

DealWithIt - Real Time Price Checker with Object Recognition

By

Yvonne Eng Xin Yee

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: DealWithIt - Real Time Price Checker with Object Recognition


Academic Session: JAN 2024

I YVONNE ENG XIN YEE
(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:

140, LORONG BATU NILAM 12,

BANDAR BUKIT TINGGI 1,

41200 KLANG SELANGOR

Aun Yichiet

Supervisor's name

Date: 26/04/2024

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 COMMUNICATION AND TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 26/04/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that YVONNE ENG XIN YEE (ID No: 20ACB01886) has completed this final year project/ dissertation/ thesis* entitled "DealWithIt - Real Time Price Checker with Object Recognition" under the supervision of Aun Yichiet (Supervisor) from the Department of _____, Faculty/Institute* of _____, and _____ (Co-Supervisor)* from the Department of _____, Faculty/Institute* of _____.

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,



(YVONNE ENG XIN YEE)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**METHODOLOGY, CONCEPT AND DESIGN OF A 2-MICRON CMOS DIGITAL BASED TEACHING CHIP USING FULL-CUSTOM DESIGN STYLE**” 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 : YVONNE ENG XIN YEE

Date : 26/04/2024

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Aun Yichiet who has given me this opportunity to engage in an IT price checker mobile application design project. It is my first step to establish a career in IT design field. A million thanks to you.

To a very special person in my life, Chee Ngek Kiew, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

This project is focused on the field of Computer Vision (CV) for size measurement and object detection, machine learning for product recognition, Kivy for the front-end displaying in mobile application and real-time database with automated web scraping. It aims to solve the issues of consumers who always need to spend a large amount of time comparing the prices for the same product from different supermarkets. They also face confusion when the price is not as advertised, or the size of the product is not the same as previously bought. As such, a real-time price checker with product recognition and size measurement system in this project is needed to solve their problems by using web scraping to extract updated pricing information from online retailer websites to be stored in the real-time database. A machine learning model for product recognition will be built in this project to automatically recognize the type of product and retrieve its related information from the database. CV will be used to get the size dimensions from the product detected. The prices in the database will be updated regularly according to the prices from their respective supermarkets so that consumers will not get the wrong information. Overall, the functionalities include object detection, product recognition, size measurement, volume estimation, real-time database and web scraping with some steps in preprocessing of images. The functions will then be integrated and displayed on a mobile application so that consumer can just use their smartphone to scan and get product information in real-time easily.

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	x
LIST OF TABLES	xi
LIST OF SYMBOLS	xii
LIST OF ABBREVIATIONS	xiii
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	3
1.3 Project Scope and Direction	4
1.4 Contributions	5
1.5 Report Organization	6

CHAPTER 2 LITERATURE REVIEW	7
2.1 Review of the Technologies	7
2.1.1 Realtime Database	7
2.1.2 Web Crawling	8
2.1.3 Web Scraping	12
2.1.4 Kivy Python Mobile Application	13
2.1.5 Summary of the Technologies Review	14
2.2 Review of the Existing Systems/Applications	15
2.2.1 Object Size Measurement for Volume Estimation	15
2.2.2 Product Recognition with CNN	18
2.2.3 Price Checker Application	19
2.2.4 Summary of Existing System	22
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH	25
3.1 Methodologies and General Work Procedures	25
3.2 System Design Equations	25
3.2.1 Size Measurement Equations	25
3.3 System Design Diagrams	27
3.3.1 System Architecture Diagram	27
3.3.2 Use Case Diagram and Description	28
3.3.3 Activity Diagram	30
CHAPTER 4 SYSTEM SETUP	32
4.1 Hardware Used	32
4.2 Software Setup	32
4.2.1 Pycharm	33
4.2.2 Jupyter Notebook	33
4.2.3 Chrome	34
4.2.4 Ubuntu	34
4.3 Tools requirements	34
4.3.1 Chrome Web Driver	34
4.3.2 BrowserMob Proxy	35
4.3.3 Android Debug Bridge (adb)	35

4.3.4 OpenCV Android SDK	36
CHAPTER 5 SYSTEM DESIGN/ IMPLEMENTATION	37
5.1 System Block Diagram	37
5.2 Kivy Mobile Application	37
5.2.1 OpenCV Object Detection	38
5.2.2 OpenCV Size Measurement	38
5.2.3 Loading Product Recognition Model and Estimating Volume	39
5.2.4 Estimated 3D Volume	40
5.2.5 Size Grading Categories	41
5.2.6 Display Related Product Latest Price	41
5.2.7 Deploying Kivy Application	41
5.3 CNN Product Recognition Model	42
5.4 Real-Time Firebase	43
5.5 Selenium Web Scraper	46
5.6 Window Task Scheduler	48
5.7 RestAPI	50
5.8 System Operation	52
5.8.1 Mobile Application User Interface	52
5.8.2 Scraped Data on Firebase	54
5.8.3 Rest API Data	55
5.9 Implementation Issues and Challenges	55
5.10 Concluding Remark	56
CHAPTER 6 SYSTEM TESTING AND EVALUATION	58
6.1 System Testing and Performance Metrics	58
6.2 Testing Result	58
6.2.1 Size Scale, Dimensions and Product Recognition at a Constant Distance	59
6.2.2 Size Scale, Dimensions and Product Recognition at Different Distance	61
6.2.3 Test Accuracy for Products Recognition Model	62

6.3 Project Challenges	62
6.4 Objectives Evaluation	63
CHAPTER 7 CONCLUSION AND RECOMMENDATION	64
7.1 Conclusion	64
7.2 Recommendation	65
REFERENCES	66
WEEKLY LOG	70
POSTER	73
PLAGIARISM CHECK RESULT	74
FYP2 CHECKLIST	76

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1.1	Architecture of a Web Crawler	8
Figure 2.1.2	Basic Flowchart of Crawler	9
Figure 2.1.3	Architecture of Priority Based Focused Crawler	9
Figure 2.1.4	Structure of Web Scraping	12
Figure 2.1.5	Selenium components	13
Figure 2.2.1	Overflow of proposed method in project	16
Figure 2.2.2	General principles of image formation for optical cameras	17
Figure 2.2.3	Example of pixel-wise dimensioned measured by active contour model	17
Figure 2.2.4	Flowchart of product image recognition system	18
Figure 2.2.5	CNN architecture for Object Identification	18
Figure 2.2.6	Algorithms for Dataset Creation	20
Figure 2.2.7	Algorithms for Related Searching Products	21
Figure 3.3.1	System Architecture Diagram	27
Figure 3.3.2	Use Case Diagram	28
Figure 3.3.3	Activity Diagram	30
Figure 4.2.1	Pycharm Logo	33
Figure 4.2.2	Chrome Logo	33
Figure 4.2.4	Ubuntu Logo	34
Figure 4.3.1	Chrome Version	35
Figure 4.3.2	SDK Platform-Tools Page	35
Figure 5.1	Main System Components	37
Figure 5.2.1	Aruco Marker with 5cm x 5cm dimension	39
Figure 5.2.2	Configuration for Getting Aruco Marker	40
Figure 5.2.3	Loading Product Recognition Model and Predicting	40
Figure 5.2.4	Functions for Loading Model, Preprocessing and Predicting	40
Figure 5.2.5	Calculation of Estimated Volume	40
Figure 5.2.6	Guideline Size Grading based on Diameter	41

Figure 5.2.7	Page for download Linux kernel package	42
Figure 5.2.8	Successful Packaging Kivy Program Output	42
Figure 5.3.1	Code Snippet for Saving Model in tflite format	43
Figure 5.4.1	Pages for Creating a Firebase Project	44
Figure 5.4.2	Pages for Registering Firebase as a Web App	44
Figure 5.4.3	Important Firebase Config	45
Figure 5.4.4	Pages for setting up realtime database	45
Figure 5.4.5	Pages for Firebase rules configuration	46
Figure 5.4.6	Pages for Realtime database URL	46
Figure 5.5.1	Initializing Browser-mob Proxy	46
Figure 5.5.2	Configuration of Chrome Web Browser	47
Figure 5.5.3	Closing Up Pop-up Message	47
Figure 5.5.4	Firebase Initialization and Pushing Data	48
Figure 5.6.1	Creating Basic Task	48
Figure 5.6.2	Settings for Window Task Scheduler	49
Figure 5.6.3	Bash File	49
Figure 5.6.4	Command for Program/script	50
Figure 5.7.1	Firebase configuration API code	50
Figure 5.7.2	Retrieving Firebase Data Code	51
Figure 5.7.3	Filter Firebase data Code	51
Figure 5.8.1	User capturing photo	52
Figure 5.8.2	Warning Message that Aruco Marker not detected	52
Figure 5.8.3	Confirmation Message on Correct Product Name	53
Figure 5.8.4	Result for Object Detected, Dimensions, Product Name, Estimated Volume, Size Scale	53
Figure 5.8.5	Scraped Price Result with Related Product Name for Different Supermarkets	54
Figure 5.8.6	Successful Scraped Supermarket Name	54
Figure 5.8.7	Formatted Product Name and Price on Firebase	55
Figure 5.8.8	Filtered Retrieved Data from Firebase	55
Figure 5.9.1	Blocked from Grocery Stores Website	56
Figure 5.9.2	Grocery Stores Websites Maintenance	56
Figure 6.2.1	Examples of Products to be Tested	59

Figure 6.2.2	Result for Small Apple, Medium Orange, Large Orange	60
Figure 6.2.3	Real Measurement of Width for Large Orange	60
Figure 6.2.4	Real Measurement of Height for Large Orange	61
Figure 6.2.5	Size Measurement for Large Orange at Different Distance from Camera	62
Figure 6.2.6	Test Accuracy of Products Recognition Model	62

LIST OF TABLES

Table Number	Title	Page
Table 2.1.1	Distributed Crawler Statistics	11
Table 2.1.2	Review of Technologies Implemented	15
Table 2.2.1	Various Architectural Types of CNN	19
Table 2.2.2	Comparison of previous work with current solution in size measurement	22
Table 2.2.3	Comparison of previous work with current solution in product recognition	23
Table 2.2.4	Comparison of previous work with current solution in price checker	24
Table 4.1	Specifications of Laptop	32
Table 4.2	Specifications of Android Mobile Phone	32
Table 6.2.1	Comparison of Size Dimension Between Application and Real Life	61

LIST OF SYMBOLS

π

Pi

LIST OF ABBREVIATIONS

<i>2D</i>	Two-dimension
<i>3D</i>	Three-dimension
<i>SI</i>	International System of Units
<i>UI</i>	User Interface
<i>AR</i>	Augmented Reality
<i>CNN</i>	Convolutional Neural Network
<i>CV</i>	Computer Vision
<i>AI</i>	Artificial Intelligence
<i>JSON</i>	JavaScript Object Notation
<i>cm</i>	Centimeter
<i>m</i>	Meter

Chapter 1

Introduction

In this chapter, background information, problem statement and motivation of this research, project scope and direction, project objectives, the impact, significance and our contributions to the advancement of the IT field to the targeted audience of consumers are presented.

1.1 Problem Statement and Motivation

The findings from [1] suggest that most customers have a shopping pattern of spending less time in making their selection of products and many tend to overlook the price of the item they purchased. As such, when they end up paying money at the cashier, they often find out that they need to pay more than what they expect. This happens mostly when the consumers are unwilling to waste their time in comparing the prices or even to compare the difference in scale of the items between different grocery stores or supermarkets. In fact, many consumers are not good at estimating a change in product sizing. People are said to routinely underestimate something much bigger. Their visual perception is more inclined to apply addition rather than multiplication using height, width and length when trying to gauge the volume of the product as they don't have sufficient details for the products [2]. Moreover, the sizing standards defined by every retailer are different, the typical case is that the labeled medium-sized apple from Seller A might be significantly larger than the labeled medium-sized apple sold by Seller B. There is currently less universal sizing scale system that consumers can make use of to get a similar scale for the product of the same category in every supermarket or retail shop. Not only that, but more wastage also occurs when consumers purchase a portion that is not suitable for their household size as they misjudge the size.

Next, many argue that they cannot persist in comparing products to get the best deals as they feel exhausted after a long time of searching, finding, assessing, comparing, and deciding the best price. However, as stated by Dawar [3], they are the target for marketers as laziness is easy prey as marketers are always in tune to make the comparison of pricing and sizing tough. Generally, people get cheated as they rarely realize the changes in products even if the retailer has secretly reduced the product size while maintaining the price for their

profit. Consumers fall for the trap of paying more when they just want the products quickly and are too lazy to plan ahead.

Besides, it is a common scenario that most consumers may find themselves being deceived as the price being displayed for certain products on the shelves is not the same as the price they paid at the checkout. The prices for some retail stores are not updated immediately with the price changes due to factors like supermarket staff forgetting to renew the labeling prices or caused by the system delay. In [4], many customers declared that their shopping bills occupy a big portion of their overall spending as supermarkets are accused of not keeping up with the increased price. Based on past statistics, consumers are often persuaded to buy more for the fear of losing something worthy [5]. As such, consumers get disappointed and even angry as they thought they got the promotion for the item, but it turns out that they need to pay higher as the discount has passed its valid date. Consumers are often tricked by the pricing strategies employed by the modern retailer by instilling them a concept that their prices are lower than other competitors in the market. Wells [5] also mentioned that consumers' brains will process \$5.00 and \$4.99 as different values which gives them an interpretation that the \$4.99 price is better than the round number \$5.00 price point. The perception of people is that price is smaller as they carry fewer syllables.

It is important to address these issues faced by consumers to enhance their shopping experience and to cultivate an assistive and positive environment for shopping with the advancement of technology in the future. We need to build a system that can streamline the shopping process of consumers, saving time and effort in browsing, accessing, comparing prices between supermarkets, and making purchasing decisions. A stress-free shopping environment should be fostered for consumers without the frustrations of dealing with sizing and pricing discrepancies of the products or missing, inaccurate, and outdated pricing labels. It should be noted that a nation's progress is not only measured by economic growth but also by the well-being and welfare of its citizens who also act as consumers. Consumers need to be given the power and their rights to make smart choices by taking control of their spending and purchasing decisions back from the marketers to not be easily manipulated by them. Finally, it will boost the economy when consumers gain back their faith that their consumer rights are protected and spend more in the grocery stores.

1.2 Objectives

The main objective of this project is to develop a mobile application of a real-time price checker with size measurement and product recognition models applicable to mainly Android users. Users will have knowledge of the estimated 3D volume calculated based on a 2D product's image according to the dimensions of width and height detected by the system. The diameter obtained helps in representing their respective size category derived from the current common industry practices for grading fresh products of fruits and vegetables. To retrieve and obtain the most updated price for the specific product from different supermarkets, the product recognition model and web scraper are utilized with the real-time database to store the product's name and prices. The sub-objectives are discussed further below to strengthen the innovation of this project:

- a) To design a parallax corrective method for width and height estimation using the referenced object

After the object from the image is detected by keeping track of its continuous contour, a minimum rectangular border will be drawn to identify its width and height. The distance for width and height in pixels from the screen will be converted to a measurable unit in cm which is more relatable to real-life applications through the ratio of pixel to cm deriving from the referenced object with a known dimension. With that, the Aruco Marker which is an artificial square marker characterized by a broad black border and an internal binary matrix that serves as its identifier (id) will be used as the referenced object in this project as it can be detected easily by the OpenCV.

- b) To develop a 3D volumetric estimation method based on the approximate geometric shape of the product from a 2D image

With the dimensions obtained from the 2D image, the estimated volume can be calculated using the mathematical formula from the approximate geometric shape of the product like sphere or cylinder. The category for the products' shape will be organized following the product's name. With that, a trained CNN model covering various types of fruits and vegetables from the market will be used for automatic product recognition to retrieve the product name held by the user and do the subsequent computation.

- c) To capture, save and maintain updated pricing information on the real-time database
- A real-time database which is the Firebase will be used to store products' names and prices from different grocery stores. The extraction of product information from online website sources of the major grocery retailers will be done by the web scraping technique of Selenium due to its flexible interactivity with the webpages. The automation of job scheduling for the web scraper task will be done through the Task Scheduler from Windows so that the updated price can be scraped regularly with the chosen settings to update the Firebase. It helps to ensure that the data will always be changing and replaced by the new data without manually updating them every time. A Flask server will then be needed to access and retrieve data from Firebase, preprocess, and filter the data before passing it to the mobile application to be displayed.

The areas that will not be covered in this project are the AR size measurement from a 3D model to get a more detailed dimension, depth estimation which requires calibration of the camera from the extrinsic and intrinsic parameters and weight estimation in grams derived from the estimated volume of product. The project focuses on the category of fresh products covering fruits and vegetables. As there are many brands for the same category of products, it is difficult to train the product recognition model as their appearance and size look almost the same just the country imported varies.

1.3 Project Scope and Direction

This project develops a mobile application that can be used by every user having a smartphone. It will act as a great time saver for consumers to compare product prices and sizes across different supermarkets all in one screen without having to search them one by one by physically traveling to other supermarkets and doing the comparison. The details of the products are displayed on the screen after the product is recognized by the product recognition deep learning model through the built-in camera from the phone to provide an intuitive user interface. For the backend part, there will be a real-time Firebase set up for storing the updated products' pricing information with the use of Selenium. It ensures the pricing to be tally with the current market as web scraping can help in ensuring the most up-to-date pricing and product information due to their ability to detect changes in real-time. As

such, consumers can obtain real and direct information about the products without any misleading title or advertising influence from social media.

The object detection and product recognition models will be done by using OpenCV and Python deep learning modules like TensorFlow. As there is a wide range of products across the global market, this project is confined to recognizing the products coming from the category of fresh products only which are fruits and vegetables. In this project, the products list to be recognized are lemon, ginger, cucumber, turnip, garlic watermelon, potato, lettuce, mango, tomato, grapes, pear, orange, cauliflower, jalepeno, eggplant, carrot, peas, capsicum, beetroot, chilli pepper, cabbage, pomegranate, paprika, corn, sweetpotato, raddish, apple, soybeans, spinach, sweetcorn, pineapple, onion, bell pepper, kiwi and banana. After the product name has been identified, size measurement using OpenCV with a referenced object will take place from a 2D image to get the dimension needed to calculate the estimated volume. With this textual information, users get to compare the sizing of the products more easily as they no longer need to rely on their visual observation. The volume calculation will follow the specific formula of 3D geometric shapes resembling the products' shapes. For example, the volume for orange will be computed using the formula of ellipsoid or sphere. All the output information for estimated volume, sizing category, product name and pricing with the grocery store's name will be displayed in the mobile application.

