TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1.1 Project Context

In today's rapidly advancing world, image recognition stands as one of the prominent AI abilities currently in use. It entails the computer's capacity to interpret and comprehend visual data. It has been utilized in a variety of applications, such as identifying defects, interpreting medical images, and monitoring security. The techniques employed encompass machine learning and deep learning.

The “CyberFruition: AI-Based Fruit Classification and Recognition Framework” research involves the development of a mobile application that scan and identify fruits. Using image recognition technology, it also gives the user useful information about the fruit, such as its classification, health benefits, and interesting fact. It is essential to recognize that people's perspectives on various subjects, especially fruits, may differ. Lack of knowledge about fruits or any other subject can be due to several factors such as cultural differences, lack of exposure, personal interests, or educational background. This mobile application can foster curiosity and inquiry, and offer educational resources to individuals which would promote overall health and well-being. Educating people about the health advantages of fruits can also help to promote knowledge and appreciation for these natural foods.

In this study, the project will develop a mobile application that can recognize multiple fruits and provide beneficial information to its users. This application will also provide multiple recognition of fruits. This research is conducted to promote fruits in the Philippines and to address the lack of knowledge about fruits.

1.2 Purpose and Description

There are several factors for the lack of knowledge of fruits. First, the cultural differences as people from different cultures may have varying knowledge about fruits specific to their regions. Second, the lack of exposure. For instance, when someone grows up in an area where certain fruits are not readily available, they may not be familiar with them. Third, the personal interest, as some people may have little interest in or exposure to fruits because of personal preferences and dietary choices. Lastly, educational background, as people who have limited access to education or who have not received formal education may have a limited understanding of fruits and other topics.

To address these factors, the proposed project will provide users essential information from the fruit such as its classification, health benefit, other uses, and fun facts of the fruits. It will also offer users multiple-recognition where multiple fruits can be identified at once.

1.3 Objectives

1.3.1 General Objective

This study aims to develop a mobile-based system for the multiple recognition of fruits that will identify and provide beneficial information to the users. By harnessing AI capabilities, the objective of this system is to assist consumers by identifying the fruit with their health benefits and different uses, thereby removing the need to search online for information.

1.3.2 Specific Objectives

1. To scan and identify the fruits being detected.
2. To perform solo or multiple recognition of fruits by pair.
3. To create an interface of fruit classifications for simple navigation and browsing.
4. To develop our own custom datasets of various fruits that will be utilized to train and assist in the creation of an image recognition model.
5. To integrate image recognition feature to capture fruit using the device’s built-in camera.

1.4 Scope and Limitations

The project aims to create an offline mobile application for Android users to give them beneficial knowledge based on the fruits that they have on-hand. The open-source software development kit will be used, with Android Studio as a deployment environment. Ten fruits and random objects will be trained to analyze content and contextual data using the teachablemachine with google.com.

The application is compatible with Android mobile phones and it can identify ten fruits only. It scans through capturing using the camera of the phone or uploading an image. The system can only recognize fruits that are trained in TensorFlow by the developer such as: apple, orange, banana, pineapple, mango, papaya, calamansi, durian, jackfruit, lanzones, and unknown fruits or random objects. The information provided will only be its classification, benefits, and other use.

1.5 Significance of the Project

The "CyberFruition: AI-Based Fruit Classification and Recognition Framework" project holds relevance in daily life. The mobile application improves the user experience by including image recognition technology, which enables users to take pictures of fruits and get information based on them. Allowing the user to find out beneficial details about the fruit not only saves time and effort but also encourages appreciation and consumption. This app is a useful tool to individuals who look up on internet for fruits information.

The study will benefit the following:

Consumers - This will make their learning process about the health benefits of fruits faster, saving them time.

Individuals with limited knowledge about fruits - The study will provide them with new information about various fruits.

Technology - The research aims to enhance technological productivity by providing assistance to individuals in society. One significant contribution is the system's capability to detect a fruit's information and its benefits to consumers.

1.6 Definition of Terms

- CyberFruition - refers to the specific framework or system developed for fruit classification and recognition using AI (Artificial Intelligence) techniques. It is designed to automate the process of identifying and categorizing different types of fruits based on their visual features.

- Fruit Classification - involves categorizing fruits into different classes or types based on their characteristics, such as shape, size, color, and texture. Berries, Pits, Core, Citrus Fruits, Melons, Tropical Fruits are the six main fruit categories.

