Final Report

Date: 25th October 2019
Prepared by **Group 11**

Jinvara Vesvijak	#470500387
Lakshay Anand	#480525952
Yetong Wang	#480161509
Bibarys Amirzhanov	#480577403
Pooja Devaraja Murthy	#480459174

Table of Contents

1. The Application

1.1 The Problem

Cooking food, for many, is a hard and complex task. Millions of people today who don't know how to cook or what to cook, consider it a hassle to cook. Combine that with the limited ingredients they have in their inventory or their kitchen, it becomes a hard task to make a decent nutritious meal which also tastes good, and many resort to eating out which turns out to be unhealthy as well as expensive.

1.2 The Solution

Considering the above problem, we have come up with a solution that will fulfill each and every need of the people (the users of this app) called "Chef InProgress". The android app named "Chef InProgress" is a food recipe recommender application that considers the existing kitchen inventory of the user while also tracking the nutritional intake of the user based on the dish he/she consumed. "Chef InProgress" will follow a freemium service model and will be available to order on Google Play Store for free and is also planned for release in Spring 2021. The Chef InProgress would allow users to create their profile using their email, ID, and would allow them to add the ingredients manually which they already have at the moment or automatically by scanning the grocery store receipt. The app will then recommend the users dishes and their recipes based on the inventory, and would also mention the nutritional values per average serve size. After making the dish, once the user presses the "Done" button, the nutritional data is recorded and added to their in-app calendar.