1.4 Contributions

A convenient mobile application that enables contactless and effortless size measurement in knowing dimensions, estimated volume, size category and pricing of a product. It helps in reducing the necessity to use the traditional physical measurement tool. Incorporating the real-time price checker by retrieving updated product data from the database, consumers can always check the price and size of the same product under different retail stores using minimal time and steps to get the best deal. They will be empowered to make a more informed purchasing decision when they have the idea that the same price doesn't represent the same size or portion from retailers. They will possess the capability to detect the changes even if the retailer secretly cuts down the size while maintaining or increasing the price. Not only the comparison can be carried out between supermarkets, but consumers can also use it to compare available products in supermarkets with their previously bought products in their house so that they can get back the similar preferred size for the old product. This also helps to prevent wastage as the possibility for them to get an

inappropriate portion that cannot be consumed within the timeframe or expired date is decreased. They get to know what size is more appropriate for the food intake capacity of their household and are able to finish it before the food spoils. As such, the goal for SDG 12 which is responsible consumption and production can be attained when the food loss is getting lower from the consumer side and the surplus food resources get distributed more equitably to other needy.

Furthermore, the product recognition model eases the automation processes and reduces consumer's time and effort in manually searching and looking for a product from a long name list. OpenCV implemented in the application can also help the consumer visualize the discrepancy between products more clearly. It gives them a better view to do a direct comparison in terms of sizing and pricing as the product data are all shown on one screen at a place at a time. With that, price transparency can be achieved while at the same time eliminating price discrimination which is when the seller is charging the customer different prices for the same products based on the maximum possible amount that the seller thinks the buyer is willing to give with their respective age groups or appearances.

1.5 Report Organization

The report is divided into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology/Approach, Chapter 4 System Setup, Chapter 5 System Design/ Implementation, Chapter 6 System Testing and Evaluation and Chapter 7 Conclusion and Recommendation. The first chapter is the introduction of this project which includes the problem statement, project background and motivation, project scope, project objectives, project contribution, highlights of the project, and report organization. The second chapter is the literature review carried out on several technologies and existing applications to evaluate their strengths and weaknesses. The third chapter discusses the overall system design of this project. The fourth chapter is regarding the initial setup and the necessary software or tools. The fifth chapter reports the details on how the design of the system is carried out and how to implement them. The sixth chapter demonstrates the test cases and performance of the system. The seventh chapter conclude the project with some future recommendations.

Chapter 2

Literature Review

2.1 Review of the Technologies

2.1.1 Realtime Database

A real-time database is a database hosted in the cloud, where data is stored in JSON format and continuously synchronized with each connected client. When cross-platform applications are developed using IOS, Android, and JavaScript SDKs, a significant portion of users will rely on a single Real-time Database instance to receive real-time updates with the latest data [6].

Firestore

There are many functionalities that make the Firestore popular such as authentication, hosting, messaging, analytics and storage. In [6], the control access to the application will occur securely with many login options including Gmail, GitHub, Twitter, and Facebook, or create custom authentication methods. It also efficiently deploys web apps and static content to a Content Delivery Network (CDN) with Hosting by including support for custom domains, a Global CDN, and SSL certificates. It will notify users about new content or engage them with notification messages for reengagement across different platforms at no cost. To ensure data security, after it stores and retrieves images, videos and audio from the client SDK, the uploading and downloading will happen in the background.

Strength

The strength is that it enables seamless and secure file transfer for Firestore apps, irrespective of network conditions. It's supported by Google Cloud Storage, an economical object storage service that allows developers to store various types of content like images, audio, video, and user-generated data with ease [7]. Also, in the case of crashing, errors will be grouped into different groups according to their severity. This helps users in creating custom events to capture and handle different kind of errors accordingly.

Weakness

Although Firestore supports offline operation, there can be issues with data inconsistencies when a client goes offline and then reconnects. Also, unlike traditional SQL

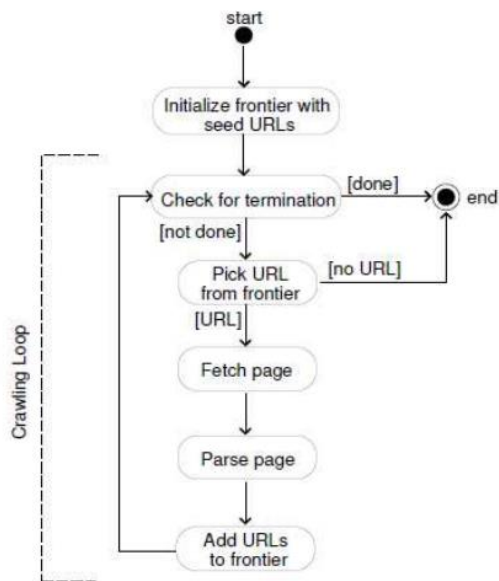


Figure 2.1.2 Basic Flowchart of Crawler [8]

Self Adaptive Semantic Focused Web Crawler

A Focused Crawler is also referred to as a Topic Crawler due to its working approach. This is because it gathers specific and pertinent documents related to the given subject topic [9][10]. In [11], it aims to retrieve interconnected web pages. This specialized crawler evaluates two key aspects: relevance and way forward. It assesses the degree of relevance of a given page to the chosen topic and the optimal path to proceed further.

Its priority is work as shown in Figure 2.1.3. In [12], the web crawler starts with an initial seed and downloads web pages. It then extracts new URLs from these pages to be downloaded and calculates the semantic similarity. Web pages are prioritized based on semantic scores. Finally, it will continue to download more new URLs.

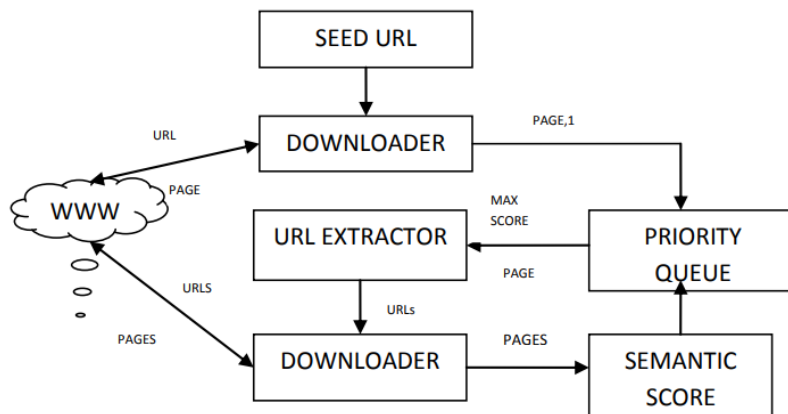


Figure 2.1.3 Architecture of Priority Based Focused Crawler [12]

Strength

The strength that it held included its cost-effectiveness concerning hardware and network resources. They are capable of minimizing network traffic and downloads, as mentioned in [13]. Additionally, this crawler provides extensive search exposure and can give a high harvest rate.

Weakness

However, as the focused crawling techniques often download irrelevant web pages, it's straining network bandwidth. To maintain database freshness, they use polling or multiple crawlers, both bandwidths intensive. Continuous server access impacts server performance. The challenge is to retrieve highly relevant pages selectively and use alternative polling methods to avoid server overload [14].

Solution

The solution proposed is to improve relevance filtering by enhancing the relevance assessment algorithms to filter out irrelevant pages during the crawling process. Utilize advanced natural language processing and machine learning techniques to identify highly related content. Besides, a more selective page selection strategy can be implemented. Prioritize crawling based on content significance, historical relevance, or user interaction data to target highly relevant pages first.

Crawling Page Based on Incremental Web Crawler

It methodically updates the current page collection by making frequent visits to those pages. This frequency of visits is determined based on estimations of how frequently the pages undergo changes [15]. Furthermore, it replaces less significant pages with newer and more crucial ones as part of its ongoing maintenance. The crawler was used on the basis of a crawler architecture named Scrapy in [16]. After it crawls the information from websites, the Bloom filter can help monitor the website and store them in the database.

Strength

It exchanges less important pages with new and more important pages. It resolves the problem of the freshness of the pages. The benefit of the method in [17] is that only the valuable data is provided to the user, thus network bandwidth is saved and data enrichment is achieved. The user will also be notified if the web link has been updated.

Weakness

However, it is quite challenging to keep the local collection of web pages fresh and up to date. This challenge arises because the freshness of the collection can significantly vary based on the strategies employed by the crawler for web crawling [18].

Solution

The solution proposed is that the incremental crawler can implement dynamic crawling policies that adapt to changes in web content. Prioritize revisiting pages with a higher likelihood of updates, based on historical data and page characteristics. Besides, keep on monitoring webpages in real-time for changes using techniques like change detection algorithms. When a change is detected, the page can be revisited and updated accordingly.

Design and Implementation of Distributed Crawler

Distributed web crawling is a method of distributed computing. Multiple crawlers collaborate during the web crawling process to achieve extensive web coverage. A central server oversees the communication and coordination of these distributed nodes, even when they are spread across different geographical locations [19].

The crawler in [20] performed a checkpoint every 4 hours, resulting in a brief period of slow crawling activity, as depicted in the graph showing incoming bytes. These low points in the graph can help estimate the bandwidth consumed by the crawler versus other campus traffic. This pattern is not observed in outgoing bytes because the crawler primarily sends small requests. However, it is noticeable in the number of outgoing frames, partially due to HTTP requests and the DNS system. Table 2.1.1 shows the statistics for the distributed crawler in [21]

Table 2.1.1 Distributed Crawler Statistics [21]

HTTP requests	161,549,811
network errors	5,873,685
read_timeout_exceeded errors	2,288,084
robots.txt requests	16,933,942
successful non-robots requests	138,742,184
average size of page	13,354 bytes
total data size	1.85 TB

Strength

The strength is its approach to employing the PageRank algorithm, which enhances both the efficiency and the quality of search results [19]. Additionally, it is robust against

system crashes and unforeseen events. It can also be tailored to suit a wide range of crawling applications.

Weakness

However, it is ineffective in partitioning the collection. The collection needs to be partitioned in a way that allows the query answering phase to query the smallest necessary subset of partitions possible but not all. Thus, increasing the difficulty in the maximum retrieval of the relevant document [18].

Solution

The solution suggested is to utilize dynamic partitioning strategies that adapt to changing data and query patterns. Regularly doing the analysis of the collection and adjust partition boundaries as needed to maintain efficiency. Besides, content-based partitioning can be implemented where documents with similar content or topics are grouped together. This helps ensure that queries can be directed to the most relevant partitions, reducing the need to search across all partitions.

2.1.3 Web Scraping

Web scraping is a good technique in capturing unstructured data from websites and converting it into structured formats that are more suitable for storage and analysis in databases. It is also referred to as screen scraping, web data extraction, web harvesting, web data scraping or, it serves as a vital tool in the realm of data mining. The primary objective of web scraping is to glean information from websites and organize it into easily understandable common structures such as databases, spreadsheets or comma-separated values (CSV) files. This process enables the retrieval of diverse data types including item pricing, stock pricing, reports, market trends, and product details [22]. By extracting targeted information from websites, web scraping empowers users to make informed decisions and optimize their operations effectively.



Figure 2.1.4 Structure of Web Scraping [22]

Selenium

Being a browser automation tool, it enables the developers to manage web browser programmatically as if there is a real human interacting with the websites. It is available in many popular programming languages like Python, C# and Java. The APIs for web page

engagement encompass the functionalities such as clicking, typing, entering, scrolling and navigating. The component of Selenium is divided into different components as shown in Figure 2.1.5.

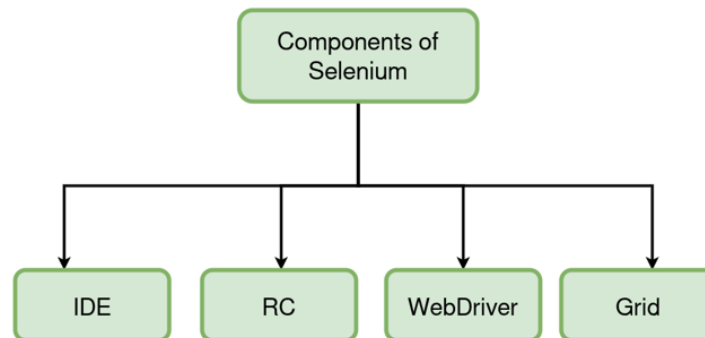


Figure 2.1.5 Selenium components [22]

Strength

Selenium stands out as an open-source automation testing tool, offering cost-free usage. It empowers testers with considerable flexibility to craft sophisticated and intricate test cases. Furthermore, Selenium accommodates test scripts scripted in a variety of user-preferred languages, including C#, Java, Python Perl, and PHP. Its versatility extends to supporting test case execution across multiple operating systems such as Windows, Linux, Android, Mac, and iOS. Additionally, Selenium facilitates testing on diverse web browsers, encompassing Chrome, Firefox, Internet Explorer (IE), Opera, and Safari. Notably, it permits test case execution even when the browser window is minimized and supports parallel test execution. Selenium's integration capabilities extend to seamless collaboration with TestNG and JUnit for generating test reports and managing test cases. Moreover, it seamlessly integrates with Jenkins, Docker, and Maven to enable continuous testing. [23]

Weakness

Selenium's testing capabilities are limited to web applications, excluding software and desktop applications from its purview. It lacks the ability to access web elements beyond the scope of the applications under test. Furthermore, users must rely on customer communities for support, as there is no assured user assistance provided. Notably, Selenium lacks built-in support for image testing and necessitates integration with Sikuli for this purpose.

2.1.4 Kivy Python Mobile Application

Kivy is a powerful open-source Python framework designed to simplify the process of building dynamic and engaging mobile applications for multi-platforms that can run on

Windows, OS X, GNU/Linux, iOS, Android and Raspberry Pi. It allows the access to mobile APIs to utilize the features on the phones like camera, GPS, Internet and so on. It offers a comprehensive set of tools and widgets for building intuitive user interfaces that can support multi-touch functionality, enhancing user experience [24].

Strengths

The significant advantages of Kivy lies in its complete free-of-charge nature, making it easily accessible for developers for both development and commercial purposes. Kivy offers robust multi-platform support, enabling developers to create projects once and effortlessly deploy them across various platforms. Kivy facilitates swift application development with its rich library ecosystem by leveraging the simplicity of the Python programming language. Moreover, Kivy simplifies the development process by allowing developers to write code once and reuse it for publishing multiple applications. Additionally, Kivy's user-friendly built-in widgets, equipped with multi-touch support, enhance ease of use and streamline development efforts. Notably, Kivy's superior performance, coupled with its intuitive GUI, distinguishes it from HTML5 alternatives, ensuring rapid application development and optimal user experiences [24].

Weakness

The user interface of the Kivy platform appears non-native, which poses a challenge for users who are unfamiliar with the framework, requiring them to possess prior knowledge for intermediate programming skills. Moreover, the considerable package size of the Kivy framework need a longer download times due to the inclusion of the Python interpreter within the package, thus amplifying its bulkiness. As the Kivy framework suffers from inadequate community support, characterized by its small size, it results in prolonged wait times for developers seeking assistance with their queries and issues. The subpar documentation further exacerbates usability issues, as users encounter difficulties navigating the framework without comprehensive guidance. In contrast, competing platforms such as React Native offer robust functionality for mobile application development and boast stronger community support, positioning them as formidable alternatives to the Kivy framework.

2.1.5 Summary of the Technologies Review

Table 2.1.2 Review of Technologies Implemented

Technologies	Strength	Weakness
Firestore	-store various types of content like	-data inconsistencies when it goes

	images, audio, video, and user-generated data with ease	offline and then reconnects
Selenium	-facilitates testing on diverse web browsers	-lacks built-in support for image testing
Kivy	-simplifies the development process by writing code once and reuse it for publishing multiple applications	-need longer download times due to the inclusion of the Python interpreter within the package

2.2 Review of the Existing Systems/Applications

2.2.1 Object Size Measurement for Volume Estimation

The project in [25] are composed of object detection and object measurement sections. Objects are detected from the pre-processed video frame using the algorithm of canny edge detector including dilation and erosion, Gaussian filter and Morphological operations. During the gradient computation stage, the edge gradient and direction for each pixel will be identified by utilizing Sobel operator for a smoothed frame. Full frame scan was performed to eliminate undesired pixels that are not contributing to the formation of edge. Non maximum suppression help transforms smoothed edges in the gradient magnitude into sharp edges. The final step involves hysteresis thresholding, which distinguishes between definite edges and non-edges. In this step, two threshold values chosen empirically based on the frame's content will determine whether an edge is strong or weak for determining the final edge. With that, after a bounding box rectangle is drawn on the object detected, pixels per metric variable will be calculated with the midpoints of the bounding box through fixed setting of reference object being placed at the left-most part of the image. Figure 2.2.1 shows the flowchart for the proposed system.

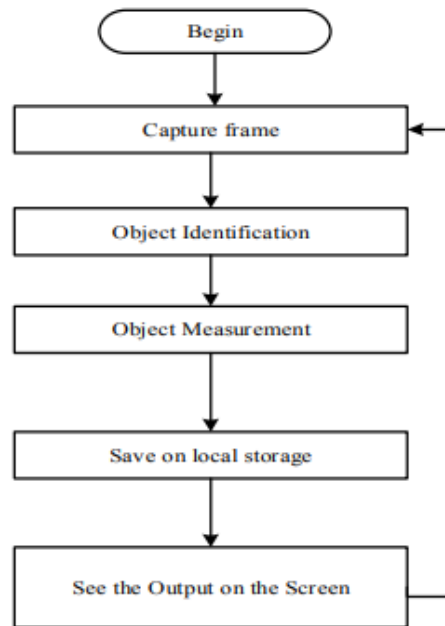


Figure 2.2.1 Overflow of proposed method in project [25]

Besides, the general principles of image formation and calibration of the camera from [26] is illustrated in Figure 2.2.2, where the green eclipse if the object of interest, its physical dimensions, like height and width, are represented by "L" in inches, the corresponding size of the ellipse on the image plane is denoted as "x" in inches, distance from the center of the object to the camera lens signified by "m" in inches, while the focal length, denoted by "d," is the distance between the lens and the image plane. With this, the formula of $\frac{L}{m} = \frac{x}{d}$ is formulated by geometry.

