

**MOBILE APPLICATION FOR REAL-TIME FOOD IMAGE SEGMENTATION AND
NUTRITIONAL GUIDANCE AT GROCERY STORE**

BY
TAN CHEE LIN

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2024

REPORT STATUS DECLARATION FORM

Title: Mobile Application for Real-Time Food Image Segmentation and
Nutritional Guidance at Grocery Store

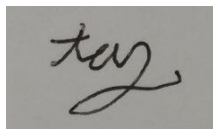
Academic Session: JAN 2024

I TAN CHEE LIN
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

10, Jalan GR 3/5, Jacaranda
Garden Residence, 63100
Cyberjaya, Selangor

Mogana Vadiveloo

Supervisor's name

Date: 26/4/2024

Date: 26/04/2024

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY/INSTITUTE* OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

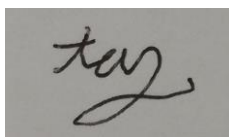
Date: 26/4/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Tan Chee Lin (ID No: 20ACB01595) has completed this final year project/ dissertation/ thesis* entitled "Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store " under the supervision of Dr. Mogana a/p Vadiveloo (Supervisor) from the Department of Computer Science , Faculty/Institute* of Information and Communication Technology .

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

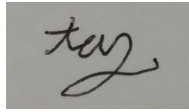


(TAN CHEE LIN)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**MOBILE APPLICATION FOR REAL-TIME FOOD IMAGE SEGMENTATION AND NUTRITIONAL GUIDANCE AT GROCERY STORE**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : TAN CHEE LIN

Date : 26/4/2024

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Mogana who has given me this bright opportunity to partake in the execution of this FYP title and their invaluable guidance throughout the duration of the project. This area of study is new to me, and I had an enjoyable experience learning new deep learning algorithm and techniques for segmenting images in real time. I am looking forward to the further execution of this project and learning new skills to face any new challenges. I will do my best in producing satisfactory and good results.

ABSTRACT

This project is a development-based initiative that aims to help consumers make informed choices about their food purchases by improving existing health and nutrition applications. There are existing similar applications such as MyFitnessPal, Open Food Facts, Food Visor and Food Check that offers some common features that could be found in most health and nutrition applications. However, these applications contain limitations such as inefficiency in searching for food products, insufficient and inaccurate nutritional information, and complexity caused by the application and human errors and other limitations that will soon be discussed. The proposed Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store is to solve their limitations by combining both real-time image segmentation and nutritional information provision, allowing users to capture and analyse food product images through their mobile phone's camera. The application identifies and segments food items within these images, providing users with detailed nutritional information, including macronutrients, micronutrients, and caloric contents. Furthermore, the application will include a personalized system that allows users to login with authentication and update their profile's health status and a robust recommendation system that generates recommendations based on a user's health status by calling the Gemini API. In this work, DeepLabV3+ architecture will be utilized to train the model specifically for real-time image segmentation. Once trained, it will be converted into a TensorFlow Lite model, enabling its integration into a mobile application for performing image segmentation tasks. The images from the Freiburg Dataset are used and manually labelled, to provide 14 different groceries category with a total of 50 images each with their respective ground truth images, totalling only 700 images. The evaluation measurement to determine the performance for the DeeplabV3+ model is the training and validation loss and accuracy. An average accuracy of 87.55% for training set and 66.84% for validation set. In order to evaluate the overall performance of the model, the trained model is applied on the testing set and achieved an accuracy of 82.70%

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	xi
LIST OF TABLES	xiv
LIST OF ABBREVIATIONS	xv
CHAPTER 1 INTRODUCTION	1
1.1 Background Information	1
1.2 Problem Statement and Motivation	2
1.3 Objectives	3
1.4 Project Scope and Direction	3
1.5 Impact and Contributions	4
1.6 Report Organization	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Objective of Literature Review	5
2.2 MyFitnessPal:Calorie Counter	5
2.2.1 MyFitnessPal Features and Functionalities	6
2.2.2 MyFitnessPal Advantages	10
2.2.3 MyFitnessPal Disadvantages	11
2.3 Open Food Facts	12
2.3.1 Open Food Facts Features and Functionalities	12
2.3.2 Open Food Facts Advantages	16
2.3.3 Open Food Facts Disadvantages	18
2.4 Foodvisor	18

2.4.1 Foodvisor Features and Functionalities	19
2.4.2 Foodvisor Advantages	22
2.4.3 Foodvisor Disadvantages	23
2.5 Food Check: Product Scanner	24
2.5.1 Food Check Features and Functionalities	25
2.5.2 Food Check Advantages	29
2.5.3 Food Check Disadvantages	30
2.6 Summary of all Applications	31
2.6.1 Advantages	31
2.6.2 Disadvantages	32
2.7 Understanding Image Segmentation	32
2.8 Traditional Image Segmentation Methods	33
2.8.1 Thresholding Method	33
2.8.2 Edge Based Method	34
2.8.3 Region Based Method	34
2.8.4 Clustering Based Method	35
2.8.5 Summary of Traditional Segmentation Methods	36
2.9 Traditional Segmentation Methods vs Deep Learning Methods	36
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH	38
3.1 Methodology	38
3.2 System Requirement	39
3.2.1 Hardware	39
3.2.2 Software	40
3.3 System Architecture Diagram	41
3.4 User Requirement	42
3.5 Use Case Diagram and Description	43
3.6 Timeline	49
3.6.1 Timeline of FYP1	50
3.6.2 Timeline of FYP2	51

CHAPTER 4 SYSTEM DESIGN	53
4.1 Flowchart of the Overall Flow of the System	53
4.2 Dataset	56
4.3 Model Architecture	57
4.4 System performance Evaluation	58
CHAPTER 5 SYSTEM IMPLEMENTATION	59
5.1 Importing Necessary Libraries	59
5.2 Initialize Parameters and Seedings	60
5.3 Data Augmentation	61
5.3.1 Defining Image Data Generator	61
5.3.2 Applying Data Augmentation	63
5.4 Data Splitting	64
5.5 Data Preprocessing and Loading	65
5.6 Building DeepLabV3+	66
5.7 Hyperparameter Set Up and Model Training	70
5.8 Model Evaluation and Conversion	71
5.9 Prerequisites for Mobile Application Development	73
5.10 Account and Login Feature	76
5.11 Profile Health Status Feature	77
5.12 Loading Model	78
5.13 Segment Frame	79
5.14 Upload Image Segmentation	82
5.15 Real Time Segmentation	83
5.16 Gemini API	84
5.17 Grocery Class Info Page	86
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	87
6.1 Comparison between trained models	87
6.1.1 Backbones	87

6.1.2 System Performance Definition	88
6.1.3 Trained Models Result	89
6.2 Project Challenges and Implementation Issues	90
CHAPTER 7 CONCLUSION AND RECOMMENDATION	91
7.1 Conclusion	91
7.2 Recommendation	91
REFERENCES	93
APPENDIX	98
Upload Image Segmentation Results	98
Real Time Segmentation Results	105
WEEKLY LOG	112
POSTER	118
PLAGARISM CHECK RESULT	119
FYP2 CHECKLIST	121

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	MyFitnessPal Search Engine and Nutritional Profile of Apple	6
Figure 2.2	MyFitnessPal Meal Scan	7
Figure 2.3	MyFitnessPal Recipe Discover and Recipe Details	8
Figure 2.4	MyFitnessPal Recipe Details	8
Figure 2.5	Open Food Facts Search Engine and Food Product Profile	12
Figure 2.6	Open Food Facts Food Product Profile	13
Figure 2.7	Open Food Facts Food Preferences Customization	15
Figure 2.8	Open Food Facts Barcode Scanner	16
Figure 2.9	Food Visor Search Engine and Food Profile	19
Figure 2.10	Food Visor Food Profile	20
Figure 2.11	Food Visor Food Recognition Camera	21
Figure 2.12	Food Check Search Engine and Food Product Profile	25
Figure 2.13	Food Check Food Product Profile	25
Figure 2.14	Food Check Recipe Search Engine	27
Figure 2.15	Food Check Recipe Profile	27
Figure 2.16	Food Check Articles and Barcode Scanner	28
Figure 2.17	Growing of Seed Pixel	35
Figure 2.18	Division and Merging of an Image	35
Figure 3.1	Agile Development Approach	38
Figure 3.2	System Architecture Diagram	41
Figure 3.3	Use Case Diagram of the Application	43
Figure 3.4	Activity done during FYP 1	50
Figure 3.5	Gantt Chart of FYP1 Project Timeline	50
Figure 3.6	Activity done during FYP 2	51
Figure 3.7	Gantt Chart of FYP2 Project Timeline	51
Figure 4.1	Flowchart of Training DeepLabV3+ model	53
Figure 4.2	Flowchart of the Proposed Mobile Application	54

Figure 4.3	Frieburg Groceries Dataset	56
Figure 4.4	DeepLabV3+ Architecture	57
Figure 5.1	Importing the Necessary Libraries	59
Figure 5.2	Parameters and Seedings	60
Figure 5.3	Defining Image Data Generator	61
Figure 5.4	Applying Data Augmentation	63
Figure 5.5	Data Splitting	64
Figure 5.6	Total Images for Training, Validation and Testing Set	65
Figure 5.7	Data Preprocessing and Loading	65
Figure 5.8	ASPP	66
Figure 5.9	DeepLabV3+ with VGG16	67
Figure 5.10	DeepLabV3+ with VGG19	67
Figure 5.11	DeepLabV3+ with ResNet50	68
Figure 5.12	DeepLabV3+ with ResNet101	69
Figure 5.13	Hyperparameter Set Up and Model Training	70
Figure 5.14	Model Evaluation 1	71
Figure 5.15	Model Evaluation 1	72
Figure 5.16	Model Conversion	72
Figure 5.17	Firestore Authentication	73
Figure 5.18	Connection between Android Studio and Firestore Authentication	73
Figure 5.19	Gemini API key	74
Figure 5.20	Android Studio Dependencies	75
Figure 5.21	Android Studio Permissions	76
Figure 5.22	Register and Login Page	76
Figure 5.23	Profile Page	77
Figure 5.24	Loading Model	78
Figure 5.25	Segment Frame 1	79
Figure 5.26	Segment Frame 2	80
Figure 5.27	Upload Image Segmentation	82
Figure 5.28	Real Time Segmentation	83
Figure 5.29	getResponse	84

Figure 5.30	getModel	85
Figure 5.31	Grocery Class Info Page	86

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Advantages of applications review	31
Table 2.2	Disadvantages of applications review	32
Table 2.3	Advantages and Disadvantages of Traditional Methods	36
Table 3.1	Specifications of Laptop	39
Table 3.2	Specifications of Mobile Device	39
Table 3.3	Use Case Description for “Register Account” Use Case	44
Table 3.4	Use Case Description for “Login Account” Use Case	45
Table 3.5	Use Case Description for “Logout Account” Use Case	46
Table 3.6	Use Case Description for “Upload Image Segmentation” Use Case	47
Table 3.7	Use Case Description for “Real Time Segmentation” Use Case	48
Table 3.8	Use Case Description for “Update Health Status” Use Case	49
Table 6.1	Trained Models Result	89

LIST OF ABBREVIATIONS

<i>DALYs</i>	Disability-Adjusted Life Years
<i>SDK</i>	Software Development Kit
<i>AI</i>	Artificial Intelligence
<i>CO</i>	Carbon Dioxide
<i>ANNs</i>	Artificial Neural Networks
<i>CNN</i>	Convolutional Neural Networks
<i>IDE</i>	Integrated Development Environment
<i>UI</i>	User Interface
<i>ASPP</i>	Atrous Spatial Pyramid Pooling

Chapter 1

Introduction

1.1 Background Information

The global increase in diet-related health issues, such as obesity, diabetes, and heart diseases, has underscored the importance of making informed and healthier food choices. According to [1], poor diets were responsible for 10.9 million deaths and 22% of all deaths among the adults are with cardiovascular disease as the leading cause, followed by cancers and diabetes. Additionally, it has also led to 255 million disability-adjusted life years (DALYs), which represent the total of years lost due to premature death and years spent with impaired health or disability. Research also indicates that over half of diet-related deaths and 66% of DALYs can be attributed to three dietary factors: insufficient consumption of whole grains and fruits, along with excessive sodium intake. Conversely, the remaining 50% of deaths and 34% of DALYs are linked to the overconsumption of red meat, processed meats, sugar-sweetened beverages, trans fatty acids, and other food items.

With the increased awareness of the important of balanced diet, more and more health-conscious consumers are now inclined to seek nutritional information and make conscious decisions about the foods and products they purchase. However, this process can be overwhelming, time-consuming, and prone to inaccuracies, especially for consumers that are new to this topic. The existing methods of manually reading food labels or searching through databases require a significant amount of effort and nutritional knowledge [2]. This often leads to frustration and discourages individuals from making healthier choices. Furthermore, human error in estimating portion sizes and nutritional content can result in misleading information, undermining the very purpose of dietary awareness [3]. In addition to searching inefficiency and human error, these search results can sometimes be inaccurate, insufficient, and outdated. The dynamic nature of the food industry with constant updates of food products and changes in their ingredients also further complicating the process of making informed dietary choices. In response to these challenges, there is a growing demand for innovative solutions that simplify the task of accessing accurate nutritional data while shopping for foods and products.

Due to the recent advancements in computer vision and deep learning, there are now more opportunities and possibilities for revolutionizing different industries like healthcare, automotive, manufacturing, and various other industries. In reference to [4], with the assistance of deep learning models, computer vision applications have now achieved a notable improvement in their ability to rapidly identify objects in images or videos with a higher degree of accuracy and speed. One of its most captivating advancements in the field of computer vision is the emergence of real-time image segmentation system, making them ideal for applications such as autonomous vehicles, video surveillance, medical imaging, and plenty more [5]. Most importantly, it is also perfect for developing applications that can analyze images of food products in real time and provide users with instant information about their nutritional content, helping them make more informed choices at the point of purchase.

1.2 Problem Statement and Motivation

After reviewing existing applications such as MyFitnessPal [6], Open Food Facts [7], Food Visor [8] and Food Check [14], these applications are able to provide sufficient nutritional information for foods and products, information about the ingredients used in the making of the products and even food logging, progress tracker and various other tools to assist their users to reaching their goal. Surprisingly, some of the applications which are MyFitnessPal and FoodVisor have also implemented computer vision to automatically detects a food in an image taken by the user's mobile device's camera and tries to identify it for the user [6][8]. However, based on the reviewed applications, it is clear that these applications have food image recognition but do not provide sufficient and accurate nutritional information while applications like Open Food Facts and Food Check contains more detailed nutritional information that helps user understand more about the food products with ingredient analysis and nutrition scores but did not implement any computer vision to help users search for their food products [7][14]. Furthermore, all these applications except Open Food Facts do not even provide tips that are able to educate their users with health and nutrition, as well as providing recommendations based on the user's preferences to their user experience. On top of these limitations, most applications except Open Food Facts even require their users to pay for some of their advanced features causing some users to be reluctant to pay, in order to use their features [7].

Therefore, the motivation of this project being proposed is to reduce diet related health issues which has been responsible for a significant number of death cases. This is achieved by developing a health and nutrition mobile application that uses real-time image segmentation that allows its users access accurate nutritional data from any food found at the grocery store. By offering such a user-friendly and accessible tool, this project strives to promote healthier eating habits, reduce the risk of diet-related diseases and ultimately improving and saving lives. Beyond its health benefits, this project may also contribute to increased transparency in the food industry, encouraging manufacturers to provide accurate nutritional information of their foods and products and the recognizing the potential of computer vision and using it for quality-of-life improvements.

1.3 Objectives

The main objective of this project is to develop a health and nutrition mobile application that segments visual objects of food products that can be found in a grocery store in real time. The following are the sub-objectives to achieve this:

- To segment food products from an image or in real time using deep learning models
- To provide nutritional information about the food products segmented in the camera.
- To provide recommendations of the food products to the user.

1.4 Project Scope and Direction

For this project, a health and nutrition mobile application will be developed for Android platform. This application will employ computer vision technology to perform real-time food image segmentation and classification, enabling users to capture and identify various food items while shopping in a grocery store. Additionally, the application will offer users with accurate and up-to-date nutritional information for the identified food products obtained from external databases and authoritative nutritional sources. Beyond this, the project aims to provide recommendations tailored to individual user's unique preferences and specific health conditions by having them answer questions like allergies, medical conditions, and many more before using the application. The application will also be designed with user-friendly interface and to provide real-time performance and accuracy during food image recognition, allowing users to effortlessly make informed decisions about identified foods as they shop in the grocery store.

1.5 Impact and Contribution

Currently there are many health and nutrition mobile applications in the market that are free to download but some still require their users to pay for some of their features. As previously mentioned, these applications share common limitations such as insufficient and inaccurate nutrition data, as well as inefficiency in searching foods and products. While some applications offer advanced features and excel in certain areas, they are lacking features that others provide like providing more detailed and additional nutrition information and food image recognition. Hence, the proposed mobile application in this project will solve the above-mentioned issue where the application will be focused on providing sufficient and accurate information of the food products that could be found in the grocery store like nutritional information, ingredients information and even recommendations to the users while also implemented a food image recognition feature that works in real time. The innovation of this mobile application will allow users to access nutritional information for food products from the grocery store for free and improve the efficiency of searching the correct items using food image recognition.

1.6 Report Organization

This report is organized into 7 chapters such as Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology/Approach, Chapter 4 System Design, Chapter 5 System Implementation, Chapter 6 System Evaluation and Discussion and Chapter 7 Conclusion and Recommendation. The introduction of this project including background information, problem statement and motivation, project scope, project objectives, impact and contribution will be included in Chapter 1. For Chapter 2, it is the literature review of different mobile applications that provide nutritional information or guidance and the effectiveness of existing deep learning models. Chapter 3 will be discussing the methodology, system, and user requirements, use case diagram and the timeline of the overall project. Chapter 4 will be discussing the flow chart of the system, dataset, and the model architecture. Then Chapter 5 is discussing the details on how to implement the system. Following with Chapter 6, it will discuss the evaluation of the system and model with the project challenges and implementation issues. Finally, the report will be concluded with the conclusion and recommendations in Chapter 7.

Chapter 2

Literature Review

2.1 Objective of Literature Review

The main objective of this literature review is to understand the functionalities of existing applications that provide nutritional information or guidance and the effectiveness of existing deep learning models compared to traditional segmentation methods used for real-time image segmentation to be implemented in an application. By evaluating the features, advantages, and disadvantages of different existing applications, then can understand what the core functions of the applications are and how effective are in assisting the consumer in accomplishing their goals. The literature review begins with reviewing and comparing existing applications followed by comparing traditional segmentation methods and deep learning methods to give us a better insight into why deep learning models offer a significant advantage over traditional segmentation methods.

2.2 MyFitnessPal: Calorie Counter

MyFitnessPal is a health and nutrition application, designed to help its users learn about their habits, observe how they eat, make smarter food choices, find motivation, or support, and conquer their health goals [6]. The platform implemented a variety of tools like a food tracker, calorie counter, macro tracker, food diary, and other handy tools to assist their users in completing their objectives. One of its unique characteristics is that the platform has access to over 14 million food databases which enables the application to create more features to assist users like breaking down calories and nutrients more accurately and efficiently [9]. The application supports both Android and IOS mobile devices and can also be accessed through its website on any computer. However, while anyone is free to register, free users are still limited with some features not accessible and will need to subscribe for premium to experience its full capability.

2.2.1 MyFitnessPal Features and Functionalities

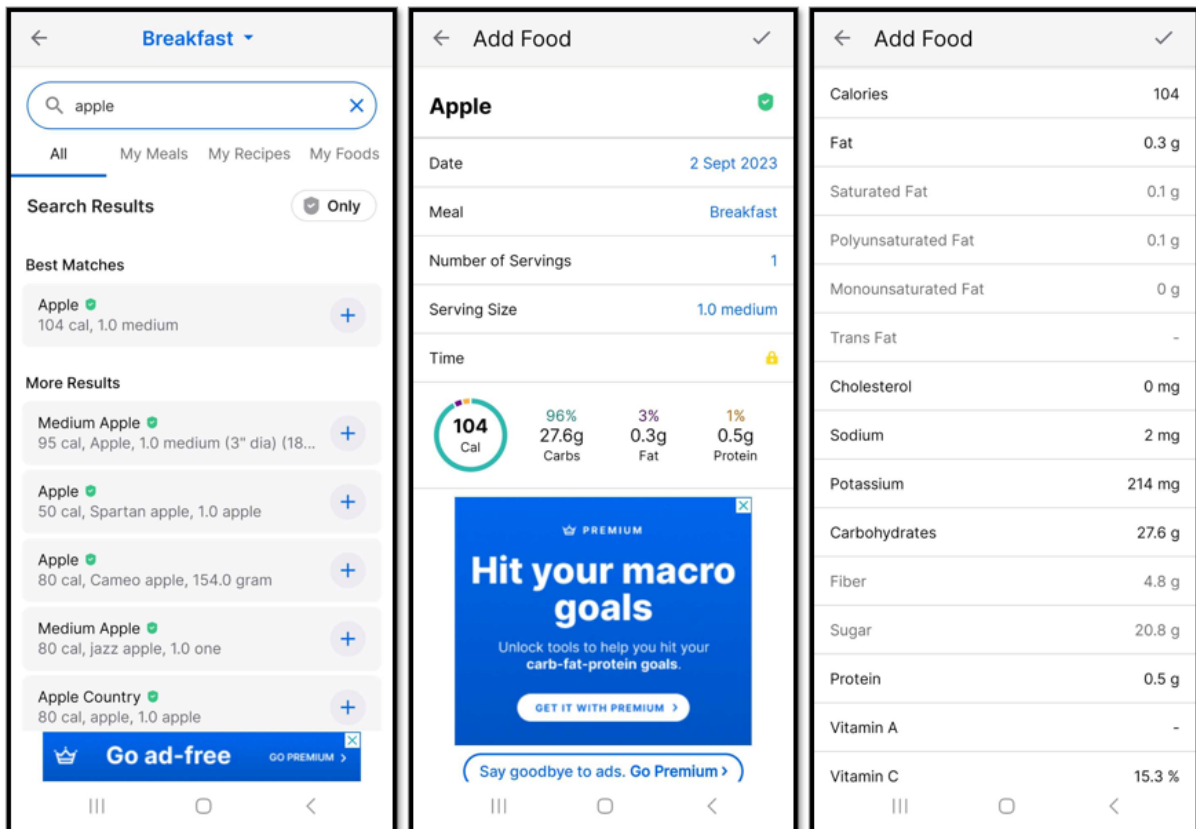


Figure 2.1 MyFitnessPal Search Engine and Nutritional Profile of Apple from [6]

There are several main features for nutritional information and guidance application in this application one of which is providing a nutritional profile on any selected food that the user is interested in. Selecting any food from the search engine provided by the application will display the total calories of the food according to its portion size followed by 3 primary macronutrients including Carbohydrates, Fat, and Protein. For overall nutritional information, users can swipe up to view a full list of nutrient content containing calories, macronutrients, and even micronutrients like Fiber, Sugar, Vitamin A, and more.

Based on the example above in Figure 2.1, by typing “apple” into its search engine it shows its best matches consecutively showing other results that are like the search input. Each result will have a summary of its total calories, portion size, and a green check to indicate that the item is verified by the application itself. By selecting the first item which is Apple, it shows the selected portion size and its total calories presented with a donut chart with its 3 macronutrients. The portion size of the selected item is changeable, and its nutritional information will change according to it. Lastly as stated earlier, by swiping up, a list of nutrient information with its amount represented using mg, g, or percentage can be seen.

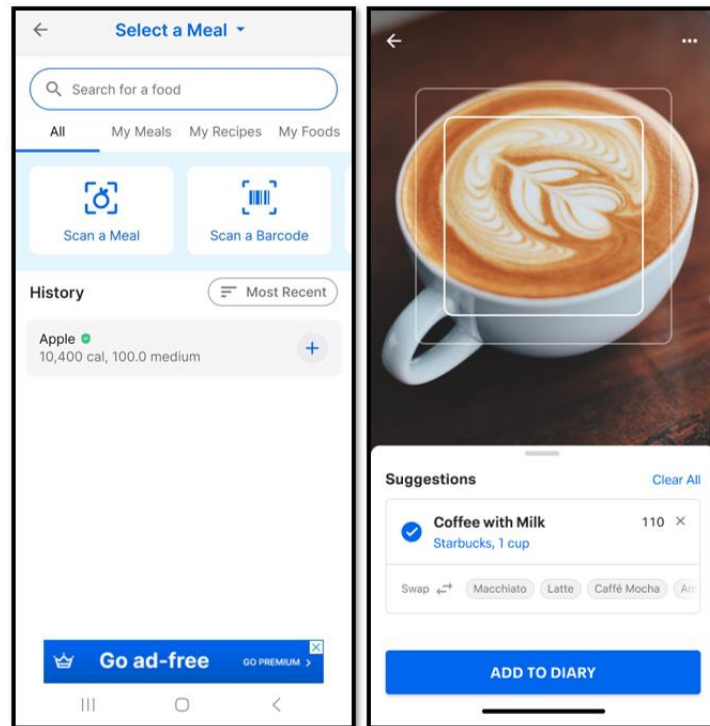


Figure 2.2 MyFitnessPal Meal Scan from [6] and [10]

On top of its search engine, the application has also implemented Meal Scan a main feature that identifies every food by the user's camera and suggests verified foods from its food database. According to [10] and [11], it uses Passio Software Development Kit (SDK) which enables on-device food recognition using Machine Learning and Computer Vision powered by the world's leading nutrition AI technology. Unfortunately, it is considered a premium feature in the application which requires its users to pay for the premium subscription, for them to access this feature.

By referring to the example above in Figure 2.2, in the searching interface, users can access the feature by clicking on the scan a meal with an icon that resembles a camera scanning an item in the middle. After the user scans an image like in the second picture above in Figure 2.2, the application will suggest food that is identified and can either be selected or swapped to other suggestions. By tapping on the selected item, users will be redirected to its profile that displays its portion size and all its nutritional information as previously mentioned.

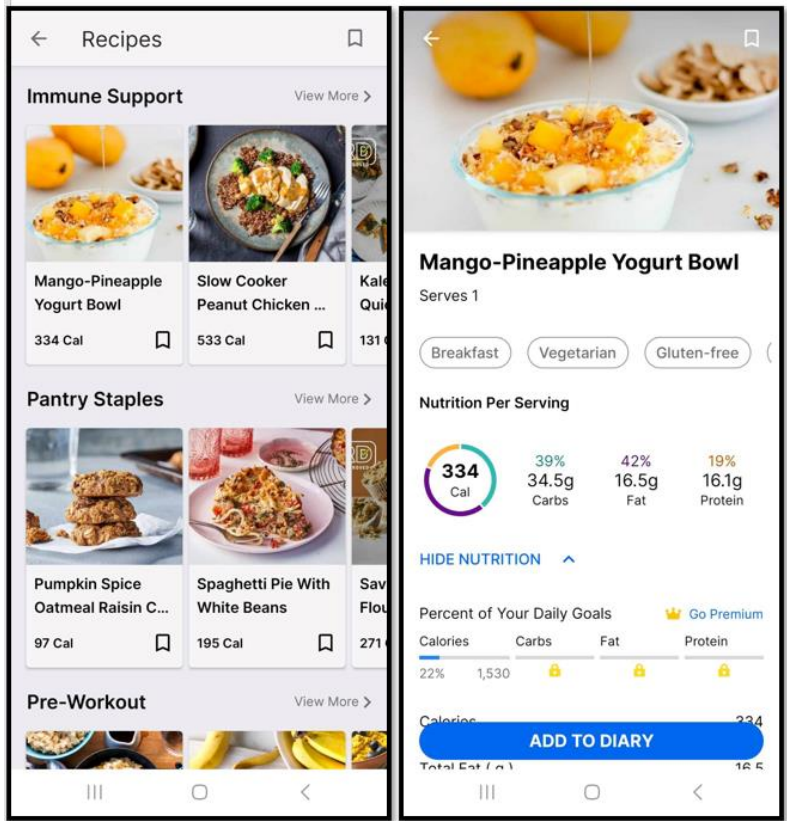


Figure 2.3 MyFitnessPal Recipe Discover and Recipe Details from [6]

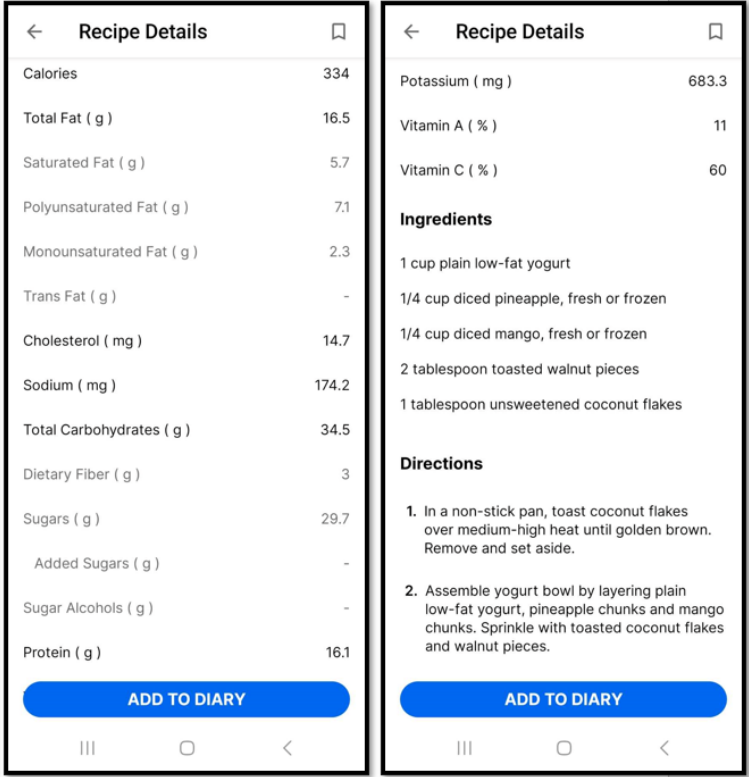


Figure 2.4 MyFitnessPal Recipe Details from [6]

Furthermore, the application has another main feature that allows its users to discover recipes that fit a user's specific dietary needs whether gluten-free, vegan, low-carb, or high-protein, helping them make healthier food choices. Like selecting any food in the application, users can select any recipe that they are interested in and view its nutritional information, ingredients, and even directions on how to make them.

