REAL-TIME DOCUMENT READER ASSISTANCE APP FOR THE VISUALLY IMPAIRED

BY

DARREN CHOONG YU XUEN

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)

FEBRUARY 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ts Dr Saw Seow Hui and my moderator, Prof. Leung Kar Hang who has given me this bright opportunity to engage in a mobile assistive application development for contributing to visually impaired. It is my first step to establish a career in mobile application development, computer vision and machine learning field. A million thanks to you.

ABSTRACT

Due to the fast-paced lifestyle today, blind people find it hard to access documents independently and efficiently. The growing demand for accessible technology has brought attention to the challenges faced by visually impaired individuals when accessing and reading physical and digital documents. Furthermore, most assistive technology requires a subscription plan, therefore leading to financial difficulty for visually impaired individuals. To tackle these problems, a real-time document reader assistance application using Flutter is developed for free to use. It will be developed on both the iOS and Android platforms. It will also be focused more on the targeted audience, which is the visually impaired individuals. Hence, the proposed solution will integrate OCR text recognition techniques for text extraction using Google ML Kit. Additionally, a transfer learning on pre-trained model (YOLOv8n) will be used to detect documents in the camera preview. Moreover, Text-to-Speech (TTS) will be implemented for real-time audio feedback. Furthermore, an AI chatbot will be implemented to aid users in further clarification of the recognized text from the document. Speech-to-Text (STT) will also be implemented to convert audio from the user into text when questioning the AI chatbot assistance without typing. This project development process follows the Agile methodology, which involves several phases such as planning, analysis, design, implementation, and testing, with continuous iterations. The result of this project will serve as an innovative tool that improves the accessibility of information for visually impaired users and contributes to the MAHAL application. In conclusion, this project will provide a significant contribution to enhancing the independence and accessibility of visually impaired users.

Area of Study (Minimum 1 and Maximum 2): Mobile App Development, Artificial Intelligence

Keywords (Minimum 5 and Maximum 10):Text Recognition, Document Detection, Yolov8n, STT, Ambient Light Sensor, Google ML Kit, TTS, Chatbot

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

WHO	World Health Organization
MAHAL	Malaysian Assistive Hub for Advanced Livelihood
MAB	Malaysian Association of the Blind
TTS	Text-to-Speech
STT	Speech-to-text
ML	Machine Learning
SDK	Software Development Kit
YOLO	You Only Live Once

CHAPTER 1

Project Background

This chapter will present the background and motivation of our research, the specific problem addressed by this project, the objectives that this project aims to achieve, and the contributions to the field of assistive technology for the visually impaired.

Introduction

In today's fast-paced world, highly individuals need to have access to the information they require in case this information is needed, including individuals who have vision problems. Though there have been profound pressures in technology, the visually impaired group struggles to read regular and digital information conveniently and efficiently in real-time. In the past, there have been some forms of methods for the vision disabled such as Braille and audiobooks. Although they have some functionality, they are not as convenient and versatile in various situations.

The ability to read and understand documents independently is a fundamental aspect of daily life from accessing educational materials or other documents. For the visually impaired, this task is often facing significant barriers and might lead to a reliance on other aid from people, and sometimes even paying for costly technology assistance. This dependency might impact on their daily lives in education, social interaction, and personal growth. Moreover, many existing solutions require extensive training or are not adaptable to different types of documents, making them less practical in real-world situations.

Thus, the development of a Real-Time Document Reader Assistance App is specifically developed for the visually impaired. It acts as an assistant to empower visually impaired individuals to independently read and understand a variety of documents in real-time and enhance their participation and personal growth on social activities in their daily life. This proposal aims to develop mobile applications on both IOS and Android Platform for visually impaired individuals. The project is using Flutter to develop both Android and IOS. Additionally, it will be using Google ML Kit (Machine Learning) Kit to develop real-time processing text recognition from documents. Additionally, YOLOv8n will be used to detect any document in the camera preview. Furthermore, a Flutter TTS (text-to-speech) is used to allow the app to read out the text after real-time processing. Moreover, Flutter STT (speech-to-text) is also integrated to convert user spoken audio into text. Also, chatbot is integrated to response back to user based on their question. Besides, TensorFlow Lite will be used to convert the trained YOLOv8n model into a mobile friendly format.

1.1 Problem Statement and Motivation

i) Visual Impaired individuals faced significant barriers in accessing, reading and understanding documents independently.

As stated by WHO [1], as of 2019, at least 2.2 billion people are facing a near or distance vision impairment globally. According to [2], most of the people who are visually impaired, whether they are blind or have low vision, often face significant challenges in their daily life. For instance, many visually impaired individuals find it difficult to interpret the details in the receipt during payment transaction or to read and understand documents. Furthermore, these challenges not only impact their independence and quality of life but can also affect their psychological well-being, leading to increased feelings of frustration, stress, and isolation. Therefore, to tackle this issue, developing an effective document reader that is accurate, and accessible information is crucial for the visually impaired.

ii) Time consuming to process document and text recognition

Nowadays, most mobile applications are not integrating real-time features for processing the images. Instead, most of the apps are using the standard method to process the images where the visually impaired user uses the camera from their mobile device and captures the images. Then, the images are processed and analyzed by the application algorithms later. Hence, it is time consuming and not efficient where the user is not able to obtain immediate feedback from the application and the user needed to press a button in the middle to capture the images for processing the recognition. Sometimes, this delay might cause inconvenience when the user is in an urgent situation or in a fast-paced environment. Therefore, to address this issue, a real-time feature is needed to implement to allow them to have immediate feedback and reducing waiting time.

iii) Absence of document detection functionality

Some of the assistive applications although having text recognition capabilities, however it still brought difficulties to user where user might not be able to locate documents within the camera's frame, making it difficult for users to position the camera correctly. To address this issue, a document detection model will need to implement to detect any document exist in the camera preview and provide immediate feedback to user.

The aim of the thesis is to propose a real-time document reader assistance app to enhance their ability to perform daily tasks and improve their quality of life. By providing a solution that empowers visually impaired users and integrates seamlessly into their daily routines, the app addresses a significant need for accessible technology. Besides that, integrating the real-time features in the app could reduce the time when processing the image and provides immediate feedback. Lastly, in this thesis, free-to-use features is proposed as one of the crucial parts where the visually impaired user can access essential functionalities without concerning financial difficulties.

1.2 Research Objectives

i) To develop free to use real-time document reader assistance on mobile application

The main objective of this project is to develop a real-time document reader mobile application that allows visually impaired individuals to independently access, read, and understand both printed and digital documents. By developing this project, it will be able to enhance their ability to perform daily task more efficiently and improve their quality of life.

ii) To reduce the time when processing text recognition

A key sub-objective of this project is to optimize the efficiency of the text recognition process and duration. By integrating the real-time image capturing and real-time audio feedback features, it can ensure that visually impaired users can receive immediate feedback more efficiently and seamlessly.

iii) **To provide a real-time document detection for visually impaired** Another critical sub-objective of this project is to develop a model to enable the application to detect document and notify user if document detected. This capability minimizes user errors and enhances the overall usability of the application for visually impaired individuals.

1.3 Project Scope and Direction

The purpose of this project is to develop a real-time document reader assistance application using Flutter for the visually impaired which could allow visually impaired users to access and read documents including text independently. The development and design of mobile applications specifically focus on the target audience which is the visually impaired users. Therefore, the app will use OCR text recognition to extract the text from document. The OCR can perform text extraction in either Chinese or English language. Furthermore, it will use transfer learning on pretrained model (Yolov8n) with custom dataset of document in white-based background to detect objects that appear in the camera preview. The model able to detect most type of the document, but compulsory needed in white-based background. Additionally, the app will have a user-friendly interface where it can enhance the user accessibility including flashlight with ambient light sensor, clear audio feedback, and tutorial to guide the user on how to operate this application. Besides, the app will also utilize Flutter (TTS) to convert the text information into audio and voice out to user. Furthermore, an AI chatbot will be developed to assist users after the text recognition has been completed. A Flutter Speech-To-Text feature will also be integrated for users to use voice command to seek for AI assistance instead of keyboard typing. The chatbot is mainly developed to interpret document data on receipt and can only process static questions and does not support dynamic responses or generate outputs in natural, conversational formats like GPT-based models. This project aims to provide a comprehensive, accessible, and user-friendly application for visually impaired individuals which can significantly enhance their ability to interact with various types of documents independently. In the end of the project, a real-time document reader assistance app for the visually impaired will be developed and delivered to the visually impaired users and contribute to MAHAL.

1.4 Contributions

This project will end up contributing to the MAHAL (Malaysian Assistive Hub for Advanced Livelihood) where it represents as an application for MAB (Malaysian Association of the Blind) as well as to public users that are visually impaired and developers that are working towards the field of visually impaired.

Figure 1.4.1 MAHAL Logo

MAHAL is an application which designed to provide a range of services and tools to support the blind people in all aspects of their lives. The development of real-time document reader assistance features in this project can helps in contributing as essential features and resources for MAHAL to help the users navigate the challenges of living with visual impairment. Furthermore, by integrating this feature, this project would foster greater independence and improve the overall quality of life for visually impaired individuals in public.

Furthermore, this project will serve as a valuable reference for developers that are working in the field of assistive technology for the visually impaired. By developing real-time document reader assistance, it can offer insights and innovative approaches which could inspire further research and development in assistive technology of visually impaired for other developers.

1.5 Report Organization

The details of this research are shown in the following chapters. Chapter 1 provides a brief overview of problem statements, objectives, scope and contribution in this project. In Chapter 2, it involves technical review on related studies and reviewing, comparing the advantages and disadvantages of application in the visually impaired field. Chapter 3 describes the methodology, suggested approach and system requirement of hardware and software, and the timeline that involves this project. Chapter 4 provides details of the system design which involves flowcharts and detailed architecture of this development project. Chapter 5 provides details of the project's implementation work to achieve the complete development of this project. Chapter 6 includes system evaluation, test cases and objectives evaluation to identify whether the objectives have been achieved. Lastly, Chapter 7 provides a summary of the project which includes conclusion and future recommendations.

Chapter 2

Literature Review

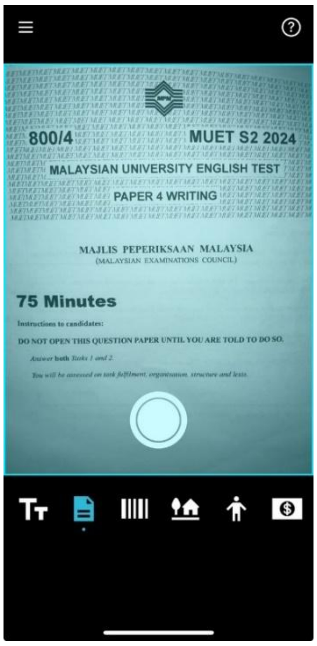
2.1 Previous Works on Document Reader Assistance Application for Visually Impaired

2.1.1 Seeing AI

Figure 2.1.1.1 Seeing AI Logo

Seeing AI is an artificial intelligence application developed by Microsoft on iOS and Android platform [3]. This app is specifically designed for the blind and low vision community. Furthermore, it includes a wide variety of features that are essential for visually impaired individuals such as short text, documents, products, scenes, people, currency, handwriting, light and images in other apps. Moreover, these features are free-to-use without any subscription plan by users [3].

Furthermore, it includes 20 languages where the user can choose to choose the relevant languages that best suit their needs. It also includes a wide range of voice spoken that user can select based on their own personal preferences. In this app, it also allows users to adjust the paces of spoken languages from 0% up to 100%. Besides, it also includes a feature where Seeing AI will automatically adjust the camera flash or torch based on the lightning conditions [3].


800/4

MUET S2 2024

MALAYSIAN UNIVERSITY ENGLISH TEST

PAPER 4 WRITING

MAJLIS PEPERIKSAAN MALAYSIA

(MALAYSIAN EXAMINATIONS COUNCIL)

75 Minutes

Instructions to candidates:

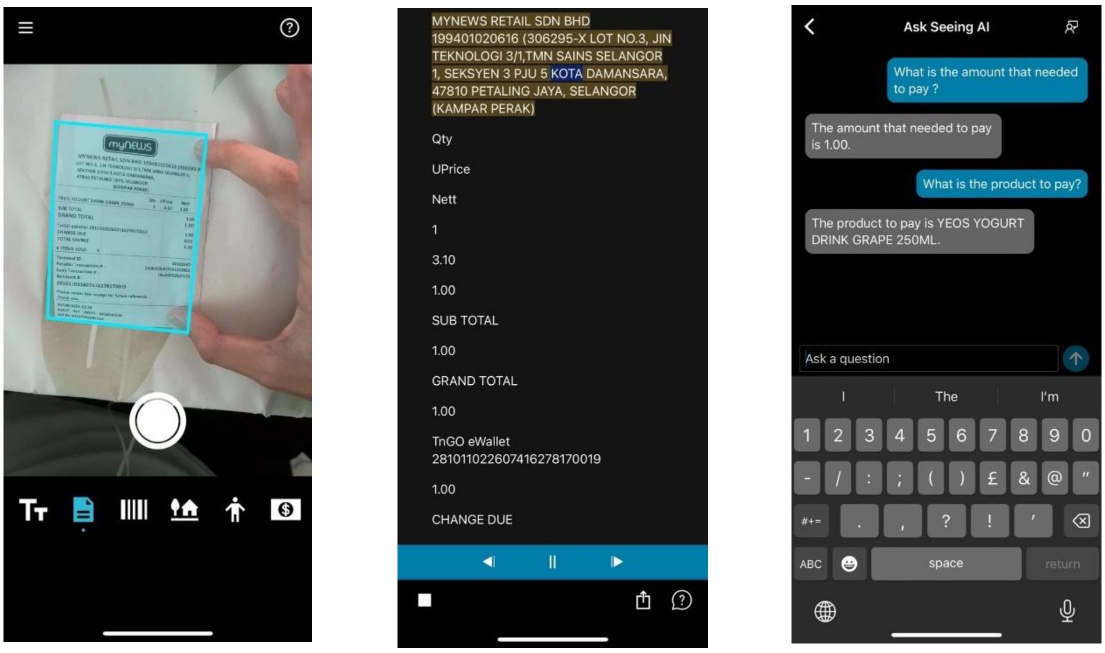
DO NOT OPEN THIS QUESTION PAPER UNTIL YOU ARE TOLD TO DO SO.

Answer **both** *Tasks 1 and 2.*

You will be assessed on task fulfilment, organisation, structure and texts.

Figure 2.1.1.2 Document reader interface (Seeing AI)

A demonstration is provided on reading a receipt, and then the image is processed in real-time into text and output in another interface provided by Seeing AI. Then, questions and answer are performed on the document to the seeing AI. Although there is a capture button in the middle bottom of the document reader interface, it also can support real-time features where user no need to manually press the capture button to capture the receipt.


MYNEWS RETAIL SDN BHD 199401020616 (306295-X LOT NO.3, JIN TEKNOLOGI 3/1,TMN SAINS SELANGOR 1, SEKSYEN 3 PJU 5 KOTA DAMANSARA, 47810 PETALING JAYA, SELANGOR (KAMPAR PERAK)

Qty

UPrice

Nett

1

3.10

1.00

SUB TOTAL

1.00

GRAND TOTAL

1.00

TnGO eWallet
281011022607416278170019

1.00

CHANGE DUE

Ask Seeing AI

What is the amount that needed to pay ?

The amount that needed to pay is 1.00.

What is the product to pay?

The product to pay is YEOS YOGURT DRINK GRAPE 250ML.

Ask a question

Figure 2.1.1.3 Demo for reading a receipt (Seeing AI)

Strength of Seeing AI:

In the application, it includes real-time automatic capture and processing image features. In this case, users have no need to press any button when capturing the images. Instead, it will automatically capture and process the images and deliver voice feedback immediately to the user within a few seconds. This real-time feature is important because it enhances the user's experience without any manual intervention. Furthermore, visual impaired users could also quickly identify and understand the document without any delay of time.

Additionally, it allows users to perform questions that are open-ended without restriction, providing users with the flexibility to ask questions and provide comprehensive responses. This indicates that visually impaired users are not limited to predefined or specific types of questions. Instead, they can ask about a specific detail of the recognized document in the AI chat assistance. This capability greatly enhances the user experience and ensures that visually impaired users can obtain a detailed answer.

Furthermore, Seeing AI also provides a button where the user can add additional pages for capturing multiple documents at the same time. This functionality is user friendly where it involves tasks where users need to scan multi-page documents such as reading a long-text document or making comparison in different document. Whether it is comparing the details across different receipts, reviewing a multi-page document, or reading through a lengthy document, this user-friendly feature provides efficiency and convenience to user. By enabling users to capture multiple pages in a single session, Seeing AI enhances the usability and practicality of the app, which solves the needs of users in handling multiple tasks.

Weakness of Seeing AI:

While Seeing AI offer numerous of advantages, it also having some of the drawbacks. Notably in document reader, it does not support the recognition and voice out of other text languages other than English. This limitation restricts its usability for users who need to scan and process documents in different languages. As a result, visually impaired individuals who encounter texts in non-English languages may find it challenging to use the app effectively. Such limitations might bring frustration and anxiety for user in encountering Multilanguage environment, thus affecting their daily live.

Furthermore, the next limitation of Seeing AI is the absence of speech-to-text features when user is questioning AI chat. This indicates that visually impaired users cannot speak their questions or commands directly to the app and instead must rely on typing them out using a keyboard. For visually impaired users, this can be particularly challenging and inconvenient, as typing can be more difficult for them to perform their task.

Seeing AI can recognize and voice out the document in a very short period. However, its performance is mostly dependent on the mobile camera quality. Users with older smartphone or low-end of devices might experience lower accuracy and longer processing time in recognizing the document. This dependency might bring issue to visually impaired users where they needed to purchase a high-end device with clear mobile camera quality for a high accuracy and shorter processing time of document recognition.

Table 2.1.1.1 Strength and Weakness (Seeing AI)

Strength	Weaknesses
• Real-time automatic capture and processing image • Allow user perform question that are open-ended • Able to add additional pages for capturing multiple documents at the same time	• Unable to recognize and voice out other text languages other than English • Absence of speech-to-text feature when user is questioning the AI • Dependency on mobile camera quality

Table 2.1.1.2 Recommended solution to resolve weakness (Seeing AI)

Weaknesses	Recommended Solution
Unable to recognize and voice out other text languages other than English	Integrate Multilanguage TTS and OCR
Absence of speech-to-text feature when user is questioning the AI	Integrate speech-to-text technology
Dependency on mobile camera quality	Implement image enhancement algorithm to address low-resolution image.

2.1.2 Be My Eyes

Figure 2.1.2.1 Be My Eyes Logo

Be my eyes is a mobile application that develop by Hans Jørgen Wiberg and Christian Erfurt in 2015 [4]. It is developed on both IOS and Android platform [4]. The aim is to help blind people to recognize objects and text within their daily lives. Furthermore, it had a unique feature where users can connect with sighted volunteers via live video calls [4]. These features allow users to obtain assistive support such as reading, identifying objects, or navigating through their surroundings from people. Besides, it also can capture photos and process the image to obtain the text information from the images and display it out [4].

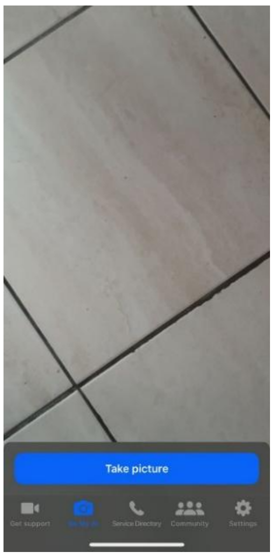
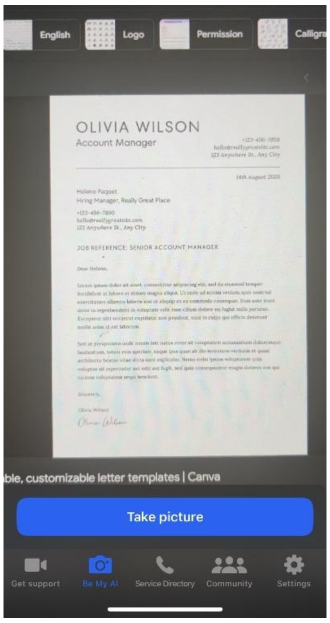


Figure 2.1.2.2 Image capture interface (Be My Eyes)

A demonstration is presented on reading a receipt and a letter. Be My Eyes can interpret the type of receipt and display it back to the user.



OLIVIA WILSON
Account Manager

+123-456-7890
hello@reallygreatsite.com
123 Anywhere St., Any City

14th August 2020

Helene Paquet
Hiring Manager, Really Great Place
+123-456-7890
hello@reallygreatsite.com
123 Anywhere St., Any City

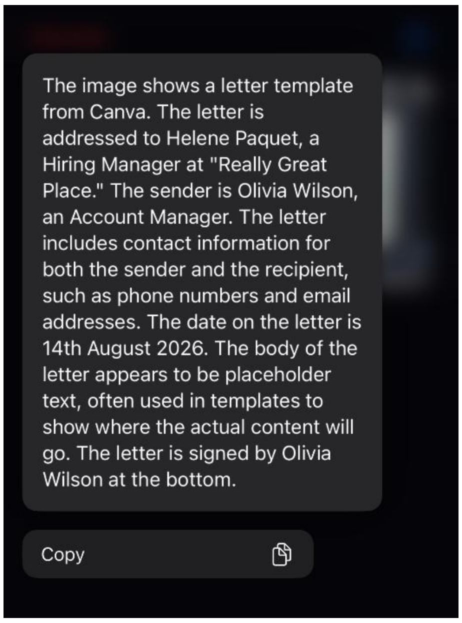
JOB REFERENCE: SENIOR ACCOUNT MANAGER

Dear Helene,

[illegible]

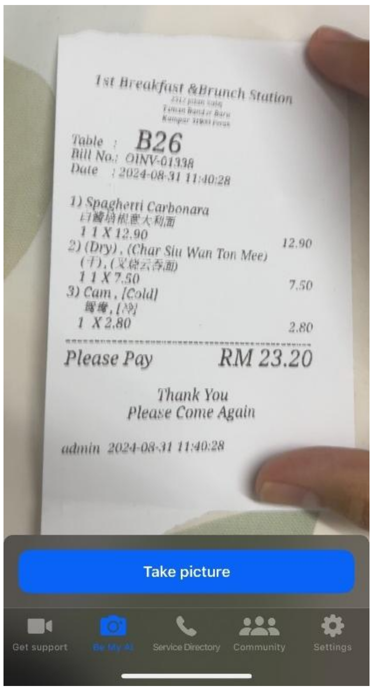
Sincerely,

Olivia Wilson



The image shows a letter template from Canva. The letter is addressed to Helene Paquet, a Hiring Manager at "Really Great Place." The sender is Olivia Wilson, an Account Manager. The letter includes contact information for both the sender and the recipient, such as phone numbers and email addresses. The date on the letter is 14th August 2026. The body of the letter appears to be placeholder text, often used in templates to show where the actual content will go. The letter is signed by Olivia Wilson at the bottom.

Figure 2.1.2.3 Demo for reading a letter (Be My Eyes)



1st Breakfast &Brunch Station
2312 Jalan Kolej
Taman Bandar Baru
Kampar 31900 Perak

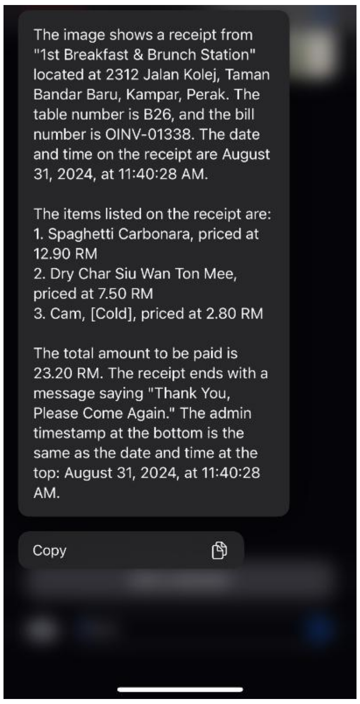
Table : B26
Bill No.: OINV-01338
Date : 2024-08-31 11:40:28

1) Spaghetti Carbonara
白酱培根意大利面
1 1 X 12.90 12.90
2) (Dry) , (Char Siu Wan Ton Mee)
(干), (叉烧云吞面)
1 1 X 7.50 7.50
3) Cam , [Cold]
鸳鸯 , [冷]
1 X 2.80 2.80

Please Pay RM 23.20

Thank You
Please Come Again

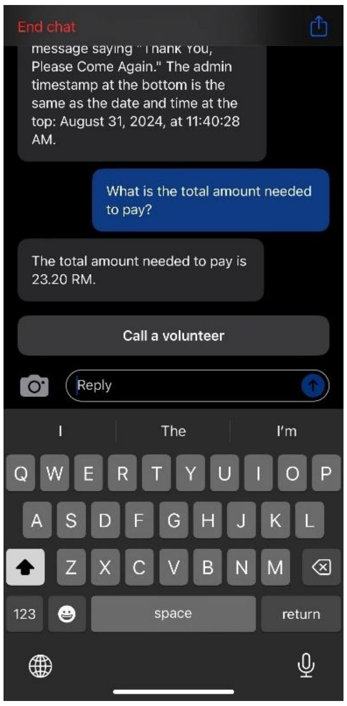
admin 2024-08-31 11:40:28



The image shows a receipt from "1st Breakfast & Brunch Station" located at 2312 Jalan Kolej, Taman Bandar Baru, Kampar, Perak. The table number is B26, and the bill number is OINV-01338. The date and time on the receipt are August 31, 2024, at 11:40:28 AM.

The items listed on the receipt are:
1. Spaghetti Carbonara, priced at 12.90 RM
2. Dry Char Siu Wan Ton Mee, priced at 7.50 RM
3. Cam, [Cold], priced at 2.80 RM

The total amount to be paid is 23.20 RM. The receipt ends with a message saying "Thank You, Please Come Again." The admin timestamp at the bottom is the same as the date and time at the top: August 31, 2024, at 11:40:28 AM.



message saying "Thank You, Please Come Again." The admin timestamp at the bottom is the same as the date and time at the top: August 31, 2024, at 11:40:28 AM.

What is the total amount needed to pay?

The total amount needed to pay is 23.20 RM.

Figure 2.1.2.4 Demo for reading a receipt (Be My Eyes)

Strength of Be My Eyes:

In Be My Eyes, after the user captures an image of a document, it can recognize the type of document based on the content and structure of the document. This app extracts and interprets the text, then categorizes and determines the type of document accurately. It benefits user where it able to interpret and display the type of document which helps user to understand the document without manual effort.

Another strength of Be My Eyes is where it able to organize the data extracted from document to user. Rather than simply displaying all the recognized data in a raw format back to the user, the app processes and structures the information in a user-friendly manner. This feature ensures that users can easily access the information they need without being overwhelmed by unstructured text.

Furthermore, Be My Eyes also included a build in feature that able to call a volunteer for assistance after the data is structured. If the visually impaired user is unclear about the information from the document, they can easily call and connect with a volunteer and seek assistance. This feature ensures that users have access to real-time human assistance for further clarification or guidance.

Weakness of Be My Eyes:

While Be My Eyes contains a lot of unique features, it was found out that this app relies on network connectivity. It cannot access any of the features in this app where it does not have any internet connections. This might bring inconvenience to the user where it potentially impedes the user from performing tasks and feeling anxiety.

The second limitation of Be My Eyes is it does not support audio feedback. The lack of audio feedback means the app does not voice out back to the visual impaired users. Furthermore, audio feedback is crucial for the visual impaired users as most of them are relying on hearing. It is challenging and difficult for the visually impaired user to interpret the content of the document without audio feedback. This shortcoming highlights the need for implementing audio feedback for visually impaired users.

The following drawback of Be My Eyes is the absence of speech-to-text feature when user is questioning the AI chat. . This limitation requires visually impaired users to manually type their questions, which can be inconvenient and time-consuming. For users who rely on voice input due to its own sight difficulties when typing, the lack of a speech-to-text option can be a major inconvenience for them.