In this proposed system, the active contour model is utilized to identify and measure the dimensions (height or width) of the object of interest in a captured photo. Compared to various other object detection algorithms, the active contour model is favored because it avoids misinterpreting the details pattern within an object as a separate object. A frame is applied before activating the contour model, allowing the selection of a region or sub-image that includes the object of interest after each photo is taken. Subsequently, the active contour model is employed to identify the object's boundary and measure its pixel-wise dimensions. Figure 2.2.3 shows an example of the measured dimensions for the object of interest, p_1^{width} and p_1^{height} , using the active contour model.

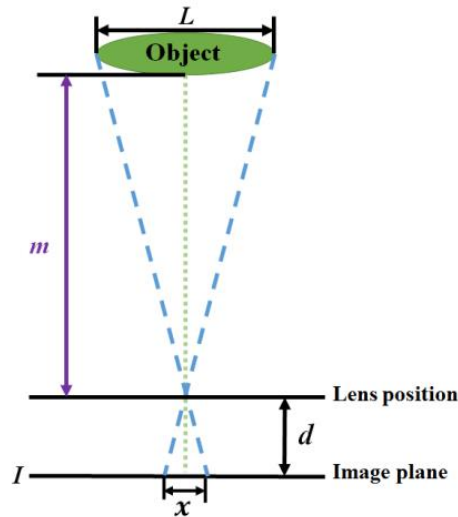


Figure 2.2.2 General principles of image formation for optical cameras [26]

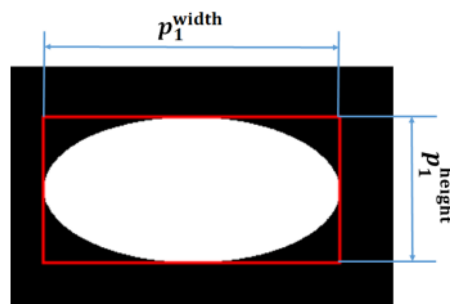


Figure 2.2.3 Example of pixel-wise dimensioned measured by active contour model [26]

According to [27], one way to estimate the surface area and volume of irregular objects involves using the finite element method (FEM). In this method, a two-dimensional coordinate grid is created to represent the object's shape. The object is then divided along a chosen axis, and measurements of these sections are utilized to estimate the surface area and volume of a reconstructed object. Researchers, such as Goñi et al. [27], applied this approach to estimate the surface area and volume of objects like apples and meat pieces, achieving an estimation error of less than 2%. They found that employing the lofting technique enables obtaining an accurate representation of the real shape of irregular multi-dimensional food items.

Strength

Preprocessing of images was done for a more accurate object detection for the object of interest. In [25], the noise is removed through Gaussian filter to smoothen the frame and discard non-important edges while the image processing algorithm called active contour model is used in [26].

Weakness

Bachelor of Computer Science (Honours)
Faculty of Information and Communication Technology (Kampar Campus), UTAR

The result for both methods in [25] and [26] will be affected by the angle taking the images and lens deformation. For example, the accuracy for taking the images from top view and side view will be different. Furthermore, the measurement system from [26] uses its own measuring scale scheme to compare different object in contrast to [25] which use metric as the unit which are more relatable to consumers to compare the object. Additionally, the work in [27] requires cross-section of products which consist of many images to get the precise volume. Many steps, time with suitable place and equipment like knife will be needed to achieve it.

2.2.2 Product Recognition with CNN

With the advancement of various electronic gadgets for photographing, the amount of good-quality image data for better analysis and processing is also increasing. There are 5 main steps to achieve for product recognition, (1) Image capture to collect images from phones and cameras, (2) Image preprocessing to provide cleaned and quality-focused images, (3) Feature extraction to determine the critical characteristics, (4) Feature classification to classify the features and (5) Recognition output where trained model will be involved to predict the label class for the retail product [28]. The steps for the product image recognition system are composed as in Figure 2.2.4.

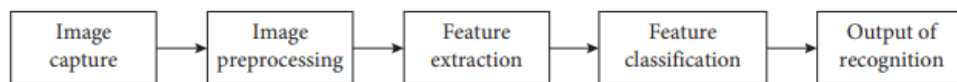


Figure 2.2.4 Flowchart of product image recognition system [28]

SIFT and SURF algorithms are considered in [28] to extract the features from images. CNN which is one type of deep neural network has been exploited for object detection. Before detecting the product, it is vital to obtain the region of interest from the different objects. The object detection methods are categorized into two ways: two-stage-model (region proposal-based) and one-stage-model (regression/ classification-based) [29]

In [30], it stated that the correlation between pixels needs to be maintained when processing the images and mathematical neural networks exist to keep the spatial relationship between pixels. Nonetheless, it is unnecessary to include information on every pixel in an image as the size can be unpredictably huge. Before putting pixels into CNN, all information from the RGB upsides of each pixel has to be prepared for the conventional neural organization when layers proceed toward the start of the neural organization and produce a

modest outcome. Table 2.2.1 demonstrates the different architectural types of CNN while Figure 2.2.5 shows the CNN architecture for object identification.

Table 2.2.1 Various Architectural Types of CNN [30]

Sno	Architecture Type	No of Arguments
1	LeNet	60 k
2	AlexNet	60Mill
3	ZFNet	—
4	GoogLeenet	4Mil
5	VggNet	138Mil

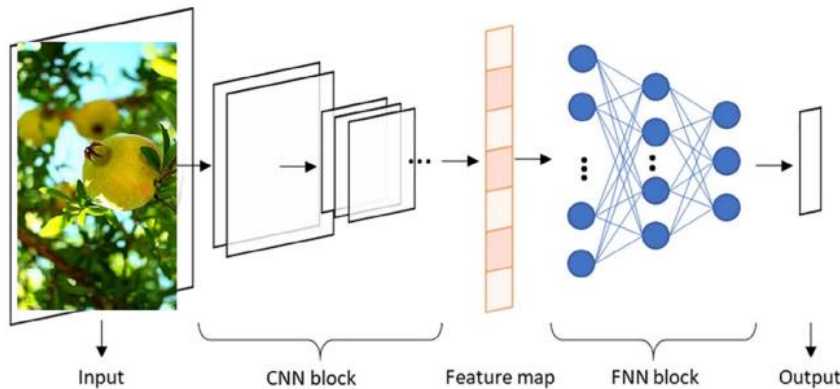


Figure 2.2.5 CNN architecture for Object Identification [30]

Strength

SIFT feature extractor brings many benefits like scale invariance, translation invariance, distinctiveness and wide applicability. The two-stage model takes the textures, edges and colours from images into account for a high recall rate with low windows selected. This is because it needs a region proposal algorithm to deduce the location of the object in a graph. On the other hand, the one-stage method computes faster as categories are regressed from multiple locations of the image.

Weakness

However, being traditional feature extraction technology, SIFT and SURF cannot reflect fully all the required information as the features extracted are hand-crafted. Also, even though the accuracy of the two-stage method stands out, both of the stage methods are still slower than other representative detection algorithms like YOLO [31] and SSD [32]. Moreover, CNN is incapable of memorizing some previously learned objects when adjusted to a new dataset [30].

2.2.3 Price Checker Application

A Bangladesh price comparison system has been created in [33] for the benefits of their citizens so that they can make a better purchase decision motivated by the similar tools from India listed as www.mysmartprice.com and the other one is www.compareraja.in. Figure 2.2.6 represents the pseudocode for the dataset creation from e-commerce websites while Figure 2.2.7 demonstrates the pseudocode for the detected language processing and comparison approaches. The project applied Scrapy framework to construct spiders tasked with crawling websites and extracting information [34]. The scraped data is then stored in a CSV database. To determine similarity between search queries and dataset text, they employ vectorization of titles and utilize the Cosine Similarity algorithm [35]. Integration between Python and the website is facilitated through the Flask framework [36].

Algorithm 1 Dataset creation

```
1: for each E-commerce site do
2:   for each category in the site do
3:     for each product in the category do
4:       ProductDetails→Name,Price,link,Image
5:       Use CSS selector to find ProductDetails
6:       remove garbage data
7:       CSV dataset ← ProductDetails
8:     end for
9:   end for
10: end for
```

Figure 2.2.6 Algorithms for Dataset Creation [33]

Algorithm 2 Product search

```
1: InSent ← Product to be searched
2: T ← Minimal similarity threshold
3: DetectLanguage(InSent)
4: if detected language of the input is Bengali then
5:   Transliterate InSent to English as TSent
6: else
7:   Transliterate InSent to Bengali as TSent
8: end if
9: for each website in our data set do
10:  generate bag of words with TF-IDF
11:  for all products do
12:    PN ← product name
13:    if PN matches with InSent then
14:      put product details in ExactList
15:    end if
16:    if PN matches with TSent then
17:      put product details in ExactList
18:    end if
19:    if cosine similarity(PN, InSent) ≥ T then
20:      put product details in SimilarList
21:    end if
22:    if cosine similarity(PN, TSent) ≥ T then
23:      put product details in SimilarList
24:    end if
25:  end for
26: end for
27: Sort ExactList according to price in ascending order
28: Sort SimilarList according to similarity in descending
    order and price in ascending order
```

Figure 2.2.7 Algorithms for Related Searching Products [33]

Strength

The feature with transliteration allows users to type in both Bangla and English language whichever that is more familiar to them in the search bar for getting the products information. It will detect the language of the input and converted to English to scrape the related products from the website. They have also developed a custom exact match function from scratch to precisely extract the product that matches the user's search query. Products are sorted in descending order, prioritizing lower-priced items. The cosine similarity algorithm is performed with the Sci-kit learn library to determine the similarity between user input and database entries. Unlike traditional distance metrics, cosine similarity evaluates documents based on their orientation rather than magnitude. It calculates the cosine of the angle between two vectors, representing the documents in multidimensional space. This approach focuses on common words and their occurrence frequencies, effectively capturing similarities between documents of varying sizes and overcoming the limitation of zero matches.

Weakness

The project is facing limitations with generic indic-transliteration arise from the fundamental differences between the English and Bangla alphabets. Bangla's extensive character set, featuring numerous consonants, vowels, and modifiers, presents complexities for transliteration systems that aim to map Bangla characters to their approximate phonetic equivalents in the Latin script. Due to this character mismatch and the limitations of transliteration algorithms, achieving accurate romanization can be difficult. Complex Bangla characters, compound forms, and semantic ambiguities further compound the challenge, often resulting in incomplete or inaccurate transliterated output, particularly for less common or nuanced expressions.

2.2.4 Summary of Existing System

Size Measurement

Table 2.2.2 Comparison of previous work with current solution in size measurement

Works	Strengths	Weaknesses
[25]	- Many preprocessing steps like canny edge detection. Gaussian filter, edge gradient computation. Sobel operator, non-maximum suppression, hysteresis thresholding, morphological operation for defining edges of object from the images to get a more accurate object detection.	- Referenced object not easy to be detected and it is not a fixed dimension which requires measuring the referenced object every time when scanning the products. - Result affected by angle taking the images.
[26]	- Employing active contour model to find object of interest.	- Not applicable to real life application as it uses its own measuring scale scheme to compare different object.
[27]	- Higher accuracy of volume estimation for irregular shape of products	- Requires cross-section of object, slicing objects into many pieces
Proposed	- Referenced object of Aruco	- Result affected by angle taking the

solution	<p>Marker is easily detected with its known dimension and there is no need to need to measure it when scanning products every time.</p> <ul style="list-style-type: none"> - Measure using measuring unit of centimeter which is more applicable to consumer's daily usage. - Other required dimensions can be defined by user to get 3-dimension volume. - Fast estimated volume calculation using 3-dimension geometric shape formula compared the need to cut product into many slices. 	<p>images.</p> <ul style="list-style-type: none"> - Lower accuracy of volume estimation
----------	---	--

Product Recognition

Table 2.2.3 Comparison of previous work with current solution in product recognition

Works	Strengths	Weaknesses
[28], [30]	<ul style="list-style-type: none"> - Covering more category of retail products from market including products from daily life or household. 	<ul style="list-style-type: none"> - The saved model exported difficult to be loaded and used in mobile application. - Slower in getting result.
Proposed solution	<ul style="list-style-type: none"> - Saved model is easy to be loaded and used to retrieve result from image in mobile application with the usage of MobileNet. - Fast result returned. 	<ul style="list-style-type: none"> - Less category of product being covered as the focus is on fresh product.

Price Checker

Table 2.2.4 Comparison of previous work with current solution in price checker

Works	Strengths	Weaknesses
[33]	- transliteration feature allows users to type in both Bangla and English language	- Long time in getting output after search as need to process the query everytime
Proposed solution	- fast in getting output as used with realtime Firebase, RestAPI and automated job scheduler	- Long list of products may return as not exact match as in [33]

Chapter 3

System Methodology/Approach

3.1 Methodologies and General Work Procedures

The overall system developed will cover building mobile application using Python and Kivy, OpenCV for size measurement and object detection and deep learning model of product recognition, volume estimation, size grading, price retrieval from real-time database and web scraper and some CV displays on mobile phone. A trained machine learning model of product classification using CNN with TensorFlow and Keras framework will be used to calculating estimated 3D product volume according to similar products' geometric shape and grading the size categories for the product. Real-time database using Firebase will be built to store updated products' pricing information from the scheduled and automated web scraping and web crawling. RestAPI will be built for mobile application to retrieve data from Firebase. Finally, all the information of the products for its dimensions, size categories, estimated volume, price from different supermarkets will be displayed onto the application.

The expected output for this project is a mobile application in which consumers can use their smartphone to scan and measure the product from a 2D image to get its estimated volume in the unit of cm^3 . The volume will be calculated and displayed following their respective product shape according to the category detected from the product recognition machine learning model. Then, consumers can view the pricing information of the product across different supermarkets to compare the price and size.

3.2 System Design Equations

Size Measurement Equations

The equations used for the size measurement will be shown in this section.

The dimension for the product is obtained from the length of line drawn by user in pixel. Euclidean distance is used in this case, which the formula as below is applied:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

where,

(x_1, y_1) is the coordinate of the starting point of the line

(x_2, y_2) is the coordinate of the ending point of the line

The pixel to cm ratio is calculated to get the real measurement of the product using the referenced object of Aruco Marker:

$$\frac{\text{Aruco marker perimeter(pixel)}}{\text{Aruco marker perimeter (cm)}} = \frac{\text{Product width (pixel)}}{\text{Product width (cm)}}$$

The volume (cm³) covering at the current phase of project apply the formula from geometric shape resembling the product's shape. The formula for geometric shape:

Cylinder:

$$V = \pi r^2 h$$

Sphere:

$$V = \frac{4}{3} \pi r^3$$

Cube:

$$V = lwh$$

where,

V = volume of object

r = radius of object

h = height of object

l = length of object

w = width of object

3.3 System Design Diagrams

3.3.1 System Architecture Diagram

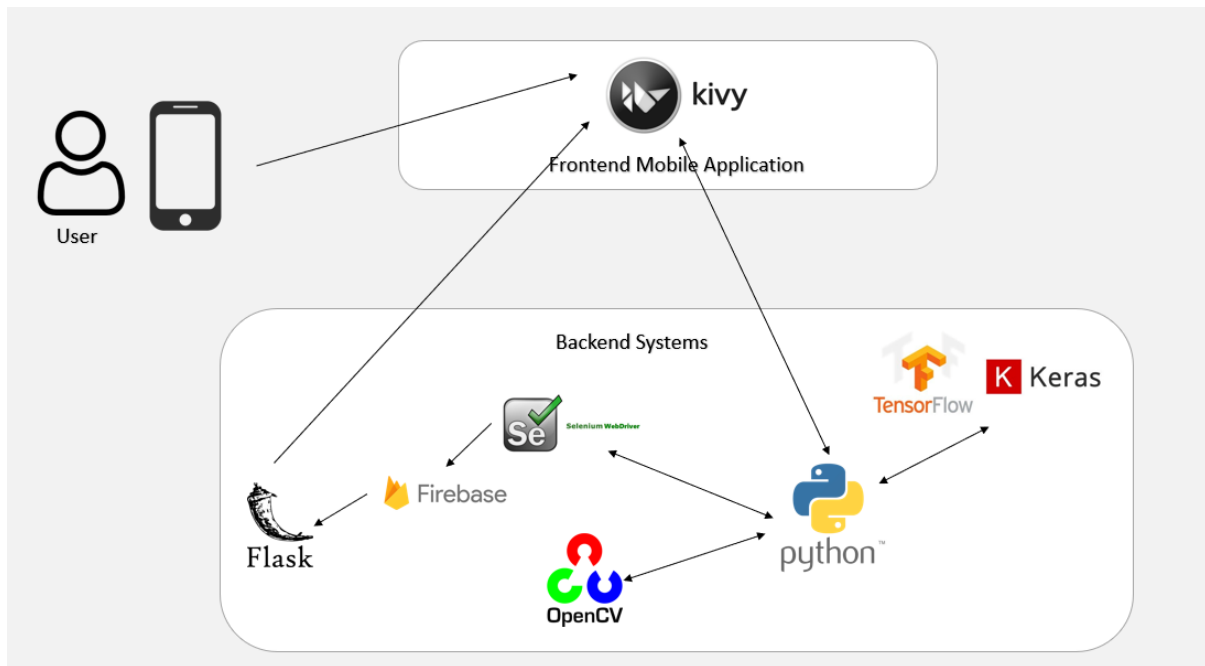


Figure 3.3.1 System Architecture Diagram

The technologies used in this project is Flask, Firebase, Selenium, OpenCV, TensorFlow Keras framework and Python programming language. The frontend for mobile application will be built using Kivy. Kivy is used for the easier implementation of backend like tflite runtime or OpenCV for Android package. It requires fewer steps for integration to fit into the Android compared to using other mobile application language like Flutter. Firebase is used due to its ability to meet the demand of real-time data synchronization. Microsoft Azure is employed for hosting Flask server in the cloud rather than hosting locally.

