FINAL PROJECT REPORT

Development of mobile application to aid consumers in interpreting nutrition label on pre-packaged food

**Student Name:** Niger AngYi Dong

**Date Submitted:** 15 May 2019

**Degree Title:** Creative Computing

**Supervisor Name:** Ronnie Peh

**Local Institution:** Singapore Institute of Management

**Student ID:** 150271176

Contents

[1. Abstract 5](#_Toc8877329)

[2. Introduction 6](#_Toc8877330)

[2.1 Project Background 6](#_Toc8877331)

[2.2 Aims 7](#_Toc8877332)

[2.3 Objectives & Justifications 7](#_Toc8877333)

[3. Literature Review 9](#_Toc8877334)

[3.1 Consumer understanding and use of nutrition labelling 9](#_Toc8877335)

[3.2 Consumer response to nutrition information 10](#_Toc8877336)

[3.3 Global Health and Wellness Report 10](#_Toc8877337)

[3.4 Similar applications 11](#_Toc8877338)

[4. Methodology 12](#_Toc8877339)

[4.1 Cowboy Methodology 12](#_Toc8877340)

[4.2 Justification for Cowboy Methodology 13](#_Toc8877341)

[4.3 Requirements 13](#_Toc8877342)

[4.4 Tools and Technologies 21](#_Toc8877343)

[5. Development Phase 1 24](#_Toc8877344)

[5.1 Initial Mockups/Wireframes 24](#_Toc8877345)

[5.2 Setting up the project 27](#_Toc8877346)

[5.3 Testing 28](#_Toc8877347)

[6. Development Phase 2 29](#_Toc8877348)

[6.1 Redesign Mockups/Wireframes 29](#_Toc8877349)

[6.3 Permissions 32](#_Toc8877350)

[6.4 Image to text function 35](#_Toc8877351)

[6.5 Testing 40](#_Toc8877352)

[7. Development phase 3 48](#_Toc8877353)

[7.1 Additional Page Mockup / Wireframes 48](#_Toc8877354)

[7.2 Image Cropper Activity 49](#_Toc8877355)

[7.3 Ingredients Database 50](#_Toc8877356)

[7.4 Fixing bugs 53](#_Toc8877357)

[7.5 Testing 54](#_Toc8877358)

[8. Results 57](#_Toc8877359)

[9. Discussion 61](#_Toc8877360)

[10. Conclusion 61](#_Toc8877361)

[11. Evaluation 62](#_Toc8877362)

[12. Appendices 63](#_Toc8877363)

[12.1 Enlarged Version of Gantt Chart 63](#_Toc8877364)

[12.2 Requirements Survey Questionnaire 64](#_Toc8877365)

[12.3 End-User Survey Questionnaire 66](#_Toc8877366)

[12.4 AndroidManifest.xml 68](#_Toc8877367)

[12.5 MainActivity.java 69](#_Toc8877368)

[12.6 TableActivity.java 76](#_Toc8877369)

[12.7 Textholder.java 78](#_Toc8877370)

[12.8 FoodArrayAdapter.java 79](#_Toc8877371)

[12.9 CSVReader.java 80](#_Toc8877372)

[12.10 activity\_main.xml 81](#_Toc8877373)

[12.11 activity\_table.xml 83](#_Toc8877374)

[12.12 list\_item.xml 85](#_Toc8877375)

[12.13 main\_menu.xml 85](#_Toc8877376)

[12.14 table\_menu.xml 86](#_Toc8877377)

[12.15 strings.xml 86](#_Toc8877378)

[12.16 styles.xml 86](#_Toc8877379)

[12.17 colors.xml 87](#_Toc8877380)

[12.18 searchable.xml 87](#_Toc8877381)

[12.19 build.gradle (Module:app) 87](#_Toc8877382)

[12.20 data.csv 88](#_Toc8877383)

[12.21 Video Link to Application Demo 93](#_Toc8877384)

[13. References 93](#_Toc8877385)

## Abstract

The trend of healthy eating is on the rise. Consumers are paying more attention to nutrition labels as they get more health conscious. However, they may not truly understand what the information portrayed on the label is about.

This project looks to provide an interpretational visual aid for consumers to translate complex data shown on nutrition label into a simplified, more understandable information.

The aims and objectives were met in the project. The mobile application was able to detect ingredients and match them to the database. A list of information regarding the uses of ingredients is then displayed. However, the feature of highlighting nutrients that was beyond the healthy range was not completed due to an unresolvable bug.

Keywords: Nutritional Label, Interpretational aid, Mobile application

## Introduction

### Project Background

Health is one of the driving factors of fundamental human behaviors. People make food purchase decisions daily and one of the primary concerns is the health impact the food brings. Product labels are designed to simplify the process of making healthy choices when selecting food. It consists of a nutritional label which displays the nutritional content and ingredients of the product at the point of purchase and reduces information search costs. This is generally available to pre-packaged food and hardly available for unpackaged or pre-prepared food. However, according to Cowburn & Stockley[[1]](#footnote-1), “nutritional labelling is likely to offer limited success as a strategy to improve the nutritional health of a population.”

Studies have shown indications that the information on the nutritional label may be misunderstood, or consumers do not understand what it means. European Heart Network (EHN)[[2]](#footnote-2) conducted a study reviewing the way consumer use nutrition labels and reported that most consumers claimed to look at nutrition labels. Nevertheless, the actual use seems to be considerably lower. It appears that consumers are able to understand certain terms but some information are too complex for them. The study has also shown that “interpretational aids may serve a very useful function in getting consumers to use this information more and in a better way.”[[3]](#footnote-3)

A mobile application can be developed to aid user in interpreting whether certain nutritional contents are higher than recommended (E.g. Sodium per 100g is higher than recommended amount) and explain what each ingredient are used for (E.g. Ammonium Hydroxide are pH Control Agents and acidulants – Control acidity and alkalinity, prevent spoilage)[[4]](#footnote-4)

### Aims

The aim of this project is to develop a mobile application which can be used as a convenient interpretational aid to simplify decision-making process for users who want to purchase healthier options on pre-packaged food.

In addition, the application is to provide information on complex ingredients so that users can feel safe when purchasing food products which they do not understand.

A user is able to download a mobile application which utilizes the camera of the phone to scan the nutrition label. During the scan, the application would interpret the label and highlight contents that are above recommended servings and would inform user what each ingredient are used for.

### Objectives & Justifications

Firstly, literature reviews on consumers’ behavior and attitude towards the use of nutritional label when purchasing pre-packaged food needs to be done. This will help identify what are the major concerns that consumers have when looking at the labels and determine how the mobile application can be developed to serve as an effective interpretational aid. Questions to be answered includes, but not limited to, “Do nutrition labels affect how consumers purchase food products?” and “What are the consumers’ reaction to unfamiliar ingredients?”

When developing features on the mobile application, the research could provide answers for questions like “After scanning of nutrition labels, does the additional information ease the decision-making process when making a healthier choice?” and “Does the information on what individual ingredients are used for provides a peace of mind when purchasing certain products?”. Furthermore, research needs to be done for which text recognition tool would be the most suitable for this project.

Secondly, it would be to study and examine similar mobile application. Similar applications may have already matured after several iterations and made improvements based on users’ feedback. This would provide insights on what users are looking for in a mobile application served to be an interpretational aid on nutrition label. Patches and updates by developers can demonstrate what users expect. With this research, a feasibility and requirements analysis would need to be conducted for the mobile application before proceeding.

The mobile application would be developed on the Android Operating system as I do not own or have access to Apple’s product and mobile application on iOS can only be developed on Apple’s Operating System.

Thirdly, it would be the design and development of the mobile application. After requirements gathering, a short survey together with guidance from CO3348 Interaction Design could help with the wireframing of the application. There may be a need to set up a remote SQL database to store relevant information for the application.

Lastly, tests would be conducted to evaluate the effectiveness of the mobile application. Two similar food products would be in place for the participants to test out the application. Participants will assess whether the application has met its intended purpose and answer questions like “Do you think this application helped you decide which product was the healthier choice?” and “Do you think the application help you understand what food components are inside the product?” The demographic for the participants would be wide as many people are conscious of living a healthy lifestyle. The most representative group would be people aged from 20 to early 30s as they are found to be more proficient in using their mobile devices.

The tests may be conducted from 1 to 5 days depending on participants schedules. Preferably, the test can be concluded in 2 to 3 days so that further iterations can be done if there is a need for improvement.

A video demonstration of the application would be uploaded on YouTube at the end of the project for examiners to evaluate and the video link will be placed in the Appendix.

## Literature Review

### 3.1 Consumer understanding and use of nutrition labelling

Before proceeding with the design and development of the application, there is need to find out whether consumers actually use the nutrition label when purchasing food and if they do use the nutrition label, do they truly understand what each nutrition element mean.

From the study published by Gill Cowburn and Lynn Stockley [[5]](#footnote-5), the authors reviewed research papers, which was about consumer understanding or use of nutrition labelling, across Europe and provided a comprehensive review on the subject matter.

They discovered that “use of nutrition labels is high but more objective measures suggest that actual use of nutrition labelling during food purchase may be much lower”. Furthermore, the study brought up an important point which was the understanding of the nutrition label depends on the purpose of the task. For example, if a consumer is looking to lose weight, the consumer will look for products that are low in sugar as the consumer understands that sugar would cause weight gain. However, if a consumer wants to avoid food that will trigger high blood pressure, the consumer must possess knowledge on what nutritional element would trigger that.

The study adds on that “interpretational aids can help consumers assess the nutrient contribution of specific foods to the overall diet”. This statement coincides with the objectives of the project and gives us confidence to proceed to design and development.

However, the researches reviewed in this paper was across Europe and some part of North America, it might not be applicable to the Singapore context, whom will be the users and testers of the application. Therefore, a survey should be conducted to test whether if the results of this research paper is applicable in Singapore.

### 3.2 Consumer response to nutrition information

Klaus Grunert and Josephine Willis started a study to review research conducted in 2003- 2006 in the EU-15 countries. This study[[6]](#footnote-6) provides an updated insight to the literature reviewed in section 3.1. The study’s results aligned with Cowburn and Stockley and it discovered that consumers prefer a simplified nutrition label and understand more about a product when the labels are simplified. The study then calls for a need to create a new format of presenting nutritional information in a real-world setting.

This study suggest that the application developed should not be cluttered with unwanted information and should be present information as simple as possible. Minimalism will have to be taken into account when it comes to application design.

### Global Health and Wellness Report

Nielsen Global Health and Wellness report[[7]](#footnote-7) provides us with insight on what consumers are thinking when it comes to purchasing food. In the report, it states that package label has an impact on how consumers purchase food but it must coincide with the consumers’ purpose. This statement further validates the findings in Section 3.1. The example given in the report was that sales potato chips with low or reduced sodium labels increased 18% because consumers may think potato chips are a salty snack, so a low sodium option would be healthier. The report also discovered a change in consumers’ diet. A trend of “eating fewer fats and sugary sweet, more fresh foods” is on the rise.

This report validates that a need for this project’s application is on the rise as consumers are becoming more health conscious and that nutrition label are gaining more attention from healthy-minded consumer.

### Similar applications

#### Open Food Facts

Open Food Facts gains information about a food product but scanning the barcode. I tested the application and it does not work every time. I suspect that is because certain barcodes are not in the application’s database. Furthermore, even if the database has information regarding the barcodes, it might not detect it.

When the barcode scanning it successful, users are directed to the results page which has 3 tabs: overview, ingredients and nutrition. The ingredients tab merely shows the ingredients but not its uses. However, it did provide a hyperlink to Wikipedia for complex ingredients. This may be a possible solution for finding ingredients’ uses but it is a backup because it requires too much action from the user and forces the user to exit the app.

#### Nutrition Info

Nutrition info allows user to manually search the food that they are looking at to gain its information. It then provides a pie chart to show the proportion each nutrient take. This app also scans barcode to obtain the product information. However, it forces the camera to go into a landscape mode. This was a terrible experience for me as I would prefer the app to maintain portrait mode in picture mode. Notes will be taken to avoid such a user experience during development phase.

#### Ingred

Ingred has a manual search function for users to find ingredients. When the ingredient is clicked on, it provides the uses of ingredients and even show a change in the icon when an ingredient is harmful. The application provides many useful information in a very minimal way. The design aspect of this application should be modelled upon.

However, the application has bad text detection. Out of a long list of ingredients, it was only able to successfully find and match 1 ingredient. It also took quite a while for this process to complete. It possibly due to large web API database it is calling or maybe internet connection. Upon seeing this, it may be better to attach the database to the application so that the application can work without internet connection and the matching process can be faster.

Screenshots of using these applications can be found in the appendix.

## Methodology

### Cowboy Methodology

For this project, I planned to adopt an agile methodology called Cowboy methodology. According to Hollar[[8]](#footnote-8), it is an agile method for solo developers and is based on four other agile methodology named Extreme Programming, SCRUM, Agile Unified Process and Getting Real. By combining the best of those practices, it allows solo programmers to focus on developing according to the requirements and deliver the project on time.

Cowboy is an incremental iterative process. Each cycle will have specific features to be added and bugs of the previous cycles to be fixed. The development process of each cycle is built on core elements from the mentioned methodologies: XP - spike solutions, coding in small portion over unfamiliar terrain; SCRUM – creating a list with features or bugs to be checked off; AUP – keep artifacts simple, ensure that core features are completed first.

As suggested by the dissertation, the length of each iteration should be no more than two weeks. In addition, the larger and more important features should be solved first before moving on to smaller issues as stated in the Getting Real Methodology and detailed comments included when 2updating codes.

Similar to many agile practice, Cowboy Methodology practices a test-driven development. Unit and functional tests should be run constantly to ensure that the current code is working and is of quality.

### Justification for Cowboy Methodology

This method would be suitable for this project as this project is to be done by a single developer and is imperative that a functional application is delivered at the date of submission. Requirements and goals are generally clear at the start before the design and development phase as well.

There are two core functions for this project: highlight nutritional contents that are above recommended servings and provide information on uses of each ingredient. Following the core elements of Getting Real, this would be the large major features that have to be tackled first.

The iteration cycle and to-do-list, essentially from SCRUM, would keep me on task when coding and build momentum. This methodology would allow me to focus on the important tasks first and help me keep track what needs to be done so that I would not diverge and waste valuable project time on useless features.

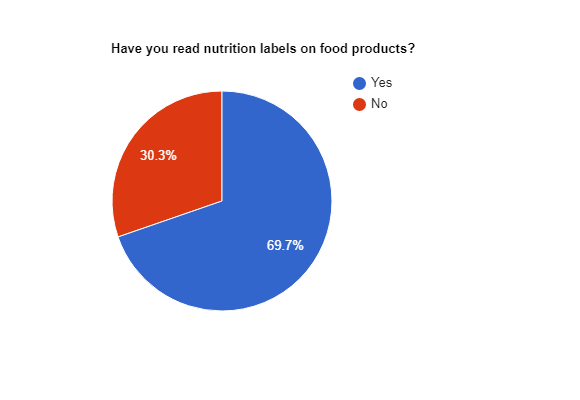
Another reason for using this methodology is because it ensures I produce quality code by constantly testing it with each iteration. However, I must ensure all requirements are clearly gathered before development begins.

### Requirements

It is important in an iterative cycle to scope out the requirements of the project before proceeding with the development phase. Data was gathered from a sample of participants aged 20 to 35. Aligning to SCRUM and AUP core elements in the Cowboy methodology, requirements analysis was conducted to generate a list of features to be completed first and identify which are the core features to be completed first before proceeding with minor, less important aspects of the development.

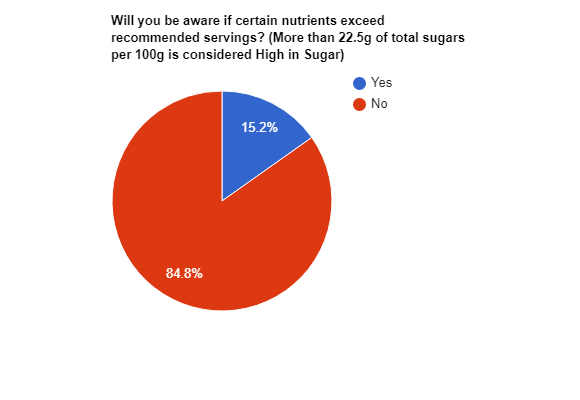
#### 4.3.1 Requirements Survey

The survey was conducted over 2 weeks during the Preliminary Project Report. It was carried using Typeform as it was free to use and readily accessible by participants across various platforms. The age of the participants ranges from the age of 20 to 35 which was in our targeted demographic. The survey was designed to check three things: the participants use and knowledge of nutrition labels; their willingness to use the application proposed to identify whether there is a need; and how they would like the application to work to get and idea of the application Design.



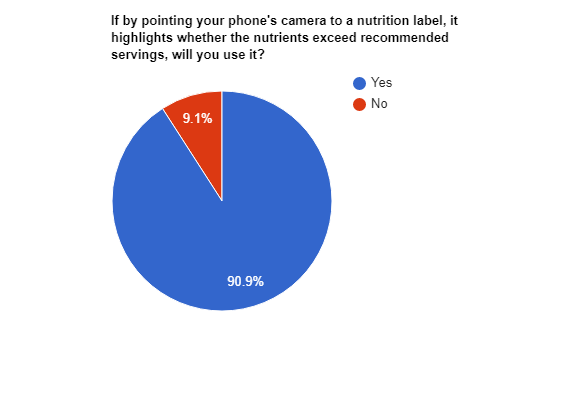
##### Figure 4.1

The first question asked the participants if they have read nutrition labels on food products before. Close to 70% of participants claim to have read nutrition labels on food products or are at least aware of its existence. The significant proportion answering “Yes” to this question suggest that a large percentage of users in this demographic, in general, may utilize nutrition labels when purchasing a food product but it is not conclusive.



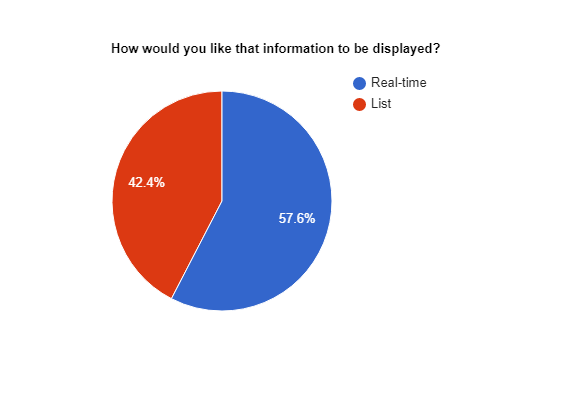
*Figure 4.2*

The second question was designed to check users’ nutritional knowledge. As stated in studies presented in the introduction, reading the labels and truly understanding what it means are separate matters. Majority of the participants answered “No” when questioned if they know the recommended servings of each nutrients. It shows that a vast majority do not properly utilize the nutrition label. This question solidifies the findings of the studies[[9]](#footnote-9) mentioned in the introduction and that the issue is applicable to Singaporean consumer as well.



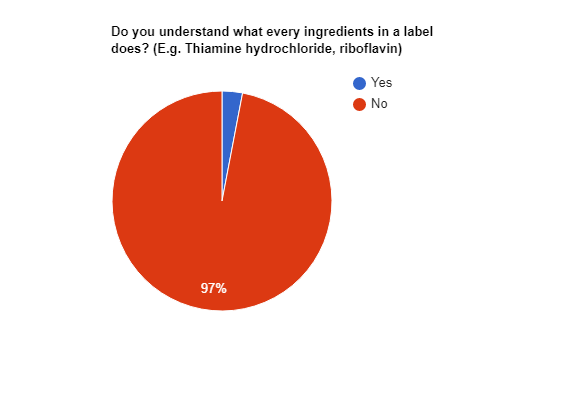
*Figure 4.3*

The third question was to survey whether there is a need for such a mobile application and if participants are willing to use it. Participants were asked if they are willing to use an application to help them understand contents of a nutrition label. Majority of the participants say they would use application with a function like this. The results of this question validated the first core feature of the project idea, which is to aid user in interpreting whether certain nutritional contents are higher or lower than recommended.



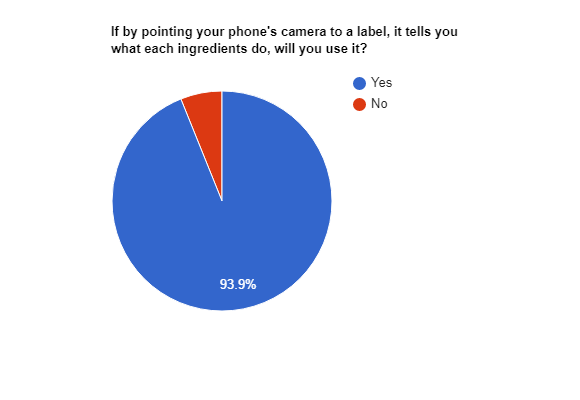
*Figure 4.4*

The fourth question was used to gather insights from the participants on how they would like the highlighted information to be displayed. This question helped to identify how users want the data to be displayed. Figure 4.4 shows that nearly 60% would prefer the information to be highlighted in real time. Despite of the results, the final design should align with User Interface Guidelines and Nielsen’s Heuristics. Inputs from the survey will be taken into consideration when it comes to interaction design of the application but should not be solely relied upon.