Table 2.1.2.1 Strength and Weakness (Be My Eyes)

Strength	Weaknesses
• Able to recognize the type of document • Able to organize the data of document for user • Build in feature that able to call a volunteer for assistance	• Rely on network connectivity • Does not support audio feedback • Absence of speech-to-text feature when user is questioning the AI chat

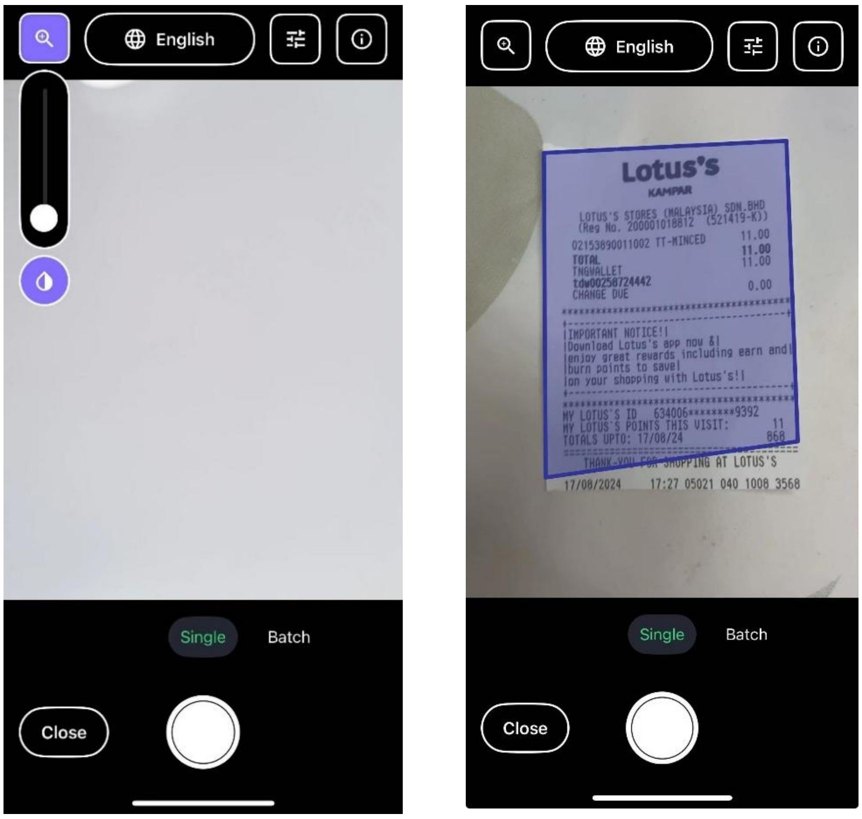
Table 2.1.2.2 Recommended solution to resolve weakness (Be My Eyes)

Weaknesses	Recommended Solution
Rely on network connectivity	Develop offline capabilities and local storage processing
Does not support real-time audio feedback	Integrate text-to-speech for converting text into real-time audio
Absence of speech-to-text feature when user is questioning the AI	Integrate speech-to-text functionalities

2.1.3 Envision AI

Figure 2.1.3.1 Envision AI Logo

Envision AI is a free visual assistance app that was developed by Karthik Mahadevan and Karthik Kannan in late 2017 [5]. Envision AI is supported on both Android and iOS [5]. The purpose of this app is to offer multiple accessibility features, including speech-to-text, magnifier, and other accessibility features that make it easier for visually assistance people to get the information they need [5].



```
Lotus's
KAMPAR
LOTUS'S STORES (MALAYSIA) SDN.BHD
(Reg No. 200001018812  (521419-K))
02153890011002 TT-MINCED          11.00
                                  11.00
TOTAL                             11.00
TNGWALLET
tdw00258724442                     0.00
CHANGE DUE
*****************************************
+---------------------------------------+
|IMPORTANT NOTICE!|
|Download Lotus's app now &|
|enjoy great rewards including earn and|
|burn points to save|
|on your shopping with Lotus's!|
+---------------------------------------+
*****************************************
MY LOTUS'S ID   634006********9392
MY LOTUS'S POINTS THIS VISIT:        11
TOTALS UPTO: 17/08/24               868
=========================================
    THANK-YOU FOR SHOPPING AT LOTUS'S
17/08/2024    17:27 05021 040 1008 3568
```

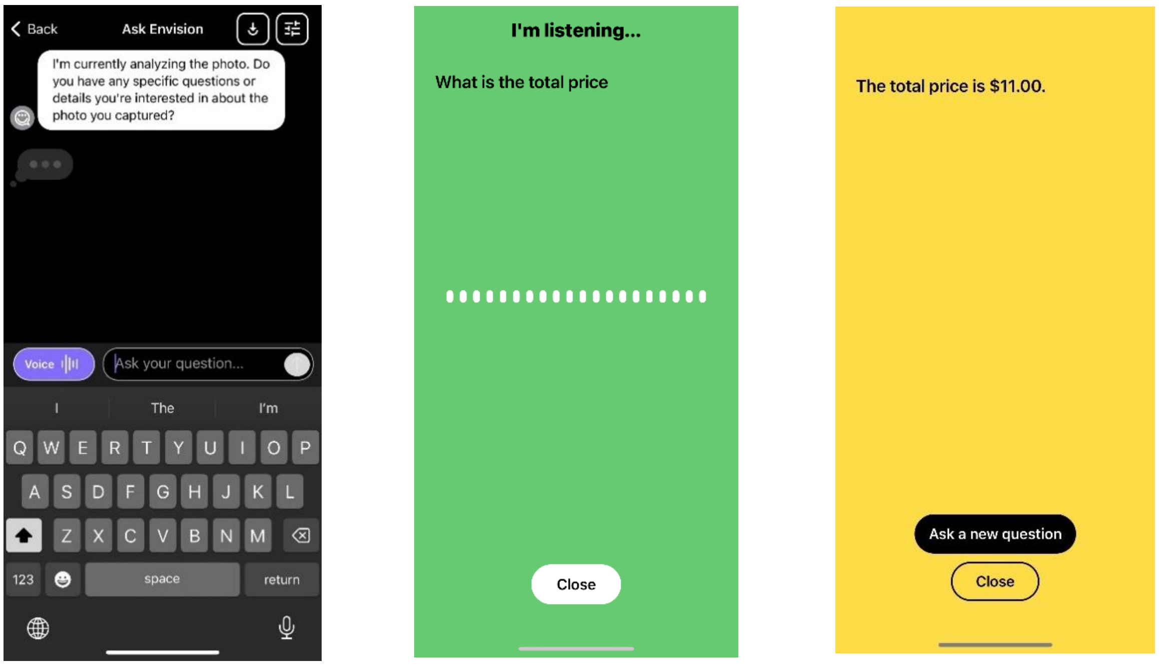
Figure 2.1.3.2 Document reader interface (Envision AI)

Strength of Envision AI:

Envision AI provides a feature that supports speech-to-text when visually impaired user is interacting with the chat. This feature will benefit the user as it allows them to use voice messages to ask questions of the recognized document, rather than typing out from keyboard. It further enhances the accessibility of the app and improves the overall user experience when interacting with Envision AI chat.

Furthermore, another significant advantage of Envision AI is its ability to recognize and voice out other text languages other than English. This multilingual support enhances the app accessibility and usability for the visually impaired user in their daily life. In fact, sometimes the visually impaired user encounters situations where the text is in other languages such as Mandarin. Therefore, users can swap for recognition and voice out in different languages.

In addition, Envision AI also includes a user-friendly interface. Users can easily switch in different language recognition, modify the magnifier, and apply a color inversion filter without going to setting for adjustment. These customizable options provide visually impaired user with greater control, allowing them to make adjustment easily based on their specific needs.



I'm currently analyzing the photo. Do you have any specific questions or details you're interested in about the photo you captured?

I'm listening...

What is the total price

The total price is $11.00.

Figure 2.1.3.3 Speech-to-text feature (Strength of Envision AI)

Weakness of Envision AI:

However, a significant limitation of Envision AI is due to its inaccurate document and text recognition. Sometimes, the AI might misinterpret the details of documents and thus provide wrong information to users. Such inaccuracies might lead to confusion to the visually impaired user, and they might be struggling especially in a situation where precise information is crucial for them to make decisions.

While Envision AI contains a lot of user-friendly features, it was found out that its inability of real-time automatic capture and process image. The absence of this feature implies that users must manually take pictures and wait for the app to process the images. This drawback can cause inconvenience, particularly for visually impaired users who may rely from a more hands-free experience.

Table 2.1.3.1 Strength and Weakness (Envision AI)

Strength	Weaknesses
• Support speech-to-text feature • Ability to recognize and voice out other text languages other than English • User friendly interface	• Inaccurate document and text recognition • Inability of real-time automatic capture and process image

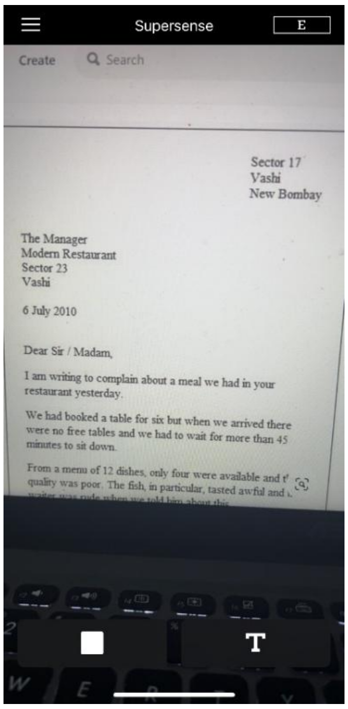
Table 2.1.3.2 Recommended solution to resolve weakness (Envision AI)

Weaknesses	Recommended Solution
Inaccurate document and text recognition	Enhance the apps machine learning model by using a more diverse dataset of document
Inability of real-time automatic capture and processing image	Implement real-time features on capturing and processing image

2.1.4 Supersense

Figure 2.1.4.1 Supersense Logo

Supersense is a mobile application to assist visual impaired and developed by Mediate [6]. Supersense was available on both IOS and Android platforms. This application can work in offline mode without relying on network connectivity, Furthermore, the features are differentiated into free plan and premium plan. In free plan, visually impaired users have unlimited access to quick read mode, import, read history and magnifier [6]. In premium plan, it had additional features such as smart scanner, currency recognition, scene description and other specific features [6]. However, visually impaired users need to perform subscription to have full access to this application.



Sector 17
Vashi
New Bombay

The Manager
Modern Restaurant
Sector 23
Vashi

6 July 2010

Dear Sir / Madam,

I am writing to complain about a meal we had in your restaurant yesterday.

We had booked a table for six but when we arrived there were no free tables and we had to wait for more than 45 minutes to sit down.

From a menu of 12 dishes, only four were available and t[illegible] quality was poor. The fish, in particular, tasted awful and [illegible]

Figure 2.1.4.2 Interface (Supersense)

Strength of Supersense:

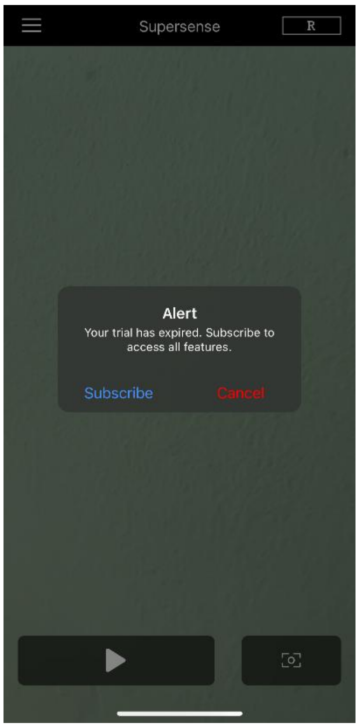
Supersenses provide a strength where it does not rely on network connectivity. This feature indicates that users can access and utilize the app without requiring an internet connection. As a result, Supersense offers significant flexibility and usability in areas which are poor or no network access.

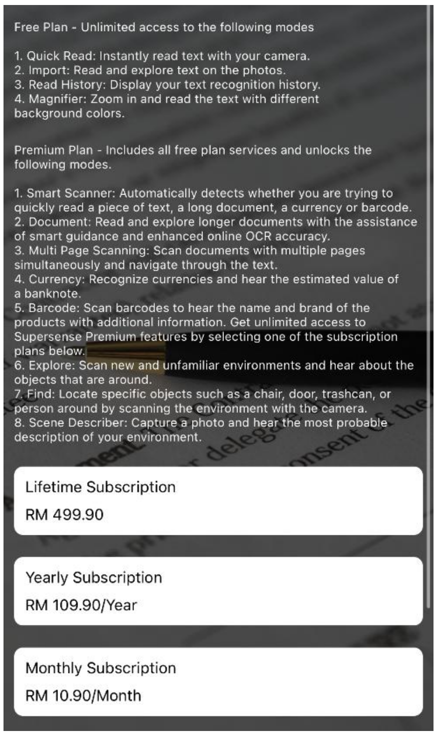
In addition, Supersense also offers real-time audio feedback in quick-read mode. This feature allows us to receive immediate feedback without delay. Thus, Supersense enhances efficiency on reading documents, without having visually impaired users to wait for processing recognition.

Weaknesses of Supersense:

However, a significant limitation is where it requires subscription plan to access full features of the app. Users needed to subscribe to the premium plan to unlock and utilize its comprehensive capabilities. This subscription plan might cause difficulties for users, particularly those who may not be able to afford the extra cost. As a result, users with limited financial resources might find themselves unable to access all the features that could enhance their experience of this application.

Another drawback is where it does not include AI chat for assistance. This absence means that visually impaired users cannot interact with a virtual AI assistant through text or voice chat. The lack of an AI chat can limit the app's ability to provide additional assistance and clarification to answer user questions. Without this feature, it might potentially impact on the overall experience of visually impaired users.




Free Plan - Unlimited access to the following modes

1. Quick Read: Instantly read text with your camera.
2. Import: Read and explore text on the photos.
3. Read History: Display your text recognition history.
4. Magnifier: Zoom in and read the text with different background colors.

Premium Plan - Includes all free plan services and unlocks the following modes.

1. Smart Scanner: Automatically detects whether you are trying to quickly read a piece of text, a long document, a currency or barcode.
2. Document: Read and explore longer documents with the assistance of smart guidance and enhanced online OCR accuracy.
3. Multi Page Scanning: Scan documents with multiple pages simultaneously and navigate through the text.
4. Currency: Recognize currencies and hear the estimated value of a banknote.
5. Barcode: Scan barcodes to hear the name and brand of the products with additional information. Get unlimited access to Supersense Premium features by selecting one of the subscription plans below.
6. Explore: Scan new and unfamiliar environments and hear about the objects that are around.
7. Find: Locate specific objects such as a chair, door, trashcan, or person around by scanning the environment with the camera.
8. Scene Describer: Capture a photo and hear the most probable description of your environment.

Lifetime Subscription
RM 499.90

Yearly Subscription
RM 109.90/Year

Monthly Subscription
RM 10.90/Month

Figure 2.1.4.3 Require of subscription plan (Limitations of Supersense) [7]

Table 2.1.4.1 Strength and Weakness (Supersense)

Strength	Weaknesses
• Does not rely on network connectivity • Real-time audio feedback	• Require Subscription Plan • Does not include AI chat for assistance

Table 2.1.4.2 Recommended solution to resolve weakness (Supersense)

Weaknesses	Recommended Solution
Require subscription plan	Develop for free-to-use purposes
Does not include AI chat for assistance	Implement an AI chatbot to assist user

2.1.5 Google Lookout

Figure 2.1.5.1 Google Lookout Logo

Google Lookout is a visual assistance app to help people with vision loss in the world [8]. Google Lookout is available on Android platforms. The purpose of this app is to make it easier to get information about the world around us and do daily task more efficiently like reading text and documents [8]. It is also an apps that build in collaboration with the blind and low-level vision community.

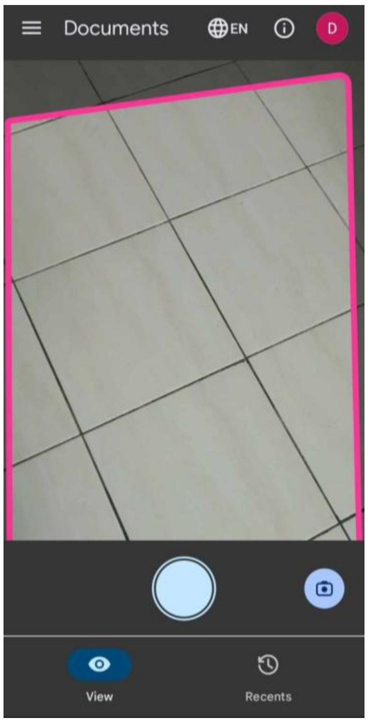


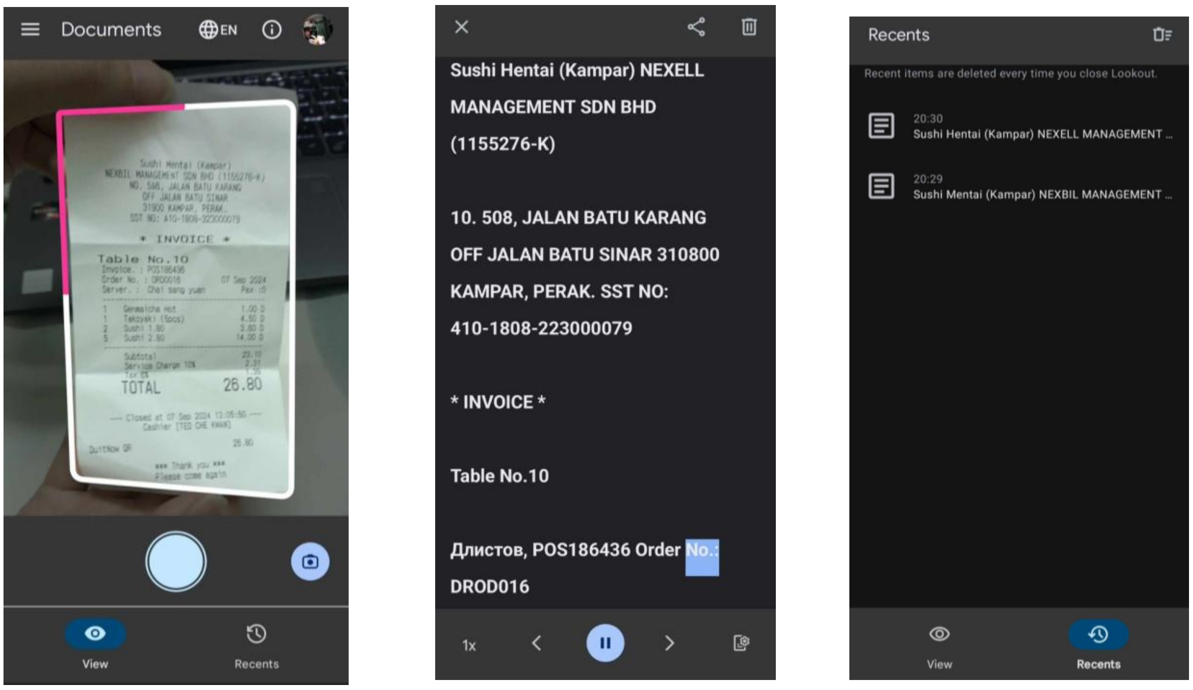
Figure 2.1.5.2 Document reader interface (Google Lookout)

Strength of Google Lookout:

Google Lookout provides a strength where is has real-time automatic capture and processing image . In this case, visually impaired users do not need to press any buttons to take a picture, as the app captures and processes the images automatically, and able to provide voice feedback within seconds. This real-time ability significantly improves the user experience by eliminating the need for manual actions to capture documents.

In addition, Google Lookout also offers a user-friendly interface. In the interface the user can simply adjust the language for document and text recognition in the document interface needed to navigate through the setting menu. Furthermore, when user captures a document, it able to change the font setting such as enlarging or line spacing depending on their personal needs. This flexibility enhances accessibility and usability for the visually impaired.

Another advantage of Google Lookout is that it able to autosave the document in an app when the user reads a document. It had an additional interface where visually impaired users were able to locate their previous recognized document in the 'recent' interface. Therefore, it is a convenient and flexible way to revisit important files without the need for manual saving for visually impaired users.



Sushi Hentai (Kampar) NEXELL MANAGEMENT SDN BHD (1155276-K)

10. 508, JALAN BATU KARANG OFF JALAN BATU SINAR 310800 KAMPAR, PERAK. SST NO: 410-1808-223000079

* INVOICE *

Table No.10

Длистов, POS186436 Order No.: DROD016

Recent items are deleted every time you close Lookout.

20:30
Sushi Hentai (Kampar) NEXELL MANAGEMENT ...

20:29
Sushi Mentai (Kampar) NEXBIL MANAGEMENT ...

Figure 2.1.5.3 Autosave the document in app (Strength of Google Lookout)

Weaknesses of Google Lookout:

However, a significant limitation is where it only can be downloaded on android platform. This limitation restricts access for users who rely on IOS operating systems. For visually impaired individuals who use iPhones, the inability to access this app may hinder their access to efficient document reading and recognition tools. Therefore, this shortcoming highlights the need for expanding compatibility to other platforms to enhance the app accessibility and usability for a broader range of visually impaired users.

A notable drawback is the absence of an AI chat feature for assistance. This limitation indicates that visually impaired users cannot engage with a virtual AI assistant via text or voice message. This deficiency of AI chat reduces the application ability to provide extra support and assistance to visually impaired users to clarify their queries, thus negatively affect the overall experience for visually impaired users.

Table 2.1.5.1 Strength and Weakness (Google Lookout)

Strength	Weaknesses
• Real-time automatic capture and processing image • User-friendly interface • Autosave the document in app when user read a document	• Only can be downloaded on android platform • Does not include AI chat assistance

Table 2.1.5.2 Recommended solution to resolve weakness (Google Lookout)

Weaknesses	Recommended Solution
Only can be downloaded on android platform	Develop on both IOS and Android platform
Does not include AI chat assistance	Implement an AI chatbot to assist user

2.2 Comparison of Previous Works

Table 2.2.1 Comparison Table of Previous Works

Applications / **Features**	Seeing AI	Be My Eyes	Envision AI	Supersense	Google Lookout	Proposed Solution
Platform Compatibility	iOS & Android	iOS & Android	iOS & Android	iOS & Android	Android	iOS & Android
Required Subscription				✓		
Offline	✓		✓	✓	✓	✓
Voice Capturing Assistance	✓	✓	✓	✓	✓	✓
Real-time automatic capture	✓				✓	✓
Real-time audio feedback	✓		✓	✓	✓	✓
AI chat for assistance	✓	✓	✓			✓
Support Speech-To-Text			✓			✓
Display and Voice Out the text/document recognized	✓		✓	✓	✓	✓
Friendly user interface			✓		✓	✓
Recognize and voice out text languages other than English			✓		✓	✓

2.3 Technical Review of Related Work

2.3.1 Text Detection and Recognition Using OCR

Previous work from Y. He [14]:

In the study of *"Research on Text Detection and Recognition Based on OCR Recognition Technology"* [14], the authors investigate the application of Optical Character Recognition (OCR) for automated text extraction from document images. The paper evaluates traditional OCR techniques, Tesseract alongside deep learning-based approaches, emphasizing improvements in accuracy and robustness for complex layouts. A hybrid method combining connected-component analysis for text detection and CNN-LSTM networks for recognition achieved 96.2% accuracy on a dataset of scanned documents.

2.3.2 Image based Text Translation using Firebase ML Kit

Previous work from V. Bagal et.al [15]:

The paper titled *"Image Based Text Translation using Firebase ML Kit"* presents a mobile application that utilizes Google's ML Kit for real-time text recognition and translation. Specifically, it employs the ML Kit Text Recognition API to extract textual information from images, including printed and handwritten text, across various backgrounds. This implementation showcases the robustness of Google ML Kit in handling diverse real-world scenarios, reinforcing its value in mobile-based OCR solutions.

2.3.3 Stamp Detection using different Yolo (You Only Look Once) model

Previous work from J. Bento et.al [16]:

In the study of "*Performance Evaluation of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 for Stamp Detection in Scanned Documents*", the studies indicates that YOLOv9s achieved the highest performance with a mean Average Precision (mAP) of 98.7% and a precision and recall of 97.6%. In terms of computational efficiency, YOLOv11s stood out for its lower computational cost and shorter inference time. These results suggest that while YOLOv8s performs adequately, newer versions like YOLOv9s and YOLOv11s offer improvements in accuracy and efficiency for stamp detection tasks.

Chapter 3

System Methodology

3.1 Methodology

The methodology that will be used in this project is **Prototyping Model**. Prototyping Model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved [9]. The phases are divided into 6 phases which are requirements gathering and analysis, quick design, build prototype, initial user evaluation, refining prototype and implementation and maintain. The phase of user evaluation and refining prototype can loop back as needed based on feedback and evolving requirements from users [10].

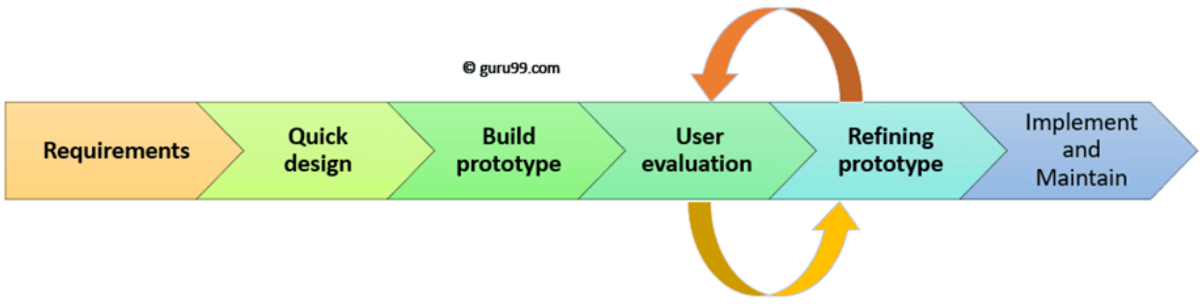


Figure 3.1.1 Flowchart of Prototyping Model Methodology [10].

The real-time document reader assistance app for visually impaired will be developed starting from:

1. **Requirement gathering and analysis phase**

   Identify the scope and primary goal of this mobile application, and the purpose is to assist visually impaired users by providing real-time document reading capabilities, enabling visually impaired users to access and interact with various types of documents without significant barriers. Additionally, requirements are gathered on how this app should perform and the essential features. Then, a proposal will be prepared to outline the project scope, features and specifications on this mobile application.

2. **Quick Design phase**

At this point, a basic outline of the system is developed. A wireframe is designed to construct a visual representation of the app's layout and structure to map out the user interface and interactions. It is not a full design but to give users and supervisors a general understanding of the system.

3. **Prototype Building**

In this phase, an actual prototype is designed based on the information gathered from quick design such as the User Interface and some button.

4. **Initial User Evaluation**

In this stage, the proposed system is presented to the user and supervisor for an initial evaluation. It helps to find out the strengths and weaknesses of the development application. Comment and suggestions are recorded for further improvement.

5. **Refining Prototype**

After gathering the suggested improvement, the protype is refines according to the user feedback. Once the user is satisfied with the developed prototype, a final system is then developed.

6. **Implementation and maintain**

Once the final system is developed based on the final prototype, it is thoroughly tested and deployed for public to use, and maintenance is performed such as version updates of the packages in application.

3.2 Technology Involved

- **Flutter framework**

  Flutter is an open-source framework that used for developing the mobile application for both iOS and Android platforms. It provides a cross-platform UI toolkit that enables the applications from a single codebase. It also provides functionality such as accessing mobile camera devices to manipulate on and off the flashlight.

- **Google ML Kit**

  Google ML kit is a mobile SDK that brings Google's machine learning expertise to Android and iOS application. It provides wide range of recognizing text, detecting faces scanning barcodes, labelling images, and identifying the language of text.

- **Roboflow**

  Roboflow is a powerful platform designed to help developers create, manage, and deploy computer vision models efficiently. It streamlines the process of plotting data annotation, training machine learning models, and allows them to deploy into embedded devices.

- **YOLOv8n (You Only Look Once) model**

  YOLOv8n is a real-time pretrained model that can be used for document detection in real-time and deploy in lightweight format and integrate into flutter application.

- **Language Translation API**

  An API which used for translating into multiple languages from a single languages text.

- **Flutter (TTS) (Text-to-Speech) Library**

  A plugin which used to provide real-time audio feedback and to convert the text into audio and voice out back to the user.

- **Flutter (STT) (Speech-to-text) Library**

  A plugin to convert user audio input and display into text for questioning to the AI chatbot assistance.

3.3 System Requirement

3.3.1 Hardware

The hardware involved in this project is a computer and two different platforms of mobile devices, Android & IOS. Computers are written and develop codes for the document reader application for Android and IOS. The mobile devices are used for testing the application real-time document and text recognition capabilities in different across various environments.

Table 3.3.1.1 Specifications of laptop

Description	Specifications
Model	VivoBook_ASUSLaptop X515DA_M515DA
Processor	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx
Operating System	Windows 11
Graphic	AMD Radeon (TM) Vega 8 Graphics
Memory	16GB
Storage	512 GB

Table 3.3.1.2 Specifications of mobile device (Android)

Description	Specifications
Model	CPH2121
Processor	Octa-core
Operating System	Android Version 12
Graphic	V12.1
Memory	8 GB
Storage	128 GB

Table 3.3.1.3 Specifications of mobile device (IOS)

Description	Specifications
Model	iPhone 13
Processor	A15 Bionic chip
Operating System	iOS 17.6.1
Graphic	Apple-designed 4-core GPU
Memory	4GB
Storage	256 GB

3.3.2 Software

The software and technology that will be used for this project is Visual Studio Code and Google Colab.

Visual Studio Code

Figure 3.3.2.1 Visual Studio Code Logo

Visual Studio Code is a flexible code editor that seamlessly combines simplicity with powerful developer tools for code completion and debugging [11]. Furthermore, it is also used to code and integrate the trained model into the app. The usage of Visual Studio Code in this project is for the flutter development in Dart Languages.