- Fruit Recognition - is the process of identifying and distinguishing different types of fruits based on their visual attributes.

- Teachablemachine - a web tool that enables fast and easy machine learning model creation for your projects without the need for coding. It can teach and train a computer to recognize images, sounds, and pose, then export a model for deployment in websites, applications, and more.

- Android - a mobile operating system for touchscreen gadgets like smartphones and tablets, built upon a customized Linux kernel and other open-source software.

- Java XML - Extensible Markup Language (XML) is a simple text-based language that is often used to communicate between different applications.

- Tensorflow - it is an open-source, free software library for machine learning and artificial intelligence. Though it is used in different range of tasks, deep neural network training and inference are given special attention.

CHAPTER 2

REVIEW OF RELATED LITERATURE

2.1 Introduction

The fruit industry requires further technological tools for fruit pattern recognition. To accomplish this objective, conducting additional research is crucial. This study contains several articles that will aid in the further analysis and broader of the study of fruit classification and recognition.

2.2.1 Studies on Fruits at the Philippines

According to Reyes (2019) and Punzalan (2016) their research revealed a diverse array of fruits grown in the Philippines. Among the major fruit species, Banana, Pineapple, Mango, Papaya, Calamansi, Durian, Jackfruit, and Lanzones were identified as having the highest volume of production. These fruits are known for their distinct characteristics, including varying sizes, flavors, and colors. The availability of these fruits throughout the year allows Filipinos to incorporate them into their daily cuisine, enhancing both the taste and nutritional value of meals.

2.2.2 Studies on Knowing the Importance of Fruits Benefits

Slavin (2012) and Lloyd's (2012) study emphasized the consistent promotion of fruits as healthy choices in dietary guidelines. They provided essential nutrients and dietary fiber linked to lower rates of cardiovascular disease and obesity. Fruits contain phytochemicals with antioxidant, phytoestrogenic, and anti-inflammatory properties. They found out that the classification attempts have improved understanding of their nutritional composition. Epidemiological and clinical studies support the health benefits of consuming fruits.

2.2.3 Studies on Machine Learning of Classification of Fruits

A study conducted by Alkahlout et al. (2021) used the Kaggle website to teach users about identifying different fruits in photos through deep learning. They utilized a pre-trained CNN model called VGG16 and trained it using a dataset for testing. The model achieved a 100% accuracy rate, indicating its effectiveness in accurately predicting and classifying fruits. They plan to further expand their study by evaluating the framework with more fruit classes and exploring the impact of various parameters such as activation functions, pooling functions, and optimization techniques. They also mentioned the potential use of a cloud-based framework alongside their proposed model.

According to the Joseph et al. (2021), they utilized convolutional neural networks within the field of deep learning to develop the model. The model underwent training for 50 epochs on the Fruit 360 dataset, resulting in an impressive accuracy of 94.35%. The substantial influence of deep learning in the present era, along with its capability to handle an increasing number of features, contributes to mitigating the limitations associated with traditional methods.

Patel et al. (2011) conducted a study on Fruit Detection using Improved Multiple Features based Algorithm. This study allows for automatic extraction of fruit regions without user intervention. The method is not domain-specific and can be applied to various images with meaningful fruit regions. The authors suggest exploring additional features like symmetry and investigating the use of different imaging devices in future work. The proposed method for fruit detection involves extracting features like intensity, color, edge, and orientation from the image. These features are combined using weights based on their impact on the image region. The integrated map is then segmented using global thresholding to create a binary map. Finally, fruit regions are extracted based on the binary image.

According to Naranjo-Torres, J et al. (2020), previous studies did not emphasize CNN as a relevant approach for fruit analysis, but their review highlighted its importance and focuses on CNN-based approaches for fruit image processing. They identified three main application areas: fruit classification, fruit quality control, and fruit detection for harvesting. Different CNN architectures are used across these applications, and no single architecture is deemed superior. Pre-trained CNN models can be modified, or new models can be built from scratch. The evaluation results indicate that CNN-based approaches have achieved excellent results, including up to 100% accuracy in some cases. The review also notes significant growth in the use of CNN in robotic harvesting applications.