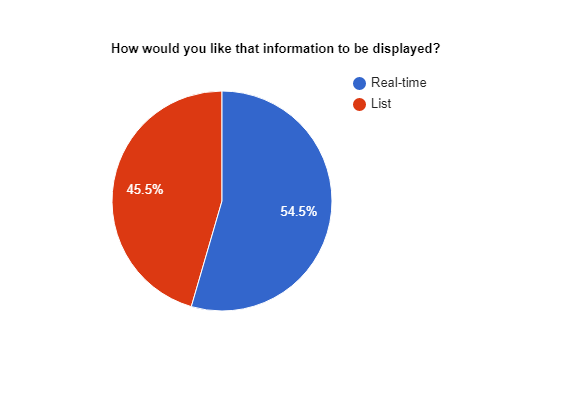
*Figure 4.5*

Similar to the second question in testing participants’ nutritional knowledge, the fifth question was designed to check whether participants understand the purpose of each ingredient, including complex chemicals, shown on the label. The results showed that 97% of participants do not understand what the functionality of every ingredient is.



*Figure 4.6*

Like the third question, this question was designed to check whether there was a need for such a mobile application and the participants’ willingness to use it. 93.9% of participants say that they would use such a function. This result validated the second core feature of the project idea, which was explaining what each ingredient are used for. This suggest that we can proceed with the development of this project with confidence.



*Figure 4.7*

The seventh question asked for the users’ opinion when it comes to displaying the information for the second core feature of the mobile application. The result of this question is almost even with 54.5% of the participants choosing real-time display. However, similar to question four, this will only be a useful consideration when designing.

To summarize, the above results show us that the project idea is feasible as majority of participants show a willingness to use the functions and proved that the results of studies conducted in Europe is also reflected in Singapore. We also get a better understanding on what users are looking for, in terms of design, and can start developing wireframes for the mobile application before commencing on actual programming. Further details of the survey questionnaire can be found in appendix 1.

### Tools and Technologies

#### 4.4.1 Initial tools and technologies

There are currently many text recognition tools available online. After reading several reviews, a website[[10]](#footnote-10) has conducted an extensive experiment with reliable controls and variables, comparing 3 of the more well-known text recognition tools namely Google Cloud Vision, Microsoft CS and AWS rekognition.

| **Provider** | **#Correct (C)** | **#Wrong (W)** | **#NoResult (N)** | **Precision (C/(C+W))** | **Recall**  **((C+W)/total)** |
| --- | --- | --- | --- | --- | --- |
| **Microsoft cognitive services** | 142 | 76 | 283 | 65% | 44% |
| **Google cloud vision** | 322 | 80 | 99 | 80% | 80% |
| **AWS rekognition** | 58 | 213 | 230 | 21% | 54% |

*Figure 4.8 Experiment results taken from https://dataturks.com/blog/compare-image-text-recognition-apis.php*

Judging from the above results, Google Cloud Vision has proven to be much more accurate compared to its competitors as it has achieved 80% precision. Moreover, it has an 80% recall rate as well. This suggest that it was able to recognize a greater number of texts compared to the other 2 provider and still retain a higher accuracy. The Vision API can detect and extract text from images[[11]](#footnote-11), which is what is required for the core feature of the application. Therefore, Google Cloud Vision would be the text recognition tool utilized for this project.

As I am a beginner in Android development, many experts recommend beginning with Android studio as the programming tool because the user interface is easy for beginners to familiarize with. Furthermore, it supports Java, which was the programming language taught by UOL and one that I am familiar in.

The initial project plan was to call a web API that would provide access to a food database that has information about different uses of each ingredients. However, there was none that was available to public or was easily accessible for students. Therefore, the workaround was to manually get data from a reliable website that provides such information. I was able to find such information from FDA’s website[[12]](#footnote-12) and insert the data into a CSV excel sheet. A snippet of the CSV file can be found in Appendix. Next, a CSV reader class would have to be created to import the CSV file into the application

The advantages of this workaround are that users would not need an internet connection to fully utilize the application and it gives a better user experience because they do not have to update frequently if there are changes to the database. On the other hand, the users are downloading the entire database directly into their phones and this may take up a significant amount of the user’s phone memory. Nonetheless, this is the only option that I can conceived of at this point of the project.

The best tool to use for such a task would be SQLite database, which was suggested in the google android developer course. It can be used to establish a database for information that needs to be reflected on the application. For example, when text detected in real-time matches items in the ingredients database, it will display the ingredients and its uses. It utilizes SQL commands which was taught in UOL’s curriculum as well.

#### 4.4.2 Additional tools

During the setup process in development phase 1, I tried to integrate the CSV file into the SQLite database with a CursorAdapter class, but it yielded no success. Instead, I have decided to feed the data into an ArrayList of String Arrays.

Furthermore, in development phase 2, the notion of a displaying information in real-time was scrapped. Text detection would be used on static images taken from the camera or the gallery. In addition, cropping images to focus on the correct body of text ensure that the information is loaded quickly and accurately. I incorporated an Android Image cropping library[[13]](#footnote-13) created by Arthur Teplitzki to solve this issue in development phase 3.

## Development Phase 1

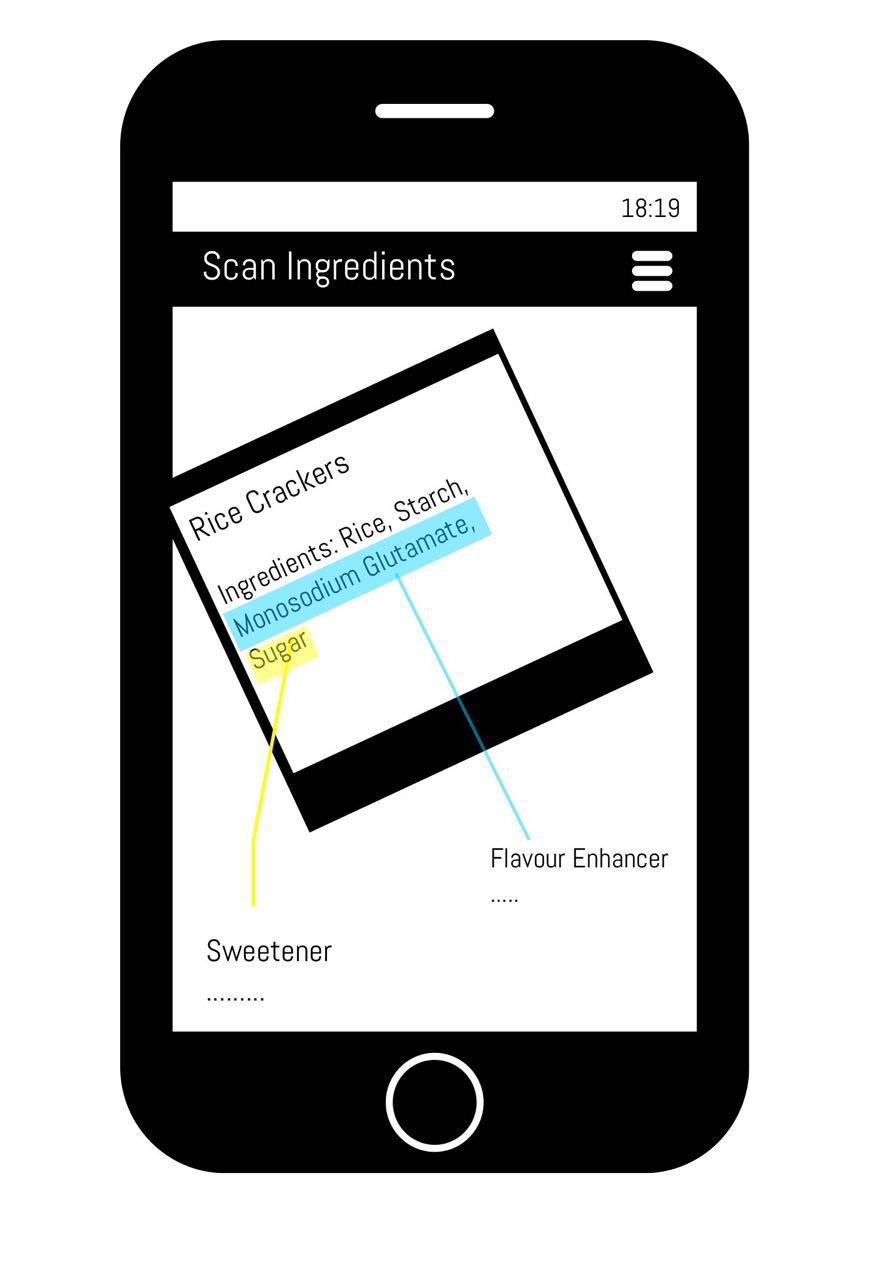
### 5.1 Initial Mockups/Wireframes

#### 

##### Figure 5.1.1 Launch page / Scanning of nutritional label

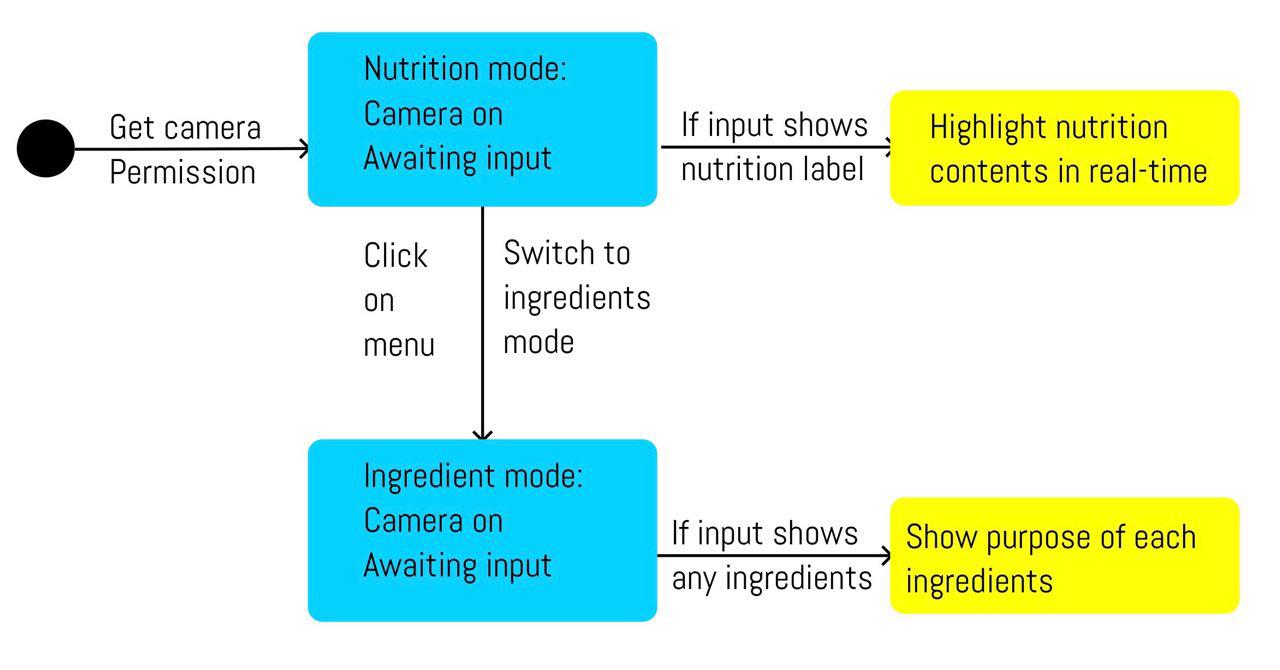
According to survey results collected in the preliminary project report, majority of the users prefer a real-time display of the information. *Figure 5.1.1* is an initial mockup on how the app should look like. Upon launching the app, the user will be asked permission to use the camera. After accepting, the app will be in picture mode ready to receive any input that resemble a nutritional label. *Figure 5.1.1* depicts a nutrition label being detected by the camera.

As shown in the image, the respective elements in the table is being highlighted: “CALORIES” is highlighted green which means that it is within the healthy range; “FATS” is highlighted yellow which suggest that it is near the unhealthy range; and “SUGAR” is highlighted red which means that there is a very unhealthy amount of sugar present in the food. The hamburger menu on the top right of the phone would switch to “scan ingredient label mode”.

****

##### Figure 5.1.2 Scanning ingredients

*Figure 5.1.2* represents the other core function of the application, which is the scanning of ingredients to find its’ uses. The image depicts a rice cracker being picked up to scan. Even though the rice cracker is held at an angle, the application was still able to identify the ingredients in the list and display the uses of each ingredients. For example, Monosodium glutamate as a flavor enhancer and more information which is shown as ellipses below “Flavor enhancer”. The hamburger menu button is for switching back to “Scan nutritional label mode”.



*Figure 3.1.3 UML Diagram for first iteration*

This UML diagram demonstrates the flow a user will experience when using the application. The black circle on the left is the starting point. Upon launching the application, user is asked to grant the application permission to use the phone’s camera. The application will first land on the “nutrition mode” and only switch to “ingredient mode” when menu icon is clicked.

### 5.2 Setting up the project

//Google image to text library  
implementation 'com.google.android.gms:play-services-vision:17.0.2'

##### Figure 5.2 Google Vision Dependency

Start a new project in the android studio and add the line of code shown in *Figure 5.2* into dependencies in the build.gradle to declare Google Vision’s dependency. At this phase of the project, we would need to create a CSVReader class to import the CSV data.

public class CSVReader {  
 InputStream inputStream;  
  
 public CSVReader(InputStream is){  
 this.inputStream = is;  
 }  
  
 public List<String[]> read(){  
 List<String[]> resultList = new ArrayList<>();  
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));  
  
 try{  
 String line;  
 while((line = reader.readLine()) != null){  
 String[] row = line.split(",");  
 resultList.add(row);  
 }  
  
 }catch(IOException e){  
 throw new RuntimeException("Error in reading csv file:" + e);  
 }finally {  
 try{  
 inputStream.close();  
 }catch (IOException e){  
 throw new RuntimeException("Error while closing input stream: " + e);  
 }  
 }  
 return resultList;  
 }  
}

##### Figure 5.2.2 CSVReader class

The CSVReader class awaits an InputStream from whichever class it will be called upon. It possesses a method called read() that reads the content of the CSV file line by line, in this case row by row, and each row of content is represented as a single String. CSV files are comma-separated value files. It means that multiple values in a single row are separated by commas. Hence, once the reader reaches the end of the row, it would divide a single row into multiple String values with the delimiter “comma”. The values are then added into an array list of String arrays so that it still maintains a table structure.

InputStream inputStream = getResources().openRawResource(R.raw.*data*);  
CSVReader csv = new CSVReader(inputStream);

foodList = csv.read();

##### Figure 5.2.3 CSVReader class being called

As shown in *Figure 5.2.3* the CSVReader class was called in the MainActivity class. Before it is called, the CSV file named “data” was saved in the resource/raw folder. The InputStream accesses the folder for the “data” file and the class is called with inputStream as the parameter.

Attempts were made to feed the data that was read into a CursorAdapter so that integration into the SQLite database and the use of its command will be more convenient. However, none of the attempts were successful and data was fed into an array list named “foodList” instead.

In this phase of the project, the function that display information in real-time should be developed and text detection of nutritional label should be completed. However, it was not achieved within the stipulated time frame that was established in the project plan. As it is still early in the development phase, extra time shall be dedicated to planning before moving on.

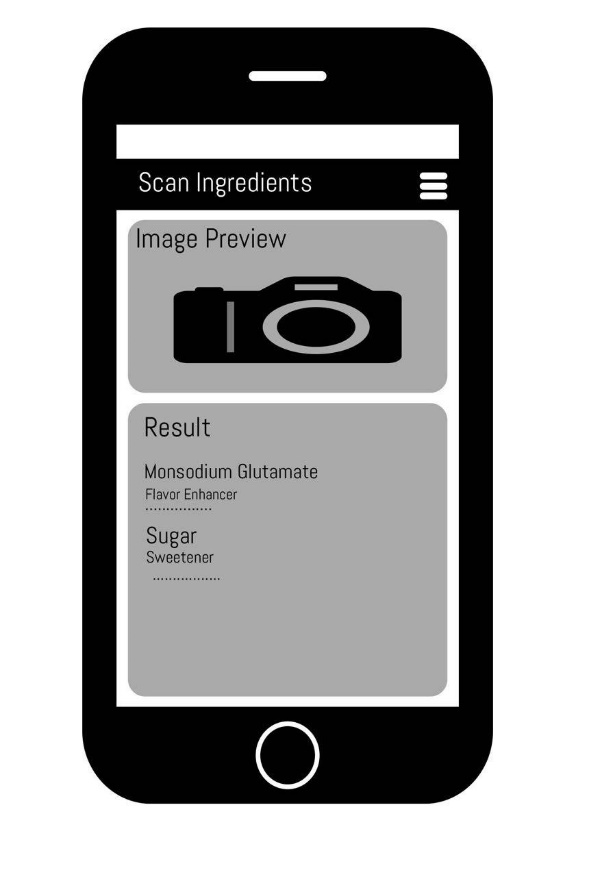
### Testing

As there was no working product developed in this phase, no test was conducted.

## Development Phase 2

### 6.1 Redesign Mockups/Wireframes

After much consideration, there will be a major change in the application design. There are many factors that attribute to this change. First and foremost, for the ingredients scanning feature of the application, there are usually many ingredients stated in the product label. Due to that, all the information about the ingredients would clutter the screen and would be messy for users to follow. The real-time display of nutritional information would oppose Nielsen’s ten heuristics of interaction, especially in terms of minimalism. Therefore, it is decided that the results should be displayed in a static list format instead.

*****Figure 6.1.1 Launch Page Figure.6.1.2 Results Page*

*Figure 6.1.1* shows the new launch page which does not start in the picture mode. When the user clicks on the menu image, “camera” and “gallery” option will cascade down from the icon. Selecting the “camera” option will ask the user for permission to use the camera while selecting the “gallery” option will ask the user for permission to access the user’s storage.

For the “camera” option, after the user snap a photo of the, it will direct users to the results page as shown in *Figure 6.2.2.* On the results page, the image taken will be shown as the image preview so that the user can verify what they have taken. In the “Results” card below image preview, a list of ingredients with its purpose and uses will be generated based on the text detected.

For the “gallery” option, the user will be directed to the user’s gallery page where they can choose images of food product which they would like to scan. Similarly, they will be directed to the results page upon selecting their image.

#### 6.2 Creating Launch Page

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

//Find view for respective sections

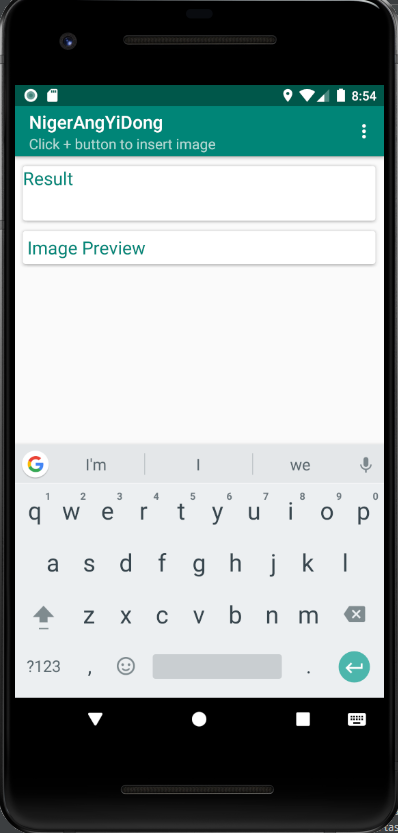
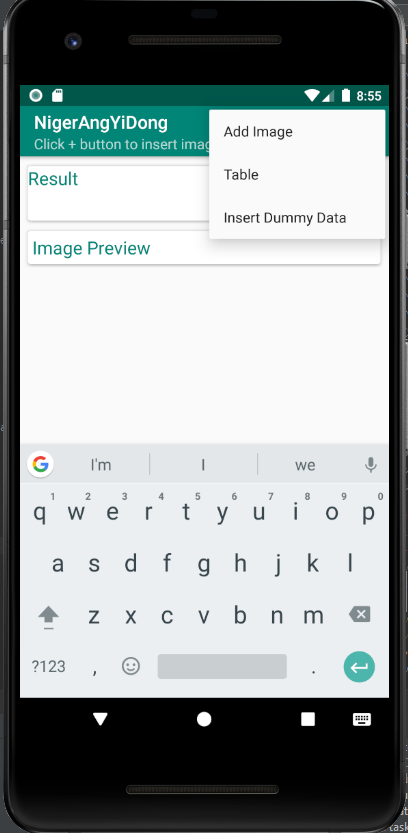
resultEditView = findViewById(R.id.resultEt);

imagePreview = findViewById(R.id.*imageIv*);

//Access and stream res/raw/data.csv  
 //Load into FoodArray with CSVReader class  
 InputStream inputStream = getResources().openRawResource(R.raw.*data*);  
 CSVReader csv = new CSVReader(inputStream);  
 foodList = csv.read();  
}

*Figure 6.2.1 Code snippet from Launch Page (MainActivity.java)*

The above code snippet shows how the MainActivity.java interacts with the XML layout file to create the appearance for the launch page. findViewByID links the view layout in the XML file to the output from the java class. The CSVReader is called immediately upon the launch of the app as well, so that the database is always ready to match text detected from the images.

##### Figure 6.2.2 Example of Launch Page Figure 6.2.3 Example of Menu

The above image is the results of the code in MainActivity.java and activity\_main.xml. Dummy options are added to the menu as shown in *Figure 6.2.3* to get a better idea on how the interaction would be like with the end user. However, there was an issue with the user interface. Upon launching the application, the keyboard would appear with a blinking cursor on the “Result” card. This issue is minor so in accordance to the Cowboy methodology, it will be left undone till the more important functions are done.