Google Colab

Figure 3.3.2.2 Google Colab Logo

Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. It can be used to develop in Python programming languages to train model.

Roboflow

Figure 3.3.2.3 Roboflow

Roboflow is a comprehensive platform designed to simplify the process of data annotation, training and deploying models.

TensorFlow Lite

Figure 3.3.2.4 TensorFlow Lite

TensorFlow Lite is a lightweight, open-source framework designed to run machine learning models on mobile and embedded devices [13].

3.3.2.1 Coding Language

- **Python**

  Python is a language that is used to develop and train machine learning models such as YOLOv8n in Google Colab for document detection.

- **Dart**

  Dart is a programming language used with Flutter to develop mobile applications including the user interface, camera access, flashlight, user interaction and Pub.dev plugins.

3.4 Ghant Chart

3.4.1 Ghantt Chart of FYP1

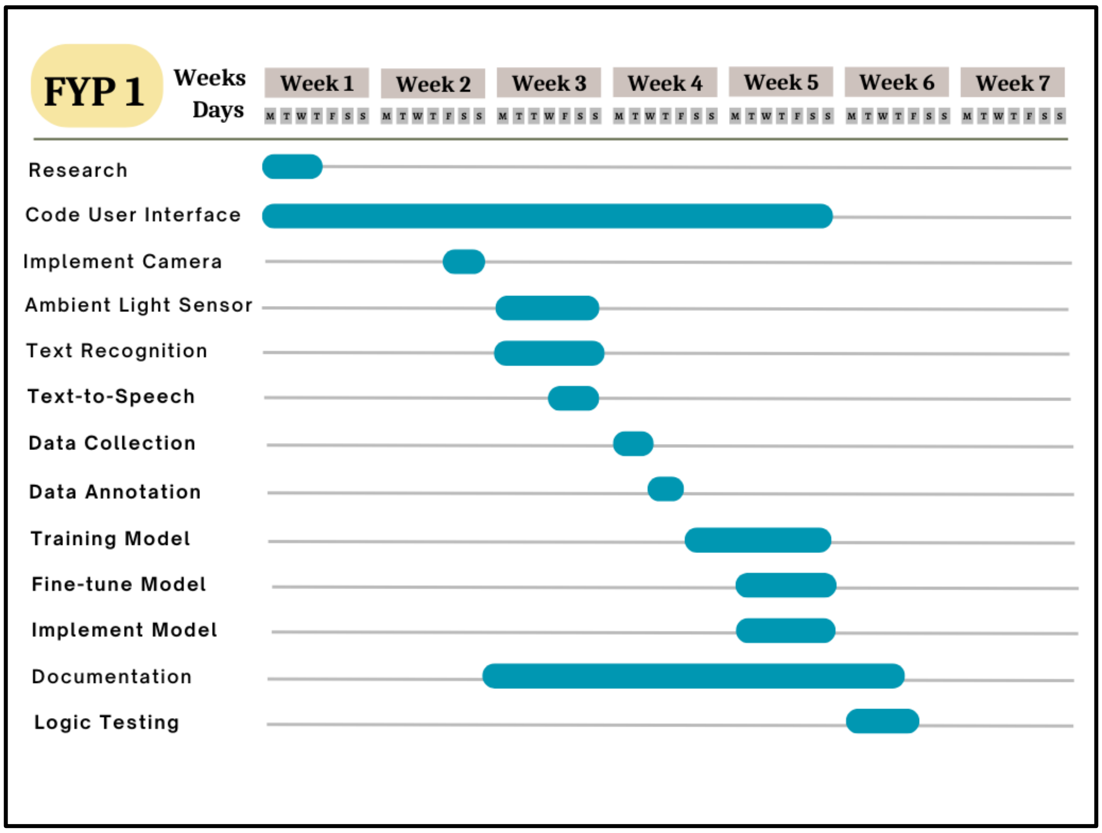


Figure 3.4.1 Ghant Chart of FYP1 Timeline

3.4.2 Ghantt Chart of FYP2

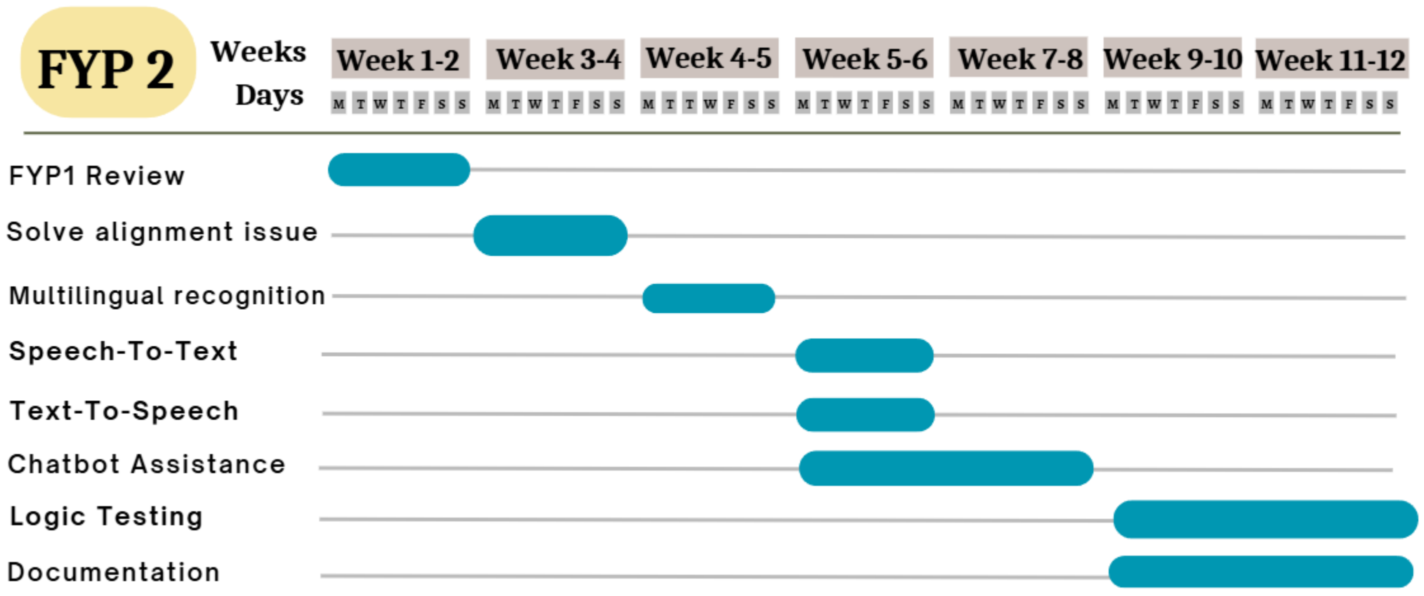


Figure 3.4.2 Ghant Chart of FYP2 Timeline

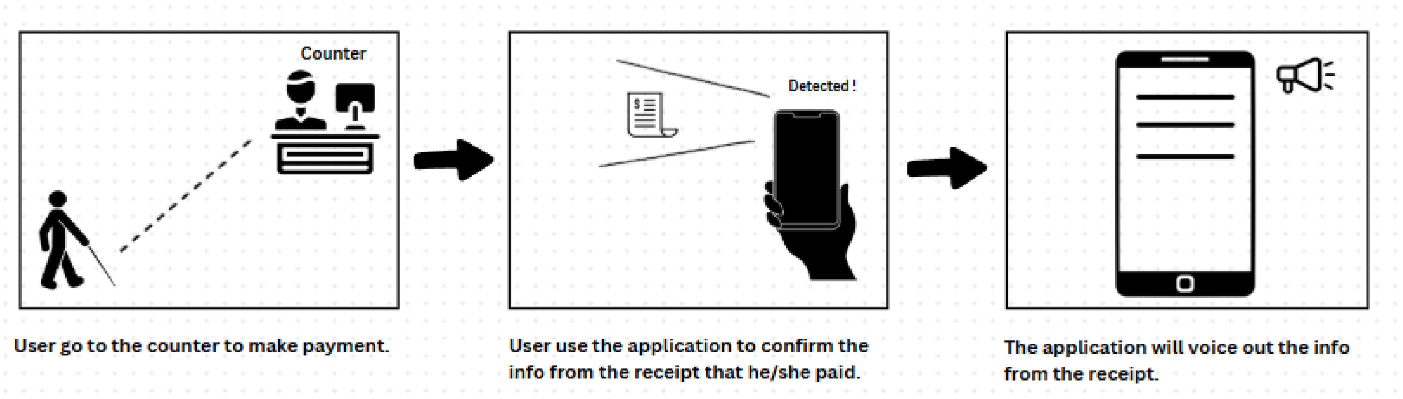
Chapter 4

System Design

4.1 User Storyboard

Scenario: An app to allow users to access various types of documents independently.

4.1.1 User Story 1



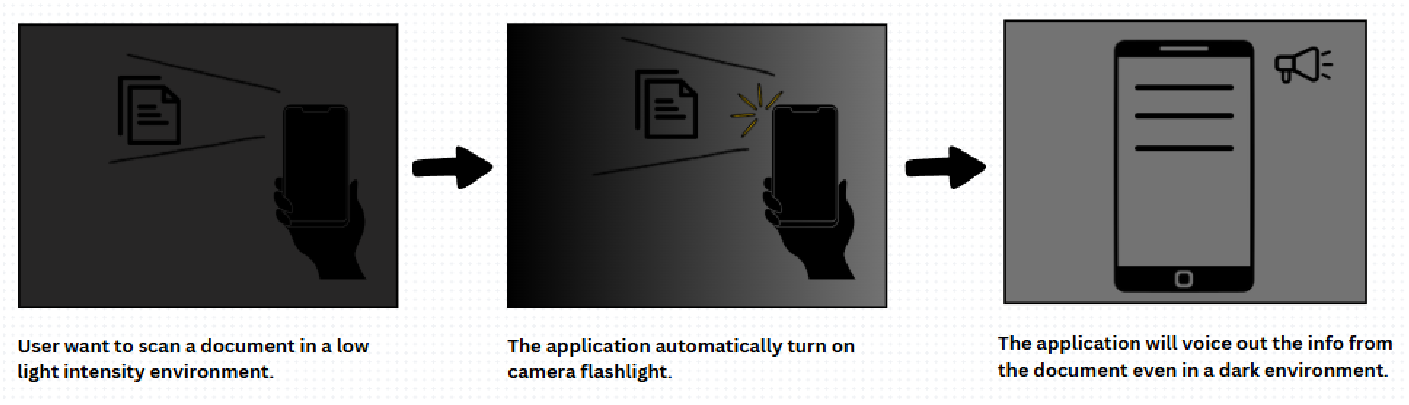
User go to the counter to make payment.

User use the application to confirm the info from the receipt that he/she paid.

The application will voice out the info from the receipt.

Figure 4.1.1 User Story 1

4.1.2 User Story 2



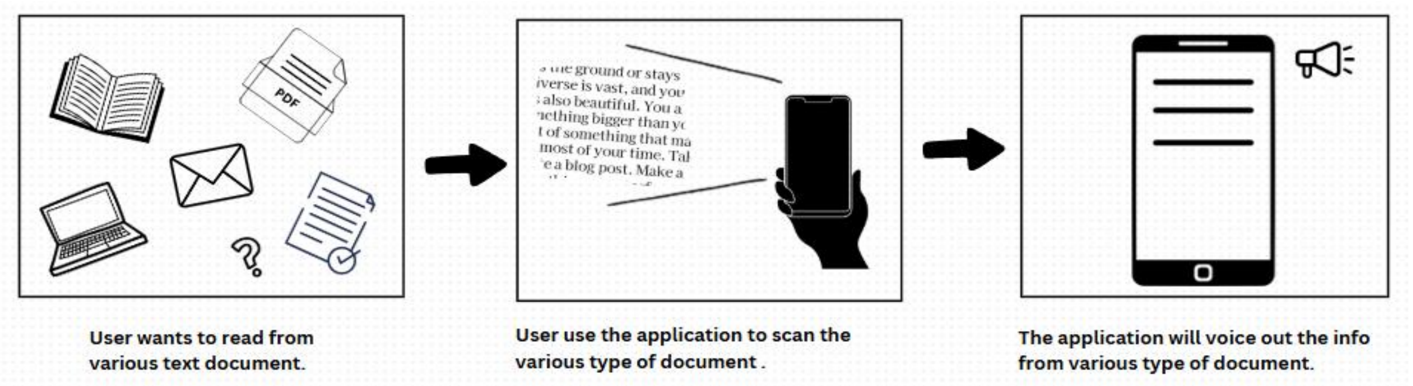
User want to scan a document in a low light intensity environment.

The application automatically turn on camera flashlight.

The application will voice out the info from the document even in a dark environment.

Figure 4.1.2 User Story 2

4.1.3 User Story 3



User wants to read from various text document.

User use the application to scan the various type of document .

The application will voice out the info from various type of document.

Figure 4.1.3 User Story 3

4.2 Overall System Flowchart

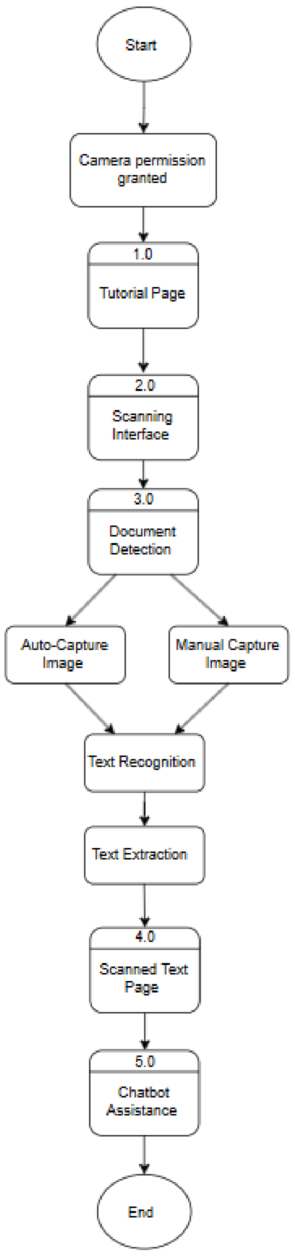


Figure 4.2.1 Overall System Flowchart

Figure 4.2.1 above outlines the overall system functionality of the real-time document reader mobile application. Initially, when the user starts the application, it will ask for the camera permission to access the application. Upon successful granted permission, a tutorial page will be explained on this application and how it works. After that, users will enter the scanning interface which is the main interface of the application. In this interface, users can adjust the setting upon pressing a button and adjust based on their personal preferences. Next, user can place any document to scan and perform text extraction. Simultaneously, a real-time document detection will be assisting users to detect any document that are present in the camera. If a document exists, it will auto capture after a few seconds of delay to ensure the documents are completely focused. If user wants to scan some short text but is not likely to be a document, user can also manually capture without relying on the auto-capture feature Then, a text recognition will be used to extract the text from images, and the text will be display and voice out to user in the scanned text page. If the user has any questions to ask or needs clarification about the scanned text of its content, user can interact with the chatbot assistance to obtain the information about the scanned text.

Sub-Flowcharts

4.2.1 Tutorial Page

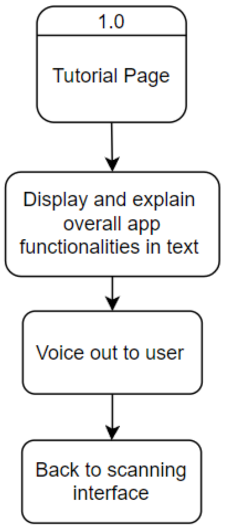

1.0
Tutorial Page
Display and explain overall app functionalities in text
Voice out to user
Back to scanning interface

Figure 4.2.1.1 Sub-Flowcharts of Tutorial Page

4.2.2 Scanning Interface

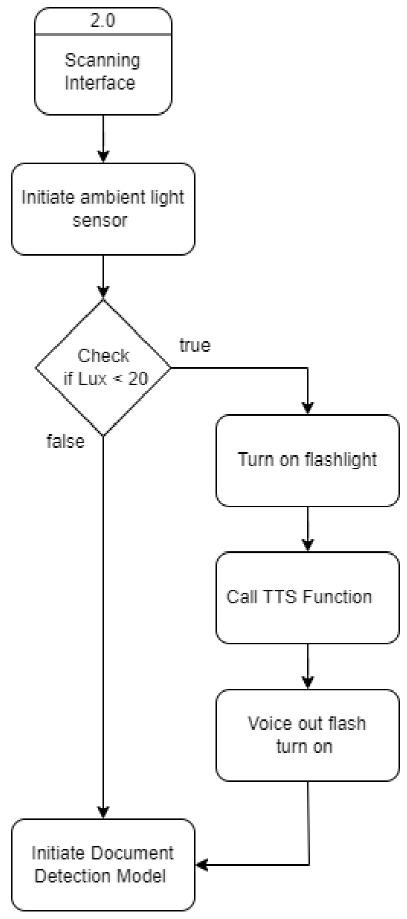


Figure 4.2.2.1 Sub-Flowcharts of Scanning Interface

4.2.3 Document Detection

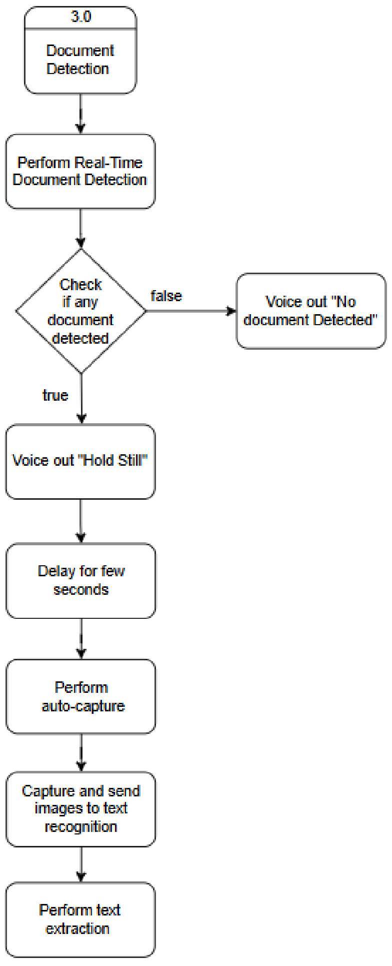

3.0
Document Detection
Perform Real-Time Document Detection
Check if any document detected
false
Voice out "No document Detected"
true
Voice out "Hold Still"
Delay for few seconds
Perform auto-capture
Capture and send images to text recognition
Perform text extraction

Figure 4.2.3.1 Sub-Flowcharts of Document Detection

4.2.4 Scanned Text Page

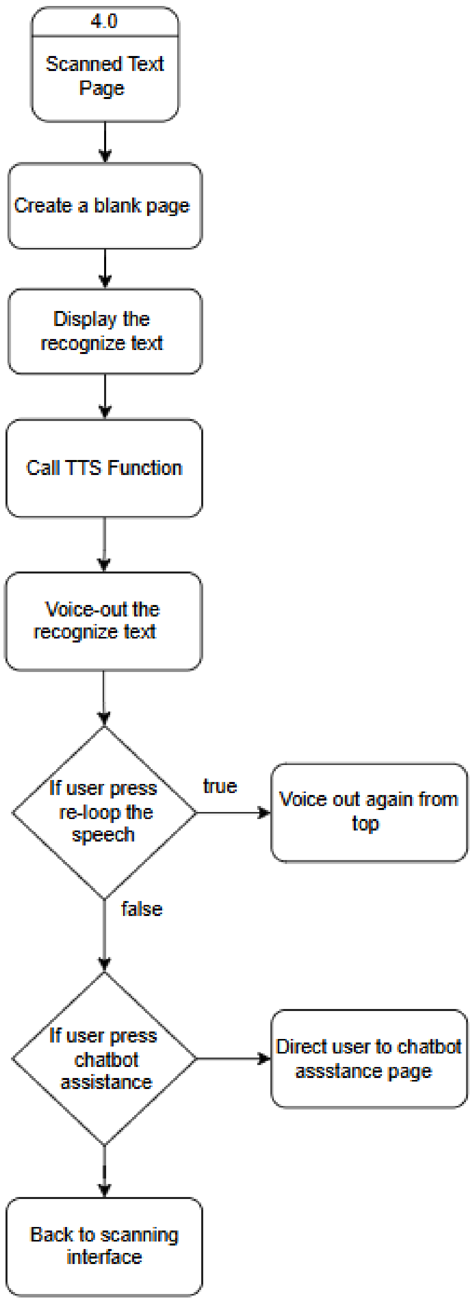

4.0
Scanned Text Page
Create a blank page
Display the recognize text
Call TTS Function
Voice-out the recognize text
If user press re-loop the speech
true
Voice out again from top
false
If user press chatbot assistance
Direct user to chatbot assstance page
Back to scanning interface

Figure 4.2.4.1 Sub-Flowcharts of Scanned Text Page

4.2.5 Chatbot Assistance

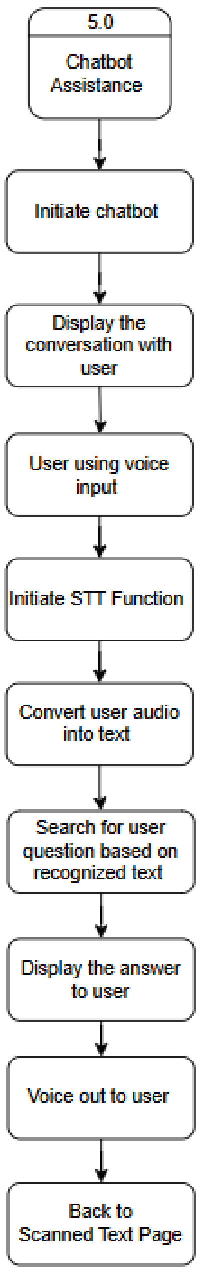

5.0
Chatbot Assistance
Initiate chatbot
Display the conversation with user
User using voice input
Initiate STT Function
Convert user audio into text
Search for user question based on recognized text
Display the answer to user
Voice out to user
Back to Scanned Text Page

Figure 4.2.5.1 Sub-Flowcharts of Chatbot Assistance

4.3 Overall System Architecture Diagram

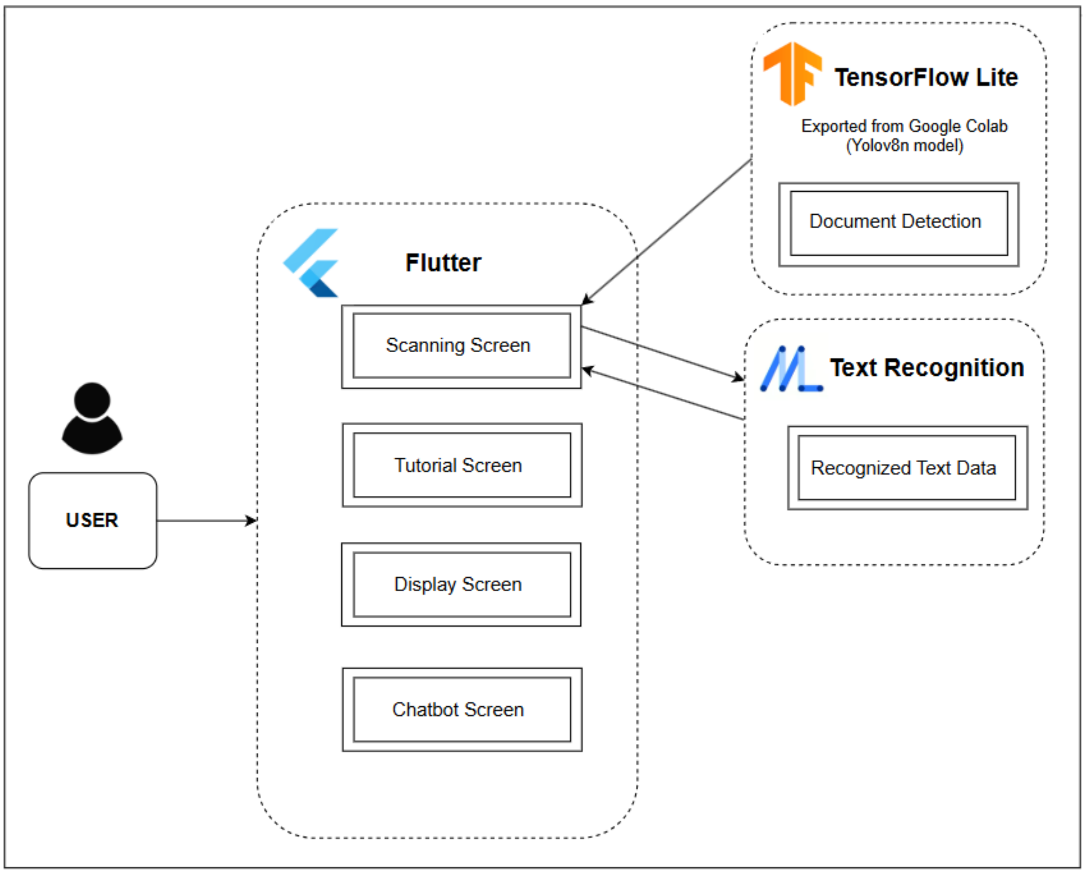


Figure 4.3.1 Overall System Architecture Diagram

Figure 4.3.1 shows the overall system architecture. Where the user acts as the primary interaction with the system. Users can interact with the application through scanning screens. Tutorial screen, display screen and chatbot screen. In the scanning screen. It interacts with the TensorFlow Lite for initiating document detection using a pre-trained YOLOv8n model that exported from Google Colab. Furthermore, it also interacts with Google ML Kit Text Recognition to extract text data from the detected document. It will pass the image data that capture using the document detection model to process the text recognition, and a recognized text data will be returned to user. Then, a chatbot will be assisting user where user provide a question to the chatbot and the chatbot will search the keyword and return to the user.

4.3.1 Yolov8n System Architecture Diagram

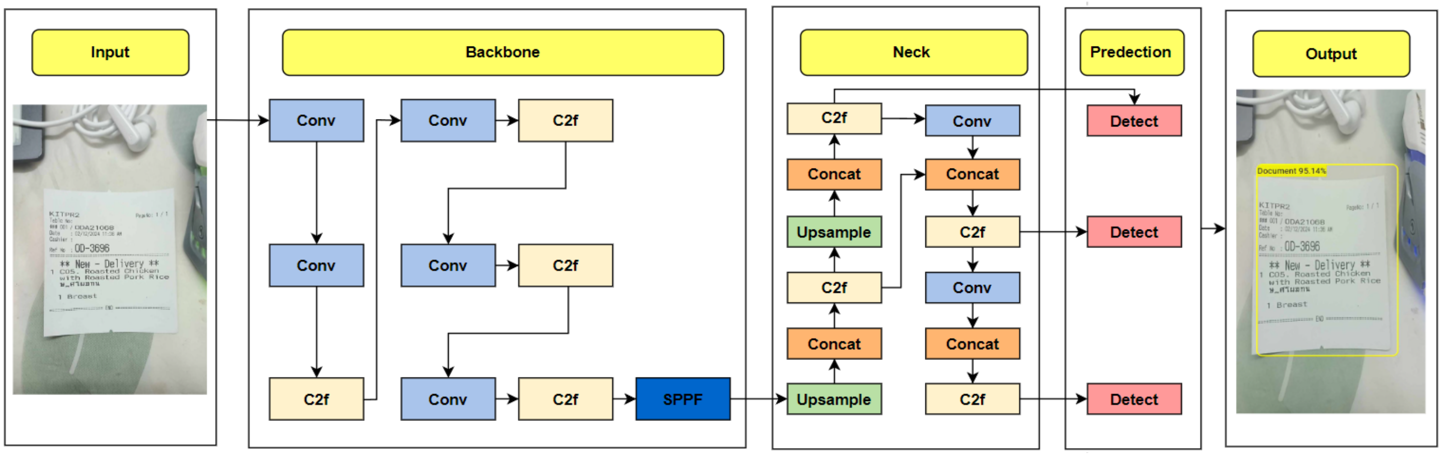


Figure 4.3.1.1 Yolov8n Object Detection System Architecture Diagram

Figure 4.3.1.1 above shows the Yolov8n Object Detection System Architecture in five stages, which is the input, backbone, beck, prediction and output. Initially at the input stage, this stage receives an input image such as a photo or a frame from video. The image size is resized to a fixed resolution such as (640x640) to standardize the input dimensions.

Next, in the second stage, the backbone will extract the features from the input images where it will be divided into multiple blocks. The first one is the convolutional layer where basic convolutional layers are used to detect low-level features such as edges and textures. Next is the C2f blocks will split into two parts, the first part is where a portion of the input bypasses the main computation and the second part will be sent through a series of transformations such as convolutional layers, batch normalization and activation functions. Lastly, the processes features will be concatenated with the bypassed feature from the first part which combines the low-level and high-level feature to enhance the feature map. Then, it will pass to the convolutional layer to refine the combined feature map and optimize it for the next stage.