3.2.2 Use Case Diagram and Description

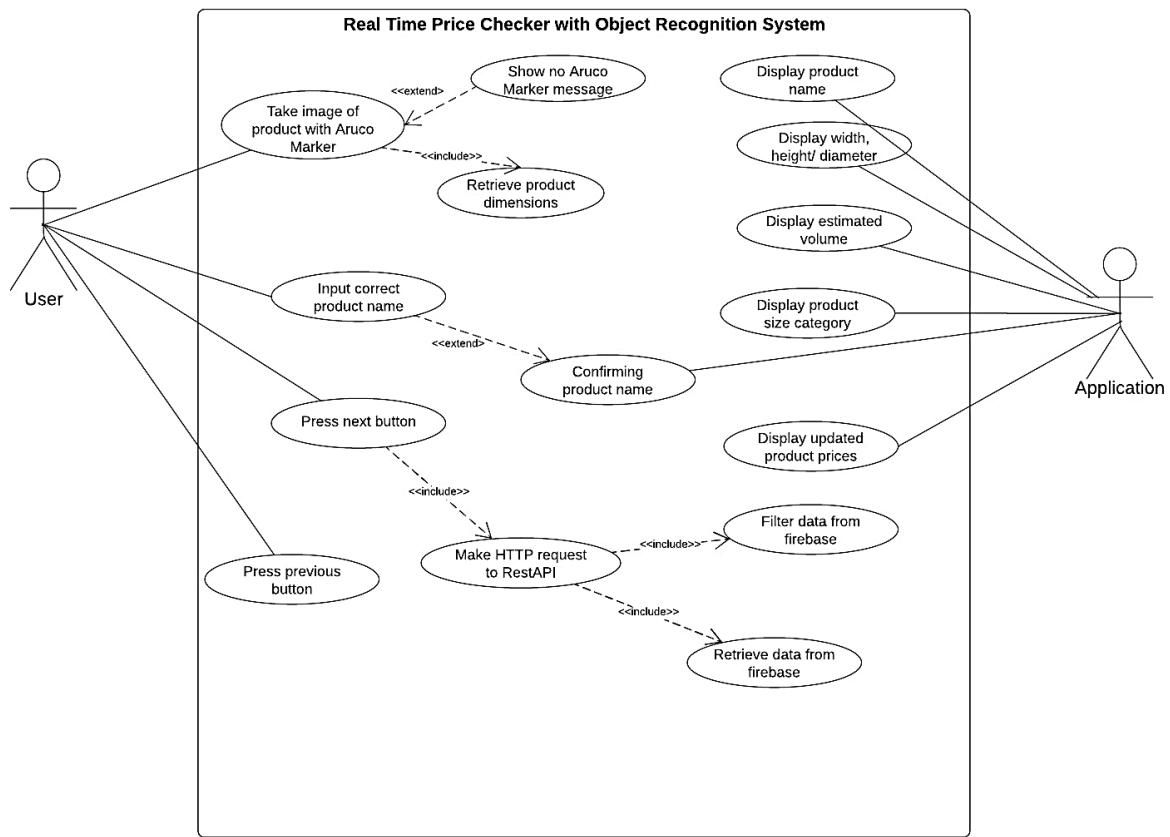


Figure 3.2.2 Use Case Diagram

Use Case ID	00001
Use Case Name	Mobile Application
Brief Description	This use case describes how system process images from users and display the desired output to application.
Actor	User, Application
Trigger	When user open the application on their mobile devices
Precondition	User must have the Aruco Marker printed with 5cm x 5cm
Normal Flow of Events	<ol style="list-style-type: none"> 1. User takes product image with Aruco Marker 2. System validates existence of Aruco Marker in image 3. System validates product's name from image 4. Application displays product name, estimated volume, size category, dimensions of product in OpenCV image 5. User press next button

	<p>6. System makes HTTP request to RestAPI</p> <p>7. System retrieve data from Firebase</p> <p>8. System filter data into respective categories</p> <p>6. Application displays related products name and price lists under different supermarkets for the product.</p> <p>7. User press previous button to view back first screen result.</p>
Sub Flows	
Alternative Flows	<p>2a. The system shows error popup message “No Aruco Marker detected” and allow user to view and close.</p> <p>3a. The system shows confirmation popup message “Is the product name: xxx?” and allow user to reinput product name if the product name recognized is incorrect.</p>

3.2.3 Activity Diagram

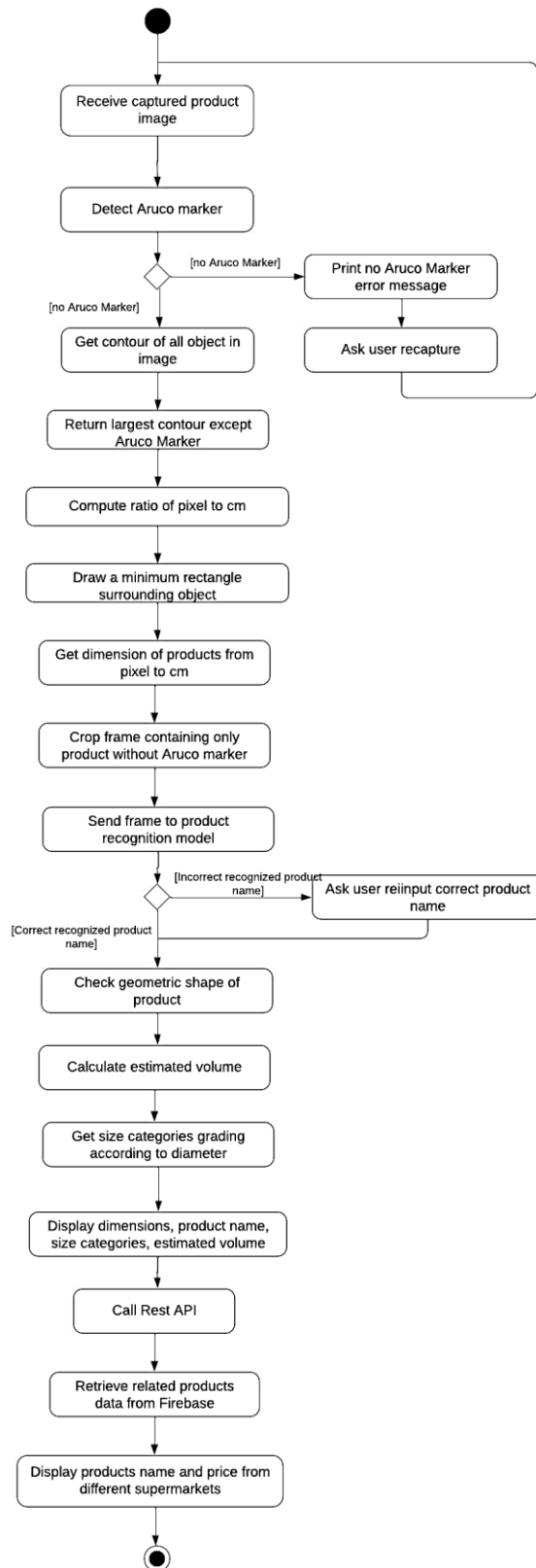


Figure 3.2.3 Activity Diagram

The activity diagram shows the brief process of how application handles images from users and process it to get the desired output on the screen.

Chapter 4

System Setup

4.1 Hardware Used

The hardware involved in the implementation of this project is a laptop and an Android mobile phone device. A computer will be used to collect and store dataset from various sources, train and save machine learning model for product classification and recognition, set up real-time database to store pricing information, doing web scraping and web crawling, and building mobile application with OpenCV. A mobile device is needed to test the usability and visibility for the deployment of mobile application from user's perspective.

Table 4.1 Specifications of Laptop

Description	Specifications
Model	HP Pavilion Laptop 15-eg2xxx
Processor	12th Gen Intel(R) Core(TM) i5-1240P
Operating System	Windows 11
Graphic	Intel(R) Iris(R) Xe Graphics
Memory	12GB DDR4 RAM
Storage	512GB SATA HDD

Table 4.2 Specifications of Android Mobile Phone

Description	Specifications
Model	Samsung Galaxy S22 Ultra
Processor	Qualcomm Snapdragon 8 Gen 2
Operating System	Android 12
Graphic	Adreno 730 GPU
Memory	16GB LPDDR5 RAM
Storage	512GB UFS 3.1 Flash Storage

4.2 Software Setup

Before starting to develop the system, there are a few softwares/platforms required to be installed and downloaded in the laptop:

1. Pycharm
2. Jupyter Notebook
3. Chrome
4. Ubuntu

4.2.1 Pycharm



Figure 4.2.1 Pycharm Logo

It eases the individual development for each project as an isolated virtual Python environment can be created. It helps to prevent conflict between different versions of library packages and ensure the reproducibility across systems on different Python platforms. Its intelligent code completion suggests relevant snippets, variable names, and function names, enhancing productivity and minimizing errors. The integrated debugger facilitates efficient issue identification and resolution through breakpoint setting, variable inspection, and step-by-step code traversal. With the built-in code analysis tools, PyCharm helps identify code imperfections and suggests enhancements, ultimately contributing to the creation of polished and efficient Python codebases.

4.2.2 Jupyter Notebook



Figure 4.2.2 Jupyter Logo

Suitable for machine learning project for building and finetuning deep learning model for product recognition model with the features offered such as interactive development, debugging tools, and visualization capabilities, making it easier to prototype, experiment, and iterate on models. It provides an interactive environment to write and execute code in a step-by-step manner, allowing for easy experimentation and prototyping of different approaches. Its integration with Python libraries such as TensorFlow, PyTorch, and Keras enables seamless development and testing of neural network architectures. Overall, Jupyter Notebook

enhances the development process by providing a flexible, interactive, and collaborative environment for building deep learning models.

4.2.3 Chrome



Figure 4.2.3 Chrome Logo

Chrome is used for browsing and scraping the contents from the grocery stores websites due to its popularity and extensions helper tools like XPath Helper.

4.2.4 Ubuntu



Figure 4.2.4 Ubuntu Software Logo

To create a package for android from Kivy program (mainly APK files) as Kivy cannot be directly used in Android Phone, Buildozer will be implemented to automate the packaging process. As Buildozer currently supported on Linux system, to use it in Windows, Ubuntu act as Linux distribution will be used to execute the Linux commands from Buildozer. Ubuntu can be just downloaded from Microsoft Store.

4.3 Tools requirements

1. Chrome Web Driver
2. Browser mob proxy
3. Android Debug Bridge (adb)
4. OpenCV Android SDK

4.3.1 Chrome Web Driver

Due to the wide domination of Chrome as the commonly used web browser globally, Chrome Web Driver is considered for running the test script from Selenium WebDriver. To interact with the Chrome from Selenium, it requires a separatable executable which is the Chrome Web Driver. Check the version of Google Chrome from settings as shown in Figure 3.2 and

download the respective version for the Chrome Web Driver that fit with the Chrome browser version from the link:

<https://chromedriver.chromium.org/downloads>.

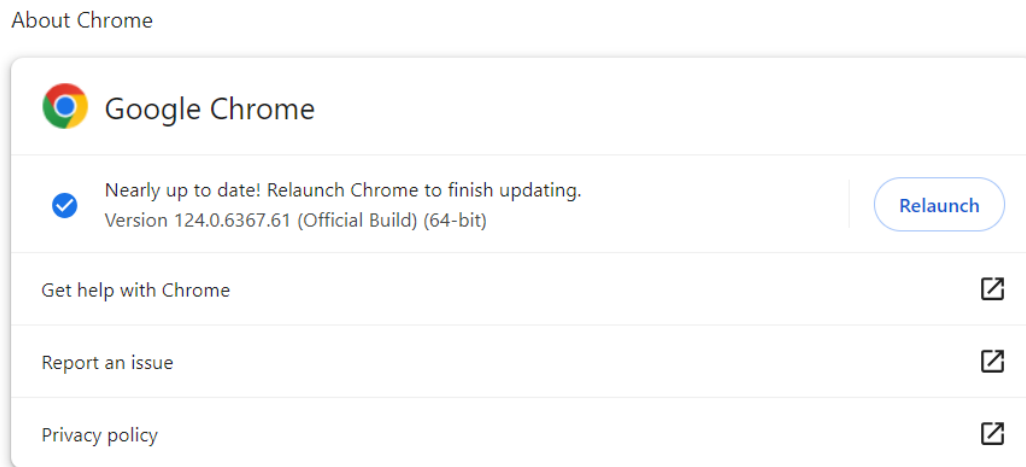


Figure 4.3.1 Chrome Version

4.3.2 BrowserMob Proxy

It serves as a proxy that mask the real source of where the browsing and scraping is coming from to prevent the real IP address from actual laptop from being blocked. It acts as intermediary between the browser and the internet, allowing for the interception and manipulation of HTTP requests and responses. The proxy used in this project can be retrieved from: <https://github.com/lightbody/browsermob-proxy>.

4.3.3 Android Debug Bridge (adb)

It helps to debug on Android Phone, prompting error message from phone to laptop cmd terminal screen. Download SDK Platform-Tools for Windows from the link:

<https://developer.android.com/tools/releases/platform-tools#downloads>.

Downloads

If you're an Android developer, you should get the latest SDK Platform-Tools from Android Studio's [SDK Manager](#) or from the [sdkmanager](#) command-line tool. This ensures the tools are saved to the right place with the rest of your Android SDK tools and easily updated.

But if you want just these command-line tools, use the following links:

- [Download SDK Platform-Tools for Windows](#)
- [Download SDK Platform-Tools for Mac](#)
- [Download SDK Platform-Tools for Linux](#)

Figure 4.3.2 SDK Platform-Tools Page

Unzip the zip folders and check the adb version by typing “./adb –version”.

4.3.4 OpenCV Android SDK

To use OpenCV in mobile phone, the Android SDK package need to be downloaded. Download it from the link: <https://opencv.org/releases/> with the 3.4.5 version. Extract the zip files, copy only the folders under native (C:\Users\name\Downloads\opencv-3.4.5-android-sdk.zip\OpenCV-android-sdk\sdk\native) and paste it to the Kivy mobile application program project directory.

Chapter 5

System Design/ Implementation

5.1 System Block Diagram

Figure 5.1 shows the main modules for the project. Mobile application built using Kivy will utilize the product recognition model, and call to the Firebase using RestAPI. The data stored in Firebase will be retrieved from Selenium web scraper that runs automatically using Window Task Scheduler at a regular interval time being set.

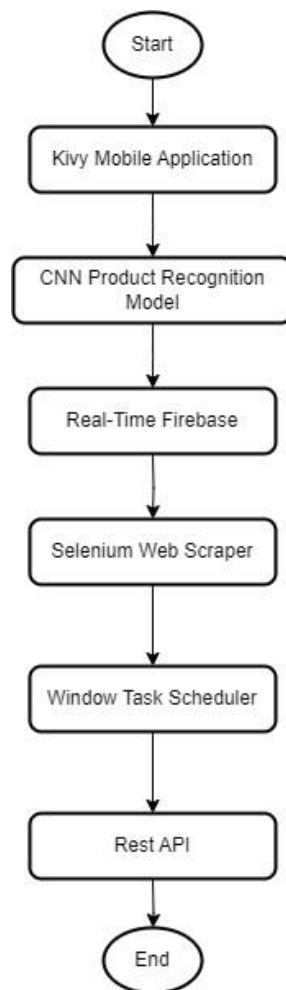


Figure 5.1 Main System Components

5.2 Kivy Mobile Application

Before installing Kivy using “pip install kivy”, ensure that Python and pip are preinstalled. Start to build the application using any preferred Python IDLE, in this project,

Pycharm will be used. The code for the Kivy application built in this project is available in the link: <https://github.com/Zeckszan/dealwithITappFYP> (with installed opencv package, adb, product recognition tflite model and buildozer files). Install all the required packages from requirement.txt using the command “pip install -r requirements. txt” to use the functions in mobile application.

5.2.1 OpenCV Object Detection

Preprocessing of frames will first be done to blur and convert it to gray scale in order to reduce the background noises. Then, adaptive thresholding on a grayscale image is performed to produce a binary image where regions of interest are highlighted based on local intensity variations. The resulting mask image will contain binary values representing foreground (objects of interest) and background pixel. After a list of objects contour is found in the binary image, it will be returned to be used by Kivy program to extract product from images taken by user. Only the contours that are long or contours area that are large enough will be considered and the rest will be discarded.

5.2.2 OpenCV Size Measurement

A printed squared Aruco Marker with known dimension of 5cm x 5cm composed by 25 units and perimeter of 20cm as shown in Figure 5.2.1 will act as reference point for the calibration of the camera. The critical part of pixel to cm ratio is computed by checking how many pixels for the perimeter of Aruco Marker on screen corresponds to 20cm perimeter of Aruco Marker in reality. After the product from the image is detected from the largest contour, it will be drawn with a minimum rectangle bounding box. The width and height of the object in pixel are obtained from the bounding box drawn. The computed pixel to cm ratio will then be applied to convert the dimensions from pixel to cm. Figure 5.2.2 shows how the Aruco Marker is being configured on the Kivy mobile application using python.

