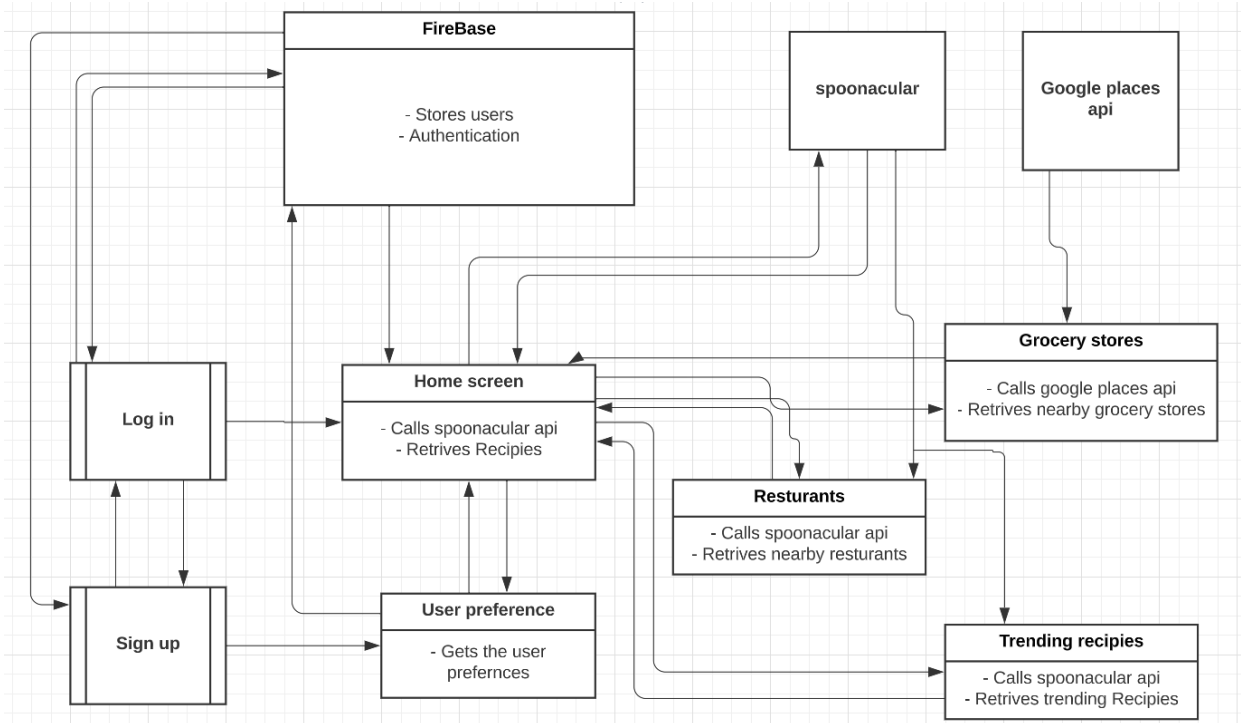








## 6 Second Release









The second release was released on 31<sup>st</sup> March which added onto the first scenario. The new things that were added were that you can now change your preferences from the home screen. There is a navigation option on each of the screens which now lets you go from any one screen to another. The new screens were a grocery screen which shows you the nearby grocery stores using the Google Places API and the user's location. The restaurant screen also uses the location and spoonacular API to get the nearby restaurants.

Another feature that was added was a search button on the home screen which allows users to search recipes while being filtered according to their preferences. The trending page displays recipes which are currently popular so that the user can be open to new recipes.