2.2.4 Studies on Mobile Multiple Recognition Applications

Various studies have explored multiple recognition approaches, yet with different areas of focus. A novel method for automatic disease identification based on digital image analysis have been presented. This method utilizes color transformations, intensity histograms, and a pairwise-based classification system, specifically designed to handle a wide range of diseases under uncontrolled conditions. It allows for the inclusion of new diseases without modifying the pre-trained system, simplifying the process. Extensive testing was conducted using a diverse dataset of leaf images, encompassing 74 diseases, 4 pests, and 4 abiotic disorders across 12 plant species. The dataset included images with non-disease symptoms to enhance diagnostic accuracy. The method was subjected to challenging conditions, revealing insights into the complexities of disease identification when dealing with multiple diseases simultaneously. (Barbedo et al., 2016)

2.2.5 Studies on Multiple Data Sets in Study

A study by Mostafa Kabolizadeh investigates the enhanced accuracy of classifying the area under crops by leveraging multiple datasets. Building on research on improving classification accuracy in agriculture, their study focuses on Shush County, Iran, utilizing multi-temporal Sentinel-1 and Sentinel-2 images. Employing Sequential Forward Selection (SFS), Support Vector Machine (SVM), and Random Forest (RF) algorithms, they identify optimal features for four datasets This demonstrates that the use of information from multiple datasets contributed to improved accuracy in classifying the area under crops, emphasizing the significance of feature integration for enhanced precision in monitoring agricultural landscapes.

Additionally, the thesis explores the effectiveness of Separation Logic in software engineering, drawing insights from the work of David Pym and Jonathan M. Spring. Separation Logic by integrating software engineers' conceptual models with logical frameworks, enhancing scalability in reasoning tasks. The research delves into the intricate interplay between conceptualization and logic within Separation Logic, showcasing its capabilities in facilitating scalable and rigorous reasoning in software development.

And in this thesis delves into the realm of Separation Logic, as elucidated by Pym (1999), Ishtiaq and O'Hearn (2001), and Reynolds (2002). Separation Logic introduces a novel connective, 'and, separately,' within standard logic, addressing the intricate challenges associated with reasoning about the resources required by a computer program during execution. The overarching goal is to dissect the inherent difficulties in resource reasoning and articulate the unique arrangement of properties within Separation Logic that renders it a potent tool for effective program verification. By examining what makes reasoning about computer resources arduous, this thesis aims to provide a comprehensive understanding of Separation

Logic's capabilities and its contribution to overcoming the complexities inherent in program verification problems.

2.3 Summary

From the gathered related studies and literature for fruit detection and classification, it was concluded that different approaches have been studied, with focus on different aspects. A study focused on using multiple features such as intensity, color, edge, and orientation. CNN-based approaches, specifically using the VGG16 model, are utilized for fruit image processing and are applied in fruit classification, fruit quality control, and fruit detection for harvesting. The activation function, pooling function, and optimization techniques are employed within convolutional neural networks (CNNs) to develop the model. Additionally, the use of sonar videos is explored to enhance computer vision applications in low signal-to-noise scenarios and address domain generalization in tasks like multiple-object tracking and counting. For multiple recognition, a classification system has incorporated color transformations, intensity histograms, and pairwise-based classification was also implemented to handle various diseases under uncontrolled conditions without modifying the pre-trained system.

Based on the gathered approaches, fruit classification and recognition encompass various focuses, such as ripeness, time of harvesting, and other related factors. Multiple recognition approaches have also been explored, but not specifically targeted towards fruits. The proponents can use these approaches to make informed decisions on which features to prioritize and implement, aiming to enhance the existing methods.

2.4 Synthesis

Application	Description	Features	Similarity with the proposed application
Fruits Detector	Fruits Detector is an application that can identify fruits and vegetables by using a	The "take photo" feature allows users to capture a picture of the fruit alone; no	The goal is to create an intuitive and simple-to-use offline mobile application.

	live camera or by taking a picture. Its user interface is limited and straightforward. It has just one screen: the live camera and the home screen.	content can be uploaded. The live camera feature allows for real-time access to the user's camera. It detects fruit by locating it and putting a name above it.	The suggested application enables users to capture fruit photos. The user can also upload a picture from their gallery using this feature. And last, one of the project's objectives is the live camera
FruitVegetableSnap	FruitVegetableSnap identifies names of nearly a thousand fruits and vegetables. Relevant information is given to users, including suggestions for fruits and vegetables, collocation taboos, and nutritional value. It does not have an offline functionality and is dependent on the internet.	It lets the user take a picture of a fruit or upload one. The accuracy results percentage is also displayed. The fruit being detected displays multiple results in addition to the accuracy rate. The user is limited to three identification attempts. For unlimited identifications, the user must sign up for a subscription plan. In addition to changing the	Both place emphasis on creating a mobile application that is easy to use and offers details about the fruit that is being detected. It will offer a brief description of the fruits identified in the picture.