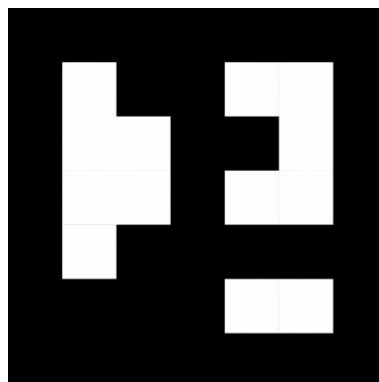


Figure 5.2.1 Aruco Marker with 5cm x 5cm dimension

```
# Load Aruco detector
parameters = cv2.aruco.DetectorParameters_create()
aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_5X5_50)

# Load Object Detector
detector = HomogeneousBgDetector()

# Get Aruco marker
corners, _, _ = cv2.aruco.detectMarkers(frame, aruco_dict, parameters=parameters)
```

Figure 5.2.2 Configuration for Getting Aruco Marker

5.2.3 Loading Product Recognition Model and Estimating Volume

The product recognition model is loaded and used in the mobile application referencing the model.py from the link: https://github.com/macrodriques/butterfly_detector/tree/master. The image captured from user will undergo the same preprocessing as in product recognition model built such as resizing into the target size of (224,224). Figure 5.2.3 shows how to load the model and get the prediction label. Figure 5.2.4 shows how the function for loading model, preprocessing images and prediction for the output is done.

```
#load the tflite model
model = 'GrocRecogModel.tflite'
model_to_pred = TensorFlowModel()
model_to_pred.load(os.path.join(os.getcwd(), model))

# Read image and predict
img = PIL.Image.open(os.path.join(os.getcwd(), SAVE_PATH))
img_arr = img.resize((224, 224))
img_arr = np.array(img_arr, np.float32)
img_arr = img_arr[:, :, :3] / 255.0
img_arr = np.expand_dims(img_arr, axis= [0])
answer = model_to_pred.pred(img_arr)
y_class = answer.argmax(axis=-1)
ylabel = " ".join(str(x) for x in y_class)
ylabel = int(ylabel)
```

Figure 5.2.3 Loading Product Recognition Model and Predicting

```

import tensorflow as tf

2 usages
class TensorFlowModel:
    1 usage
    def load(self, model_filename, num_threads=None):
        self.interpreter = tf.lite.Interpreter(model_filename,
                                                num_threads=num_threads)
        self.interpreter.allocate_tensors()

    def resize_input(self, shape):
        if list(self.get_input_shape()) != shape:
            self.interpreter.resize_tensor_input(0, shape)
            self.interpreter.allocate_tensors()

1 usage
    def get_input_shape(self):
        return self.interpreter.get_input_details()[0]['shape']

1 usage
    def pred(self, x):
        # assumes one input and one output for now
        self.interpreter.set_tensor(
            self.interpreter.get_input_details()[0]['index'], x)
        self.interpreter.invoke()
        return self.interpreter.get_tensor(
            self.interpreter.get_output_details()[0]['index'])

```

Figure 5.2.4 Functions for Loading Model, Preprocessing and Predicting

5.2.4 Estimated 3D Volume

After the output for the product name is successfully predicted by the model, a confirmation message will be prompted to confirm the correct product name with the user. The geometric shape of the product is identified based on the product name, the estimated volume of the product will then be calculated using the formula covered in section 3.2 according to their shape like sphere or cylinder by importing math library. Figure 5.2.4 shows how the estimated volume is calculated.

```

if self.res in sphereproduct:
    object_volume = (4 / 3) * math.pi * (object_width / 2) ** 2
    self.shape="spherical"
else: # cylinder
    object_volume = object_height * math.pi * (object_width / 2) ** 2
    self.shape = "elongated"

```

Figure 5.2.5 Calculation of Estimated Volume

5.2.5 Size Grading Categories

The Figure 5.2.5 shows the guideline of diameter used for identifying size scale of fruit and vegetable from the user image (in cm). The sources for the diameter ranges are based on a combination of common industry practices, general knowledge of fruit and vegetable grading standards, and insights gained from agricultural research and publications particularly USDA (link: <https://www.ams.usda.gov/grades-standards>).

```
size_categories = {  
    'Small': {'spherical': (5.08, 6.35), 'elongated': (1.27, 2.54)},  
    'Medium': {'spherical': (6.35, 7.62), 'elongated': (2.54, 3.81)},  
    'Large': {'spherical': (7.62, 8.89), 'elongated': (3.81, 5.08)},  
    'Extra-Large or Jumbo': {'spherical': (8.89, float('inf')), 'elongated': (5.08, float('inf'))}  
}
```

Figure 5.2.6 Guideline Size Grading based on Diameter

5.2.6 Display Related Product Latest Price

The data for product's name, product's size category, width and height with the OpenCV image result will then be displayed in mobile application by Kivy.

5.2.7 Deploying Kivy Application

Buildozer is needed to package Kivy program into the APK files so that it can be run on Android mobile devices. As Buildozer currently only supported on Linux system, to use it in Windows, a Virtual Machine called Window Subsystem for Linux (WSL) is needed to run the code.

To set up the environment:

Setting up windows subsystem for Linux, a Linux terminal to run from Windows system.

from the document link: <https://learn.microsoft.com/en-us/windows/wsl/install-manual>

Steps 1: open Windows Power Shell in administrator mode and run the command “dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart”

Step 2: run the command

“dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart”

Step 3: restart laptop

Step 4: download Linux kernel update package as shown in Figure 5.2.7.

Step 4 - Download the Linux kernel update package

The Linux kernel update package installs the most recent version of the [WSL 2 Linux kernel](#) for running WSL inside the Windows operating system image. (To run [WSL from the Microsoft Store](#), with more frequently pushed updates, use `wsl.exe --install` or `wsl.exe --update`.)

1. Download the latest package:

- [WSL2 Linux kernel update package for x64 machines](#)

Figure 5.2.7 Page for download Linux kernel package

Step 5: run the command “`wsl --set-default-version 2`”

Step 7: Open downloaded Ubuntu from section 4.2.4 and set up username and password

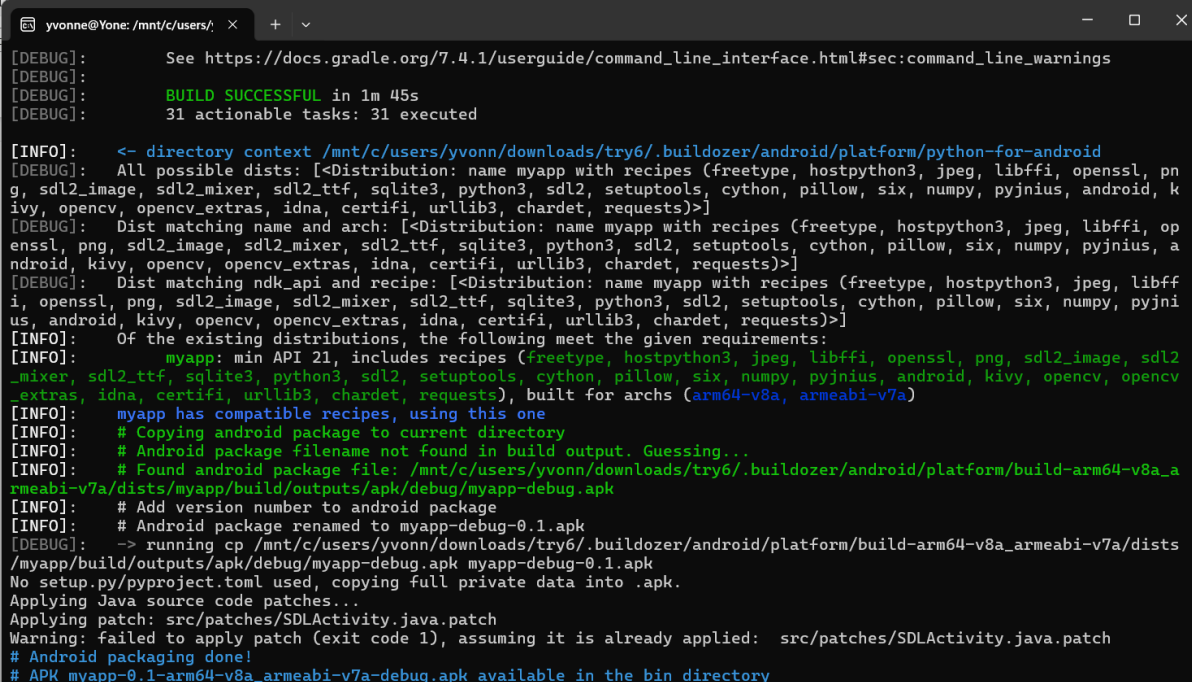
Step 8: navigate to the directory where mobile application project files is saved in Window, example command to use: `cd /mnt/c/Users/yourname/filesdirectoryOfmobileapplication`

Step 9: install buildozer guided by link:

<https://buildozer.readthedocs.io/en/latest/installation.html>

Step 10: run “`sudo apt update`” to install pip3

The buildozer is now ready to be used. Get the mobile application files from section 5.3 github link and run the command “`buildozer -v android debug`” to start package the application. The successful packaging output is as shown in Figure.



```
[DEBUG]: See https://docs.gradle.org/7.4.1/userguide/command_line_interface.html#sec:command_line_warnings
[DEBUG]:
[DEBUG]: BUILD SUCCESSFUL in 1m 45s
[DEBUG]: 31 actionable tasks: 31 executed

[INFO]: <- directory context /mnt/c/users/yvonn/downloads/try6/.buildozer/android/platform/python-for-android
[DEBUG]: All possible dists: [<Distribution: name myapp with recipes (freetype, hostpython3, jpeg, libffi, openssl, png, sdl2_image, sdl2_mixer, sdl2_ttf, sqlite3, python3, sdl2, setuptools, cython, pillow, six, numpy, pyjnius, android, kivy, opencv, opencv_extras, idna, certifi, urllib3, chardet, requests)>]
[DEBUG]: Dist matching name and arch: [<Distribution: name myapp with recipes (freetype, hostpython3, jpeg, libffi, openssl, png, sdl2_image, sdl2_mixer, sdl2_ttf, sqlite3, python3, sdl2, setuptools, cython, pillow, six, numpy, pyjnius, android, kivy, opencv, opencv_extras, idna, certifi, urllib3, chardet, requests)>]
[DEBUG]: Dist matching ndk_api and recipe: [<Distribution: name myapp with recipes (freetype, hostpython3, jpeg, libffi, openssl, png, sdl2_image, sdl2_mixer, sdl2_ttf, sqlite3, python3, sdl2, setuptools, cython, pillow, six, numpy, pyjnius, android, kivy, opencv, opencv_extras, idna, certifi, urllib3, chardet, requests)>]
[INFO]: Of the existing distributions, the following meet the given requirements:
[INFO]: myapp: min API 21, includes recipes (freetype, hostpython3, jpeg, libffi, openssl, png, sdl2_image, sdl2_mixer, sdl2_ttf, sqlite3, python3, sdl2, setuptools, cython, pillow, six, numpy, pyjnius, android, kivy, opencv, opencv_extras, idna, certifi, urllib3, chardet, requests), built for archs (arm64-v8a, armeabi-v7a)
[INFO]: myapp has compatible recipes, using this one
[INFO]: # Copying android package to current directory
[INFO]: # Android package filename not found in build output. Guessing...
[INFO]: # Found android package file: /mnt/c/users/yvonn/downloads/try6/.buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/myapp/build/outputs/apk/debug/myapp-debug.apk
[INFO]: # Add version number to android package
[INFO]: # Android package renamed to myapp-debug-0.1.apk
[DEBUG]: -> running cp /mnt/c/users/yvonn/downloads/try6/.buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/myapp/build/outputs/apk/debug/myapp-debug.apk myapp-debug-0.1.apk
No setup.py/pyproject.toml used, copying full private data into .apk.
Applying Java source code patches...
Applying patch: src/patches/SDLActivity.java.patch
Warning: failed to apply patch (exit code 1), assuming it is already applied: src/patches/SDLActivity.java.patch
# Android packaging done!
# APK myapp-0.1-arm64-v8a_armeabi-v7a-debug.apk available in the bin directory
```

Figure 5.2.8 Successful Packaging Kivy Program Output

5.3 CNN Product Recognition Model

A deep learning module of CNN with TensorFlow and Keras framework is used to train on Fruits and Vegetables Image Recognition dataset discovered from Kaggle, a public dataset covering the products of fruits and vegetables with the link: <https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition> to recognize product scanned by user on the mobile application. The dataset consists 3 folders, train, validation and test images. The data will be resize into the target size of (224,224) with image augmentation of rotation, zooming, flipping extending the ImageDataGenerator. Pretrained model of MobileNetV2 is used with the weight of imagenet and average pooling layers. The final model with highest validation accuracy score will then be saved in the format GrocRecogModel.tflite so that it can be used in mobile application. The model is trained based on the reference:

https://github.com/Spidy20/Fruit_Vegetable_Recognition/tree/master?tab=readme-ov-file

The work can be found on: <https://github.com/Zeckszan/productRecogModelFYP>

```
tf.keras.models.save_model(model, 'model.pbtxt')
converter = tf.lite.TFLiteConverter.from_keras_model(model=model)
model_tflite = converter.convert()
open("GrocRecogModel.tflite", "wb").write(model_tflite)
INFO:tensorflow:Assets written to: model.pbtxt\assets
INFO:tensorflow:Assets written to: model.pbtxt\assets
INFO:tensorflow:Assets written to: C:\Users\yvonn\AppData\Local\Temp\tmpv52f4wri\assets
INFO:tensorflow:Assets written to: C:\Users\yvonn\AppData\Local\Temp\tmpv52f4wri\assets
9599668
```

Figure 5.3.1 Code Snippet for Saving Model in tflite format

5.4 Real-Time Firebase

Create a new project in Firebase with the any preferred name and location. Register the web app to retrieve the configuration details to get access into the Firebase so that the creating, updating or reading operation can be done from other programs as shown in Figure 5.5.2. After the registration is done firebaseConfig is recorded.

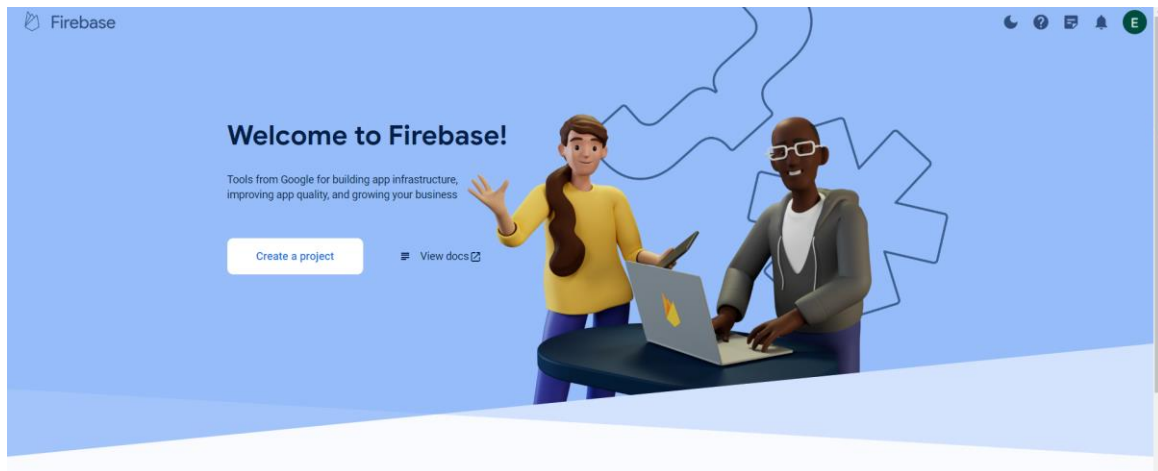


Figure 5.4.1 Pages for Creating a Firebase Project

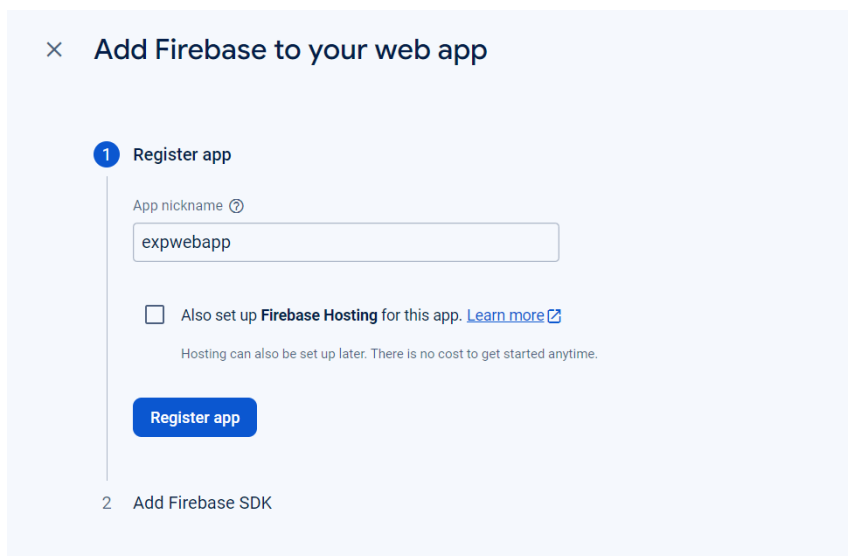


Figure 5.4.2 Pages for Registering Firebase as a Web App

Use npm Use a <script> tag

If you're already using [npm](#) and a module bundler such as [webpack](#) or [Rollup](#), you can run the following command to install the latest SDK ([Learn more](#)):

```
$ npm install firebase
```

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```

// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDIKsJdVbvVQ83xdQdWfs2Vf_piVmrogng",
  authDomain: "expfirebasefyp.firebaseio.com",
  projectId: "expfirebasefyp",
  storageBucket: "expfirebasefyp.appspot.com",
  messagingSenderId: "71181374281",
  appId: "1:71181374281:web:9b268c67dff51ea9728098",
  measurementId: "G-6HX57S21E9"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
  
```

Note: This option uses the [modular JavaScript SDK](#), which provides reduced SDK size.

Figure 5.4.3 Important Firebase Config

Create a realtime database as shown in Figure 5.5.4 and choose start in test mode so that other external program can access and modify the database as shown in Figure 5.5.5.

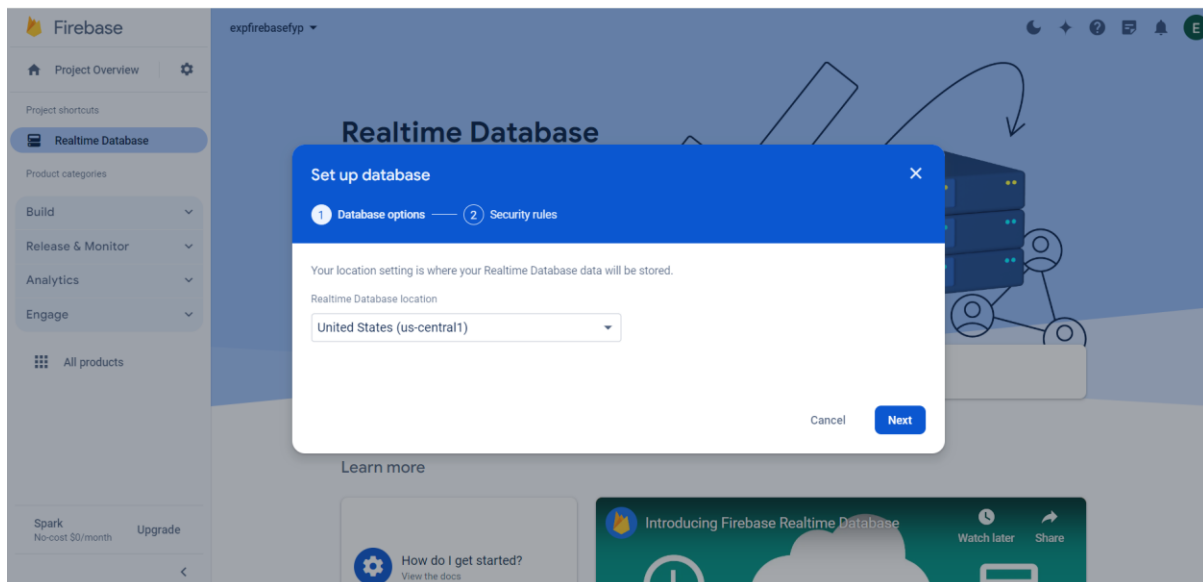


Figure 5.4.4 Pages for setting up realtime database

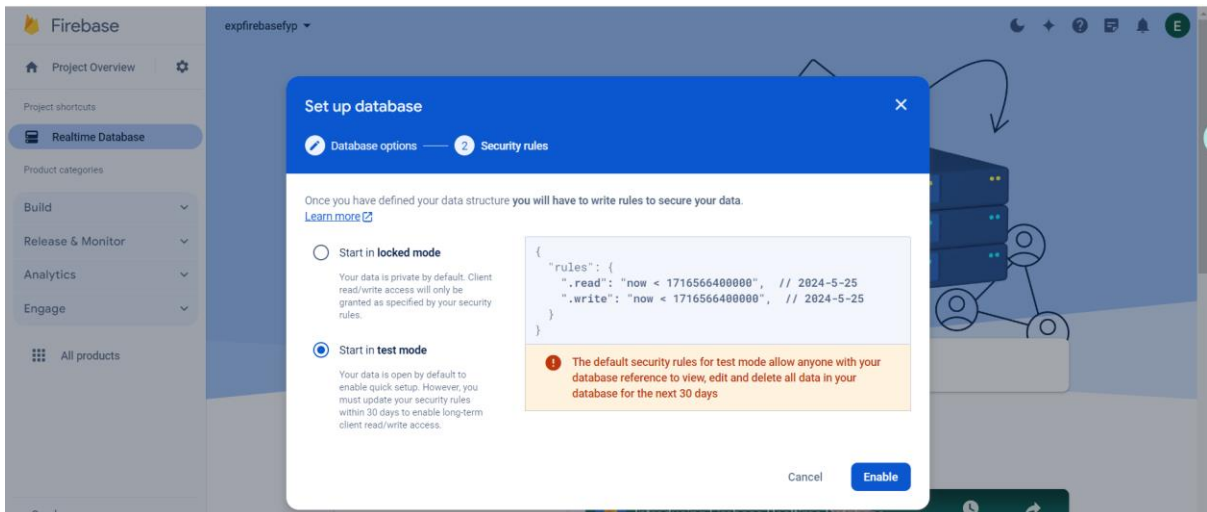


Figure 5.4.5 Pages for Firebase rules configuration

Copy the realtime database URL and add to the configuration code needed by other external program to access the database.



Figure 5.4.6 Pages for Realtime database URL

5.5 Selenium Web Scraper

The updated product information from different supermarkets are collected from the respective website through Selenium web scraping to be stored into the Firebase. After downloading the web driver and browser mob proxy based on the sections 4.3.1 and 4.3.2, unzip them, copy the chromedriver.exe and browser mob proxy folders to the directory near the C drive (or any drive). Under Pycharm, install selenium package by using “pip install selenium”. Initialize the settings for browser mob proxy as in Figure 5.5.1.