For example, users are presented with a variety of different recipes under different themes like Immune Support, Pantry Staples, Pre-Workout, and many more as the user starts going down the list. These recipes can be bookmarked or selected in this case as "Mango-Pineapple Yogurt Bowl". It will then show other themes or labels that the recipe is also known for like Breakfast, Vegetarian, Gluten-free, and more. Then, similar to the content displayed above in Figure 2.1 it shows the total calorie with its 3 macronutrients and a full list of nutrition information containing calories, macronutrients, and micronutrients. Swiping up more will show the recipe's ingredients and directions for users to understand what to use and how to make the recipe for themselves.

Additionally, there are various types of supporting features implemented in the application like Food and Activity Logging, Barcode Scanner, Set Goals, Progress Tracker, Plans Discovery, Newsfeed, and Community. First, MyFitnessPal provides a diary that records the calories gained from any meal during any mealtime and calories burned from any physical activities to calculate the total calories per day. Other than searching or scanning meals to show item details or record any meal eaten for the day, the application also implemented a barcode scanner to scan packaged food that will automatically show information about the scanned item and easily log if the user wants to. Besides that, users are also able to set personalized goals like losing weight and the application will automatically calculate the limit of calorie intake per day, to help users control their diet and reach their goal. The application also utilizes graphs and charts to present information on the user's overall nutrition intake and progress toward their goal. For inexperienced users, the application also prepares plans discovery for them to discover any diet or workout plans for them to follow or gain ideas from. Lastly, the application contains a newsfeed for health and diet-related news and an active and supportive community that the users can access to gain more knowledge and motivation during their fitness journey.

2.2.2 MyFitnessPal Advantages

- Extensive Food Database

Have access over 14 million foods database that offers valuable information about foods like portion sizes and nutritional information which will allow users to find the information they need.

- Detailed Nutritional Information

For every food profile displayed in the application, it contains detailed nutritional information like the name and quantity of calories, macronutrients and micronutrients using the correct percentages and units like *g* and *mg*, therefore allowing users to understand the quantities easier and make better food choices with the provided information and. It even varies with different portion size of the same item to match the suppose quantity of a larger or smaller version of the item. This would allow users to match their food as close as possible, leading to greater accuracy in its nutritional information.

- Implementation of Various Tools

Besides providing nutritional information for its users, it implemented various other tools using nutritional information that may assist them in achieving their goals like losing weight or living a healthier lifestyle. For example, users may track their progress on their weight, discover more healthier options and educate themselves more on health and nutrients using tools like Recipe Discovery, Barcode Scanner, Food Dairy and various other tools.

- Use of Image Recognition

It utilizes the image recognition in identifying any food that could be found in the image captured using the user's mobile device's camera and suggest the verified food that fits the description of it from the database to the user. It may improve the efficiency of searching the food or help users that do not know the name of the food to identify its name and learn about its nutritional information.

- Food Verification

It made sure show which food that its information has been officially verified to the users. By doing so, users may know which items will be able to trust and contain more accurate information.

2.2.3 MyFitnessPal Disadvantages

- Data Inaccuracy

While the database is extensive, not all data are verified and accurate which could display incorrect nutritional information, leading to users making food choices that do not align with their goals or food preferences. It would also increase the frustration and loss of trust in the application from application.

- Paid Premium Features

Some of its advanced features like Food Image Recognition, Barcode Scanner and other functions are locked behind a premium subscription which limits some access to certain functionalities for users that does not plan to subscribe.

- Privacy Concerns

The application starts by asking some questions related about some of their personal information that are related to health, to enhance the user experience and customization. This however will lead to some users being reluctant to share such data, especially if they are unsure about the application's security measures.

- Lack of Ingredients Information or Analysis

While it provides nutritional information, it does not provide the information of ingredients used to make the food if there is any. This makes users that have strict diet restrictions or allergies to certain ingredients to be unsure whether the food is suitable for them to consume or have any hidden additives and unhealthy components.

- Complexity for Beginners

For new users, most of them may be first time downloading a nutritional application and trying to utilize it to learn about nutrients and make healthier choices of consuming the right foods. Therefore, there are many unfamiliar icons like the blue tick and terms like the name of nutrients that they do not understand which cause misunderstandings and frustration.

2.3 Open Food Facts

Open Food Facts is a collaborative application with an open-source database of food product information from all around the world [7]. It aims to collect and share data on food products including their ingredients, nutritional information, labels, and various other details. The goals of Open Food Facts are also to promote transparency and provide users with access to accurate and even allow users to add pictures or data to get the Nutri-Score and NOVA score on food processing. According to [7], Open Food Facts is also known as “the Wikipedia of food” that enabled the creation of over 100 applications. It supports both IOS and Android mobile devices and can be accessed through its website with similar functionalities. It is also a free application that is available for download for everyone and does not include any premium subscription to access any features.

2.3.1 Open Food Facts Features and Functionalities

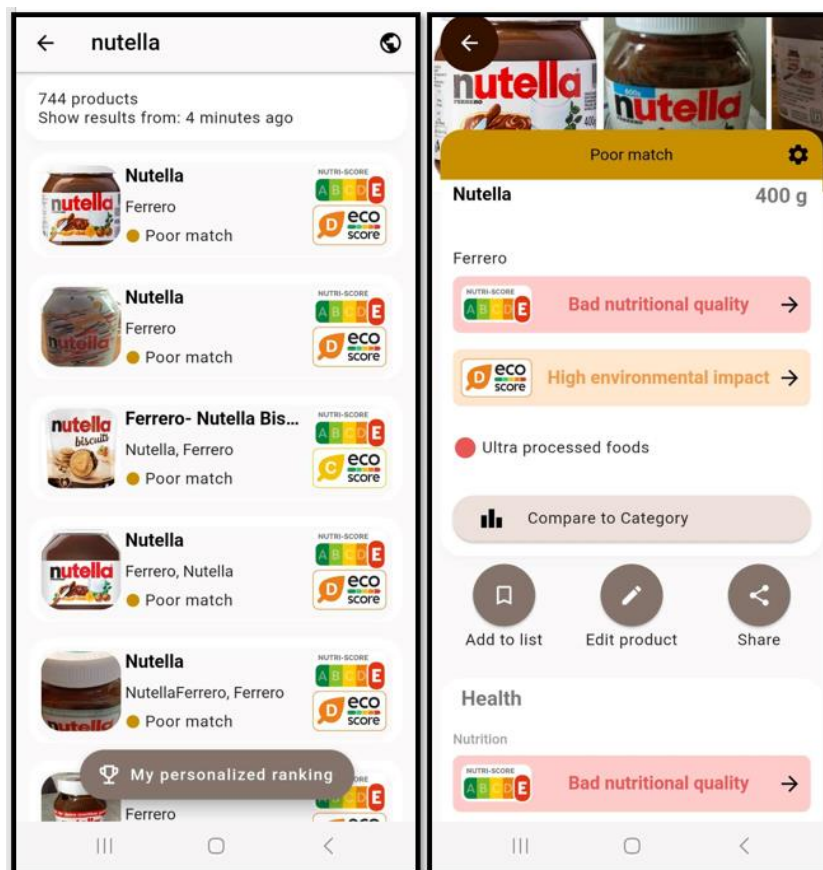


Figure 2.5 Open Food Facts Search Engine and Food Product Profile from [7]

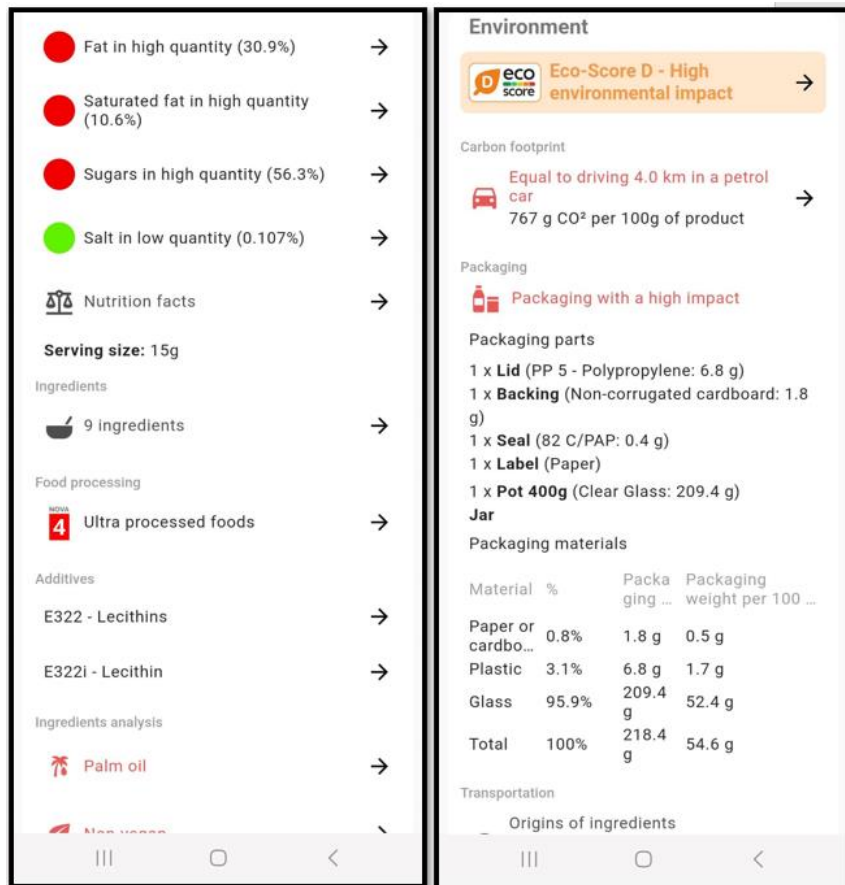


Figure 2.6 Open Food Facts Food Product Profile from [7]

This application offers a range of main features related to nutritional information and guidance. One of its main features is providing nutritional information for a variety of food products followed by additional information like company and brand name, Nutri-score, Eco-score NOVA, and Ingredients analysis. It is divided into 3 different sections, the first section is about its Nutri-score, Eco-score, and NOVA, the second section is about nutrition facts and ingredients analysis, and the third section is about environmental impact and packaging information. Besides the scoring of the food product, it even has a button for users to compare similar products in the same category to find maybe healthier or more sustainable alternatives.

For a demonstration of how it works, when the user searches for a food product like “Nutella”, it will display the total and a list of products found based on the user input as shown above in Figure 2.5. Each product will be labeled with its name followed by its company or brand name, its grade in Nutri-score, Eco score, and whether it’s a good match based on the user’s food preferences that are configurable. For more user accessibility, the application implements a filter for filtering the search result based on its best match by clicking on “My Personalized Ranking”.

By selecting the first item in the list, it will display a profile of the product with 3 different sections, overall score, health, and environment. The first section, tells the user that Nutella from a company named Ferrero has bad nutritional quality, has high environmental impact, and is categorized as an ultra-processed food which shows that Nutella is not only bad for an individual's health but has a high impact on the environment to create or transport the product. For additional information, users can even tap on them to learn more about the reason behind their bad scores and even classify them as ultra-processed food.

Besides that, as shown above in Figure 2.6, it first shows some nutrition advice that talks about the quantity of Fat, Saturated fat, Sugars, and Salt contained in the product with red symbolizing it's too high in quantity while green symbolizes its perfectly low in quantity. Tapping either one of them, will educate users about how high consumption of each of these nutrients will increase the risk of users getting harmful diseases and recommendations on how to limit themselves from consuming too much. For overall nutrition information, users can tap on Nutrition Facts which will display a list of nutrition from the product that is calculated according to the portion size of the product. Besides, there's also ingredient analysis on food processing status and what ingredients and additives are used in the product. It may help users find products according to their nutrition preference and avoid foods that could trigger allergies or violate dietary restrictions and learn about how additives are made and whether the effects of the additives are harmful to their health.

Finally, the third section shows the total carbon footprint generated equal to driving 4.0 km in a patrol car just to make the product and in this case, 767g of CO² is generated per 100 g of the product. By tapping on it, it shows the percentage of impact covering different stages of making the product including Agriculture, Processing, Packaging, Transportation, Distribution, and Consumption. For the rest of the section, it shows packaging parts and materials, origins of ingredients, and what species will be threatened from the making of this product.

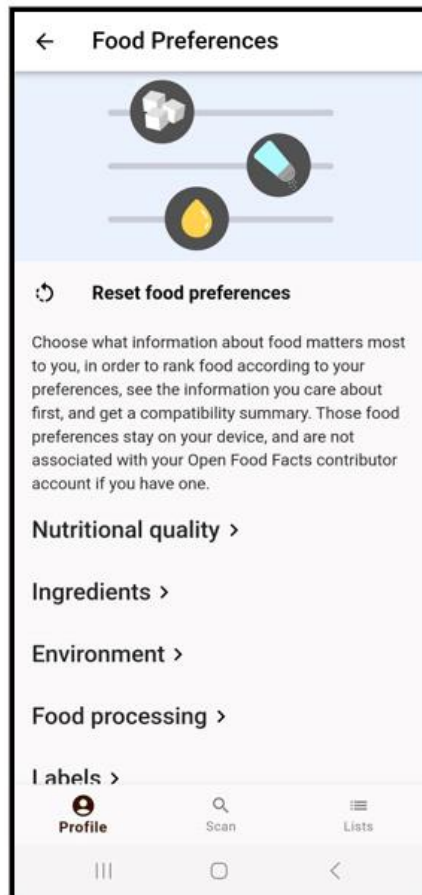


Figure 2.7 Open Food Facts Food Preferences Customization from [7]

Moreover, the application includes food preference customization as another main feature associated with nutritional information and guidance. Whenever a new user opens the application, they will be asked to select a few food preferences which will later be used to provide personalized recommendations to the user and rank food according to the user's preference. Users are still allowed to customize their food preferences as shown above in Figure 2.6, to accommodate their diverse dietary needs or restrictions, and food allergies and cater to their individual preferences, which can evolve over time.

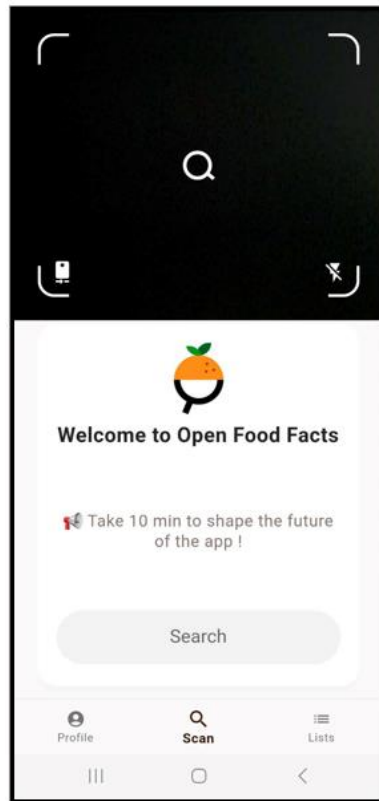


Figure 2.8 Open Food Facts Barcode Scanner from [7]

Lastly, the supporting feature that could be found in this application is a barcode scanner which acts as another way of searching for food products as above in Figure 2.7. It will try to identify the product by scanning the barcode and displaying the product details like above in Figure 2.5 and Figure 2.6. If the application is unable to identify the product, it will allow the user to add it as a new product and add data like brand name, nutrition information, and various other product details.

2.3.2 Open Food Facts Advantages

- Extensive Food Database

The database of Open Food Facts is a result of contributions from a community of volunteers, users and organizations who continually add food products and update existing entries. This collaborative approach allows the database to cover a wide range of food products that will provide the nutritional information, most users wanted to learn about and continues to grow over time.

- Detailed Nutritional Information

For every food product profile in the application, it contains nutrition facts like Fat, Saturated fat, Carbohydrates and many more for users to make more informed food choices with the provided information. The quantity of the food product does not have a fixed portion size and can be changed using the unit *g* as a measure of quantity. By changing its quantity, the quantity of its nutrients will automatically be calculated according to it and will be displayed using the correct percentages and units like *kJ*, *g* and *vol* that can be understood by most users. With this dynamic approach, it gives a ton of flexibility to the users by allowing them to calculate the quantity of the nutrition facts however they like and

- **Ingredient Information and Analysis**

Like nutritional information, every food product profile in the application makes sure to display the information of its ingredients used to make the food product including allergens for users that have allergies to avoid them and even the hidden additives which also shows transparency to its users as the additives may be unhealthy for them.

- **Nutrition Scores**

The application calculates each food products' Nutri-score and NOVA and display them on each of them to indicate their nutritional quality and the level of processing in the food. This will greatly help users that are trying to avoid unhealthy food products with bad nutritional quality or high levels of processing.

- **Recommendations**

When searching for food products, the application will recommend users which food product is a good match according to their food preferences saved in the application which will improve user satisfactions. Furthermore, the application also gives recommendations on how to limit the consumption of fat, saturated fat, sugar, and salt which will increase the risk of diseases and other unhealthy consequences to increase the user's awareness.

- **Food Comparison**

The application allows users to compare food products in the same category would allow finding alternatives, especially for people that have food allergies or strict diet restrictions since they will be able to find some other similar food products that could align with their food preferences.

2.3.3 Open Food Facts Disadvantages

- Data Inaccuracy

Since most of the data in the database of the application is from volunteers and users, it may cause duplicate entries leading to redundancy and different information for the same food product. There may even be outdated information and malicious data from irresponsible users that forgets about their existing entries or attempting to troll or manipulate the database to personal gain. In the end, it may cause loss of trust in the application from the users.

- Overemphasis on Scoring

With each food products displaying their Nutri-score, Eco-score and NOVA, users may be encouraged to focus too much on their scoring and neglect other context like individual's overall diet and specific nutrients that are essential to them. Users will be obsessed with only food products with high scoring, which usually sometimes is not a good choice for them.

- Complexity for Beginners

With the use of unfamiliar terms like different nutrients, Nutri-score, and many more, new users that are trying to learn do not understand them and will have a difficult time using the application which could leads to various misunderstandings on what they represent.

- Scope Limited to only Food Products

While it provides tons of information about a food product, it does not contain information for organic foods also known as pure foods like vegetables or fruits. The application is limited to only having information for food products and information about other foods will not be accessible to the users.

2.4 Foodvisor

Food Visor is also a health and nutrition application that acts as a personal nutritionist that helps users make healthier food choices, monitor their daily intake, and reach weight goals sustainably [8]. It is known to have Instant Food Recognition Camera that uses image recognition technology to analyze photos of food and provide users with detailed information about the nutrients and calories in their dishes. Like most applications, it supports IOS or Android mobile devices and contains its own website mainly used for account settings or acts

as a blog page that shows the latest articles related to nutrition or diet and even some guides for visitors to learn about. Even so, some of its premium features are also locked behind a subscription that requires users to pay, in order to gain access to them.

2.4.1 Foodvisor Features and Functionalities

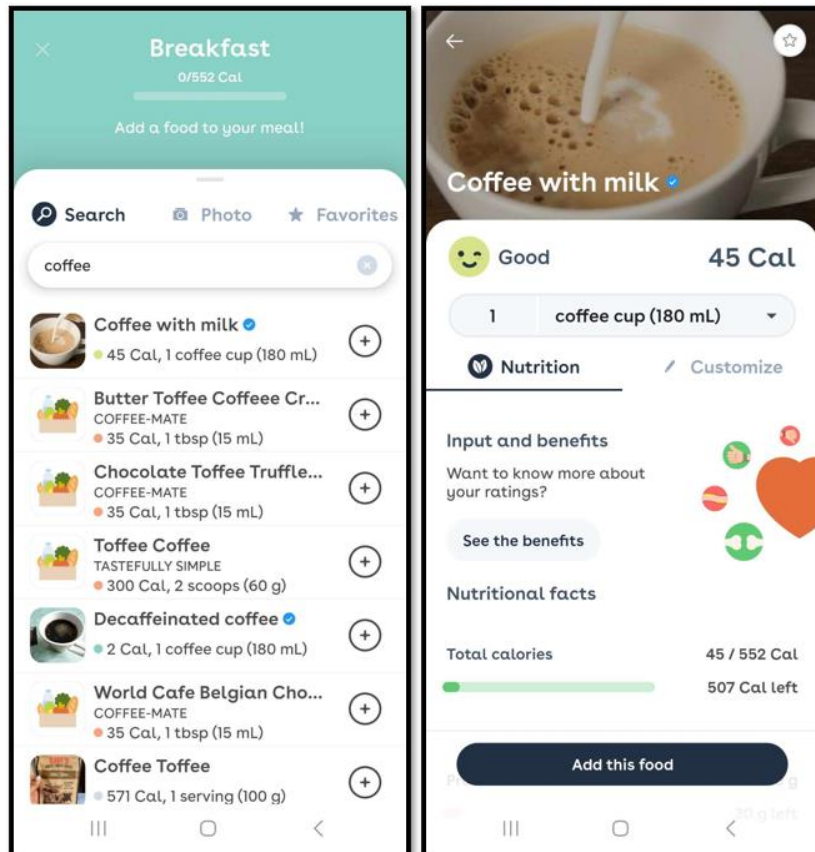


Figure 2.9 Food Visor Search Engine and Food Profile from [8]

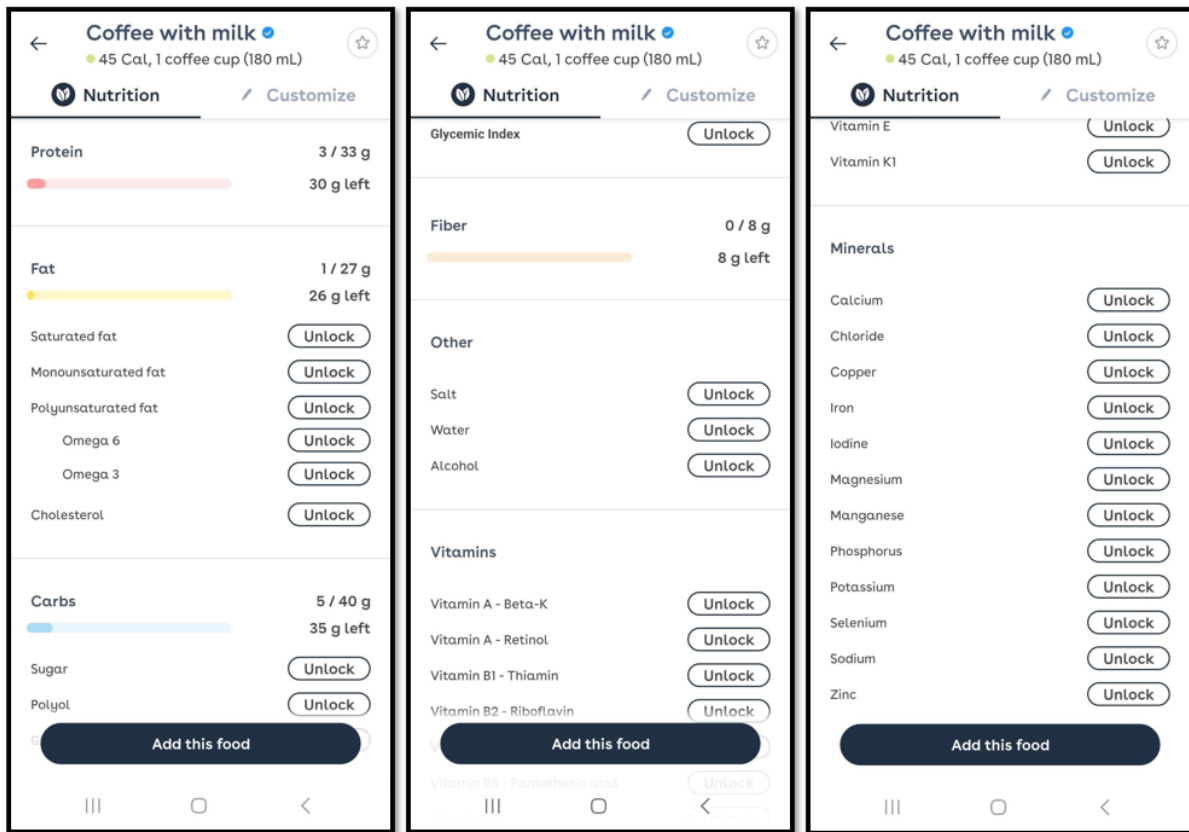


Figure 2.10 Food Visor Food Profile from [8]

Among the main features related to nutritional information and guidance that could be found in this application is the provision of nutrition details in each of the food’s profiles.

For instance, users will input the food name “Coffee” into the search engine implemented by the application to search for any item in the database that is similar to it as shown above in Figure 2.9. Then, each item from the search result will be displayed row by row, each one labeled with its name, total calorie, and portion size. According to [12], the food color that can be seen beside the item indicates different meanings where blue means very good, green means good, yellow means correct and orange means to be limited. If the item is also verified in the database of Foodvisor, a blue tick beside the item’s name can also be seen in the list of searched items.

By selecting either one of those items, it will then navigate the user to the item’s profile which in this scenario is the first item on the list, “Coffee with milk”. The food’s profile will first show its benefits and risks based on the user’s health and weight from answering questions when first launching the application or customized in the profile section. Then, it shows the nutritional facts including total calories, protein, fat, carbs, fiber, vitamins, minerals, various micronutrients, and other nutrients as shown above in Figure 2.9 and Figure 2.10. Nevertheless,

it is unfortunate that users must make payments to access information on the benefits and risks of the food as well as the information on micronutrients, and some other nutrient information.

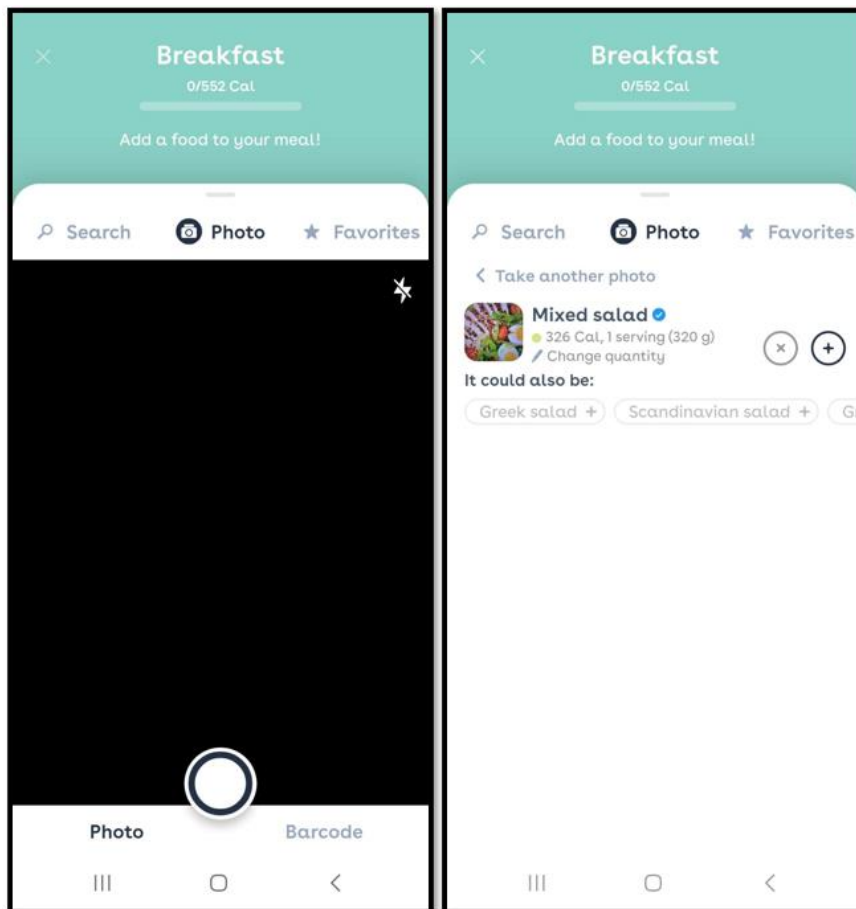


Figure 2.11 Food Visor Food Recognition Camera from [8]

Other than its simple searching using the application's search engine, Food Visor has also implemented a food recognition feature. It is a main feature that uses deep learning and image recognition to detect what the user's food is and even try to estimate its portion size. According to [13], the algorithm used by Foodvisor will try to evaluate the distance between the food and the user's phone using camera autofocus data and then calculate the area of each of the food items in the photo.

As demonstrated above in Figure 2.11, the food recognition can be accessed just beside the search tab, and the user can just take a photo of their food using their mobile device's camera. After capturing the image, it will show the items that the algorithm thinks resemble the food in the input image. It will also suggest other food that is also similar to the suggested item. For instance, above in Figure 2.11, the Mixed Salad is suggested as the item detected in the input image and stated that it could also be Greek salad, Scandinavian salad, and other options that the user can choose to change. Even so, users are still given the option to take

another photo if the results are not accurate or manually correct some of the details of the suggested item like quantity or nutrition information if only the details of the result are incorrect.

In addition, the application also includes various supporting features such as a Calorie Tracker, Personalized Nutrition Plan, 1:1 Coaching with Experts, Progress Tracker, and In-depth analysis. First and foremost, Foodvisor has prepared a food journal to record any food taken during any mealtime, water, and activities that involve exercise. By logging them, users will be able to track their total calories or macronutrient intake from all the calories gained from food or water and calories burned from exercises. Besides that, users are able to set personalized goals and limit daily calorie and macronutrient intake to reach their goals. The user's progress will also be tracked by adding weight entry and the application will display statistics on the user's nutrient consumption and goal progress using graphs and charts which will help users gain some better insight into their eating habits and make informed decisions to adjust their diet. Lastly, there's also 1:1 coaching with experts for guiding newer or inexperience by educating them with advice, and knowledge related to diet and health, as well as providing motivation and confidence to help them stay committed to their health and nutrition journey.