### 6.3 Permissions

<!--Ask user for permission to camera and storage-->  
<uses-permission android:name="android.permission.CAMERA"/>  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

*Figure 6.3.1 Permissions in AndroidManifest.xml*

Before the application can function as intended, it needs to ask the user’s permission to use the phone’s camera for taking photos of labels and to access their gallery so that they can retrieve past photos that they want to scan. Therefore, it is essential to put these two lines of code in the android manifest for it to work, as shown in *Figure 6.3.1.*

private void showImageImportDialog() {  
 //items to display in dialog  
 String[] items = {" Camera", " Gallery"};  
 AlertDialog.Builder dialog = new AlertDialog.Builder(this);  
 //set title  
 dialog.setTitle("Select Image");  
 dialog.setItems(items, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if(which == 0){  
 //camera option clicked  
 if(!checkCameraPermission()){  
 //camera permission not allowed, request it  
 requestCameraPermission();  
 }else{  
 //camera permission allowed  
 pickCamera();  
 }  
 }  
 if(which == 1){  
 //gallery option clicked  
 if(!checkStoragePermission()){  
 //storage permission not allowed  
 requestStoragePermission();  
 }else{  
 //storage permission allowed  
 pickGallery();  
 }  
 }  
 }  
 });  
 dialog.create().show(); //show dialog  
}

*Figure 6.3.2 Code snippet for popup dialog asking for user’s choice*

The code in *figure 6.3.2* will show a popup when user intends to add an image. It will check whether the user wants to add the image via the camera or their own gallery. With each selection, there are if-else statements to check whether the application has permission to execute the action.

For example, as shown in public void onClick, if the camera function is chosen, it will check for camera permission. If camera permission is denied (if(!checkCameraPermission())), the application will request for the camera permission. Otherwise, if permission was allowed before, it would go directly into picture mode. This is similar for gallery mode.

private boolean checkCameraPermission() {  
 boolean cameraResult = ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*CAMERA*) == (PackageManager.*PERMISSION\_GRANTED*);  
 boolean storageResult = ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == (PackageManager.*PERMISSION\_GRANTED*);  
 return cameraResult && storageResult;  
}

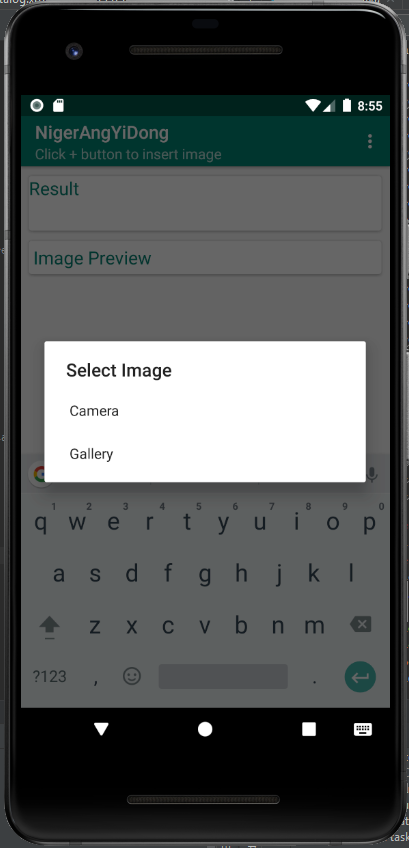
*Figure 6.3.3 Check camera permission*

This method checks with AndroidManifest.xml whether application is granted permission and returns true if permission is granted, likewise for checkStoragePermission() method (Gallery mode).

private void requestCameraPermission(){  
 ActivityCompat.*requestPermissions*(this, cameraPermission, *CAMERA\_REQUEST\_CODE*);  
}

*Figure 6.3.4 Request for camera permission*

This method requests for permission to use phone’s camera. The method for requesting storage permission is like this as well.



*Figure 6.3.4 Popup dialog*

*Figure 6.3.4* shows how the application should look like upon executing the codes mentioned in this section.

### 6.4 Image to text function

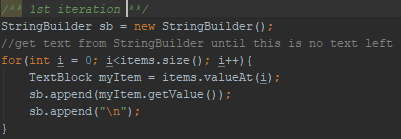
The image to text function is the most important feature in this entire project. It is imperative to get this function working to a high degree of accuracy before proceeding. However, there is an issue with Google Vision when it comes to scanning text in arbitrary order. The words are detected and scanned in correctly but the sequence in which it is printed is correct.

//get drawable bitmap for text recognition  
BitmapDrawable bitmapDrawable = (BitmapDrawable) imagePreview.getDrawable();  
Bitmap bitmap = bitmapDrawable.getBitmap();  
  
TextRecognizer recognizer = new TextRecognizer.Builder(getApplicationContext()).build();

*Figure 6.4.1 Text Recognizer*

Frame frame = new Frame.Builder().setBitmap(bitmap).build();  
SparseArray<TextBlock> items = recognizer.detect(frame);

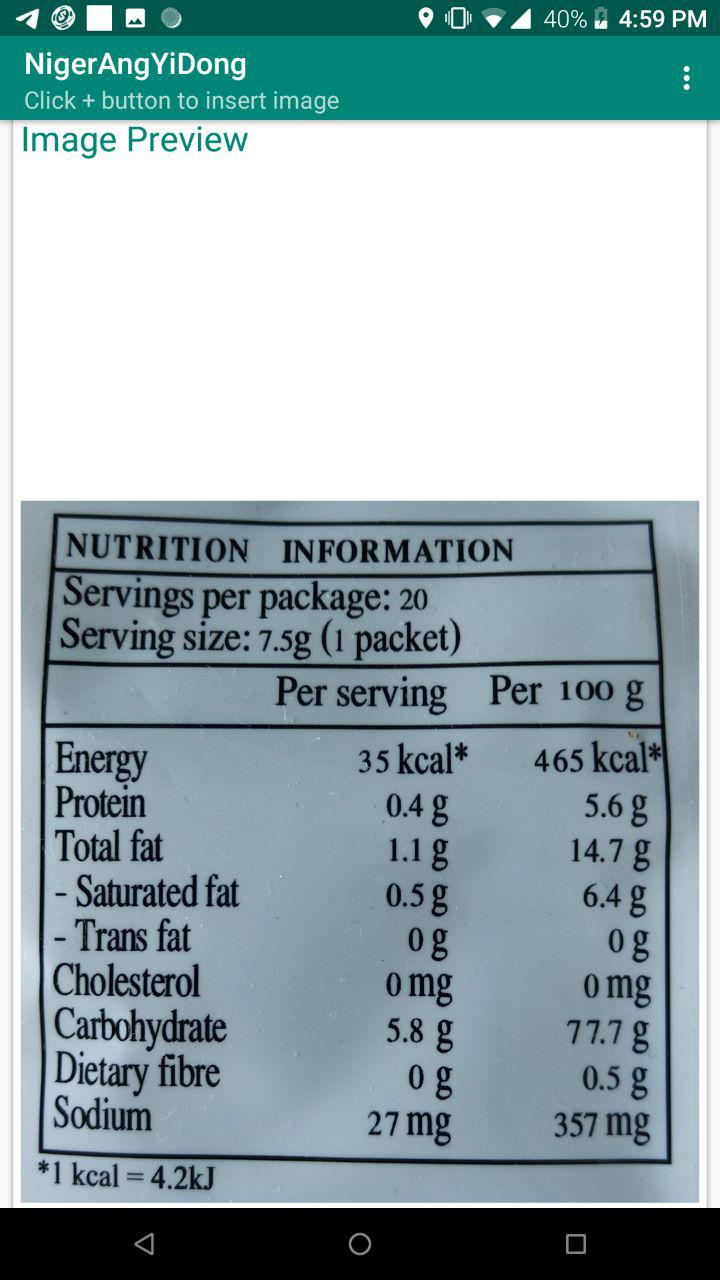
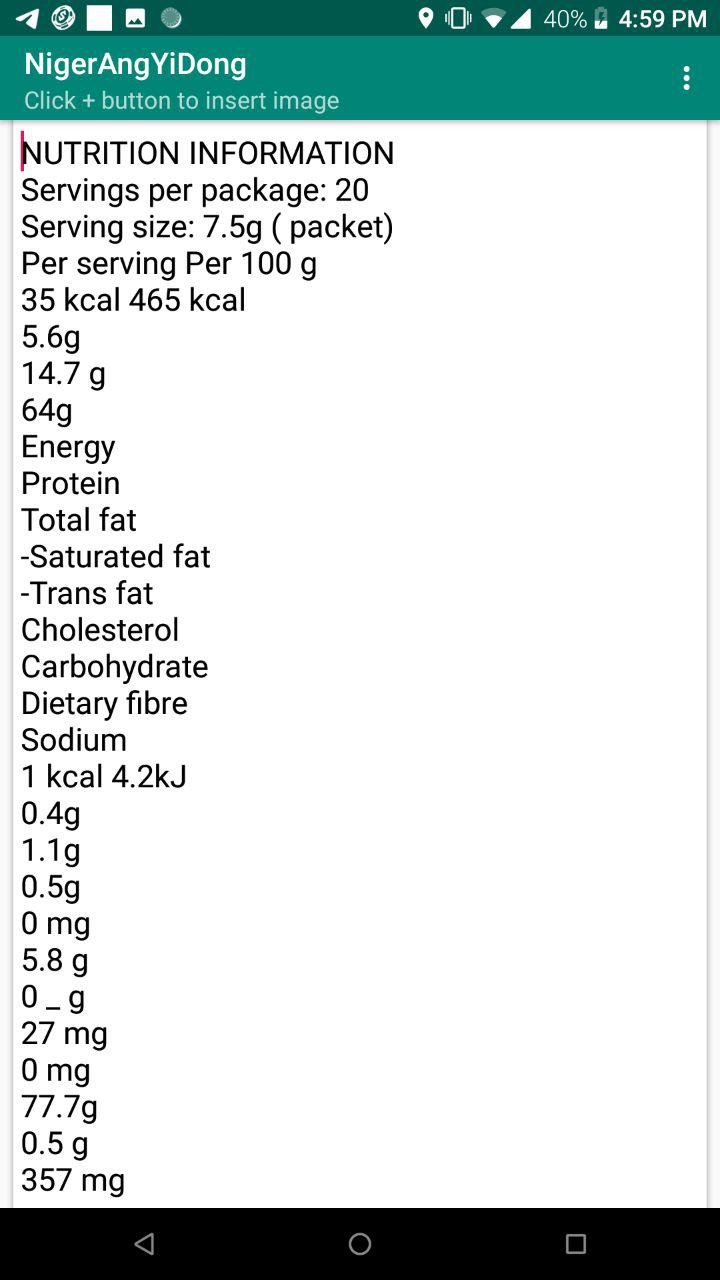
*Figure 6.4.2 Detecting frames*



*Figure 6.4.3 Stringbuilder*

Figures 6.4.1 to 6.4.3 are code snippets taken from onActivityResult method in MainActivity.java. The image is converted into a bitmap for the Text Recognizer to detect the words. Words that are detected are then store into an array as TextBlock. The contents of the array are then added into a Stringbuilder before being printed out.

Referring to *figure 6.4.3,* the TextBlocks are added in a specific order as they go through the for loop. In theory, it should yield the same results if we scan image multiple times. However, upon multiple rounds of testing, the results constantly shown in different order.

*Figure 6.4.4 Image preview of nutrition facts Figure 6.4.5 Example of scanned results*

As pointed out in *figure 6.4.4 and figure 6.4.5,* the results printed are not in the same order as the nutrition table. The same image was tested out for 4 more times and the order of the results differ for every test. There were no recurring patterns in the results for logical debugging.

In addition, there were additional white spaces above and below the image preview is not aesthetically pleasing and may confuse the user as they do not see the results immediately. This is a minor design issue and will be left at the end to be complete with other unsolved problems.

Frame frame = new Frame.Builder().setBitmap(bitmap).build();  
SparseArray<TextBlock> items = recognizer.detect(frame);  
String lines = "";  
ArrayList<Textholder> textholder = new ArrayList<Textholder>();  
ArrayList<Textholder> textholder1 = new ArrayList<Textholder>();  
for(int index = 0; index < items.size(); index ++){  
 int key = items.keyAt(index);  
 TextBlock textBlock = items.valueAt(index);  
 if(textBlock != null){ //if there is a block, get its components  
 for(Text line : textBlock.getComponents()){ //go through each line in block  
 float topLine = line.getBoundingBox().top;  
 Textholder lineHolder = new Textholder(topLine, line.getValue());  
 textholder1.add(lineHolder);  
 for(Text element: line.getComponents()) {  
 float top = element.getBoundingBox().top; //record the line's box and coordinates  
 Textholder th = new Textholder(top, element.getValue());  
 textholder.add(th);  
 }  
 }  
 }

Collections.*sort*(textholder);  
for(int i = 0; i < textholder1.size(); i++){  
 Textholder t = textholder1.get(i);  
 lines = lines + t.getText();

*Figure 6.4.5 Solution for text not printing in order*

After researching online, I found a solution for the sequence problem. This solution was found in an issue forum for android-vision.[[14]](#footnote-14) As the text are detected in blocks, the solution was to record the coordinates of the top of each textblock along with the text itself. A textholder class was created to hold the coordinates and text. Full code of the textholder class can be found in the appendix. The arraylist is then sorted according to its coordinates and printed out. I was able to rectify this problem for the ingredients list but not for the nutritional facts portion. As too much time was dedicated to solving the problems regarding the nutritional label, I will have to forgo the function of highlighting nutritional elements in this project for now.

*/\*\*  
 \* Calculates the similarity between two strings.  
 \* Exact match = 1.0  
 \* No match = 0.0  
 \*/*public static double similarity(String s1, String s2) {  
 String longer = s1, shorter = s2;  
 if (s1.length() < s2.length()) { // longer should always have greater length  
 longer = s2; shorter = s1;  
 }  
 int longerLength = longer.length();  
 if (longerLength == 0) {  
 return 1.0; /\* both strings are zero length \*/  
 }  
 return (longerLength - *editDistance*(longer, shorter)) / (double)longerLength;  
}

*Figure 6.4.6 Code snippets of Similarity method*

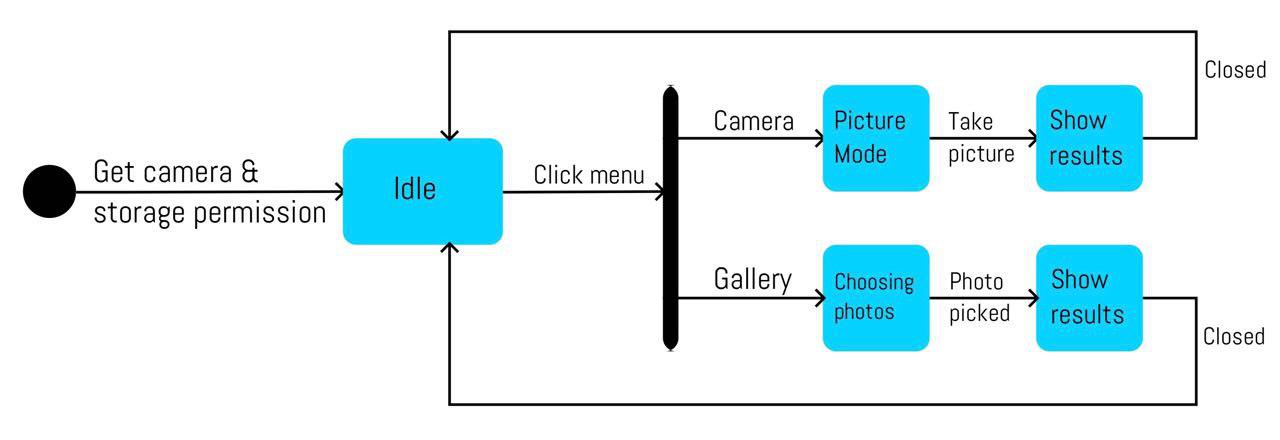
// Source: https://rosettacode.org/wiki/Levenshtein\_distance#Java

public static int editDistance(String s1, String s2) {  
 s1 = s1.toLowerCase();  
 s2 = s2.toLowerCase();  
  
 int[] costs = new int[s2.length() + 1];  
 for (int i = 0; i <= s1.length(); i++) {  
 int lastValue = i;  
 for (int j = 0; j <= s2.length(); j++) {  
 if (i == 0)  
 costs[j] = j;  
 else {  
 if (j > 0) {  
 int newValue = costs[j - 1];  
 if (s1.charAt(i - 1) != s2.charAt(j - 1))  
 newValue = Math.*min*(Math.*min*(newValue, lastValue),  
 costs[j]) + 1;  
 costs[j - 1] = lastValue;  
 lastValue = newValue;  
 }  
 }  
 }  
 if (i > 0)  
 costs[s2.length()] = lastValue;  
 }  
 return costs[s2.length()];  
}

*Figure 6.4.7 Code snippets of editDistance method*

As shown in *figure 6.4.6* and *figure 6.4.7*, Levenshtein’s distance was used to match the ingredients in the database. “The Levenshtein distance is a string metric for measuring the difference between two sequences.”[[15]](#footnote-15) Levenshterin distance, or edit distance, is based on the number of single-character edits (insertions, deletions, substitutions etc) it takes to change from one word to another.

In the editDistance method, it takes in two strings as a parameter. Before comparing the edit distance, the strings are converted into lowercase so that there will be no additional edits. The smaller the distance, the higher the similarity between the two strings as it takes lesser number of edits to convert from one to another. In the similarity method, the length of the longer string is subtracted by the edit distance and then divided by the longer string’s length. If the output of the method is 1, it means that either both strings have zero length or both strings are exact match as edit distance would be zero.

****

*Figure 6.4.8 UML Diagram for 2nd iteration cycle*

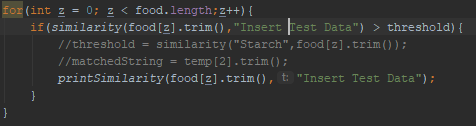
This UML diagram shows the new workflow of the application. The application will be in an idle state unless the user commits any actions as mentioned previously. Clicking on menu shows 2 ways a user can add image into the app, camera and gallery. Choosing camera will take user to the picture mode and choosing gallery will let user browse through their gallery.

### Testing

During development, some inaccuracies in text detection was discovered. For example, the square brackets “[ ]” can sometimes be interpreted as “J” or “I” by the text recognizer and commas”,” can sometimes be interpreted as full stops “.”. Additionally, the application sometimes output data that do not match the ingredients scanned at all. This is because the code always takes the closest similarity the ingredient has to the database even though those ingredients are absent in the database.

For example, if the database contains a data named “Random TestData”, it would score 0.133 with the ingredient “Rice”. If there are no higher score for “Rice”, “Random TestData” would be printed out as a match with “Rice” despite them not being similar at all.

The solution to this problem is to set threshold value for the similarity score. To be considered as a match, the score between words from the ingredient list and the database must exceed the threshold value. Therefore, a test would be conducted to find this threshold value, while trying to solve misinterpretation problem from the Text Recognizer.