		application's language, the user can log in to access identification content.	
PictureThis	98% of the plants in the world can be identified with this application. It offers solutions to inquiries about gardening. Through photos, it can identify trees, flowers, herbs, and plants.	The application offers the following features: Accurate Plant Identifier, Plant Disease, Auto Diagnose & Cure, Plant Care Tips & Reminders, One-on-one Expert Consultation, Toxic Plant Warning, Manage Your Plant Collection, and Plant Recommendation.	Similar strategies are used by PictureThis to offer user-friendly platforms that deliver precise results for the fruits that are being detected. By taking a picture or uploading one, it can also recognize fruits through image recognition.

Table 1. Synthesis

CHAPTER 3

TECHNICAL BACKGROUND

This chapter focuses on the technical aspects involved in creating the system. It encompasses the conceptual framework, software requirements, and hardware requirements utilized throughout the development and implementation phases of the application.

3.1 Conceptual Framework

This section presents the study's conceptual framework and the tasks that must be accomplished by the end of the study.

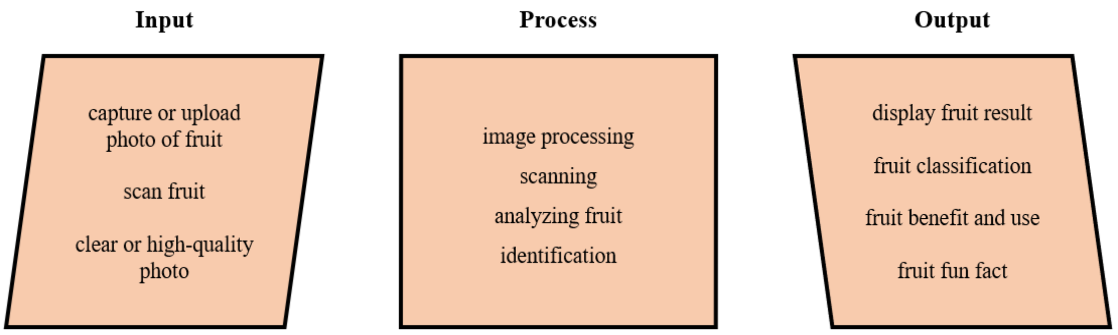


Figure 1. Conceptual Framework of the Application (CyberFruition)

This framework as shown above presents the possible courses of action of designing and developing a mobile AI-based application for fruit recognition and classification. Before starting, the user may either access the phone camera to capture scan the fruits or upload a photo. On the middle portion of the figure is the process, it involves analyzing user input before giving the user the results.

The findings in the related literature supports the idea of creating AI- based fruit classification and recognition through mobile application. Adopting efficient technical methods provides convenience making the project effective.

3.2 Software Requirements

The software requirements used in the development and implementation is listed below.

3.2.a Programming IDE

Android Studio is a powerful and versatile tool for creating Android applications, created by Google and built on the IntelliJ IDEA platform. It includes a code editor, a visual layout editor, debugging tools, and a program emulator for testing programs.

3.2.b Programming Language

Java XML works with XML, a markup language used for data storage and transfer, and is an extension of the Java programming language. The DOM, SAX, and JAXB APIs allow developers to parse, modify, and create XML documents. For online services, data interchange, and configuration files, it is widely utilized.

3.2.c Software Development kit

Android SDK is a collection of resources, frameworks, and tools to create apps for the Android platform. The Android SDK contains all the necessary tools for developers to design, test, and debug Android applications.

3.2.d Visual Recognition Integration

Teachable Machine is a web-based tool that Google developed which enables users to build machine learning models without any prior coding or machine learning knowledge. Through the use of transfer learning, users can adapt pre-trained models to suit their particular task by leveraging them. It is intended to provide machine learning to a larger range of users.

3.3 Hardware Requirements

The hardware requirements for the development phase are listed below.