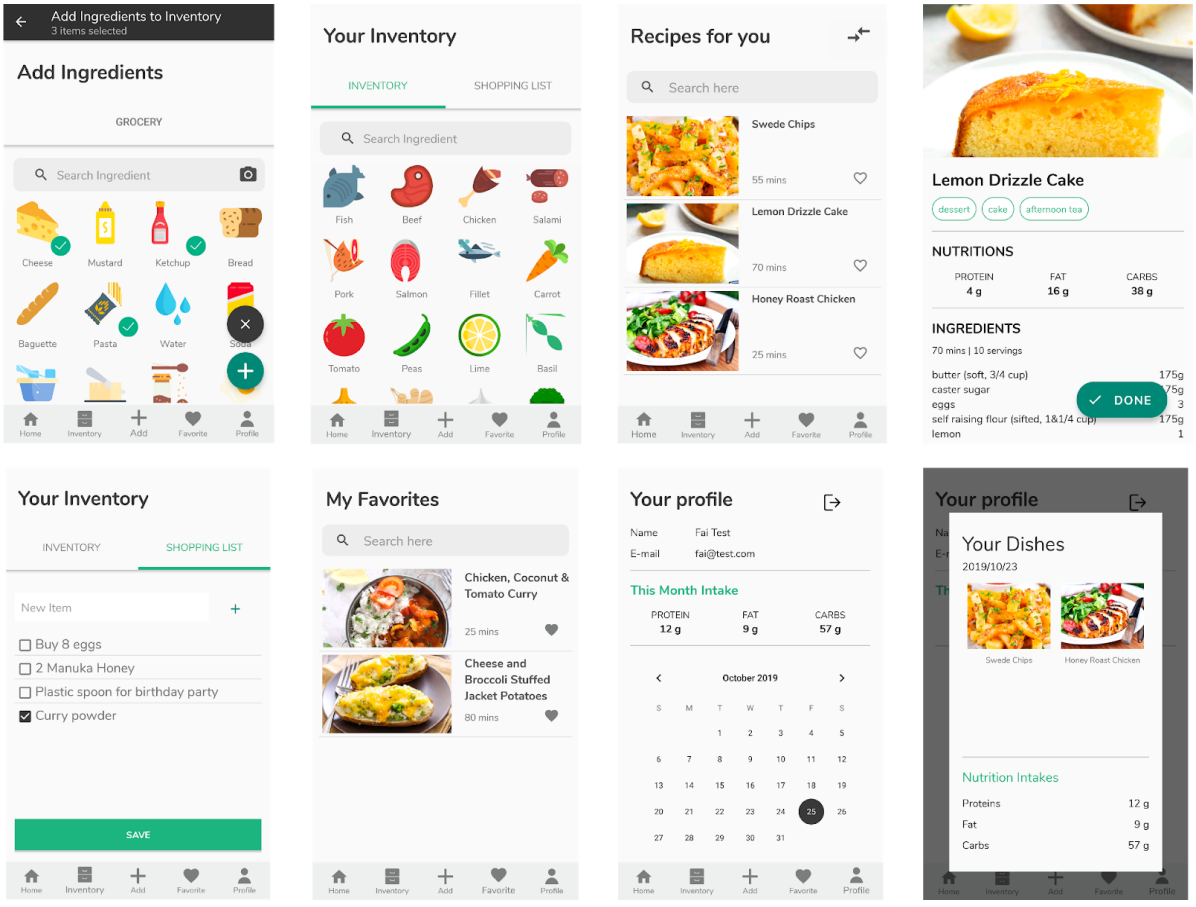

Figure 1.1 Overall views of Chef InProgress

2. The Workflow of the Application

The overall workflow of the application starting from when the application is launched is as shown below in Figures 2.1, 2.2, and 2.3. Several user cases and business logic are being taken into consideration in developing the application workflow.