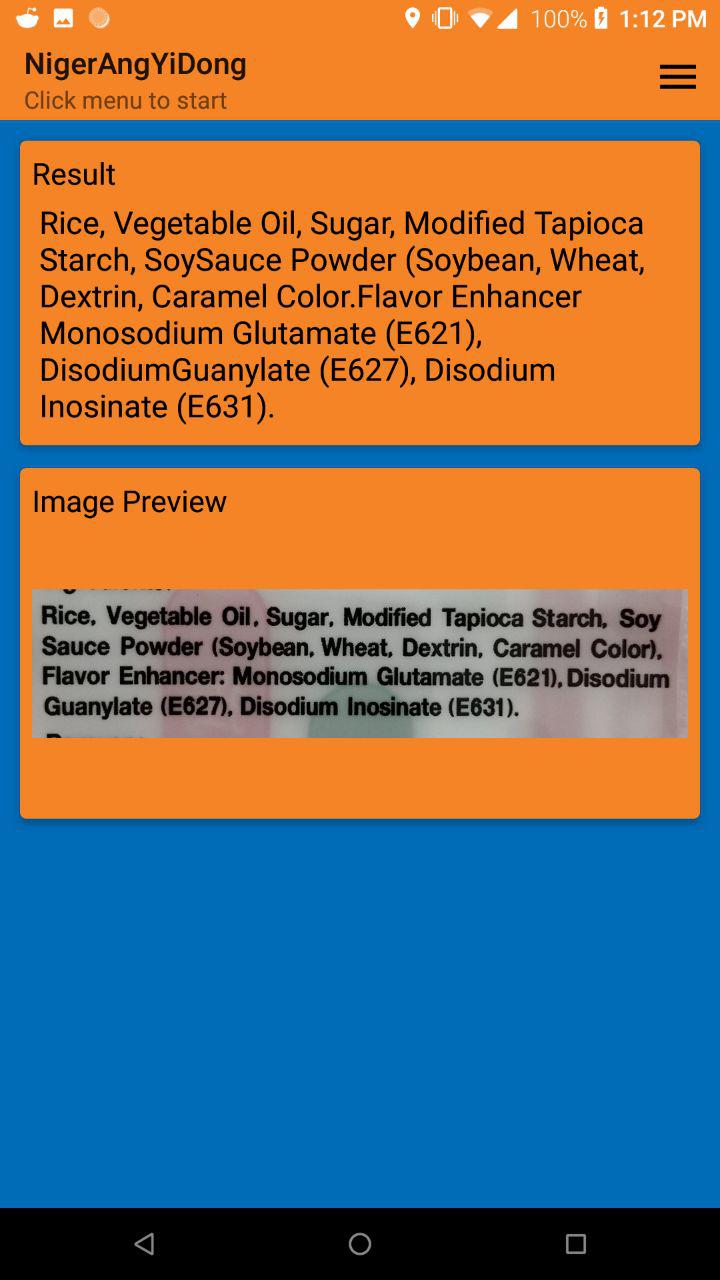
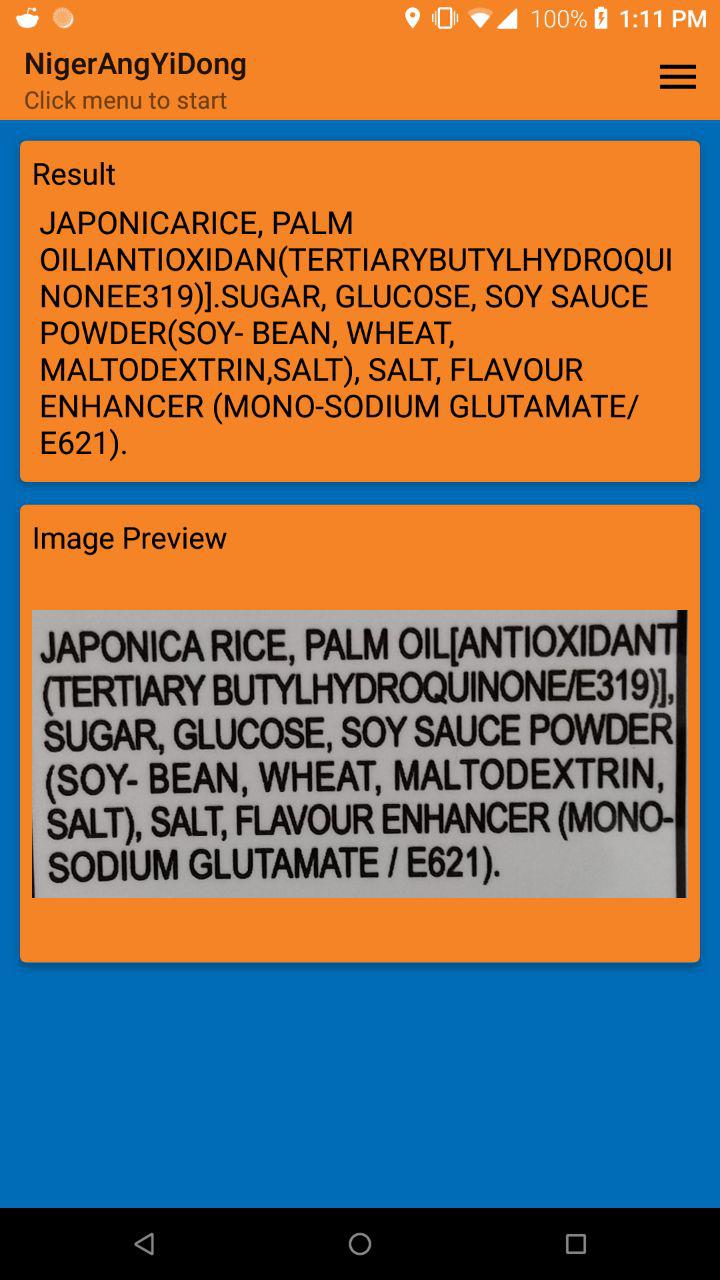


*Figure 6.5.1 Accuracy Test*

In this accuracy test, we set the value of threshold to be 0. We will run this code 3 times with different string values in the “Insert Test Data” portion and using 2 product labels to get a more accurate average.



*Figure 6.5.2 Version 1 test*

*Figure 6.5.3 Version 1Bin Bin Image Figure 6.5.4 Version 1 Wang Wang Image*

The first version has the original delimiter with no changes. The product used in these tests are Bin Bin Rice Crackers and Wang Wang Rice Crackers. *Figure 6.5.3 and figure 6.5.4* are the image previews and results generated for this test.

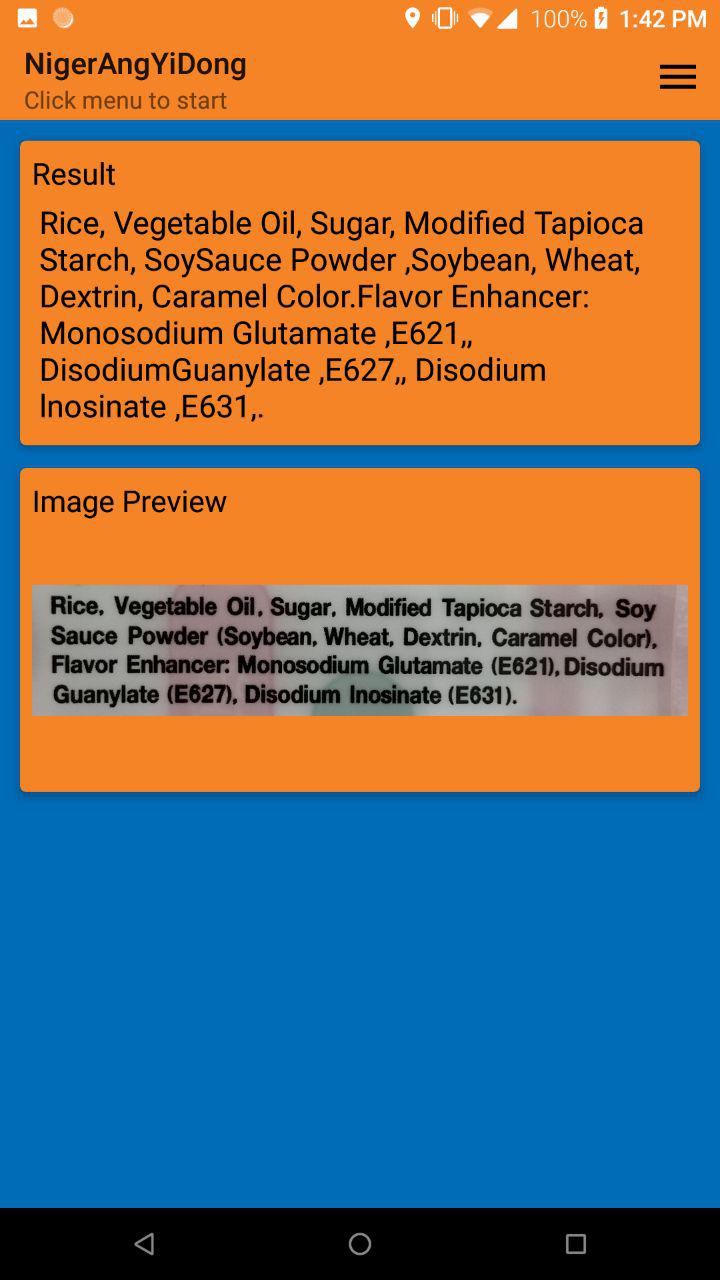
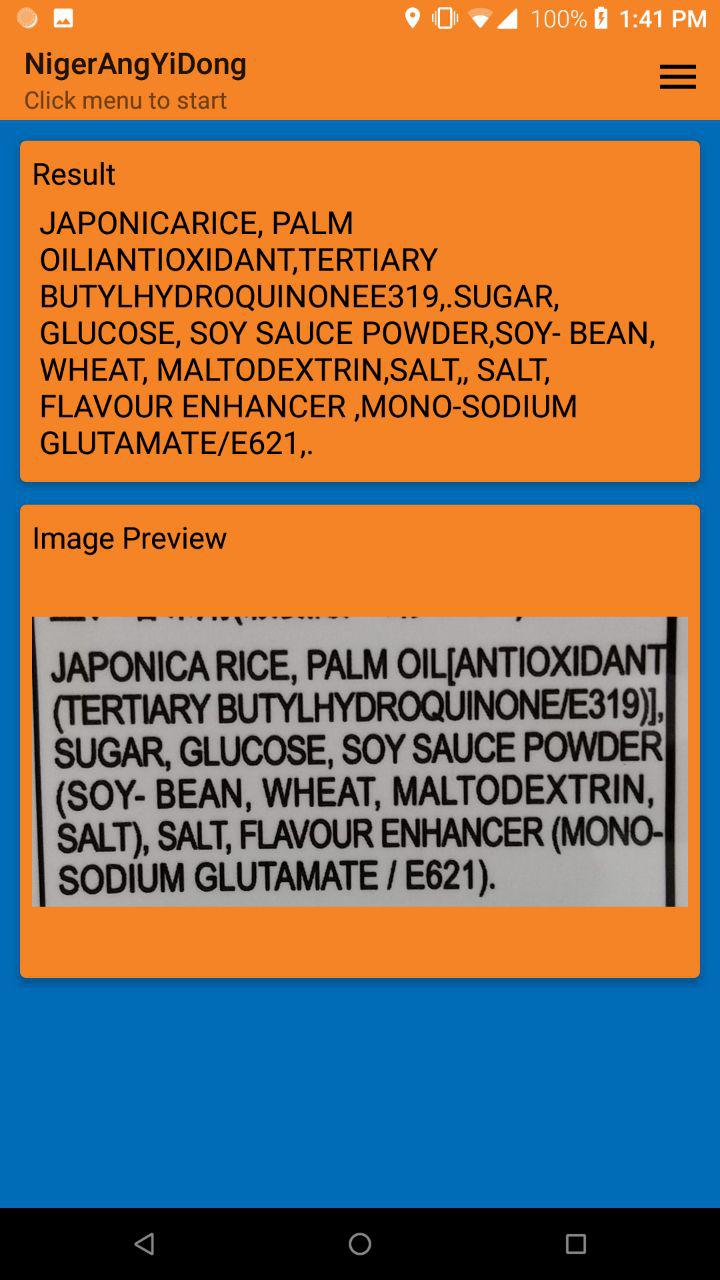
|  |  |  |
| --- | --- | --- |
| Test data word | Maltodextrin | Similarity Score |
| Bin Bin Rice Cracker |  | 0.583 |
| Wang Wang Rice Cracker |  | 1.000 |
| Test data word | Monosodium Glutamate | - |
| Bin Bin Rice Cracker |  | 0.351 |
| Wang Wang Rice Cracker |  | 0.435 |
| Test data word | Starch | - |
| Bin Bin Rice Cracker |  | 0.133 |
| Wang Wang Rice Cracker |  | Not Applicable |

*Table 1 Accuracy Test Version 1*

From the first test, we can see that some ingredients are not separated properly as “,” are viewed as “.” and in some ingredient list, brackets are supposed to be a delimiter but it the data is printed as one long sentence. If a long sentence is present in the comparison, it would greatly lower the similarity score. This is reflected under “Monosodium Glutamate” for Bin Bin Rice Cracker, you can see that monosodium glutamate is clearly present however it yields a low similarity score.

**

*Figure 6.5.5 Version 2 test*

* *

*Figure 6.5.6 Version 2 Bin Bin Image Figure 6.5.7 Version 2 Wang Wang Image*

The second test replaces all normal brackets “()” with commas before splitting them with comma as a delimiter. *Figure 6.5.6 and figure 6.5.7* are the image previews and results generated for this test.

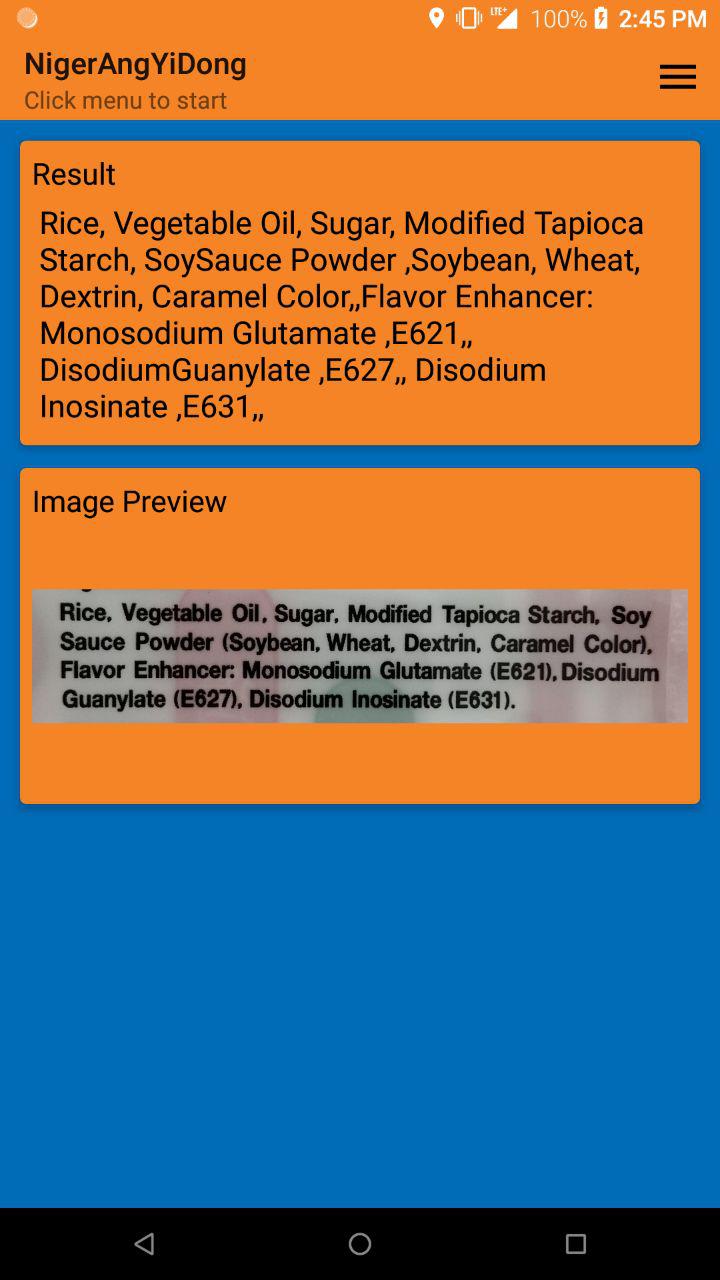
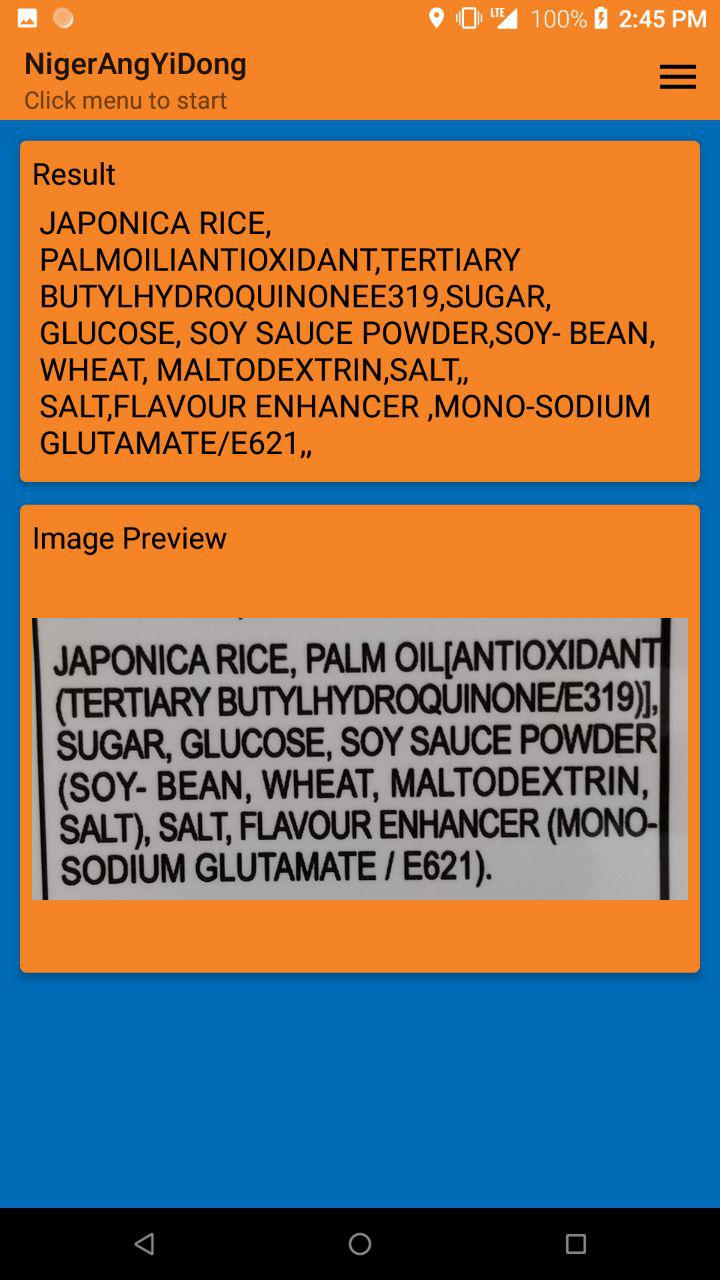
|  |  |  |
| --- | --- | --- |
| Test data word | Maltodextrin | Similarity Score |
| Bin Bin Rice Cracker |  | 0.583 |
| Wang Wang Rice Cracker |  | 1.000 |
| Test data word | Monosodium Glutamate | - |
| Bin Bin Rice Cracker |  | 0.392 |
| Wang Wang Rice Cracker |  | 0.769 |
| Test data word | Starch | - |
| Bin Bin Rice Cracker |  | 0.261 |
| Wang Wang Rice Cracker |  | Not Applicable |

*Table 2 Accuracy Test Version 2*

From the second test, under “Starch” for Wang Wang Rice Cracker, the text blocks have been further broken down and we are getting more accurate results as similarity scores have increased for some of the test cases. However, we can still see a comma being misinterpreted as a full stop.



*Figure 6.5.8 Version 3*

* *

*Figure 6.5.9 Version 3 Bin Bin Image Figure 6.5.10 Version 3 Wang Wang Image*

For the final test, adding on to what was done in version 2, the full stop “.” will be replaced with a comma “,” as well. *Figure 6.5.9 and figure 6.5.10* are the image previews and results generated for this test.

|  |  |  |
| --- | --- | --- |
| Test data word | Maltodextrin | Similarity Score |
| Bin Bin Rice Cracker |  | 0.583 |
| Wang Wang Rice Cracker |  | 1.000 |
| Test data word | Monosodium Glutamate | - |
| Bin Bin Rice Cracker |  | 0.541 |
| Wang Wang Rice Cracker |  | 0.769 |
| Test data word | Starch | - |
| Bin Bin Rice Cracker |  | 0.261 |
| Wang Wang Rice Cracker |  | Not Applicable |

*Table 3 Accuracy Test Version 3*

From the final test, we only see a huge improvement under “Monosodium Glutamate” under Bin Bin Rice Cracker for the similarity score. Other results of the test did not have any significant changes. In addition, the problem of comma being misinterpreted has been eliminated.

|  |  |  |  |
| --- | --- | --- | --- |
| Similarity Score | Version 1 | Version 2 | Version 3 |
| Bin (Maltodextrin) | 0.583 | 0.583 | 0.583 |
| Wang (Maltodextrin) | 1.000 | 1.000 | 1.000 |
| Bin (MSG) | 0.351 | 0.392 | 0.541 |
| Wang (MSG) | 0.435 | 0.769 | 0.769 |
| Bin (Starch) | 0.133 | 0.261 | 0.261 |
| Wang (Starch) | Not Applicable | Not Applicable | Not Applicable |

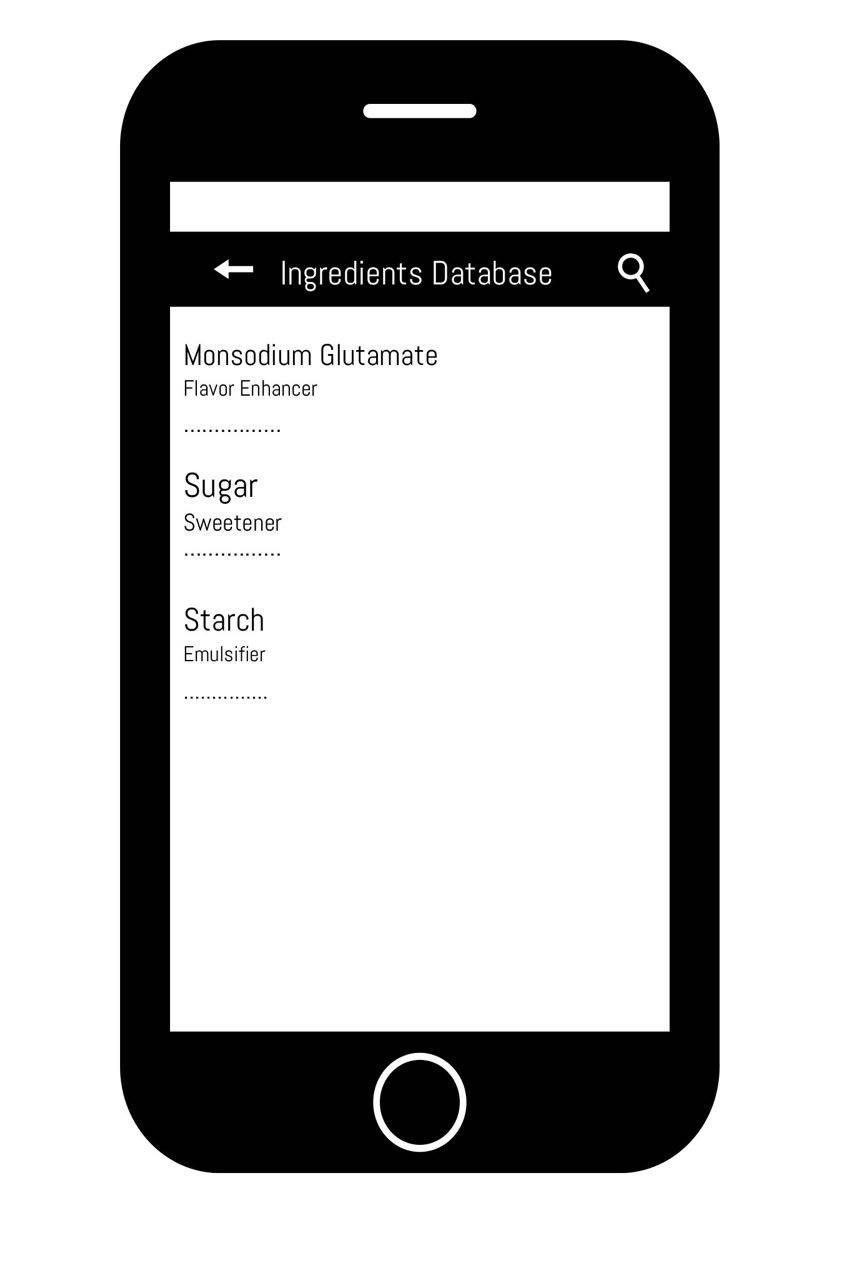
*Table 4 Comparison of similarity score across 3 versions*

From Table 4, we see a gradual improvement of each result, especially in the case of Bin (MSG). Judging from the results in Table 4 and overall output from these tests, one can deduce that the threshold value should be slightly higher than 0.50. By averaging the results, the threshold value that should be used is 0.520. Images in table 1 to 3 are only snippets of the output generated for each test. A more detailed output can be found in the appendix. Furthermore, through this test, we were able to reduce some text detection error.