2.4.2 Foodvisor Advantages

- Detailed Nutritional Information

For every food profile in the application, each of them will display nutritional facts like what is the total calories, macronutrients and micronutrients that can be found within the food, allowing users to inform themselves of the nutrition contained in the food and make better food choices. The quantity of every nutritional fact will be measured using correct units like *Cal*, *g* and *mL* that can be understood by most users and calculated according to the portion size of the food to try providing more accurate nutritional fact that the users can work with.

- Benefits and Risk Analysis

Foodvisor allows users to learn more about the benefits and risks of each food according to their health and weight inputted from questions when users first open the application or manually configured in their profile settings. With this analysis, users will be able to

understand the benefits and risks from consuming the food and make better informed decisions.

- Use of Image Recognition

The application utilizes image recognition in one of its features, Food Image Recognition that uses the user's mobile device's camera to capture any food and the algorithm will try to identify it and suggest food that resembles the food detected in the captured image. It will help users search for their food easier and help identify the food's name and learn about its nutritional information.

- Food Verification

When searching for food, user will be able to see foods that have been verified with a blue tick located beside their name in the search result or even in their profile. With food verification, users will be able to know which food's information is more trustable and reliable.

2.4.3 Foodvisor Disadvantages

- Data Inaccuracy

Even with food verification, there's a ton of items that are yet to be verified in the application. There are also food profiles that are outdated, incomplete and totally incorrect with having 0 total calories with the wrong portion sizes. This may lead to users using the wrong nutritional information provided in the application and loss of trust in the application due to their frustration.

- Paid Premium Features

For free users that do not plan to pay for its premium subscription, they will not be able to access some of its advanced features. For example, the information of some nutrients and macronutrients like total calories, protein, fat, carbs, and fiber is accessible to all users, however as for the information of other macronutrients and micronutrients, they are all locked behind a paywall that only allows premium users to access.

- Privacy Concerns

For every first-time user in this application, they will need to answer questions related to their personal information and health information to provide customization and a better

user experience. Nevertheless, due to various amount of data breaches and data misuse, some users may hesitate to share such data with the application, fearing for their own safety from being targeted for cybercriminal activities.

- Lack of Ingredients Information or Analysis

Although it offers nutritional information, it does not display the information of ingredients used in the food. This may leave users with specific dietary limitations or allergies uncertain about the food compatibility with them and whether it contains unhealthy components and hidden additives.

- Complexity for Beginners

For new users, most of them may be unfamiliar with the symbols like blue tick and terms like the name of nutrients and do not understand what they are, and purpose do the nutrients have when they are consumed. This may cause them to struggle to navigate the application and undermining the application's usability and effectiveness in providing nutritional information.

2.5 Food Check: Product Scanner

Food Check is an application related to food scanning and nutritional information. It aims to help users avoid food products that do not fit their food standards by identifying how the products were made and indicating warnings for dangerous additives or unbalanced nutrients contained in the products [14]. The application is available on both IOS and Android devices with slightly different names. For IOS, it is called Food Scanner: Grocery Coach, while for Android is called Food Scanner: Product Scanner. However, to access certain features, users are required to acquire a premium subscription to the application, as these functionalities are restricted unless a purchase is made.

2.5.1 Food Check Features and Functionalities

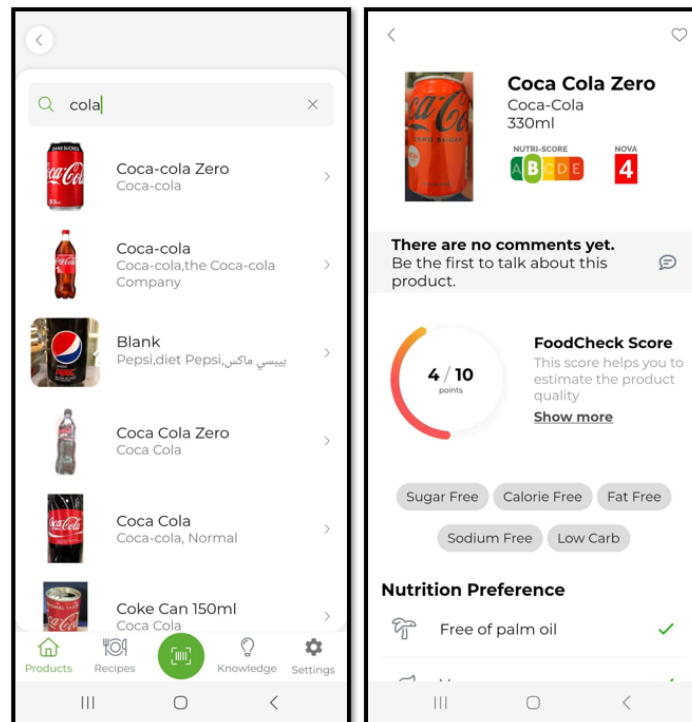


Figure 2.12 Food Check Search Engine and Food Product Profile from [14]

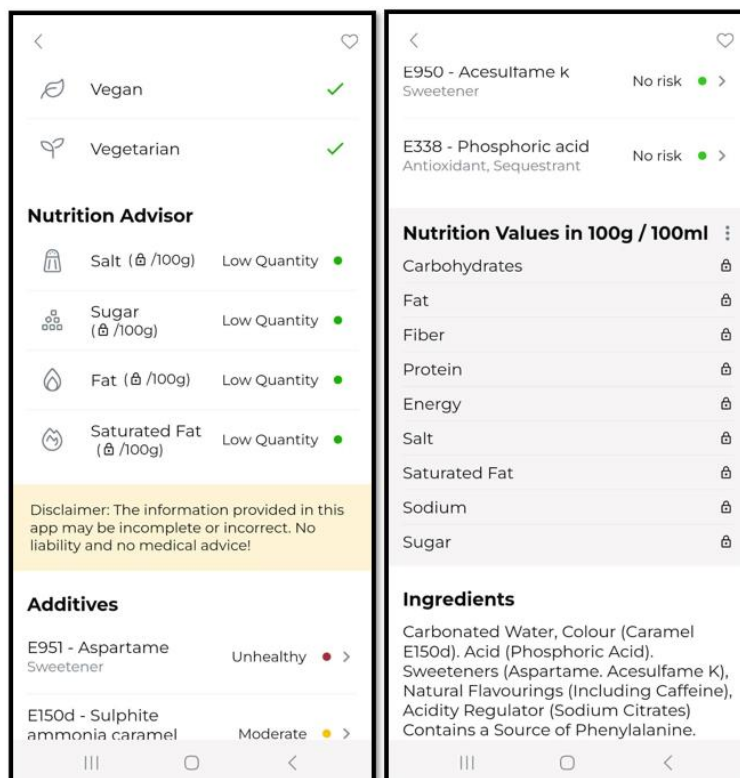


Figure 2.13 Food Check Food Product Profile from [14]

One of the main features that is associated with nutritional information and guidance found in this application is the ability to provide users with accurate nutritional information of

any food product that they are interested in. Besides nutritional information, the application also provides additional information like Nutri-score, NOVA, FoodCheck score, and ingredients used to make the product.

To give an example, the image above in Figure 2.12 shows the user can do simple search for any product like “Cola”, which will then show different search results that may have cola in their name or are similar to it. After selecting the first product from the results which is “Coca Cola Zero”, it will show the profile of the product with its name, portion size, and various information. It first shows its Nutri-score, NOVA, and FoodCheck Score where the Nutri-score is grade B and NOVA indicates that it is an ultra-processed drink product with a low FoodCheck Score. Tapping on them will also show more information on how they determine their grade and scores, and what the grade and scores mean.

Swiping down the user interface shows the nutrition and ingredients information of the product. Nutrition preference can be seen first with green ticks on each row indicating it is true as shown in Figure 2.13, where the product is free of palm oil and is suitable for Vegan or Vegetarian. Afterward, the nutrition advisor will show the quantity of Salt, Sugar, Fat, and Saturated Fat contained in the product and advise the user on the risk of consuming too much of these nutrients. Then, a list of additives contained in the product, each labeled with their ID, name, the purpose of this additive, and risk status is displayed and can be tapped to learn more about them. While above in Figure 2.13, the product contains 4 different additives with 2 of them having no risk and the other 2 having moderate risk and unhealthy. As the user goes to the bottom of the user interface, a list of nutrition values like Carbohydrates, Fat, Fiber, Protein and many other nutritional values and ingredients used in the making of this product can be seen. But then again, the list of nutrition values is locked to free users and only users with premium subscriptions may access the information.

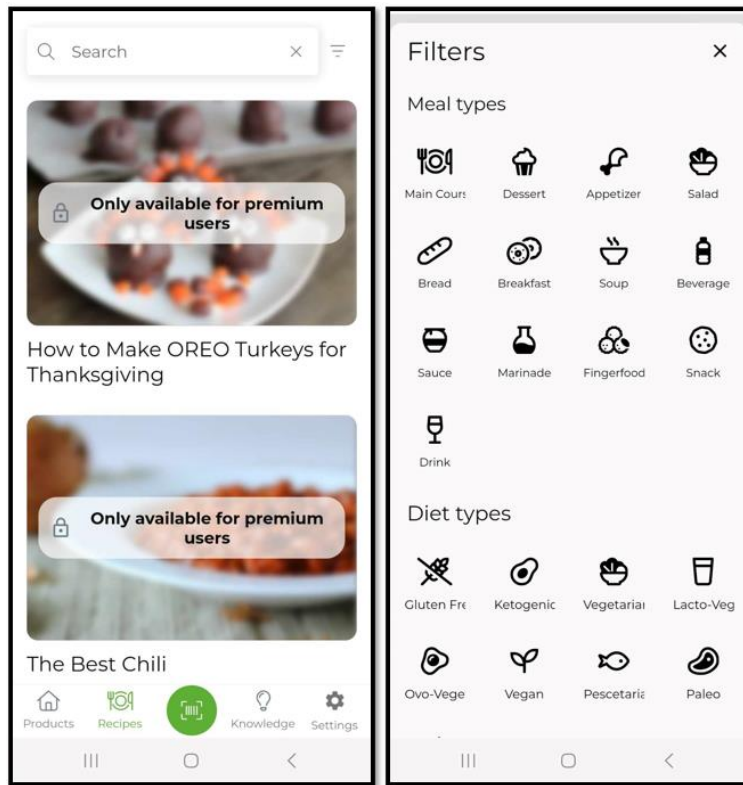


Figure 2.14 Food Check Recipe Search Engine from [14]

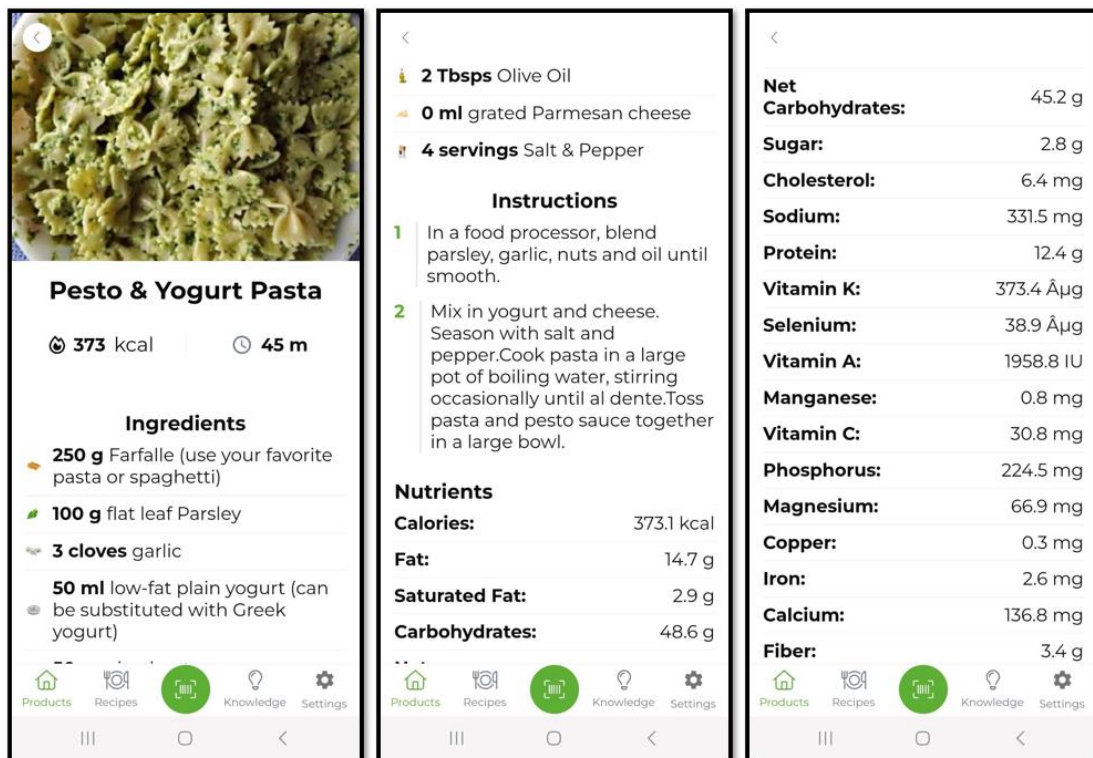


Figure 2.15 Food Check Recipe Profile from [14]

Other than that, the application also implemented Recipe Discovery a main feature that allows users to discover recipes with different meal types or diet types. Similar to the profile

of a food product, it also displays nutritional information and various other information to educate and assist on how to make them. However, while this feature is advantageous to any users, some recipes are considered premium recipes that still only be accessible to premium users as shown above in Figure 2.14.

As to ease the process of searching for suitable recipes for all users, the application added a filter function to the search engine to filter the recipe's meal type like Salad, Breakfast, Snack, and many more, as well as filter diet types like Vegetarian, Gluten Free, Ketogenic, and various types. After selecting a recipe and navigating to its profile interface, it will first show its total calories and estimated time to make the recipe. Then like any other recipe feature, it shows the ingredients required to make the recipe followed by detailed instructions for users to follow step by step. Lastly, users will be able to see the recipe's nutritional information like the quantity of Fat, Saturated Fat, Carbohydrates, and several other nutrients as shown above in Figure 2.15.

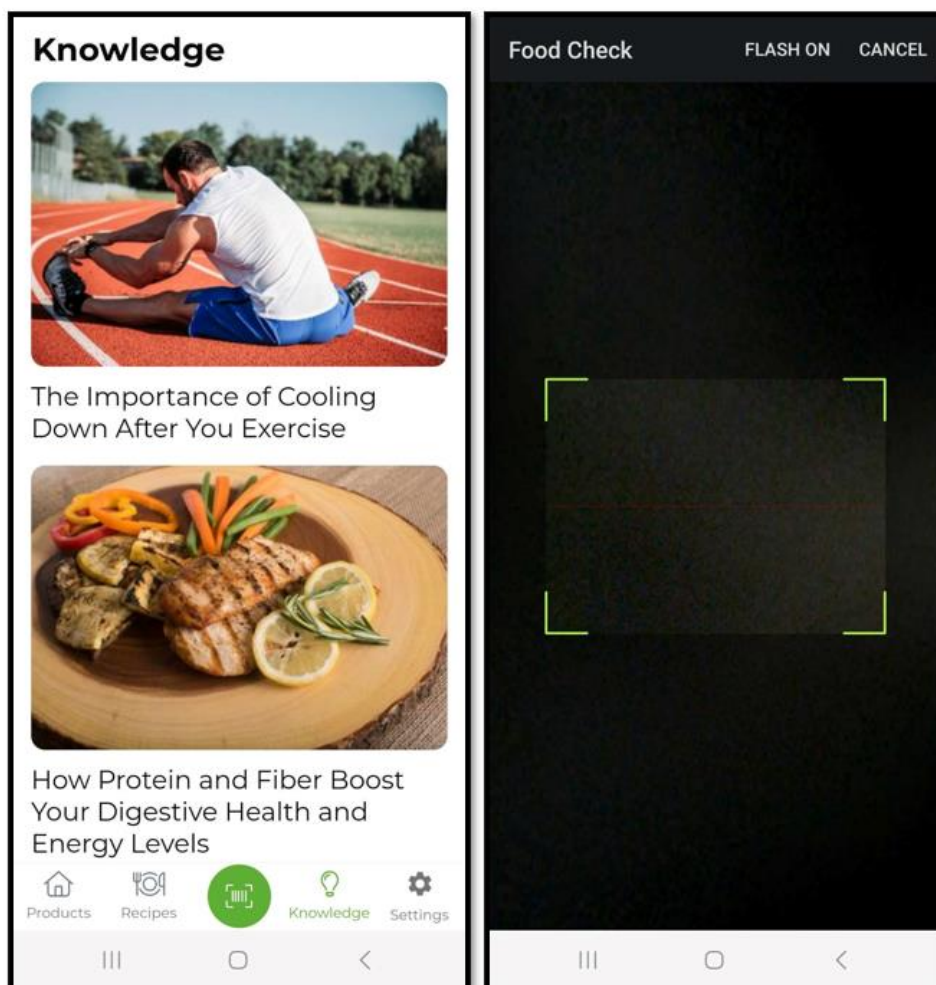


Figure 2.16 Food Check Articles and Barcode Scanner from [14]

On top of its main features, Food Check also contains other two support features, that is a barcode scanner and articles associated with health and nutrition knowledge as shown above in Figure 2.16. The barcode scanner acts as another search method by scanning the barcode on any product which will then automatically identify the product and display its information including name, portion size, nutritional information, and various information just like above in Figure 2.12 and Figure 2.13. As for the articles provided by Food Check, they enable users to educate themselves about nutrition and health and promote healthy lifestyles by having users make better health choices or maintaining a healthy lifestyle.

2.5.2 Food Check Advantages

- Detailed Nutritional Information

Each of the food product profile in the application, contains nutrition facts like Fat, Fiber, Carbohydrates and many more for users to make more informed food choices with the provided information. The quantity of the nutrition values is also displayed according to per 100g or 100ml of the food product using the correct units like *g* and *ml* that can be understood by most users.

- Ingredient Information and Analysis

Similar to nutritional information, every food product profile within the application makes sure to provide a list of its ingredients used to make the food product including information about allergens to help users that have allergies avoid them and even emphasizing transparency with it users by disclosing hidden additives which may be unhealthy for them.

- Nutrition Scores

The application calculates each food products' Nutri-score and NOVA, as well as generating their own unique FoodCheck score. These scores are then presented for each of them, serving as indicators of their nutritional quality and the extent of food processing. This will give users a heads-up on food products that have bad nutritional quality or high levels of processing, in order to avoid them.

- Implementation of Various Tools

Other than providing nutritional information for its users, it implemented various other tools related to nutritional information that may assist them like recipe discovery, articles and barcode scanner. For instance, users will be able to educate themselves more about

health and nutrition by going through articles provided in the application and discover new recipes that might align with their food preferences. The barcode scanner acts as another search method for users to search for their products just by scanning the barcode located on them.

2.5.3 Food Check Disadvantages

- **Data Inaccuracy**

When searching for a specific food product, duplicated items with same names but different information can be seen in the search result. There are also food profiles that are outdated and have empty information in their profile despite still able to search for it. This may cause users more inefficiency as users may not know which items represent their food product and have doubts about the accuracy of the provided food product information.

- **Paid Premium Features**

Unfortunately, free users are unable to access the quantity of each nutrition values contained in a food product, information on additives, information on nutrition advisor and how they calculate their own unique FoodCheck Score, since they are all locked and only available for premium users that pay for the subscription.

- **Overemphasis on Scoring**

With each food products displaying their Nutri-score, NOVA and FoodCheck score, users may be encouraged to only fixate on these scores, potentially disregarding other essential factors like their overall dietary needs and specific vital nutrients. Users could become preoccupied with exclusively selecting items with high scores, even though such choices might not always align with their individual nutritional requirements.

- **Complexity for Beginners**

Many newcomers to the application are likely to encounter unfamiliar terms like the Nutri-score, NOVA and nutrient names, possibly leading them to a confusion and frustration due to their lack of understanding.

- **Scope Limited to only Food Products**

Although it provides tons of information about a food product, it lacks information for organic foods often referred as pure foods like different vegetables or fruits. In the end,

users will not have access to information about these other types of foods as the application is restricted to providing details solely for processed food products.

2.6 Summary of All Applications

2.6.1 Advantages

Table 2.1 Advantages of applications review

Advantages	MyFitnessPal	Open Food Facts	Foodvisor	Food Check
Extensive Food Database	✓	✓	✗	✗
Detailed Nutritional Information	✓	✓	✓	✓
Ingredient Information Analysis	✗	✓	✗	✓
Benefits and Risk Analysis	✗	✗	✓	✗
Nutrition Scores	✗	✓	✗	✓
Recommendations	✗	✓	✗	✗
Food Comparison	✗	✓	✗	✗
Implementation of Various tools	✓	✗	✓	✓
Use of Image Recognition	✓	✗	✓	✗
Food Verification	✓	✗	✓	✗

2.6.2 Disadvantages

Table 2.2 Disadvantages of applications review

Disadvantages	MyFitnessPal	Open Food Facts	Foodvisor	Food Check
Data Inaccuracy	✓	✓	✓	✓
Paid Premium Features	✓	✗	✓	✓
Privacy Concerns	✓	✗	✓	✗
Lack of Ingredients Information or Analysis	✓	✗	✓	✗
Overemphasis on Scoring	✗	✓	✗	✓
Complexity for Beginners	✓	✓	✓	✓
Scope Limited to only Food Products	✗	✓	✗	✓

2.7 Understanding Image Segmentation

Image Segmentation is a Computer Vision technique that involves dividing an image into multiple meaningful and homogeneous regions or objects based on certain characteristics like color, texture, shape, or brightness. It aims to simplify and refine an image into something more meaningful and accessible to analyze [15]. It also plays a crucial role in enabling computers to understand and interpret visual information in a manner similar to human perception which is important in applications in various fields, including medical imaging, autonomous vehicles, and satellite imagery.

According to [15], there are 3 types of segmentation. First, Instance Segmentation focuses on identifying each unique object instance in the image. This proves valuable in scenarios where precise identification and tracking of individual objects are required. Following that, Semantic Segmentation assigns a class label to every pixel in an image, proving beneficial in

situations where identifying different classes of objects is important. Lastly, Panoptic Segmentation is a hybrid approach of combining both semantic and instance segmentation. This becomes advantageous in applications where the computer vision model is tasked with detecting and engaging with different objects in its environment.

2.8 Traditional Image Segmentation Methods

Traditional Image Segmentation methods have been a staple in computer vision for many years, aimed at extracting valuable insights from images. These methods rely on mathematical models and algorithms that identify regions of an image with common characteristics like colour, texture, or brightness. Traditional methods aim to separate objects or regions of interest from the background or any other objects in an image based on defined rules and principles, making them usually computationally efficient and relatively straightforward to implement [15].

However, traditional methods might struggle in complex scenarios where objects in various images have varying shapes, sizes, appearances, lighting conditions, and other backgrounds that are not well-defined. In such cases, some traditional methods might perform better segmentations, while others will struggle, making their effectiveness to be highly scenario-dependent and inconsistent.

Nevertheless, several popular traditional image segmentation techniques will be briefly discussed including Threshold Method, Edge Based method, Region Based Method, and Clustering Based Method.

2.8.1 Thresholding Method

Thresholding Method is a simple segmentation method and easy to implement with its straightforward concept. It produces a binary image, where each pixel is represented using only 0 and 1. This process involves taking a grayscale image and dividing it into 2 regions using a specified threshold value. In the output image, which will be a binary image, pixels with intensities higher than the threshold will become white and 1 in value while the rest will become black and 0 in value [16]. According to [17], this technique can also be expressed as:

$$T=T[x, y, p(x, y), f(x, y)] \quad (2.1)$$

where T is the threshold value, x and y are the coordinates of the threshold value point, $p(x, y)$ and $f(x, y)$ are points in the grey level image. The threshold image $g(x, y)$ can also defined as:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > 1 \\ 0 & \text{if } f(x,y) \leq 0 \end{cases} \quad (2.2)$$

Moreover, thresholding method is also classified into Global Thresholding and Local Thresholding. Global Thresholding is a technique that focuses on separating pixels of an entire image into foreground and background regions based on pixel intensity values. As for Local Thresholding, also known as Adaptive Thresholding, is a technique that focuses on separating pixels of an image into foreground and background regions by adjusting the threshold value locally based on the image features. Still, both techniques are still sensitive to variations in lightning conditions and noise levels which can affect the accuracy and reliability of the segmentation results.

2.8.2 Edge Based Method

Edge Based Method is a segmentation method that is effective for images with better contrast between objects as it identifies and delineate the object boundaries within an image based on the rapid change of intensity value in an image, since a single intensity value does not provide sufficient information about the edges [18]. The process encompasses two primary tasks which are Edge Detection and Edge Linking. Firstly, Edge Detection usually uses Canny Edge Detection or Sobel Edge Detection to detect pixels that are considered edges which essentially outlines the objects. Subsequently, Edge Linking refines the initially detected edges by linking the adjacent edge points together and form complete object boundaries to segment the required regions [19]. Nevertheless, it does not work well dealing with images with smooth transitions and low contrast and sensitive to noises like images with too many edges which could lead to over-segmentation.

2.8.3 Region Based Method

Region Based Method is a segmentation method that divides an image into several regions based on similar pixel characteristics like colour, texture, or intensity. It is also considered more reliable and effective compared to other methods since it is more robust towards noise levels and effective dealing with similarity criteria that are simple and straightforward. However, it uses a lot of its resources and not very efficient with its time and memory usage. Following [18], there are 2 basic methods which are Region Growing Method and, Region Splitting and Merging Method. Region growing method involves selecting initial pixels, also known as seeds, either manually or automatically. Then, it will now segment the image into

various regions by including neighbouring pixels with similar characteristics based on the growing of seeds controlled by its connectivity between pixels and similar characteristics as demonstrated below in Figure 2.17.

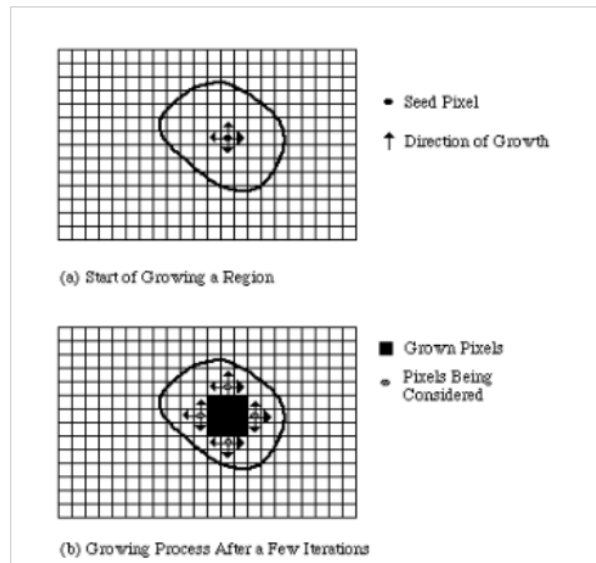


Figure 2.17 Growing of Seed Pixel from [19]

Region splitting and merging method involves splitting and merging an image into various regions. The process involves dividing an image into regions that have the same characteristics and then merging adjacent similar regions iteratively until there are no further splitting or merging is possible as shown an example below in Figure 2.18.

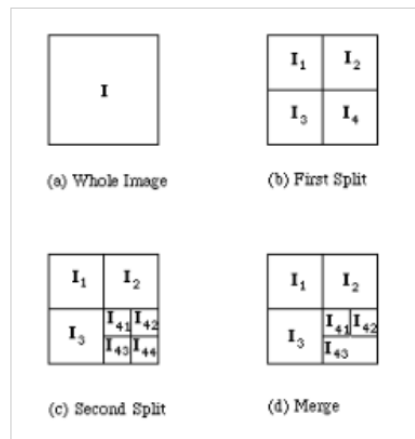


Figure 2.18 Division and Merging of an Image from [19]

2.8.4 Clustering Based Method

Clustering Based Method is a segmentation method that groups pixels that shares common traits into clusters. The process involves grouping pixels into clusters base on their similarities, where each cluster represents a segment which could help discover obscure patterns and

relationships [15]. Still due to it relying on clusters, the effectiveness of this method can be influenced by the initial choice of clustering algorithm. According to [18], there are 2 basic categories of clustering methods which are Hierarchical method and Partition based method. Hierarchical methods are based on the concept of trees where the root of the tree symbolizes the entire database, and the internal nodes represent the clusters. On the other hand, Partition based methods rely on iterative optimization methods to minimize an objective function. These methods employ various clustering algorithms such as K-means clustering, mean shift clustering and fuzzy clustering.

2.8.5 Summary of Traditional Segmentation Methods

Table 2.3 Advantages and Disadvantages of Traditional Methods

Segmentation techniques	Advantages	Disadvantages
Thresholding Method	Simple to understand and implement.	Sensitive to variations in lighting conditions and noise levels
Edge Based Method	Effective for images with better contrast between objects	Over-segmentation when there's too many edges
Region Based Method	More robust to noise levels Effective when similarity criteria are simple and straightforward	Not very efficient with its time and memory usage
Clustering Based Method	Able to discover obscure patterns and relationships	Sensitive to the initial choice of clustering algorithm

2.9 Traditional Segmentation Methods vs Deep Learning Methods

Since a brief understanding on traditional segmentation methods has been established, it is known that all traditional methods rely on features extracted from the image such as intensity, colour, shapes, spatial relationships and many more. These features are characteristics or properties of the image that provide information about the content and structure of the image. The challenges that come with using traditional methods is that it is necessary to select which

crucial features within each specific images. As the number of classes to classify increases, the process of extracting features becomes more and more complicated. The process also involves iterative processes, where segmentation is refined over multiple iterations with some manual tuning of the parameters until a satisfactory result is achieved [20]. Despite all these computer vision problems, deep learning has emerged as a transformative approach in recent years.