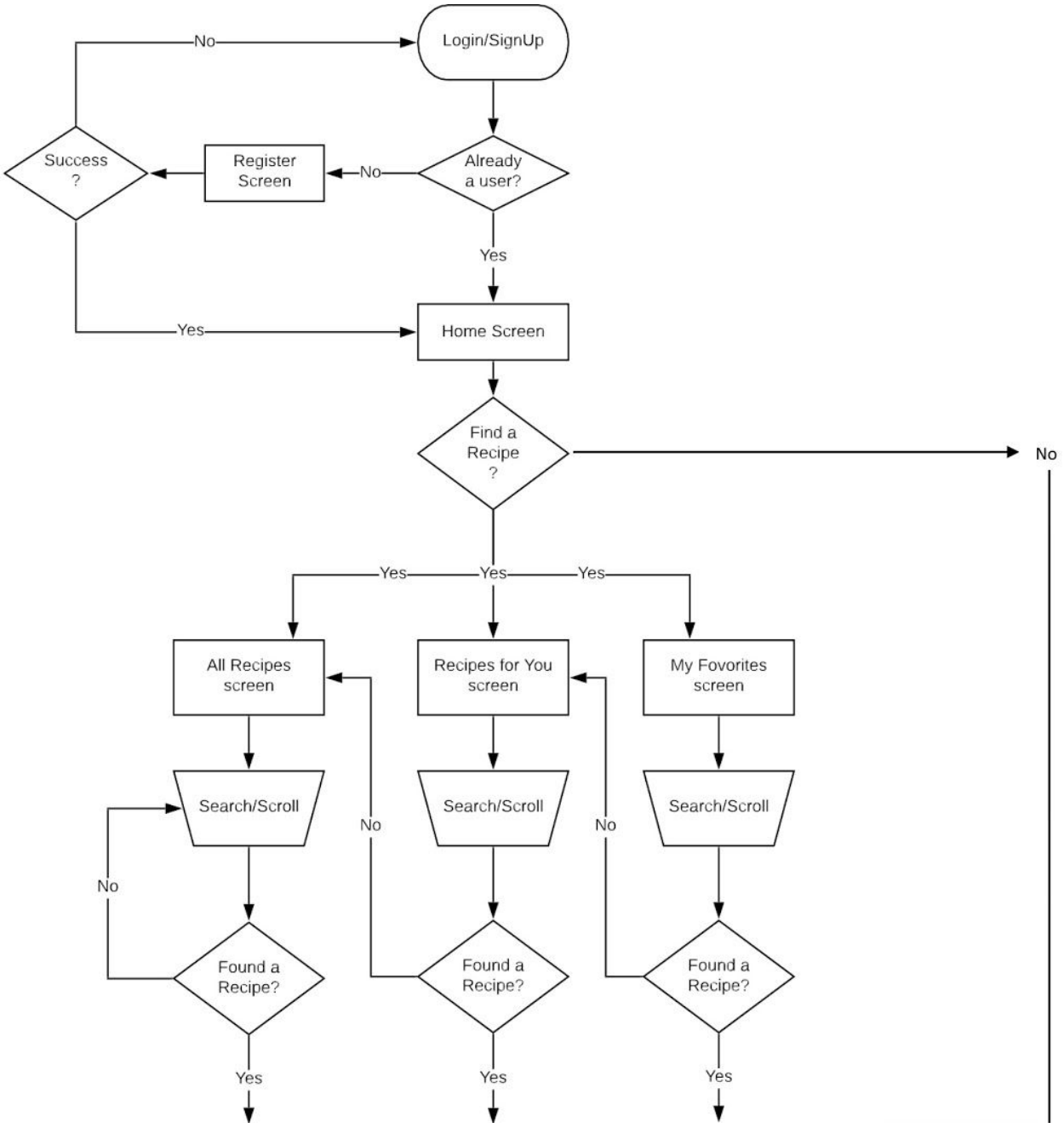

Figure 2.1 Application Workflow 1

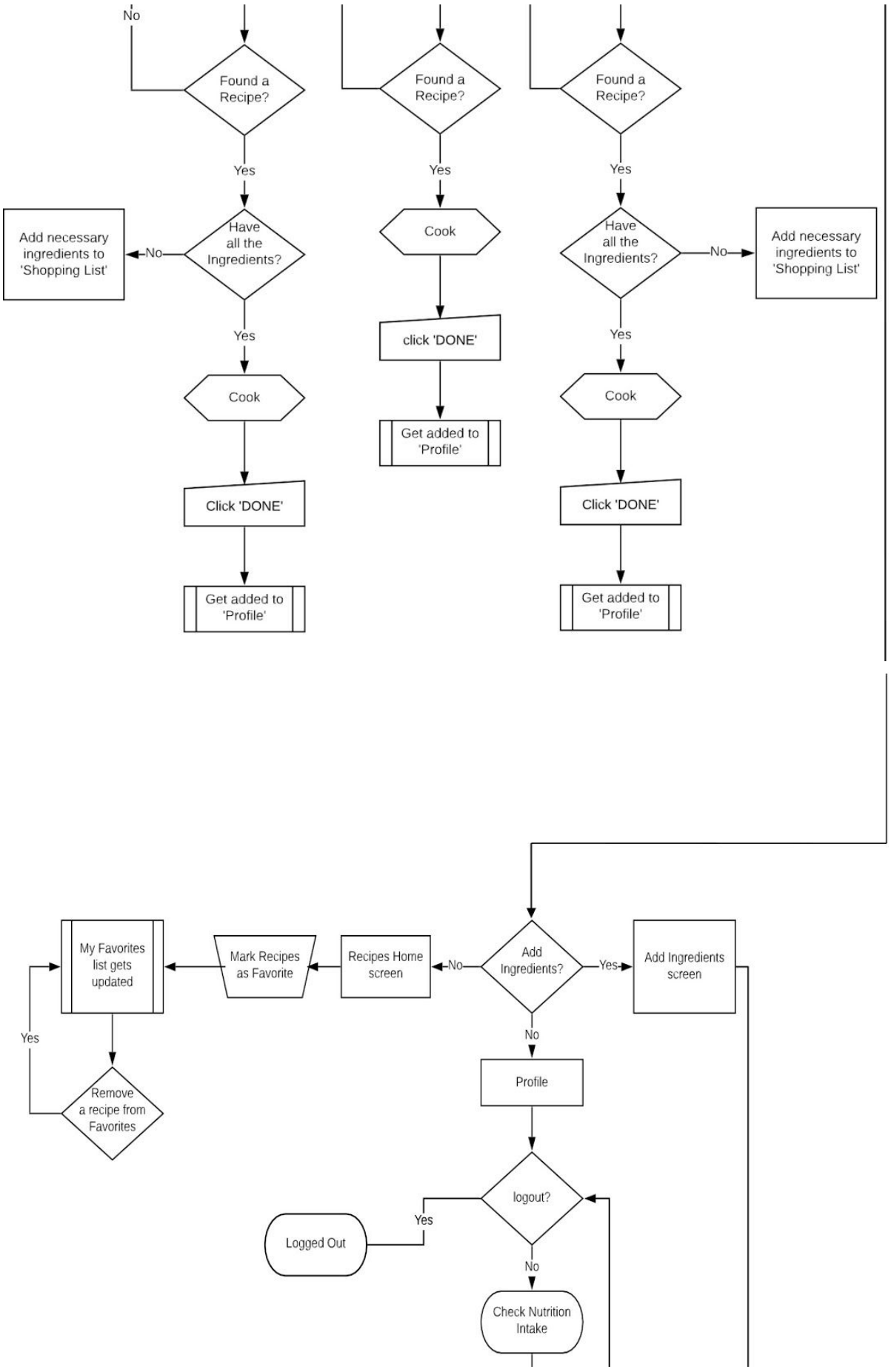

Figure 2.2 Application Workflow 2

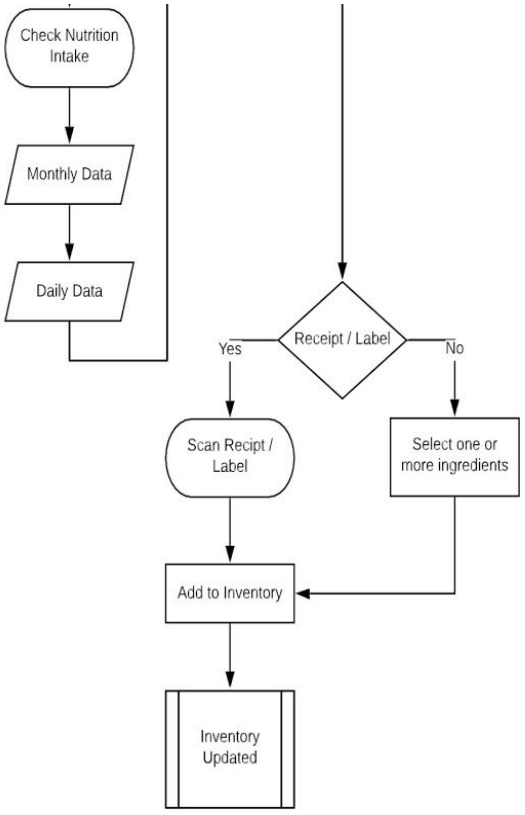

Figure 2.3 Application Workflow 3

3. Validation of the Application

3.1 User Authentication

FirebaseUI Auth provides several ways of user authentication, such as email authentication or integration with social platforms. For our application, we used email authentication. FirebaseUI Auth also allows to use of anonymous authorization, but since our application based on the user's requirements, we need to identify the user behind the phone. After authorization backend of the FirebaseUI Auth verifies the user's credentials and returns the user profile data.

3.2 Database Implementations

The applications communicate with two databases including device local database and database on Cloud.

- **Cloud database:** we use Firebase realtime database to store User and Recipe data. The recipe data is always retrieved from Firebase and requires an internet connection to successfully obtain the data. The User data is only retrieved at the time that the application is launched. Once the user data is obtained from Firebase, the logged-in user data is saved to local storage.
- **Local database**: we use SQLiteOpenHelper to manage database transactions between the application and local device storage. The application's ingredients data and logged-in user data is stored on the local device.

3.3 Google Text recognition API

We utilized the Google Mobile Vision Text API enabled by Optical Character Recognition (OCR) for implementing receipt scanner feature to obtain ingredients' text from receipts. The OCR gives the application an ability to read text from the camera view.