For development purposes, the hardware recommended requirements for the processor is intel i5 processor. Regarding hardware memory, the recommended requirement is 8 Gigabytes (GB) of RAM. 256GB of ROM is the recommended hardware requirement.

Hardware used by the developer for the development phase:

- Computer/Processor: 2.50 gigahertz (GHz) / Intel Core i5.
- Memory/RAM: 8GB RAM
- Hard Disk: 512gb SSD & 500gb HDD
- Computer Speed: 2.50ghz
- Operating System: Windows 10

Hardware requirements for user (Mobile)

- Android 8 or higher

CHAPTER 4

DESIGN AND METHODOLOGY

4.1 Introduction

This chapter describes the design and methodology used in implementation and development of the application. Agile was the methodology opted by the proponents as indicated in the figure below.

Software Development Life Cycle

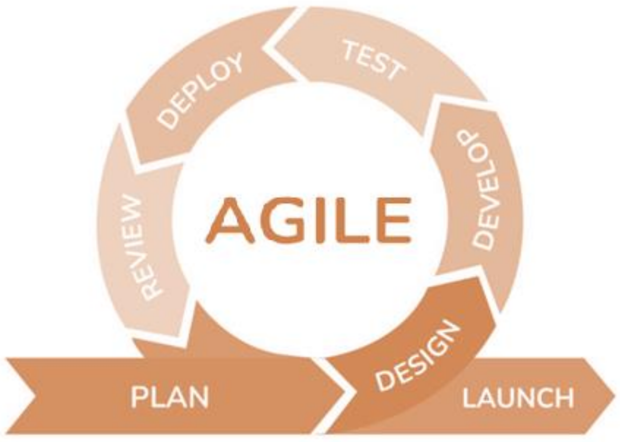


Figure 2. Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a structured approach used in software development projects. It encompasses various stages, including requirements gathering, design, development, testing, deployment, and maintenance). It provides a systematic framework for managing the entire software development process from start to finish.

The developers will follow Agile Methodology with these steps:

First, Planning, by gathering requirements for the mobile application, including the specific fruit datasets to be utilized by the AI. Prioritize features related to dataset accuracy and create a roadmap for development. Second, Designing, building the design of the application's architecture to integrate the AI functionality effectively. Create a user-friendly interface for users to interact with the AI-powered feature. Third, Development, code and implement the mobile application, including the necessary algorithms and models to ensure accuracy in processing and utilizing the datasets for AI fruit detection. Iterate on development with frequent updates. Fourth, Testing, conduct through testing to verify the accuracy of the AI model and their integration with the mobile application. Perform functional testing to ensure accurate dataset processing and validate the AI-driven functionalities. Fifth, Release, once the mobile application and AI functionalities are stable and meet the desired level of accuracy, proceed with deploying the application to users or app stores. Finally, feedback, to collect feedback from users regarding the accuracy of the datasets and the performance of the AI features. Use this feedback to further enhance and improve the accuracy of the datasets and refine the AI functionality.

By following the Agile methodology, the developers can develop the system iteratively and refine the coding for dataset accuracy, ensuring the AI-powered features effectively utilize the datasets in the mobile application.

4.2 Requirement Analysis

The data gathered will guide the analysis phase for determining the system requirements for the project, which includes the necessary software and hardware components. The proponents will present the project based on the collected data, seeking suggestions and approval from the panel before proceeding to the development phase.

4.2.1 Logic Diagram

The diagram below shows the logic diagram of the user's application view.