```
server = Server(path="C:/browsermob-proxy-2.1.4/bin/browsermob-proxy.bat", options={'port': 8090})
server.start()
proxy = server.create_proxy()
```

Figure 5.5.1 Initializing Browser-mob Proxy

Configure the settings for chrome web browser using the settings as shown in Figure 5.5.2.

```

chrome_options = Options()
chrome_options.add_argument('--disable-gpu')
chrome_options.add_argument('--blink-settings=imagesEnabled=false')
chrome_options.add_argument('--disable-notifications')
chrome_options.add_argument('--ignore-certificate-errors')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument("--proxy-server={0}".format(proxy.proxy)) # Add this line to configure the proxy
chrome_options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome(CHROME_PATH, options=chrome_options)
driver.maximize_window()
driver.get(url)

```

Figure 5.5.2 Configuration of Chrome Web Browser

Several settings are done to improve performance like “disable-gpu” helps to reduce resource consumption, “blink-settings=imagesEnabled=false” enables web page to load faster without image, “disable-notifications” disable browser notifications. After navigating to the web page link, any pop up messages will be closed by imitating human behavior in feeding the input to the webpage like typing the postcode “50450” and click the verify button as shown in Figure 5.5.3.

```

try:
    postcode = WebDriverWait(driver, timeout=10).until(
        EC.presence_of_element_located((By.XPATH, '//input[@class="cpk-input"]'))
    )
    postcode.send_keys("50450")
    time.sleep(5)

    verify_btn = WebDriverWait(driver, timeout=10).until(
        EC.presence_of_element_located(
            (By.XPATH, '//*[@id="postcode-popup-inner"]/div[3]/div'))
    )
    verify_btn.click()
except Exception as e:
    print("Error closing pop up ",e)
pass

```

Figure 5.5.3 Closing Up Pop-up Message

The total number of available pages to scrape is discovered and recorded in advance, before looping through every page to scrape the content. CSS selectors or XPath for product name and price are identified through inspecting the HTML element of the webpage. The data will be preprocess to do the standardization for proper price formatting like adding the unit ‘RM’ in front of the price and normalizing the product name into a consistent format like using title() from Python to capitalize each first letter from the product name. After list of product name and price is retrieved from the grocery website, get access into the Firebase that have been set up and push all the data inside as shown in Figure 5.5.4. The products’ data are then channelled to the Firebase to be stored and maintained.


```

firebase = pyrebase.initialize_app(config)
database = firebase.database()
database.child("jaya grocer").set(result)

```

Figure 5.5.4 Firebase Initialization and Pushing Data

The work for the web scraper can be obtained from the link:

<https://github.com/Zeckszan/scrapeGroceryFYP>

5.6 Window Task Scheduler

Task scheduler from Window will be used to schedule the task of scraping the related product name and price from different grocery stores websites regularly. Open the task scheduler, click “Create Task...” and fill in required information for task name as shown in Figure 5.6.1 (optional for description)

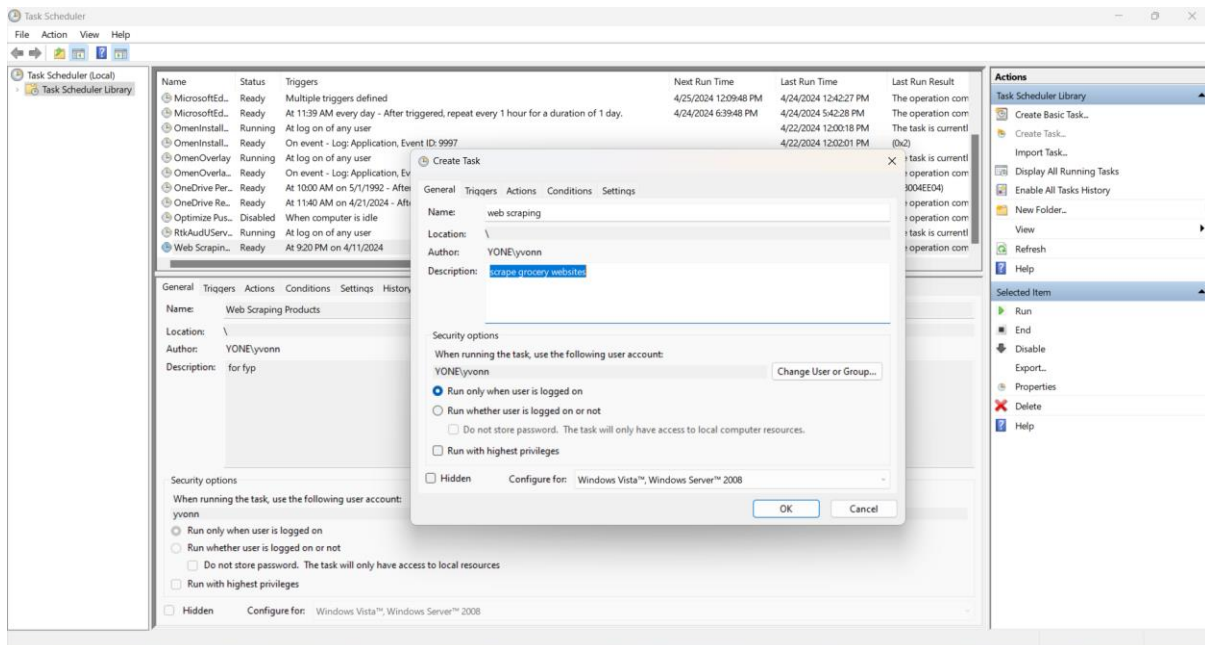


Figure 5.6.1 Creating Basic Task

Click “Triggers” and “New”, schedule the task based on the preferred starting time and settings (like daily, weekly, monthly). The settings used in this project is as shown in Figure 5.6.2.

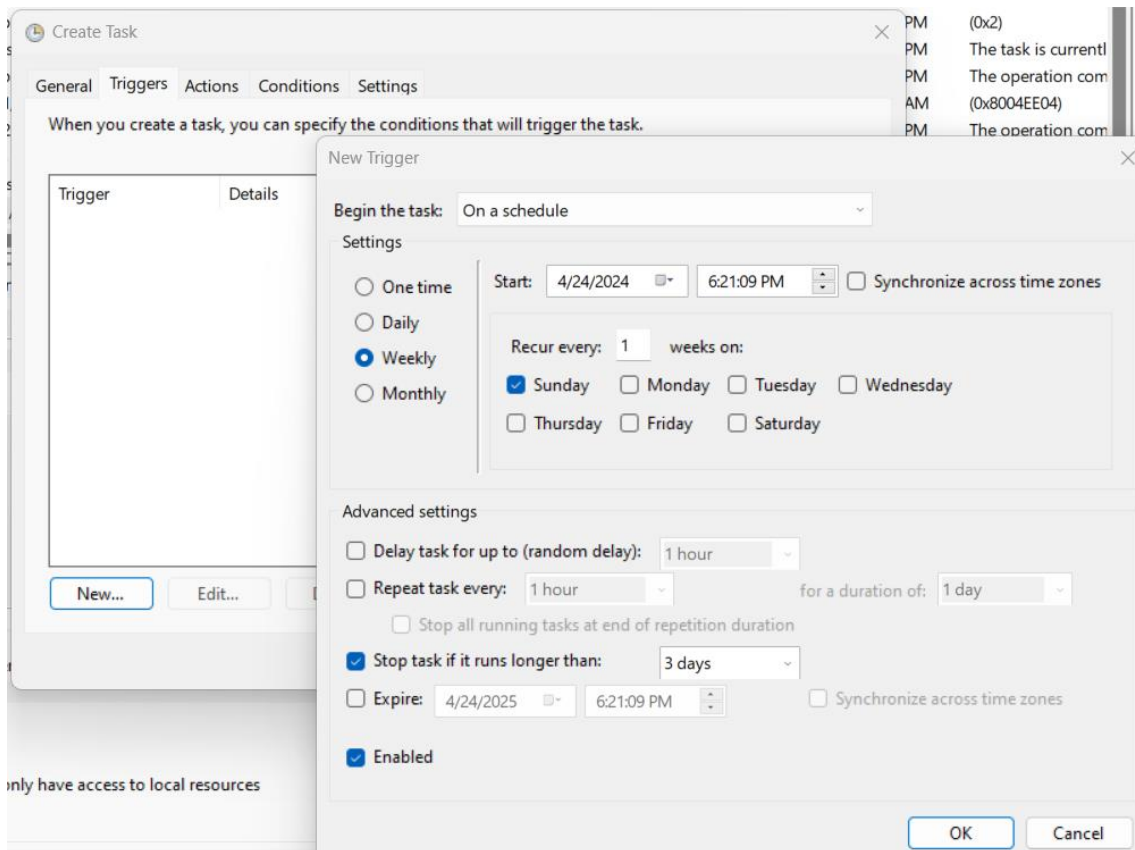


Figure 5.6.2 Settings for Window Task Scheduler

Create a bash file to the directory of where web scraper program is stored command to activate the virtual environment. The bash file created in this project is as shown in Figure 5.6.3. Click “Action” and fill in directory to where the bash file is stored as shown in Figure 5.6.4.

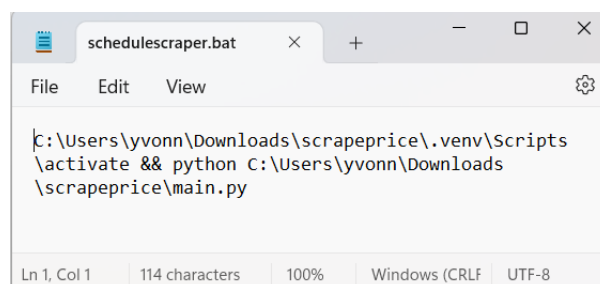


Figure 5.6.3 Bash File

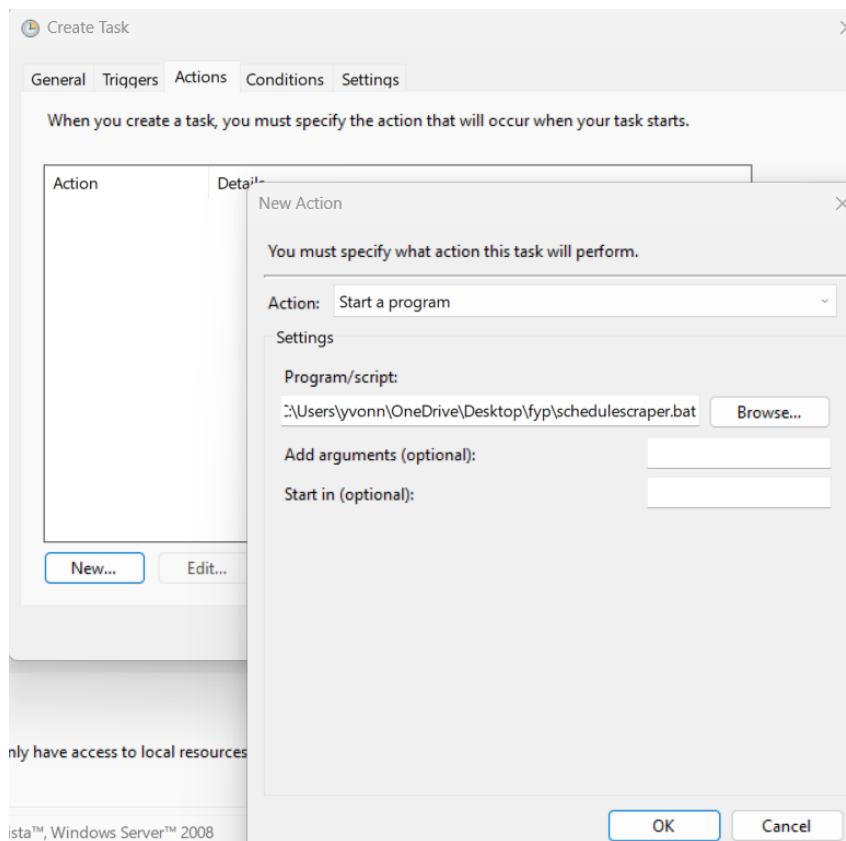


Figure 5.6.4 Command for Program/script

5.7 RestAPI

It is built using Python, Flask and Pyrebase4 packages to connect between the database, Firebase and the mobile application from Kivy. The application will call URL from the Flask server hosted in Microsoft Azure to make HTTP request to this API to retrieve the related product name and prices from different supermarkets and display them on the application. The program is built by first getting access into the Firebase web app with the configuration code as shown in Figure 5.7.1 and retrieving the data from Firebase as shown in Figure 5.7.2.

```

config = {
    "apiKey": "AIzaSyCvrfjVrBgcobvJgYcE_Yda70rN3xbBhoQ",
    "authDomain": "pricescrape2.firebaseio.com",
    "projectId": "pricescrape2",
    "databaseURL": "https://pricescrape2-default-rtdb.firebaseio.com/",
    "storageBucket": "pricescrape2.appspot.com",
    "messagingSenderId": "83119362702",
    "appId": "1:83119362702:web:b89a01e53e3dbd3f202fdf",
    "measurementId": "G-Z27T8MY96"
}
firebase = pyrebase.initialize_app(config)
database = firebase.database()

```

Figure 5.7.1 Firebase configuration API code

```

dataJaya = database.child("jaya grocer").get()
dataVillage = database.child("village grocer").get()
dataLotus = database.child("lotus").get()

dataJaya = dataJaya.val()
dataVillage = dataVillage.val()
dataLotus = dataLotus.val()

```

Figure 5.7.2 Retrieving Firebase Data Code

The data retrieved from Firebase will then be filtered using re package based on product name recognized from the user captured image from the mobile devices when the URL “https://apifyf.azurewebsites.net/get/<productname>” with the parameter <productname > is called. Figure 3.2 shows how the data is filtered and formatted to pass back the result to the caller the mobile application. The Flask will then be hosted into a web application using cloud platform called Microsoft Azure so that it can be accessed by any mobile device at any location.