In the last stage of the backbone, an SPPF will act as a bottleneck that outputs the compressed feature representation before passing it to the **neck** for further processing. In the neck stage, it involves sampling up the handle fine-grained details for small objects and a concatenation to merge feature maps from different levels. Furthermore, the C2f blocks, and convolutional layers are used to refine aggregated features to capture object at different scale such as small, medium and large.

Lastly, the refined features from the neck are processed into the prediction head stage which involves detect block for making the final detections such as outline bounding boxes, classes prediction and confidence score. Finally, the predicted result will be shown in the output stage.

4.3.2 Google ML Kit Text Recognition Architecture Diagram

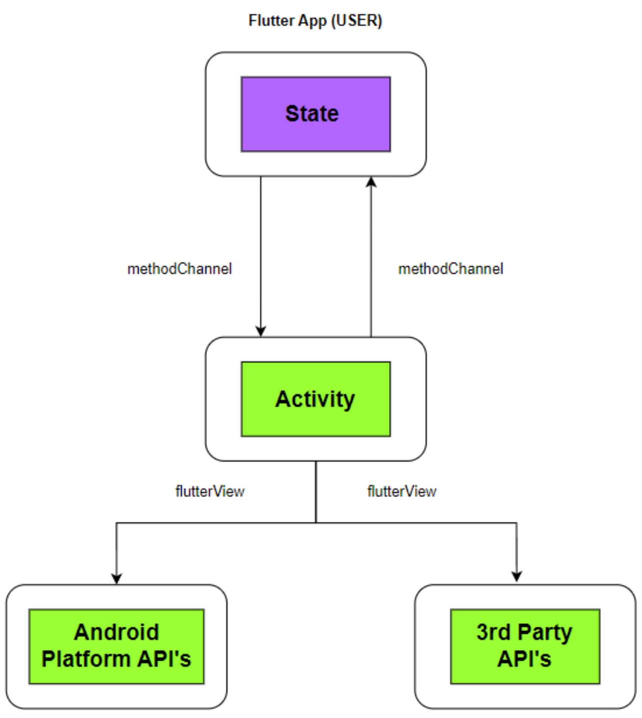


Figure 4.3.2.1 Google ML Kit Text Recognition Architecture Diagram

From Figure 4.3.2.1 above, initially flutter app acts as the front-end UI and user interaction layer. This flutter app uses a methodChannel to communicate with the native code. Furthermore, the method Channel is a two-way communication bridge between the flutter application and the native host code. For example, the flutter app will send the request such as "process camera image" to the platform layer via the methodChannel. Then, the platform layer will process the request using native API's and sends the result back to the flutter app. The activity component acts as the entry point for the native code. Then, the android platform API's and 3rd party API's handle the camera frame and process the text recognition using google ML Kit text recognition. Finally, the recognized text is sent back to the Flutter App through the Method Channel.

4.4 Use-Case Diagram

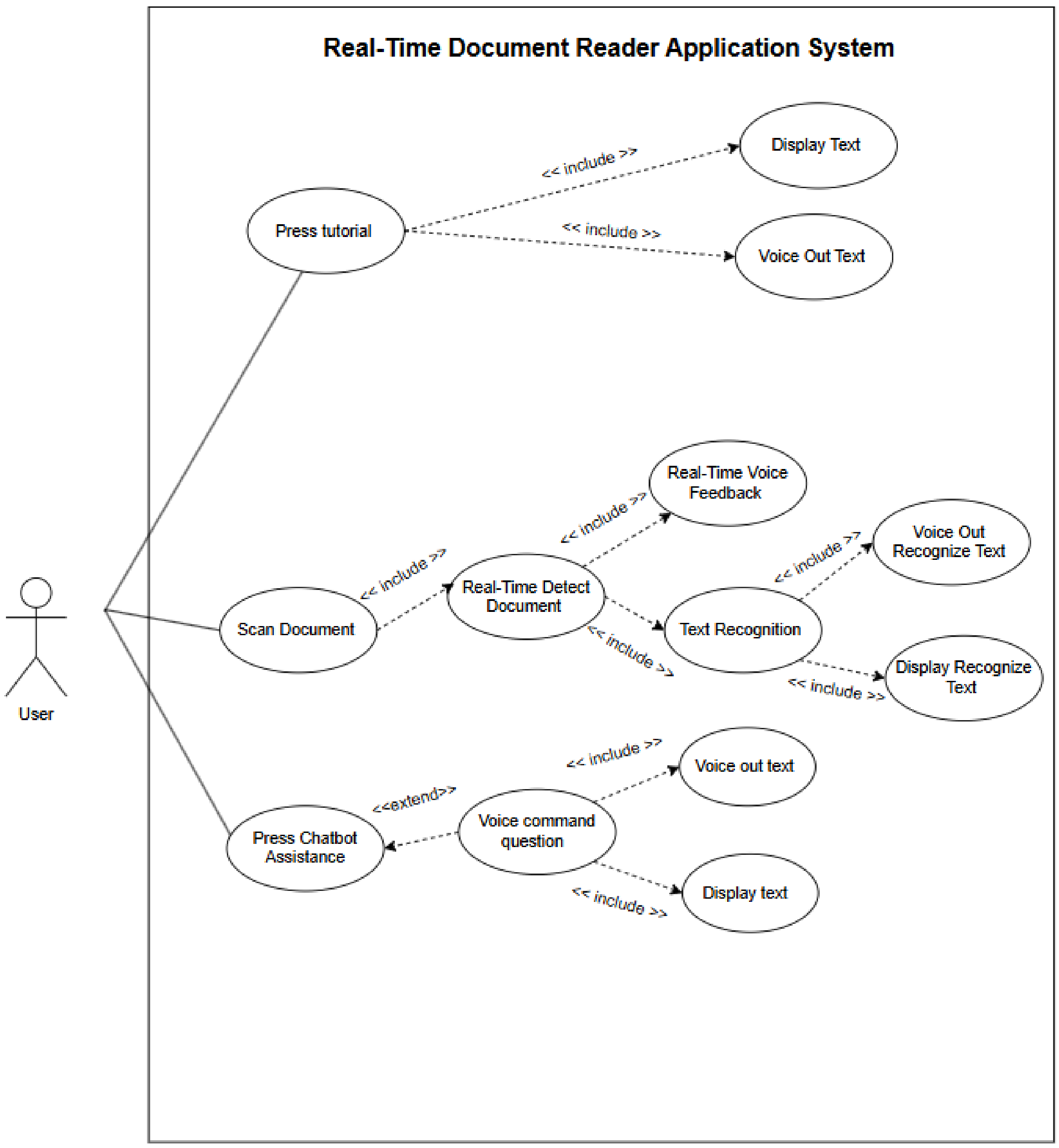


Figure 4.4.1 Use-Case Diagram

4.5 Activity Diagram

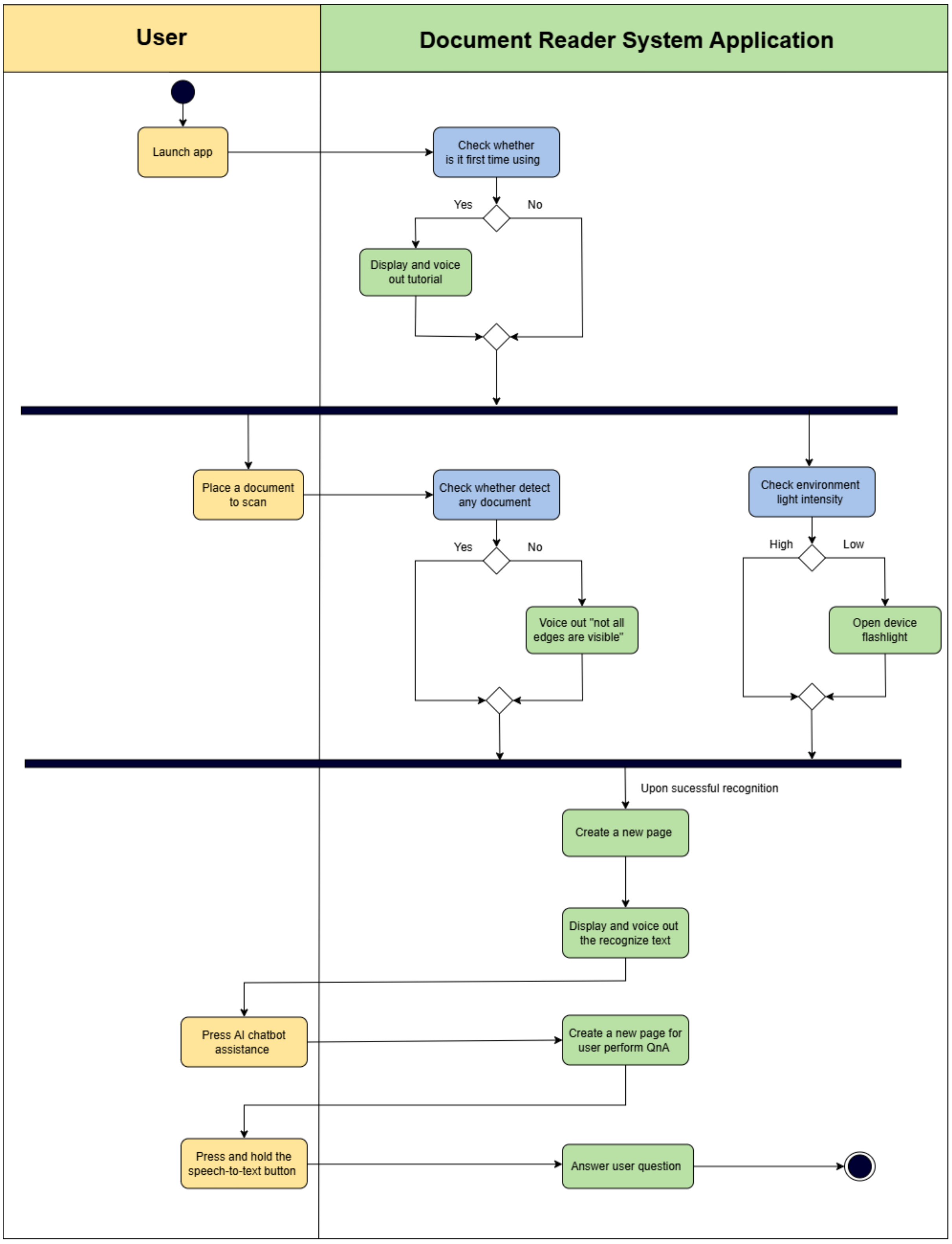

User
Document Reader System Application
Launch app
Check whether is it first time using
Yes
No
Display and voice out tutorial
Place a document to scan
Check whether detect any document
Yes
No
Voice out "not all edges are visible"
Check environment light intensity
High
Low
Open device flashlight
Upon sucessful recognition
Create a new page
Display and voice out the recognize text
Press AI chatbot assistance
Create a new page for user perform QnA
Press and hold the speech-to-text button
Answer user question

Figure 4.5.1 Activity Diagram

4.6 User Interface Design

4.6.1 Wireframes Prototype Design

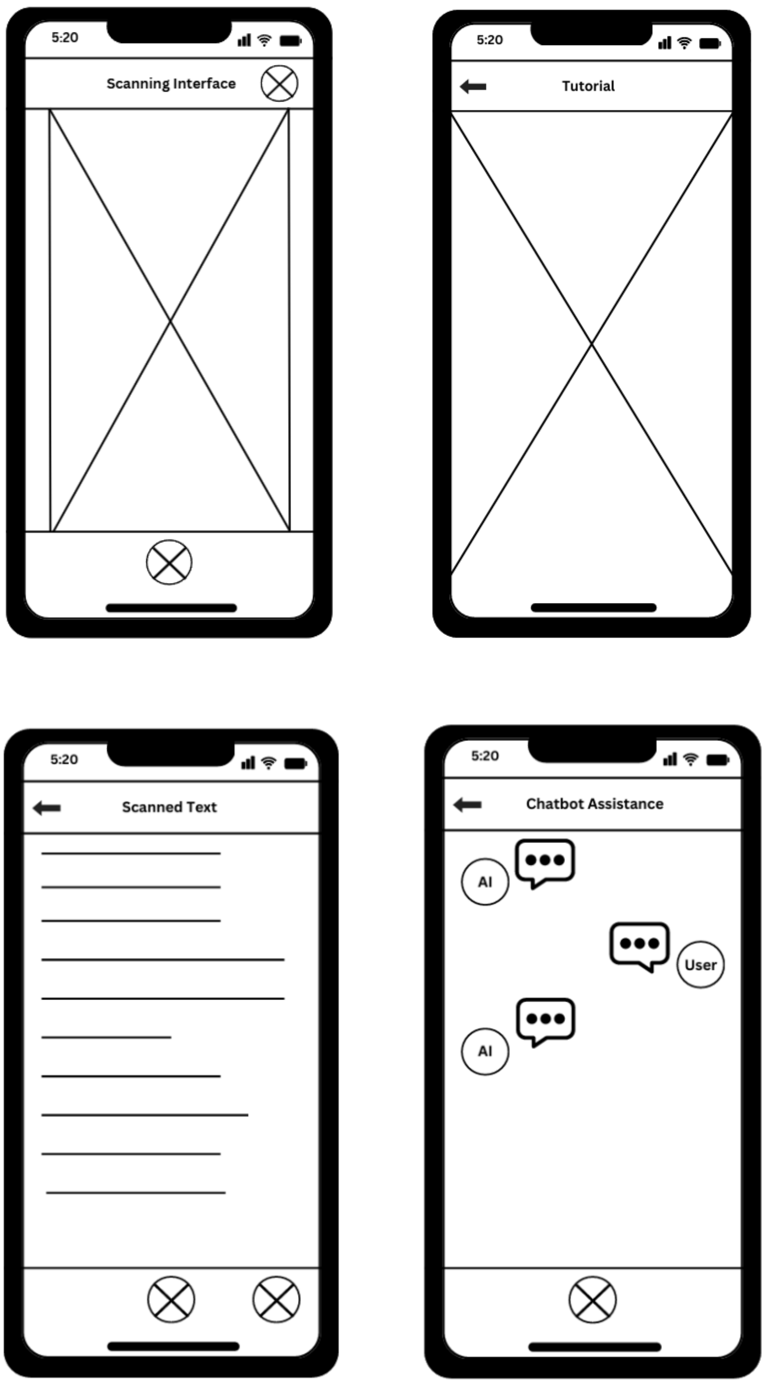


Figure 4.6.1.1 Wireframe Prototype Design

4.6.2 Low-Fidelity Prototype Design

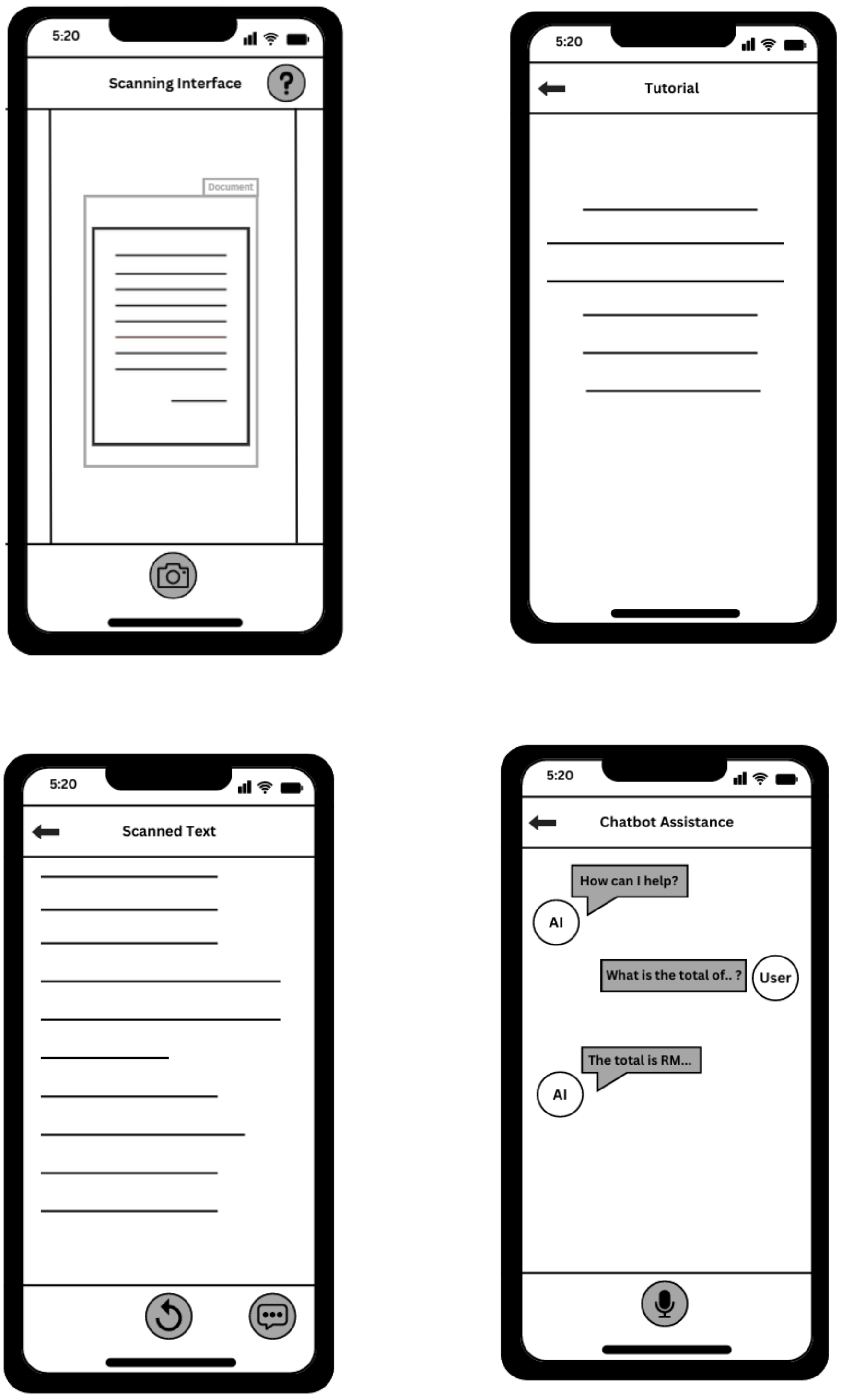
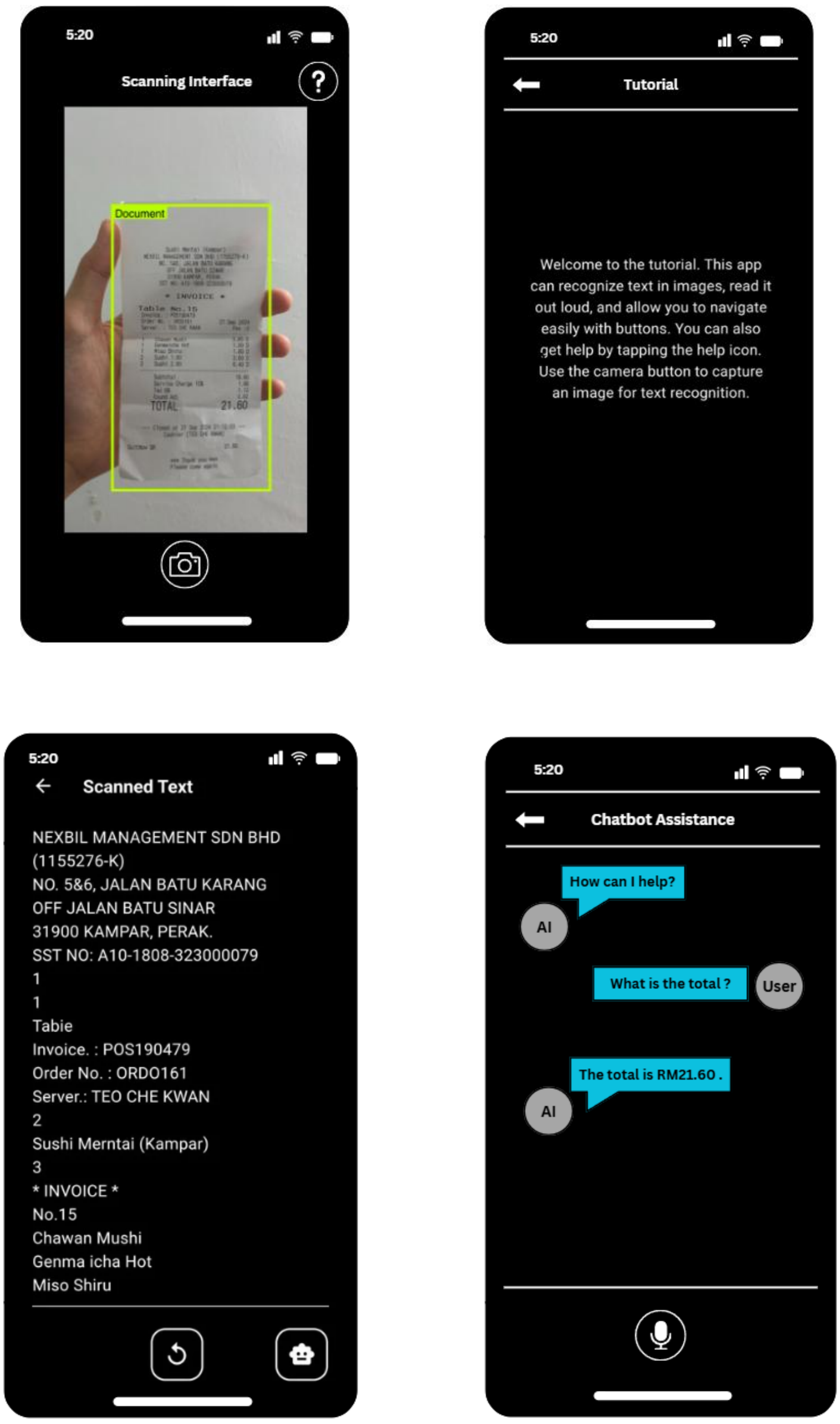


Figure 4.6.2.1 Low-fidelity Prototype Design

4.6.3 High Fidelity Prototype Design



Welcome to the tutorial. This app can recognize text in images, read it out loud, and allow you to navigate easily with buttons. You can also get help by tapping the help icon. Use the camera button to capture an image for text recognition.

NEXBIL MANAGEMENT SDN BHD
(1155276-K)
NO. 5&6, JALAN BATU KARANG
OFF JALAN BATU SINAR
31900 KAMPAR, PERAK.
SST NO: A10-1808-323000079
1
1
Tabie
Invoice. : POS190479
Order No. : ORDO161
Server.: TEO CHE KWAN
2
Sushi Merntai (Kampar)
3
* INVOICE *
No.15
Chawan Mushi
Genma icha Hot
Miso Shiru

How can I help?
AI

What is the total ?
User

The total is RM21.60 .
AI

Figure 4.6.3.1 High-fidelity Prototype Design

Chapter 5

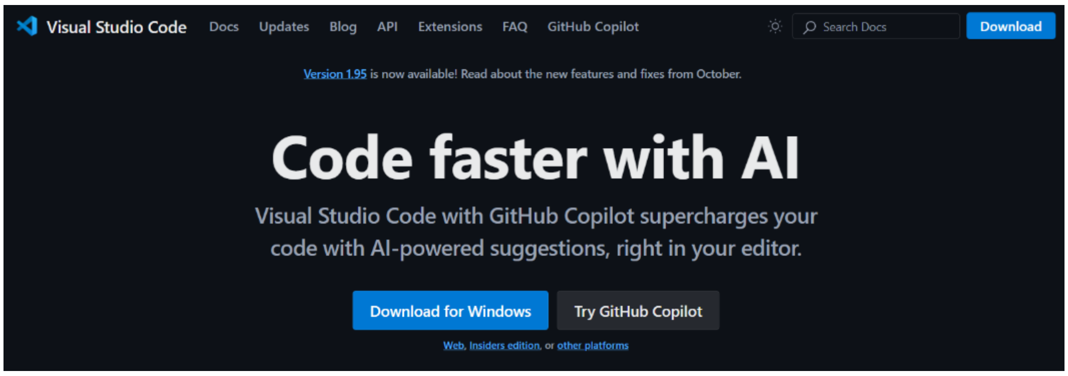
System Implementation

5.1 Setting Up

5.1.1 Software

Before starting the development of real time document reader, there are three main software needed to be installed and downloaded in my laptop:

1. Visual Studio Code
2. Android Studio
3. Flutter

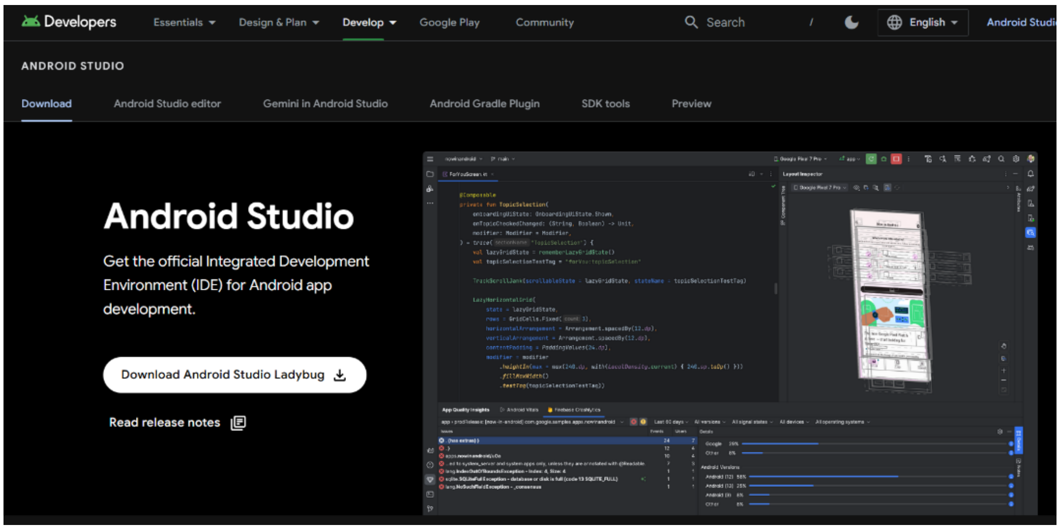


Version 1.95 is now available! Read about the new features and fixes from October.

Code faster with AI

Visual Studio Code with GitHub Copilot supercharges your code with AI-powered suggestions, right in your editor.

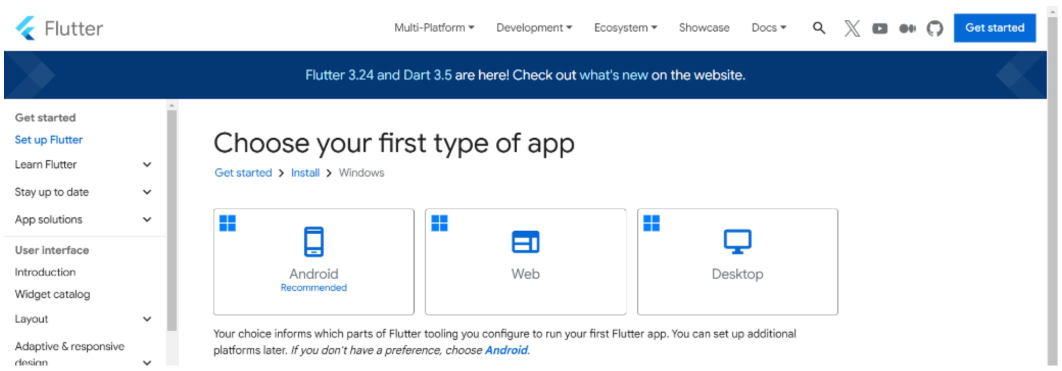
Figure 5.1.1.1 Installation Page of VS Code



Android Studio

Get the official Integrated Development Environment (IDE) for Android app development.

Figure 5.1.1.2 Installation Page of Android Studio



Flutter 3.24 and Dart 3.5 are here! Check out what's new on the website.

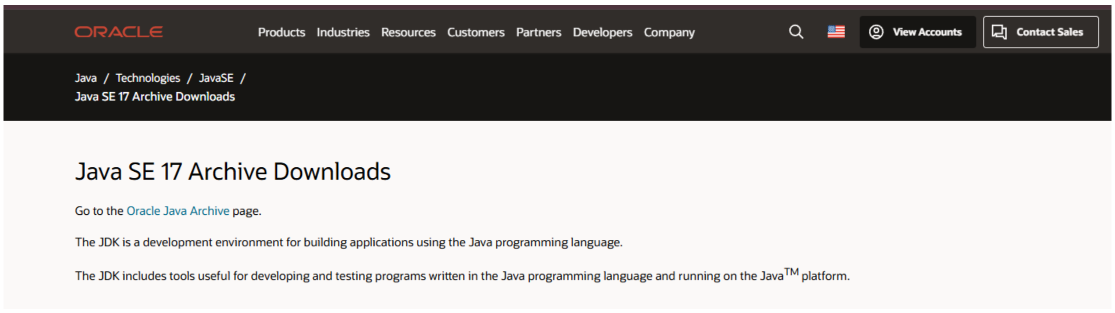
Choose your first type of app

Get started > Install > Windows

Your choice informs which parts of Flutter tooling you configure to run your first Flutter app. You can set up additional platforms later. *If you don't have a preference, choose* ***Android.***

Figure 5.1.1.3 Installation Page of Flutter

After finishing installation of these 3 software, we need to download java-jdk-17 to be running in my laptop environment because Android toolchain in Flutter relies on the Java Development Kit (JDK) for android development, Gradle dependency and ensure compatibility and stability. After unzipping the files, we created a JAVA_HOME path directory and set up jdk-17 in our local PC environment.