According to [20], Deep learning is a subset of machine learning that uses Artificial Neural Networks (ANNs), a computing paradigm that functions like the human brain. The neural network is composed of layers of artificial neurons that can process complex patterns and features. Compared to traditional segmentation methods, deep learning has the ability to learn from raw data with minimal preprocessing, capture complex and nonlinear relationships, and handle larger and unstructured data and achieve higher accuracy in most tasks including image classification, semantic segmentation, object detection and many more. Given that neural networks utilized in deep learning are mostly trained rather than programmed applications, this approach often requires less analysis and fine tuning. There are also various Convolutional Neural Networks (CNN) models like U-Net, SegNet and DeepLab. According to [15], U-Net uses encoder-decoder structure, where its encoder is used to extract features using a shortcut connection and allows it to capture more information by concatenating high-level features with low-level ones. As for SegNet [15], it also consists of encoder and utilizes max-pooling indices from the encoder's max-pooling layer for non-linear up sampling, eliminating the need to learn the up-sampling process. Lastly for DeepLab [15], it utilizes features from all convolutional blocks and concatenating them to their deconvolutional block. It also uses dilated convolution for up sampling, reducing computational cost while capturing extensive information.

In conclusion, deep learning methods is better for larger and complex datasets that achieves high accuracy and performance in difficult tasks while traditional segmentation methods are better for smaller or medium datasets that require simpler or linear models, as well as interpretability and exploitability in decision making [21].

Chapter 3

System Design

3.1 Methodology

After much deliberation, Agile Development Approach was chosen to develop the application [22]. The reason being that the time to develop the application is limited, making this approach much more suitable to complete the project in time as it works well with shorter turnaround times and tight deadlines. Besides, due to the fact there will be consumer involvement giving continuous feedback and validation, there will also be frequent changing requests usually updating new features to be add into the application or existing features to either improve their functionalities or fix any bug [22]. By doing so, there will also be multiple versions and incremental deliverables that will be advantageous to the project because it allows stakeholders or consumers to start getting values sooner compared to other approaches.

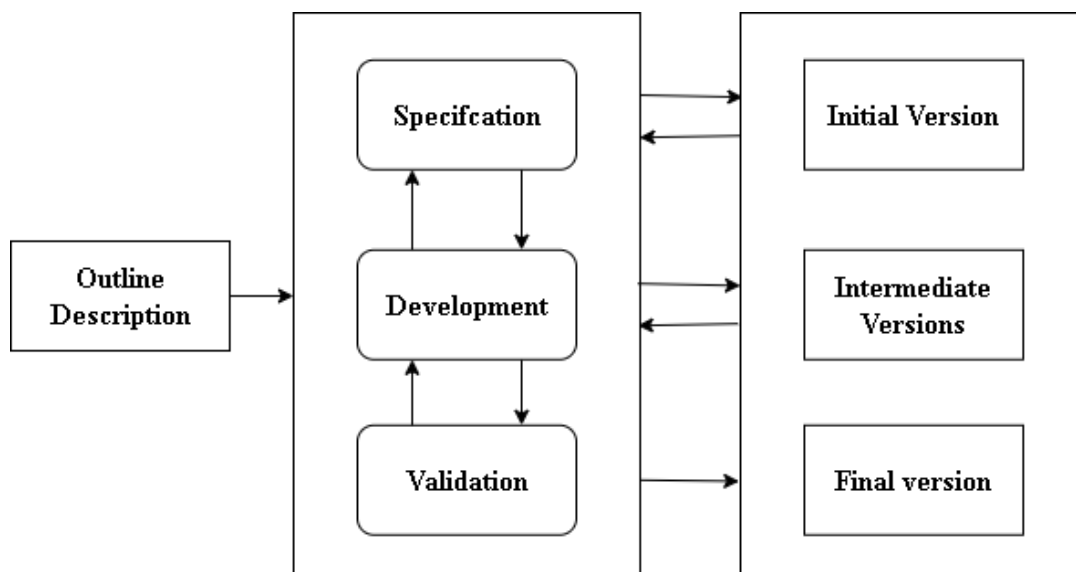


Figure 3.1 Agile Development Approach from [23]

By referring to the diagram above in Figure 3.1, there are three phases that can be seen in this approach including specification, development, and validation are interleaved. First and foremost, specification involves determining functional or non-functional requirements, often using user stories that consists of descriptions of desired functionality from consumers which are implemented directly as a series of tasks. In development phase, features and functionalities defined in the user stories will begin to be implemented into the application by organizing tasks into short time-bound iterations or sprint cycles that usually last 2 to 4 weeks. Lastly, validation

phase is mainly the application being tested and validated to ensure that each version or increments meets the specified requirements to identify issues and provide feedback [23].

At the beginning of the project, an initial version of the application will be developed first and will be released to the consumer to gain feedback. Then, this invaluable input will serve as the foundation for iteration cycles across all 3 phases. These phases will be revisited and refined multiple times, further evolving the application through several intermediate versions until ultimately developing a more robust and refined application that can be considered the final version.

3.2 System Requirement

3.2.1 Hardware

Laptop

Table 3.1 Specifications of Laptop

Description	Specifications
Model	Acer Nitro AN515-52
Operating System	Windows 10, 64-bit
Processor	Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz
Graphics Card	Intel® UHD Graphics 630
Graphics Processor	NVIDIA GeForce GTX 1050
Memory	12GB DDR4 RAM
Storage	5GB NVMe SDD 1TB WDC HDD

Mobile Phone

Table 3.2 Specifications of Mobile Phone

Description	Specifications
Model	Samsung Galaxy A33 5G
Operating System	Android 13, One UI 5.1
Processor	Samsung Exynos 1280 2.40 GHz
Memory	8GB RAM
Storage	128GB

3.2.2 Software

- Jupyter Notebook

Jupyter Notebook is an interactive, web-based computational environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. Its flexibility supports various programming languages, including Python, R, and Julia, enabling seamless integration of code and explanations within a single interface [24].

- Python Programming Language

Python is a high-level, general-purpose programming language and is known for its simplicity and ease of use. It has access to vast number of libraries and frameworks that are well-suited for image processing and computer vision tasks [25].

- Tensorflow/Keras

TensorFlow is an open-source machine learning library that provides a flexible framework to design, train and deploy various machine learning models. Within this ecosystem, Keras also serves as a high-level API that provides a highly productive and approachable interface to tackle any machine learning challenges, particularly in modern deep learning. Its comprehensive scope also covers the entirety of machine learning process from hyperparameter tuning to deployment [26][27].

- Android Studio

Android studio is an Integrated Development Environment (IDE) designed specifically for Android application development. It is a powerful code editor that provides wide range of features and tools like code completion, syntax highlighting and real-time error checking. It also consists of drag and drop elements which speeds up the UI design and even provides emulators and tools for testing applications on virtual devices [28].

- Firebase

Firestore is a comprehensive platform developed by Google for building mobile and web applications. It provides developers with various tools and services to help develop high-quality apps, grow user bases, and earn more revenue. Firestore offers a wide range of features, including authentication, real-time database, cloud storage, hosting, machine learning, and more [29].

- CVAT

CVAT is an open-source tool designed for image and video annotation. It simplifies the process of labeling images and videos for computer vision tasks such as object detection, image classification, and image segmentation. With an intuitive interface, CVAT allows users to annotate objects with polygons, rectangles, ellipses, and key points, among other shapes [30]

3.3 System Architecture Diagram

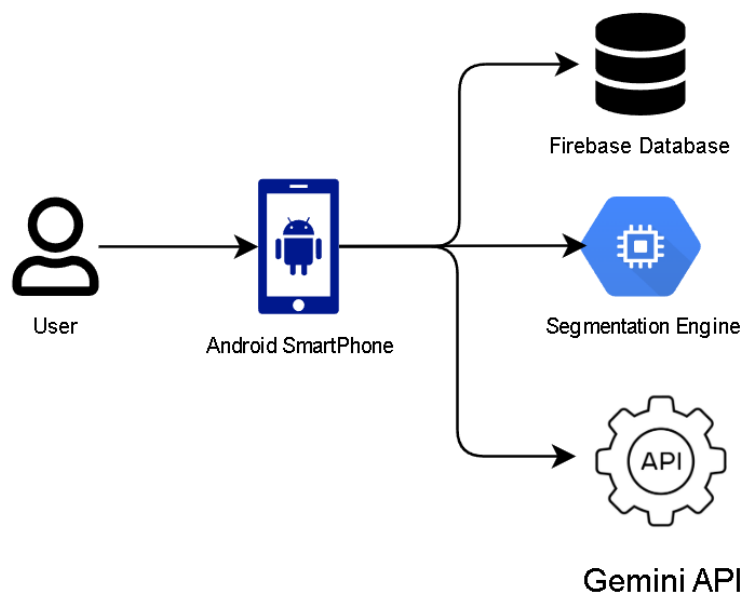


Figure 3.2 System Architecture Diagram

This system architecture utilized in the Grocery Segmentation Mobile Application, uses Firestore as a database to manage logins and user accounts, Gemini API for generating recommendations and a segmentation engine built-in locally for segmentation tasks. By utilizing the Firestore Authentication and Database service, all user's information including their account email, password and each of their health status can be retrieved and saved for account authentication and personalizing the generation of recommendations. Upon uploading an image or opening the phone's camera for segmentation tasks, the application will run the

segmentation engine built-in locally that manages the preprocessing of the image or frames, running the model and generating the results to be displayed on the user interface. Lastly, the Gemini API will be called by querying the grocery's nutrition information and health status obtained from the database to generate a more personalized recommendation for the user about selected grocery.

3.4 User Requirement

- The user should be able to upload image in their phone to be segmented.
- The user should be able to view segmentation results from an image.
- The user should be able to view segmentation results captured from the camera in real time.
- The user should be able to view labels of the food products from segmentation results.
- The user should be able to read information about the food products by selecting them.
- The user should be able to register a new account.
- The user should be able to login with an existing account.
- The user should be able to logout the current account.
- The user should be able to update their health status.
- The user should be able to reset their health status.
- The user should be able to get recommendations based on the food product selected and their health status.

3.5 Use Case Diagram and Description

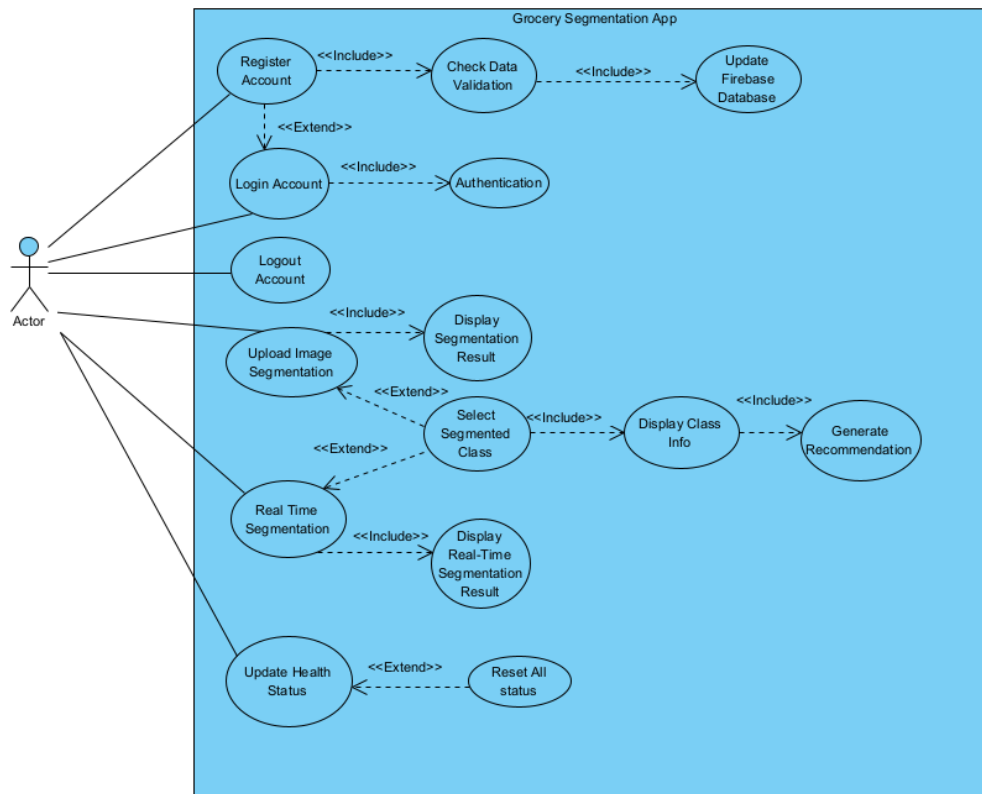


Figure 3.3 Use Case Diagram of the Application

Figure 3.3 is a use case diagram that shows the tasks, users can perform on the mobile application. By referring to the use case description below Table 3.3, 3.4, 3.5, 3.6, 3.7 and 3.8, there are 6 distinct use cases for the user to engage with the system. First and foremost, users will be able to register for a new account, if the user is new and don't have an existing account. By submitting the relevant information for registering an account, the system will validate the data inputted by the user before updating it to the database. Then, the user can login the account by inputting his/her account information which the system will authenticate the provided information before providing access to the user. Moreover, users can also choose to log out of the account whenever the user is done using the application or just want to switch accounts. Additionally, the user will be able to upload image for segmentation which will display the segmentation result upon uploaded an image. If the user is interested, the predicted groceries can be selected to display its nutrient information and generate a recommendation based on the user's profile. Furthermore, the user can also use the application for real time segmentation by pointing the phone's camera lens to the object of interest and it will display the segmentation result. Like uploading image for segmentation use case, the predicted groceries can be selected to display its nutrient information and generate a recommendation. Lastly, users will be able to

update their health status in the profile page by manually changing any status one by one or reset all status to default which is false.

Table 3.3 Use Case Description for “Register Account” Use Case

Use case ID	UC001	Use Case Name	Register Account
Primary Actor	User		
Brief Description	Users can register an account to login to the application		
Trigger	Users submit his/her inputted email, password and confirm password in the Register page by tapping on the Register button		
Precondition	User must have all the information necessary to register		
Scenario Name	Step	Action	
Main Flow	1	User accesses the Register page	
	2	User inputs his/her email	
	3	User inputs his/her password	
	4	User inputs his/her confirm password	
	5	User taps on the Register Button	
	6	System retrieve email, password and confirm password and check its validation	
	7	System updates Firebase database	
Sub Flow – Password not longer than 5	6a.1	System displays error of “Password must be at least 6 characters long”	
	6a.2	User inputs his/her email again	
	6a.3	User inputs his/her password with correct format	
	6a.4	User inputs his/her confirm password with correct format	
	6a.5	User taps on the Register Button	
	6a.6	System retrieve email, password and confirm password and check its validation	
	6a.7	System updates Firebase database	

Table 3.4 Use Case Description for “Login Account” Use Case

Use case ID	UC002	Use Case Name	Login Account
Primary Actor	User		
Brief Description	Users can login with an account to access the application’s homepage		
Trigger	Users inputs his/her email, password in the login page and submit by tapping on the Login button		
Precondition	User must have an existing registered account		
Scenario Name	Step	Action	
Main Flow	1	User accesses the Login page	
	2	User inputs his/her email	
	3	User inputs his/her password	
	4	User taps on the Login Button	
	6	System retrieves email and password and check its authentication	
	7	System displays Successfully Login	
	Sub Flow – Inputted email or password is wrong	6a.1	System displays error of “Login Failed”
6a.2		User inputs the correct email	
6a.3		User inputs the correct password	
6a.4		User taps on the Register Button	
6a.5		System retrieves email and password and check its authentication	
6a.6		System displays Successfully Login	

Table 3.5 Use Case Description for “Logout Account” Use Case

Use case ID	UC003	Use Case Name	Logout Account
Primary Actor	User		
Brief Description	Users can logout of the current account in the application		
Trigger	Users taps on the LOG OUT button in the Profile page		
Precondition	User must login with an existing account and access the Profile page		
Scenary Name	Step	Action	
Main Flow	1	User taps on the LOG OUT button	
	2	System logs out the current account and redirect user to the Login page	

Table 3.6 Use Case Description for “Upload Image Segmentation” Use Case

Use case ID	UC004	Use Case Name	Upload Image Segmentation
Primary Actor	User		
Brief Description	Users can upload image from the phone’s gallery for image segmentation		
Trigger	Users uploads an image from his/her phone’s gallery		
Precondition	User must have images in the phone’s gallery for image segmentation. User must have access to the Home page		
Scenario Name	Step	Action	
Main Flow	1	User accesses the Upload Image Segmentation page by tapping on the Upload Image Segmentation button in the Home page	
	2	User uploads an image from the phone’s gallery	
	3	System retrieves the image and runs the segmentation model	
	4	System displays the segmented image and predicted class or classes	
Sub Flow – User wants to see the class details and recommendations	4a.1	System redirects user to the specific class Info page and displays nutrient info about the class	
	4a.2	System calls the API for Gemini API with the account’s health status and class’s nutrient info to generate recommendations	
	4a.3	System displays the recommendations	

Table 3.7 Use Case Description for “Real Time Segmentation” Use Case

Use case ID	UC005	Use Case Name	Real Time Segmentation
Primary Actor	User		
Brief Description	Users can segment images captured in real time through the phone’s camera lens		
Trigger	Users taps on Real Time Segmentation button in the Home page		
Precondition	User’s phone’s camera lens must be working. User must have access to the Home page.		
Scenario Name	Step	Action	
Main Flow	1	User accesses the Real Time Segmentation page by tapping on the Real Time Segmentation button in the Home page	
	2	System opens the user’s phone’s camera lens and capture frame by frame	
	3	System retrieves the frames and runs the segmentation model	
	4	System displays the segmented frames and predicted class or classes in real time	
Sub Flow – User wants to see the class details and recommendations	4a.1	System redirects user to the specific class Info page and displays nutrient info about the class	
	4a.2	System calls the API for Gemini API with the account’s health status and class’s nutrient info to generate recommendations	
	4a.3	System displays the recommendations	

Table 3.8 Use Case Description for “Update Health Status” Use Case

Use case ID	UC006	Use Case Name	Update Health Status
Primary Actor	User		
Brief Description	Users can update its health status to be used for generating recommendations		
Trigger	Users taps on any toggle button health status display in the Profile page to change its status to either True or False		
Precondition	User must login with an existing account User must have access to the Profile page.		
Scenario Name	Step	Action	
Main Flow	1	User taps on the specific health status toggle button to update	
	2	System updates the health status to either True or False	
Sub Flow – User wants to reset all health status to default	1a.1	User taps on the RESET button	
	1a.2	System updates all health status to False	

3.6 Timeline

3.6.1 Timeline of FYP1

#	Title	Start Date	End Date	Days
1	Review of previous works	30/10/23	5/11/23	7
2	Understanding project objectives	6/11/23	7/11/23	2
3	Data collection	8/11/23	9/11/23	2
4	Data preprocessing	10/11/23	11/11/23	2
5	Develop work plan	12/11/23	12/11/23	1
6	Model development	13/11/23	16/11/23	4
7	Data Loading and Augmentation	17/11/23	20/11/23	4
8	Model training and evaluation	21/11/23	23/11/23	3
9	Optimization of the Model	24/11/23	26/11/23	3
10	Develop mobile app	27/11/23	1/12/23	5
11	Integrate app module and model	1/12/23	2/12/23	2
12	FYP1 report writing	3/12/23	7/12/23	5
13	FYP1 report submission	8/12/23	8/12/23	1

Figure 3.4 Activity done during FYP 1

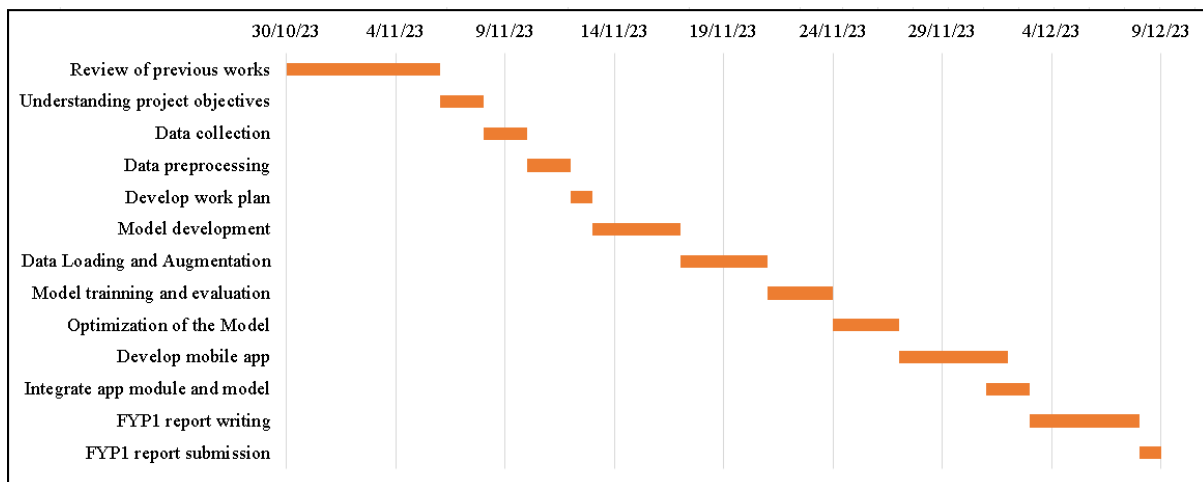


Figure 3.5 Gantt Chart of FYP1 Project Timeline

Based on Figure 3.4 and 3.5 above, the Gantt chart and Activity table show the timeline and activities done during the Final Year Project 1 (FYP1). At the beginning of the week, reviews on previous and existing works that are similar is first done to finalize the problem statements and propose the project objectives and scopes. Then, data collection and data preprocessing are done to search for suitable dataset and apply various preprocessing methods on them. Soon, a work plan was developed, detailing tasks to be done and deadlines to be reached. Following the work plan, the DeepLabV3+ model was structured and developed.

Then, begins the activity to train the model such as data loading, augmentation, and evaluation. After that, the model will be optimized to further refine it and converting it to TensorFlow Lite model, making it compatible with mobile apps. With the model finish, the development of the mobile application for real time segmentation and nutritional guidance is started. It took a week to finish developing the mobile application and integrating the application modules with the model allowing the app to do segmentation tasks. Then finally, the project concluded with the FYP1 report writing and submission.

3.6.2 Timeline of FYP2

#	Title	Start Date	End Date	Days
1	Understanding multiclass segmentation with DeepLab Model	29/1/24	2/2/24	5
2	Data labeling and preprocessing	3/2/24	3/3/24	30
3	DeepLab model development	4/3/24	10/3/24	7
4	Model training and evaluation	11/3/24	15/3/24	5
5	Optimization of the Model	16/3/24	19/3/00	4
6	Integrate app module and model	20/3/24	21/3/24	2
7	Implement register and login feature	22/3/24	28/3/24	7
8	Implement health status feature	29/3/24	6/4/24	9
9	Implement grocery info feature	7/4/24	15/4/24	9
10	Implement recommendation feature	16/4/24	21/4/24	6
11	FYP2 report writing	22/4/24	25/4/24	4
12	FYP2 report submission	26/4/24	26/4/24	1

Figure 3.6 Activity done during FYP 2

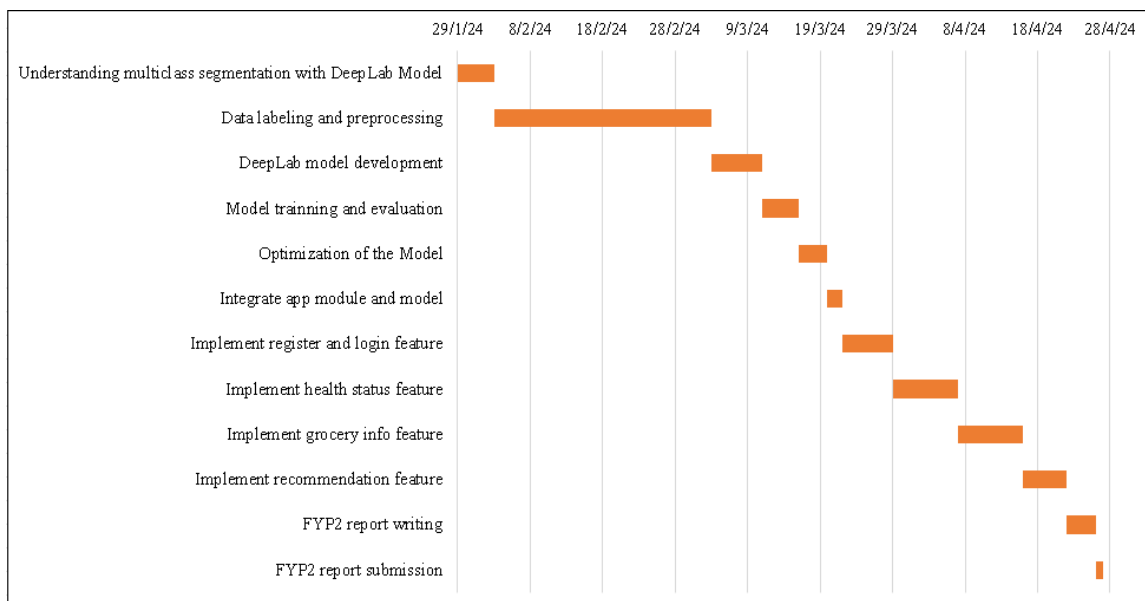


Figure 3.7 Gantt Chart of FYP2 Project Timeline

Based on Figure 3.6 and 3.7 above, the Gantt chart and Activity table show the timeline and activities done during the Final Year Project 2 (FYP2). To achieve the main objective, understanding multiclass segmentation with DeepLab model must first be carry out to switch from binary segmentation to multiclass segmentation, in order to classify and segment the object of interest. Then, for almost a month, data labeling and preprocessing will be done to create a custom dataset to be used for training. After finish creating the dataset, thus begin the development of the DeepLab model and the training and optimization of the model. It took 3 weeks of time to finalize the model and integrate it into the application for multiclass segmentation tasks which then begins the development of multiple features for the application. First, the register and login feature to ensure the security and privacy of users. Then, the health status feature that allows users to update their health status to be used for recommendations. Following that, grocery info and recommendation feature will be implemented to allow users to view information about the grocery and provide recommendations to the user. Lastly, the FYP2 report writing, and submission is done on the final week.

Chapter 4

System Design

4.1 Flowchart of the Overall Flow of the system

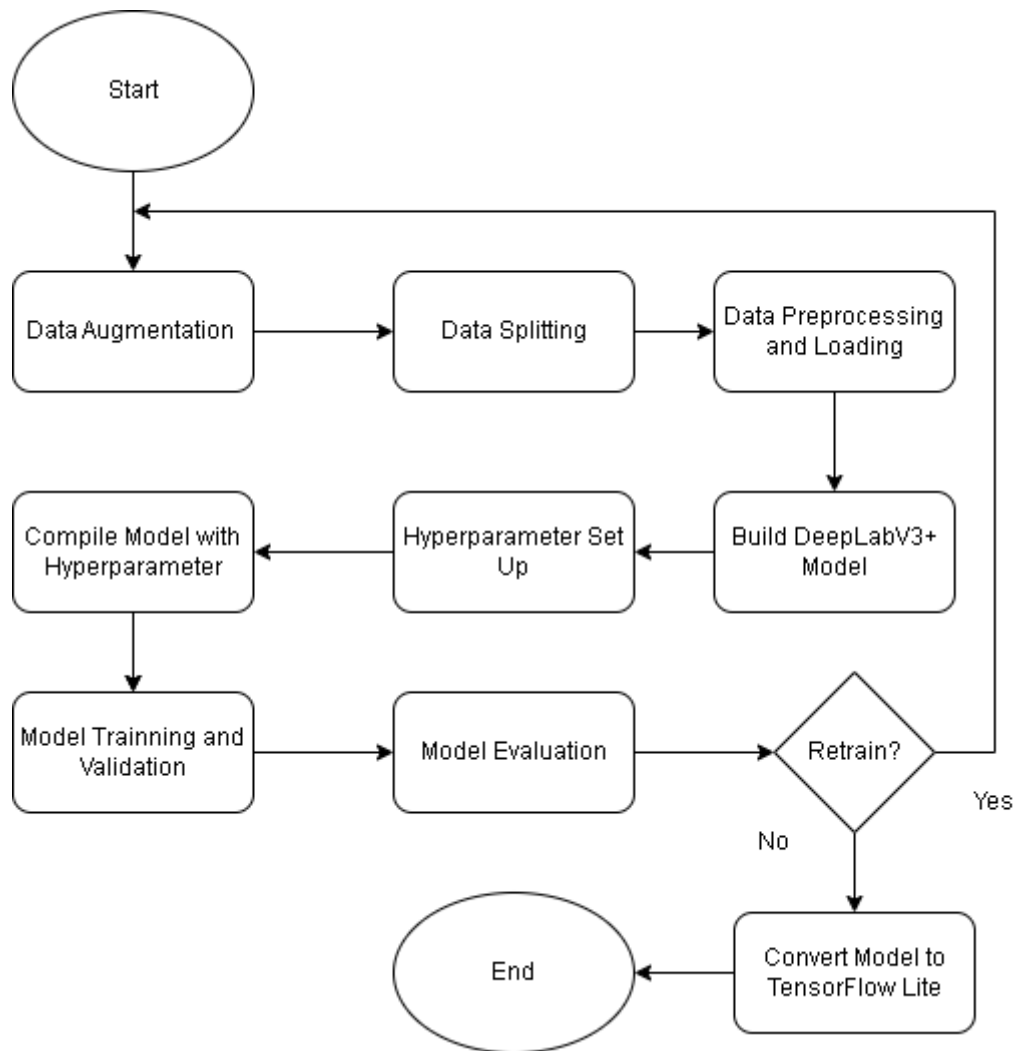


Figure 4.1 Flowchart of Training DeepLabV3+ model

The Figure 4.1 above shows the flowchart for training the DeepLabV3+ model using TensorFlow Lite [31]. Firstly, the images from the initial dataset generated from the Freiburg dataset will be augmented using ImageDataGenerator which double the size of the dataset with different random augmentations [33]. Then, the images are split into training, validation and testing sets according to the ratio of 8:1:1. Following this each of sets are then preprocessed into the right data type and image size and loaded in as Tensorflow Datasets with batch sizes. Moving on, the DeepLabV3+ model will be built from scratch and setting up hyperparameters

like Adam optimizer, Sparse Categorical Cross Entropy loss function and Accuracy Metrics to be compiled with the model [31]. Soon after, the model will begin training with train dataset and validation dataset with callbacks and 1000 epochs. After the model have finished training, it will be evaluated based on its losses and accuracy using graphs. If the model’s performance did not improve, it can be retrained again but with different augmentations or hyperparameters or if the current model’s performance is improved or acceptable, it will be converted to TensorFlow Lite model which soon to be integrated with the mobile application for segmentation tasks.