## Development phase 3

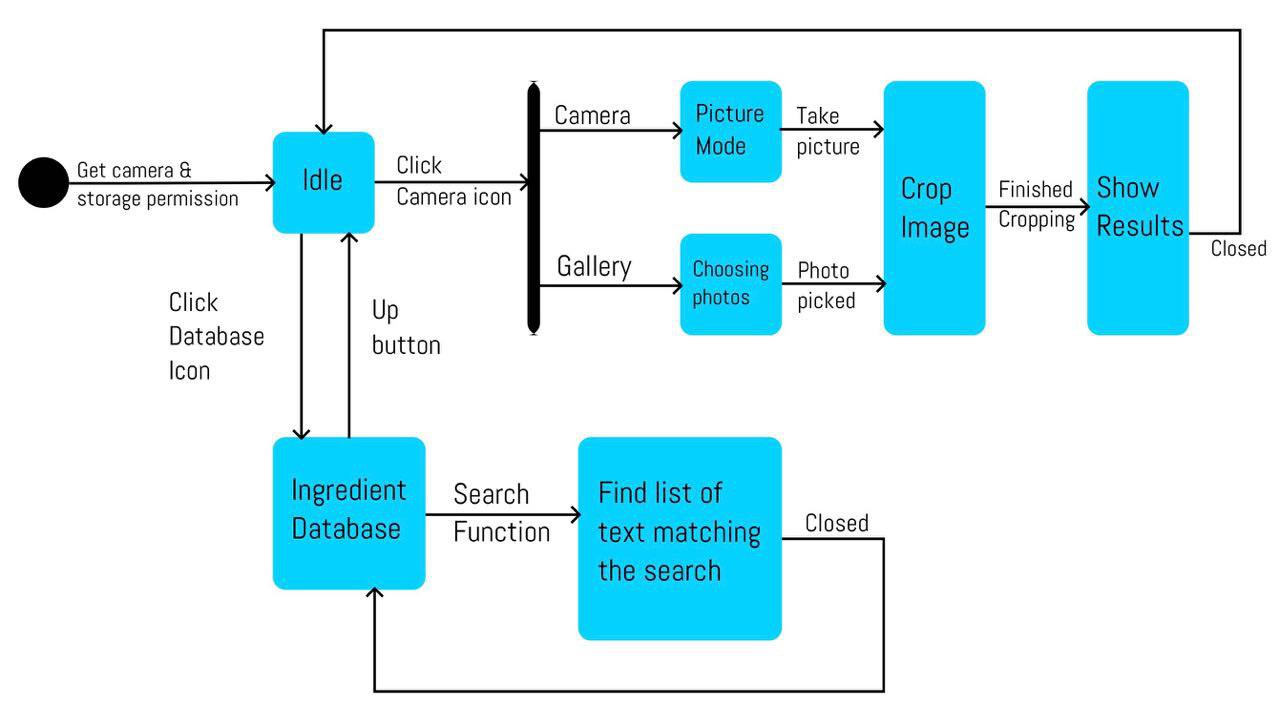
### 7.1 Additional Page Mockup / Wireframes

This would be the last iteration cycle in this project. In this final cycle, the plan would be to fix any minor issues encountered in the previous development phases and improve on areas that can be done better. Regrettably, the problem about the jumbled text output from the nutrition facts table remains unresolved. The function of highlighting nutrition elements will not be included in the final product.



*Figure 7.1.1 Database Page*

This database page design is an additional feature that was not included in the initial project plan. The idea to create this was because there were still issues regarding the application not accurately detecting text. This can be due to many external factors like the reflection on the product label or clarity of image from users’ phones. Therefore, as a countermeasure to the accuracy issue, I have decided to add a database page where ingredients are listed out with its uses. A search function was included to ease the process of finding any ingredients. A back button was added to allow users to navigate back to the launch page.



*Figure 7.1.2 UML Diagram for 3rd iteration cycle*

*Figure 7.1.2* shows the UML diagram for application in the final development cycle. Like the previous iteration, the application starts in an idle state. However, with the additional database button, it was decided that the application shall have 2 separate icons: camera icon for adding images, either through camera or gallery; and a list icon to direct user to the ingredient database page. By separating the menu into 2 icons, lesser step was taken to navigate through the application. This fulfills the flexibility and efficiency of use aspect in Nielsen’s Heuristics as taught in Interaction Design.

### 7.2 Image Cropper Activity

//Android Image Cropper Library Source: https://github.com/ArthurHub/Android-Image-Cropper  
implementation 'com.theartofdev.edmodo:android-image-cropper:2.7.0'

*Figure 7.1.2 Image Cropper dependency*

<!--Crop image manifest-->  
<activity android:name="com.theartofdev.edmodo.cropper.CropImageActivity"  
 android:theme="@style/CropTheme"/>

*Figure 7.1.3 Crop image manifest*

As shown in *figure 7.1.3,* a crop image activity was added to improve on the accuracy of the text detection in the application. After taking photo or choosing a photo in the gallery, the user can crop the image to focus only on the important points of the label.

### Ingredients Database

static class FoodViewHolder{  
 TextView Type;  
 TextView Purpose;  
 TextView Name;  
}  
public FoodArrayAdapter(Context context, int resource) {  
 super(context, resource);  
}  
  
//method to add objects into foodList array  
public void add(String[] object){  
 foodList.add(object);  
 super.add(object);  
}

View group = convertView;  
 FoodViewHolder viewHolder;  
 if(group == null){  
 LayoutInflater inflater = (LayoutInflater) this.getContext().getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);  
 group = inflater.inflate(R.layout.*list\_item*, parent, false);  
 viewHolder = new FoodViewHolder();  
 viewHolder.Type = group.findViewById(R.id.*type*);  
 viewHolder.Purpose= group.findViewById(R.id.*purpose*);  
 viewHolder.Name= group.findViewById(R.id.*name*);  
 group.setTag(viewHolder);  
 }else{  
 viewHolder = (FoodViewHolder) group.getTag();  
 }  
  
 String[] stat = getItem(position);  
 viewHolder.Type.setText(stat[0]);  
 viewHolder.Purpose.setText(stat[1]);  
 viewHolder.Name.setText(stat[2]);  
 return group;  
}

*Figure 7.3.1 Code snippet of FoodArrayAdapter.java*

For ingredients and its uses to be listed as shown in the mockup. An array adapter is needed to pass the information into an xml layout. The array adapter holds the 3 components type, purpose and name in 3 separate textviews.

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="24dp">  
  
 <TextView  
 android:id="@+id/name"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif"  
 android:textAppearance="?android:textAppearanceMedium"  
 android:textColor="@color/colorDarkGray" />  
  
 <TextView  
 android:id="@+id/type"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif-medium"  
 android:textAppearance="?android:textAppearanceSmall"  
 android:textColor="@color/colorGray" />  
  
 <TextView  
 android:id="@+id/purpose"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif"  
 android:textAppearance="?android:textAppearanceSmall"  
 android:textColor="@color/colorGray" />  
</LinearLayout>

*Figure 7.3.2 list\_item.xml*

The list\_item,xml is grouping the 3 data together in a layout and shows how the ingredients data are being displayed.

foodListView = findViewById(R.id.*list*);

// Setup an Adapter to create a list item for each row  
mArrayAdapter = new FoodArrayAdapter(getApplicationContext(), R.layout.*list\_item*);  
foodListView.setAdapter(mArrayAdapter);  
  
//Access and stream res/raw/data.csv  
//Load into FoodArray with CSVReader class  
InputStream inputStream = getResources().openRawResource(R.raw.*data*);  
CSVReader csv = new CSVReader(inputStream);  
foodList = csv.read();  
for(String [] foodData : foodList){  
 mArrayAdapter.add(foodData);  
}

*Figure 7.3.3 Code snippet of TableActivity.java*

TableActivity.java is the logic behind how the ingredient database page works. *Figure 7.3.3* demonstrates how the page is populated by the data. First, the CSV file is loaded in onCreate() and fed into the FoodArrayAdpter. The foodListview is set to the FoodArrayAdapter create a list item for each row. A similar code base is found in MainActivity.java when the scanned text results match the ingredient database.

<?xml version="1.0" encoding="utf-8"?>  
  
<searchable xmlns:android="http://schemas.android.com/apk/res/android"  
 android:label="@string/database\_name"  
 android:hint="Search" />

*Figure 7.3.4 Searchable.xml*

This xml file is needed in order for the search function to work.

// Associate searchable configuration with the SearchView  
SearchView searchView = (SearchView) menu.findItem(R.id.*search*).getActionView();  
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String s) {  
 return false;  
 }  
  
 @Override  
 public boolean onQueryTextChange(String s) {  
 ArrayList<String[]> tempList = new ArrayList<>();  
 FoodArrayAdapter adapter = new FoodArrayAdapter(getApplicationContext(), R.layout.*list\_item*);  
 foodListView.setAdapter(adapter);  
  
 for (String [] temp : foodList) {  
 if(temp[2].toLowerCase().contains(s.toLowerCase())){  
 adapter.add(temp);  
 }  
 }  
 return true;  
 }  
});

*Figure 7.3.5 Search function*

The search function is found in onCreateOptionsMenu() method. It sets it view to a menu xml file and the waits for the user’s input. When user starts typing in the search bar, it creates a temporary array list of items that fulfils the condition of what was in the search bar. The arraylist is then inserted into the FoodArrayAdapter to displayed in foodListView after adapter is set.

### 7.4 Fixing bugs

As mentioned in the previous development phases, there are 2 minor bugs to be fixed. The first bug was that upon launching the application, the keyboard would appear with a blinking cursor on the “Result” card as shown in *Figure 6.2.2 Example of launch page.* The second bug was that there was too much whitespace at the top and bottom of the image preview.

//Hide keyboard when app starts  
getWindow().setSoftInputMode(WindowManager.LayoutParams.*SOFT\_INPUT\_STATE\_HIDDEN*);

*Figure 7.4.1 First bug solution*

The above figure shows the solution to hiding the keyboard upon application launch. This solves the first bug easily.

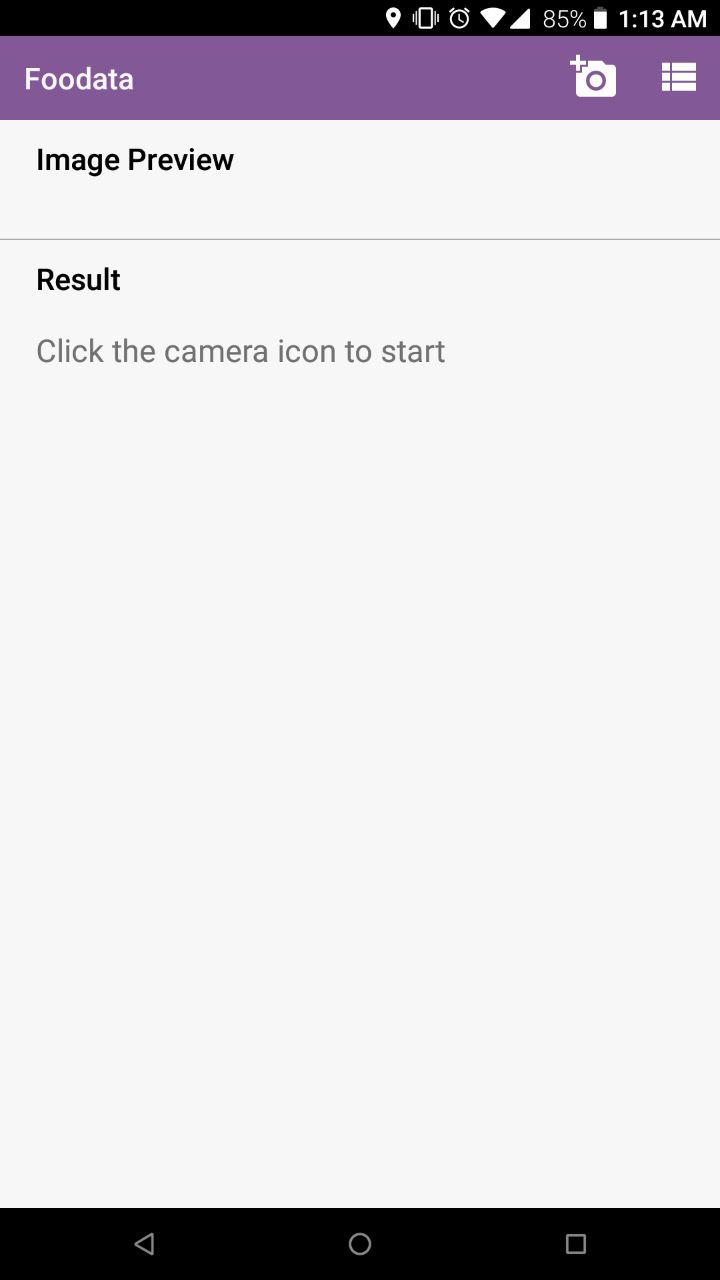
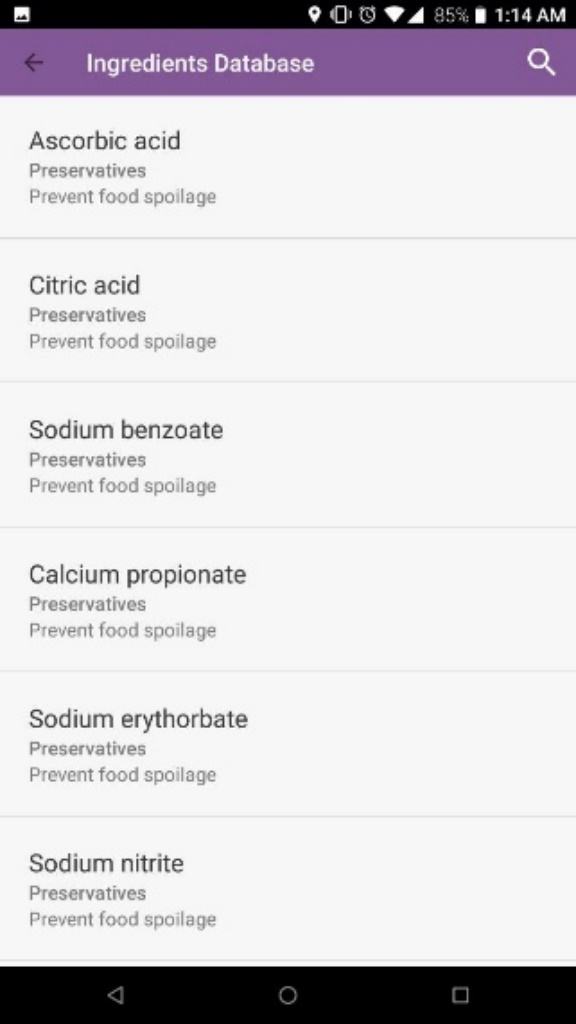
<ImageView  
 android:id="@+id/imageIv"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:adjustViewBounds="true"  
 android:maxHeight="200dp"  
 android:paddingTop="24dp"  
 android:paddingLeft="24dp"  
 android:paddingRight="24dp"/>

*Figure 7.4.2 Second bug solution*

*Figure 7.4.2* shows the portion of the activity\_main.xml file which is in charge of image preview on the main page. The codes that solved the problem are adjustViewBounds=“true” and maxHeight=“200dp”. AdjustViewBounds shrinks the image to fit the aspect ratio which was constrained by maxHeight= “200dp”

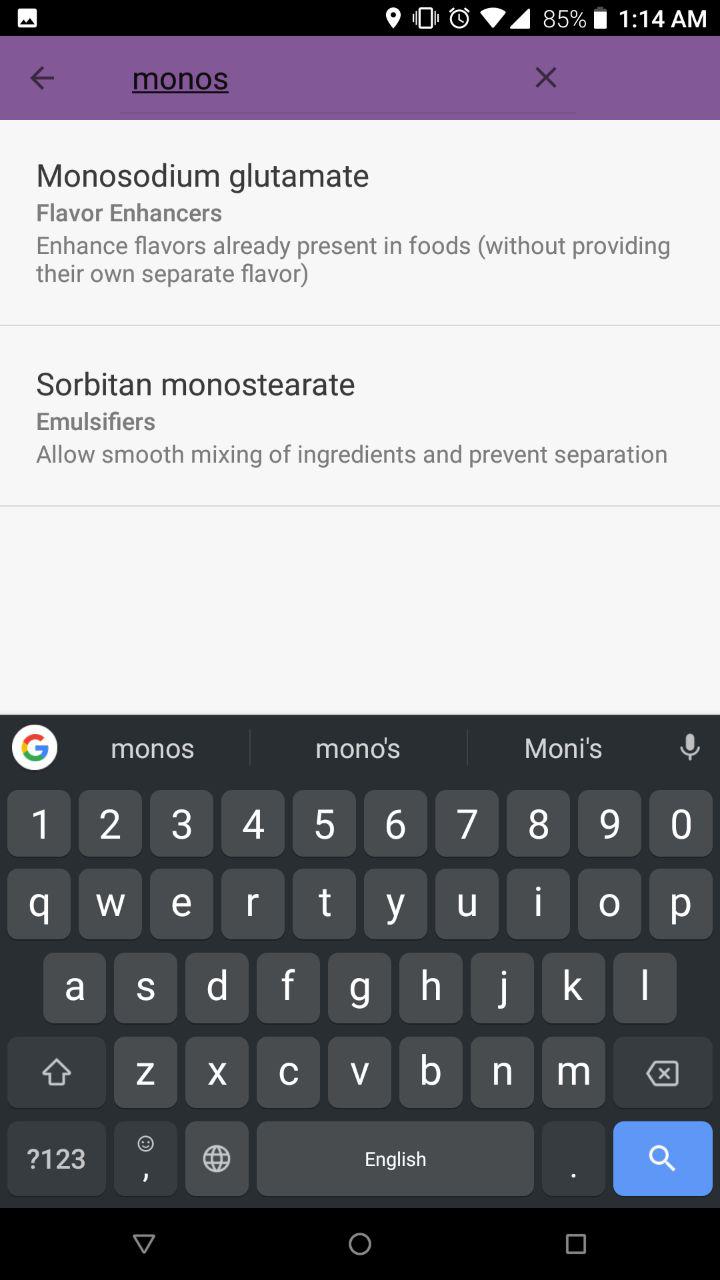
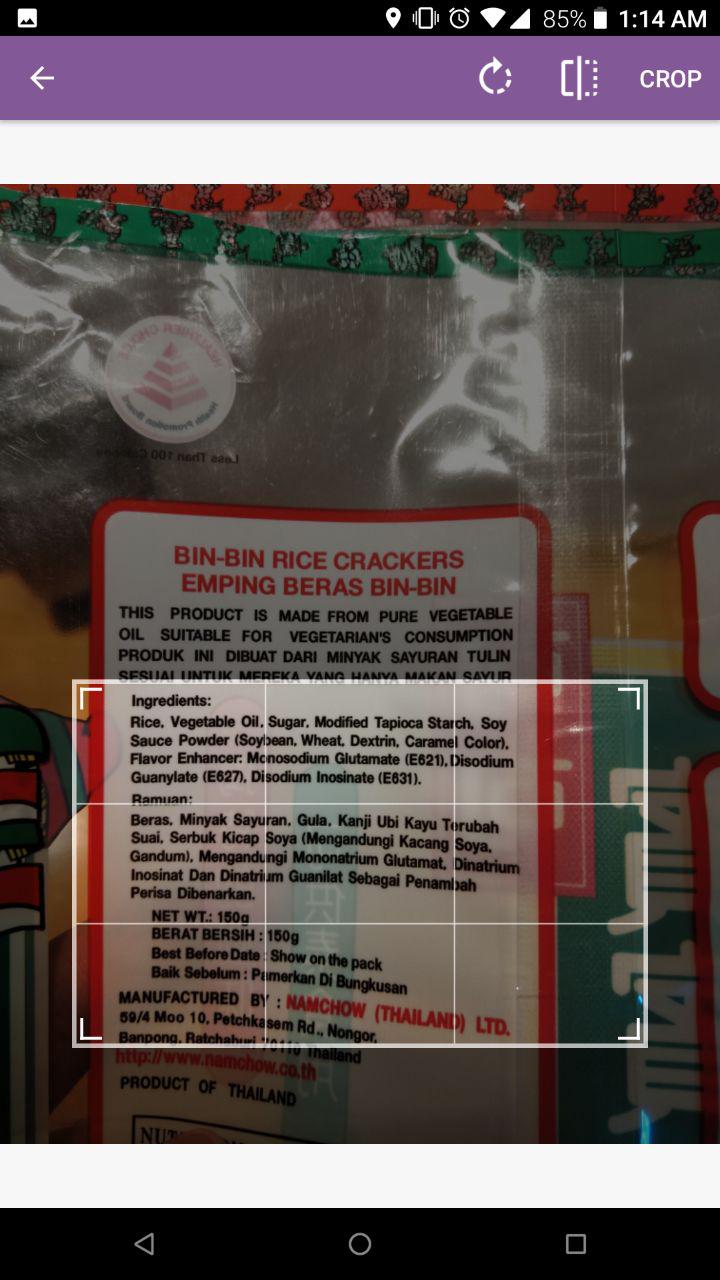
### 7.5 Testing

For the final testing, we will do navigate through the entire application and ensure that every function works as intended and that the bugs are fixed.