Java / Technologies / JavaSE /
Java SE 17 Archive Downloads

Java SE 17 Archive Downloads

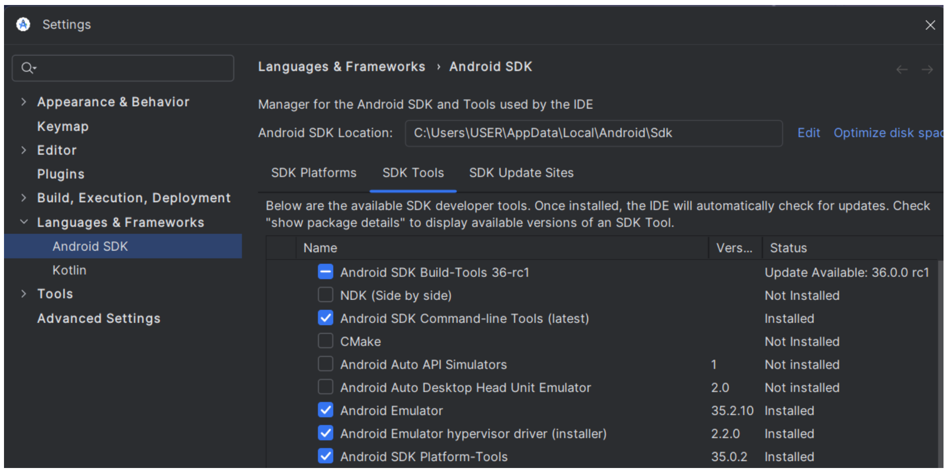
Go to the Oracle Java Archive page.

The JDK is a development environment for building applications using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java^{TM} platform.

Figure 5.1.1.4 Installation Page of Java-jdk-17

After finishing setup up the environment, we need to setup the emulator and the android SDK Command-line Tool for for building, testing, and debugging Android applications.



Languages & Frameworks › Android SDK

Manager for the Android SDK and Tools used by the IDE

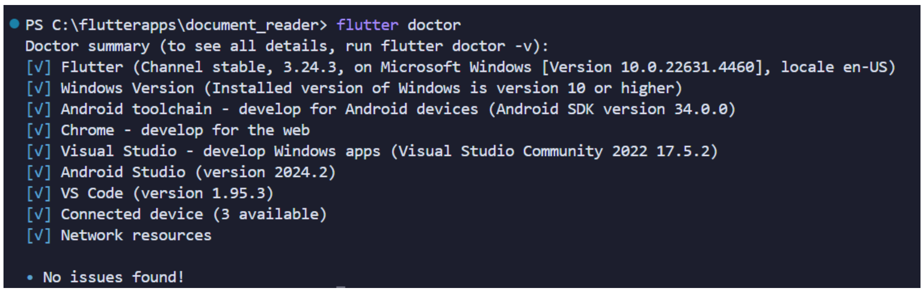
Android SDK Location: C:\Users\USER\AppData\Local\Android\Sdk

Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

	Name	Vers...	Status
▬	Android SDK Build-Tools 36-rc1		Update Available: 36.0.0 rc1
☐	NDK (Side by side)		Not Installed
☑	Android SDK Command-line Tools (latest)		Installed
☐	CMake		Not Installed
☐	Android Auto API Simulators	1	Not installed
☐	Android Auto Desktop Head Unit Emulator	2.0	Not installed
☑	Android Emulator	35.2.10	Installed
☑	Android Emulator hypervisor driver (installer)	2.2.0	Installed
☑	Android SDK Platform-Tools	35.0.2	Installed

Figure 5.1.1.5 Setup in Android Studio

After successful setting up in the android studio, we will be verifying the environment in VS Code. We use a command “flutter doctor” to check the necessary packages and tools had installed in our environment.



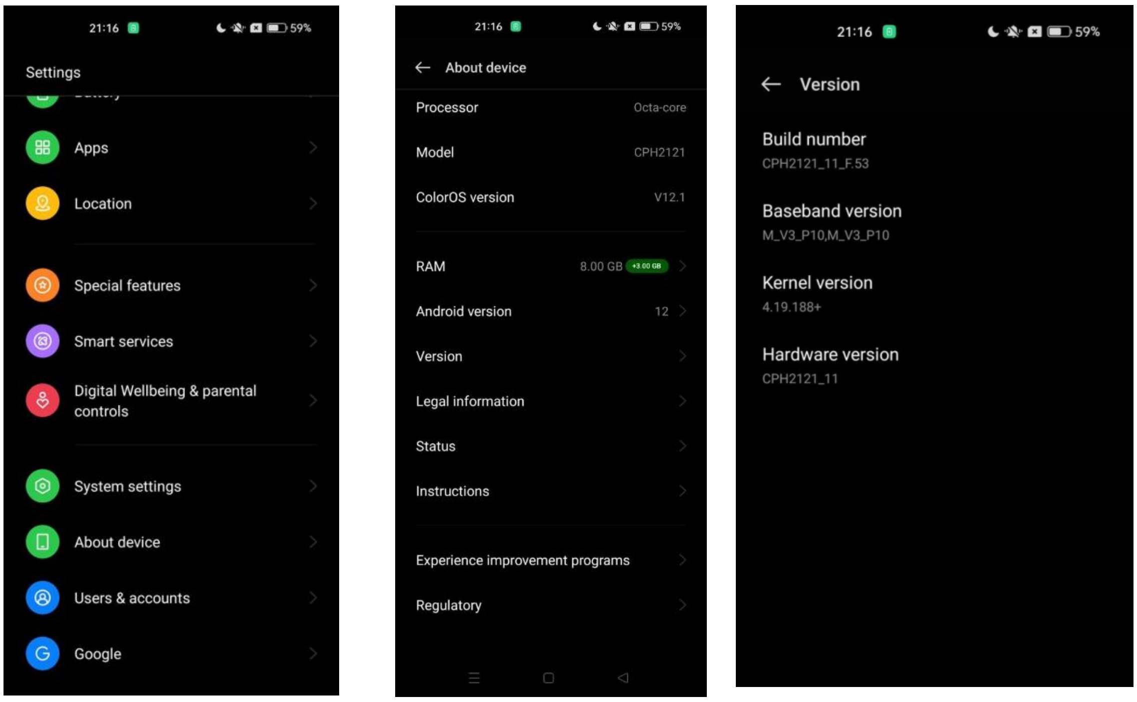
```
● PS C:\flutterapps\document_reader> flutter doctor
  Doctor summary (to see all details, run flutter doctor -v):
  [√] Flutter (Channel stable, 3.24.3, on Microsoft Windows [Version 10.0.22631.4460], locale en-US)
  [√] Windows Version (Installed version of Windows is version 10 or higher)
  [√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
  [√] Chrome - develop for the web
  [√] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.5.2)
  [√] Android Studio (version 2024.2)
  [√] VS Code (version 1.95.3)
  [√] Connected device (3 available)
  [√] Network resources

  • No issues found!
```

Figure 5.1.1.6 Packages and Tools Verification in VS Code

If it returns no issue found, that means our setup using flutter in VS Code had successfully completed and we can proceed to the development of application.

5.1.2 Hardware

After setting up in software, we need to perform some setting in our hardware which is the mobile devices to debug in real devices.



```
21:16                59%
Settings
Apps
Location
Special features
Smart services
Digital Wellbeing & parental controls
System settings
About device
Users & accounts
Google
```

```
21:16                59%
← About device
Processor               Octa-core
Model                   CPH2121
ColorOS version         V12.1
RAM                     8.00 GB +3.00 GB
Android version         12
Version
Legal information
Status
Instructions
Experience improvement programs
Regulatory
```

```
21:16                59%
← Version
Build number
CPH2121_11_F.53
Baseband version
M_V3_P10,M_V3_P10
Kernel version
4.19.188+
Hardware version
CPH2121_11
```

Figure 5.1.2.1 Setup on real devices

We go to Setting > About Devices > Version > and tap the Build number 7 times to enable developer mode to debug in real hardware mobile devices. After successful enabling, we turn on the USB debugging in the developer options in our setting, and it is successful.

5.2. Pub dev Flutter Plugins

5.2.1 Camera and flashlight

The first major feature that we want to implement is the camera. We go to Pub.dev and search for a camera plugin.

pub.dev

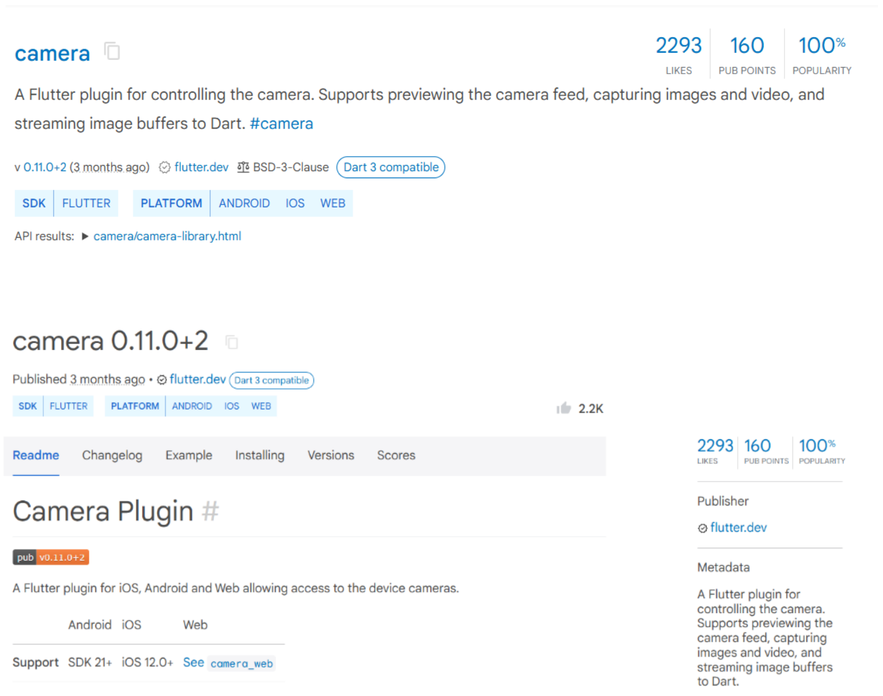
Search packages

The official package repository for Dart and Flutter apps.

Supported by Google

Figure 5.2.1.1 Pub.dev official website

Then, we search the camera plugin and implement in our code.



camera

2293 LIKES | 160 PUB POINTS | 100% POPULARITY

A Flutter plugin for controlling the camera. Supports previewing the camera feed, capturing images and video, and streaming image buffers to Dart. #camera

v 0.11.0+2 (3 months ago) flutter.dev BSD-3-Clause Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS WEB

API results: ▸ camera/camera-library.html

camera 0.11.0+2

Published 3 months ago • flutter.dev Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS WEB 2.2K

Readme Changelog Example Installing Versions Scores

2293 LIKES | 160 PUB POINTS | 100% POPULARITY

Camera Plugin #

pub v0.11.0+2

A Flutter plugin for iOS, Android and Web allowing access to the device cameras.

	Android	iOS	Web
Support	SDK 21+	iOS 12.0+	See camera_web

Publisher

flutter.dev

Metadata

A Flutter plugin for controlling the camera. Supports previewing the camera feed, capturing images and video, and streaming image buffers to Dart.

Figure 5.2.1.2 Pub.dev camera plugin

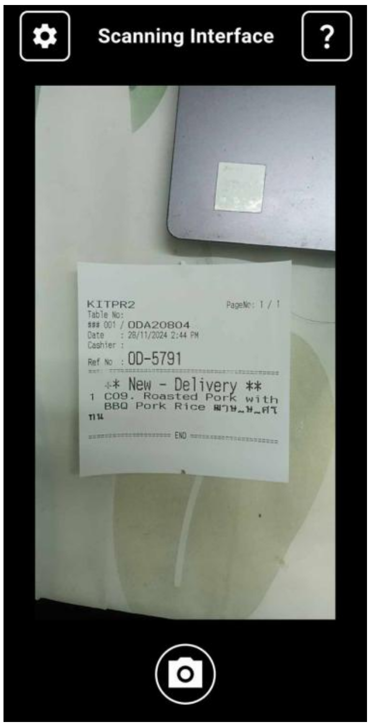
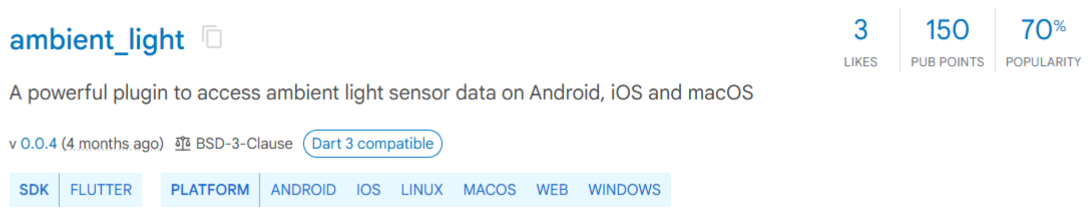


Figure 5.2.1.3 Scanning Interface with camera preview

5.2.2 Ambient Light Sensor

After successful implementing the camera preview in our code and we able to see the camera preview, we then proceed to implement the ambient light sensor. Our purpose is to allow the flashlight to turn on automatically whenever the light level (lux) is < 20 using the implementation of this sensor. Therefore, users can scan documents even in a dark environment.



ambient_light

3 LIKES | 150 PUB POINTS | 70% POPULARITY

A powerful plugin to access ambient light sensor data on Android, iOS and macOS

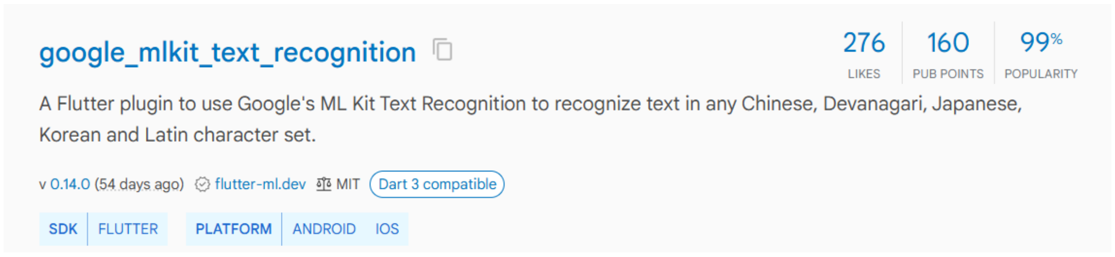
v 0.0.4 (4 months ago) BSD-3-Clause Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

Figure 452.2.1 Pub.dev ambient light sensor plugin

5.2.3 Google ML Kit Text Recognition

Following next, we implement an OCR google ml kit text recognition from pub dev to extract the text when user capture and display out.



google_mlkit_text_recognition

276 LIKES | 160 PUB POINTS | 99% POPULARITY

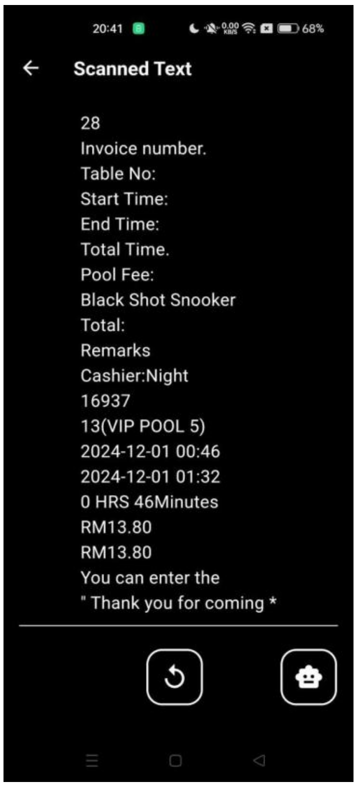
A Flutter plugin to use Google's ML Kit Text Recognition to recognize text in any Chinese, Devanagari, Japanese, Korean and Latin character set.

v 0.14.0 (54 days ago) flutter-ml.dev MIT Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS

Figure 5.2.3.1 Pub.dev google ml kit text recognition plugin

After successful implementation we were able to recognize the text and display to user.

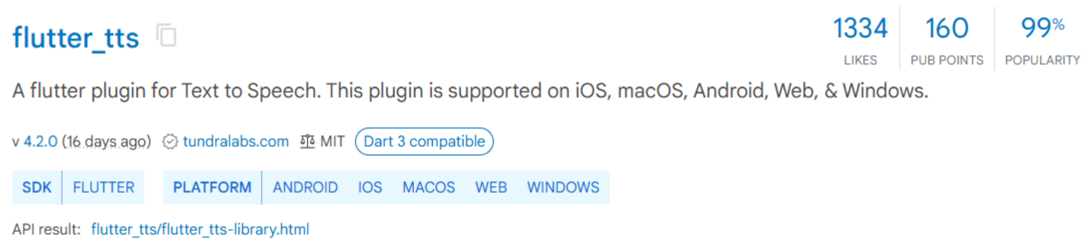


```
Scanned Text

28
Invoice number.
Table No:
Start Time:
End Time:
Total Time.
Pool Fee:
Black Shot Snooker
Total:
Remarks
Cashier:Night
16937
13(VIP POOL 5)
2024-12-01 00:46
2024-12-01 01:32
0 HRS 46Minutes
RM13.80
RM13.80
You can enter the
" Thank you for coming *
```

Figure 5.2.3.2 Scanned text using google ml kit text recognition

5.2.4 TTS

Upon successful recognition of text based on document, we implement a flutter TTS to voice out the recognized text to the visually impaired user.



flutter_tts

1334 LIKES | 160 PUB POINTS | 99% POPULARITY

A flutter plugin for Text to Speech. This plugin is supported on iOS, macOS, Android, Web, & Windows.

v 4.2.0 (16 days ago) tundralabs.com MIT Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS MACOS WEB WINDOWS

API result: flutter_tts/flutter_tts-library.html

Figure 5.2.4.1 Pub.dev flutter TTS plugin

5.3 Data Collection

After implementing the necessary plugin, we then proceed to implement the document detection model in our mobile application. Initially we collect the receipt that the text is in a white background as the data to train our model. We capture every receipt in different angles, light intensity and background environment to avoid our model overfitting.

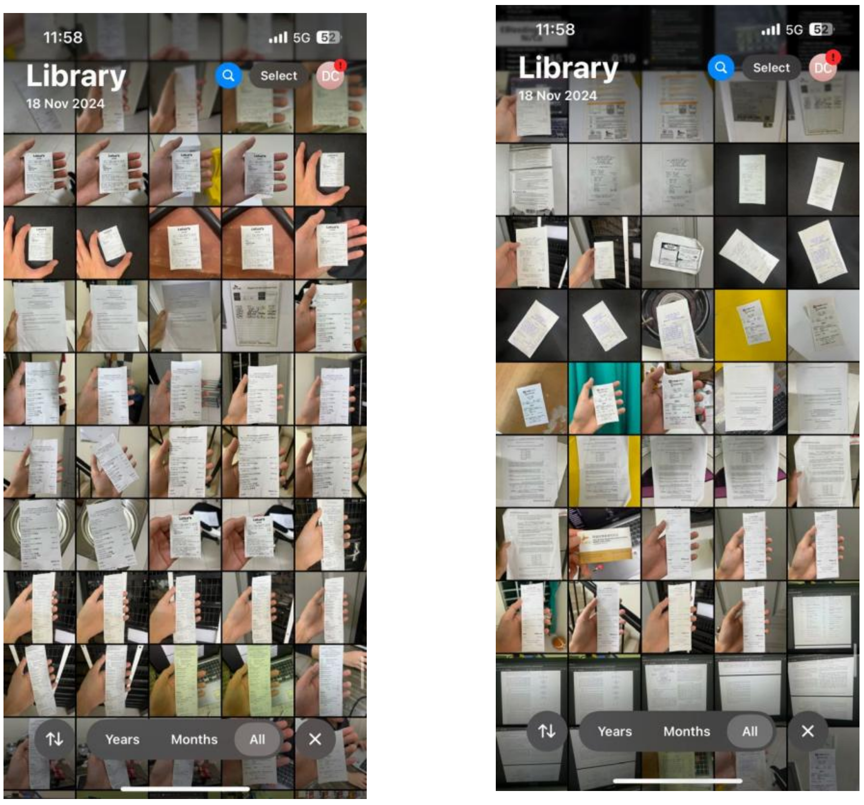
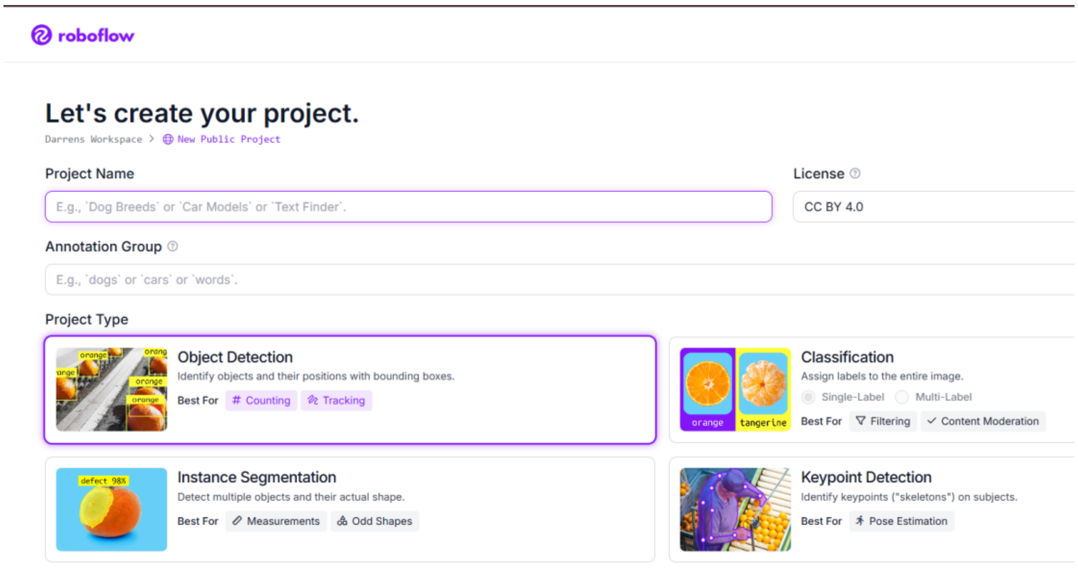


Figure 5.3.1 Data collection of white background documents

After collecting all the datasets, we create a new project in Roboflow and select object detection as our aim is for document detection.



roboflow

Let's create your project.

Darrens Workspace > New Public Project

Project Name: E.g., 'Dog Breeds' or 'Car Models' or 'Text Finder'.

License: CC BY 4.0

Annotation Group: E.g., 'dogs' or 'cars' or 'words'.

Project Type

Object Detection
Identify objects and their positions with bounding boxes.
Best For: Counting, Tracking

Classification
Assign labels to the entire image.
Single-Label, Multi-Label
Best For: Filtering, Content Moderation

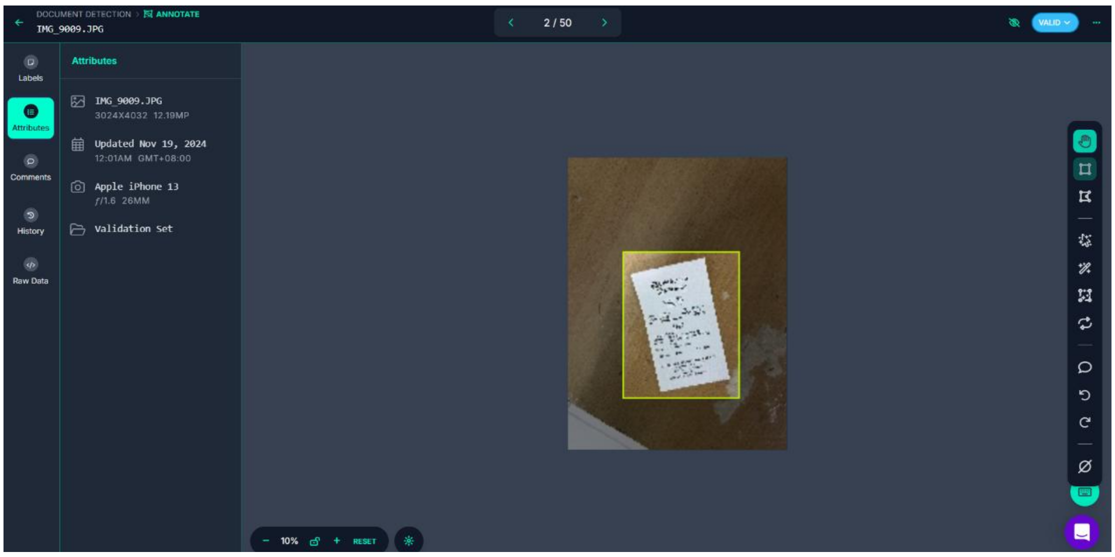
Instance Segmentation
Detect multiple objects and their actual shape.
Best For: Measurements, Odd Shapes

Keypoint Detection
Identify keypoints ("skeletons") on subjects.
Best For: Pose Estimation

Figure 5.3.2 Object Detection Project

5.4 Data Annotation

After creating an object detection project, we insert all our dataset into Roboflow and use it to annotate our dataset and label all document in a ‘document’ class.



DOCUMENT DETECTION > ANNOTATE
IMG_9009.JPG
2 / 50
VALID

Attributes
IMG_9009.JPG
3024X4032 12.19MP
Updated Nov 19, 2024
12:01AM GMT+08:00
Apple iPhone 13
f/1.6 26MM
Validation Set

Figure 5.4.1 Data annotation of document

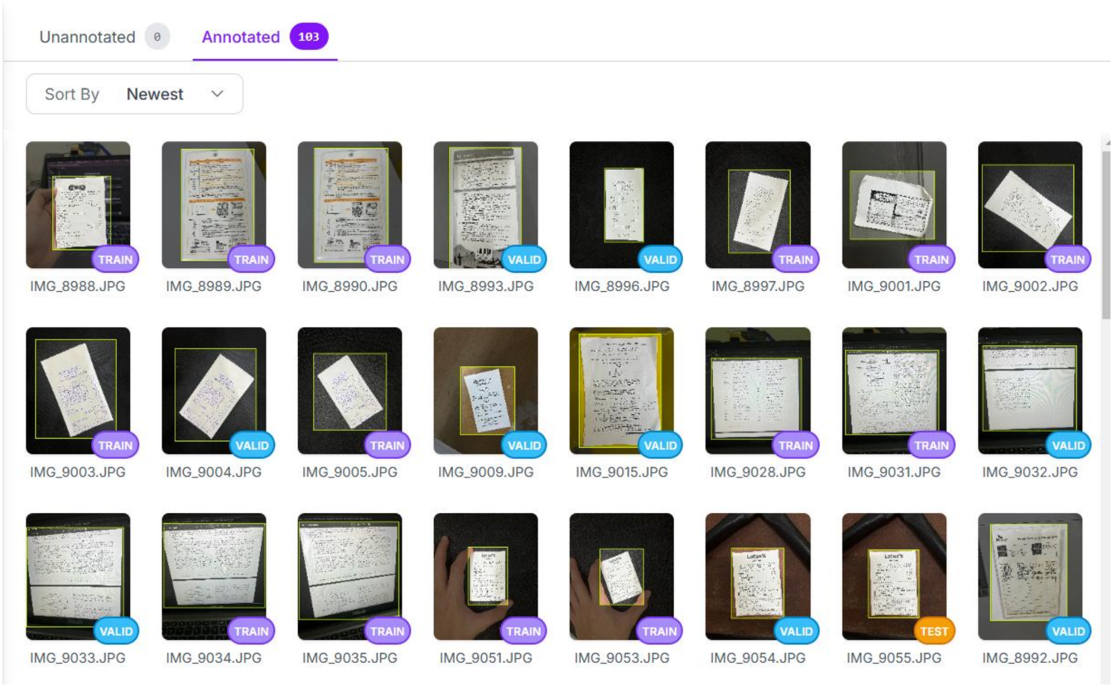


Figure 5.4.2 Annotated Document

5.5 Data Augmentation

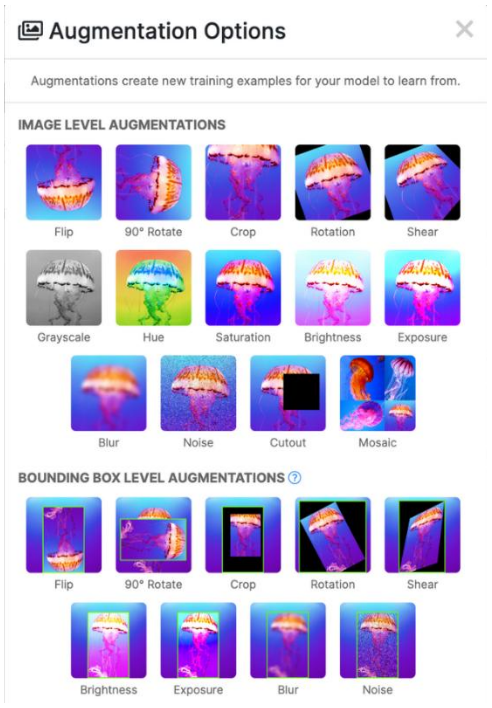


Figure 5.5.1 Data Augmentation

After successful data annotation, we perform few data augmentation such as applying the brightness and exposure adjustments to make the model more robust to real-world scenarios.