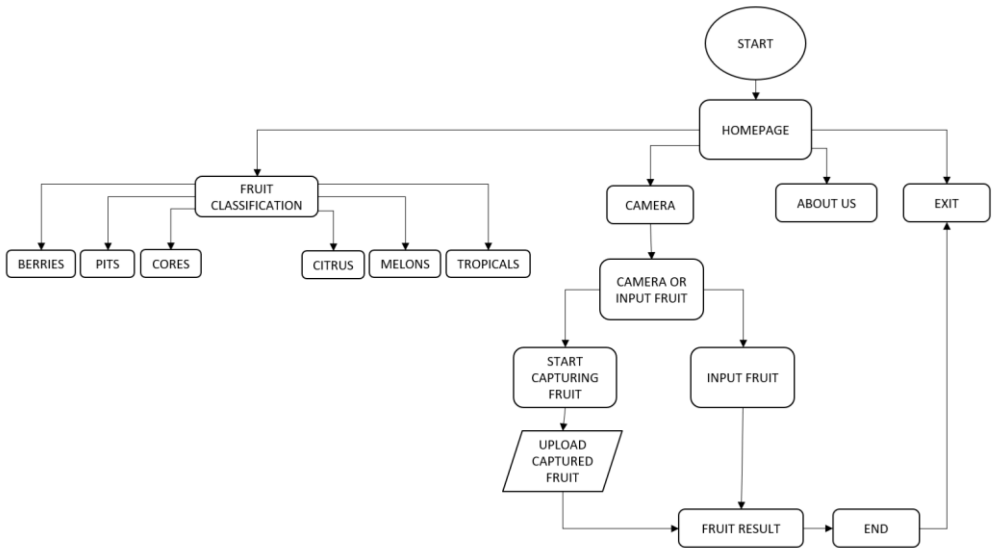


Figure 3. Logic Diagram

When a user launches an application, the name and logo are displayed, and they are then taken to the loading screen before being taken to the homepage. They may utilize the camera to employ image recognition or to input fruits, read fruit classifications, and see ‘about’.

4.2.2 Use Cases

Depicted in the diagram is the set of actions or functionalities that the application does for the user. It shows the users and system interaction.

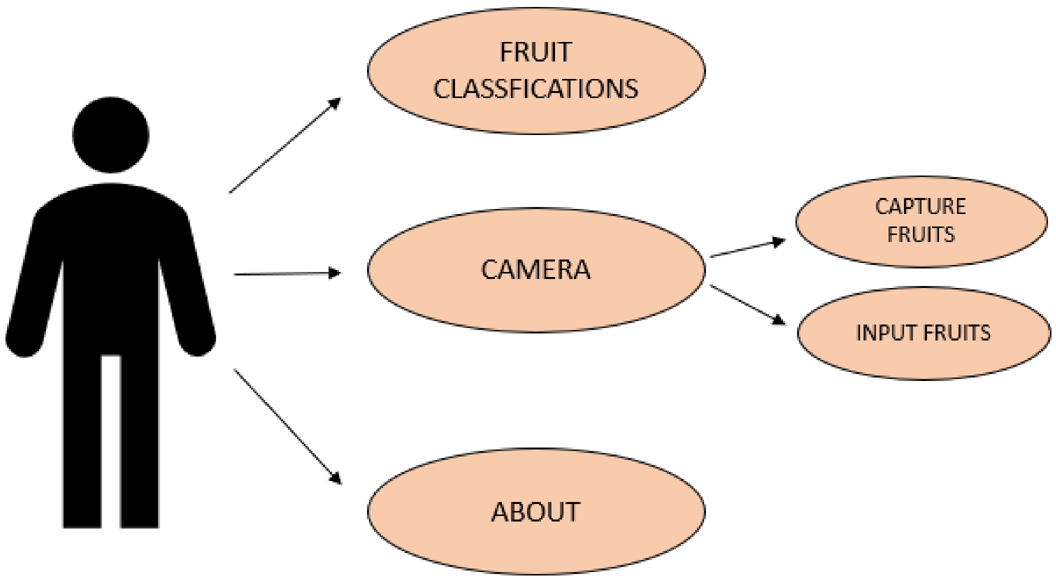


Figure 4. Use Case

The figure above depicts the "CyberFruition: AI-Based Fruit Classification and Recognition Framework" use case. Interaction between the actors, or between the user and the application. On the application's homepage, the user can read about fruit classifications by selecting fruit classifications. The user can also see and read more about the system by clicking on about us . The user can also access a feature for taking photos or entering fruits by pressing the camera button.

4.3 Requirement Specification/Documentation

This presents what are the needs in the requirement documentation. It will set the expected project outcome.

4.3.1 Purpose

This document presents the software requirements and comprehensive details of the project, CyberFruition. It is focused on providing information beneficial for the users. This will help them identify and enlighten them about the fruits being recognized.

4.3.2 Goals

The main objective of this project is to create an offline mobile application that promote fruits in the Philippines and to address the lack of knowledge about fruits. Android devices with built-in cameras and are compatible with this system. The users may look up a fruit using image recognition.

4.3.3 Project Scope

This study is limited only for users within Zamboanga City. The application can only recognize a minimum of two fruits visible in the camera. It can only detect ten (10) kinds of fruits and its primary function is to recognize multiple fruits presented.