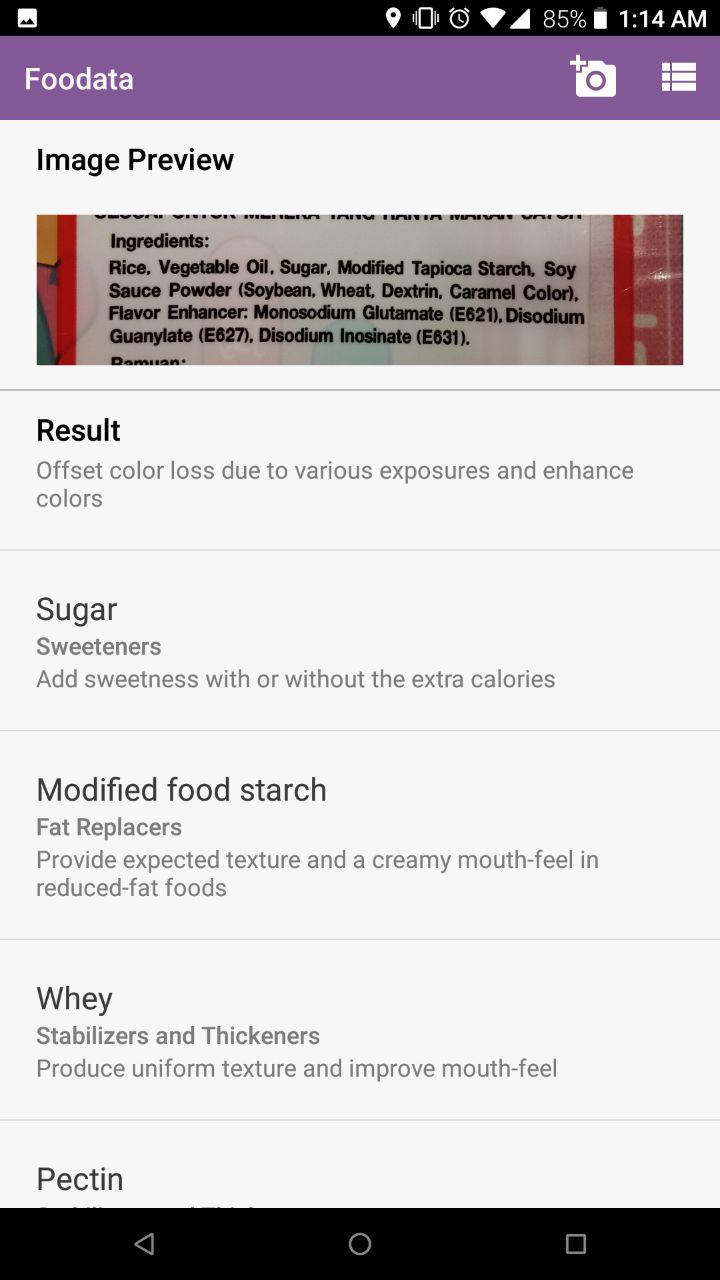
*Figure 7.5.1 Finalized Launch Page Figure 7.5.2 Database page*

As shown in *figure 7.5.1*, the keyboard no longer appear on app launch and the cursor is now hidden. The order of the image preview and results was also swapped as it looks cleaner aesthetically for the users. *Figure 7.5.2* demonstrates that the database populates without any issues and its ingredient data are presented neatly.

*Figure 7.5.3 Search Function Figure 7.5.4 Crop image activity*

*Figure 7.5.3* shows that the search function is working. *Figure 7.5.4* demonstrates how the cropping image works. User will scale the grid box to fit whatever information they want and press crop on the top right corner once they are done. The circle arrow icon on the toolbar rotates the image while the square icon flips the image horizontally/vertically.

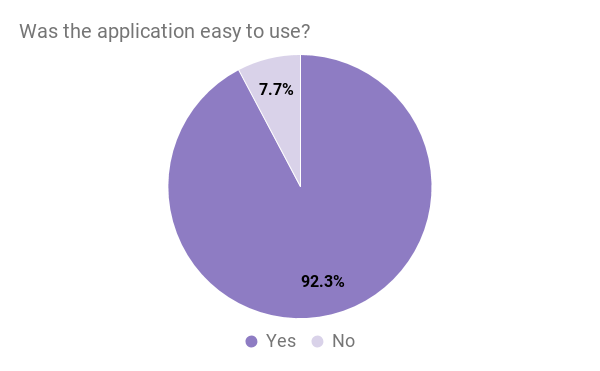


*Figure 7.5 Finalized Results page*

The figure above depicts how the results is supposed to look like. The image preview is at the top with little whitespace around it. The result is shown in a list view below the image preview and it is scrollable. A full demonstration on how the app function can be found in this link provided: <https://www.youtube.com/watch?v=oFw1YLyo4R8&feature=youtu.be>

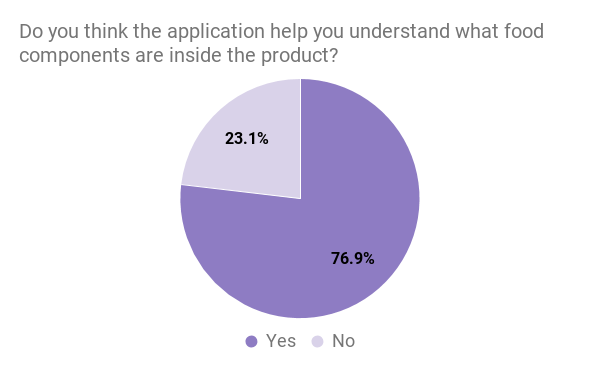
## Results

After completing the application, 13 users aged 22 to 33 was consulted to test whether the application achieve its aims and objectives. Users were briefed that the app is used to analyze ingredients on product label and then they were handed a food package along with the phone the application was installed on. No other instructions were given to test whether the user interface was intuitive enough for users to navigate.



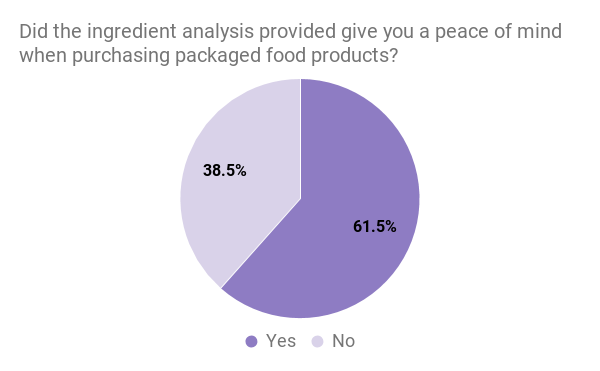
*Figure 8.1 First Question*

The users were firstly asked on the ease of use of the application. A large majority answered “Yes” which suggest that the application’s design was intuitive and user-friendly



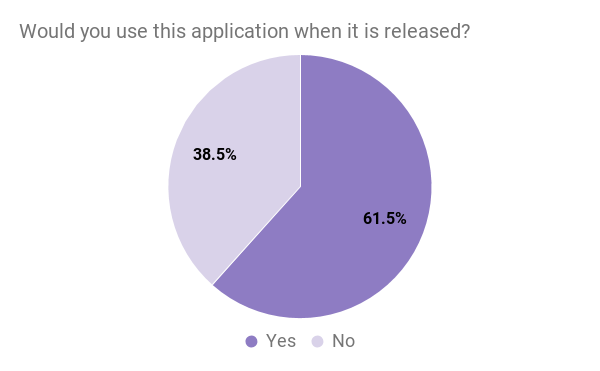
*Figure 8.2 Second Question*

The second question can be found in aims and objectives section of the report. The answers from the users would determine if the objectives were fulfilled. A majority of users think that the application did help them understand the uses of each ingredients in the packaging. Many users agree that the application achieved its aim and would consider this application to be successful in what it was striving to do.



*Figure 8.3 Third Question*

The third question seeks to find an answer to the question that was asked in the objectives section of the report. Similar to the previous question, the application garners a positive response.



*Figure 8.4 Fourth Question*

The last question was to find out users’ willingness to use this application if it was available and more than 60% of the participants gave their positive support for this application. In conclusion, the general impression of the application is positive.

## Discussion

Based on the results section, it was heartening to see that majority of users have a positive response for the application. It was good to see that the product was easy for users to navigate. They were able to find information and locate what they want instantly without any prompts or instructions.

However, the application may gather more positive supports and willingness from users to download the application if the second core feature was able to work.

## Conclusion

Overall, the project was successful in achieving its aims and objectives of a visual interpretational aid for nutrition label. The final product was intuitive and user-friendly. This success can be credited to proper planning during the design phase, where I try to utilize knowledge from the Interaction Design module.

Even though the delivered application is the final product of this project, it is not yet completed and still have room for improvements. If there was more time to experiment and research, I might be able to find a solution for the display of nutrition table in a non-arbitrary order. Once this obstacle is overcome, the integration of this feature into the existing mobile application should be easy.

In the future, I would preferably create a web database API to be called instead of using a local database in the application. This is because if I were to expand the database, it would require the users to update their application and it will take up huge amount of the users’ phone storages.

Furthermore, there are food products that do not have an English nutrition label. A feature that can be incorporated in the future would be to include translations. It would be great for users, especially those with food allergies, when they are travelling abroad and have no idea what the food product contains.

## Evaluation

This project has provided me with a lot of insights about image-to-text technology, nutrition label and consumers’ mindset. It showed me how every detail matter in mobile application.

I adopted the Cowboy methodology for this project because it was designed for solo developers. I believe that this methodology helped kept me on task and ensure I delivered on time. The project Gantt chart was also useful in time management. The 2 weeks iteration cycle constantly kept me on my toes and when I complete a task ahead of time, it gave me a sense of accomplishment. The Gantt Chart can be found in the appendix.

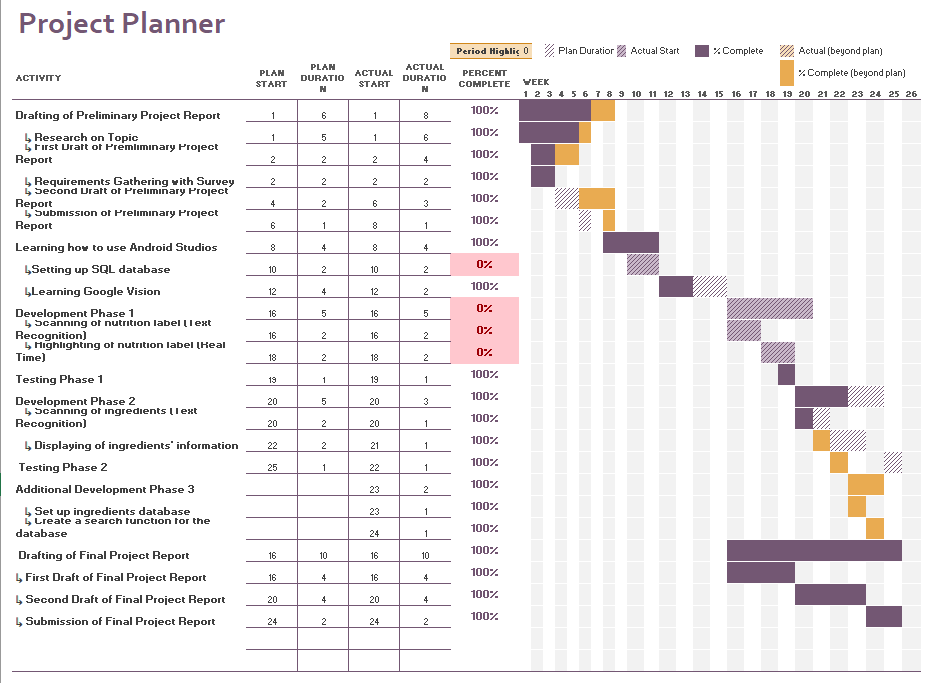
I personally think that unless governing bodies or corporations repackaged their product label to be more user-friendly. However, in my opinion, they will not be totally transparent about the contents in their product because it might contain something that would affect their reputation and it is understandable that the protect their best interest. Therefore, the demand for interpretational aids like my project will grow.

This project was both productive and enlightening for me. This is the first ever android mobile application that I have created, and it was rewarding to see the Java language that I have learnt during the course of my education put into good use. It has given me a direction on what career path I would like to take. I would continue developing application in the future.

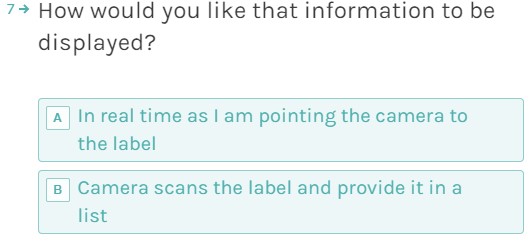
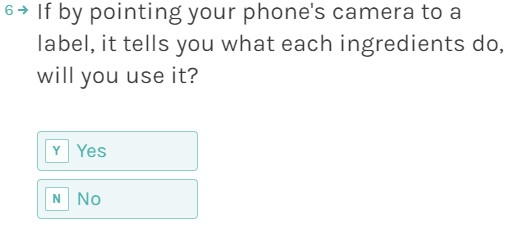
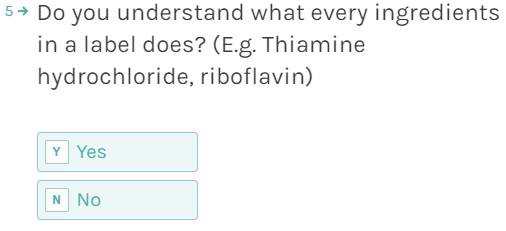
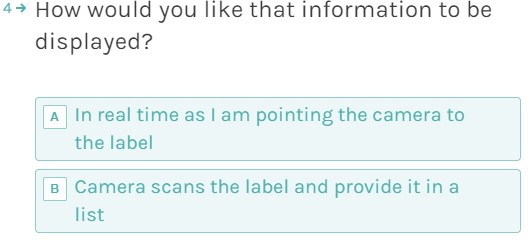
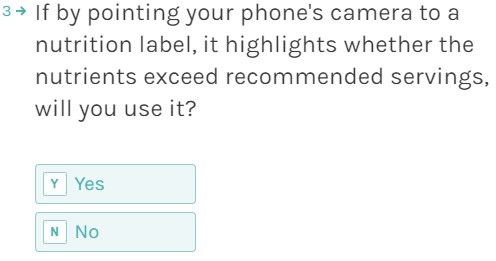
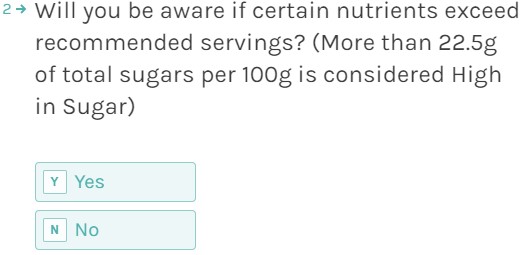
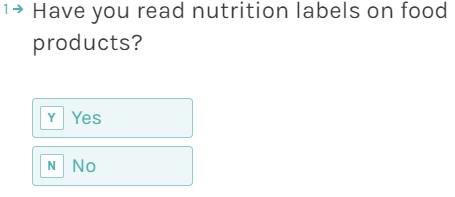
Through this project, I was pushed to improve fast, learn fast and use all the knowledge that I have gathered over the year as a Creative Computing Student.

## Appendices

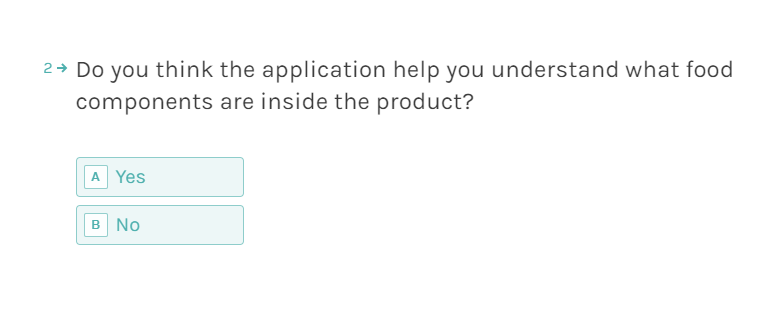
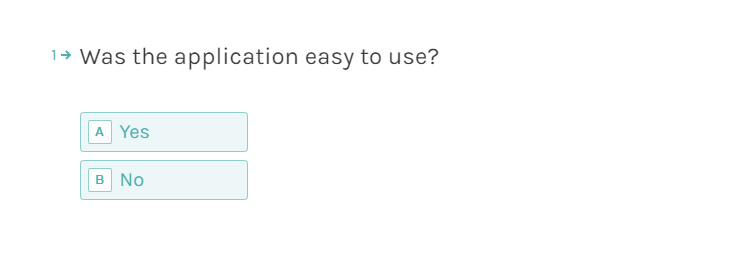
### Enlarged Version of Gantt Chart

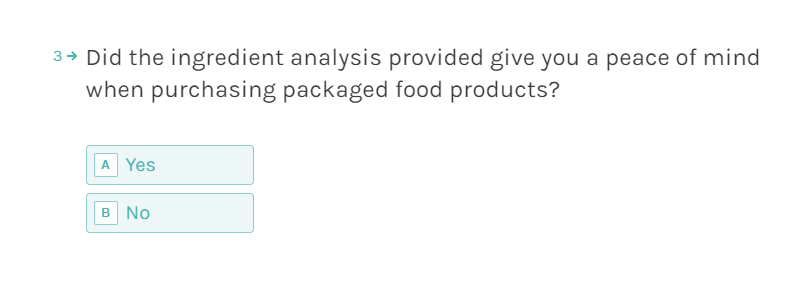


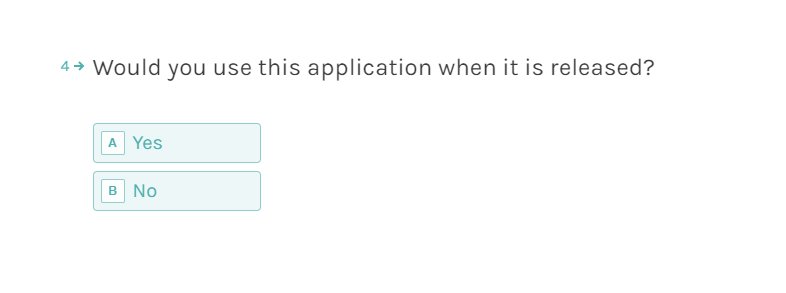
### 12.2 Requirements Survey Questionnaire



### 12.3 End-User Survey Questionnaire







### 12.4 AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.niger.nigerangyidong\_150271176\_co3320">  
  
 <!--Ask user for permission to camera and storage-->  
 <uses-permission android:name="android.permission.CAMERA"/>  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
 <activity  
 android:name=".MainActivity"  
 android:label="@string/app\_name"  
 android:theme="@style/AppTheme">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity  
 android:name=".TableActivity"  
 android:label="Ingredients Database"  
 android:theme="@style/AppTheme"  
 android:parentActivityName=".MainActivity" >  
 <!-- Parent activity meta-data to support 4.0 and lower -->  
 <meta-data  
 android:name="android.support.PARENT\_ACTIVITY"  
 android:value=".MainActivity" />  
 <!--Search function in database page-->  
 <meta-data android:name="android.app.searchable"  
 android:resource="@xml/searchable" />  
 </activity>  
 <!--Crop image manifest-->  
 <activity android:name="com.theartofdev.edmodo.cropper.CropImageActivity"  
 android:theme="@style/CropTheme"  
 />  
 </application>  
  