5.6 Data Splitting

After finishing data augmentation, we split the data into train(0.7), valid(0.2) and test (0.1) set using Roboflow.

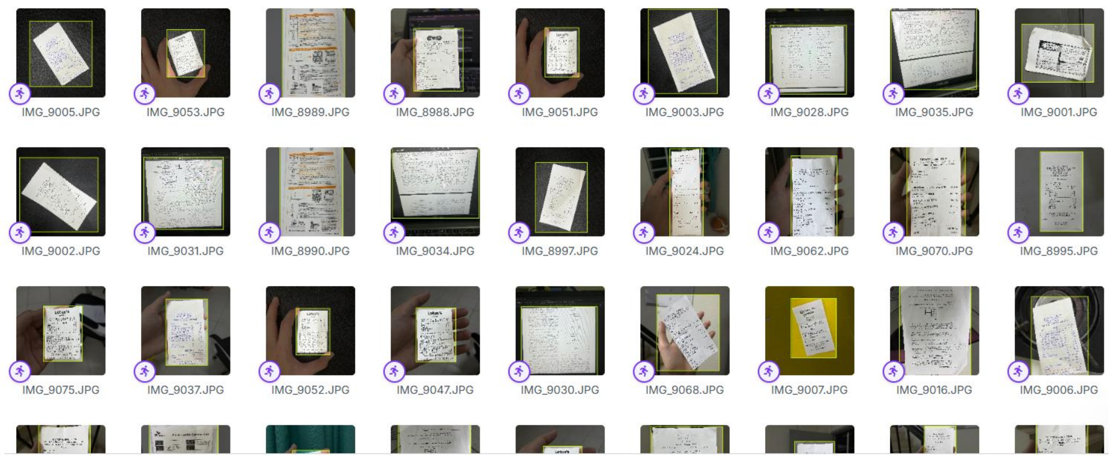


Figure 5.6.1 Training set data

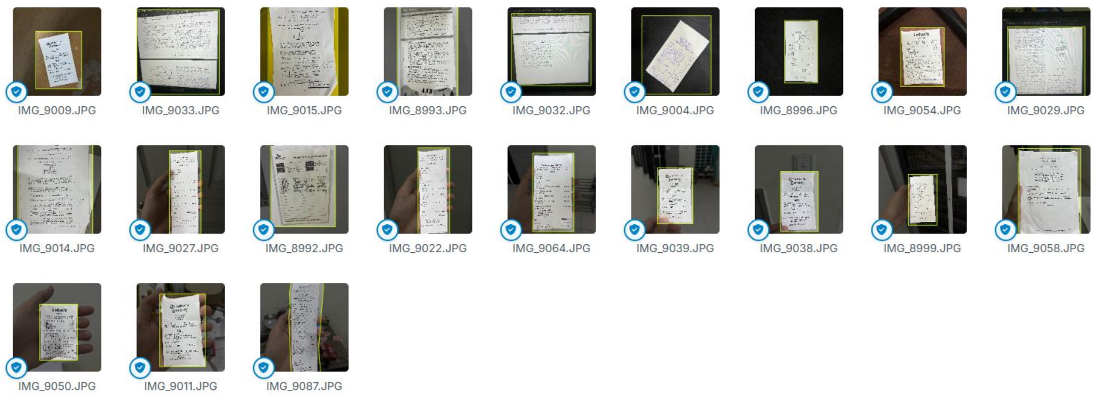


Figure 5.6.2 Valid set data

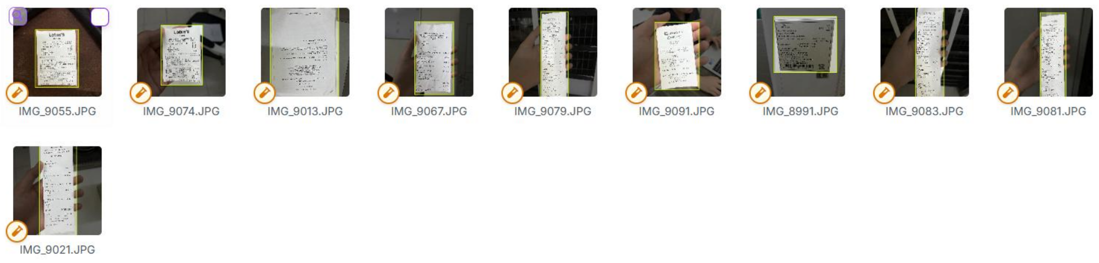
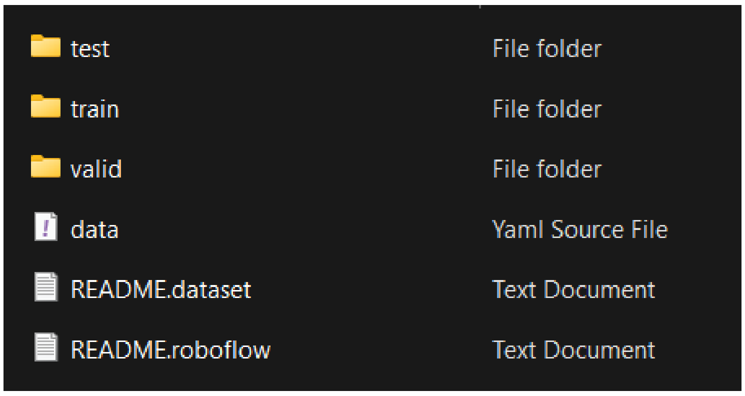


Figure 5.6.3 Test set data

5.7 Data Exportation

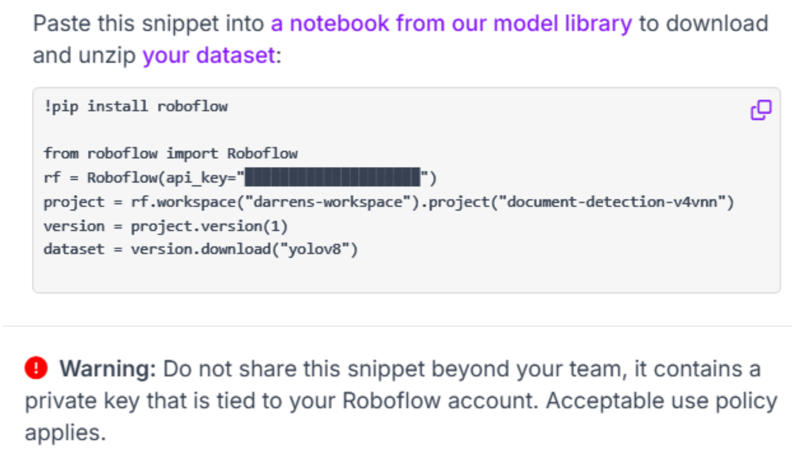
After finishing data annotating, we export our dataset from Roboflow. Roboflow are user friendly where it can export in a yolo format which include these files organized where user do not need to upload the zip file data to google drive and mount the data from google drive into Google Colab and unzip for training further.



test	File folder
train	File folder
valid	File folder
data	Yaml Source File
README.dataset	Text Document
README.roboflow	Text Document

Figure 5.7.1 Zip File of Yolo input format

Besides, the second approach is where we can directly copy the code snippet from Roboflow instead of downloading the zip folder and uploading it to google drive. It is much simpler where we only use the API key that stored all our data sets in the Roboflow cloud services. It used the API key that contains the private key of our user Roboflow account to connect to the Roboflow cloud services and extract the dataset for further training.



Paste this snippet into a notebook from our model library to download and unzip your dataset:

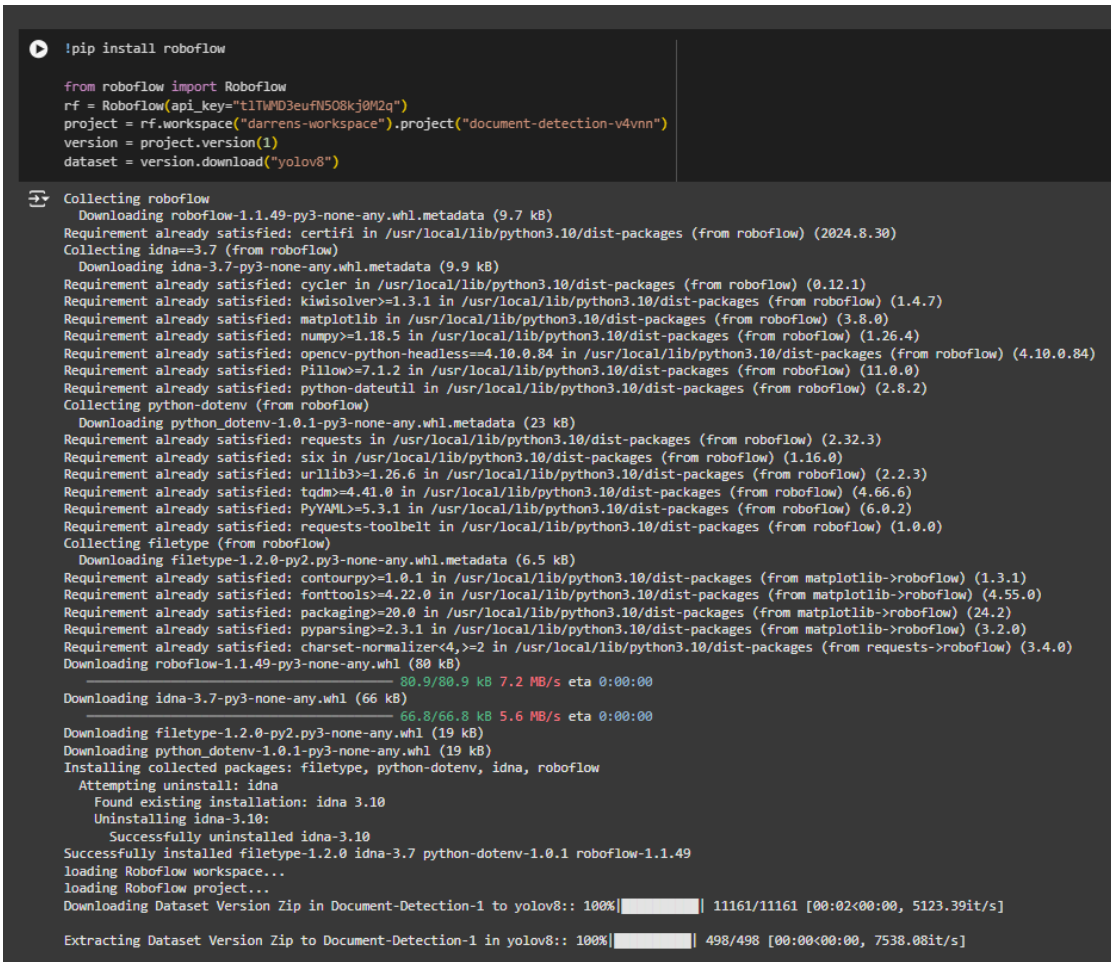
```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("darrens-workspace").project("document-detection-v4vnn")
version = project.version(1)
dataset = version.download("yolov8")
```

Warning: Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

Figure 5.7.2 Code Snippet of dataset

5.8 Model Training

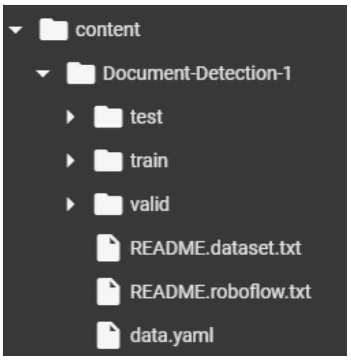
Next, we copy the code snippet previously and paste it into Google Colab and Run to extract the data from Roboflow into Google Colab.



```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="t1TWMD3eufN5O8kj0M2q")
project = rf.workspace("darrens-workspace").project("document-detection-v4vnn")
version = project.version(1)
dataset = version.download("yolov8")
```

```
Collecting roboflow
  Downloading roboflow-1.1.49-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from roboflow) (2024.8.30)
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: cycler in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.7)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.8.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.26.4)
Requirement already satisfied: opencv-python-headless==4.10.0.84 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.10.0.84)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (11.0.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.8.2)
Collecting python-dotenv (from roboflow)
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.32.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.16.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.2.3)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.66.6)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (6.0.2)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.0)
Collecting filetype (from roboflow)
  Downloading filetype-1.2.0-py2.py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (1.3.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (4.55.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (3.2.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->roboflow) (3.4.0)
Downloading roboflow-1.1.49-py3-none-any.whl (80 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.9/80.9 kB 7.2 MB/s eta 0:00:00
Downloading idna-3.7-py3-none-any.whl (66 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.8/66.8 kB 5.6 MB/s eta 0:00:00
Downloading filetype-1.2.0-py2.py3-none-any.whl (19 kB)
Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: filetype, python-dotenv, idna, roboflow
  Attempting uninstall: idna
    Found existing installation: idna 3.10
    Uninstalling idna-3.10:
      Successfully uninstalled idna-3.10
Successfully installed filetype-1.2.0 idna-3.7 python-dotenv-1.0.1 roboflow-1.1.49
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Document-Detection-1 to yolov8:: 100%|██████████| 11161/11161 [00:02<00:00, 5123.39it/s]

Extracting Dataset Version Zip to Document-Detection-1 in yolov8:: 100%|██████████| 498/498 [00:00<00:00, 7538.08it/s]
```

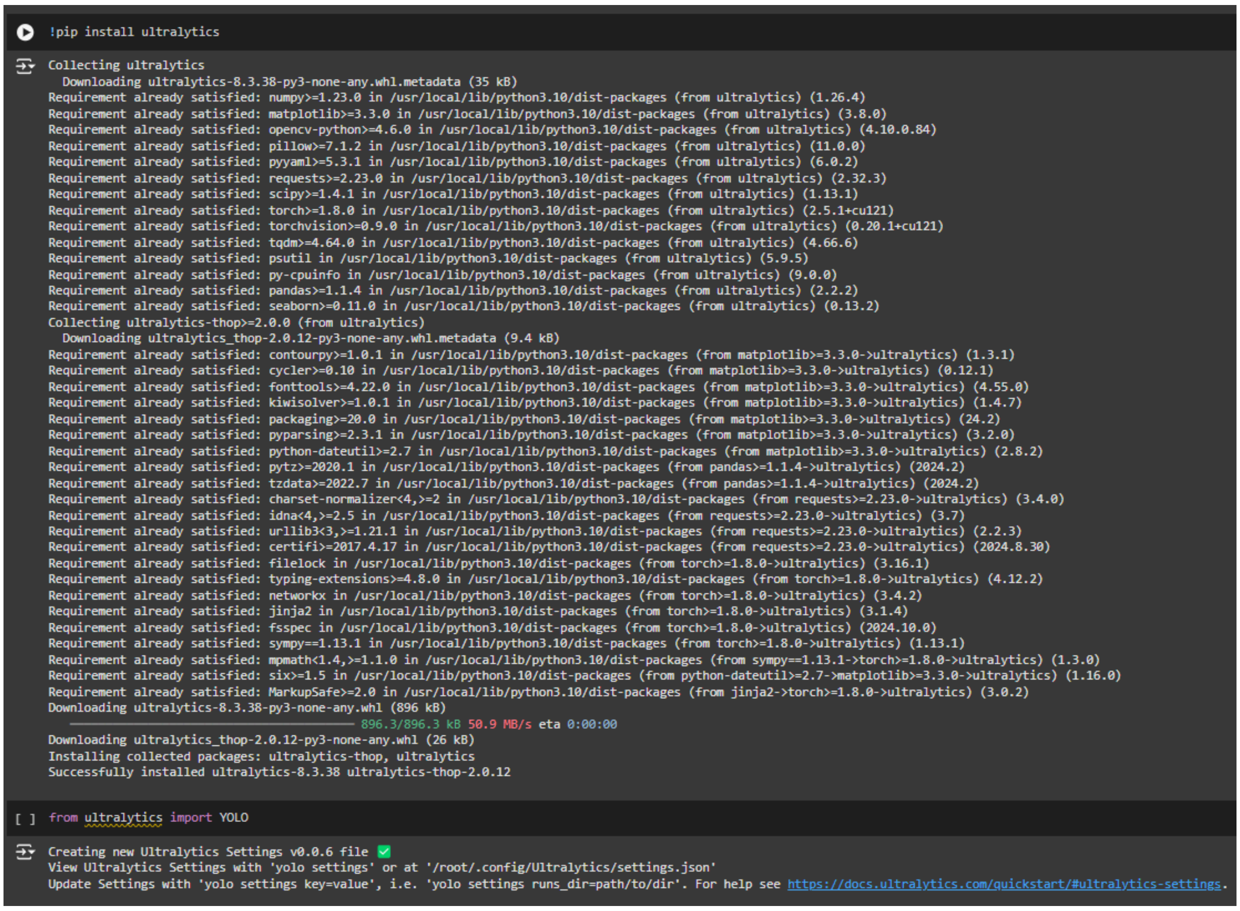
Figure 5.8.1 Data Extraction Process



```
content
  Document-Detection-1
    test
    train
    valid
    README.dataset.txt
    README.roboflow.txt
    data.yaml
```

Figure 5.8.2 Extracted Folder

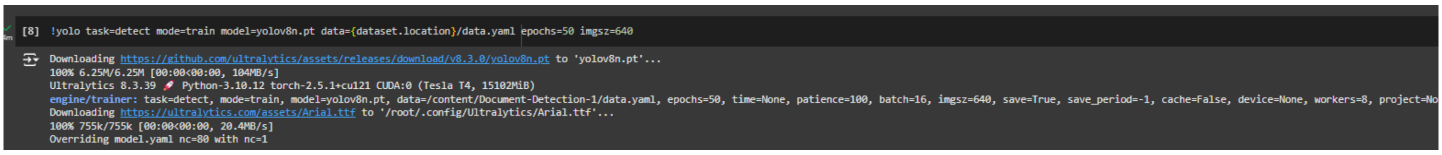
After that, our dataset was successfully extracted and import to Google Colab. Then we install ‘ultralytics’ and ‘yolo’ library to train our model.



```
!pip install ultralytics

Collecting ultralytics
  Downloading ultralytics-8.3.38-py3-none-any.whl.metadata (35 kB)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.26.4)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.8.0)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.10.0.84)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (11.0.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.2)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.13.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.5.1+cu121)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.20.1+cu121)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.6)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.2.2)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.13.2)
Collecting ultralytics-thop>=2.0.0 (from ultralytics)
  Downloading ultralytics_thop-2.0.12-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (4.55.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2024.8.30)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch>=1.8.0->ultralytics) (1.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.3.0->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0->ultralytics) (3.0.2)
Downloading ultralytics-8.3.38-py3-none-any.whl (896 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 896.3/896.3 kB 50.9 MB/s eta 0:00:00
Downloading ultralytics_thop-2.0.12-py3-none-any.whl (26 kB)
Installing collected packages: ultralytics-thop, ultralytics
Successfully installed ultralytics-8.3.38 ultralytics-thop-2.0.12

[ ] from ultralytics import YOLO

Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://docs.ultralytics.com/quickstart/#ultralytics-settings.
```

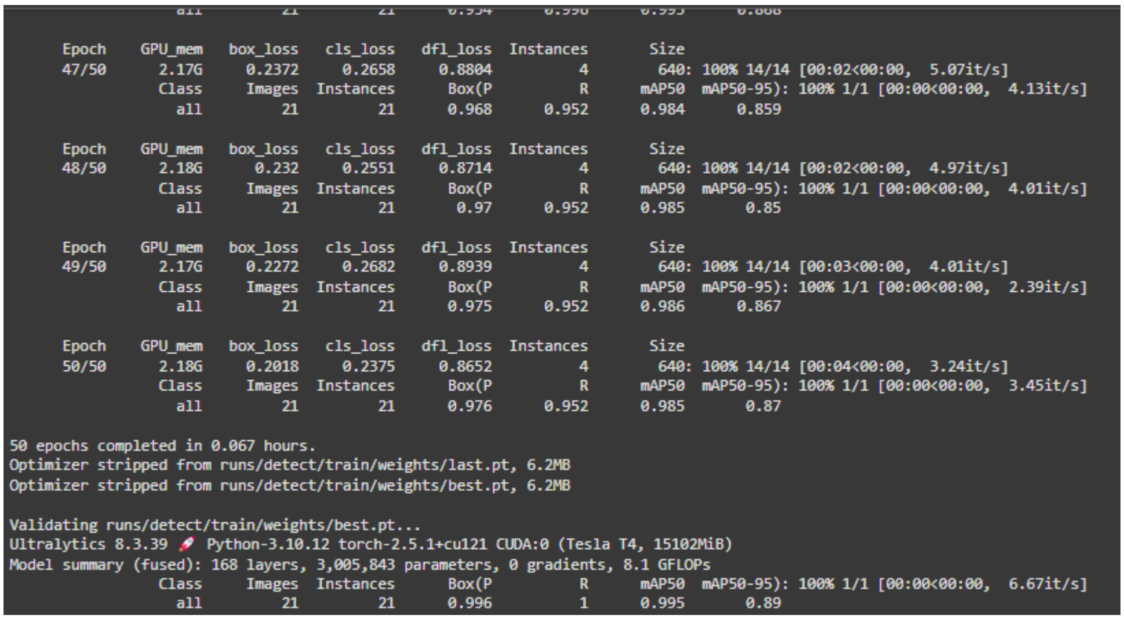
Figure 5.8.3 Import YOLO library

After successful import, we trained our model using a yolov8n pretrained model as it is lightweighted that designed for resource-constrained environments and suitable to deploy in mobile devices to enhance performance. We trained the model with 50 epoch and image size of 640 as the default image size.



```
[8] !yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml epochs=50 imgsz=640

Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8n.pt to 'yolov8n.pt'...
100% 6.25M/6.25M [00:00<00:00, 104MB/s]
Ultralytics 8.3.39 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data=/content/Document-Detection-1/data.yaml, epochs=50, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=No
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100% 755k/755k [00:00<00:00, 20.4MB/s]
Overriding model.yaml nc=80 with nc=1
```

Figure 5.8.4 Code for training model with 50 epoch using YOLOv8n



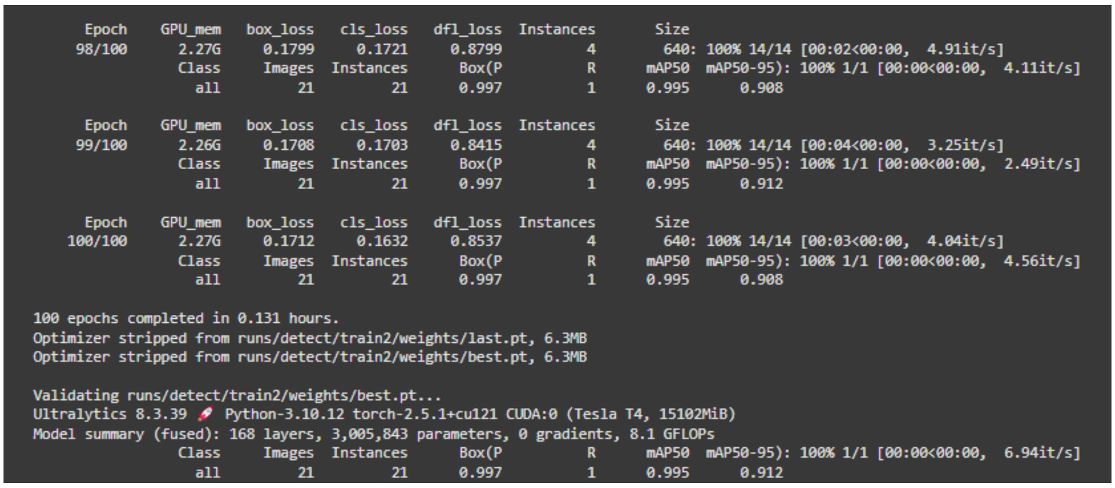
```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      47/50      2.17G     0.2372     0.2658     0.8804          4        640: 100% 14/14 [00:02<00:00,  5.07it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.13it/s]
                   all         21         21      0.968      0.952      0.984      0.859

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      48/50      2.18G      0.232     0.2551     0.8714          4        640: 100% 14/14 [00:02<00:00,  4.97it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.01it/s]
                   all         21         21       0.97      0.952      0.985       0.85

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50      2.17G     0.2272     0.2682     0.8939          4        640: 100% 14/14 [00:03<00:00,  4.01it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.39it/s]
                   all         21         21      0.975      0.952      0.986      0.867

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50      2.18G     0.2018     0.2375     0.8652          4        640: 100% 14/14 [00:04<00:00,  3.24it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  3.45it/s]
                   all         21         21      0.976      0.952      0.985       0.87

50 epochs completed in 0.067 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.2MB

Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.39 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  6.67it/s]
                   all         21         21      0.996          1      0.995       0.89
```

Figure 5.8.5 Result with 50 epoch using YOLOv8n

After that, we found out that the best mAP (50-95) result it reach for 50 Epoch is 0.87.

5.9 Model Evaluation

We try to improve see whether the mAP (50-95) can reach 0.90 above by setting the epoch to 100.



```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     98/100      2.27G     0.1799     0.1721     0.8799          4        640: 100% 14/14 [00:02<00:00,  4.91it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.11it/s]
                   all         21         21      0.997          1      0.995      0.908

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     99/100      2.26G     0.1708     0.1703     0.8415          4        640: 100% 14/14 [00:04<00:00,  3.25it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.49it/s]
                   all         21         21      0.997          1      0.995      0.912

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    100/100      2.27G     0.1712     0.1632     0.8537          4        640: 100% 14/14 [00:03<00:00,  4.04it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.56it/s]
                   all         21         21      0.997          1      0.995      0.908

100 epochs completed in 0.131 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 6.3MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.39 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  6.94it/s]
                   all         21         21      0.997          1      0.995      0.912
```

Figure 5.9.1 Result with 100 epoch using YOLOv8n

From figure 4.8.2 above, we identify that map (50-95) with 100 epoch is > 50 epoch, where it had achieved 0.908. Therefore, we selected this trained model with training on 100 epoch because higher mAP means a better performance in object detection tasks.

Next, we then proceed to visualize the predicted data using our test set data.

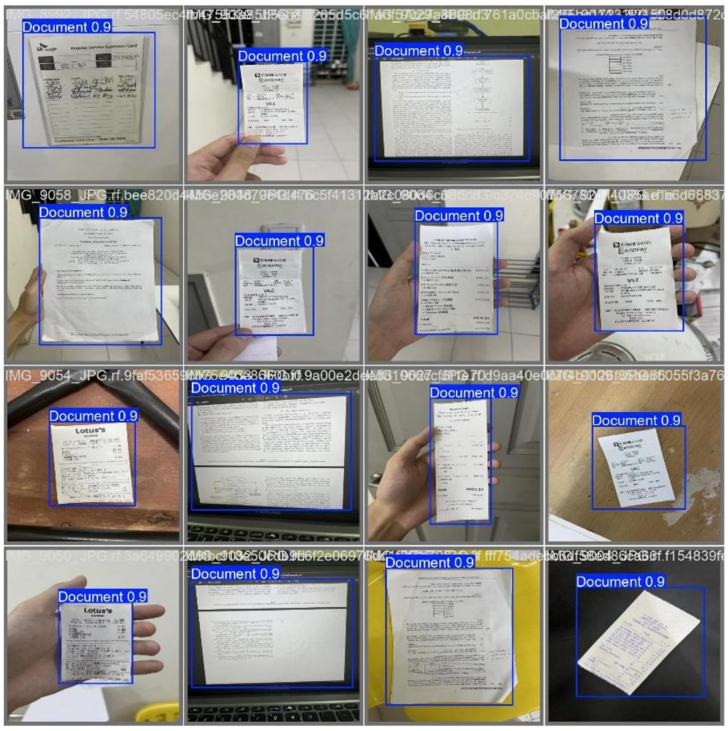


Figure 5.9.2 Document Prediction using Test Set

We can visualize that the model is working well where it can detect all the document with a confidence level of 0.9. Next, we visualize the confusion matrix to identify whether there is any misclassification document.