★	GoldCoast Social No rating...	Price:\$\$\$\$\$
★	Georgia's Pizza No rating...	Price:\$\$\$\$\$
★	Gotham Bagels Gold Coast No rating...	Price:\$\$\$\$\$
★	Edwardos Natural Pizza No rating...	Price:\$\$\$\$\$
★	Sparrow No rating...	Price:\$\$\$\$\$
★	Eduardos Enoteca 4.0	Price:\$\$\$
★	Wingstop 4.6	Price:\$

	<p>Name: Trader Joe's</p> <p>Distance: 1.25 mi mi Status: Open Rating: 4.6</p>
	<p>Name: Whole Foods Market</p> <p>Distance: 2.00 mi mi Status: Open Rating: 4.5</p>
	<p>Name: Potash Markets - State St</p> <p>Distance: 0.65 mi mi Status: Open Rating: 4.2</p>
	<p>Name: Bookwinkle's</p> <p>Distance: 2.06 mi mi</p>

			
Grandma B's Rhubarb Cake	Risotto With Fresh Peas	Palak-tofu (bean curd)	Easy Slow Cooker Artichoke Garlic Chicken
			
Brown Rice Vegetable Pulao	Wild Blackberry Sorbet With Garden Mint & Lavender	Goat Cheese and Fig Crostini	Grilled Ham and Cheese French Toast For A Quick Weeknight Dinner

## 7 Comparison with Original Project Design Document

Our prototype of the application is very similar to the application which was described by the previous group. We took the ideas for login screen, signup screen and the home screen exactly as the other group described and intended it to be. The preferences and the trending page were an idea we came up with. The restaurants page was an inspiration from the grocery store page which they had described.

## III Testing

### 8 Items to be Tested

ID #1: Get Recipes

ID #2: Get Random Recipes

ID #3: Get Restaurants

ID #4: Get Grocery Stores

### 9 Test Specifications

#### 1# - Get recipes

**Description:** Ensure there are recipes getting returned by the spoonacular API.

**Items covered by this test:** Recipes filtered by preference parameters. ID #1

**Requirements addressed by this test:** Recipe (5) [1]

**Environmental needs:** Flutter SDK.

**Intercase Dependencies:** NA.

**Test Procedures:**

Run the following command in the terminal:

```
dart test .\test\api_test.dart -n "Spoonacular API get recipes endpoint"
```

**Input Specification:** Whether the API call to get recipes has an empty list of recipes.

**Output Specifications:** A Boolean value.

**Pass/Fail Criteria:** The API call returns a non-empty list of recipes.

#### 2# - Get random recipes

**Description:** Ensure there are random recipes getting returned by the spoonacular API

**Items covered by this test:** Random Recipes. ID #2

**Requirements addressed by this test:** Recipe (5) [1]

**Environmental needs:** Flutter SDK.

**Intercase Dependencies:** NA.

**Test Procedures:**

Run the following command in the terminal:

```
dart test .\test\api_test.dart -n "Spoonacular API get random recipes endpoint"
```

**Input Specification:** Whether the API call to get random recipes has an empty list of recipes.

**Output Specifications:** A Boolean value.

**Pass/Fail Criteria:** The API call returns a non-empty list of recipes.

### **3# - Get restaurants**

**Description:** Ensure there are restaurants getting returned by the spoonacular API.

**Items covered by this test:** Restaurants. ID #3

**Requirements addressed by this test:** Not applicable.

**Environmental needs:** Flutter SDK.

**Intercase Dependencies:** NA.

**Test Procedures:**

Run the following command in the terminal:

```
dart test .\test\api_test.dart -n "Spoonacular API get restaurants endpoint"
```

**Input Specification:** Whether the API call to get restaurants has an empty list of restaurants.

**Output Specifications:** A Boolean value.

**Pass/Fail Criteria:** The API call returns a non-empty list of restaurants.

#### **4# - Get grocery stores**

**Description:** Ensure there are grocery stores getting returned by the Google Places API.

**Items covered by this test:** Grocery stores. ID #4

**Requirements addressed by this test:** Not applicable.

**Environmental needs:** Flutter SDK. Google API key.

**Intercase Dependencies:** NA.

**Test Procedures:**

Open the project in VS code. Open the tests tab. Run the grocery stores test dart file.