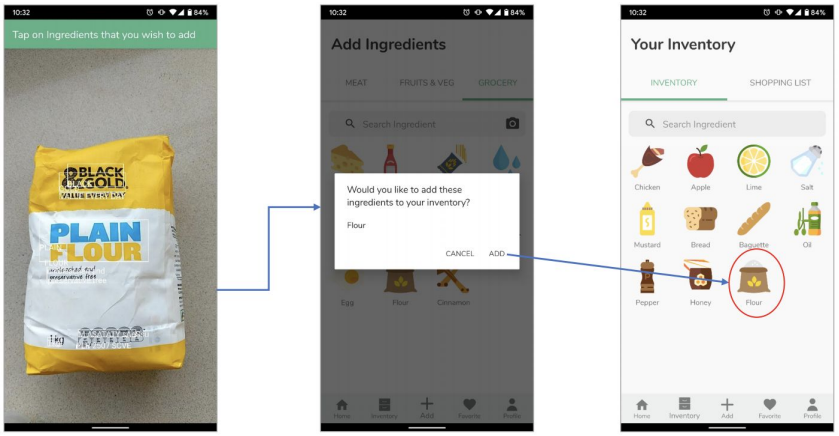

Figure 3.1 Adding ingredients using Google Mobile Vision Text API

3.4 Other Third party API

- Styleable Toast: enable styling to Toast
- Glide: helps loading image smoothly when scrolling the list view

3.5 Application optimization

3.5.1 Network Usage

As every application with online features, our application requires an Internet connection. The application uses the network for authentication and synchronization with the database.
From the database, the application retrieves User and Recipe data. Recipe data includes images for every item. To identify the data usage of the application we conducted application testing. The Application uses data in the foreground and background. We tested network usage of the application for accessing FirebaseUI Auth and Database separately. Output data of the network test shown in Table 1. Figures 3.2 and 3.3 illustrates graphs for the abovementioned testing for the Auth module and Database respectively.

		Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Avg
Auth	*Received KB/s*	296.7	3.3	2	1.9	2.6	1.8	3.6	3.5	2.7	3.9	32.2
	Sent KB/s	11.2	1.4	0.6	0.2	0.4	0.9	1.2	2.1	3.3	0.3	2.16
DB	*Received MB/s*	0.5	0.2	1.1	0.5	0.6	0.2	0.5	0.2	0.6	0.1	0.45
	Sent MB/s	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Table 1 - Application Data Usage