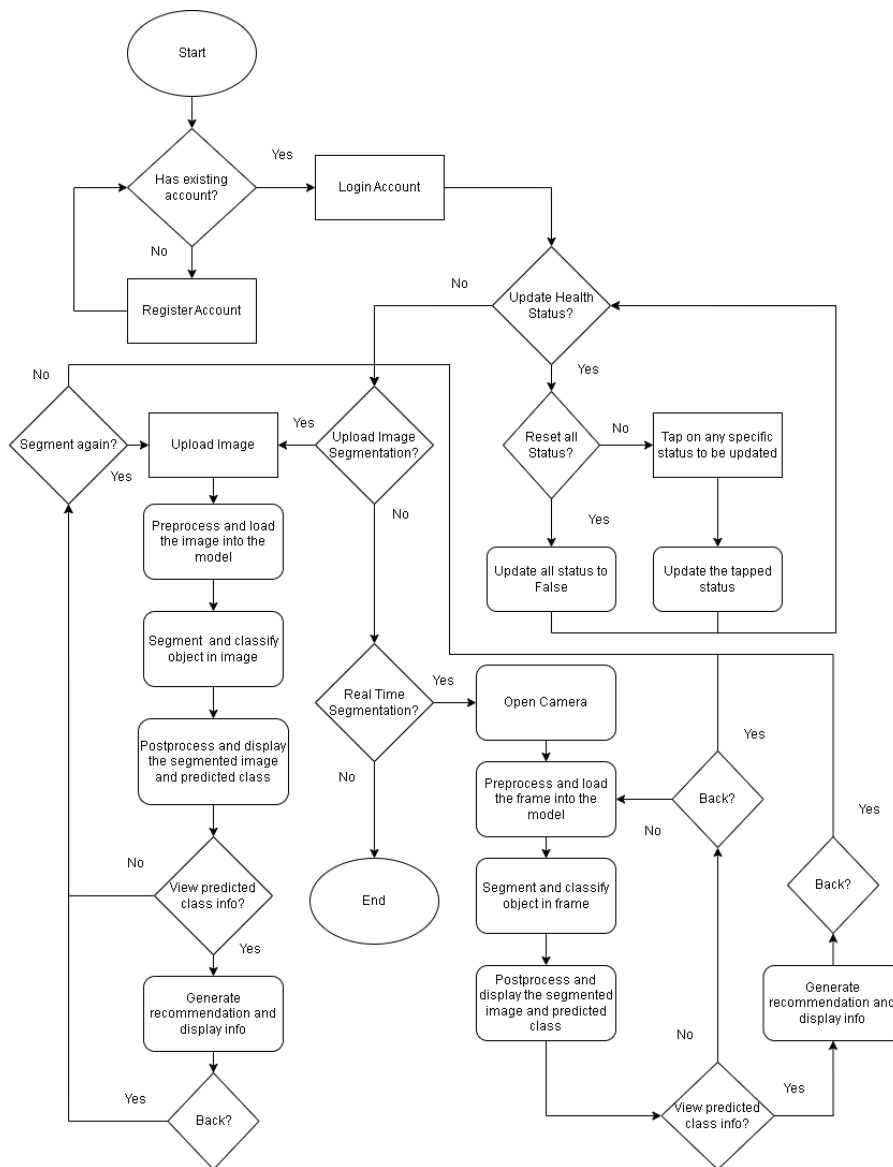


Figure 4.2 Flowchart of the Proposed Mobile Application

The Figure 3.5 above shows the flowchart of the proposed mobile application. First, if the user has an existing account, then login with it. If the user does not have an existing account, then register a new account and login with it. Upon successful login, if the user has any medical

history or body condition, the user may update the health status by going to profile page and either tapping on any specific status to be updated or tap on the reset button to reset all status to default which is False.

After finishing the update or the user has nothing to update, the user can now proceed to choose either upload image segmentation or real time segmentation located in the home page. If the user chose upload image segmentation, the application would redirect the user to the upload image segmentation page where the user can upload an image. Upon uploading an image, the system will preprocess and load the image into the model. The model will segment and classify the object, following with postprocessing and displaying the segmented image with its predicted groceries. If the user is interested in the grocery's info, the user may tap on the predicted grocery button which will redirect the user to the specific grocery info page. The application will generate the recommendation base on the user's health status and display all the nutrient info of the grocery. After finish reading, the user may go back to upload image segmentation and choose to segment again. If yes, then repeat the whole process again and if not then return to home page. If the user in the first place is not interest in the grocery's info, the user may also choose to segment again or return to home page.

Back to the home page, if the user now chooses real time segmentation, the application will redirect the user to the real time segmentation page. Upon entering the page, it will automatically preprocess each frame captured by the camera and load them into the model. The model will then segment the frames and immediately display the segmented frame to the user. The whole process will then continue in a loop, segmenting and displaying the frames in real-time. Like upload image segmentation, if the user is interested in the grocery's info, the user may tap on the predicted grocery button which will redirect the user to the specific grocery info page. The application will then generate the recommendation base on the user's health status and display all the nutrient info of the grocery. the user may go back to real time segmentation after finish reading or return to home page. If the user originally not interested in the grocery's info, the user may choose to continue let it segment or return to the home page. Finally, all options in the application have been fully explored, the user can choose to continue the segmentation or end the system by closing the application.

4.2 Dataset



Figure 4.3 Freiburg Groceries Dataset from [32]

In this work, the dataset is created from the University of Freiburg in Germany and named the Freiburg Groceries Dataset [33]. It consists of 4947 images in PNG format covering 25 grocery classes, with 97 to 370 images per class. The images were captured using 4 different smartphone cameras at various stores in Freiburg, Germany, thus it also includes a large variety of perspectives, lightning conditions, and degrees of clutter. However, it does not contain any labelled mask ask ground truth for each of the images, since it is originally used for classification tasks. Therefore, CVAT an image annotation tool is used to label the images manually which helps simplify the process of labelling images. Still, it took almost a month, to label and transform only 50 images for each 14 different grocery classes with ground truth. In the end, the custom dataset that has been created has a total of 700 images with 14 grocery classes including, Beans, Cake, Candy, Cereal, Chips, Chocolate, Coffee, Corn, Flour, Honey, Jam, Juice, and Milk.

4.3 Model Architecture

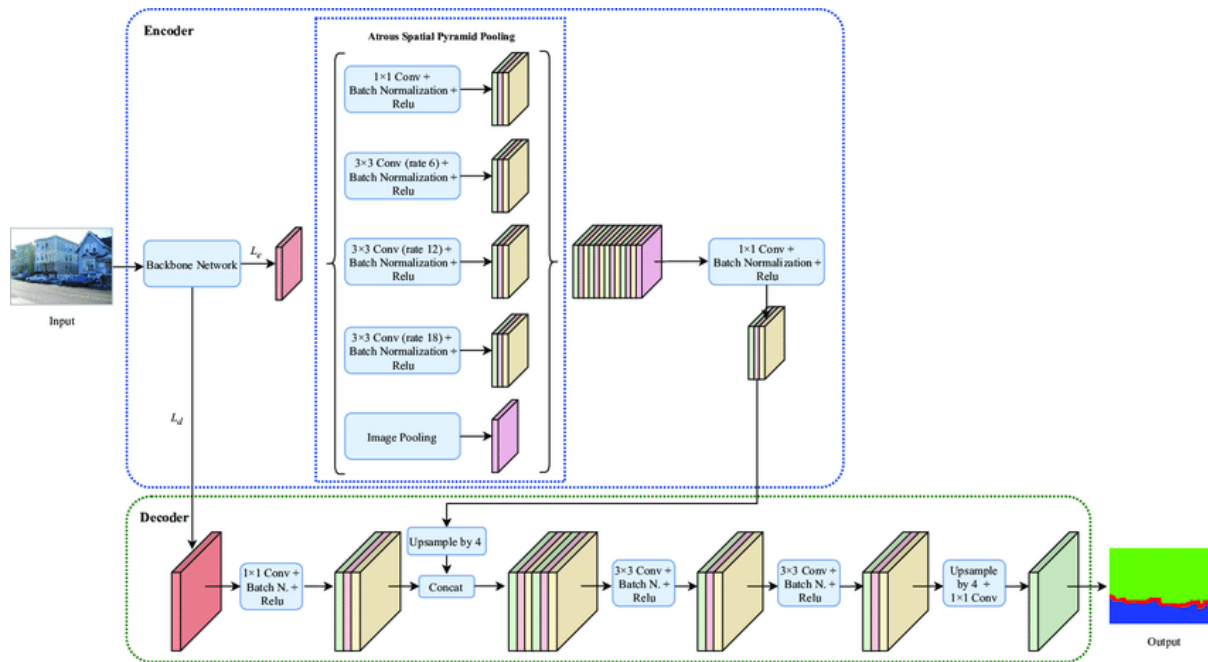


Figure 4.4 DeepLabV3+ Architecture from [31]

DeepLabV3+ is a state-of-the-art deep learning model, which is widely used for semantic image segmentation. This model is an extension of the original DeepLabV3, which combines Atrous Spatial Pyramid Pooling (ASPP) from DeepLabv1 and Encoder Decoder Architecture from DeepLabv2. It is first introduced in the paper titled "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," that enhances the original DeepLabV3 model by incorporating an additional decoder module [34]. The introduction of the decoder module has improved in capturing sharper object boundaries and finer details in the segmented images.

In the encoder of DeepLabV3+, the backbone network is a critical component of DeepLabV3+ and is responsible for extracting features from the input image [35]. These features are then used by subsequent modules for making pixel-wise predictions. The backbone network typically comprises a convolutional neural network (CNN) pre-trained on a large dataset, such as ImageNet [35]. DeepLabV3+ supports various backbone networks, including ResNet and VGG. These networks will serve as the feature extractors and extracting features to be passed to Atrous Spatial Pyramid Pooling (ASPP) and decoder part of the DeepLabV3+ architecture.

Once the input image is passed through the backbone network, the extracted features are then further processed by the ASPP module. It is responsible for capturing multi-scale contextual information, enabling the model to achieve state-of-the-art performance in semantic

segmentation tasks [34]. It aggregates features from multiple atrous convolutional layers with different rates of dilation, allowing the model to understand the image at various scales [34]. Based on the architecture above in Figure 4.4, ASPP consist of one convolution with a kernel size of 1×1 , three convolutions with a kernel size of 3×3 and dilation rates of 6, 12, and 18, respectively and image-level features with image pooling. The resulting features from the five branches are concatenated together and passed through a 1×1 convolution and upsample by a factor of 4 to be passed to the decoder for segmentation [31].

In the decoder of DeepLabV3+, the extracted features from the backbone network will be passed through a 1×1 convolution which then will later be concatenated with the features of the encoder to create a skip connection to incorporate both low-level and high-level features, enhancing the model's ability to capture fine-grained details and contextual information. The resulted feature maps will then pass through 3×3 convolutional layers and finally upsample by a factor of 4 again to be fed through a 1×1 convolution to produce the final output [31].

4.4 System Performance Evaluation

To evaluate the performance of the groceries' segmentation, after segmenting an uploaded image or during real-time segmentation, accuracy percentage of all the predicted groceries will be calculated and displayed together with their respective groceries. All predicted groceries will also be displayed on the user interface with distinct colors that correspond to each mask generated of the image, helping in clearing the identification of the segmented items. The evaluation metrics used is Accuracy which is defined as in equation 3.2: [36]

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \quad (3.2)$$

where True Positive (TP) is the pixels that are actual positive pixels predicted as positive, True Negative (TN) is actual negative pixels predicted as negative, False Positive (FP) is pixels predicted as positive but is actual negative pixels and False Negative (FN) is actual positive pixels but predicted as negative pixels. This metric will help monitor how well the model performs and provides a quick insight into the model's performance.

Chapter 5

System Implementation

5.1 Importing Necessary Libraries

```
import os
import random
import shutil
import cv2
import pickle
import matplotlib.pyplot as plt
from glob import glob
import numpy as np
import sklearn as sk
from PIL import Image

import tensorflow as tf
from tensorflow import image as tf_image
from tensorflow import data as tf_data
from tensorflow import io as tf_io
from tensorflow import keras
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard

#For Deeplab
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, UpSampling2D
from tensorflow.keras.layers import AveragePooling2D, Conv2DTranspose, Concatenate, Input
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.applications import ResNet50, VGG16, ResNet101, VGG19
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Figure 5.1 Importing the Necessary Libraries

Before model creation, all the necessary libraries for the model are imported initially as shown above in Figure 5.1. They are important as it ensures that all required functionalities and tools are available and accessible during the model construction process. Majority of the libraries are from TensorFlow and Keras for data preprocessing and model building such as `tf_image`, `tf_data`, `tf_io`, image data generator, layers, models, optimizers, losses, applications. There are also other imported libraries that are necessary in model development are including `os` for tasks like file and directory, `random` for generating random numbers or shuffling sequences, `shutil` for copying and removing files and directories, `cv2` for image processing tasks, `pickle` for saving or loading model configurations, `matplotlib` for plotting graphs and

visualizations, glob for finding pathnames, numpy for numerical operations, sklearn for splitting datasets and finally PIL for image adding image processing capabilities to your Python interpreter

5.2 Initialize Parameters and Seedings

```
IMAGE_SIZE = 256
BATCH_SIZE = 16
NUM_CLASSES = 15
TRAIN_PERCENT = 0.8
VAL_PERCENT = 0.1
TEST_PERCENT = 0.1

ROOT_DIR = "custom_dataset/multiclass"
IMAGES_DIR = "custom_dataset/multiclass/images"
MASKS_DIR = "custom_dataset/multiclass/masks"
AUG_IMAGES_DIR = "custom_dataset/multiclass/aug_images"
AUG_MASKS_DIR = "custom_dataset/multiclass/aug_masks"
TRAIN_DIR = "custom_dataset/multiclass/train"
TRAIN_IMG_DIR = "custom_dataset/multiclass/train/images"
TRAIN_MASKS_DIR = "custom_dataset/multiclass/train/masks"
VAL_DIR = "custom_dataset/multiclass/val"
VAL_IMG_DIR = "custom_dataset/multiclass/val/images"
VAL_MASKS_DIR = "custom_dataset/multiclass/val/masks"
TEST_DIR = "custom_dataset/multiclass/test"
TEST_IMG_DIR = "custom_dataset/multiclass/test/images"
TEST_MASKS_DIR = "custom_dataset/multiclass/test/masks"

SEED = 42
```

Figure 5.2 Parameters and Seedings

As shown above in Figure 5.2, the initialized parameters for image size is 256, batch size is 16, number of classes is 15, train, validation and test percentage with a ratio of 8:1:1 for splitting, all directories path for augmentation, preprocessing and loading, followed by a seed value of 42 and utilizing random number generator for the NumPy to produce the same results each time any random operations using NumPy functions that are executed will produce the same result. All of these parameters will be configured later to fine tune the model.

5.3 Data Augmentation

5.3.1 Defining Image Data Generator

```
# Set up ImageDataGenerator for data augmentation
image_data_gen_args = dict(
    rotation_range=30,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    brightness_range=[0.4,1.5],
    fill_mode='nearest')

mask_data_gen_args = dict( rotation_range=30,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest')

image_datagen = ImageDataGenerator(**image_data_gen_args)
mask_datagen = ImageDataGenerator(**mask_data_gen_args)

# Flow from directory
image_generator = image_datagen.flow_from_directory(
    IMAGES_DIR,
    class_mode=None,
    seed=SEED,
    color_mode='rgb',
    save_format='png',
    target_size=(IMAGE_SIZE, IMAGE_SIZE))

mask_generator = mask_datagen.flow_from_directory(
    MASKS_DIR,
    class_mode=None,
    color_mode='grayscale',
    seed=SEED,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    save_format='png')

# Combine generators into one which yields image and masks
train_generator = zip(image_generator, mask_generator)
```

Figure 5.3 Defining Image Data Generator

After that as shown above in Figure 5.3, different augmentation techniques using ImageDataGenerator from Keras are utilized to perform data augmentation on the dataset such as

- `rotation_range = 30` for randomly rotating the image within the range of -30 to 30 degrees.
- `shear_range = 0.3` for applying shear transformation to the image, changing the angle of individual pixels.
- `zoom_range = 0.3` for randomly zooming into images up to 30 %.
- `horizontal_flip = True` for randomly flipping the images horizontally.
- `vertical_flip = True` for randomly flipping the image vertically.
- `brightness_range = [0.4,1.5]` for adjusting the brightness of the image randomly within the specified range.
- `fill_mode = nearest` for filling in newly created pixels with the nearest existing pixel values after augmentation

All of these techniques are applied into both image and mask datagen dictionaries, except for `brightness_range` which is only applied to the image datagen since it would affect the pixel values of the mask. Then, both datagen will call flow from directory from respective paths to load all the images and masks as generators and proceed to zip them together as train generator.

5.3.2 Applying Data Augmentation

```
# Define the directory to save augmented images and masks
aug_img_dir = f'{ROOT_DIR}/aug_images'
aug_mask_dir = f'{ROOT_DIR}/aug_masks'
if not os.path.exists(aug_img_dir):
    os.makedirs(aug_img_dir)
if not os.path.exists(aug_mask_dir):
    os.makedirs(aug_mask_dir)

# Number of augmented images you want to generate
num_images_to_generate = 1400

# Loop through the generator to generate and save augmented images and masks together
for i, (image, mask) in enumerate(train_generator):
    if i >= num_images_to_generate:
        break

    # Get the augmented image and mask
    augmented_image = image[0] # Assuming batch size is 1
    augmented_image = augmented_image.astype('uint8') # Convert to uint8 before saving

    augmented_mask = mask[0] # Assuming batch size is 1
    augmented_mask = augmented_mask.astype('uint8') # Convert to uint8 before saving

    # Save the augmented image
    augmented_image_path = os.path.join(aug_img_dir, f'augmented_image_{i}.png')
    Image.fromarray(augmented_image).save(augmented_image_path)

    # Save the augmented mask
    augmented_mask_path = os.path.join(aug_mask_dir, f'augmented_mask_{i}.png')
    Image.fromarray(augmented_mask.squeeze(axis=-1)).save(augmented_mask_path) #D Assuming masks are grayscale

    print(f"Saved augmented image and mask {i+1}/{num_images_to_generate}")

print("Augmented images and masks saved successfully.")
```

Figure 5.4 Applying Data Augmentation

After finish creating the train generator, the script shown above In Figure 5.4 then loops through the generator, generating and saving augmented images and masks until there are 1400 augmented images generated which is double the size of the dataset. It then will retrieve the augmented images and masks from the generator, transforming and saving them into augmented images and augmented masks directories.

5.4 Data Splitting

```
# Define the directories
source_image_dir = AUG_IMAGES_DIR
source_mask_dir = AUG_MASKS_DIR
os.makedirs(TRAIN_IMG_DIR, exist_ok=True)
os.makedirs(TRAIN_MASKS_DIR, exist_ok=True)
os.makedirs(VAL_IMG_DIR, exist_ok=True)
os.makedirs(VAL_MASKS_DIR, exist_ok=True)
os.makedirs(TEST_IMG_DIR, exist_ok=True)
os.makedirs(TEST_MASKS_DIR, exist_ok=True)
train_dir = TRAIN_DIR
val_dir = VAL_DIR
test_dir = TEST_DIR

# Create train, val, and test directories if they don't exist
for directory in [train_dir, val_dir, test_dir]:
    if not os.path.exists(directory):
        os.makedirs(os.path.join(directory, 'images'))
        os.makedirs(os.path.join(directory, 'masks'))

# List all image and mask files
image_files = os.listdir(source_image_dir)
mask_files = os.listdir(source_mask_dir)

# Zip together image and mask files
image_mask_pairs = list(zip(image_files, mask_files))

# Shuffle the pairs
random.shuffle(image_mask_pairs)

# Calculate split sizes
total_pairs = len(image_mask_pairs)
train_size = int(0.8 * total_pairs)
val_size = int(0.1 * total_pairs)

# Split the pairs
train_pairs = image_mask_pairs[:train_size]
val_pairs = image_mask_pairs[train_size:train_size + val_size]
test_pairs = image_mask_pairs[train_size + val_size:]

# Function to copy files from source to destination
def copy_files(pairs, src_dir_images, src_dir_masks, dest_dir):
    for image_file, mask_file in pairs:
        shutil.copy(os.path.join(src_dir_images, image_file), os.path.join(dest_dir, 'images', image_file))
        shutil.copy(os.path.join(src_dir_masks, mask_file), os.path.join(dest_dir, 'masks', mask_file))

# Copy files to train directory
copy_files(train_pairs, source_image_dir, source_mask_dir, train_dir)
# Copy files to val directory
copy_files(val_pairs, source_image_dir, source_mask_dir, val_dir)
# Copy files to test directory
copy_files(test_pairs, source_image_dir, source_mask_dir, test_dir)
```

Figure 5.5 Data Splitting

After that, the script shown above in Figure 5.5 will split the augmented images and augmented masks into 80% for training, 10% for validation and 10% for testing or ratio of 8:1:1 and copied into training, validation and testing directories. This results in a total of 1120 images for the training set and 140 images for both validation and testing set as shown above in Figure 5.6

```

train_images = sorted(glob(os.path.join(TRAIN_IMG_DIR, "**")))
train_masks = sorted(glob(os.path.join(TRAIN_MASKS_DIR, "**")))
val_images = sorted(glob(os.path.join(VAL_IMG_DIR, "**")))
val_masks = sorted(glob(os.path.join(VAL_MASKS_DIR, "**")))
test_images = sorted(glob(os.path.join(TEST_IMG_DIR, "**")))
test_masks = sorted(glob(os.path.join(TEST_MASKS_DIR, "**")))
print("Total train images:", len(train_images))
print("Total train masks:", len(train_masks))
print("Total val images:", len(val_images))
print("Total val masks:", len(val_masks))
print("Total test images:", len(test_images))
print("Total test masks:", len(test_masks))

Total train images: 1120
Total train masks: 1120
Total val images: 140
Total val masks: 140
Total test images: 140
Total test masks: 140

```

Figure 5.6 Total Images for Training, Validation and Testing Set

5.5 Data Preprocessing and Loading

```

def read_image(image_path, mask=False):
    image = tf.io.read_file(image_path)
    if mask:
        image = tf_image.decode_png(image, channels=1)
        image.set_shape([None, None, 1])
        image = tf_image.resize(images=image, size=[IMAGE_SIZE, IMAGE_SIZE])
    else:
        image = tf_image.decode_png(image, channels=3)
        image.set_shape([None, None, 3])
        image = tf_image.resize(images=image, size=[IMAGE_SIZE, IMAGE_SIZE])
    return image

def load_data(image_list, mask_list):
    image = read_image(image_list)
    mask = read_image(mask_list, mask=True)
    return image, mask

def data_generator(image_list, mask_list):
    dataset = tf_data.Dataset.from_tensor_slices((image_list, mask_list))
    dataset = dataset.map(load_data, num_parallel_calls=tf_data.AUTOTUNE)
    dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
    return dataset

train_dataset = data_generator(train_images, train_masks)
val_dataset = data_generator(val_images, val_masks)
test_dataset = data_generator(test_images, test_masks)

print("Train Dataset:", train_dataset)
print()
print("Val Dataset:", val_dataset)
print()
print("Test Dataset:", test_dataset)

Train Dataset: <BatchDataset element_spec=(TensorSpec(shape=(16, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(16, 256, 256, 1), dtype=tf.float32, name=None))>

Val Dataset: <BatchDataset element_spec=(TensorSpec(shape=(16, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(16, 256, 256, 1), dtype=tf.float32, name=None))>

Test Dataset: <BatchDataset element_spec=(TensorSpec(shape=(16, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(16, 256, 256, 1), dtype=tf.float32, name=None))>

```

Figure 5.7 Data Preprocessing and Loading

After splitting, images from training, validation and testing directories will be preprocessed and loaded in as TensorFlow Datasets using `load_data` and `read_image` function. By running the script shown above in Figure 5.7, the images will be resized to a square of the required

image size with three channels while the masks will be resized to a square of the required image size with single channels, making sure all the images size and type are correct. Then, the data_generator function will be mapping and batching the respective image data and their corresponding masks, setting them up as train dataset, validation dataset and test dataset.

5.6 Building DeepLabV3+

```

""" Atrous Spatial Pyramid Pooling """
def ASPP(inputs):
    shape = inputs.shape

    y_pool = AveragePooling2D(pool_size=(shape[1], shape[2]), name='average_pooling')(inputs)
    y_pool = Conv2D(filters=256, kernel_size=1, padding='same', use_bias=False)(y_pool)
    y_pool = BatchNormalization(name='bn_1')(y_pool)
    y_pool = Activation('relu', name='relu_1')(y_pool)
    y_pool = UpSampling2D((shape[1], shape[2]), interpolation="bilinear")(y_pool)

    y_1 = Conv2D(filters=256, kernel_size=1, dilation_rate=1, padding='same', use_bias=False)(inputs)
    y_1 = BatchNormalization()(y_1)
    y_1 = Activation('relu')(y_1)

    y_6 = Conv2D(filters=256, kernel_size=3, dilation_rate=6, padding='same', use_bias=False)(inputs)
    y_6 = BatchNormalization()(y_6)
    y_6 = Activation('relu')(y_6)

    y_12 = Conv2D(filters=256, kernel_size=3, dilation_rate=12, padding='same', use_bias=False)(inputs)
    y_12 = BatchNormalization()(y_12)
    y_12 = Activation('relu')(y_12)

    y_18 = Conv2D(filters=256, kernel_size=3, dilation_rate=18, padding='same', use_bias=False)(inputs)
    y_18 = BatchNormalization()(y_18)
    y_18 = Activation('relu')(y_18)

    y = Concatenate()([y_pool, y_1, y_6, y_12, y_18])

    y = Conv2D(filters=256, kernel_size=1, dilation_rate=1, padding='same', use_bias=False)(y)
    y = BatchNormalization()(y)
    y = Activation('relu')(y)
    return y

```

Figure 5.8 ASPP

```

def DeepLabV3Plus(shape, encoder):
    """ Inputs """
    inputs = Input(shape)

    """ Pre-trained VGG16 """
    if encoder == "vgg16":
        base_model = VGG16(weights='imagenet', include_top=False, input_tensor=inputs)
        print("Using VGG16 as Encoder")

        """ Pre-trained VGG16 Output """
        image_features = base_model.get_layer('block5_conv3').output # Choosing a Layer in the middle for feature extraction
        x_a = ASPP(image_features)
        x_a = UpSampling2D((4, 4), interpolation="bilinear")(x_a)

        """ Get low-level features """
        x_b = base_model.get_layer('block3_conv3').output # Choosing a Layer in the early stage for low-level features
        x_b = Conv2D(filters=48, kernel_size=1, padding='same', use_bias=False)(x_b)
        x_b = BatchNormalization()(x_b)
        x_b = Activation('relu')(x_b)

        x = Concatenate()([x_a, x_b])

        x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

        x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)
        x = UpSampling2D((4, 4), interpolation="bilinear")(x)

        """ Outputs """
        x = Conv2D(NUM_CLASSES, (1, 1), name='output_layer')(x)
        x = Activation('softmax')(x)

        """ Model """
        model = Model(inputs=inputs, outputs=x)
        return model

```

Figure 5.9 DeepLabV3+ with VGG16

```

""" Pre-trained VGG19 """
if encoder == "vgg19":
    base_model = VGG19(weights='imagenet', include_top=False, input_tensor=inputs)
    print("Using VGG19 as Encoder")

    """ Pre-trained VGG19 Output """
    image_features = base_model.get_layer('block5_conv4').output # Choosing a Layer in the middle for feature extraction
    x_a = ASPP(image_features)
    x_a = UpSampling2D((4, 4), interpolation="bilinear")(x_a)

    """ Get low-level features """
    x_b = base_model.get_layer('block3_conv4').output # Choosing a Layer in the early stage for low-level features
    x_b = Conv2D(filters=48, kernel_size=1, padding='same', use_bias=False)(x_b)
    x_b = BatchNormalization()(x_b)
    x_b = Activation('relu')(x_b)

    x = Concatenate()([x_a, x_b])

    x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = UpSampling2D((4, 4), interpolation="bilinear")(x)

    """ Outputs """
    x = Conv2D(NUM_CLASSES, (1, 1), name='output_layer')(x)
    x = Activation('softmax')(x)

    """ Model """
    model = Model(inputs=inputs, outputs=x)
    return model

```

Figure 5.10 DeepLabV3+ with VGG19

```

""" Pre-trained ResNet50 """
if encoder == "resnet50":
    base_model = ResNet50(weights='imagenet', include_top=False, input_tensor=inputs)
    print("Using ResNet50 as Encoder")

    """ Pre-trained ResNet50 Output """
    image_features = base_model.get_layer('conv4_block6_out').output
    x_a = ASPP(image_features)
    x_a = UpSampling2D((4, 4), interpolation="bilinear")(x_a)