```

listJaya = ['Jaya Grocer']
pattern = re.compile(r'\b{}\b'.format(re.escape(productname)), re.IGNORECASE)

listJaya.extend([(product['product_name'], product['price']) for product in dataJaya
                 if re.search(pattern, product['product_name'])])

listVillage = ['Village Grocer']
listVillage.extend([(product['product_name'], product['price']) for product in dataVillage
                   if re.search(pattern, product['product_name'])])

productname = english_to_malay[productname]
listLotus = ['Lotus']
listLotus.extend([(product['product_name'], product['price']) for product in dataLotus
                 if re.search(pattern, product['product_name'])])

return listJaya+listVillage+listLotus

```

Figure 5.7.3 Filter Firebase data Code

The Flask server hosted in Microsoft Azure and GitHub is available in the link:

<https://github.com/Zeckszan/fypflask>

5.8 System Operation

5.8.1 Mobile Application User Interface

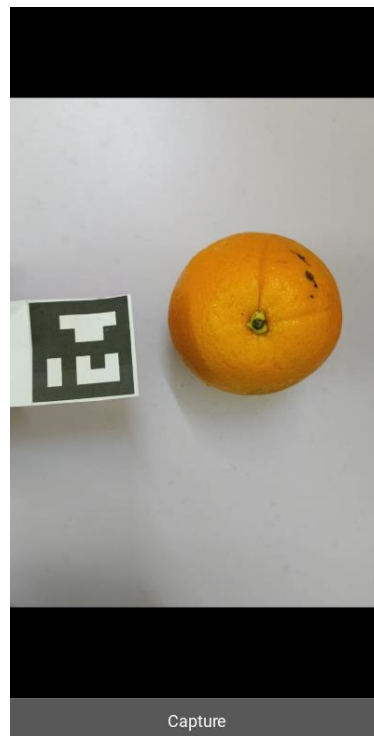


Figure 5.8.1 User capturing photo

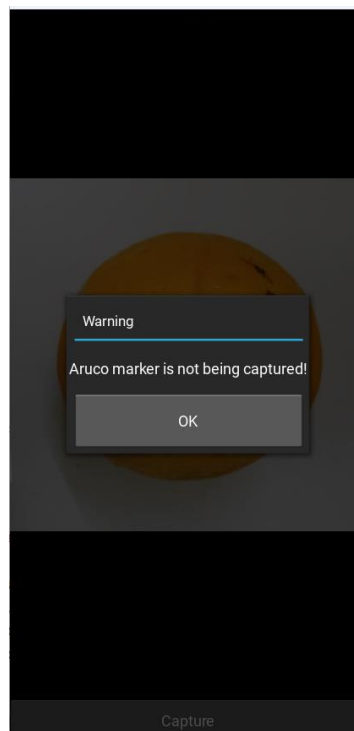


Figure 5.8.2 Warning Message that Aruco Marker not detected

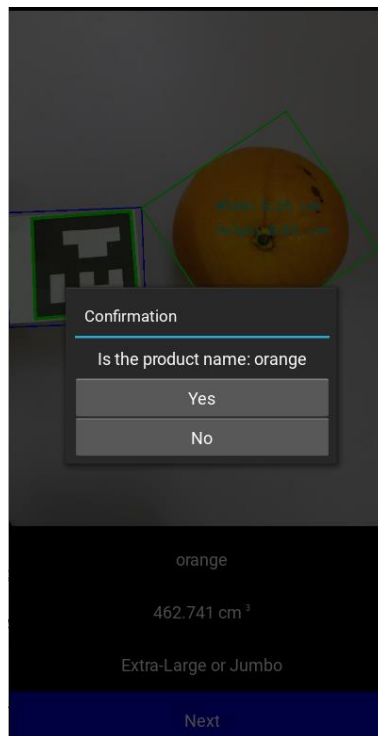


Figure 5.8.3 Confirmation Message on Correct Product Name

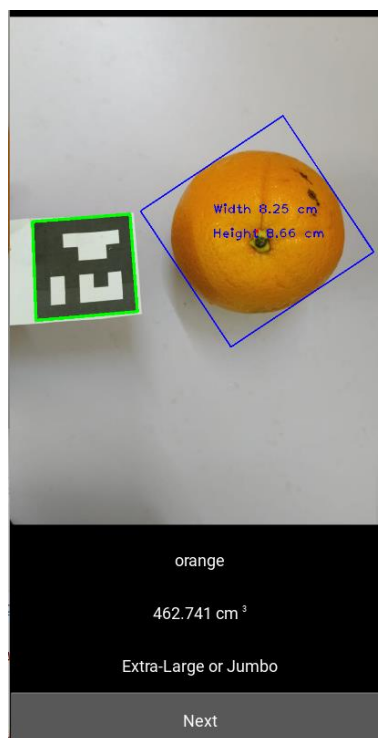


Figure 5.8.4 Result for Object Detected, Dimensions, Product Name, Estimated Volume, Size Scale

Jaya Grocer		
	Valencia Orange 8pcs/pack	RM 12.90
	Navel Orange (USA) 8pcs/pack	RM 26.90
	Sunkist Navel Orange (USA) 1 unit	RM 5
Product Name: orange	Organic Valencia Orange (Egypt) 4pcs/pack	RM 11.90
	Orange Navel (Egypt) 8pcs/pack	RM 21.88
	Orange Navel (Egypt) 3pcs/pack	RM 7
	JMG Mikan Mandarin Orange (Japan) 3kg	RM 88.88
	Orange Sweet Potato (Australia) 1kg	RM 8.45

Figure 5.8.5 Scraped Price Result with Related Product Name for Different Supermarkets

5.8.2 Scraped Data on Firebase

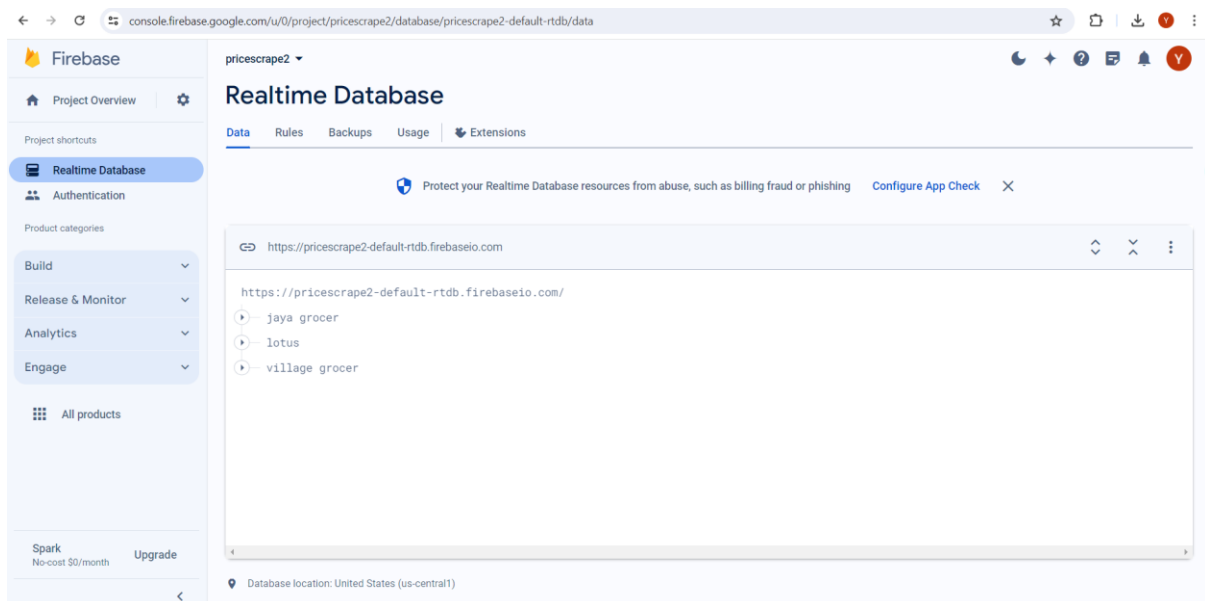


Figure 5.8.6 Successful Scraped Supermarket Name

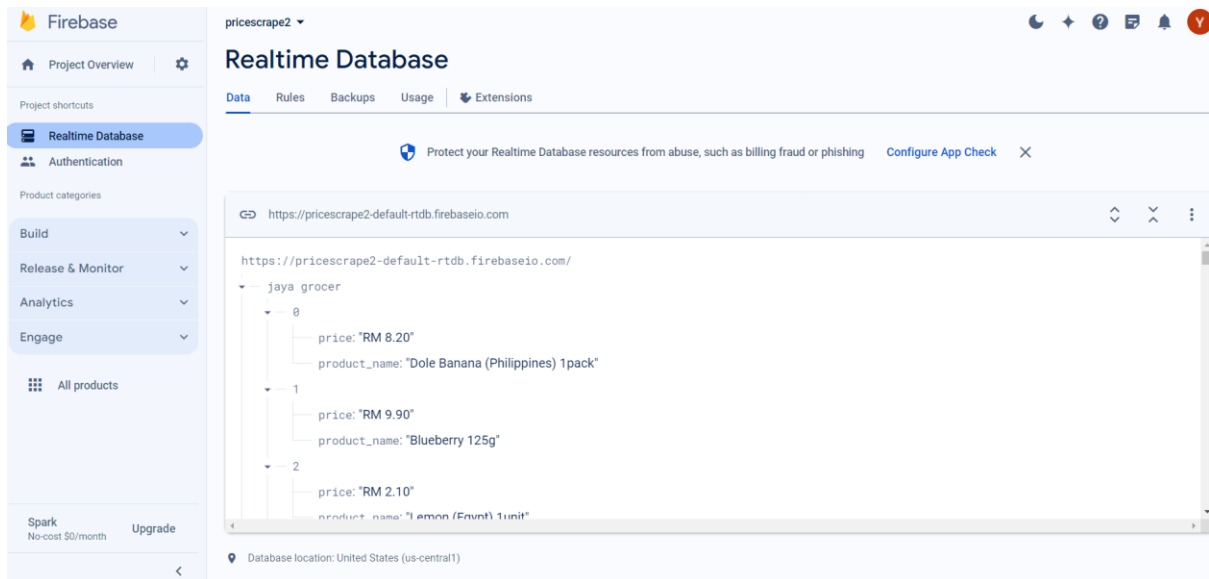


Figure 5.8.7 Formatted Product Name and Price on Firebase

5.8.3 Rest API Data

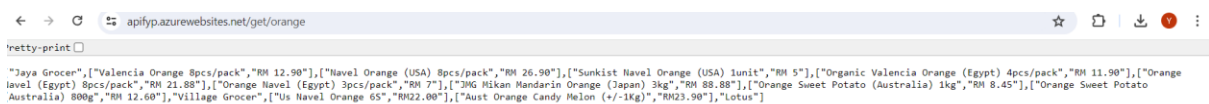


Figure 5.8.8 Filtered Retrieved Data from Firebase

5.9 Implementation Issues and Challenges

In terms of mobile application development, the most difficult part is when packaging Kivy code into APK files to work on mobile devices due to constant updating of recipes from Buildozer. It especially hard in incorporating machine learning models from product recognition model to Kivy as the documentation is not comprehensive enough on what libraries should be included in application requirements under specification of buildozer files (buildozer.spec) and what machine learning model file format (like .tflite, .h5 or .hdf5) it accept or reject. Also, it doesn't support all external libraries or modules as those work in original Python packages like web scraping libraries of Selenium, scientific computing libraries of NumPy or database libraries of Pyrebase. All the modules that require these libraries need to do separately and integrate with Kivy using other tools like RestAPI. Next, as Kivy is not having documentation and community support as big as other frameworks like Kotlin, it makes it more challenging to use as often the bug need to trace from the source through all layers which takes a long time. It also requires a long hour of packaging (typically more than 20 minutes), before the deployment and debugging can be done on mobile devices.

With different level of security being set by different grocery stores websites, different number of measures are required to bypass those protection. There are some websites with advanced bot detection or javascript detection to check if the browser is being controlled by automation tools like Selenium. They employ techniques to detect the absence of human-like interaction, such as mouse movements or keyboard inputs, and block the access accordingly when they detect any abnormalities. There are also certain websites that cannot be scraped due to undergoing maintenance and will only be back after a quarter or even half a year. Figure 5.9.1 and Figure 5.9.2 shows the hindrances encountered when scraping the grocery stores websites.

You have been blocked.

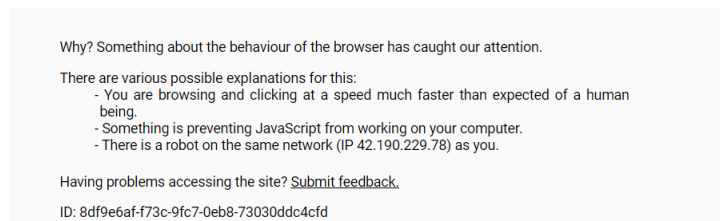


Figure 5.9.1 Blocked from Grocery Stores Website



Figure 5.9.2 Grocery Stores Websites Maintenance

5.10 Concluding Remarks

The buildup of the project involves many modules and proper integration is needed to connect them together so that the application can be used to its fullest usages. Proper environment and tools should be set up with compatible version of packages to build the system.

The novelty of this project is to measure size of a product from 2D image to get its estimated volume so that it can be easily compared with product from other supermarkets using figure. The volume is calculated based on the standard product's shape like eclipse and cylinder based on their category. The pricing details will be updated regularly to optimize its usefulness and prevent outdated information.

Chapter 6

System Testing And Evaluation

6.1 System Testing and Performance Metrics

The product recognition model should achieve a high test accuracy of above 95% in detecting products from the categories of fruits and vegetables so as the subsequent information for pricing, sizing and product geometric shape will not be retrieved wrongly. The model shall work under different conditions like brightness or degree of taking image. It should be able to detect the product even with some background noises like there are things other than the product in the image. Different type of fruits will be tested with different sizing to ensure the application work as intended. The estimated volume should be change according to the dimensions detected.

The targeted plan for size measurement is with low error, floating around the acceptable range of 1-2 cm compared to its actual real-life measurement. The dimension's value should be remained at a certain range of value even the background or the distance from camera change. Whether it is moving close or farther away from camera, it should be able to detect the Aruco Marker and keep the correct measurement for the product. The measurement for the product should be remained even after its distance from the camera has changed.

In terms of OpenCV object detection, it should be able to return the contour for the largest object detected from the image, ignoring the small objects/background noises or with some shadow.

6.2 Testing Result

Figure 6.2.1 shows the 3 example of products that will be tested for their recognized product name, dimensions measurement and graded size categories. 3 of them are different in terms of their size categories which the size scale of each product is expected to be graded as small, medium and big respectively. The system should also be able to identify the product as apple or orange. Besides, they will be tested with the application to see if they can get the correct sizing and dimensions with the acceptable error range.



Figure 6.2.1 Examples of Products to be Tested

6.2.1 Size Scale, Dimensions and Product Recognition at a Constant Distance

Figure 6.2.2 shows the output get from the application, it can be seen that the size categories for the respective product is retrieved correctly which are Small, Medium and Large based on their diameter and spherical shape. The estimated volume is also varying according to their dimensions. Figure 6.2.3 and Figure 6.2.4 demonstrates the rough real measurement for the width and height of Large orange product to compare with the measurement retrieved from the application.

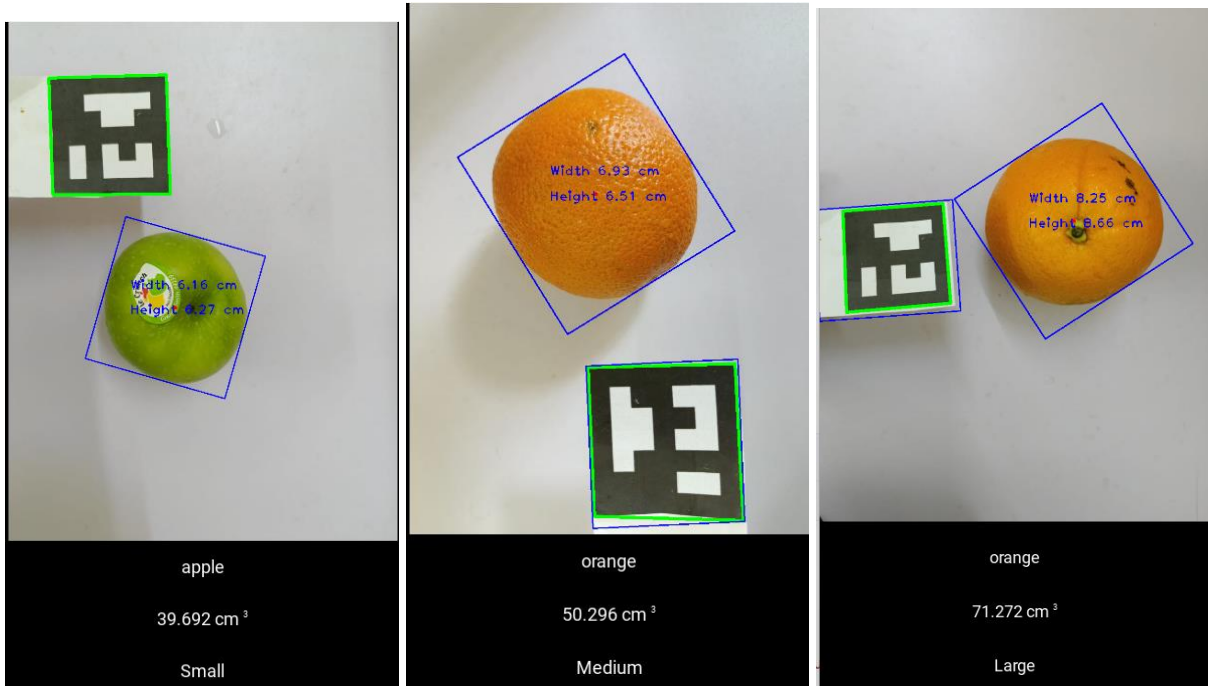


Figure 6.2.2 Result for Small Apple, Medium Orange, Large Orange



Figure 6.2.3 Real Measurement of Width for Large Orange



Figure 6.2.4 Real Measurement of Height for Large Orange

Table 6.2.1 Comparison of Size Dimension Between Application and Real Life

	Width (cm)	Height (cm)
Application	8.25	8.66
Real life	8.3	9.2

6.2.2 Size Scale, Dimensions and Product Recognition at Different Distance

Figure 6.2.5 shows that even at varies distant the product from the camera, the size dimensions obtained is still kept at a similar dimensions range without much deviations from each other. The size scale detected is still Large for all 3 images of near or distant product from camera.

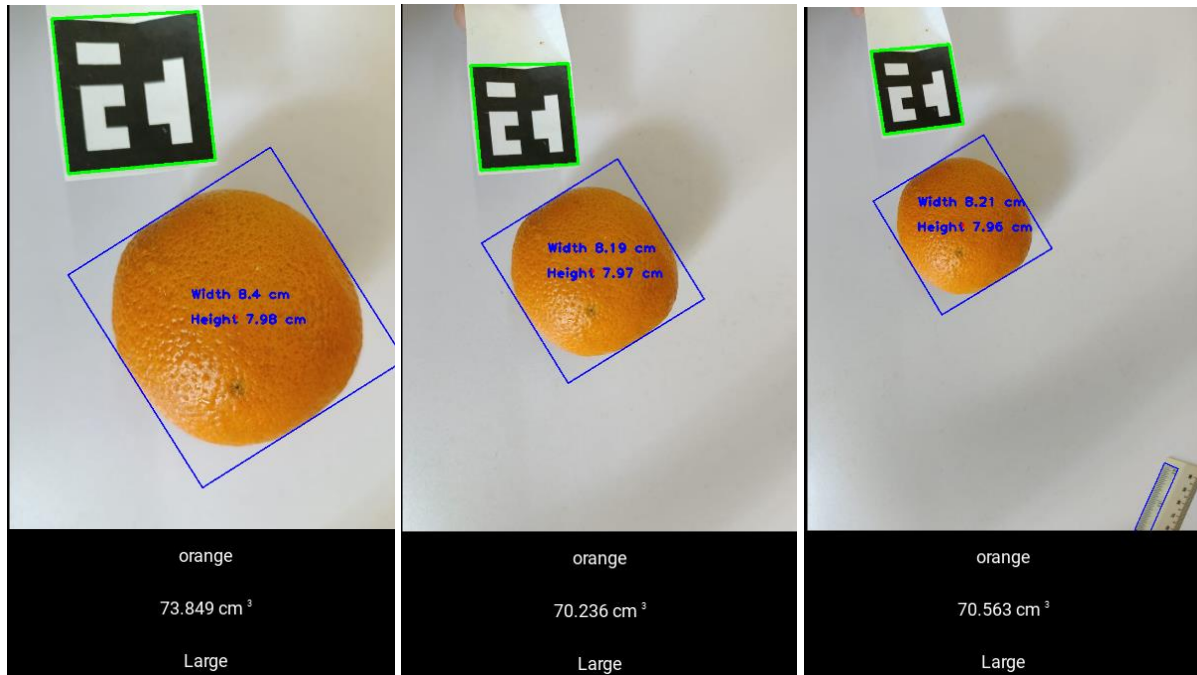


Figure 6.2.5 Size Measurement for Large Orange at Different Distance from Camera

6.2.3 Test Accuracy for Products Recognition Model

Using package from Tensorflow the accuracy for test set (from the same Kaggle dataset) achieved is up to 97.37%