Figure 3.2 - Data usage for auth session

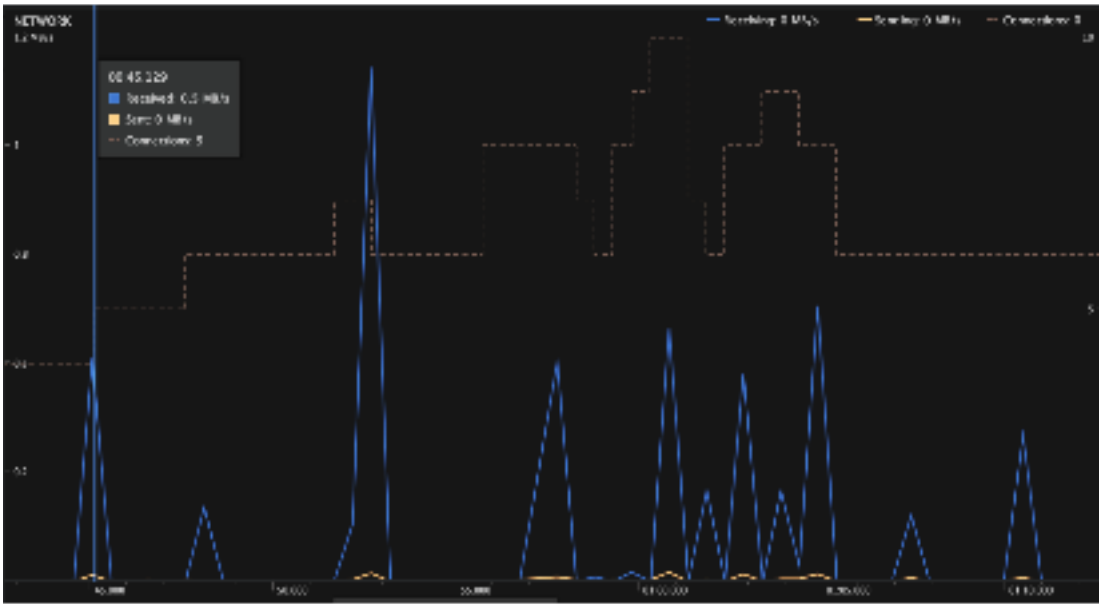

Figure 3.3 - Data usage for DB session

3.5.2 CPU and Memory

Application optimization is crucial when it comes to mobile applications. While designing our application we decided to make the UI simple, clean and match it with Material Design guidelines. By avoiding heavy UI elements and graphics we've increased speed and improved response time of the application. We have tested the CPU and Memory performance of the application with a built-in benchmark of the Android Studio.

According to the results shown in Figure 3.4 average CPU load does not exceed 10% in the main UI and about 50% while using "Camera text recognition".

The test results showed that, in any use case, the application used no more than 150 MB RAM. The memory testing process is shown in Figure 3.5.