    """ Get low-level features """
    x_b = base_model.get_layer('conv2_block2_out').output
    x_b = Conv2D(filters=48, kernel_size=1, padding='same', use_bias=False)(x_b)
    x_b = BatchNormalization()(x_b)
    x_b = Activation('relu')(x_b)

    x = Concatenate()([x_a, x_b])

    x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = UpSampling2D((4, 4), interpolation="bilinear")(x)

    """ Outputs """
    x = Conv2D(NUM_CLASSES, (1, 1), name='output_layer')(x)
    x = Activation('softmax')(x)

    """ Model """
    model = Model(inputs=inputs, outputs=x)
    return model

```

Figure 5.11 DeepLabV3+ with ResNet50

```

if encoder == "resnet101":
    base_model = ResNet101(weights='imagenet', include_top=False, input_tensor=inputs)
    print("Using ResNet101 as Encoder")

    """ Pre-trained ResNet101 Output """
    image_features = base_model.get_layer('conv4_block23_out').output
    x_a = ASPP(image_features)
    x_a = UpSampling2D((4, 4), interpolation="bilinear")(x_a)

    """ Get low-level features """
    x_b = base_model.get_layer('conv2_block2_out').output
    x_b = Conv2D(filters=48, kernel_size=1, padding='same', use_bias=False)(x_b)
    x_b = BatchNormalization()(x_b)
    x_b = Activation('relu')(x_b)

    x = Concatenate()([x_a, x_b])

    x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = UpSampling2D((4, 4), interpolation="bilinear")(x)

    """ Outputs """
    x = Conv2D(NUM_CLASSES, (1, 1), name='output_layer')(x)
    x = Activation('softmax')(x)

    """ Model """
    model = Model(inputs=inputs, outputs=x)
    return model

if __name__ == "__main__":
    input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3)
    #encoder = "vgg16"
    #encoder = "vgg19"
    #encoder = "resnet50"
    encoder = "resnet101"
    model = DeepLabV3Plus(input_shape, encoder)
    model.summary()

```

Figure 5.12 DeepLabV3+ with ResNet101

With the data preprocessing and augmentation done, the DeepLabV3+ model will be built according to the architecture from [31] with different backbone networks. Firstly, as show above in Figure 5.8, the ASPP module of the model is built by calling the ASPP function that begins by setting the shape of the input. Then, it constructs multiple branches to capture context at different scales. The first branch employs global average pooling followed by a 1x1 convolution, providing a broad context view. The subsequent branches involve 3x3 convolutions with different dilation rates (6, 12, and 18), enabling the network to capture contextual information with increasing receptive field sizes. Another branch performs a 1x1 convolution directly on the input, capturing local context. After processing all branches, their outputs are concatenated along the channel axis to fuse multi-scale context. Finally, a 1x1

convolution is applied to this concatenated result to reduce the number of channels to 256, effectively fusing the context information.

After the ASPP function is completed, the model will now be built with different backbone network including VGG16, VGG19, ResNet50 and Resnet101 based on the images above in Figure 5.9, 5.10, 5.11 and 5.12 [37] [38]. Firstly, the backbone network is loaded with pre-trained ImageNet weights. Then, the image features are taken from the higher-level layers which are “block5_conv3” from VGG16, “block5_conv4” from VGG19, “conv4_block6_out” from Resnet50 and “conv4_block23_out” from ResNet101. The output will then be applied to the ASPP module to capture multi-scale features and upsample the result by a factor of 4.

Additionally, low-level features from lower-level layers which are “block3_conv3” from VGG16, “block3_conv4” from VGG19, “conv2_block2_out” from ResNet50 and ResNet101 and pass the features through the 1x1 convolution to reduce the number of filters to 48.

Then, concatenate the high-level features and low-level features and process the result through 2 3x3 convolutional layers. Finally, the feature maps will be mapped using a 1x1 convolution to its desired number of classes which in this case is 15 and obtain the class probabilities using the SoftMax activation function.

5.7 Hyperparameter Set Up and Model Training

```
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

checkpointer = ModelCheckpoint(f'deeplab_multiclass_{encoder}.h5', verbose=1, save_best_only=True)

callbacks = [
    EarlyStopping(patience=15, verbose=1),
    checkpointer
]

history = model.fit(train_dataset, validation_data=val_dataset, epochs=1000, callbacks=callbacks)

#save history
with open(f'deeplab_multiclass_{encoder}.pkl', 'wb') as file:
    pickle.dump(history.history, file)
```

Figure 5.13 Hyperparameter Set Up and Model Training

With the DeepLabV3+ model completed, now the model can be created by specifying the input and output layer and set up the hyperparameters for model training [31]. Firstly, the optimizer will be using Adam which is an adaptive learning rate optimization algorithm used for updating the weights during training [39]. As for the loss function, Sparse categorical Cross-Entropy is utilized for multiclass classification tasks and finally Accuracy metrics be used as

the evaluation metrics during training and testing which calculates the accuracy of the model predictions.

Then, the model will begin training with its training dataset and validation dataset to evaluate the model's performance after each epochs. The dataset is 16 in batch size, learning rate is set to 0.001 to help find-tuning the model's performance gradually and epochs is set to 1000 just to allow the model to train as long as possible and save the model that achieved the best performance on validation dataset, until the EarlyStopping is called when no improvement is observed after 15 epochs. The model's history will then be dump into a pickle file to save it for model evaluation and comparison

5.8 Model Evaluation and Conversion

```
# Load saved model
model = load_model("deeplab_multiclass_resnet101.h5")

# Open the pickle file for reading
with open('deeplab_multiclass_resnet101.pkl', 'rb') as file:
    history = pickle.load(file)

avg_train_acc = np.mean(history['accuracy']) #history.history['accuracy']
print("Average Accuracy for Training Set = ", (avg_train_acc * 100.0), "%")

avg_val_acc = np.mean(history['val_accuracy']) #history.history['val_accuracy']
print("Average Accuracy for Validation Set = ", (avg_val_acc * 100.0), "%")

test_loss, test_acc = model.evaluate(test_dataset)
print("Accuracy for Testing Set = ", (test_acc * 100.0), "%")
```

Figure 5.14 Model Evaluation 1

```

plt.plot(history["loss"])
plt.title("Training Loss")
plt.ylabel("loss")
plt.xlabel("epoch")
plt.show()

plt.plot(history["accuracy"])
plt.title("Training Accuracy")
plt.ylabel("accuracy")
plt.xlabel("epoch")
plt.show()

plt.plot(history["val_loss"])
plt.title("Validation Loss")
plt.ylabel("val_loss")
plt.xlabel("epoch")
plt.show()

plt.plot(history["val_accuracy"])
plt.title("Validation Accuracy")
plt.ylabel("val_accuracy")
plt.xlabel("epoch")
plt.show()

```

Figure 5.15 Model Evaluation 2

```

#Converting a tf.Keras model to a TensorFlow Lite model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the Lite model
with open(f"deeplab_multiclass_{encoder}.tflite", "wb") as f:
    f.write(tflite_model)

```

Figure 5.16 Model Conversion

After finish training the script shown above in Figures 5.13 and 5.14 are run to obtain the model's overall accuracy on the training, validation and testing set and plot graphs base on its training loss, training accuracy, validation loss and validation accuracy to gain some insights into its performance. By evaluating and comparing all the trained models, the highest performance model will be picked and convert from a DeepLabV3+ model to TensorFlow Lite Model to be used for multiclass segmentation tasks in the mobile application [31].

5.9 Prerequisites for Mobile Application Development

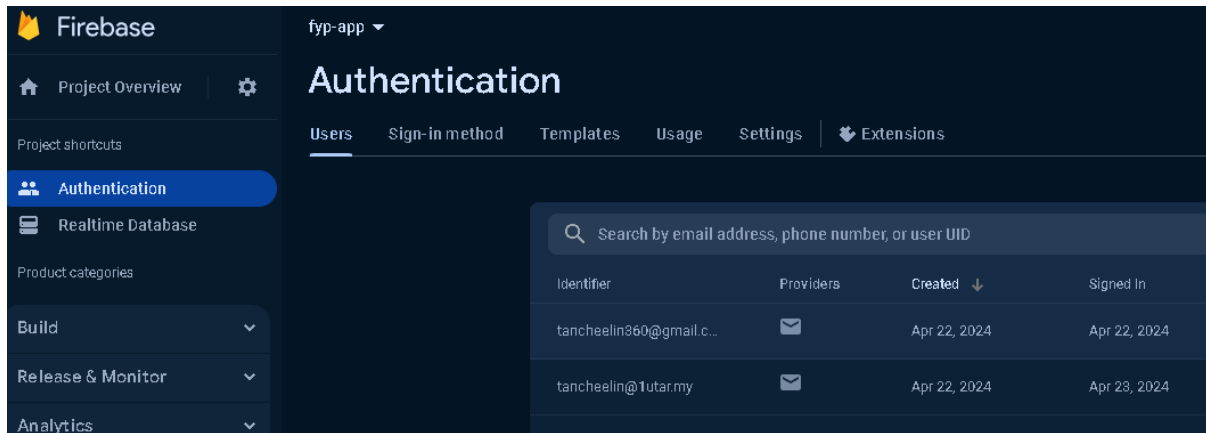


Figure 5.17 Firebase Authentication

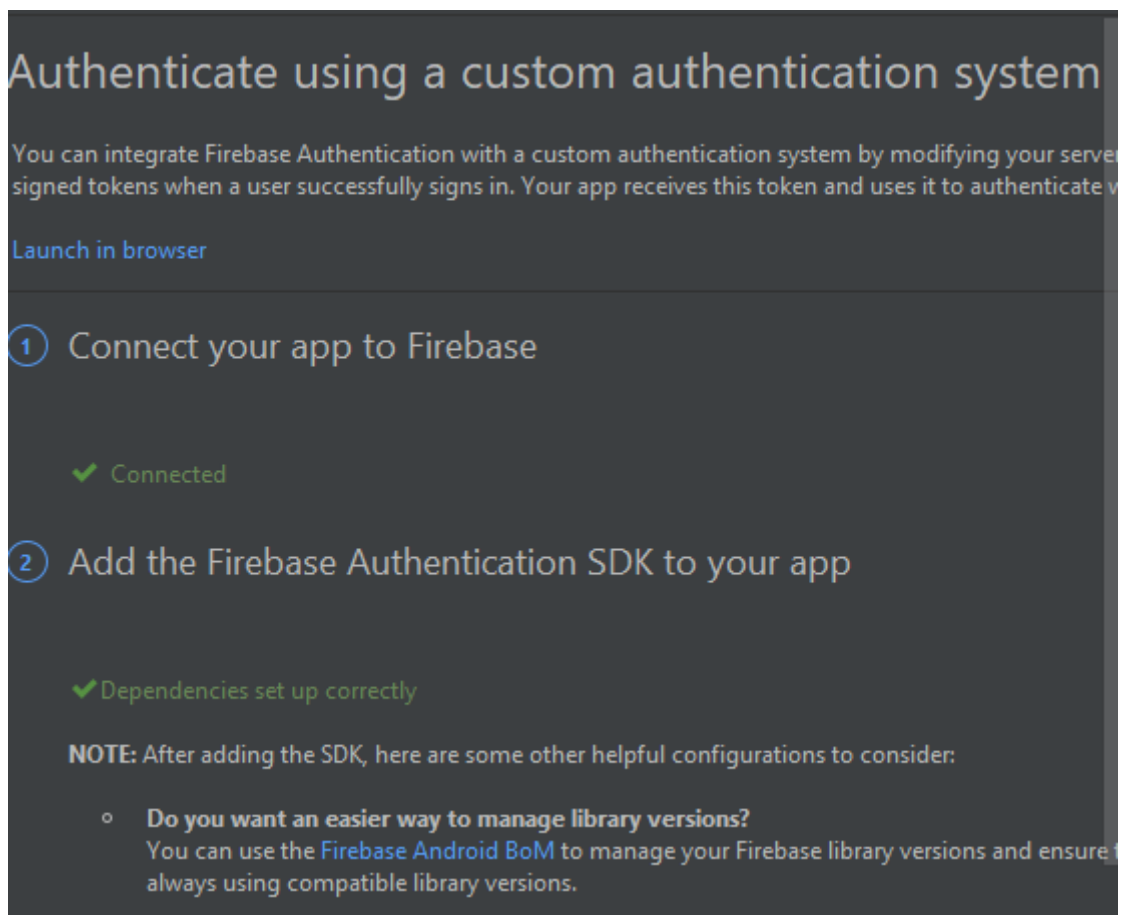



Figure 5.18 Connection between Android Studio and Firebase Authentication

API keys

You can create a new project if you don't have one already or add API keys to an existing project. All projects are subject to the [Google Cloud Platform Terms of Service](#), which you agree to when creating a new project, while use of the Gemini API and Google AI Studio is subject to the [Gemini API Terms of Service](#).

Use your API keys securely. Do not share them or embed them in code the public can view.

Starting on May 2, 2024, if you use Gemini API from a project that has billing enabled, your use will be subject to [pay-as-you-go pricing](#).

 Create API key

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.


API key	Google Cloud project name	Created	Action
...YApE	Generative Language Client <input checked="" type="checkbox"/>	Apr 16, 2024	

Figure 5.19 Gemini API key

```

dependencies {

    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.10.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation "androidx.concurrent:concurrent-futures:1.1.0"
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.6.2'

    implementation 'org.tensorflow:tensorflow-lite-support:0.4.4' //0.1.0
    implementation 'org.tensorflow:tensorflow-lite-metadata:0.4.4' //0.1.0
    implementation 'org.tensorflow:tensorflow-lite:2.9.0'
    implementation 'androidx.gridlayout:gridlayout:1.0.0'

    implementation(platform("com.google.firebase:firebase-bom:32.8.1"))
    implementation 'com.google.firebase:firebase-auth:22.3.1'
    implementation 'com.google.firebase:firebase-database:20.3.1'

    def camerax_version :String = "1.4.0-alpha02"
    implementation "androidx.camera:camera-core:${camerax_version}"
    implementation "androidx.camera:camera-camera2:${camerax_version}"
    implementation "androidx.camera:camera-lifecycle:${camerax_version}"
    implementation "androidx.camera:camera-view:${camerax_version}"

    // add the dependency for the Google AI client SDK for Android
    implementation("com.google.ai.client.generativeai:generativeai:0.3.0")

    // Required for one-shot operations (to use `ListenableFuture` from Guava Android)
    implementation("com.google.guava:guava:31.0.1-android")

    // Required for streaming operations (to use `Publisher` from Reactive Streams)
    implementation("org.reactivestreams:reactive-streams:1.0.4")

    implementation("com.squareup.okhttp3:okhttp:4.12.0")

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}

```

Figure 5.20 Android Studio Dependencies

```

<uses-permission android:name="android.permission.CAMERA" />

<uses-feature android:name="android.hardware.camera.any" />
<uses-feature android:name="android.hardware.camera.autofocus" />

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />

```

Figure 5.21 Android Studio Permissions

Before developing the mobile application, firebase authentication must be set up for authenticating account registration and logins, get a Gemini API key for generating recommendations, adding dependencies in build.gradle for Androidx Camera, TensorflowLite and Firebase and finally adding permissions in AndroidManifest.xml for Camera, Storage and Internet access.

5.10 Account and Login Feature

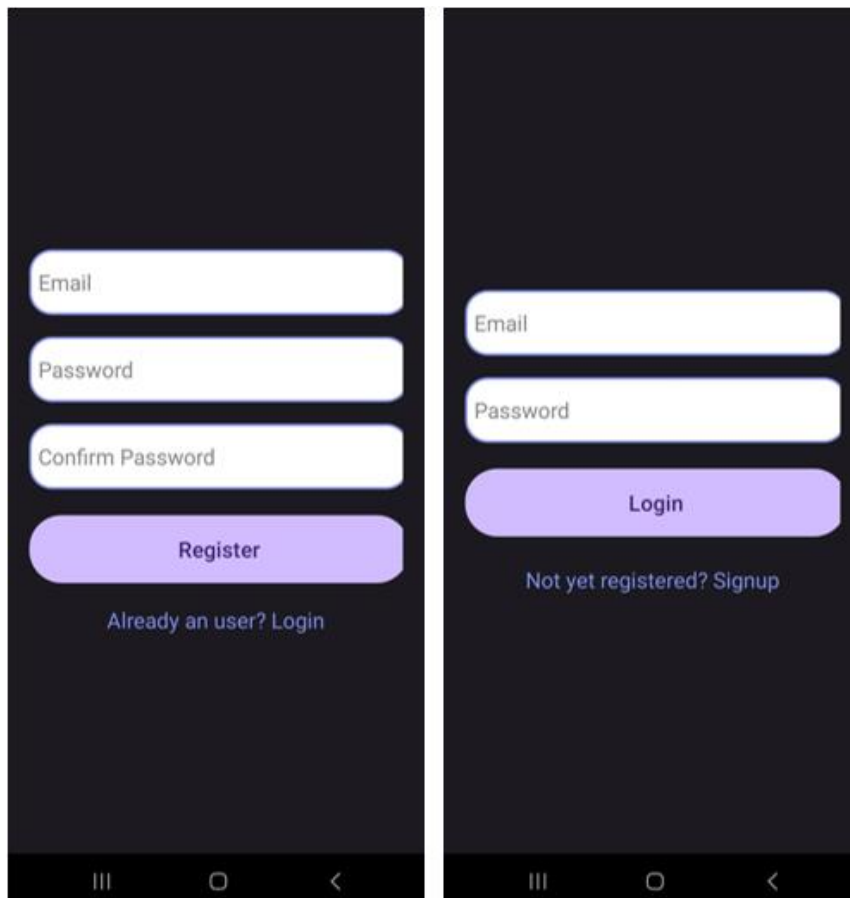


Figure 5.22 Register and Login Page

Upon launching the application, it will start with Login page first as illustrated in Figure 5.22, and the code shown above in Figure 5.21 checks whether the user is logged in. If not continue normally but if yes, it will redirect the user to the Home page.

For both Login Activity a submit button is configured, that on click, it will check whether the input email and password is empty. If yes, the system will prompt an error notification. But if it's not, FirebaseAuth will try to authenticate the email and password with the online Firebase where if successful will redirect the user to the home page and if unsuccessful will prompt an error notification.

As for the Register Activity, its quite similar where the submit button is also configured, that on click, it will check whether the input email and password is empty, password length is not over 5 characters long and the confirm password is not the same with password. If either one condition is met, the system will prompt an error notification. If neither one condition is met, then the FirebaseAuth will try to create a new user with the given email and password that upon successful, it will redirect the user to the login page to login with the newly created account, but if unsuccessful will prompt an error notification.

5.11 Profile Health Status Feature

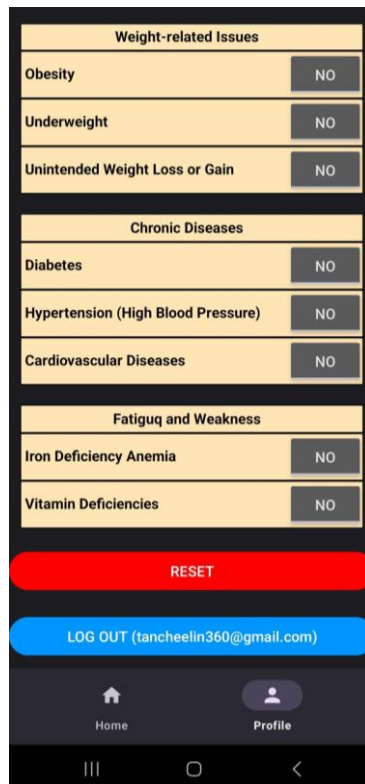


Figure 5.23 Profile Page

Upon accessing the Profile page, the system will obtain the user id and retrieve all the health status from the sharedPreferences database and apply them to the toggle buttons to display whether the status is True or False. Upon tapping on any specific toggle button to a status, it will directly update its status in the sharedPreferences database and changing the toggle button to either True or False. Moreover, for the of flexibility a reset button is also configured that on click. It will update all status in the sharedPreferences database to its default value which is False. Additionally, a logout button is also configured to allow users to either log out of their account or swap to another existing account.

5.12 Loading Model

```
//Load tflite model
try {
    tflite = new Interpreter(loadModelFile());
} catch (Exception ex) {
    ex.printStackTrace();
}
}

1 usage
private MappedByteBuffer loadModelFile() throws IOException {
    AssetFileDescriptor fileDescriptor = this.getAssets().openFd("deeplab_multiclass_resnet50.tflite");
    FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());
    FileChannel fileChannel = inputStream.getChannel();
    long startOffset = fileDescriptor.getStartOffset();
    long declaredLength = fileDescriptor.getDeclaredLength();
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
}
```

Figure 5.24 Loading Model

Upon selecting Upload Image Segmentation or Real-Time Segmentation option in the mobile application, it will first try to load the TensorFlow Lite model as a new TensorFlow Lite interpreter by calling loadModelFile() function. The function will return its model data by first trying to open the model from the assets folder to read the contents of the model file and mapping them into a memory as a MappedByteBuffer.

5.13 Segment Frame

```
public Bitmap segmentFrame(Bitmap bitmap) {
    int imageSize = 256;
    int width = bitmap.getWidth();
    int height = bitmap.getHeight();

    Bitmap resizedBitmap = Bitmap.createScaledBitmap(bitmap, imageSize, imageSize, filter: true);

    ByteBuffer imgData = ByteBuffer.allocateDirect( capacity: 4 * imageSize * imageSize * 3);
    imgData.order(ByteOrder.nativeOrder());

    ByteBuffer segmentedImage = ByteBuffer.allocateDirect( capacity: 4 * imageSize * imageSize * 15);
    segmentedImage.order(ByteOrder.nativeOrder());

    int[] intValues = new int[imageSize * imageSize];
    resizedBitmap.getPixels(intValues, offset: 0, imageSize, x: 0, y: 0, imageSize, imageSize);

    // Convert the image to floating point.
    for (int pixel = 0; pixel < intValues.length; pixel++) {
        int value = intValues[pixel];
        imgData.putFloat((float) ((value >> 16) & 0xff));
        imgData.putFloat((float) ((value >> 8) & 0xff));
        imgData.putFloat((float) (value & 0xff));
    }

    segmentedImage.rewind();
    tflite.run(imgData, segmentedImage);

    segmentedImage.rewind();
    int[] outPixels = new int[imageSize * imageSize];

    //Multiclass Mask
    HashSet<Integer> detectedClasses = new HashSet<>();
    int[] classPixelCounts = new int[15];

    for (int i = 0; i < outPixels.length; i++) {
        int maxClassIndex = 0;
        float maxClassScore = segmentedImage.getFloat();
    }
}
```

Figure 5.25 Segment Frame 1

```

for (int i = 0; i < outPixels.length; i++) {
    int maxClassIndex = 0;
    float maxClassScore = segmentedImage.getFloat();
    for (int j = 1; j < 15; j++) {
        float score = segmentedImage.getFloat();
        if (score > maxClassScore) {
            maxClassScore = score;
            maxClassIndex = j;
        }
    }

    classPixelCounts[maxClassIndex]++;

    if (maxClassIndex != 0 && !detectedClasses.contains(maxClassIndex)) {
        detectedClasses.add(maxClassIndex); // Add detected class index to set
    }

    // Assign colors based on class index
    outPixels[i] = getColorForClass(maxClassIndex);
}
int totalNonBackgroundPixels = outPixels.length - classPixelCounts[0];

// Calculate the percentage of pixels for each class
HashMap<String, Float> classPercentages = new HashMap<>();
for (int classIndex : detectedClasses) {
    String className = getClassLabel(classIndex);
    float percentage = ((float) classPixelCounts[classIndex] / totalNonBackgroundPixels) * 100;
    classPercentages.put(className, percentage);
    addClassButton(className, percentage); // Pass percentage to addClassButton
}

// Convert the binary mask to a bitmap for display
Bitmap outputBitmap = Bitmap.createBitmap(outPixels, imageSize, imageSize, Bitmap.Config.ARGB_8888);
//outputBitmap.setPixels(outPixels, 0, imageSize, 0, 0, imageSize, imageSize);
Bitmap outputBitmap1 = Bitmap.createScaledBitmap(outputBitmap, width, height, filter: true);

return outputBitmap1;

```

Figure 5.26 Segment Frame 2

As shown above in Figure 5.25 and 5.26, it shows the segmentFrame function which requires the input of a bitmap which is a format of images at the pixel level and stores information about each pixel in the image. Upon passing a bitmat format image to be segmented into the function, the application will initilize a constant integer imageSize for 256 and store the initiail width and height of the bitmap. Then, the bitmap will be rescaled to 256 for its width and height to be compatabile with the model's input format. After that, 2 ByteBuffer imgData and segmentedImage that acts as the input and output are initilized where imgData is allocated 4x256x256x3 capacity of memory and segmentedImage is allocated for 4x256x256x1 due to each element being 4 bytes long, the width and height are 256 bytes, 3 bytes for RGB images

and 1 byte for Binary images. Both ByteBuffer will then be set to the order of the buffer to the native byte order.

Moving on, the resized bitmap will execute `getPixels()` function to extract its pixels value and stores it in a newly created integer array `intValues` with the size of 256x256. Then, a loop will go through each pixel in the array to extract the red, green and blue color components and stores them as floating-point values in `imgData`. After the loop ends, `segmented Image`'s position is reset to its initial position as a precaution as it might have been changed during previous operations and then running the TensorFlow Lite model with `imgData` and `segmentedImage` which are the input and output. After running the model, `segmentedImage`'s position is again rewind and like the input data, going through a loop to store pixels value in a newly created integer array, `outPixels`.

During the loop, the system will determine the class with the highest score for each pixel and count the number of pixels for each class. Detected class indices are stored in a `HashSet` to avoid duplicates and Colors are assigned to each pixel based on the detected class index using the `getColorForClass` method. The total number of non-background pixels is also calculated to calculate the percentage of pixels for each detected class without the background. With the loop ending, the percentage of each detected class is passed to a method named `addClassButton` to display each predicted class with its percentage, that acts like a button to redirect the user to the specific class Info page. Finally, the function will also return a segmented mask which is a newly created bitmap based on the pixel values in the `outPixels` array and have been rescaled to its original size.

5.14 Upload Image Segmentation

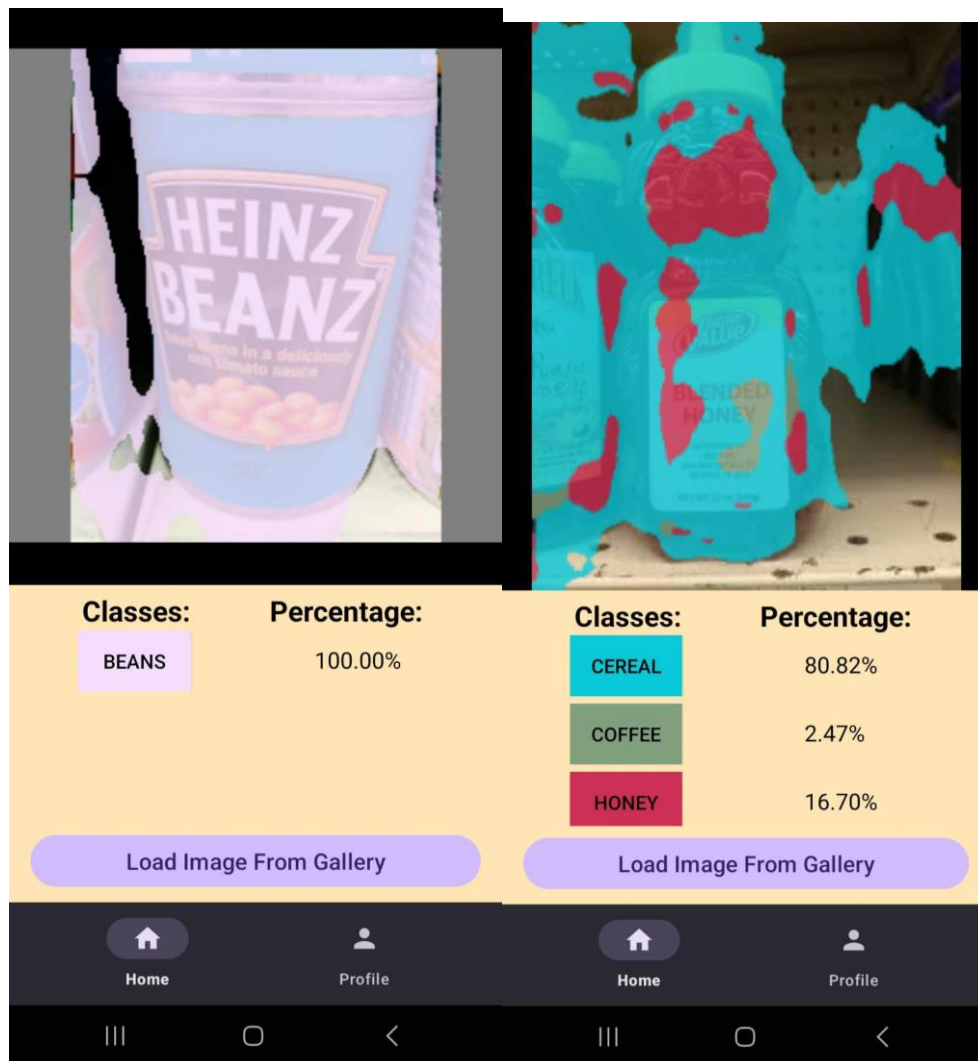


Figure 5.27 Upload Image Segmentation

Upon choosing Upload Image Segmentation option in the Home page, it will first load the model and display its layout that consist of an image view for displaying the image, mask view for displaying the mask on top of the image view and button which allows user to upload image from their phone's gallery to be segmented. After tapping on the button and uploaded an image, the image data will be converted into a bitmap and pass into the function `segmentFrame()` to be segmented. After it has finished segmented the image, the application will display the uploaded image on the image view and the segmented mask on the mask view together with all its predicted class and percentages just below the view just like above in Figure 5.27. Based on the images above in Figure 5.27, the left side of the image shows a bean object successfully predicted it 100% as a bean and the right side of the image shows a honey object predicted as cereal, coffee and honey with cereal having the highest percentage. It is mainly because the cereal and coffee images used to train the model contain similar features with honey due to

limited data, causing the model to confuse on the correct grocery. Further results for test images segmentation are shown in Appendix 1.

5.15 Real Time Segmentation

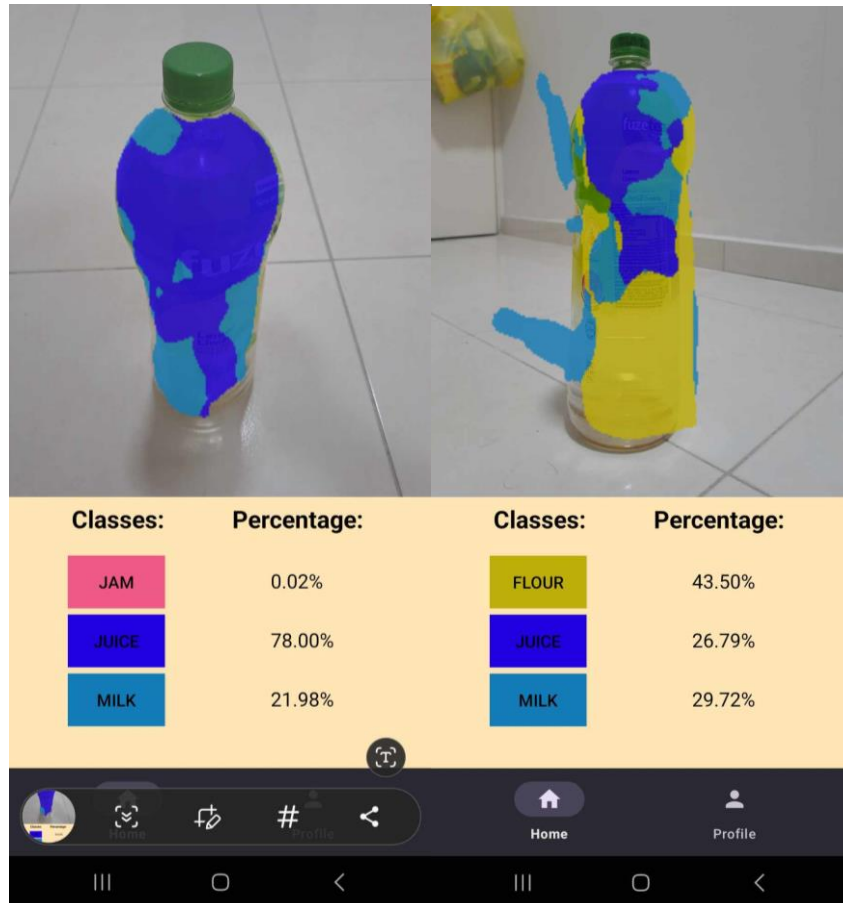


Figure 5.28 Real Time Segmentation

As for choosing the Real-Time Segmentation option, like in Image Segmentation, it will first load the model and then check if the permission for camera has already been granted. If not, send out a pop out to request for the permission for the camera. Upon having the permission, the application will immediately utilize the back lens of the camera to capture every frame to be segmented. The application will automatically analyze and retrieve the frame as a bitmap to be passed into the function `segmentFrame()` and returned as a mask. Its displayed layout would also contain image view for displaying the frames captured and mask view for

displaying the segmented mask on top of the the image view together with all its predicted class and percentages just below the view. The process will keep looping, capturing frames, segmenting them and predicting the classes in real time until the user return to the home page or close the application. An example of a frame captured in real time is shown above in Figure 5.28. Based on the images above in Figure 5.27, the left side of the image shows a juice object predicted as jam, juice and milk with juice as the highest percentage. As for the right side of the image, it shows the same juice object from a different angle but predicted as flour, juice and milk with flour having the highest percentage. This is caused by the frequent change of different angle, lighting and background during real time segmentation, causing the model to be confused and predict wrongly. Subsequently, Appendix 1 illustrates other results for the real time segmentation.

5.16 Gemini API