**Input Specification:** The status of the Google Places API call to get nearby grocery stores.

**Output Specifications:** A string value.

**Pass/Fail Criteria:** The API call returns an OK status code.

### **10 Test Results**

#### **1# - Get recipes**

**Date(s) of Execution:** 4/1/2023

**Staff conducting tests:** Mohammad Zaid

**Expected Results:** True

**Actual Results:** True. The spoonacular API successfully returned a list of recipes.

**Test Status:** Pass

#### **2# - Get random recipes**

**Date(s) of Execution:** 4/16/2023

**Staff conducting tests:** Sharva Darpan Thakur

**Expected Results:** True

**Actual Results:** True. The API successfully returned a non-empty list of random recipes.

**Test Status:** Pass

### **3# - Get restaurants**

**Date(s) of Execution:** 4/17/2023

**Staff conducting tests:** Arijus Trakymas

**Expected Results:** True

**Actual Results:** True. Provided with location coordinates, the API successfully returns a non-empty list of restaurants.

**Test Status:** Pass

### **4# - Get grocery stores**

**Date(s) of Execution:** 4/19/2023

**Staff conducting tests:** Russel Tjahjadi

**Expected Results:** String value - "OK"

**Actual Results:** "OK" status received. Provided with specific location coordinates, the API successfully returns a non-empty list of nearby grocery stores.

**Test Status:** Pass

## **11 Regression Testing**

These tests might become necessary to re-run these tests if the API endpoints get updated in the future with new parameters. If there are any services that are built on top of the existing API calls already, then the tests will become necessary to run again since they will be layered.

## **IV Inspection**

### **12 Items to be Inspected**

ID #1: Get Recipes

- Creator: Arijus Trakymas
- Inspectors: Mohammad Zaid, Russel Tjahjadi, Shava Thakur

ID #2: Get Random Recipes

- Creator: Mohammad Zaid
- Inspectors: Russel Tjahjadi, Shava Thakur, Arijus Trakymas

ID #3: Get Restaurants

- Creator: Russel Tjahjadi
- Inspectors: Shava Thakur, Mohammad Zaid, Arijus Trakymas

ID #4: Get Grocery Stores

- Creator: Shava Thakur
- Inspectors: Mohammad Zaid, Arijus Trakymas, Russel Tjahjadi

## 13 Inspection Procedures

- ID #1: Get Recipes: The inspector will run numerous calls to `getRecipes()` function from `APIService.dart` file. The inspectors need to test and make sure the return list is not empty and contains the recipe list in a readable format.

Meeting timings: Saturdays 4-6 pm

Discussion consisted of API endpoints the creator used to render the results and ways to enhance the search to get better results. The results were discussed during the group meeting to ensure completeness and bug-free development.

- ID #2: Get Random Recipes: For this inspection, we want to make sure that each recipe changes every time and consider the user entered preference parameters like Intolerances, Dietary Restrictions, etc.

Meeting timings: Saturdays 4-6 pm

Discussion consisted of API endpoints used, and use of different randomness factors to get the results. The results were discussed in group meetings where the majority of the inspection took place.

- ID #3: Get Restaurants: The inspection procedure for this feature takes in input the location coordinates and requires the user to location services on their respective Browser where the application is running. The inspector needs to test the API returns an “OK” status code when provided with working coordinates.



Meeting timings: Saturdays 4-6 pm

The meetings took place in-person and remote. The majority of the inspection was accomplished outside the meeting.

- ID #4: Get Grocery Stores. This feature also takes in the current user location as input and provides the user with a list of nearby grocery stores. The inspector again needs to test to ensure the return of “OK” status code and non-empty list of grocery stores.

Meeting timings: Saturdays 4-6 pm

The meeting took place over zoom and all the inspection was accomplished over the meeting including various ideas storing them in the waiting room.

## **14 Inspection Results**

ID #1: Get Recipes

Inspectors: Mohammad Zaid, Russel Tjahjadi, Shava Thakur

Inspection Times: Saturdays 4-5 pm, Wednesdays 6-8pm

Result: We found out that after multiple trial runs that the application stops running abnormally in some runs. After debugging we found that the returned list contained some recipes without any image URL and the code was trying to access that image due to which sometimes the code comes to be in a halting state. This was easily fixed, and the code was now running at all trials.