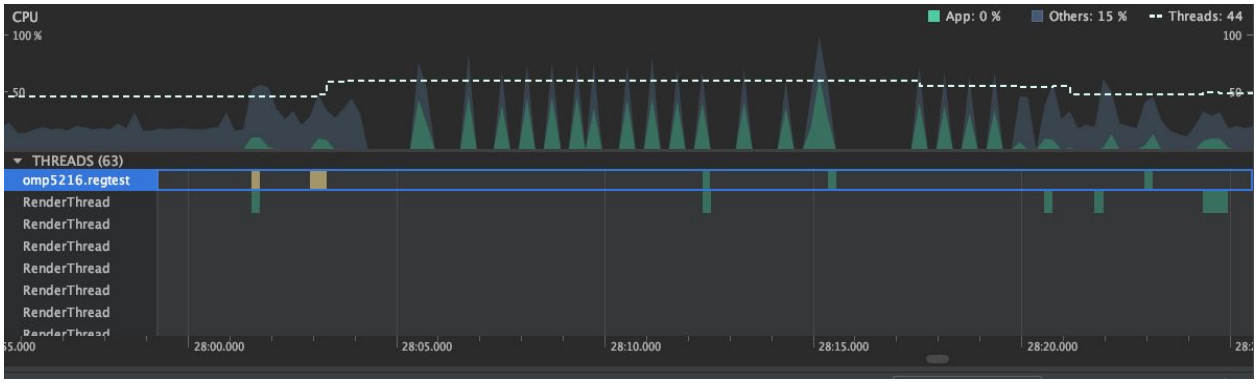

Figure 3.4 - CPU Performance

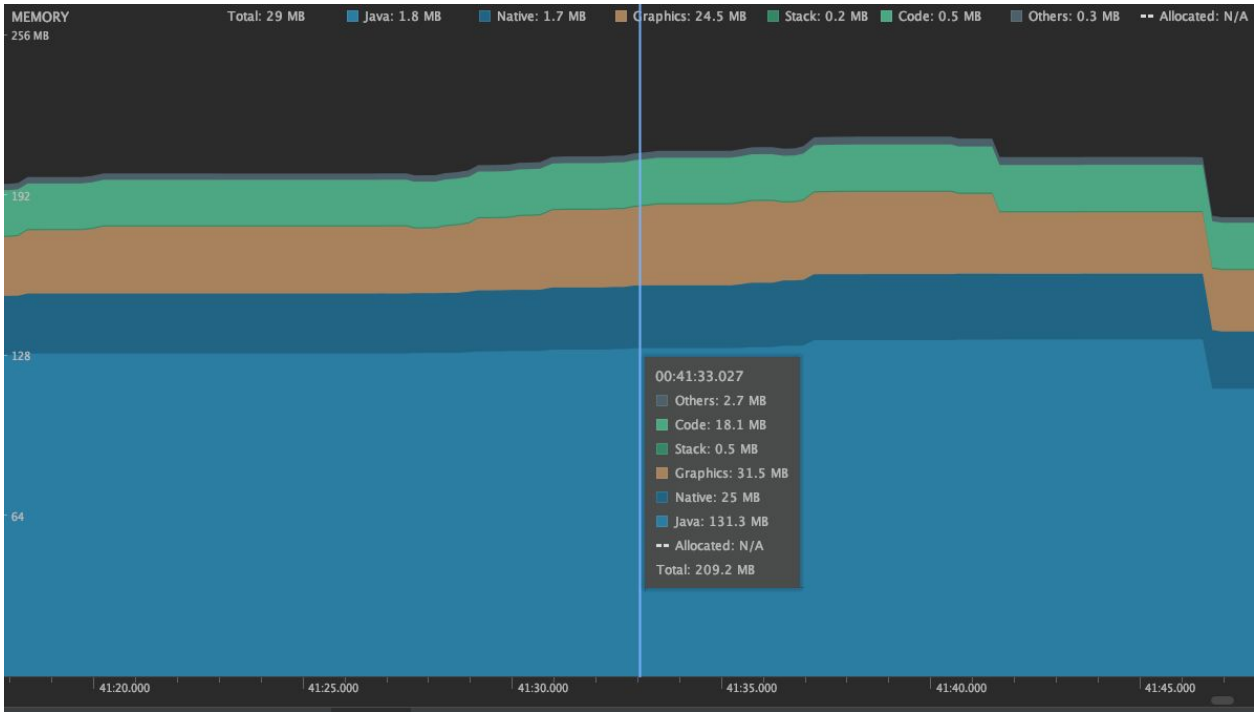

Figure 3.5 - Memory

3.5.3 Battery Optimization

Application testing demonstrated great performance in battery usage. As been shown in previous test results application decreases used memory almost twice while working in the background. Testing with sample phone demonstrated that in any use cases computed power use does not exceed 3 mAh.

4. Challenges and Setbacks

4.1 Challenges

During the development of the application, we have gone through several challenges including both technological ones and other ones such as group work and communications.

4.1.1 Technological Challenges

Control the database flow

When we developed the application, the most difficult problem is using the Firebase realtime database. The main challenge is to control the database flow within the application. In the beginning, we created the SQLite file for the local database as we started developing an application without careful planning of database access. The way SQLite saves data is saving all the data as a String type and it does not support the ArrayList of custom objects. And when we tried to link the local database to the Firebase at a later phase of the project, we realized that several columns we created in the Firebase are saved as an ArrayList such as List of ingredients or list of instruction, which has caused variable type converting issues when retrieving the data from the cloud. As a result, we have to add methods to convert the data from cloud to JSON Object and then convert it to JSON String format in order to enable the local SQLite database to read and store the data. Similarly, when uploading or updating data from local devices to the cloud, we have to convert the data type back in a similar manner, from JSON String to JSON Object and then to Array List to store the data in the correct format on Firebase, which took us a lot of time to do it and we realized that if we first planned the data communications between Firebase and SQLite well, we could have saved time in implementing the database access layer of the application.

Asynchronous Firebase

Another most challenging issues were the asynchronous characteristic of Firebase. When firebase methods are called to load recipe and user, other functions that are created to use the retrieved data from Firebase were run before the data retrieval is completed. This happened because the Asynchronous method of Firebase allows other processes to run before it finishes, causing the application to crash. We spent some time to understand this asynchronous nature of Firebase and implemented a callback function to run the functions after the data retrieval is completed.

Image Loading

Another issue that is caused by the decision to store data on the cloud is an image loading issue. The symptoms appeared in the recipe list because the image we use is from the online resources, we did not save them in the drawable directory but retrieving it as image URLs from the internet to reduce the size of the application and to make it easy for us to change and edit the images. When scrolling the list, the images of different recipes jump from one to another and does not display correctly in order. We used Glide, a third-party API, to fix the issue in a smoothly loading image for the list view.