```
public void getResponse(String query, ResponseCallback callback) {
    GenerativeModelFutures model = getModel();

    Content content = new Content.Builder().addText(query).build();
    Executor executor = Runnable::run;

    ListenableFuture<GenerateContentResponse> response = model.generateContent(content);
    Futures.addCallback(response, new FutureCallback<GenerateContentResponse>() {
        @Override
        public void onSuccess(GenerateContentResponse result) {
            String resultText = result.getText();
            callback.onResponse(resultText);
        }

        @Override
        public void onFailure(Throwable t) {
            t.printStackTrace();
            callback.onError(t);
        }
    }, executor);
}
```

Figure 5.29 getResponse

```

private GenerativeModelFutures getModel() {
    String apiKey = BuildConfig.apiKey;

    SafetySetting harassmentSafety = new SafetySetting(HarmCategory.HARASSMENT, BlockThreshold.ONLY_HIGH);

    GenerationConfig.Builder configBuilder = new GenerationConfig.Builder();
    configBuilder.temperature = 0.9f;
    configBuilder.topK = 16;
    configBuilder.topP = 0.1f;
    GenerationConfig generationConfig = configBuilder.build();

    GenerativeModel gm = new GenerativeModel(
        modelName: "gemini-pro",
        apiKey,
        generationConfig,
        Collections.singletonList(harassmentSafety)
    );