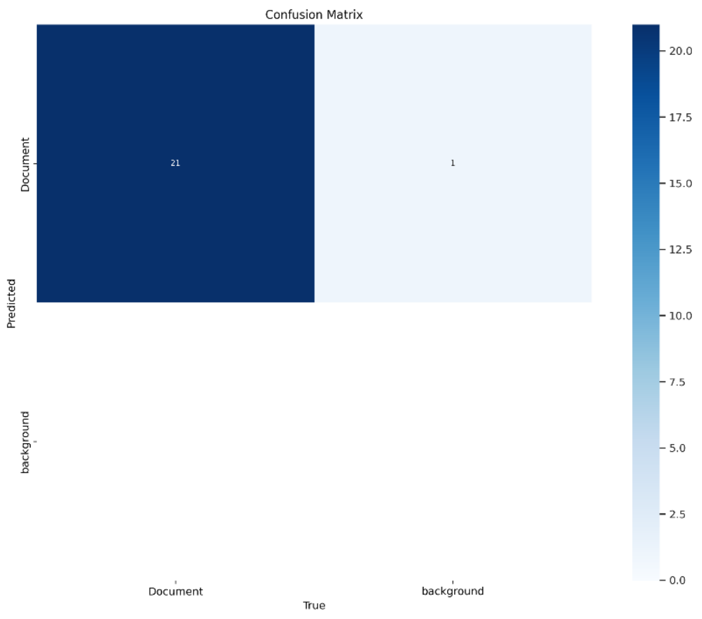


Figure 5.9.3 Confusion Matrix

From Figure 5.9.3 above, we can visualize that there are total 21 documents that are correctly predicted as true positive, and only 1 document is wrongly predicted as false positive.

$$\text{Accuracy} = \frac{21}{21+1} = \frac{21}{22} \approx 0.9545 \text{ or } 95.45\%$$

Figure 5.9.4 Accuracy of document detection

Then, we can calculate the accuracy which is 95.45% using the information from Figure 4.9.3. Furthermore, we proceed to visualize the F1 curve to evaluate our model performance.

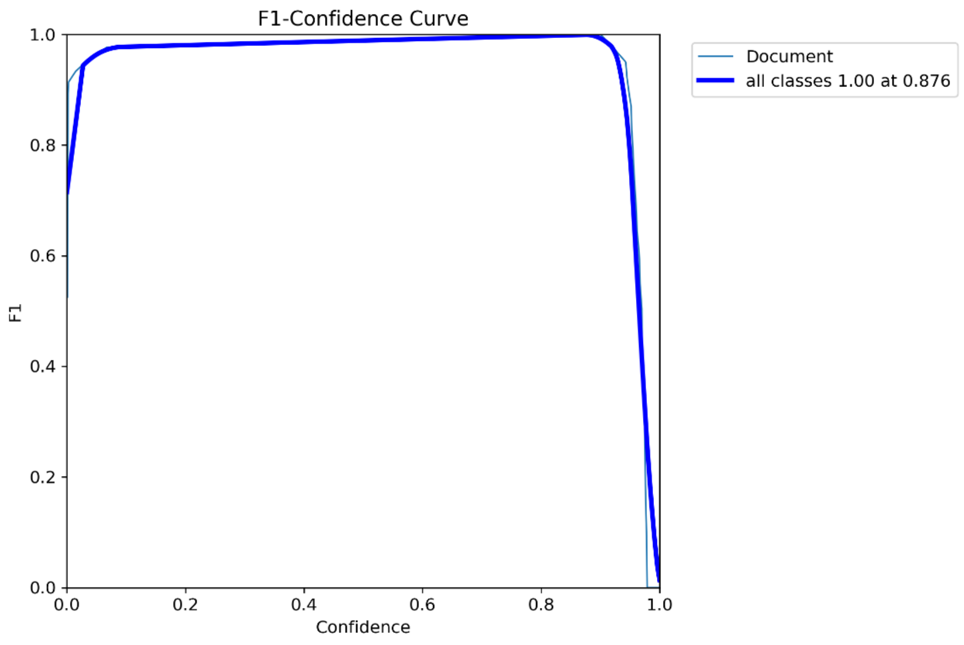
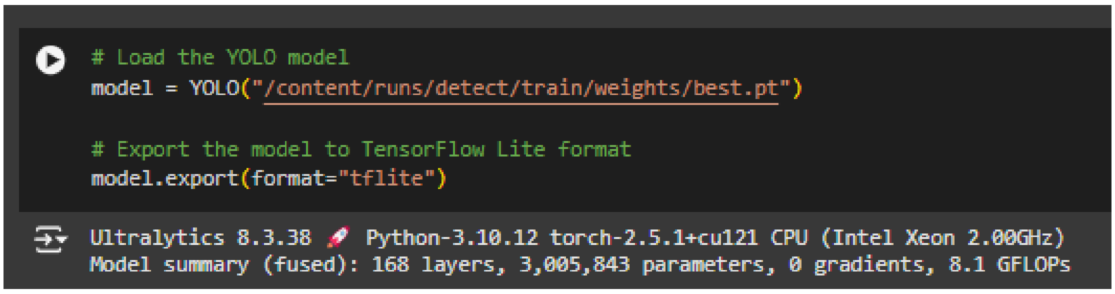


Figure 5.9.5 F1-Confidence Curve

The F1-Confidence curve indicates that the model achieves consistently high performance across a wide range of confidence thresholds, with an F1 score close to 1.0. At a confidence threshold of 0.876, the model reaches an optimal F1 score of 1.00, indicating a perfect balance between precision and recall for all classes.

5.10 Model Exportation

After model evaluation, we proceed to export the model which is the best.pt and convert it into tflite as it is lightweight and able to implement into our mobile devices.

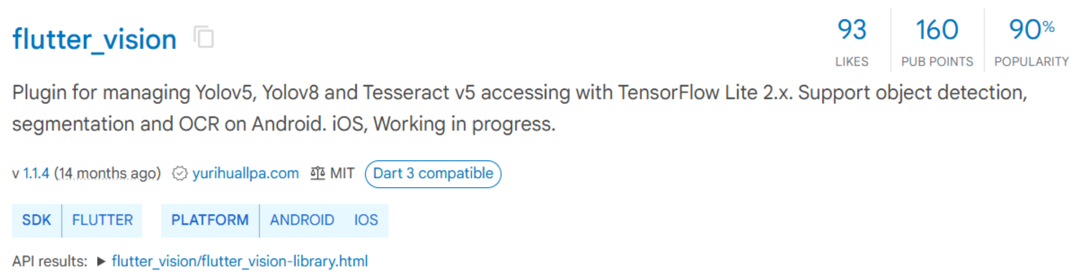


```
# Load the YOLO model
model = YOLO("/content/runs/detect/train/weights/best.pt")

# Export the model to TensorFlow Lite format
model.export(format="tflite")
```

```
Ultralytics 8.3.38 🚀 Python-3.10.12 torch-2.5.1+cu121 CPU (Intel Xeon 2.00GHz)
Model summary (fused): 168 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
```

Figure 5.10.1 Model Exportation Code

After successfully exporting our model and converting it into tflite, we started to implement the model in our codes as a document detection. We will be using a ‘flutter vision’ plugin where it supports object detection model in tflite format for us to code the object detection function.



flutter_vision

93 LIKES | 160 PUB POINTS | 90% POPULARITY

Plugin for managing Yolov5, Yolov8 and Tesseract v5 accessing with TensorFlow Lite 2.x. Support object detection, segmentation and OCR on Android. iOS, Working in progress.

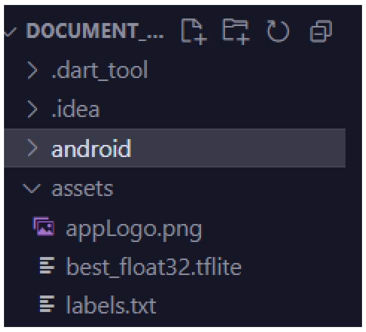
v 1.1.4 (14 months ago) yurihuallpa.com MIT Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS

API results: ▶ flutter_vision/flutter_vision-library.html

Figure 5.10.2 Pub.dev flutter vision plugin

Then, we will need to create a folder called assets which include two major components, the first one is the exported model.tflite file, and the other is labels.txt which includes the class.



```
DOCUMENT_...
> .dart_tool
> .idea
> android
v assets
    appLogo.png
    best_float32.tflite
    labels.txt
```

Figure 5.10.3 Assets Folder

After creating the folder, we will proceed to coding to call the model to perform document detection. Additionally, we will also be creating a bounding box to plot the document in real-time. Furthermore, a TTS feature will also be implemented to assist users in whether the document is detected in the camera preview.

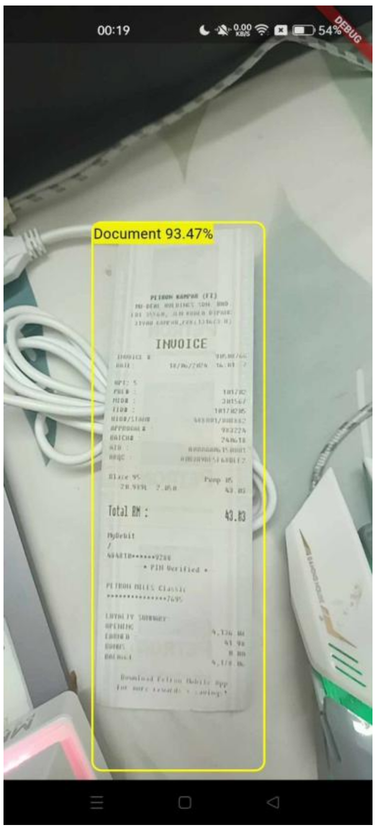
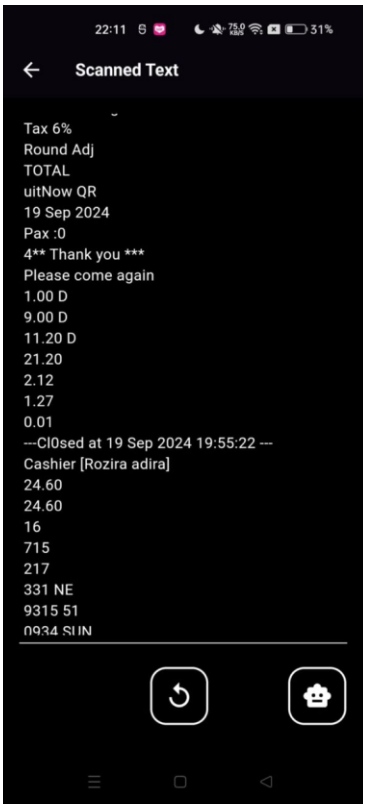


Figure 5.10.4 Implemented Real-Time Document Detection in mobile devices

5.11 Text Alignment Sorting Algorithm

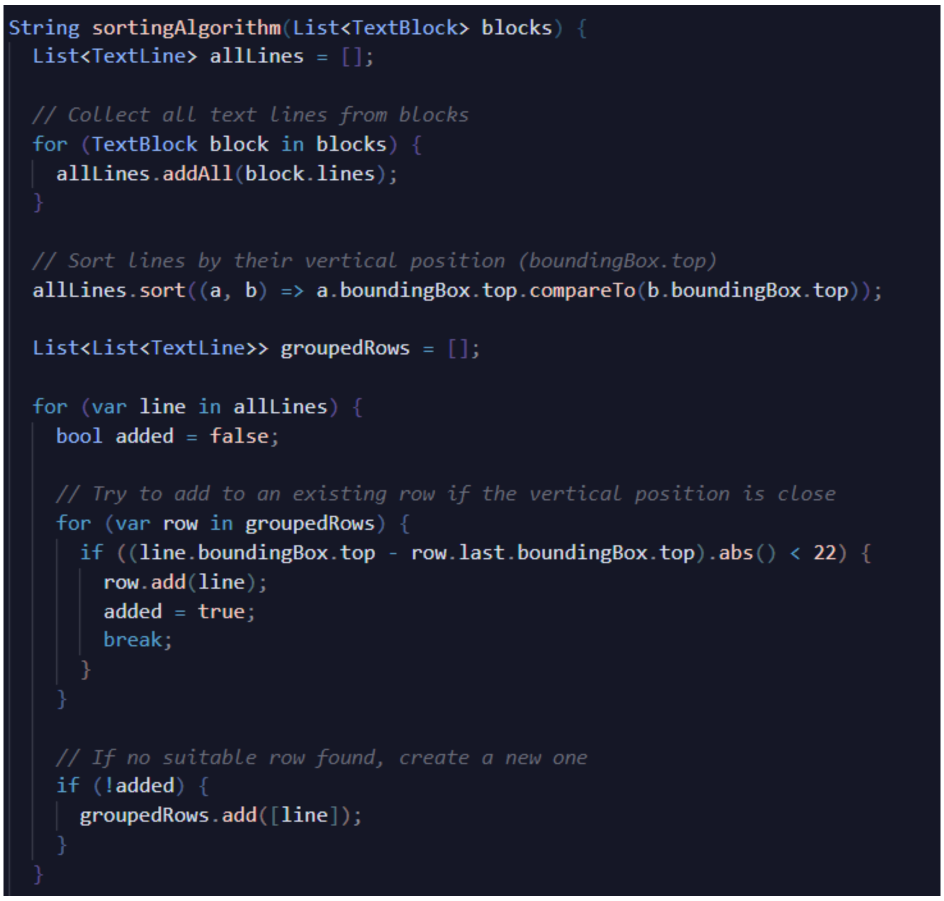
After successful implementing document detection functionalities, some issues had been noticed when scanning document in receipt format. When there exist an empty space or gap between text in document, Google ML Kit default will identify there are no text after and then insert a line break, thus causing an output alignment issue in Figure 5.11.1 below.



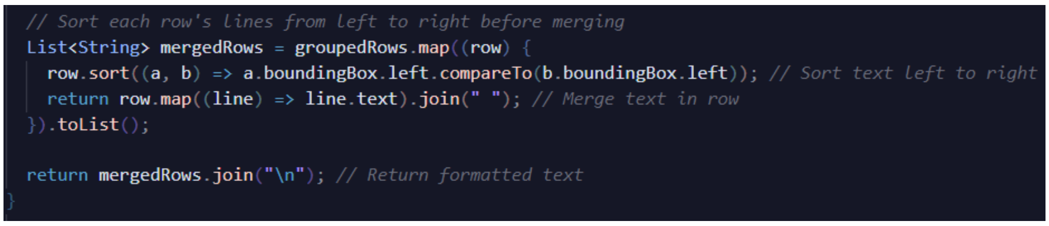
```
Tax 6%
Round Adj
TOTAL
uitNow QR
19 Sep 2024
Pax :0
4** Thank you ***
Please come again
1.00 D
9.00 D
11.20 D
21.20
2.12
1.27
0.01
---Cl0sed at 19 Sep 2024 19:55:22 ---
Cashier [Rozira adira]
24.60
24.60
16
715
217
331 NE
9315 51
0934 SUN
```

Figure 5.11.1 Example of text alignment issue on extracting text from receipt

Noticed that the output text is not logical to interpret by user and struggles when TTS is voicing out the text. Therefore, a text alignment sorting algorithm is implemented mainly to tackle this issue.



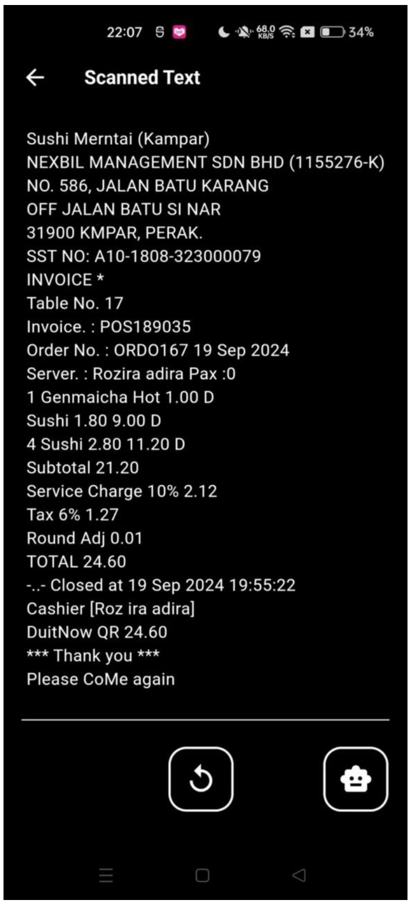
```
String sortingAlgorithm(List<TextBlock> blocks) {
  List<TextLine> allLines = [];

  // Collect all text lines from blocks
  for (TextBlock block in blocks) {
    allLines.addAll(block.lines);
  }

  // Sort lines by their vertical position (boundingBox.top)
  allLines.sort((a, b) => a.boundingBox.top.compareTo(b.boundingBox.top));

  List<List<TextLine>> groupedRows = [];

  for (var line in allLines) {
    bool added = false;

    // Try to add to an existing row if the vertical position is close
    for (var row in groupedRows) {
      if ((line.boundingBox.top - row.last.boundingBox.top).abs() < 22) {
        row.add(line);
        added = true;
        break;
      }
    }

    // If no suitable row found, create a new one
    if (!added) {
      groupedRows.add([line]);
    }
  }
```



```
  // Sort each row's lines from left to right before merging
  List<String> mergedRows = groupedRows.map((row) {
    row.sort((a, b) => a.boundingBox.left.compareTo(b.boundingBox.left)); // Sort text left to right
    return row.map((line) => line.text).join(" "); // Merge text in row
  }).toList();

  return mergedRows.join("\n"); // Return formatted text
}
```

Figure 5.11.2 Example of text alignment sorting algorithm

From Figure 5.11.2 above, a text alignment sorting algorithm is implemented. The text alignment sorting algorithm works by first gathering all the detected lines of text from the scanned document. It then arranges these lines from top to bottom based on their vertical position, ensuring the natural reading order is preserved. Once sorted vertically, the algorithm groups lines that are close to each other in height, assuming they belong to the same row. Within each row, the lines are further sorted from left to right based on their horizontal position. Finally, the text from each row is combined into a single string, and all rows are joined together with line breaks to form a well-structured and readable output that closely matches the original layout of the document. After successful implementation, the output will look like this:



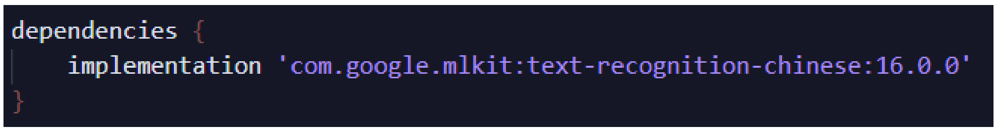
```
Sushi Merntai (Kampar)
NEXBIL MANAGEMENT SDN BHD (1155276-K)
NO. 586, JALAN BATU KARANG
OFF JALAN BATU SI NAR
31900 KMPAR, PERAK.
SST NO: A10-1808-323000079
INVOICE *
Table No. 17
Invoice. : POS189035
Order No. : ORDO167 19 Sep 2024
Server. : Rozira adira Pax :0
1 Genmaicha Hot 1.00 D
Sushi 1.80 9.00 D
4 Sushi 2.80 11.20 D
Subtotal 21.20
Service Charge 10% 2.12
Tax 6% 1.27
Round Adj 0.01
TOTAL 24.60
-..- Closed at 19 Sep 2024 19:55:22
Cashier [Roz ira adira]
DuitNow QR 24.60
*** Thank you ***
Please CoMe again
```

Figure 5.11.3 Precise and accurate text alignment during text extraction from receipt

After implementing the sorting algorithm, our output will look cleaner and able to perform TTS logically.

5.12 Multilingual Text Recognition

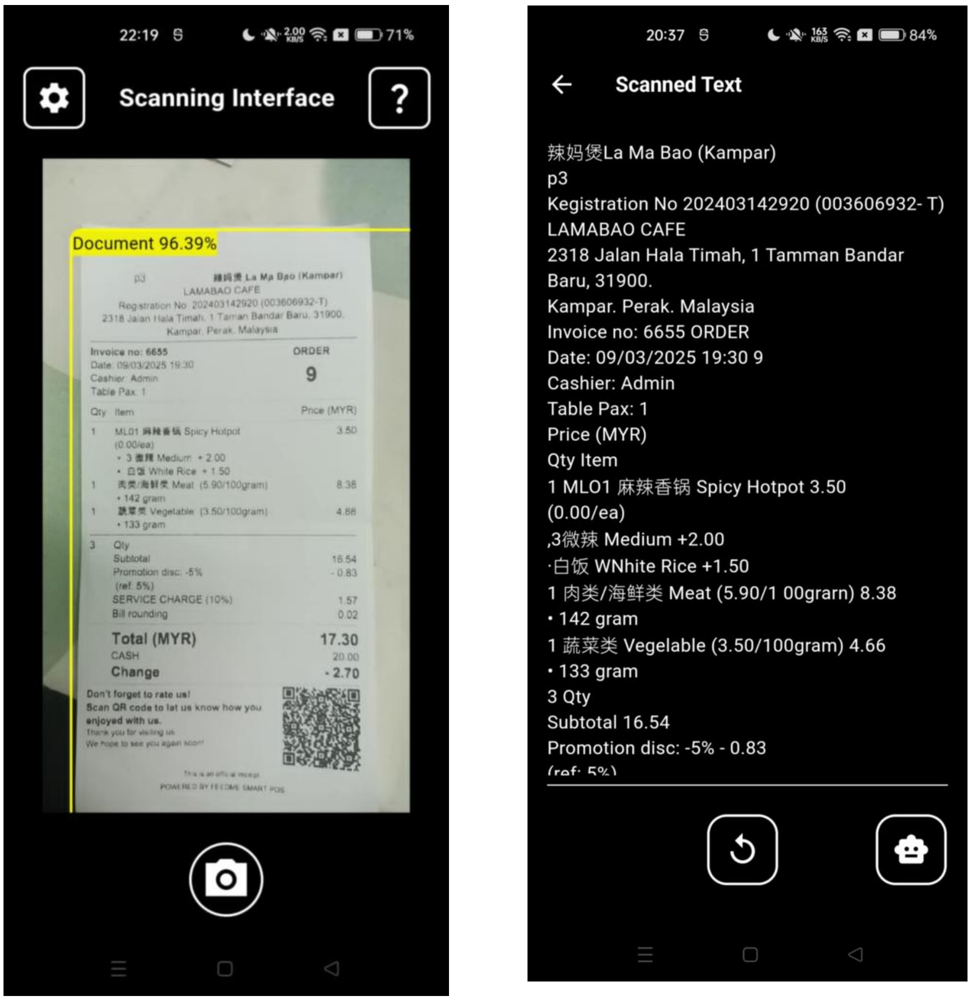
Next, a multilingual text recognition is implemented to recognize Chinese text.



```
dependencies {
    implementation 'com.google.mlkit:text-recognition-chinese:16.0.0'
}
```

Figure 5.12.1 Google ML Kit text recognition-Chinese 16.0.0

The dependencies were added on the build.gradle in the flutter project, then, the packages were implemented for the text extraction.



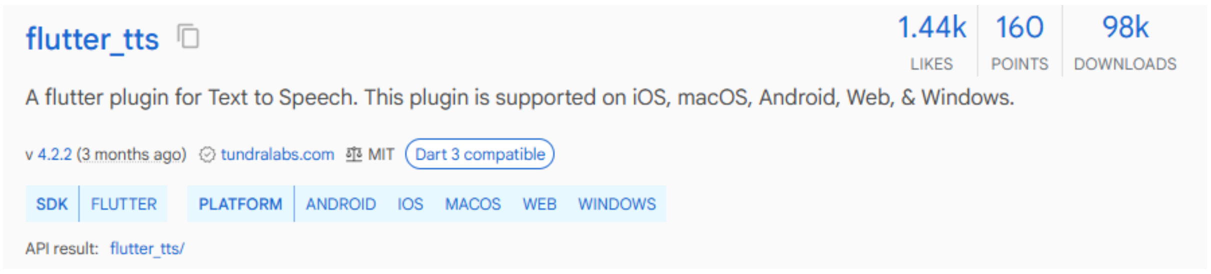
```
辣妈煲La Ma Bao (Kampar)
p3
Kegistration No 202403142920 (003606932- T)
LAMABAO CAFE
2318 Jalan Hala Timah, 1 Tamman Bandar
Baru, 31900.
Kampar. Perak. Malaysia
Invoice no: 6655 ORDER
Date: 09/03/2025 19:30 9
Cashier: Admin
Table Pax: 1
Price (MYR)
Qty Item
1 MLO1 麻辣香锅 Spicy Hotpot 3.50
(0.00/ea)
,3微辣 Medium +2.00
·白饭 WNhite Rice +1.50
1 肉类/海鲜类 Meat (5.90/1 00grarn) 8.38
• 142 gram
1 蔬菜类 Vegelable (3.50/100gram) 4.66
• 133 gram
3 Qty
Subtotal 16.54
Promotion disc: -5% - 0.83
```

Figure 5.12.2 Result of multilingual text extraction

5.13 Chatbot

After that, a chatbot UI page was implemented as the chatbot.dart. It is used for users to perform QnA on the receipt.

5.13.1 STT



flutter_tts

1.44k LIKES | 160 POINTS | 98k DOWNLOADS

A flutter plugin for Text to Speech. This plugin is supported on iOS, macOS, Android, Web, & Windows.

v 4.2.2 (3 months ago) tundralabs.com MIT Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS MACOS WEB WINDOWS

API result: flutter_tts/

Figure 5.13.1.1 Pub.dev google ml kit text recognition plugin

In the chatbot interface, a STT feature was implemented on the bottom right corner. When the user presses and holds, it will start listening to user questions until the user releases the button, then it will convert the speech to text and automatically send as the question to the AI.

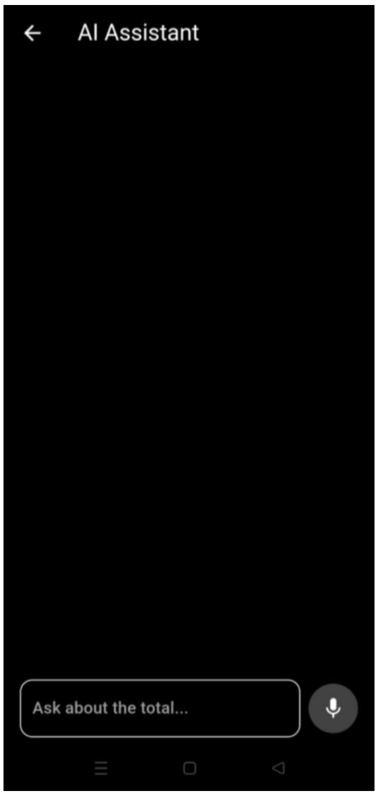
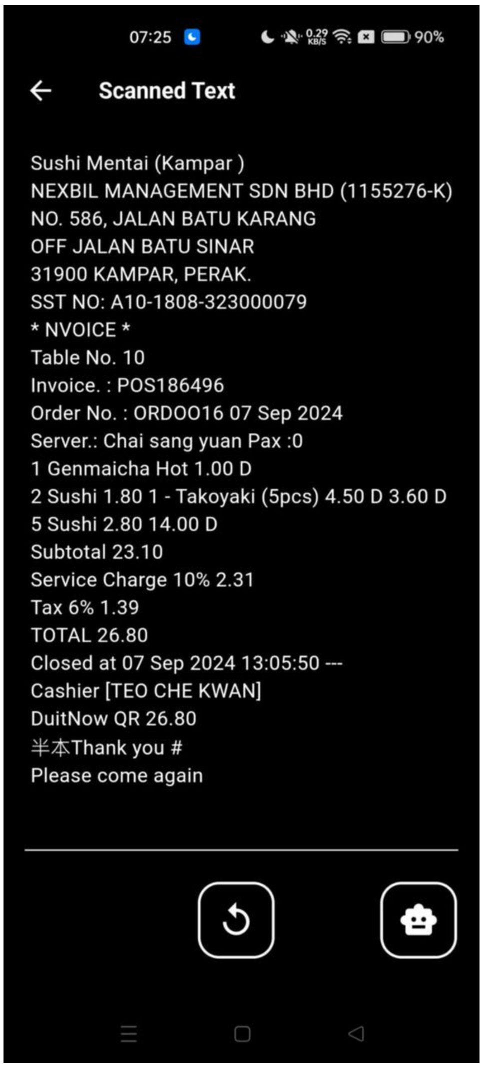
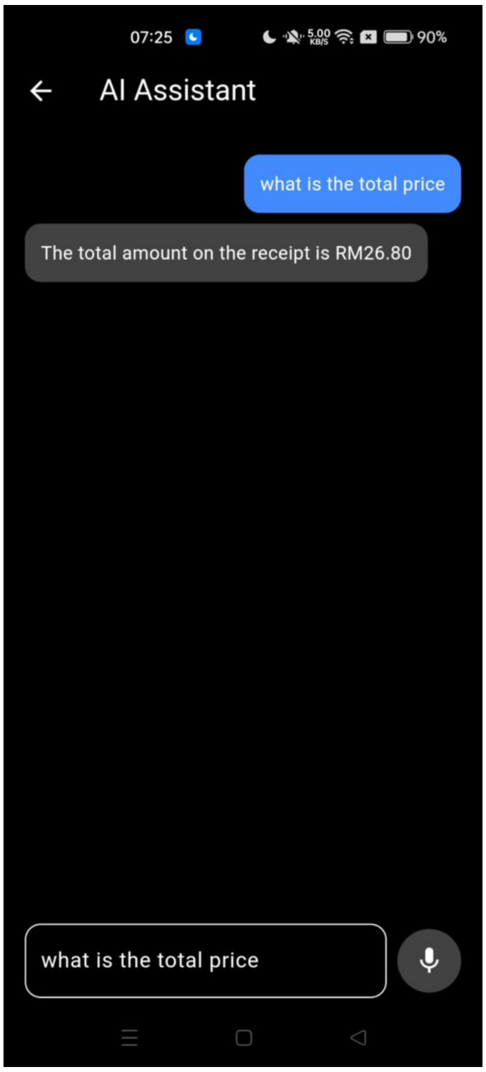


Figure 5.13.1.2 Chatbot interface with STT feature

Then, when user perform question such as “what the total price of the receipt” , the AI will search the keyword from the recognized text previously in the display interface and return to user.