```
import tensorflow as tf
test_loss, test_accuracy = model.evaluate(test_images, verbose=1)

print(f'Test accuracy: {test_accuracy}')
```

11/11 [=====] - 30s 3s/step - loss: 0.1496 - accuracy: 0.9737
 Test accuracy: 0.9736841917037964

Figure 6.2.6 Test Accuracy of Products Recognition Model

6.3 Project Challenges

Difficult check alignment on mobile devices as it requires the program to be packaged every time when there is a new update. It is good for implementing complex backend like OpenCV on the mobile application but bad at providing necessary libraries to align the layout properly on the phone with different screen sizes. There is less frontend library that can be make use of to align and beautify the mobile design, for example Kivy require manually binding text to its widget so that if the text is too long it can be multiline or jump to the next row instead of using function like Wrap class from Flutter.

Difficult in integrating OpenCV with other mobile application languages, before kivy is used, Flutter and Kotlin have been tried but it is difficult to include or link to the OpenCV packages SDK for Android mainly due to the gradle incompatibility or unknown unsupported versions. It poses a great issue in finding package for the implementation of modules with suitable version that is compatible with current platform version as some might use a deprecated Android embedding that will cause the failure in launching the application from Android phone.

Next, the product name and price web scraper will not work for website with only embedded images with no textual information on html like only having the brochure to view.

6.4 Objectives Evaluation

A mobile application is successfully created for Android user (with the packaged APK file) with the all-in-one features for true size measurement of width and height regardless of the distance based on object detection model, estimated 3D volume from 2D product image based on the approximate geometric shape and product recognition model, size category grading with the criteria from current common industry practice and real time price viewing from different supermarkets with real time firebase and automated web scraper are combined to reduce the user's hassle in finding all those information.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In conclusion, consumers nowadays have the problems of wasting large amount of time and energy from the big amount of data from the Internet in comparing the product size and pricing across various supermarkets at many locations. The product's price maybe varies depending on the retailer even the stocks are bought from the same manufacturer who spend same amount of cost in producing the goods. Some of the consumers get mislead by the fake advertisement promoted by the retail shops which turns out that they need to pay higher than they expect. As there is a lack of universal size scaling system, it is causing confusion as consumers can only compare the products' volume through their eye. Food wastage occur when they purchase a size that does not match with their prediction of proportion. With that, it is essential to build a system that can address the issues faced by consumers to enhance their shopping experience with the aids from technology. The goal for the system is to eliminate consumers' frustration encountered in numerous sizing and pricing discrepancies of the same product at the current and rest of the supermarkets and empower them to make informed purchasing decision. To ease the usage of the system by consumers at any place and at any time, mobile application incorporating object detection, product recognition, size measurement, volume estimation, real time database for updated pricing data will be built. It improves the consumers' visibility in doing an easy comparison of products by showing all the needed information in one screen. The correct data retrieval on application relied heavily on how accurate the product is detected and recognized from the images taken by users.

Besides, through the regular updating feature of the system from job scheduler and realtime database, users will be benefited from the system in terms of economic wise as they can notice how much fluctuation on the price of the goods in the market under some duration. Perhaps they can be the first few people who aware the early sign of inflation tide by understanding the market trends. By making use of the system, they may be able to get the cheapest price compared to other consumers by observing the price changes over some period or a certain season. Also, the size measurement can help consumers in not only making comparison between supermarkets but also with the old product in their house. In that way, consumers can get back their preferred size of the product that aligns with their consumption

capacity before the expiry date to remain the freshness. Hence, a more sustainable life is created when wastage is reduced.

7.2 Recommendation

It is targeted that in the near future the project will be more developed and advanced by looking into more detail in calibrating the camera so that the reference object of Aruco Marker is not needed. The field of AR will be investigated to know computation in converting pixel to real size unit like cm or m from the depth estimation of product to the camera.

REFERENCES

- [1] T. H. Davenport, L. D'Almeida and J. Patil. "Know What Your Customers Want Before They Do" *Harvard Business Review* (accessed Jun. 6, 2023).
- [2] Chandon, P. "Customers Aren't Very Good at Judging Product Sizing" in *Harvard Business Review*, Nov. 2015. [Online]. Available: <https://hbr.org/2015/11/customers-arent-very-good-at-judging-product-sizing>
- [3] N. Dawar. "The rich, the lazy, the busy, the ignorant, and the vain" INSEAD Knowledge. <https://knowledge.insead.edu/marketing/rich-lazy-busy-ignorant-and-vain> (accessed June. 5, 2023).
- [4] E. Gill, "Shoppers says they're paying over shelf price as supermarkets 'can't keep up'" M.E.N Media. <https://www.manchestereveningnews.co.uk/whats-on/shopping/checkout-price-higher-shelf-supermarket-24401023> (accessed Jun. 6, 2023).
- [5] T. Wells. "Pricing Tricks You Never Knew: Do They Really Encourage Shoppers to Buy More?" Wtaylor Mwells. <https://taylorwells.com.au/pricing-tricks/> (accessed June. 5, 2023).
- [6] Pavalam S. M., S. V. Kasmir Raja, Jawahar M., and Felix K. Akorli, "Web Crawler in Mobile Systems", *International Journal of Machine Learning and Computing*, Vol. 2, No. 4, August 2012.
- [7] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development- a study." *International Journal of Computer Applications* 179.46 (2018): 49-53.
- [8] Udupure, Trupti V., Ravindra D. Kale, and Rajesh C. Dharmik. "Study of web crawler and its different types." *IOSR Journal of Computer Engineering* 16.1 (2014): 01-05.
- [9] S.S. Dhenakaran¹ and K. Thirugnana Sambanthan², "WEB CRAWLER - AN OVERVIEW", *International Journal of Computer Science and Communication* Vol. 2, No. 1, January-June 2011, pp. 265-267.
- [10] Md. Abu Kausar, V. S. Dhaka, Sanjeev Kumar Singh, "Web Crawler: A Review", *International Journal of Computer Applications* (0975 – 8887), 2013, Volume 63– No.2.
- [11] Shashi Shekhar, Rohit Agrawal and Karm Veer Arya, "An Architectural Framework of a Crawler for Retrieving Highly Relevant Web Documents by Filtering Replicated Web Collections", 2010 International Conference on Advances in Computer Engineering, IEEE Conference Publications 2010.

- [12] Swati Mali and B.B. Meshram, "Implementation of Multiuser Personal Web Crawler", CSI Sixth International Conference on Software Engineering (CONSEG), IEEE Conference Publications, 2012.
- [13] Junghoo Cho and Hector Garcia-Molina. "The evolution of the web and implications for an incremental crawler", In Proceedings of the 26th International Conference on Very Large Databases, 2000.
- [14] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In ECCV, 2014.
- [15] Gupta, Satinder Bal. "The issues and challenges with the web crawlers." International Journal of Information Technology & Systems 1.1 (2012): 1-10.
- [16] Junghoo Cho and Hector Garcia-Molina. "The evolution of the web and implications for an incremental crawler", 2000, In Proceedings of the 26th International Conference on Very Large Databases.
- [17] Shi, Zejian, Minyong Shi, and Weiguo Lin. "The implementation of crawling news page based on incremental web crawler." In 2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD), pp. 348-351. IEEE, 2016.
- [18] Niraj Singhal, Ashutosh Dixit, and Dr. A. K. Sharma, "Design of a Priority Based Frequency Regulated Incremental Crawler", International Journal of Computer Applications (0975 – 8887) vol.1, no. 1, 2010, pp. 42-47.
- [19] Gupta, Satinder Bal. "The issues and challenges with the web crawlers." International Journal of Information Technology & Systems 1.1 (2012): 1-10.
- [20] Khedkar, Sonam, S. Thube, W. I. Estate, and C. Naka. "Real time databases for applications." *International Research Journal of Engineering and Technology (IRJET)* 4, no. 06 (2017): 2078-2082.
- [21] Shkapenyuk, Vladislav, and Torsten Suel. "Design and implementation of a high-performance distributed web crawler." *Proceedings 18th International Conference on Data Engineering*. IEEE, 2002.
- [22] D, S, Sirisuriya. (2015). A comparative study on web scraping.
- [23] M, Yerukala. "What is Selenium? - Selenium Automation Testing Introduction" MindMajix. <https://mindmajix.com/what-is-selenium> (accessed March. 22, 2024)

- [24] “What is Kivy” javaTpoint. <https://www.javatpoint.com/kivy> (accessed March. 23, 2024)
- [25] Othman, N.A., Salur, M.U., Karakose, M. and Aydin, I., “An embedded real-time object detection and measurement of its size” in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 2018, pp. 1-4.
- [26] Pu L, Tian R, Wu HC, Yan K., “Novel object-size measurement using the digital camera” in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Oct, 2016, pp. 543-54
- [27] Concha-Meyer, A., Eifert, J., Wang, H. and Sanglay, G., “Volume estimation of strawberries, mushrooms, and tomatoes with a machine vision system.” *International Journal of Food Properties*, 21(1), pp.1867-1874.
- [28] W, Yuchen, S, Tran, S, Xu, B, Kang, and M, Springer. "Deep learning for retail product recognition: Challenges and techniques." *Computational intelligence and neuroscience* 2020, 2020.
- [29] Z. Q. Zhao, P. Zheng, S. t. Xu, and X. Wu, “Object detection with deep learning: a review,” *IEEE Transactions on Neural Networks and Learning Systems*, 2019. vol. 30, no. 11, pp. 3212– 3232.
- [30] Indira, D. N. V. S. L. S., Goddu, J., Indraja, B., Challa, V. M. L., & Manasa, B. A review on fruit recognition and feature evaluation using CNN. *Materials Today: Proceedings*, 2023, 80, pp. 3438-3443.
- [31] Z. Q. Zhao, P. Zheng, S. t. Xu, and X. Wu, “Object detection with deep learning: a review,” *IEEE Transactions on Neural Networks and Learning Systems*, 2019, vol. 30, no. 11, pp. 3212– 3232.
- [32] W. Liu, D. Anguelov, D. Erhan et al., “SSD: single shot multibox detector,” in *Proceedings of the 2016 European Conference on Computer Vision, Amsterdam, Netherlands, 2016*, pp. 21–37.
- [33] Alam, A., Anjum, A. A., Tasin, F. S., Reyad, M. R., Sinthee, S. A., & Hossain, N. (2020, June). Upoma: A dynamic online price comparison tool for bangladeshi e-commerce websites. In *2020 IEEE Region 10 Symposium (TENSYP)* (pp. 194-197). IEEE.
- [34] Ahmad Tasnim Siddiqui and Sultan Aljahdali. Web mining techniques in e-commerce applications. arXiv preprint arXiv:1311.7388, 2013.

- [35] Dwijen Rudrapal, Amitava Das, and Baby Bhattacharya. Measuring semantic similarity for bengali tweets using wordnet. In Proceedings of the International Conference Recent Advances in Natural Language Processing, pages 537–544, 2015.
- [36] Fankar Armash Aslam, Hawa Nabeel Mohammed, Jummal Musab Mohd, Murade Aaraf Gulamgaus, and PS Lok. Efficient way of web development using python and flask. International Journal of Advanced Research in Computer Science, 6(2), 2015.

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:2
Student Name & ID: Yvonne Eng Xin Yee	
Supervisor: Aun Yichiet	
Project Title: DealWithIt - Real Time Price Checker with Object Recognition	

1. WORK DONE

Preprocessing of images to remove background noise

2. WORK TO BE DONE

Refine object detection model to return only 1 object, try remove noise as many as can, perhaps return biggest object so that volume more accurate

3. PROBLEMS ENCOUNTERED

Need to exclude contour for Aruco marker

4. SELF EVALUATION OF THE PROGRESS

OK



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:7
Student Name & ID: Yvonne Eng Xin Yee	
Supervisor: Aun Yichiet	
Project Title: DealWithIt - Real Time Price Checker with Object Recognition	

1. WORK DONE

deployment and deployment in mobile application

2. WORK TO BE DONE

web scraping, crawling

3. PROBLEMS ENCOUNTERED

Package to use in mobile application to get scraped data

4. SELF EVALUATION OF THE PROGRESS

OK



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	Study week no.:10
Student Name & ID: Yvonne Eng Xin Yee	
Supervisor: Aun Yichiet	
Project Title: DealWithIt - Real Time Price Checker with Object Recognition	

1. WORK DONE

Web scraping from different supermarket and built Firebase

2. WORK TO BE DONE

Create Rest API

3. PROBLEMS ENCOUNTERED

Find cloud platform and host Flask server

4. SELF EVALUATION OF THE PROGRESS

OK



Supervisor's signature



Student's signature

POSTER

DealWithIt – Real Time Price Checker with Object Recognition

Introduction

With the fluctuation of pricing and sizing of products, most consumers need to spend large amount of time and effort in comparing the same products from different supermarkets. They rarely realize the changes even if the portion for product or the price is modified. Thus, this mobile application is developed showing all the needed and newest information in one screen.



sample work



Research Objectives

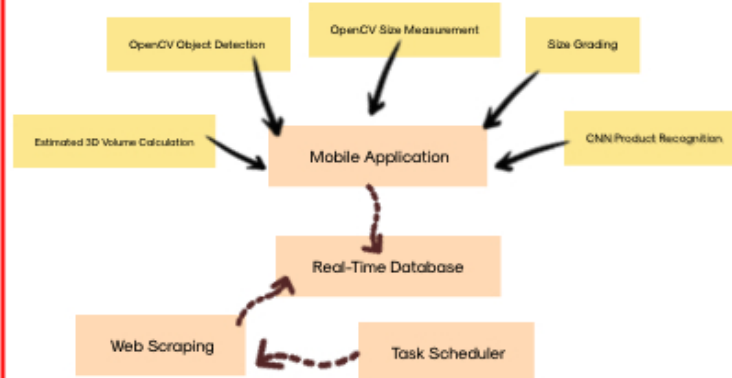
Main objective :

- mobile application of a real-time price checker with size measurement and product recognition

Sub-objectives :

- parallax corrective method for width and height estimation
- 3D volumetric estimation method from 2D image
- capture, save and update pricing information on real-time database

Methodologies



The methodologies is achieved using the technologies from CNN and Tensorflow Keras, OpenCV, Selenium, Flask, Firebase, and Kivy

Conclusion



The developed system help to eliminate consumers' frustration encountered in sizing and pricing discrepancies of the same product from different supermarkets and empower them to make informed purchasing decision.

By: Yvonne Eng Xin Yee / FICT CS

Made with PosterMyWall.com



PLAGIARISM CHECK RESULT

DealWithIt - Real Time Price Checker with Object Recognition

ORIGINALITY REPORT

9% SIMILARITY INDEX	6% INTERNET SOURCES	3% PUBLICATIONS	3% STUDENT PAPERS
-------------------------------	-------------------------------	---------------------------	-----------------------------

PRIMARY SOURCES

1	Limeng Pu, Rui Tian, Hsiao-Chun Wu, Kun Yan. "Novel object-size measurement using the digital camera", 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016 Publication	1%
2	Submitted to Universiti Tunku Abdul Rahman Student Paper	1%
3	docplayer.net Internet Source	1%
4	eprints.utar.edu.my Internet Source	1%
5	www.hindawi.com Internet Source	1%
6	www.tandfonline.com Internet Source	1%
7	mindmajix.com Internet Source	<1%

17	Submitted to Taibah University Student Paper	<1 %
18	Submitted to Technological University Dublin Student Paper	<1 %
19	Submitted to Colorado Technical University Online Student Paper	<1 %
20	Submitted to University of Central England in Birmingham Student Paper	<1 %
21	Submitted to University of Scranton Student Paper	<1 %
22	docbox.etsi.org Internet Source	<1 %
23	courses.cs.ut.ee Internet Source	<1 %
24	hugepdf.com Internet Source	<1 %
25	www.ih.europa.eu Internet Source	<1 %
26	www.protocols.io Internet Source	<1 %
27	"Machine Intelligence and Smart Systems", Springer Science and Business Media LLC, 2021	<1 %

Publication

28	Submitted to Wawasan Open University Student Paper	<1 %
29	geniusee.com Internet Source	<1 %
30	manualzz.com Internet Source	<1 %
31	repository.up.ac.za Internet Source	<1 %
32	idoc.pub Internet Source	<1 %
33	www.manualslib.com Internet Source	<1 %
34	groups.google.com Internet Source	<1 %
35	www.medicalnewstoday.com Internet Source	<1 %
36	www.usenix.org Internet Source	<1 %
37	www.windowcentral.com Internet Source	<1 %

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)	Yvonne Eng Xin Yee
ID Number(s)	20ACB01886
Programme / Course	Bachelor of Computer Science
Title of Final Year Project	DealWithIt - Real Time Price Checker with Object Recognition

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u> 9 </u> % Similarity by source Internet Sources: <u> 6 </u> % Publications: <u> 3 </u> % Student Papers: <u> 3 </u> %	
Number of individual sources listed of more than 3% similarity: <u> 0 </u>	
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.

Signature of Supervisor

Name: Aun Yichiet

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	20ACB01886
Student Name	Yvonne Eng Xin Yee
Supervisor Name	Aun Yichet

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)
√	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.

Yvonne Eng Xin Yee

(Signature of Student)

Date: 26/04/2024