ID #2: Get Random Recipes

Inspectors: Russel Tjahjadi, Shava Thakur, Arijus Trakymas

Inspection Times: Saturdays 4-5 pm, Wednesdays 6-8pm

Result: A bug was found where the function was returning an empty list when the user reloads the page. Inspectors found this bug in the api.dart file and fixed it.

ID #3: Get Restaurants

Inspectors: Shava Thakur, Mohammad Zaid, Arijus Trakymas

Inspection Times: Saturdays 4-5 pm, Wednesdays 6-8pm

Result: Inspectors found this code to be working bug free.

ID #4: Get Grocery Stores

Inspectors: Mohammad Zaid, Arijus Trakymas, Russel Tjahjadi

Inspection Times: Saturdays 4-5 pm, Wednesdays 6-8pm

Result: Inspectors found this code to be working bug free.

## **V Recommendations and Conclusions**

As of now, the items covered have passed their testing and inspection process. In the future, we aim to test more as we possibly might add more features in the app. The next step is to implement more features like obtaining user input

## **VI Project Issues**

### **15 Open Issues**

One problem that we have faced is not having enough quotas to run the API services that we have obtained online. We found that if we need more, we need to pay more to have unlimited access to the API server requests and unfortunately, our team is insufficient of funds.

Another open issue is to find a way to advertise the app so that it can attract more users. As of right now, we do not have a marketing team that is willing to promote our app and hence, make the app only known to a limited number of people.

Due to the limited technology, we may encounter constraints in fulfilling user requests since the app can only handle a limited number of requests different users can make.

When our app is in production, we also might worry if our app will comply with the rules and regulations. We would need to ensure compliance with the various regulations that will avoid legal issues and as a result maintain the user's trust.

### **16 Waiting Room**

Implement user feedback: through user feedback, we can clearly know better on how the app works for the users. By taking their feedback, we should be able to make improvements and make desired changes to the app that will enhance the user experience.

Integration with other apps: we believe that we can implement more features by integrating other services that other apps can potentially provide us.

Improve data security: with the numerous data breaches that happen online, we want to ensure that the data that the users have put in our app is safe and secure on the app.

Customer support: we want to personally assist our customers in using the app and that is why we are offering customer support to ensure users can understand how to use the app properly.

Expanding app availability: as the product continues to grow, we aim to make this app more available to more regions because they might need it. This not only helps users currently residing in the region but also helps travelling users discover what recipes/restaurants are available in that region.

## **17 Ideas for Solutions**

This section is intentionally left blank.

## **18 Project Retrospective**

As we progressed through the project, we realized that implementing an API was not as simple as we thought. We needed to learn more which API suits best for the app that we are trying to build. With a lot of APIs available on the internet, we had a hard time finding which API would give the best result for the app because all of the APIs that we searched were unique and not every API gave the same services. In the end, we figured that the Spoonacular API was a good place to start researching more into because of its full documentation that it had.

Since the spoonacular has some missing services, we decided to utilize the Google Places API which offers services to our app and that improves our features in our app. One of their services was to display nearby grocery stores. It gave us the name of the store, the distance from the user's current location, it showed us if the stores are open or closed and the rating of the stores.

Most of us have never used an API before and we needed more time and effort to understand better how it works. In addition to that, we had to really think if the project was worth the test run because of limited quotas.

We started researching the Dart language and how to use the packages and eventually with team meetings and discussions on how to use the programming language Flutter offers a package called the Material App and from then on, we decided to expand from there. Since Dart is a more object-oriented programming language, we also decided to research more about it and eventually, we think that it is a good programming language

because of its efficiency, and it can carry over the functionalities of other files to another file.

However, with the multiple obstacles that we faced, we managed to turn in the project on time and show it to our client. We believe that with our strong teamwork, determination, and persistence, we have accomplished our goals that are listed in our project scenarios.

## **VII Glossary**

No terms.

## **VIII References / Bibliography**

- [1] F. Agnes, N. Aditya, N. Gabriel and M. Patel, "Smart Grocery Project Report," Chicago, 2022.**
- [2] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.**
- [3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.**
- [4] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.**
- [5] Robertson and Robertson, Mastering the Requirements Process.**

## **IX Index**

No index entries found.