Optimize the Performance

The most important challenge we have to face is optimizing the performance of our app. The reason why we use both local and cloud databases is this will make the performance better. For example, even though all the recipes are saved initially in the cloud users cannot access them when there is no network, they can still use the inventory and shopping list function for shopping outside without any cellular signal. And the benefits of saving all the recipes on the cloud is users can access the latest updated recipes in real-time without updating the application. If the recipes are saved locally, they have to update the application day by day to get the new recipes and the size of our application will be bigger and bigger as time goes by.

In addition to the mentioned issues, there were several mistakes and bugs that we encountered while coding which did not bother us as much as the mentioned ones, we solved them either by searching for solutions online or choose an alternative way to implement decided functionalities.

4.1.2 Non-technological Challenges

Time Management
The most challenging non-technological thing is time management. This project is group work and sometimes it's hard to arrange the schedule for everyone because everyone in the group has different assignments for other units and works maybe, so we have to make the best usage of every meeting. At the same time, some parts of our project have dependencies on each other and the integration requires everyone in the group to finish individual work on time.

Communication
Another challenge for the non-technological part is communication, we made some functionalities that are not proposed because of the lack of communication, which cost us more time than we expected to finish the whole project because, for those parts, we have to recreate them from the beginning.

Well-organized Plan
We realized that planning the project well before actually developing is as crucial as the process of development itself, because planning well can help to save time in dealing with unnecessary problems. For example, the database problem, if we did enough research and realized that the Firebase wouldn't compatible with SQLite local database well, we could have created JSON ones and all the converting methods are not needed. But in reality, either we recreate the local database with JSON or create several converting methods will not save any time for us.
These can also be reflections for us when we develop future functions or extending our application, we should plan well first and make a well-agreed timetable and keep communicating.

4.2 Setbacks

Due to the time limitation for developing, some parts of our application are not good enough, we need to acknowledge them and try to optimize them in future steps

- We don't have enough time to make our recipes various in cooking time and tastes for users to select themselves.
- It's not open enough to only serve the users recipes from us, they should be able to upload their own recipes even though some of them are already there. Some of the users might have their secret ways to make the same cuisine better and other users can pick the one they prefer.
- We don't have much time to test the usability experience, so there might be some functions are not user-friendly.

For all the setbacks right now we have, we will focus on them in the future steps and optimization.

5. Next Steps

We have a lot planned for our app "Chef InProgress" and will roll-out slowly in the future updates as we carefully get the required usage data and as the features are ready.

5.1 Release on iOS platform

After the initial release on Google Play Store for Android, we are planning to release the app for iOS devices and the release window is planned for Spring 2021.

5.2 Addition and incorporation of other nutrients

We are planning to ad more nutrients information like Vitamins, Calcium, Zinc, etc. and incorporate in the recipes and profile information. Currently it only shows Carbs, Fat and Protein.

5.3 Implementation of Systems Recommender using Machine Learning

We are planning to implement smart systems recommender based on the user's previous choice of food and dishes to better serve and suggest the user based on their taste and nutritional needs.

5.4 Rate recipes

We will allow users to rate the recipes provided and suggested by the application based on the taste and ease to make. This will allow for a better user experience overall and the users will be able to sort dishes later using the high ratings. This will create room for a better user experience.

5.5 Recipe contribution by professional chefs and dieticians

We will later allow chefs and dieticians to publicly contribute recipes and dishes to the app. This will ensure a better user experience and validity regarding the nutrients and dishes being consumed. They will benefit from this as their name would be publicly shown under the recipe.

5.6 Implementation and improvement of UI/UX

We are planning to update the app's UI based on the usage data provided by users by using the concepts of Usability Engineering and make sure that the users average latency is as low as possible.

5.7 Implementation of an even smarter scanner to scan ingredients directly

We will later allow users to scan the ingredients directly, so they won't need to scan from a receipt or a written text. This will make the logging of inventory smoother and easier and will translate to a better user experience.

6. References

- Firebase: https://firebase.google.com/
- Android Developers: https://developer.android.com/
- Android tutorials: https://www.tutorialspoint.com/android/
- Mobile Vision Text API: https://codelabs.developers.google.com/codelabs/mobile-vision-ocr/#0
- StackOverflow: https://stackoverflow.com