</manifest>

### 12.5 MainActivity.java

package com.example.niger.nigerangyidong\_150271176\_co3320;  
  
import android.Manifest;  
import android.content.ContentValues;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.graphics.Bitmap;  
import android.graphics.drawable.BitmapDrawable;  
import android.net.Uri;  
import android.provider.MediaStore;  
import android.support.annotation.NonNull;  
import android.support.annotation.Nullable;  
  
import android.support.v4.app.ActivityCompat;  
import android.support.v4.content.ContextCompat;  
import android.support.v7.app.ActionBar;  
import android.support.v7.widget.Toolbar;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.text.TextUtils;  
import android.util.Log;  
import android.util.SparseArray;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.WindowManager;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.Toast;  
  
import com.google.android.gms.vision.Frame;  
import com.google.android.gms.vision.L;  
import com.google.android.gms.vision.text.Text;  
import com.google.android.gms.vision.text.TextBlock;  
import com.google.android.gms.vision.text.TextRecognizer;  
import com.theartofdev.edmodo.cropper.CropImage;  
import com.theartofdev.edmodo.cropper.CropImageView;  
  
import java.io.InputStream;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Collections;  
import java.util.List;  
import java.util.Objects;  
  
public class MainActivity extends AppCompatActivity{  
 ImageView imagePreview ;  
 private FoodArrayAdapter resultsArrayAdapter;  
 private ListView resultsListView;  
  
 private static final int *CAMERA\_REQUEST\_CODE* = 200;  
 private static final int *STORAGE\_REQUEST\_CODE* = 400;  
 private static final int *IMAGE\_PICK\_GALLERY\_CODE* = 1000;  
 private static final int *IMAGE\_PICK\_CAMERA\_CODE* = 1001;  
  
 String cameraPermission[];  
 String storagePermission[];  
  
 Uri image\_uri;  
 List<String[]> foodList;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 //Set up the navigation bar to be able to customize menu  
 Toolbar mToolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(mToolbar);  
  
 //Find view for respective sections  
 //resultEditView = findViewById(R.id.resultEt);  
 imagePreview = findViewById(R.id.*imageIv*);  
  
 //Populate List View  
 resultsListView = findViewById(R.id.*list*);  
 //Create an empty view for when there are no data in database  
 View emptyView = findViewById(R.id.*empty\_view*);  
 resultsListView.setEmptyView(emptyView);  
  
 // Setup an Adapter to create a list item for each row  
 resultsArrayAdapter = new FoodArrayAdapter(getApplicationContext(), R.layout.*list\_item*);  
 resultsListView.setAdapter(resultsArrayAdapter);  
  
 //Hide keyboard when app starts  
 getWindow().setSoftInputMode(WindowManager.LayoutParams.*SOFT\_INPUT\_STATE\_HIDDEN*);  
  
 //camera permission  
 cameraPermission = new String[] {Manifest.permission.*CAMERA*, Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*};  
 //storage permission  
 storagePermission = new String[] {Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*};  
  
 //Access and stream res/raw/data.csv  
 //Load into FoodArray with CSVReader class  
 InputStream inputStream = getResources().openRawResource(R.raw.*data*);  
 CSVReader csv = new CSVReader(inputStream);  
 foodList = csv.read();  
 }  
  
 //toolbar menu  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 //inflate menu  
 getMenuInflater().inflate(R.menu.*main\_menu*, menu);  
 return true;  
 }  
 //handle menu item clicks, response accordingly to whichever option is selected  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id = item.getItemId();  
 switch (id){  
 case R.id.*action\_add\_image*:  
 showImageImportDialog();  
 break;  
 case R.id.*action\_database*:  
 Intent intent = new Intent(getApplicationContext(), TableActivity.class);  
 startActivity(intent);  
 break;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 private void showImageImportDialog() {  
 //items to display in dialog  
 String[] items = {" Camera", " Gallery"};  
 AlertDialog.Builder dialog = new AlertDialog.Builder(this);  
 //set title  
 dialog.setTitle("Select Image");  
 dialog.setItems(items, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if(which == 0){  
 //camera option clicked  
 if(!checkCameraPermission()){  
 //camera permission not allowed, request it  
 requestCameraPermission();  
 }else{  
 //camera permission allowed  
 pickCamera();  
 }  
 }  
 if(which == 1){  
 //gallery option clicked  
 if(!checkStoragePermission()){  
 //storage permission not allowed  
 requestStoragePermission();  
 }else{  
 //storage permission allowed  
 pickGallery();  
 }  
 }  
 }  
 });  
 dialog.create().show(); //show dialog  
 }  
  
 private boolean checkCameraPermission() {  
 boolean cameraResult = ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*CAMERA*) == (PackageManager.*PERMISSION\_GRANTED*);  
 boolean storageResult = ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == (PackageManager.*PERMISSION\_GRANTED*);  
 return cameraResult && storageResult;  
 }  
  
 private void requestCameraPermission(){  
 ActivityCompat.*requestPermissions*(this, cameraPermission, *CAMERA\_REQUEST\_CODE*);  
 }  
  
 private void pickCamera() {  
 //take image from camera, save to storage  
 ContentValues values = new ContentValues();  
 values.put(MediaStore.Images.Media.*TITLE*, "Scanned\_Label"); // title of the picture  
 values.put(MediaStore.Images.Media.*DESCRIPTION*, "Image To text"); //description of picture  
 image\_uri = getContentResolver().insert(MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*, values);  
  
 Intent cameraIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 cameraIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, image\_uri);  
 startActivityForResult(cameraIntent, *IMAGE\_PICK\_CAMERA\_CODE*);  
 }  
  
 private boolean checkStoragePermission() {  
 boolean storageResult = ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == (PackageManager.*PERMISSION\_GRANTED*);  
 return storageResult;  
 }  
  
 private void requestStoragePermission(){  
 ActivityCompat.*requestPermissions*(this, storagePermission, *STORAGE\_REQUEST\_CODE*);  
 }  
  
 private void pickGallery() {  
 //take image from storage  
 Intent galleryIntent = new Intent(Intent.*ACTION\_PICK*);  
 //set intent type to image  
 galleryIntent.setType("image/\*");  
 startActivityForResult(galleryIntent, *IMAGE\_PICK\_GALLERY\_CODE*);  
 }  
  
 //handle permission  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 switch (requestCode){  
 case *CAMERA\_REQUEST\_CODE*:  
 if(grantResults.length > 0){  
 boolean cameraAccepted = grantResults[0] == PackageManager.*PERMISSION\_GRANTED*;  
 boolean writeStorageAccepted = grantResults[0] == PackageManager.*PERMISSION\_GRANTED*;  
  
 if(cameraAccepted && writeStorageAccepted){  
 pickCamera();  
 }else{  
 Toast.*makeText*(this,"Permission denied", Toast.*LENGTH\_SHORT*);  
 }  
 }  
 break;  
  
 case *STORAGE\_REQUEST\_CODE*:  
 if(grantResults.length > 0){  
 boolean writeStorageAccepted = grantResults[0] == PackageManager.*PERMISSION\_GRANTED*;  
  
 if(writeStorageAccepted){  
 pickGallery();  
 }else{  
 Toast.*makeText*(this,"Permission denied", Toast.*LENGTH\_SHORT*);  
 }  
 }  
 break;  
 }  
 }  
  
 //handle image result  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
  
 if(resultCode == *RESULT\_OK*){  
 if(requestCode == *IMAGE\_PICK\_GALLERY\_CODE*){  
 //image from gallery and enable grid lines  
 CropImage.*activity*(data.getData()).setGuidelines(CropImageView.Guidelines.*ON*).start(this);  
 }  
 if(requestCode == *IMAGE\_PICK\_CAMERA\_CODE*){  
 //image from camera  
 CropImage.*activity*(image\_uri).setGuidelines(CropImageView.Guidelines.*ON*).start(this);  
 }  
 }  
 //get cropped image  
 if(requestCode == CropImage.*CROP\_IMAGE\_ACTIVITY\_REQUEST\_CODE*){  
 CropImage.ActivityResult result = CropImage.*getActivityResult*(data);  
 if(resultCode == *RESULT\_OK*){  
 Uri resultUri = result.getUri(); //get image uri  
 //set image to image view  
 imagePreview.setImageURI(resultUri);  
  
 //get drawable bitmap for text recognition  
 BitmapDrawable bitmapDrawable = (BitmapDrawable) imagePreview.getDrawable();  
 Bitmap bitmap = bitmapDrawable.getBitmap();  
  
 TextRecognizer recognizer = new TextRecognizer.Builder(getApplicationContext()).build();  
  
 if(!recognizer.isOperational()){  
 Toast.*makeText*(this,"Error", Toast.*LENGTH\_SHORT*).show();  
 }else{  
 Frame frame = new Frame.Builder().setBitmap(bitmap).build();  
 SparseArray<TextBlock> items = recognizer.detect(frame);  
 String lines = "";  
 ArrayList<Textholder> textholder = new ArrayList<Textholder>();  
 ArrayList<Textholder> textholder1 = new ArrayList<Textholder>();  
 for(int index = 0; index < items.size(); index ++){  
 int key = items.keyAt(index);  
 TextBlock textBlock = items.valueAt(index);  
 if(textBlock != null){ //if there is a block, get its components  
 for(Text line : textBlock.getComponents()){ //go through each line in the block  
 float topLine = line.getBoundingBox().top;  
 Textholder lineHolder = new Textholder(topLine, line.getValue());  
 textholder1.add(lineHolder);  
 for(Text element: line.getComponents()) {  
 float top = element.getBoundingBox().top; //record the line's box and coordinates  
 Textholder th = new Textholder(top, element.getValue());  
 textholder.add(th);  
 }  
 }  
 }  
 }  
 if(items.size() == 0){ //if no text found  
 Toast.*makeText*(this, "No Text Found", Toast.*LENGTH\_SHORT*).show();  
 }else{  
 Collections.*sort*(textholder);  
 for(int i = 0; i < textholder1.size(); i++){  
 Textholder t = textholder1.get(i);  
 lines = lines + t.getText();  
 }  
  
 //Version 3  
 lines = lines.replaceAll("\\(",",").replaceAll("\\)",",").replaceAll("\\.",",");  
 String [] food = lines.split(",");  
 String [] temp;  
 double [] dist = new double[food.length];  
 double threshold = 0.52;  
 List<String[]> resultList = new ArrayList<String[]>();  
  
 //Loop through scanned results  
 for(int z = 0; z < food.length;z++){  
 String [] resultData = new String[3];  
 resultList.add(null);  
  
 for(int x = 0; x < foodList.size(); x++ ){  
 temp = foodList.get(x);  
 double value = *similarity*(food[z].trim(),temp[2].trim());  
 if(value > threshold){  
 if(value > dist[z]) {  
 dist[z] = value;  
 resultData[0] = temp[0]; //add type of uses to StringArray resultsData  
 resultData[1] = temp[1]; //add purpose  
 resultData[2] = temp[2]; //add name  
 resultList.set(z, resultData);  
 }  
 if(value == 1.000){ //if exact match found, move to next ingredient from the scanned list  
 System.*out*.println(temp[2] + " matched");  
 break ;  
 }  
 }  
 }  
 }  
 //remove all results that yield nothing  
 resultList.removeAll(Collections.*singleton*(null));  
 for(String [] resultData : resultList){  
 resultsArrayAdapter.add(resultData);  
 System.*out*.println(Arrays.*toString*(resultData));  
 }  
 //resultEditView.setText(lines);  
 recognizer.release();  
 }  
 }  
 }  
 else if(resultCode == CropImage.*CROP\_IMAGE\_ACTIVITY\_RESULT\_ERROR\_CODE*){  
 //if there is any error, show it  
 Exception error = result.getError();  
 Toast.*makeText*(this,""+error, Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
 */\*\*  
 \* Calculates the similarity between two strings.  
 \* Exact match = 1.0  
 \* No match = 0.0  
 \*/* public static double similarity(String s1, String s2) {  
 String longer = s1, shorter = s2;  
 if (s1.length() < s2.length()) { // longer should always have greater length  
 longer = s2; shorter = s1;  
 }  
 int longerLength = longer.length();  
 if (longerLength == 0) {  
 return 1.0; /\* both strings are zero length \*/  
 }  
 return (longerLength - *editDistance*(longer, shorter)) / (double)longerLength;  
 }  
  
 // Levenshtein Distance  
 // Source:https://rosettacode.org/wiki/Levenshtein\_distance#Java  
 public static int editDistance(String s1, String s2) {  
 s1 = s1.toLowerCase();  
 s2 = s2.toLowerCase();  
  
 int[] costs = new int[s2.length() + 1];  
 for (int i = 0; i <= s1.length(); i++) {  
 int lastValue = i;  
 for (int j = 0; j <= s2.length(); j++) {  
 if (i == 0)  
 costs[j] = j;  
 else {  
 if (j > 0) {  
 int newValue = costs[j - 1];  
 if (s1.charAt(i - 1) != s2.charAt(j - 1))  
 newValue = Math.*min*(Math.*min*(newValue, lastValue),  
 costs[j]) + 1;  
 costs[j - 1] = lastValue;  
 lastValue = newValue;  
 }  
 }  
 }  
 if (i > 0)  
 costs[s2.length()] = lastValue;  
 }  
 return costs[s2.length()];  
 }  
  
 public static void printSimilarity(String s, String t) {  
 System.*out*.println(String.*format*("%.3f is the similarity between \"%s\" and \"%s\"", *similarity*(s, t), s, t));  
 }  
  
}

### 12.6 TableActivity.java

package com.example.niger.nigerangyidong\_150271176\_co3320;  
  
import android.os.Bundle;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.widget.SearchView;  
import android.support.v7.widget.Toolbar;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.ListView;  
  
import java.io.InputStream;  
import java.util.ArrayList;  
import java.util.List;  
  
public class TableActivity extends AppCompatActivity {  
 //Adapter for ListView  
 private FoodArrayAdapter mArrayAdapter;  
 private ListView foodListView;  
 List<String[]> foodList;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_table*);  
 //set up similar navigation bar to the one in MainActivity  
 Toolbar mToolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(mToolbar);  
 //Create a back button in the navigation bar to navigate back to MainActivity  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
  
 //Populate List View  
 foodListView = findViewById(R.id.*list*);  
 //Create an empty view for when there are no data in database  
 View emptyView = findViewById(R.id.*empty\_view*);  
 foodListView.setEmptyView(emptyView);  
  
 // Setup an Adapter to create a list item for each row  
 mArrayAdapter = new FoodArrayAdapter(getApplicationContext(), R.layout.*list\_item*);  
 foodListView.setAdapter(mArrayAdapter);  
  
 //Access and stream res/raw/data.csv  
 //Load into FoodArray with CSVReader class  
 InputStream inputStream = getResources().openRawResource(R.raw.*data*);  
 CSVReader csv = new CSVReader(inputStream);  
 foodList = csv.read();  
 for(String [] foodData : foodList){  
 mArrayAdapter.add(foodData);  
 }  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu options from the res/menu/table\_menu.xml file  
 getMenuInflater().inflate(R.menu.*table\_menu*, menu);  
  
 // Associate searchable configuration with the SearchView  
 SearchView searchView = (SearchView) menu.findItem(R.id.*search*).getActionView();  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String s) {  
 return false;  
 }  
  
 @Override  
 public boolean onQueryTextChange(String s) {  
 ArrayList<String[]> tempList = new ArrayList<>();  
 FoodArrayAdapter adapter = new FoodArrayAdapter(getApplicationContext(), R.layout.*list\_item*);  
 foodListView.setAdapter(adapter);  
  
 for (String [] temp : foodList) {  
 if(temp[2].toLowerCase().contains(s.toLowerCase())){  
 adapter.add(temp);  
 }  
 }  
 return true;  
 }  
 });  
  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 // User clicked on a menu option in the app bar overflow menu  
 switch (item.getItemId()) {  
 // Respond to a click on the "Insert dummy data" menu option  
 case R.id.*search*:  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

### 12.7 Textholder.java

package com.example.niger.nigerangyidong\_150271176\_co3320;  
//Bug fix https://github.com/googlesamples/android-vision/issues/299  
  
*/\*\*Solution for solving text forming in random order  
 \* Create own list of blocks and assign coordinates to each block  
 \*/*class Textholder implements Comparable<Textholder> {  
  
 private float Y;  
 private String text;  
  
 Textholder(float Y, String text) {  
 this.Y = Y;  
 this.text = text;  
 }  
  
 String getText() {  
 return text;  
 }  
  
 float getY() {  
 return Y;  
 }  
  
 public int compareTo(Textholder th) {  
 if (Y == th.getY()) return 0;  
 else if (Y > th.getY()) return 1;  
 else return -1;  
 }  
}

### 12.8 FoodArrayAdapter.java

package com.example.niger.nigerangyidong\_150271176\_co3320;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.TextView;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class FoodArrayAdapter extends ArrayAdapter<String[]> {  
  
 private List<String[]> foodList = new ArrayList<>();  
  
 static class FoodViewHolder{  
 TextView Type;  
 TextView Purpose;  
 TextView Name;  
 }  
 public FoodArrayAdapter(Context context, int resource) {  
 super(context, resource);  
 }  
  
 //method to add objects into foodList array  
 public void add(String[] object){  
 foodList.add(object);  
 super.add(object);  
 }  
  
 //Find the size of the array  
 @Override  
 public int getCount(){  
 return this.foodList.size();  
 }  
  
 //find item based on position in the array  
 @Override  
 public String[] getItem(int position){  
 return this.foodList.get(position);  
 }  
  
 public View getView(int position, View convertView, ViewGroup parent){  
 View group = convertView;  
 FoodViewHolder viewHolder;  
 if(group == null){  
 LayoutInflater inflater = (LayoutInflater) this.getContext().getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);  
 group = inflater.inflate(R.layout.*list\_item*, parent, false);  
 viewHolder = new FoodViewHolder();  
 viewHolder.Type = group.findViewById(R.id.*type*);  
 viewHolder.Purpose= group.findViewById(R.id.*purpose*);  
 viewHolder.Name= group.findViewById(R.id.*name*);  
 group.setTag(viewHolder);  
 }else{  
 viewHolder = (FoodViewHolder) group.getTag();  
 }  
  
 String[] stat = getItem(position);  
 viewHolder.Type.setText(stat[0]);  
 viewHolder.Purpose.setText(stat[1]);  
 viewHolder.Name.setText(stat[2]);  
 return group;  
 }  
}

### 12.9 CSVReader.java

package com.example.niger.nigerangyidong\_150271176\_co3320;  
*/\*\*  
 Learnt and adapted from "How to read CSV files in android application"  
 By Indragni Soft Solutions  
 https://www.youtube.com/watch?v=svxKakcKnxE  
 \*\*/*import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* Access csv file and read it line by line  
 \* Seperate values by comma  
 \*/*public class CSVReader {  
 InputStream inputStream;  
  
 public CSVReader(InputStream is){  
 this.inputStream = is;  
 }  
  
 public List<String[]> read(){  
 List<String[]> resultList = new ArrayList<>();  
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));  
  
 try{  
 String line;  
 while((line = reader.readLine()) != null){  
 String[] row = line.split(",");  
 resultList.add(row);  
 }  
  
 }catch(IOException e){  
 throw new RuntimeException("Error in reading csv file:" + e);  
 }finally {  
 try{  
 inputStream.close();  
 }catch (IOException e){  
 throw new RuntimeException("Error while closing input stream: " + e);  
 }  
 }  
 return resultList;  
 }  
}

### 12.10 activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
  
<!--Design For Main page(MainActivity.java)-->  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:id="@+id/ImageLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="64dp"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Image Preview"  
 android:fontFamily="sans-serif-medium"  
 android:textColor="@color/colorBlack"  
 android:paddingTop="4dp"  
 android:paddingLeft="24dp"  
 android:textSize="20dp" />  
 <!--image taken will be displayed here-->  
 <ImageView  
 android:id="@+id/imageIv"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:adjustViewBounds="true"  
 android:maxHeight="200dp"  
 android:paddingTop="24dp"  
 android:paddingLeft="24dp"  
 android:paddingRight="24dp"/>  
  
 <View  
 android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="16dp"  
 android:background="@android:color/darker\_gray" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_below="@+id/ImageLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:layout\_marginTop="8dp">  
  
 <TextView  
 android:id="@+id/result"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif-medium"  
 android:text="Result"  
 android:textColor="@color/colorBlack"  
 android:paddingTop="4dp"  
 android:paddingLeft="24dp"  
 android:paddingBottom="4dp"  
 android:textSize="20dp" />  
 <!--Scanned text will be display here-->  
 <ListView  
 android:id="@+id/list"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@+id/result" />  
  
 <!-- Empty view for the list -->  
 <RelativeLayout  
 android:id="@+id/empty\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true">  
  
 <TextView  
 android:id="@+id/empty\_title\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:paddingTop="16dp"  
 android:paddingLeft="24dp"  
 android:text="Click the camera icon to start"  
 android:textAppearance="?android:textAppearanceMedium" />  
  
 </RelativeLayout>  
 </LinearLayout>  
  
 <!--Navigation bar at top of page and menu-->  
 <android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/colorPurple"  
 app:titleTextColor="@color/colorWhite"  
 android:minHeight="54dp"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light"  
 app:theme="@style/AppTheme" />  
</RelativeLayout>