    return GenerativeModelFutures.from(gm);
};

```

Figure 5.30 getModel

In order for the mobile application to call for Gemini API, a java class is created with `getResponse` and `getModel` method. For `getResponse` method, upon receiving a user query and a callback object, it fetches a `GenerativeModelFutures` instance using the `getModel()` method. The user query is then encapsulated in a `Content` object and passed to the generative model. Callbacks are registered for execution upon the future's completion; `onSuccess` is triggered when the generation process is successful, and `onFailure` is called in case of any error.

As for `getModel` method, it retrieves the API key from `BuildConfig.apiKey` and sets up a `SafetySetting` to ensure the generated content complies with certain standards, specifying that only content with a high level of safety regarding harassment is accepted. A `GenerationConfig` is created to define the configuration parameters for the generation process, including settings like `temperature`, `topK`, and `topP`. Subsequently, a `GenerativeModel` object is created, specifying the model's name "gemini-pro", the API key, the generation configuration, and a list containing the safety setting. Finally, a `GenerativeModelFutures` object is created from the `GenerativeModel` and returned.

5.17 Grocery Class Info Page

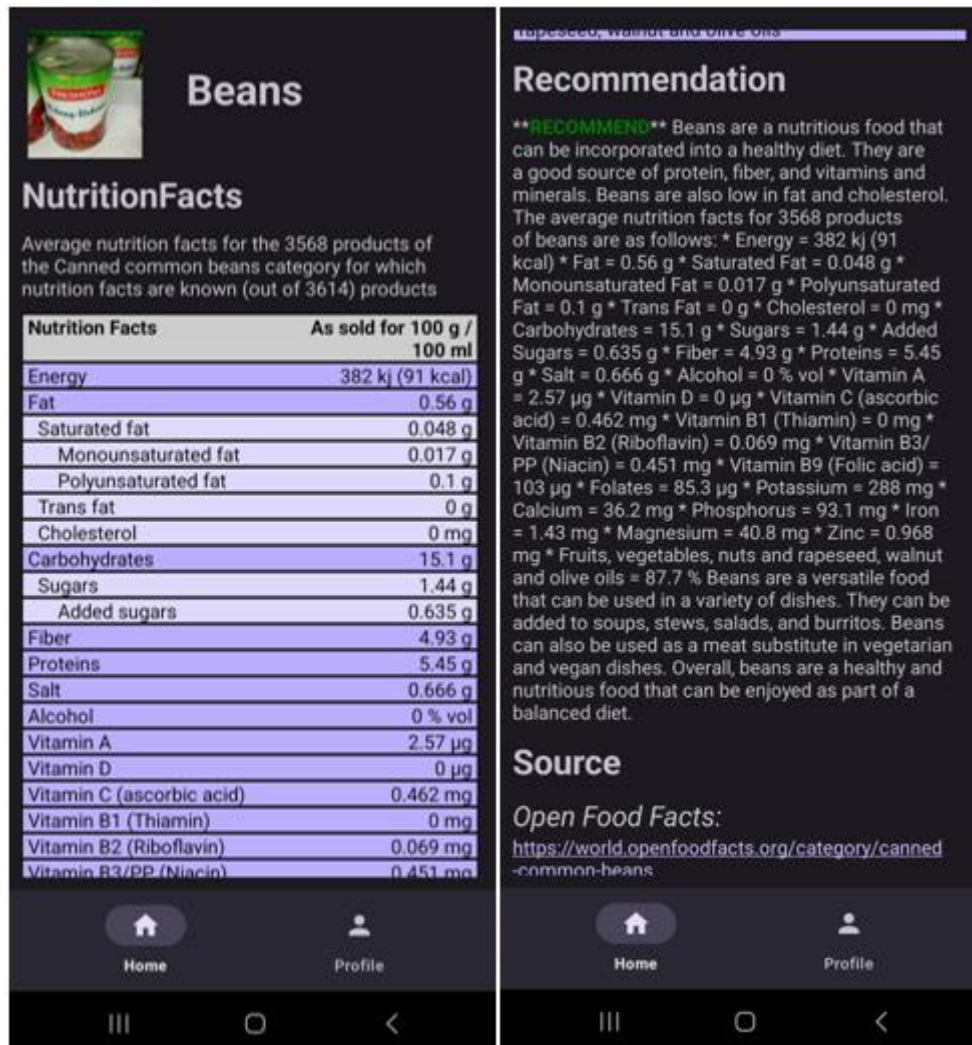


Figure 5.31 Grocery Class Info Page

During both Upload Image Segmentation and Real Time Segmentation, all the predicted classes displayed can be tapped and on click, will redirect the user to the class's Info page which then automatically displays all the nutrition facts about the class and generates a recommendation of the grocery to the user. The recommendation is generated by calling the Gemini API, by combining the nutrient facts of the grocery class and the health status of the user into a query to be sent to the API, to ensure that the recommendation aligns with the user's health needs. It may load a while and upon receiving the response from the API, the system will automatically display it for the user to read.

Chapter 6

System Evaluation and Discussion

6.1 Comparison between trained backbones

6.1.1 Backbones

- VGG16 [37]

VGG16 is a milestone in convolutional neural network (CNN) architectures and renowned for its simplicity and effectiveness. Comprising 13 convolutional layers and 3 fully connected layers, VGG16 demonstrates strong feature extraction capabilities. Its uniform structure and small kernel sizes facilitate learning rich hierarchical features, making it suitable for various vision tasks. Despite its depth, VGG16 is prone to overfitting due to a large number of parameters.

- VGG19 [37]

Building upon the success of VGG16, VGG19 extends the architecture by adding more convolutional layers, resulting in deeper feature hierarchies. With 16 convolutional layers and 3 fully connected layers, VGG19 achieves superior performance in capturing intricate patterns within images. However, the increased depth comes at the cost of higher computational complexity and resource requirements.

- ResNet50 [38]

ResNet50 introduces the groundbreaking concept of residual learning, enabling the training of extremely deep neural networks. By utilizing residual blocks, ResNet50 mitigates the vanishing gradient problem, allowing for seamless training of networks with over 50 layers. This architecture revolutionized deep learning by enabling the construction of deeper models with improved performance and faster convergence. ResNet50 strikes a balance between depth and computational efficiency, making it a popular choice for various applications.

- ResNet101 [38]

ResNet101 further extends the ResNet architecture with 101 layers, enabling the extraction of more abstract and intricate features from images. Despite its increased depth, ResNet101 maintains efficient training by leveraging residual connections. With enhanced

representational power, ResNet101 surpasses its predecessors in capturing fine-grained details and nuances in images. However, the additional layers increase computational demands, requiring ample resources for training and inference.

6.1.2 Models Performance Definition

For the groceries' segmentation, the accuracy of the segmented food products is evaluated to indicate the performance of the proposed system. In this study, DeepLabV3+ [31], a deep learning model that is utilized in the proposed system aims to achieve higher accuracy and loss values for the groceries' segmentation within the validation, This aims to prevent overfitting, a scenario where the model excels excessively within the training set, but significantly falter in performance when presented with the validation set which consist of images that the model has not encountered before.

The deep learning model is evaluated based on its training and validation loss using Sparse Categorical Cross Entropy, a loss function commonly used in machine learning for multi-class classification tasks, particularly for dealing with target labels that are provided as integers rather than one-hot encoded vector. It provides a clear measure of the difference between the predicted and true probability distributions of the classes, outputting a scalar value for model evaluation. It is also applicable to imbalanced datasets, effectively penalizing misclassification of minority classes, and compatible with the SoftMax activation function [40].

In order to test the accuracy of the groceries' segmentation, the evaluation metrics used is Accuracy as referred in Chapter 4 Section.

Other than the evaluation metrics mentioned above, the hyperparameters used in the deep learning model such as Adam Optimizer with 0.001 and 0.01 learning rate, 16, 8 and 4 Batch Size, 1000 Epochs and different data augmentation techniques, played an important role in building a good model to achieve higher accuracy for segmentation tasks.

6.1.3 Trained Models Result

Table 6.1 Trained Models Result

DeepLabV3+ Backbones	Training Accuracy	Validation Accuracy	Testing Accuracy
VGG16	0.6282	0.4536	0.5567
VGG19	0.5862	0.4724	0.4804
ResNet50	0.8755	0.6684	0.8270
ResNet101	0.8860	0.6768	0.8127

When comparing VGG16 and VGG19, it's evident that VGG16 performs marginally better in terms of training and testing accuracy. VGG16 achieves a training accuracy of 62.82% compared to 58.62% of VGG19. However, in the case of validation and testing accuracy, VGG19 slightly outperforms VGG16, with validation accuracy of 47.24% compared to 45.36% and testing accuracy of 48.04% compared to 55.67% for VGG16. This indicates that while VGG16 learns the training data better, VGG19 generalizes slightly better to unseen data.

In the comparison between ResNet50 and ResNet101, ResNet101 consistently outperforms ResNet50 across all metrics. ResNet101 achieves a training accuracy of 88.60%, significantly higher than the 87.55% attained by ResNet50. Similarly, ResNet101 yields a validation accuracy of 67.68% in contrast to ResNet50's 66.84%. In testing accuracy, ResNet101 maintains superiority, with 81.27% compared to 82.70% for ResNet50. This suggests that ResNet101 is more capable of learning complex features and generalizing better to unseen data compared to ResNet50.

Comparing VGG16/19 with ResNet50/101, it's evident that the ResNet models outperform the VGG models across all metrics. Both ResNet50 and ResNet101 achieve significantly higher training, validation, and testing accuracies compared to VGG16 and VGG19. This indicates that the deeper architecture of ResNet models allows for better feature extraction and learning, resulting in better performance. For instance, ResNet101 outperforms VGG16 by around 25% in terms of testing accuracy.

6.2 Project Challenges and Implementation Issues

During the implementation of the proposed system, one of the major challenges encountered during this study is the model training. This process demands meticulous attention to various components, including data quality, model architecture building, hyperparameter tuning, and computational resources. Achieving the desired accuracy in food image segmentation heavily relies on the quality and diversity of the dataset used for training. Additionally, the development of an efficient and effective model architecture is crucial to ensure accurate real-time processing of food images. Furthermore, fine-tuning hyperparameters to optimize the model's performance adds another layer of complexity to the training process. Lastly, considering the computational constraints of mobile platforms, obtaining the necessary computational resources to run the model effectively becomes a significant challenge.

For implementation issues, as mentioned before data quality is an important component for model training but due to the absence of available grocery dataset containing multiple classes with both images and ground truth for multiclass segmentation tasks, most of the time is dedicated to creation and labeling of a custom dataset that consist of multiple grocery classes that contains with images and their respective ground truths. Despite the significant effort, the current custom dataset still suffers from insufficiency since it only contains 700 total images with each class only consisting of 50 images, leaving the model with less data to train with. Due to the similarity in features among the images of different classes, the segmentation process has become notably challenging for the model, making it difficult to predict the correct pixel values or classes for the image.

As for computational resource, there are no issues encountered during model training, as it took me less than an hour to finish training a model using Jupyter Notebook that utilizes the resources of the local machine. However, after implementing the current model into mobile application especially during real-time segmentation, there exist lags and delays in rendering results due to the high demands of processing power and limited resources.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In conclusion, this study thus far has implemented and utilized a trained DeepLapV3+ model in a grocery segmentation mobile application for upload image segmentation and real-time segmentation. The accuracy for the model is above average, demonstrating its capability to effectively segment images and predict the class accurately for most cases, ensuring a reliable foundation for real-time segmentation within the mobile application. Although the real-time segmentation works fine, it's noticeably laggy due to limited hardware resources.

Aside from segmentation, the application incorporates essential features to enrich user interaction. A login and register feature have been implemented to ensure security and facilitate a personalized experience. Moreover, an information page for each grocery class has been integrated to display nutrient details and provide recommendations for the user. To ensure personalized recommendations, a profile page has been included, enabling users to update their health status, which in turn will be utilized for more tailored recommendations.

This comprehensive approach not only enables users to segment grocery images but also provides a platform for a more informed, health-conscious shopping experience. With further optimization, particularly in data quality and resource allocation, the application stands to become an indispensable tool for health-conscious consumers.

7.2 Recommendation

There are several recommendations that could be implemented to enhance the functionality and user experience of this mobile application project. First and foremost, to mitigate or reduce the segmentation lag experienced during real-time grocery segmentation, Edge computing solution or cloud services can be applied to offload the processing burden from the mobile device. This will enable real-time segmentation without significant lag, as the heavy computation will be performed on more powerful servers. For instance, by deploying the segmentation model on Google Cloud IoT Edge or Google Cloud Platform (GCP) the heavy computational tasks associated with the segmentation process are offloaded to nearby edge servers and cloud servers, ensuring real-time segmentation with minimal latency.

Moreover, the number of predicted grocery classes can be expanded to allow the model to recognize a larger variety of grocery classes greatly enhancing the application's utility and user experience. More diverse images of grocery classes can be collected and experiment with more advanced deep learning architectures to improve its ability to recognize a broader range of grocery classes accurately. The diversity of the training data can also be increased by employing more data augmentation techniques. A user feedback feature integration may also allow the application to collect data on misclassified groceries and utilizing this feedback to continuously improve the model's accuracy through iterative updates.

Additionally, the Profile page for personalized recommendations can also be improved by adding more health status parameters, following with the integration of health description to allow users to input more detailed description of their health status, including any medical conditions, dietary requirements, or fitness goals which will facilitate more precise and tailored recommendations. The integration of wearable devices would also ensure that the recommendations provided by the application are always up-to-date and accurate by automatically updating the user's health status base on the wearable health monitoring devices.

REFERENCES

- [1] “New study finds poor diet kills more people globally than tobacco and high blood pressure | The Institute for Health Metrics and Evaluation,” www.healthdata.org.
<https://www.healthdata.org/news-events/newsroom/news-releases/new-study-finds-poor-diet-kills-more-people-globally-tobacco-and>
- [2] “The trouble with nutrition research: What should we eat?,” www.medicalnewstoday.com, Feb. 24, 2020. <https://www.medicalnewstoday.com/articles/why-is-nutrition-so-hard-to-study#The-role-of-industry>
- [3] E. J. Otto, “3 Things Some Nutrition Apps Get Wrong - Food & Nutrition Magazine,” foodandnutrition.org, Dec. 29, 2015. <https://foodandnutrition.org/blogs/student-scoop/3-things-nutrition-apps-get-wrong/>
- [4] “The latest advancements in computer vision and image recognition,” www.linkedin.com.
<https://www.linkedin.com/pulse/latest-advancements-computer-vision-image-recognition-deepak-solanki#:~:text=With%20the%20help%20of%20deep>
- [5] admin, “Image Segmentation: The Most Interesting Applications,” DeepLobe, Mar. 08, 2023. <https://deeplobe.ai/image-segmentation-the-most-interesting-applications/>
- [6] “MyFitnessPal: Calorie Counter – Applications on Google Play,” play.google.com.
<https://play.google.com/store/applications/details?id=com.myfitnesspal.android&hl=en-MY>
- [7] “Open Food Facts – Applications on Google Play,” play.google.com.
<https://play.google.com/store/applications/details?id=org.openfoodfacts.scanner&hl=en-MY>
- [8] “Foodvisor - Nutrition & Diet – Applications on Google Play,” play.google.com.
<https://play.google.com/store/applications/details?id=io.foodvisor.foodvisor&hl=en-MY>
- [9] My Fitness Pal, “MyFitnessPal | MyFitnessPal.com,” Myfitnesspal.com, 2019.
<https://www.myfitnesspal.com>

- [10] Meal scan FAQ – myfitnesspal help. Available at: <https://support.myfitnesspal.com/hc/en-us/articles/360045761612-Meal-Scan-FAQ>.
- [11] Passio AI - enhancing human life with ai (no date) Passio AI - Enhancing Human Life with AI. Available at: <https://www.passio.ai/>
- [12] What do food colors and emojis mean - foodvisor.zendesk.com, <https://foodvisor.zendesk.com/hc/en-us/articles/360013181779-What-do-food-colors-and-emojis-mean->
- [13] Romain Dillet, “Foodvisor automatically tracks what you eat using deep learning,” TechCrunch, Oct. 14, 2019. <https://techcrunch.com/2019/10/14/foodvisor-automatically-tracks-what-you-eat-using-deep-learning/>
- [14] “Food Check: Product Scanner – Applications on Google Play,” play.google.com. https://play.google.com/store/applications/details?id=com.bytes_and_pixels.food_buddy&hl=en-MY
- [15] Guide to Image Segmentation in Computer Vision: Best Practices,” encord.com. <https://encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide/>
- [16] “Thresholding-Based Image Segmentation,” GeeksforGeeks, Jan. 11, 2023. <https://www.geeksforgeeks.org/thresholding-based-image-segmentation/>
- [17] N. Senthilkumaran and S. Vaithegi, "Image segmentation by using thresholding techniques for medical images," Computer Science & Engineering: An International Journal, vol. 6, no. 1, pp. 1-13, February, 2016.
- [18] Kaur, Dilpreet, Y. Kaur "Various image segmentation techniques: a review." International Journal of Computer Science and Mobile Computing, vol.3, no.5, pp. 809-814, May. 5, 2014.
- [19] “Region and Edge Based Segmentation,” GeeksforGeeks, Jul. 18, 2021. <https://www.geeksforgeeks.org/region-and-edge-based-segmentaion/>

- [20] N. O'Mahony et al., "Deep Learning vs. Traditional Computer Vision," *Advances in Intelligent Systems and Computing*, vol. 943, pp. 128–144, Apr. 2019, doi: https://doi.org/10.1007/978-3-030-17795-9_10.
- [21] What are the pros and cons of using deep learning vs. traditional ML methods?," [www.linkedin.com. https://www.linkedin.com/advice/0/what-pros-cons-using-deep-learning-vs-traditional#:~:text=Deep%20learning%20offers%20several%20advantages](https://www.linkedin.com/advice/0/what-pros-cons-using-deep-learning-vs-traditional#:~:text=Deep%20learning%20offers%20several%20advantages).
- [22] D. Ciccarelli, "When to Use Agile Project Management: The Definitive Guide," *GoSkills.com*. <https://www.goskills.com/Project-Management/Resources/When-to-use-agile-project-management>
- [23] "CS 410/510 - Software Engineering class notes," *Ccsu.edu*, 2019. <https://cs.ccsu.edu/~stan/classes/CS410/Notes16/02-SoftwareProcesses.html>
- [24] "Project Jupyter," *Jupyter.org*, 2019. <https://jupyter.org>
- [25] Python Software Foundation, "Welcome to Python.org," *Python.org*, Nov. 18, 2019. <https://www.python.org>
- [26] TensorFlow, "TensorFlow," *TensorFlow*, 2019. <https://www.tensorflow.org>
- [27] "Keras | TensorFlow Core | TensorFlow," *TensorFlow*, 2019. <https://www.tensorflow.org/guide/keras>
- [28] "Download Android Studio & Application Tools," *Android Developers*. https://developer.android.com/studio?gclid=Cj0KCQjw0vWnBhC6ARIsAJpJM6fHTNPKNu5-atRoOdIxbCUmFBuanMYmJBGw8x4vu-58tOwr65iHpFQaArMfEALw_wcB&gclidsrc=aw.ds.
- [29] Google, "Firebase," *Firebase*, 2019. <https://firebase.google.com>
- [30] "CVAT," *www.cvat.ai*. <https://www.cvat.ai>
- [31] *Researchgate.net*. [Online]. Available: https://www.researchgate.net/publication/354108791_Estimation_of_Road_Boundary_

for_Intelligent_Vehicles_Based_on_DeepLabV3_Architecture#pf7. [Accessed: 23-Apr-2024].

[32] “Andreas Eitel - Autonomous Intelligent Systems,” aisdatasets.informatik.uni-freiburg.de.
http://aisdatasets.informatik.uni-freiburg.de/freiburg_groceries_dataset/

[33] P. Jund, N. Abdo, A. Eitel, and W. Burgard, “The Freiburg Groceries Dataset,” arXiv:1611.05799 [cs], Nov. 2016, Available: <https://arxiv.org/abs/1611.05799>

[34] Author links open overlay panel Bhakti Baheti et al. (2020) Semantic scene segmentation in unstructured environment with modified deeplabv3+, Pattern Recognition Letters. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S0167865520302750> (Accessed: 24 April 2024).

[35] E. Zvornicanin and E. Zvornicanin, “What does backbone mean in neural networks? | Baeldung on Computer Science,” Baeldung on Computer Science, Mar. 18, 2024. <https://www.baeldung.com/cs/neural-network-backbone#:~:text=Generally%2C%20the%20term%20backbone%20refers,into%20a%20certain%20feature%20representation.>

[36] E. D, “Accuracy, Recall & Precision,” Medium, Dec. 08, 2019. <https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d>

[37] G. Boesch, “VGG Very Deep Convolutional Networks (VGGNet) - What you need to know,” viso.ai, Oct. 06, 2021. <https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/#:~:text=The%20concept%20of%20the%20VGG19>

[39] A. Gupta, “A Comprehensive Guide on Optimizers in Deep Learning,” Analytics Vidhya, Oct. 07, 2021. <https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#:~:text=Adam%20optimizer%2C%20short%20for%20Adaptive>

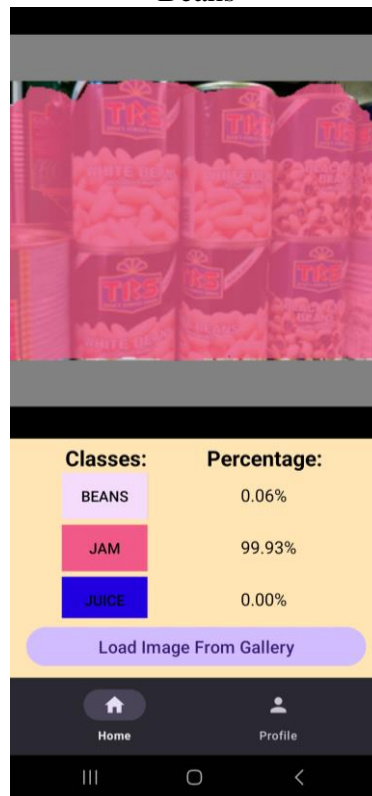
[38] T. SHARMA, “Detailed Explanation of Residual Network(Resnet50) CNN Model.,” Medium, Mar. 06, 2023. <https://medium.com/@sharma.tanish096/detailed-explanation-of-residual-network-resnet50-cnn-model-106e0ab9fa9e>

[40] M. Rahman, “What You Need to Know about Sparse Categorical Cross Entropy,” Medium, Mar. 07, 2023. <https://rmoklesur.medium.com/what-you-need-to-know-about-sparse-categorical-cross-entropy-9f07497e3a6f#:~:text=Sparse%20Categorical%20Cross%20Entropy%20is>

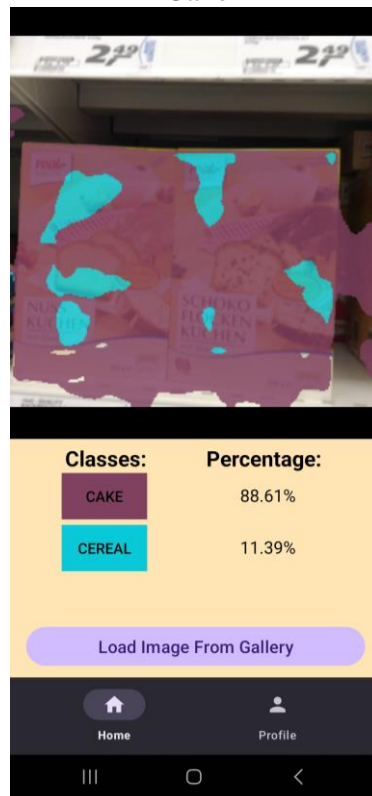
APPENDIX

Upload Image Segmentation Results

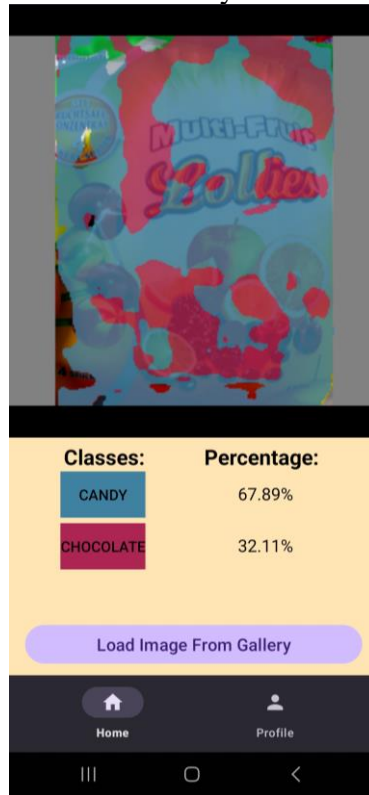
Beans



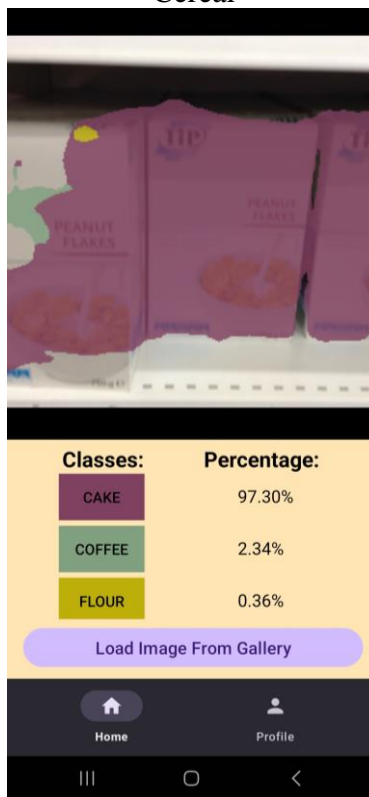
Cake



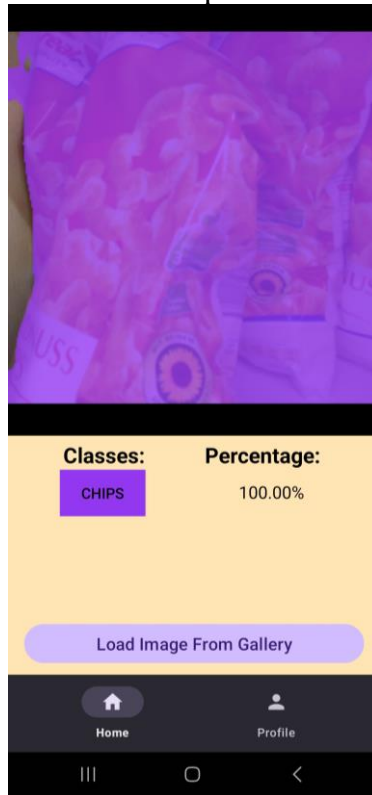
Candy



Cereal



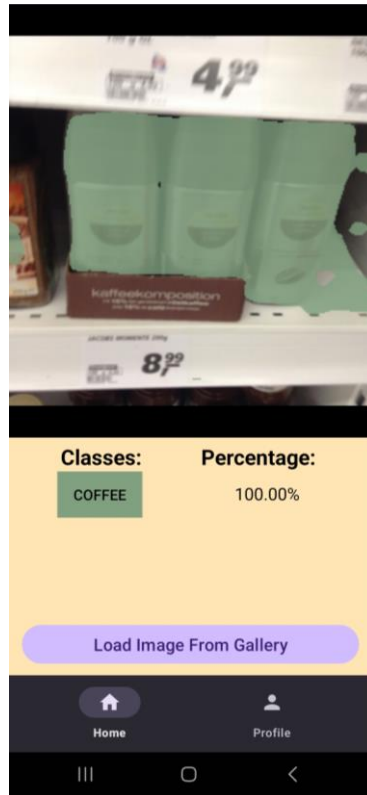
Chips



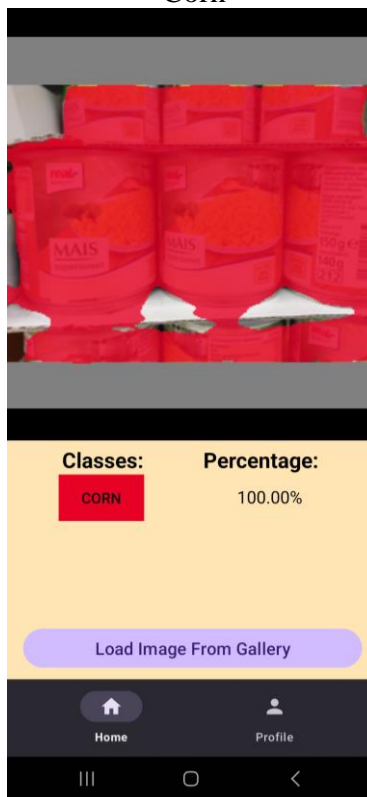
Chocolate



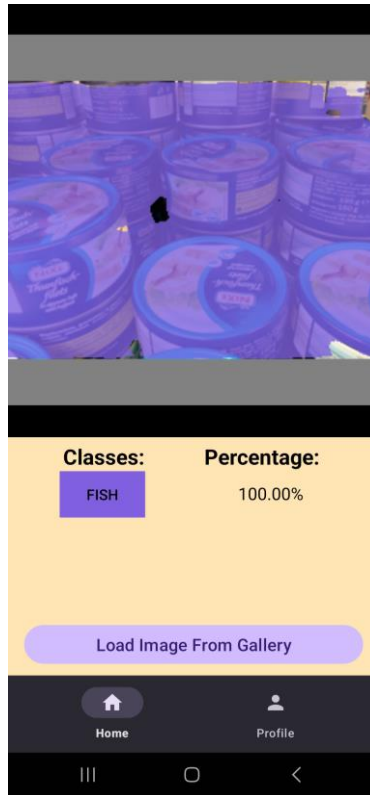
Coffee



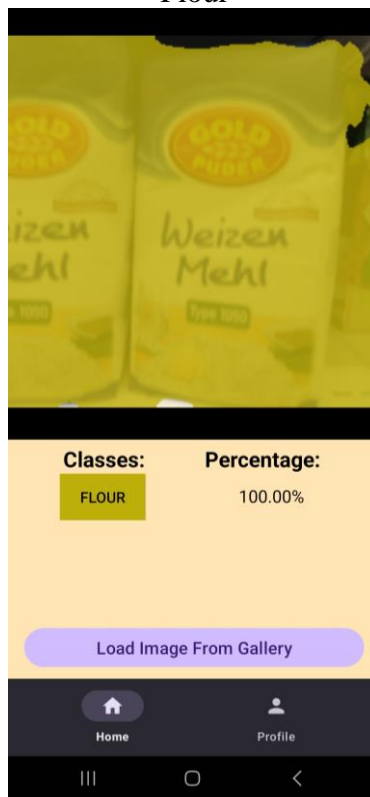
Corn



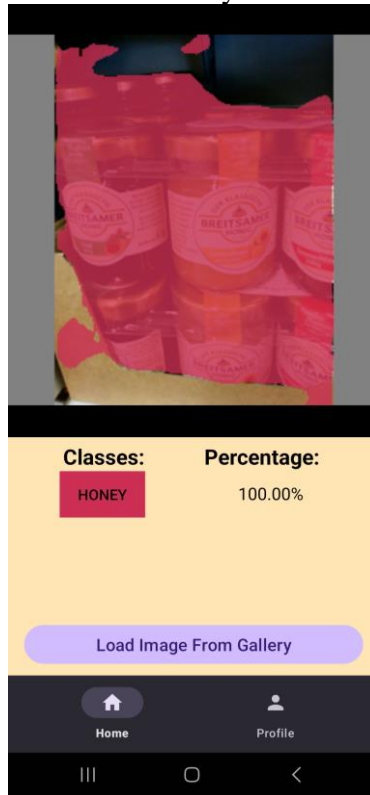
Fish



Flour



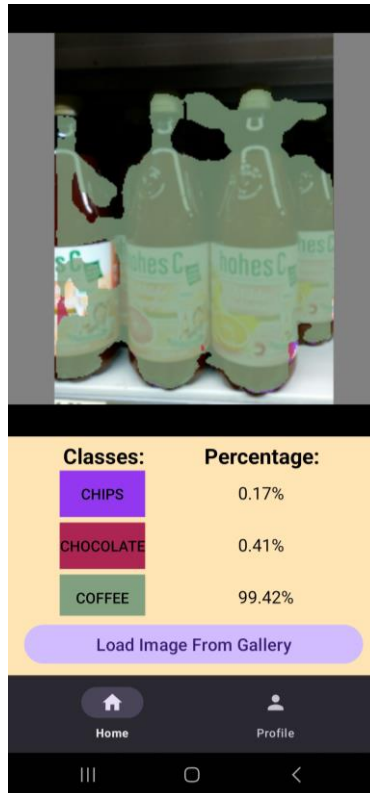
Honey



Jam



Juice

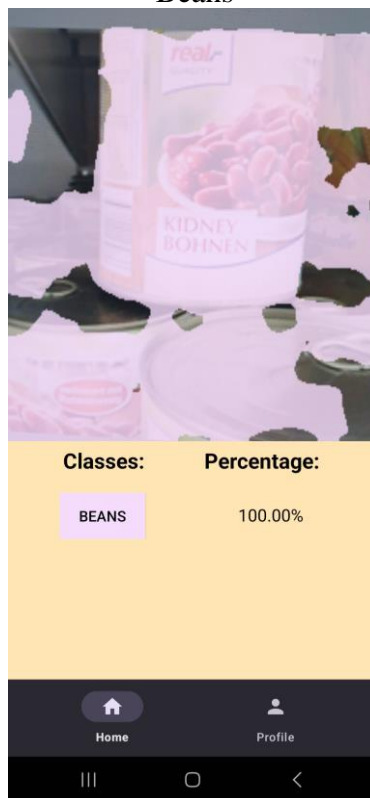


Milk

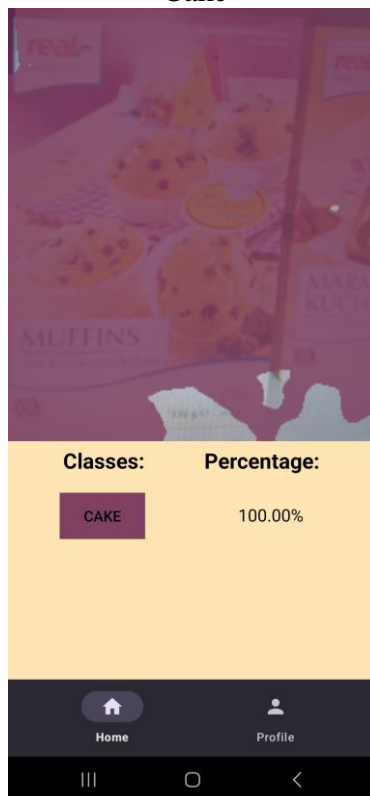


Real Time Segmentation Results

Beans



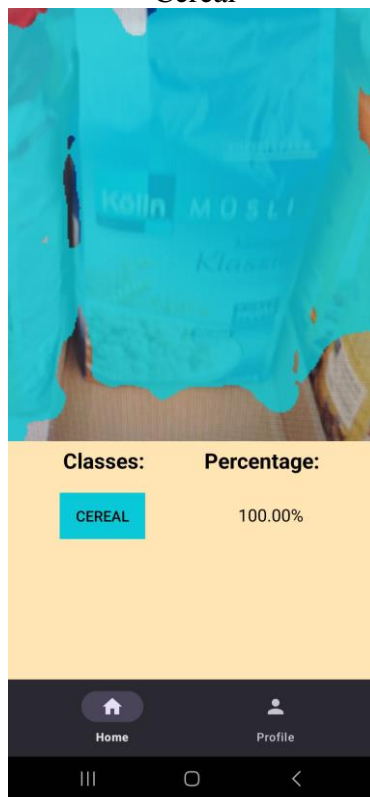
Cake



Candy



Cereal



Chips



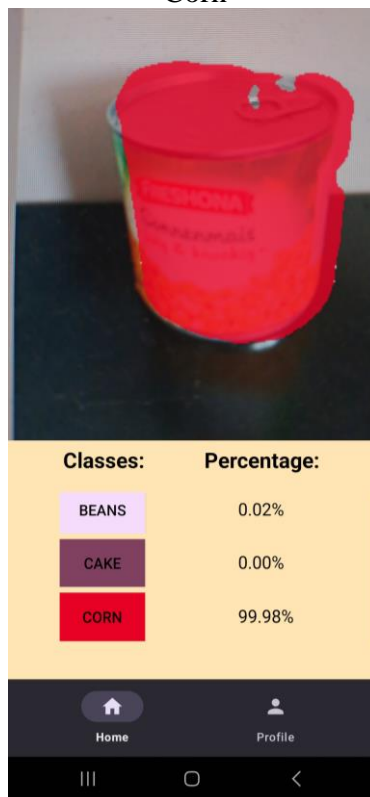
Chocolate



Coffee



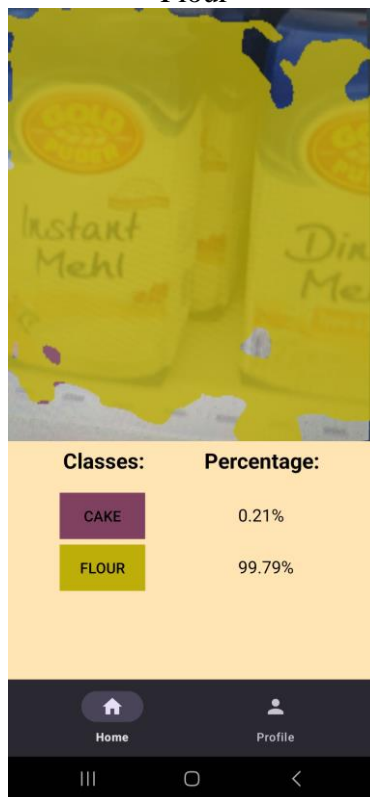
Corn



Fish



Flour



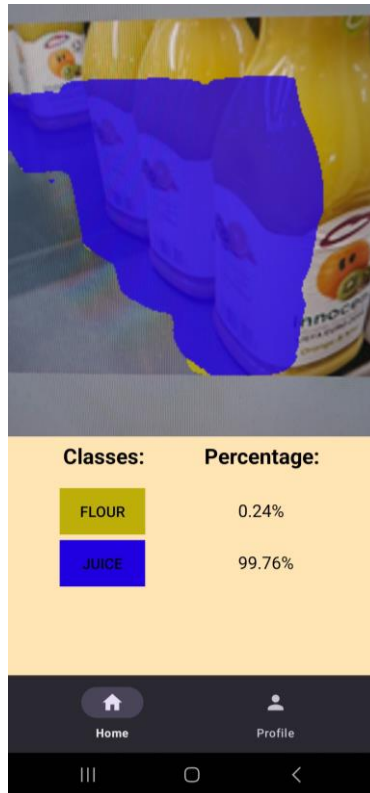
Honey



Jam



Juice



Milk



WEEKLY LOG

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:2
Student Name & ID: TAN CHEE LIN 20ACB01595	
Supervisor: Dr. Mogana Vadivelo	
Project Title: Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store	

1. WORK DONE

- Understand and learn about multiclass segmentation with DeepLabV3+ model

2. WORK TO BE DONE

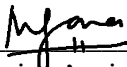
- Preprocessing and manually labelling data to create a custom dataset for multiclass segmentation.

3. PROBLEMS ENCOUNTERED

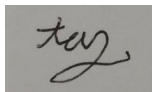
- The process of labeling the data is time consuming.

4. SELF EVALUATION OF THE PROGRESS

- Learned how to label data and create own custom dataset using CVAT.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:4
Student Name & ID: TAN CHEE LIN 20ACB01595	
Supervisor: Dr. Mogana Vadivelo	
Project Title: Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store	

1. WORK DONE

- 300+ images done labeling for the custom dataset.

2. WORK TO BE DONE

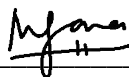
- Rest of the dataset to be labeled.
- DeepLab model development

3. PROBLEMS ENCOUNTERED

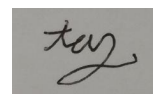
- The process of labeling the data is time consuming.

4. SELF EVALUATION OF THE PROGRESS

-



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:6
Student Name & ID: TAN CHEE LIN 20ACB01595	
Supervisor: Dr. Mogana Vadivelo	
Project Title: Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store	

1. WORK DONE

- 700 images finished labeled and completed the custom dataset.

2. WORK TO BE DONE

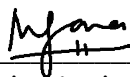
- DeepLab model development
- Model training and evaluation.

3. PROBLEMS ENCOUNTERED

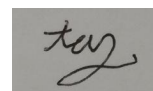
- Hard time understand DeepLab Model
- Model accuracy is very low after training

4. SELF EVALUATION OF THE PROGRESS

-



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:8
Student Name & ID: TAN CHEE LIN 20ACB01595	
Supervisor: Dr. Mogana Vadivelo	
Project Title: Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store	

1. WORK DONE

- DeepLab model architecture is developed
- Model train and evaluation.

2. WORK TO BE DONE

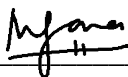
- Optimization of the model
- Integration of the app module with a multiclass model
- Implement register and login feature

3. PROBLEMS ENCOUNTERED

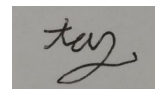
-

4. SELF EVALUATION OF THE PROGRESS

-



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:10
Student Name & ID: TAN CHEE LIN 20ACB01595	
Supervisor: Dr. Mogana Vadivelo	
Project Title: Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store	

1. WORK DONE

- Optimization of the model
- Integration of the app module with a multiclass model
- Implemented register and login feature.

2. WORK TO BE DONE

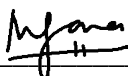
- Implement grocery info feature.
- Implement recommendation feature.

3. PROBLEMS ENCOUNTERED

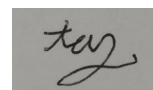
- Real time segmentation is quite laggy and slow.

4. SELF EVALUATION OF THE PROGRESS

- Learned how to use Firebase.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:12
Student Name & ID: TAN CHEE LIN 20ACB01595	
Supervisor: Dr. Mogana Vadivelo	
Project Title: Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store	

1. WORK DONE

- Implement grocery info feature.
- Implement recommendation feature.

2. WORK TO BE DONE

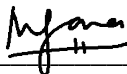
- Conclude all works and start writing on the report.

3. PROBLEMS ENCOUNTERED

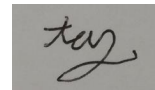
-

4. SELF EVALUATION OF THE PROGRESS

- Learned how to call Gemini API




Supervisor's signature



Student's signature


POSTER



UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Project Supervisor: Dr. Mogana Vadivelon
Project Developer : Tan Chee Lin



MOBILE APPLICATION FOR REAL-TIME FOOD IMAGE SEGMENTATION AND NUTRITIONAL GUIDANCE AT GROCERY STORE

OBJECTIVES

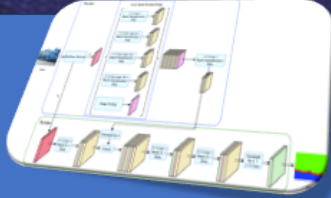
- To segment food products from an image or in real time using deep learning models
- To provide nutritional information about the food products segmented in the camera.
- To provide recommendations of the food products to the user.

INTRODUCTION

This application allows users to **segment food products** images either by uploading or capture in real time

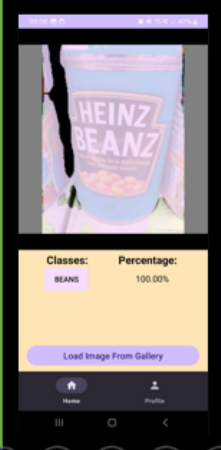
By segmenting food products, they could be selected to show their **nutritional information** and provide **recommendations**

DeepLabV3+ Model Architecture




- Achieved an average accuracy of 87.55% for training, 66.84% for validation and 82.70% for testing
- Converted into a TensorFlow Lite model to be integrated into mobile application

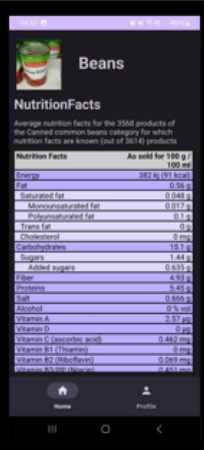
IMAGE SEGMENTATION



REAL-TIME SEGMENTATION




GROCERY PAGE



Nutrition Facts	
As sold for 100 g 352 kJ (83 kcal)	
Energy	352 kJ (83 kcal)
Fat	0.56 g
Saturated fat	0.06 g
Monounsaturated fat	0.07 g
Polysaturated fat	0.1 g
Trans fat	0 g
Cholesterol	0 mg
Carbohydrates	15.1 g
Sugars	1.41 g
Added sugars	0.631 g
Fiber	4.31 g
Protein	5.43 g
Salt	0.566 g
Iron	0.5 mg
Vitamin A	0.2 µg
Vitamin D	0 µg
Vitamin C (ascorbic acid)	0.65 mg
Vitamin B1 (thiamin)	0.09 mg
Vitamin B2 (riboflavin)	0.06 mg
Vitamin B3 (niacin)	0.43 mg

RECOMMENDATIONS



Recommendation

****Microbiome**** Beans are a nutritious food that can be incorporated into a healthy diet. They are a good source of protein, fiber, and vitamins and minerals. Beans are also low in fat and cholesterol. The average nutrition facts for 2564 products of beans are as follows: * Energy = 382 kJ (91 kcal) * Fat = 0.56 g * Saturated Fat = 0.06 g * Carbohydrates = 15.1 g * Sugars = 1.44 g * Added Sugars = 0.631 g * Fiber = 4.03 g * Protein = 5.43 g * Salt = 0.566 g * Alcohol = 0 % * Vitamin A = 0.2 µg * Vitamin D = 0 µg * Vitamin C (ascorbic acid) = 0.65 mg * Vitamin B1 (thiamin) = 0.09 mg * Vitamin B2 (riboflavin) = 0.06 mg * Vitamin B3 (niacin) = 0.43 mg * Folate = 93.3 µg * Potassium = 288 mg * Calcium = 26.2 mg * Phosphorus = 101 mg * Iron = 1.43 mg * Magnesium = 48 mg * Zinc = 0.668 mg * Protein, vegetables, nuts and processed, animal and crew oils = 87.7 % Beans are a nutritious food that can be used in a variety of dishes. They can be added to soups, stews, salads, and burritos. Beans can also be used as a meat substitute in vegetarian and vegan dishes. Overall, beans are a healthy and nutritious food that can be enjoyed as part of a balanced diet.

Source

Open Food Facts:
<https://world.openfoodfacts.org/category/beans>

PLAGIARISM CHECK RESULT

Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store

ORIGINALITY REPORT

8 %	7 %	3 %	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	123dok.com Internet Source	1 %
2	eprints.nottingham.ac.uk Internet Source	1 %
3	open-innovation-projects.org Internet Source	<1 %
4	www.ece.anits.edu.in Internet Source	<1 %
5	businessstech.co.za Internet Source	<1 %
6	www.mdpi.com Internet Source	<1 %
7	arxiv.org Internet Source	<1 %
8	Nabil Ahmed, MD. Naimujjaman, Mahbuba Akhter, Hasan Monir. "DeeplabV3+ Driven Polyp Segmentation: Advancing Colonoscopy Diagnosis", 2023 26th International	<1 %

Conference on Computer and Information Technology (ICCIT), 2023

Publication

9	play.google.com Internet Source	<1 %
10	link.springer.com Internet Source	<1 %
11	docplayer.net Internet Source	<1 %
12	apkflash.com Internet Source	<1 %
13	Semra Erpolat Taşabat, Olgun Aydin. "chapter 8 Deep Learning", IGI Global, 2021 Publication	<1 %
14	dev.to Internet Source	<1 %
15	M. Corbetta, G. L. Shulman, F. M. Miezin, S. E. Petersen. "Superior Parietal Cortex Activation During Spatial Attention Shifts and Visual Feature Conjunction", Science, 1995 Publication	<1 %
16	support.myfitnesspal.com Internet Source	<1 %
17	www.omicsdi.org Internet Source	<1 %

18	core.ac.uk Internet Source	<1 %
19	eurchembull.com Internet Source	<1 %
20	mdpi-res.com Internet Source	<1 %
21	www.tutorialspoint.com Internet Source	<1 %
22	Natthapon Pannurat, Kanjana Eiamsaard, Chollasit Suthanma, Anan Banharnsakun. "Machine learning techniques for supporting dog grooming services", Results in Control and Optimization, 2023 Publication	<1 %
23	Sourav Garg, Niko Sünderhauf, Feras Dayoub, Douglas Morrison et al. "Semantics for Robotic Mapping, Perception and Interaction: A Survey", Foundations and Trends® in Robotics, 2020 Publication	<1 %
24	ijariie.com Internet Source	<1 %
25	Tarun Rao, N. Rajasekhar, T. V. Rajinikanth. "Drainage water level classification using support vector machines", 2013 Nirma	<1 %

University International Conference on
Engineering (NUiCONE), 2013

Publication

-
- | | | |
|----|---|------|
| 26 | hackernoon.com
Internet Source | <1 % |
| 27 | "Computer Vision – ACCV 2018", Springer
Science and Business Media LLC, 2019
Publication | <1 % |
| 28 | Meng, Weizhi, Wenjuan Li, and Lam-For Kwok.
"Design of intelligent KNN-based alarm filter
using knowledge-based alert verification in
intrusion detection : Design of intelligent
KNN-based alarm filter using knowledge-
based alert verification in intrusion detection",
Security and Communication Networks, 2015.
Publication | <1 % |
| 29 | Shubhika Garg, Ben Feinstein, Shahar Timnat,
Vishal Batchu, Gideon Dror, Adi Gerzi
Rosenthal, Varun Gulshan. "Cross-modal
distillation for flood extent mapping",
Environmental Data Science, 2023
Publication | <1 % |
| 30 | www.blogarama.com
Internet Source | <1 % |
| 31 | "Pattern Recognition and Image Analysis",
Springer Science and Business Media LLC,
2022
Publication | <1 % |

32	community.graphisoft.com Internet Source	<1 %
33	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1 %
34	Keming Jiao, Zhongliang Pan. "A Novel Method for Image Segmentation based on Simplified Pulse Coupled Neural Network and Gbest Led Gravitational Search Algorithm", IEEE Access, 2019 Publication	<1 %
35	medium.com Internet Source	<1 %
36	www.arxiv-vanity.com Internet Source	<1 %
37	www.ijisae.org Internet Source	<1 %
38	Boulos, Maged, Abdulslam Yassine, Shervin Shirmohammadi, Chakkrit Namahoot, and Michael Brückner. "Towards an "Internet of Food": Food Ontologies for the Internet of Things", Future Internet, 2015. Publication	<1 %
39	Jide Julius Popoola, Thompson Emeka Godson, Yekeen Olajide Olasoji, Michael Rotimi Adu. "Study on Capabilities of Different Segmentation Algorithms in Detecting and	<1 %

Reducing Brain Tumor Size in Magnetic Resonance Imaging for Effective Telemedicine Services", European Journal of Engineering Research and Science, 2019

Publication

40	www.frontiersin.org Internet Source	<1 %
41	www.techscience.com Internet Source	<1 %
42	www.waterrf.org Internet Source	<1 %
43	etda.libraries.psu.edu Internet Source	<1 %
44	www.freelancer.hk Internet Source	<1 %
45	www.rfsafe.com Internet Source	<1 %
46	"Computer Vision – ECCV 2020", Springer Science and Business Media LLC, 2020 Publication	<1 %
47	Aaqib Zahoor, Shamsul Hauq, Umar Bashir, Ambreen Hamadani, Shabia Shabir. "A meshwork of artificial intelligence and biology", Elsevier BV, 2024 Publication	<1 %

48	Abdelaziz Testas. "Distributed Machine Learning with PySpark", Springer Science and Business Media LLC, 2023 Publication	<1 %
49	Ben Vanderlei, James J. Feng, Leah Edelstein-Keshet. "A Computational Model of Cell Polarization and Motility Coupling Mechanics and Biochemistry", Multiscale Modeling & Simulation, 2011 Publication	<1 %
50	Jan Drole, Igor Pravst, Tome Eftimov, Barbara Koroušić Seljak. "NutriGreen image dataset: a collection of annotated nutrition, organic, and vegan food products", Frontiers in Nutrition, 2024 Publication	<1 %
51	Woon Siong Gan. "Signal Processing and Image Processing for Acoustical Imaging", Springer Science and Business Media LLC, 2020 Publication	<1 %
52	bib.univ-oeb.dz:8080 Internet Source	<1 %
53	corescholar.libraries.wright.edu Internet Source	<1 %
54	escholarship.org Internet Source	<1 %

55	fict.utar.edu.my Internet Source	<1 %
56	klabs.pr.ac.rs Internet Source	<1 %
57	scholar.sun.ac.za Internet Source	<1 %
58	thesis.binus.ac.id Internet Source	<1 %
59	www.coursehero.com Internet Source	<1 %
60	Sunanda Das, Awal Ahmed Fime, Nazmul Siddique, M. M. A. Hashem. "Estimation of Road Boundary for Intelligent Vehicles Based on DeepLabV3+ Architecture", IEEE Access, 2021 Publication	<1 %
61	dr.ntu.edu.sg Internet Source	<1 %
62	etd.aau.edu.et Internet Source	<1 %
63	export.arxiv.org Internet Source	<1 %
64	faculty.ksu.edu.sa Internet Source	<1 %
	icatsd2022.iuh.edu.vn	

65	Internet Source	<1 %
66	ipco-co.com Internet Source	<1 %
67	umpir.ump.edu.my Internet Source	<1 %
68	www.ijraset.com Internet Source	<1 %
69	"Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)", Springer Science and Business Media LLC, 2013 Publication	<1 %
70	Princy K. Saji, K. C. Krishnachalitha, M. Kannan. "chapter 5 Multi-Modal MRI Images Analysis for Improved Herniated Disc Diagnosis Using Deep Learning", IGI Global, 2024 Publication	<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	TAN CHEE LIN
ID Number(s)	20ACB01595
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	Mobile Application for Real-Time Food Image Segmentation and Nutritional Guidance at Grocery Store

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>8</u> % Similarity by source Internet Sources: <u>7</u> % Publications: <u>3</u> % Student Papers: <u>0</u> %	No comments.
Number of individual sources listed of more than 3% similarity: <u>0</u>	No comments.
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Mogana
Signature of Supervisor

Name: Mogana Vadiveloo

Date: 26/04/2024

Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)**

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB01595
Student Name	TAN CHEE LIN
Supervisor Name	Dr. Mogana Vadiveloo

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
x	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 26/4/2024