Sushi Mentai (Kampar)
NEXBIL MANAGEMENT SDN BHD (1155276-K)
NO. 586, JALAN BATU KARANG
OFF JALAN BATU SINAR
31900 KAMPAR, PERAK.
SST NO: A10-1808-323000079
* NVOICE *
Table No. 10
Invoice. : POS186496
Order No. : ORDO016 07 Sep 2024
Server.: Chai sang yuan Pax :0
1 Genmaicha Hot 1.00 D
2 Sushi 1.80 1 - Takoyaki (5pcs) 4.50 D 3.60 D
5 Sushi 2.80 14.00 D
Subtotal 23.10
Service Charge 10% 2.31
Tax 6% 1.39
TOTAL 26.80
Closed at 07 Sep 2024 13:05:50 ---
Cashier [TEO CHE KWAN]
DuitNow QR 26.80
半本Thank you #
Please come again



what is the total price

The total amount on the receipt is RM26.80

what is the total price

Figure 5.13.1.3 Example of chatbot returning total amount of receipt back to user

Chapter 6

System Evaluation and Discussion

6.1 Black Box Testing

Black box testing is selected to test and evaluate the functionality of and responsiveness of the mobile application. There are a total of 5 test cases which are successful document detection, successful automatically flashlight, successful text extraction, successful text alignment when document in receipt format, and successful respond to the total amount of the receipt from chatbot.

Table 6.1.1 Test case T01

Test Case ID	T01	**Test Case Name**	Successful Document Detection
Test Case Description	To test that the application can successfully detect a white-based document using YoLov8n model		
Pre-condition	Granted Camera Permission		
Steps	1. Launch the application 2. Place a white based document in front of the device back camera 3. Detect the document in the frame 4. Voice out to user if detected		
Expected Result	Voice out "Document Detected, Hold Still"		
Actual Result	Voice out "Document Detected, Hold Still"		
Status	Pass		

Table 6.1.2 Test case T02

Test Case ID	T02	**Test Case Name**	T02
Test Case Description	To test that the application can automatically turn on the flashlight in the dark environment.		
Pre-condition	Granted Camera Permission		
Steps	1. Launch the application. 2. Initiate the ambient light sensor. **3.** Automatically turn on the flashlight if lux value <20.		
Expected Result	Flashlight turned on.		
Actual Result	Flashlight turned on.		
Status	Pass		

Table 6.1.3 Test case T03

Test Case ID	T03	**Test Case Name**	T03
Test Case Description	To test that the application can successfully extract the text from the camera and return to user using Google ML Kit OCR Recognition		
Pre-condition	Granted Camera Permission and heard document detected voice out (optional)		
Steps	1. Launch the application. **2.** Capture an image that containing text using device camera by clicking on the middle bottom of 'Camera' button. 3. Extract the text from the camera. 4. Display and voice out to user.		
Expected Result	The extracted text is displayed and voice out to user.		
Actual Result	The extracted text is displayed and voice out to user.		
Status	Pass		

Table 6.1.4 Test case T04

Test Case ID	T04	**Test Case Name**	T04
precisely	To verify that the application can successfully display text alignment precisely and accurately when the document is in receipt format.		
Pre-condition	Captured a receipt document using the device camera.		
Steps	1. Capture an image that containing text using device camera by clicking on the middle bottom of 'Camera' button. 2. Extract the text from the camera. 3. Use a sorting algorithm to sort the receipt format. 4. Display and voice out to user.		
Expected Result	The extracted text from receipt is displayed precisely and accurately to the user, and STT voice out logically and fluently.		
Actual Result	The extracted text from receipt is displayed precisely and accurately to the user voice out logically and fluently.		
Status	Pass		

Table 6.1.5 Test case T05

Test Case ID	T05	**Test Case Name**	T05
precisely	To verify that the application chatbot can respond to the total amount of the receipt.		
Pre-condition	Text is extracted and displayed successfully to user.		
Steps	1. Clicking on the right corner of 'AI' button in display screen. 2. Initiate the chatbot. 3. User perform questions using STT. 4. Search for the keyword "total" from the receipt. 5. Return the total amount of the receipt and voice out to the user.		
Expected Result	The total amount is display and voice out to user.		
Actual Result	The total amount is display and voice out to user.		
Status	Pass		

6.2 Objective Evaluation

1ST Objective: To develop a free to use real-time document reader assistance on mobile application

Evaluation: The development of the mobile application was centered on creating a free-to-use, seamless, and intuitive experience for individuals with visual impairments. By incorporating technologies such as OCR Text Recognition, Document Detection, ambient light sensor, STT and TTS feature, text alignment sorting algorithm and chatbot, the application aims to provide comprehensive support and accessibility by ensuring that individuals with visual impairments can navigate and interact with the world around them with ease. It also could enhance daily tasks of the visually impaired individuals such as reading and accessing documents independently.

Achievement: Achieved

2ND Objective: To reduce the time when processing text recognition

Evaluation: This development of project has implemented a Real-Time OCR Text Recognition using Google ML Kit's Text Recognition. The integration of this tool allowed for faster document processing by leveraging its high-performance machine learning models optimized for mobile devices. The use of Google ML Kit also contributed to maintaining high accuracy while minimizing the computational resources required for processing, ensuring that the application could deliver efficient performance even on resource-constrained devices. Furthermore, users can capture the image and process the recognized text without manually selecting a captured image from gallery and uploading to other text recognition mobile applications.

Achievement: Achieved

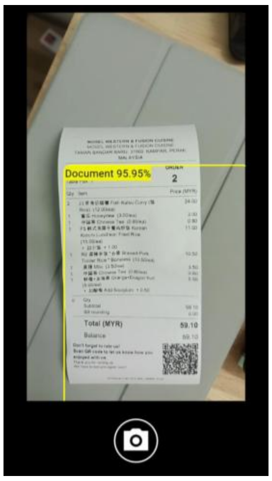
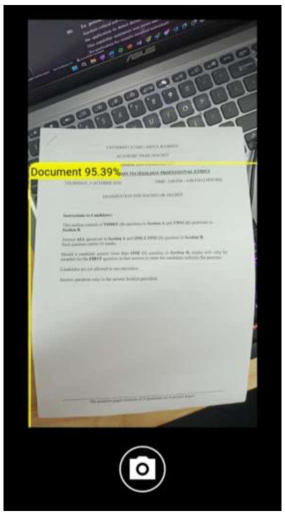
3RD Objective: To provide a real-time document detection for visually impaired

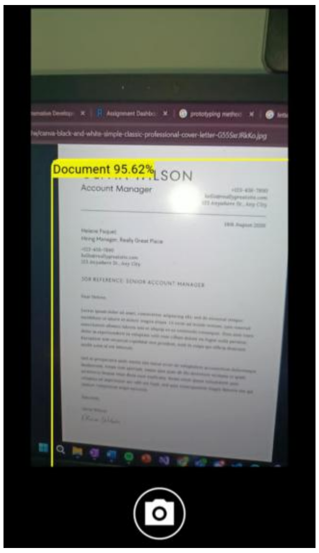
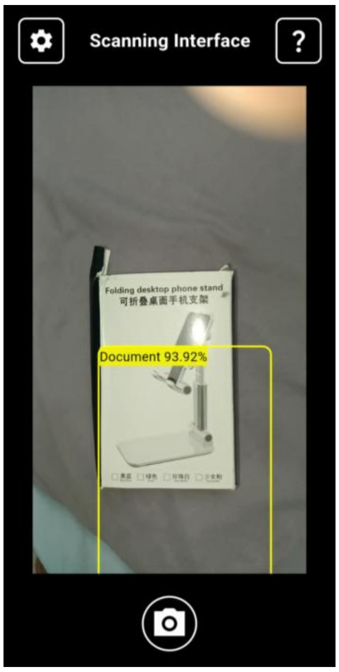
Evaluation: The project successfully implemented a YOLOv8n model, a real-time document detection model. This model was specifically designed to detect and identify documents in a white-based background. By leveraging the power of YOLOv8n, the system was able to perform real-time, accurate document detection with high efficiency. This implementation ensured that the document detection process was both fast and reliable, and formed the foundation for further text extraction and processing within the application.

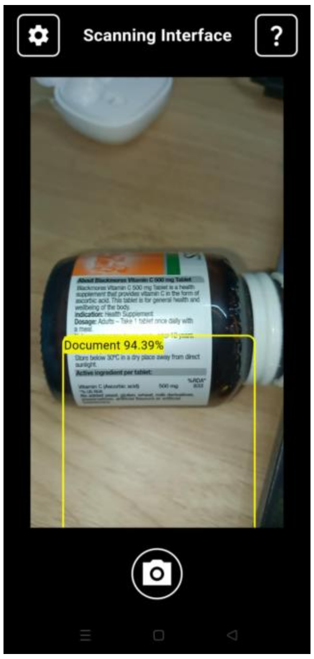
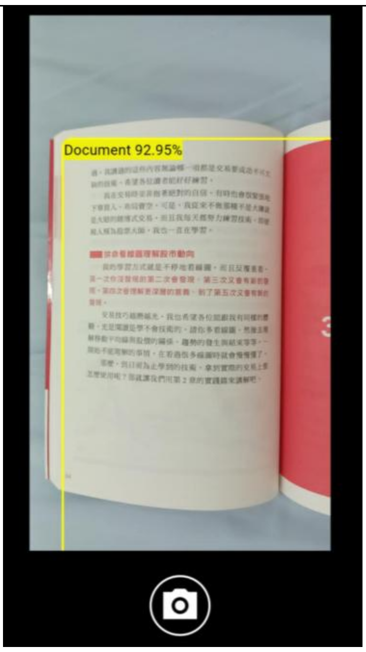
Achievement: Achieved

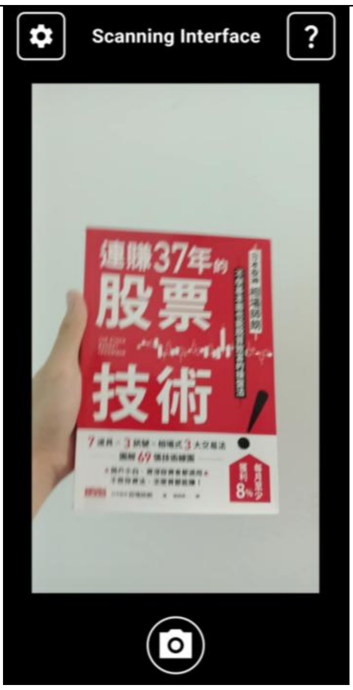
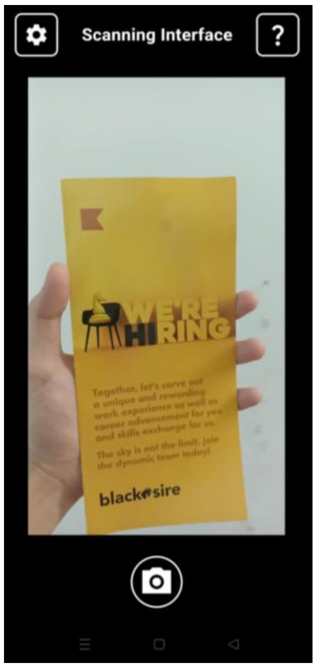
6.3 Error Analysis on Document Detection

Table 6.3.1 Error analysis on document detection

Document Type	**Output**	**Accuracy**	**Detection Outcome**
Receipt		**95.95%**	**Successful**
Examination Paper		**95.39%**	**Successful**

Digital Letter		**95.62%**	**Successful**
Phone holder packaging	 Scanning Interface Folding desktop phone stand 可折叠桌面手机支架 Document 93.92%	**93.92%**	**Successful**

Probiotics		**94.39%**	**Successful**
Book content		**92.95%**	**Successful**

Book cover with high concentration of red background		**0%**	**Unable to detect**
Brochure in yellow background		**0%**	**Unable to detect**

From the error analysis above, we can visualize that it can detect most types of documents in white background and voice out to user that detected, but unable to detect document other than white-based background such as document in red or yellow color.

Chapter 7
Conclusion and Recommendations

7.1 Conclusion

In conclusion, this project allows visually impaired users to use this mobile application independently to read and access various types of documents independently. The progress had completed the development on user interface on scanning screen, tutorial screen, display screen and chatbot screen using Flutter in VS Code. A tutorial page will guide and briefly explain how this application works Additionally; Google ML Kit Text Recognition are implemented to recognize text in real-time on scanning screen. Flutter TTS had also implemented it to allow users to receive real-time audio feedback on the recognized text. Moreover, the recognized text will be displayed on the display screen and voice out to user. Furthermore, datasets which is on white based background are collected to develop a document detection function using the pre-trained model, Yolov8n. these datasets are annotated using Roboflow and trained in Google Colab. The model had been implemented in the scanning screen to detect documents and voice out to user if detected. Then, a real-time bounding box is drawn to assist user by highlighting the detected document. Also, a sorting algorithm is implemented to solve the alignment issue on document in receipt format. Moreover, multilingual text recognition is implemented to recognize text in Chinese format. Furthermore, a STT feature is implemented to assist user on performing questions to the chatbot, without performing typing in keyboard. Lastly, an AI chatbot functionality has been implemented to assist users on searching the keyword from the receipt document. With it, this application delivers accessible and efficient document reading experience, enabling visually impaired users to interact with various documents independently and efficiently.

7.1 Recommendation

There are a few improvements for further development and enhancement of the Real-Time Document Reader Assistance App for the Visually Impaired. This could enhance the document detection model(Yolov8n) to support a wider range of background colors to enhance accuracy and usability in more diverse real-world scenarios.

Additionally, incorporating intelligent AI models such as GPT into the application could enhance user interaction by enabling more natural and context-aware responses to improve the overall user experience.

Moreover, we suggest extending the application's compatibility to additional platforms to reach a wider user base. This may include creating versions for other mobile operating systems like iOS and considering web-based or desktop applications to support users across various devices and environments.

By focusing on these recommendations and consistently improving the application through user feedback and emerging technologies, the Real-Time Document Reader Assistance App can continue to evolve in accessibility and effectiveness for visually impaired users.

REFERENCES

[1] “Vision impairment and blindness,” *www.who.int*, Aug. 10, 2023. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment#:~:text=Prevalence-. [Accessed: Aug. 20, 2024]

[2] N. Griffin-Shirley *et al.*, “A Survey on the Use of Mobile Applications for People who Are Visually Impaired,” *Journal of Visual Impairment & Blindness*, vol. 111, no. 4, pp. 307–323, Jul. 2017, doi: https://doi.org/10.1177/0145482x1711100402. Available: https://files.eric.ed.gov/fulltext/EJ1149519.pdf

[3] Microsoft, “Seeing AI,” *Microsoft Garage*. Available: https://www.microsoft.com/en-us/garage/wall-of-fame/seeing-ai/

[4] Be My Eyes, “Be My Eyes - Bringing sight to blind and low-vision people,” *Bemyeyes.com*, 2019. Available: https://www.bemyeyes.com/

[5] “Envision - enabling vision for visually impaired,” *www.letsenvision.com*. Available: https://www.letsenvision.com/

[6] “Supersense from Mediate: A First Look at a New Object and Text Recognition Mobile App,” *The American Foundation for the Blind*, 2021. Available: https://afb.org/aw/22/4/17489. [Accessed: Aug. 24, 2024]

[7] “FAQ,” *www.supersense.app*. Available: https://www.supersense.app/faq

[8] “Accessible technology: Google Lookout review - Thomas Pocklington Trust,” *Thomas Pocklington Trust*, Mar. 12, 2021. Available: https://www.pocklington.org.uk/health/technology/tech-news-and-views/google-lookout-review/. [Accessed: Aug. 24, 2024]

[9] S. Lewis, “What is the Prototyping Model?,” *SearchCIO*, Jun. 2023. Available: https://www.techtarget.com/searchcio/definition/Prototyping-Model

[10] M. Martin, “Prototyping Model in Software Engineering: Methodology, Process, Approach,” *Guru99.com*, Oct. 24, 2019. Available: https://www.guru99.com/software-engineering-prototyping-model.html

[11] Microsoft, “Visual Studio Code,” *Visualstudio.com,* 2024. Available: https://code.visualstudio.com/

[12] “Project Jupyter,” *Jupyter.org*, 2019. Available: https://jupyter.org/

[13] “TensorFlow Lite for Android,” *TensorFlow*. Available: https://www.tensorflow.org/lite/android

[14] Y. He, “Research on Text Detection and Recognition Based on OCR Recognition Technology,” 2020, doi: https://doi.org/10.1109/ICISCAE51034.2020.9236870

[15] V. Bagal and K. Gaykar, “Image based Text Translation using Firebase ML Kit,” *Academia.edu*, Feb. 17, 2023. Available: https://www.academia.edu/download/98777693/20_1_399_404.pdf. [Accessed: May 08, 2025]

[16] J. Bento, T. Paixão, and A. B. Alvarez, “Performance Evaluation of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 for Stamp Detection in Scanned Documents,” *Applied Sciences,* vol. 15, no. 6, p. 3154, Mar. 2025, doi: https://doi.org/10.3390/app15063154

APPENDIX

Code Sample
Import Library

```
import 'dart:io';
import 'package:document_readerv2/tutorial.dart';
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:flutter_tts/flutter_tts.dart';
import 'package:google_mlkit_text_recognition/google_mlkit_text_recognition.dart';
import 'package:permission_handler/permission_handler.dart';
import 'package:document_readerv2/display.dart';
import 'package:ambient_light/ambient_light.dart';
import 'package:flutter_vision/flutter_vision.dart';
```

Camera Initialization

```
Future<void> initializeCamera() async {
  cameras = await availableCameras();
  if (cameras.isNotEmpty) {
    _cameraController = CameraController(
      cameras[0], // Use the first available camera
      ResolutionPreset.veryHigh,
    );
    await _cameraController!.initialize();
    await _cameraController!.startImageStream((image) {
      if (isDetecting) {
        cameraImage = image;
        _detectDocument(image);
      }
    });
    await _cameraController!
        .setFlashMode(FlashMode.torch); // Initially turn on the flashlight
    setState(() {});
  }
```

Ambient Light Sensor

```
 void _monitorAmbientLight() {
   AmbientLight().ambientLightStream.listen((lightLevel) {
     setState(() {
       _ambientLightLevel = lightLevel;
       if (_ambientLightLevel != null && _ambientLightLevel! < 20) {
         _cameraController
             ?.setFlashMode(FlashMode.torch); // Turn on the flashlight
       } else {
         _cameraController
             ?.setFlashMode(FlashMode.off); // Turn off the flashlight
       }
     });
   });
 }
```

Google ML Kit Text Recognition

```
Future<void> _scanImage() async {
    if (_cameraController == null) return;

    final navigator = Navigator.of(context);
    try {
      // Capture the image
      final pictureFile = await _cameraController!.takePicture();
      final file = File(pictureFile.path);

      // Process the image for text recognition
      final inputImage = InputImage.fromFile(file);
      final recognizedText = await _textRecognizer.processImage(inputImage);

      // Navigate to displayscreen to display the recognized text
      await navigator.push(
        MaterialPageRoute(
          builder: (context) => ResultScreen(iniT: recognizedText.text),
        ),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Error occurred while scanning')),
      );
    }
```

Document Detection

```
Future<void> _initializeVision() async {
  vision = FlutterVision();
  await vision.loadYoloModel(
    labels: 'assets/labels.txt',
    modelPath: 'assets/best_float32.tflite',
    modelVersion: "yolov8",
    numThreads: 1,
    useGpu: true,
  );
  setState(() {
    isModelLoaded = true;
    isDetecting = true;
  });
}

Future<void> _detectDocument(CameraImage image) async {
  final result = await vision.yoloOnFrame(
    bytesList: image.planes.map((plane) => plane.bytes).toList(),
    imageHeight: image.height,
    imageWidth: image.width,
    iouThreshold: 0.4,
    confThreshold: 0.4,
    classThreshold: 0.5,
  );
  if (result.isNotEmpty) {
    setState(() {
      yoloResults = result.where((obj) => obj["box"][4] >= 0.9).toList();
    });
    _speak("Document detected. Hold still.");
  }
}

void _speak(String message) {
  DateTime currentTime = DateTime.now();
  if (currentTime.difference(previousSpeechTime) >= repeatDuration) {
    tts.speak(message);
    previousSpeechTime = currentTime;
  }
}
```

Real-time Bounding Box

```
// Function to display boxes around recognized objects
List<Widget> displayBoxesAroundRecognizedObjects(Size screen) {
  if (yoloResults.isEmpty) return [];
  double factorX = screen.width / (cameraImage?.height ?? 1);
  double factorY = screen.height / (cameraImage?.width ?? 1);

  Color colorPick = const Color.fromARGB(255, 240, 240, 13);

  return yoloResults.map((result) {
    double objectX = result["box"][0] * factorX;
    double objectY = result["box"][1] * factorY;
    double objectWidth = (result["box"][2] - result["box"][0]) * factorX;
    double objectHeight = (result["box"][3] - result["box"][1]) * factorY;

    return Positioned(
      left: objectX,
      top: objectY,
      width: objectWidth,
      height: objectHeight,
      child: Container(
        decoration: BoxDecoration(
          borderRadius: const BorderRadius.all(Radius.circular(10.0)),
          border: Border.all(color: Colors.yellowAccent, width: 2.0),
        ),
        child: Text(
          "${result['tag']} ${((result['box'][4]) * 100).toStringAsFixed(2)}%",
          style: TextStyle(
            background: Paint()..color = colorPick,
            color: const Color.fromARGB(255, 9, 9, 9),
            fontSize: 14.0,
          ),
        ),
      ),
    );
  }).toList();
}
```

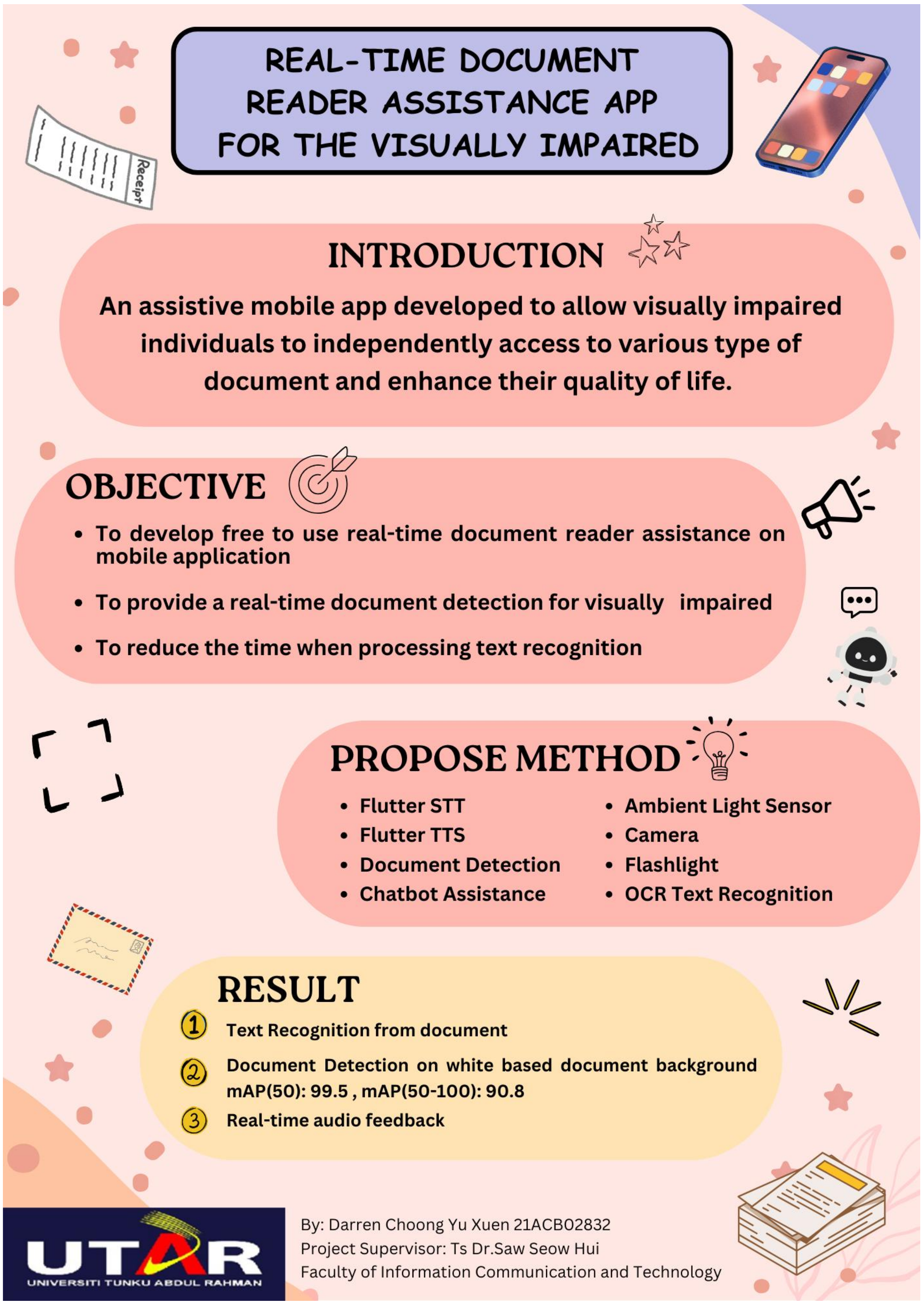
Text Alignment Sorting Algorithm

```
String sortingAlgorithm(List<TextBlock> blocks) {
  List<TextLine> allLines = [];

  // Collect all text lines from blocks
  for (TextBlock block in blocks) {
    allLines.addAll(block.lines);
  }

  // Sort lines by their vertical position (boundingBox.top)
  allLines.sort((a, b) => a.boundingBox.top.compareTo(b.boundingBox.top));

  List<List<TextLine>> groupedRows = [];

  for (var line in allLines) {
    bool added = false;

    // Try to add to an existing row if the vertical position is close
    for (var row in groupedRows) {
      if ((line.boundingBox.top - row.last.boundingBox.top).abs() < 22) {
        row.add(line);
        added = true;
        break;
      }
    }

    // If no suitable row found, create a new one
    if (!added) {
      groupedRows.add([line]);
    }
  }

  // Sort each row's lines from left to right before merging
  List<String> mergedRows = groupedRows.map((row) {
    row.sort((a, b) => a.boundingBox.left.compareTo(b.boundingBox.left)); // Sort text left to
right
    return row.map((line) => line.text).join(" "); // Merge text in row
  }).toList();

  return mergedRows.join("\n"); // Return formatted text
}
```

Extract total from receipt function

```
 void _extractTotalFromReceipt(String text) {
  final regex = RegExp(r'\bTOTAL\s*(RM)?\s*(\d+\.\d{2})', caseSensitive: false);
  final match = regex.firstMatch(text);

  if (match != null) {
    String amount = match.group(2)!; // Always the numeric value
    _sendMessageWithResponse("The total amount on the receipt is RM$amount");
  } else {
    _sendMessageWithResponse("I couldn't find the total on the receipt.");
  }
}
  void _sendMessageWithResponse(String message) {
    setState(() {
      _messages.add({'sender': 'bot', 'text': message});
    });
  }
```

POSTER



REAL-TIME DOCUMENT READER ASSISTANCE APP FOR THE VISUALLY IMPAIRED

INTRODUCTION

An assistive mobile app developed to allow visually impaired individuals to independently access to various type of document and enhance their quality of life.

OBJECTIVE

- To develop free to use real-time document reader assistance on mobile application
- To provide a real-time document detection for visually impaired
- To reduce the time when processing text recognition

PROPOSE METHOD

- Flutter STT
- Flutter TTS
- Document Detection
- Chatbot Assistance
- Ambient Light Sensor
- Camera
- Flashlight
- OCR Text Recognition

RESULT

1. Text Recognition from document
2. Document Detection on white based document background mAP(50): 99.5 , mAP(50-100): 90.8
3. Real-time audio feedback

UTAR UNIVERSITI TUNKU ABDUL RAHMAN

By: Darren Choong Yu Xuen 21ACB02832
Project Supervisor: Ts Dr.Saw Seow Hui
Faculty of Information Communication and Technology