### 12.11 activity\_table.xml

<?xml version="1.0" encoding="utf-8"?>  
  
<!--Design For Main page(MainActivity.java)-->  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:id="@+id/ImageLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="64dp"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Image Preview"  
 android:fontFamily="sans-serif-medium"  
 android:textColor="@color/colorBlack"  
 android:paddingTop="4dp"  
 android:paddingLeft="24dp"  
 android:textSize="20dp" />  
 <!--image taken will be displayed here-->  
 <ImageView  
 android:id="@+id/imageIv"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:adjustViewBounds="true"  
 android:maxHeight="200dp"  
 android:paddingTop="24dp"  
 android:paddingLeft="24dp"  
 android:paddingRight="24dp"/>  
  
 <View  
 android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="16dp"  
 android:background="@android:color/darker\_gray" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_below="@+id/ImageLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:layout\_marginTop="8dp">  
  
 <TextView  
 android:id="@+id/result"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif-medium"  
 android:text="Result"  
 android:textColor="@color/colorBlack"  
 android:paddingTop="4dp"  
 android:paddingLeft="24dp"  
 android:paddingBottom="4dp"  
 android:textSize="20dp" />  
 <!--Scanned text will be display here-->  
 <ListView  
 android:id="@+id/list"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@+id/result" />  
  
 <!-- Empty view for the list -->  
 <RelativeLayout  
 android:id="@+id/empty\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true">  
  
 <TextView  
 android:id="@+id/empty\_title\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:paddingTop="16dp"  
 android:paddingLeft="24dp"  
 android:text="Click the camera icon to start"  
 android:textAppearance="?android:textAppearanceMedium" />  
  
 </RelativeLayout>  
 </LinearLayout>  
  
 <!--Navigation bar at top of page and menu-->  
 <android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/colorPurple"  
 app:titleTextColor="@color/colorWhite"  
 android:minHeight="54dp"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light"  
 app:theme="@style/AppTheme" />  
</RelativeLayout>

### 12.12 list\_item.xml

<?xml version="1.0" encoding="utf-8"?>  
<!--Design template for how data in database should be displayed in a listView-->  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="24dp">  
  
 <TextView  
 android:id="@+id/name"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif"  
 android:textAppearance="?android:textAppearanceMedium"  
 android:textColor="@color/colorDarkGray" />  
  
 <TextView  
 android:id="@+id/type"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif-medium"  
 android:textAppearance="?android:textAppearanceSmall"  
 android:textColor="@color/colorGray" />  
  
 <TextView  
 android:id="@+id/purpose"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif"  
 android:textAppearance="?android:textAppearanceSmall"  
 android:textColor="@color/colorGray" />  
</LinearLayout>

### 12.13 main\_menu.xml

<?xml version="1.0" encoding="utf-8"?>  
  
<!--Design for navigation bar/menu in Main page (MainActivity.java)-->  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
 <item  
 android:id="@+id/action\_add\_image"  
 android:icon="@drawable/ic\_add\_a\_photo\_white\_32dp"  
 android:title="Add Image"  
 app:showAsAction="ifRoom" />  
 <item  
 android:id="@+id/action\_database"  
 android:icon="@drawable/ic\_view\_list\_white\_32dp"  
 android:title="Database"  
 app:showAsAction="ifRoom" />  
</menu>

### 12.14 table\_menu.xml

<?xml version="1.0" encoding="utf-8"?>  
<!--https://developer.android.com/training/search/setup.html-->  
  
<!--Design Template navigation bar/menu in database page(TableActivity.java)-->  
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
 <item  
 android:id="@+id/search"  
 android:icon="@drawable/ic\_search\_white\_32dp"  
 android:title="Search"  
 app:showAsAction="always|collapseActionView"  
 app:actionViewClass="android.support.v7.widget.SearchView"  
 />  
</menu>

### 12.15 strings.xml

<resources>  
 <string name="app\_name">Foodata</string>  
 <string name="database\_name">Ingredients Database</string>  
</resources>

### 12.16 styles.xml

<resources>  
  
 <!-- Base application theme. -->  
 <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">  
 <!-- Customize your theme here. -->  
 <item name="colorPrimary">@color/colorPurple</item> <!--ToolBar-->  
 <item name="colorPrimaryDark">@color/colorBlack</item> <!--notification bar-->  
 <item name="colorAccent">@color/colorWhite</item>  
 <item name="android:windowBackground">@color/colorWhite</item> <!--Background-->  
 </style>  
 <!--Theme at the cropping page-->  
 <style name="CropTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
 <!-- Customize your theme here. -->  
 <item name="colorPrimary">@color/colorPurple</item> <!--ToolBar-->  
 <item name="colorPrimaryDark">@color/colorBlack</item> <!--notification bar-->  
 <item name="colorAccent">@color/colorWhite</item>  
 <item name="android:windowBackground">@color/colorWhite</item> <!--Background-->  
 </style>  
</resources>

### 12.17 colors.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <color name="colorBlack">#000000</color>  
 <color name="colorSilver">#BEC0C2</color>  
 <color name="colorBlue">#006BB6</color>  
 <color name="colorOrange">#F58426</color>  
 <color name="colorWhite">#F7F7F7</color>  
 <color name="colorPurple">#835896</color>  
 <color name="colorDarkGray">#3A3A3A</color>  
 <color name="colorGray">#7C7C7C</color>  
</resources>

### 12.18 searchable.xml

<?xml version="1.0" encoding="utf-8"?>  
  
<searchable xmlns:android="http://schemas.android.com/apk/res/android"  
 android:label="@string/database\_name"  
 android:hint="Search" />

### 12.19 build.gradle (Module:app)

apply plugin: 'com.android.application'  
  
android {  
 compileSdkVersion 28  
 defaultConfig {  
 applicationId "com.example.niger.nigerangyidong\_150271176\_co3320"  
 minSdkVersion 16  
 targetSdkVersion 28  
 versionCode 1  
 versionName "1.0"  
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
 }  
 buildTypes {  
 release {  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 }  
 }  
}  
  
dependencies {  
 implementation fileTree(dir: 'libs', include: ['\*.jar'])  
 implementation 'com.android.support:appcompat-v7:28.0.0'  
 implementation 'com.android.support:cardview-v7:28.0.0'  
 implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
 testImplementation 'junit:junit:4.12'  
 //androidTestImplementation 'com.android.support.test:runner:1.0.2'  
 //androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'\  
  
 //Android Image Cropper Library Source: https://github.com/ArthurHub/Android-Image-Cropper  
 implementation 'com.theartofdev.edmodo:android-image-cropper:2.7.0'  
 //Google image to text library  
 implementation 'com.google.android.gms:play-services-vision:17.0.2'  
}

### 12.20 data.csv

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | Types of Ingredients | What They Do | Names Found on Product Labels |
| 1 | Preservatives | Prevent food spoilage | Ascorbic acid |
| 2 | Preservatives | Prevent food spoilage | Citric acid |
| 3 | Preservatives | Prevent food spoilage | Sodium benzoate |
| 4 | Preservatives | Prevent food spoilage | Calcium propionate |
| 5 | Preservatives | Prevent food spoilage | Sodium erythorbate |
| 6 | Preservatives | Prevent food spoilage | Sodium nitrite |
| 7 | Preservatives | Prevent food spoilage | Calcium sorbate |
| 8 | Preservatives | Prevent food spoilage | Potassium sorbate |
| 9 | Preservatives | Prevent food spoilage | BHA |
| 10 | Preservatives | Prevent food spoilage | BHT |
| 11 | Preservatives | Prevent food spoilage | EDTA |
| 12 | Preservatives | Prevent food spoilage | Tocopherols (Vitamin E) |
| 13 | Preservatives | Prevent food spoilage | Tocopherols |
| 14 | Sweeteners | Add sweetness with or without the extra calories | Sucrose |
| 15 | Sweeteners | Add sweetness with or without the extra calories | Sugar |
| 16 | Sweeteners | Add sweetness with or without the extra calories | Glucose |
| 17 | Sweeteners | Add sweetness with or without the extra calories | Fructose |
| 18 | Sweeteners | Add sweetness with or without the extra calories | Sorbitol |
| 19 | Sweeteners | Add sweetness with or without the extra calories | Mannitol |
| 20 | Sweeteners | Add sweetness with or without the extra calories | Corn syrup |
| 21 | Sweeteners | Add sweetness with or without the extra calories | Saccharin |
| 22 | Sweeteners | Add sweetness with or without the extra calories | Aspartame |
| 23 | Sweeteners | Add sweetness with or without the extra calories | Sucralose |
| 24 | Sweeteners | Add sweetness with or without the extra calories | Acesulfame potassium |
| 25 | Sweeteners | Add sweetness with or without the extra calories | Acesulfame-K |
| 26 | Sweeteners | Add sweetness with or without the extra calories | Neotame |
| 27 | Color Additives | Offset color loss due to various exposures and enhance colors | FD&C Blue Nos. 3 |
| 28 | Color Additives | Offset color loss due to various exposures and enhance colors | FD&C Blue Nos. 4 |
| 29 | Color Additives | Offset color loss due to various exposures and enhance colors | FD&C Green No. 5 |
| 30 | Color Additives | Offset color loss due to various exposures and enhance colors | FD&C Red Nos. 5 |
| 31 | Color Additives | Offset color loss due to various exposures and enhance colors | FD&C Red Nos. 42 |
| 32 | Color Additives | Offset color loss due to various exposures and enhance colors | FD&C Yellow Nos. 7 |
| 33 | Color Additives | Offset color loss due to various exposures and enhance colors | FD&C Yellow Nos. 8 |
| 34 | Color Additives | Offset color loss due to various exposures and enhance colors | F Orange B |
| 35 | Color Additives | Offset color loss due to various exposures and enhance colors | Citrus Red No. 4 |
| 36 | Color Additives | Offset color loss due to various exposures and enhance colors | Annatto extract |
| 37 | Color Additives | Offset color loss due to various exposures and enhance colors | Beta-carotene |
| 38 | Color Additives | Offset color loss due to various exposures and enhance colors | Grape skin extract |
| 39 | Color Additives | Offset color loss due to various exposures and enhance colors | Cochineal Carmine |
| 40 | Color Additives | Offset color loss due to various exposures and enhance colors | Cochineal extract |
| 41 | Color Additives | Offset color loss due to various exposures and enhance colors | Carmine |
| 42 | Color Additives | Offset color loss due to various exposures and enhance colors | Paprika oleoresin |
| 43 | Color Additives | Offset color loss due to various exposures and enhance colors | Caramel color |
| 44 | Color Additives | Offset color loss due to various exposures and enhance colors | Fruit juice |
| 45 | Color Additives | Offset color loss due to various exposures and enhance colors | Vegetable juice |
| 46 | Color Additives | Offset color loss due to various exposures and enhance colors | Saffron |
| 47 | Color Additives | Offset color loss due to various exposures and enhance colors | Colorings |
| 48 | Color Additives | Offset color loss due to various exposures and enhance colors | Color |
| 49 | Flavors and Spices | Add specific flavors (natural and synthetic) | Natural flavoring |
| 50 | Flavors and Spices | Add specific flavors (natural and synthetic) | Artificial flavor |
| 51 | Flavors and Spices | Add specific flavors (natural and synthetic) | Spices |
| 52 | Flavor Enhancers | Enhance flavors already present in foods (without providing their own separate flavor) | Monosodium glutamate |
| 53 | Flavor Enhancers | Enhance flavors already present in foods (without providing their own separate flavor) | MSG |
| 54 | Flavor Enhancers | Enhance flavors already present in foods (without providing their own separate flavor) | Hydrolyzed soy protein |
| 55 | Flavor Enhancers | Enhance flavors already present in foods (without providing their own separate flavor) | Autolyzed yeast extract |
| 56 | Flavor Enhancers | Enhance flavors already present in foods (without providing their own separate flavor) | Disodium guanylate |
| 57 | Flavor Enhancers | Enhance flavors already present in foods (without providing their own separate flavor) | Disodium inosinate |
| 58 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Olestra |
| 59 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Cellulose gel |
| 60 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Carrageenan |
| 61 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Polydextrose |
| 62 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Modified food starch |
| 63 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Modified starch |
| 64 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Food starch |
| 65 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Starch |
| 66 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Microparticulated egg white protein |
| 67 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Guar gum |
| 68 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Xanthan gum |
| 69 | Fat Replacers | Provide expected texture and a creamy mouth-feel in reduced-fat foods | Whey protein concentrate |
| 70 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Thiamine hydrochloride |
| 71 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Riboflavin (Vitamin B2) |
| 72 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Riboflavin |
| 73 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Niacin |
| 74 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Niacinamide |
| 75 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Folate |
| 76 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Folic acid |
| 77 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Beta carotene |
| 78 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Potassium iodide |
| 79 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Iron sulfate |
| 80 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Ferrous sulfate |
| 81 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Alpha tocopherols |
| 82 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Ascorbic acid |
| 83 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Vitamin D |
| 84 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Amino acids (L-tryptophan, L-lysine, L-leucine, L-methionine) |
| 85 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | L-tryptophan |
| 86 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | L-lysine |
| 87 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | L-leucine |
| 88 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | L-methionine |
| 89 | Nutrients | Replace vitamins and minerals lost in processing (enrichment) | Amino acids |
| 90 | Emulsifiers | Allow smooth mixing of ingredients and prevent separation | Soy lecithin |
| 91 | Emulsifiers | Allow smooth mixing of ingredients and prevent separation | Monoglycerides |
| 92 | Emulsifiers | Allow smooth mixing of ingredients and prevent separation | Diglycerides |
| 93 | Emulsifiers | Allow smooth mixing of ingredients and prevent separation | Egg yolks |
| 94 | Emulsifiers | Allow smooth mixing of ingredients and prevent separation | Polysorbates |
| 95 | Emulsifiers | Allow smooth mixing of ingredients and prevent separation | Sorbitan monostearate |
| 96 | Stabilizers and Thickeners | Produce uniform texture and improve mouth-feel | Gelatin |
| 97 | Stabilizers and Thickeners | Produce uniform texture and improve mouth-feel | Pectin |
| 98 | Stabilizers and Thickeners | Produce uniform texture and improve mouth-feel | Guar gum |
| 99 | Stabilizers and Thickeners | Produce uniform texture and improve mouth-feel | Carrageenan |
| 100 | Stabilizers and Thickeners | Produce uniform texture and improve mouth-feel | Xanthan gum |
| 101 | Stabilizers and Thickeners | Produce uniform texture and improve mouth-feel | Whey |
| 102 | pH Control Agents and acidulants | Control acidity/alkalinity and prevent spoilage | Lactic acid |
| 103 | pH Control Agents and acidulants | Control acidity/alkalinity and prevent spoilage | Citric acid |
| 104 | pH Control Agents and acidulants | Control acidity/alkalinity and prevent spoilage | Ammonium hydroxide |
| 105 | pH Control Agents and acidulants | Control acidity/alkalinity and prevent spoilage | Sodium carbonate |
| 106 | Leavening Agents | Promote rising of baked goods | Baking soda |
| 107 | Leavening Agents | Promote rising of baked goods | Monocalcium phosphate |
| 108 | Leavening Agents | Promote rising of baked goods | Calcium carbonate |
| 109 | Anti-caking agents | Keep powdered foods free-flowing and prevent moisture absorption | Calcium silicate |
| 110 | Anti-caking agents | Keep powdered foods free-flowing and prevent moisture absorption | Iron ammonium citrate |
| 111 | Anti-caking agents | Keep powdered foods free-flowing and prevent moisture absorption | Silicon dioxide |
| 112 | Humectants | Retain moisture | Glycerin |
| 113 | Humectants | Retain moisture | Sorbitol |
| 114 | Yeast Nutrients | Promote growth of yeast | Calcium sulfate |
| 115 | Yeast Nutrients | Promote growth of yeast | Ammonium phosphate |
| 116 | Dough Strengtheners and Conditioners | Produce more stable dough | Ammonium sulfate |
| 117 | Dough Strengtheners and Conditioners | Produce more stable dough | Azodicarbonamide |
| 118 | Dough Strengtheners and Conditioners | Produce more stable dough | L-cysteine |
| 119 | Firming Agents | Maintain crispness and firmness | Calcium chloride |
| 120 | Firming Agents | Maintain crispness and firmness | Calcium lactate |
| 121 | Enzyme Preparations | Modify proteins polysaccharides and fats | Enzymes |
| 122 | Enzyme Preparations | Modify proteins polysaccharides and fats | Lactase |
| 123 | Enzyme Preparations | Modify proteins polysaccharides and fats | Papain |
| 124 | Enzyme Preparations | Modify proteins polysaccharides and fats | Rennet |
| 125 | Enzyme Preparations | Modify proteins polysaccharides and fats | Chymosin |
| 126 | Gases | Serve as propellant or create carbonation | Carbon dioxide |
| 127 | Gases | Serve as propellant or create carbonation | Nitrous oxide |

### 12.21 Video Link to Application Demo

<https://www.youtube.com/watch?v=oFw1YLyo4R8&feature=youtu.be>

## References

Cowburn, G. and Stockley, L. (2005). Consumer understanding and use of nutrition labelling: a systematic review. Public Health Nutrition, 8(01).

Grunert, K. and Wills, J. (2007). A review of European research on consumer response to nutrition information on food labels. Journal of Public Health, 15(5), pp.385-399.

Fda.gov. (2019). Overview of Food Ingredients, Additives & Colors. [online] Available at: https://www.fda.gov/food/ingredientspackaginglabeling/ucm094211.htm#types [Accessed 14 Jan. 2019].

The Nielsen Company (2015). *Nielsen Global Health and Wellness report*. New York: The Nielsen Company, pp.1-27.

Hollar, A. (2006). *Cowboy: An Agile Programming Methodology for a Solo Programmer*. Graduate. Virginia Commonwealth University.

Dataturks.com. (2018). *Text detection API showdown: Google vision vs Microsoft Vs Amazon*. [online] Available at: https://dataturks.com/blog/compare-image-text-recognition-apis.php [Accessed 7 Oct. 2018].

Google Cloud. (2018). *Optical character recognition (OCR) | Cloud Vision API Documentation | Google Cloud*. [online] Available at: https://cloud.google.com/vision/docs/ocr [Accessed 7 Oct. 2018].

Teplitzki, A. (2013). *ArthurHub/Android-Image-Cropper*. [online] GitHub. Available at: https://github.com/ArthurHub/Android-Image-Cropper [Accessed 9 Feb. 2019].

https://rosettacode.org/wiki/Levenshtein\_distance#Java

1. Cowburn, G. and Stockley, L. (2005). Consumer understanding and use of nutrition labelling: a systematic review. Public Health Nutrition, 8(01). [↑](#footnote-ref-1)
2. Grunert, K. and Wills, J. (2007). A review of European research on consumer response to nutrition information on food labels. Journal of Public Health, 15(5), pp.385-399. [↑](#footnote-ref-2)
3. Grunert, K. and Wills, J. (2007). A review of European research on consumer response to nutrition information on food labels. Journal of Public Health, 15(5), pp.385-399. [↑](#footnote-ref-3)
4. Fda.gov. (2019). Overview of Food Ingredients, Additives & Colors. [online] Available at: https://www.fda.gov/food/ingredientspackaginglabeling/ucm094211.htm#types [Accessed 14 Jan. 2019]. [↑](#footnote-ref-4)
5. Cowburn, G. and Stockley, L. (2005). Consumer understanding and use of nutrition labelling: a systematic review. Public Health Nutrition, 8(01). [↑](#footnote-ref-5)
6. Grunert, K. and Wills, J. (2007). A review of European research on consumer response to nutrition information on food labels. Journal of Public Health, 15(5), pp.385-399. [↑](#footnote-ref-6)
7. The Nielsen Company (2015). *Nielsen Global Health and Wellness report*. New York: The Nielsen Company, pp.1-27. [↑](#footnote-ref-7)
8. Hollar, A. (2006). *Cowboy: An Agile Programming Methodology for a Solo Programmer*. Graduate. Virginia Commonwealth University. [↑](#footnote-ref-8)
9. Grunert, K. and Wills, J. (2007). A review of European research on consumer response to nutrition information on food labels. Journal of Public Health, 15(5), pp.385-399. [↑](#footnote-ref-9)
10. Dataturks.com. (2018). *Text detection API showdown: Google vision vs Microsoft Vs Amazon*. [online] Available at: https://dataturks.com/blog/compare-image-text-recognition-apis.php [Accessed 7 Oct. 2018]. [↑](#footnote-ref-10)
11. Google Cloud. (2018). *Optical character recognition (OCR) | Cloud Vision API Documentation | Google Cloud*. [online] Available at: https://cloud.google.com/vision/docs/ocr [Accessed 7 Oct. 2018]. [↑](#footnote-ref-11)
12. Fda.gov. (2010). *Overview of Food Ingredients, Additives & Colors*. [online] Available at: https://www.fda.gov/food/food-ingredients-packaging/overview-food-ingredients-additives-colors#types [Accessed 14 Jan. 2019]. [↑](#footnote-ref-12)
13. Teplitzki, A. (2013). *ArthurHub/Android-Image-Cropper*. [online] GitHub. Available at: https://github.com/ArthurHub/Android-Image-Cropper [Accessed 9 Feb. 2019]. [↑](#footnote-ref-13)
14. GitHub. (2017). *Text Dectection can return text in arbitrary order · Issue #299 · googlesamples/android-vision*. [online] Available at: https://github.com/googlesamples/android-vision/issues/299 [Accessed 4 Mar. 2019]. [↑](#footnote-ref-14)
15. dzone.com. (n.d.). *The Levenshtein Distance Algorithm - DZone Big Data*. [online] Available at: https://dzone.com/articles/the-levenshtein-algorithm-1 [Accessed 2 Mar. 2019]. [↑](#footnote-ref-15)