4.3.4 Success Criteria

CyberFruition: AI-Based Fruit Classification and Recognition Framework project will be considered a success when:

- The system’s main module is working without errors.
- The system can recognize two fruits.
- The system’s image recognition works properly.
- The application meets all the requirements.

4.3.5 User Characteristics

This app requires the ability to use an Android smartphone and simple assistance menus. It only consists of one user.

- **Users -** Individuals who are interested in searching up fruit's information. Users must be able to provide some images of fruits through capture or upload

4.3.6 Mandated Constraints

The team will use JAVA XML to develop the system.

4.3.7 Non-Functional Requirements

Usability Requirements

- A loading animation of the logo will appear that will last no longer than 3 seconds.

Operational Requirements

- The device must be compatible with Android OS version 8 or higher.
- After scanning the image area, the system must pause for three to five seconds.
- The user can either select camera, fruit classifications, or about modules.

Performance Requirements

- Within five seconds, the system should have loaded the home screen.

Other Quality Attributes

- The application design should be user-friendly so that the users can quickly understand the usage even without assistance.

Documentation and Training

- Once the application is deployed, the training will be accomplished.

4.3.8 Functional Requirements

FR1: The system can identify the fruit through scanning.

FR2: The system can classify the fruit through scanning.

FR3: The system can show the nutritional benefits of the fruit that are scanned.

FR4: The system can recognize at least two fruits together.

FR5: The system shows the six types of fruits.

4.4 Design Software Systems. Product, Processes

The designs are made user-friendly. Interface design of the project was carefully planned to assure the ease of use of the users. The developers made sure to make it easy-to-understand with no difficulties to be encountered. The system flow was first examined then used to create wireframes for each module. The written story board shall be prototype in Figma then exported for development.

4.4.1 Process

The system designs are user-friendly for the user. The system is created so that the user can easily understand it and use it without any problems, which is one of its other quality attributes.

4.4.2 Data Flow Diagram (DFD)

Presented is the information flow of the system.

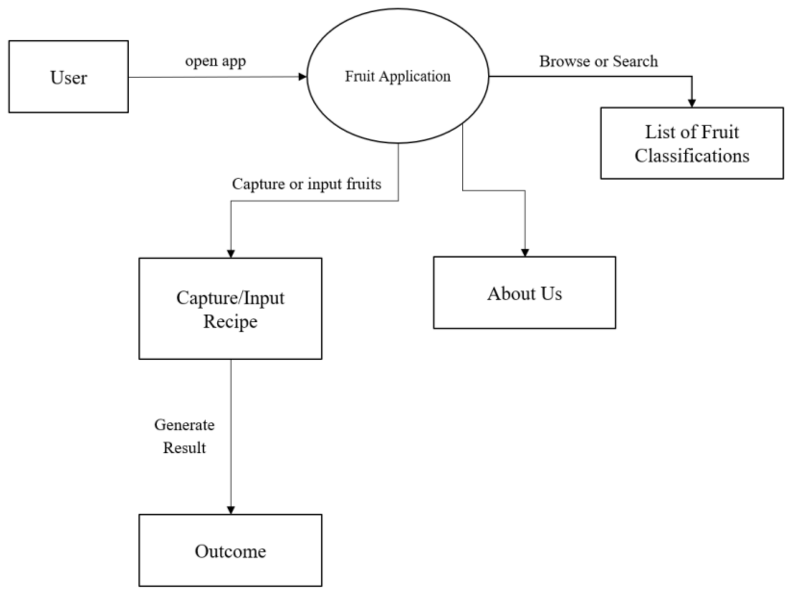


Figure 5. Data Flow Diagram

The level 0 DFD process of a fruit scanner application is comprised of the whole application.

4.4.3 User Interface Applications

The proponents utilized Figma, a design tool to create a prototype for the application interface. There is only one user of the application.

4.4.4 System Features

Capture or Input Fruits	This feature utilizes the device’s camera to identify and recognize fruits, providing information such as nutritional benefits and detailed fruit information.
Fruit Classification	This feature categorizes fruits into six distinct classifications, allowing users to easily distinguish between different types of fruits.
About	This section provides information about the developers of the application as well as the purpose behind its creation. It offers insights into the team behind the app and their motivations for developing it.

Table 2. System Features

CHAPTER 5

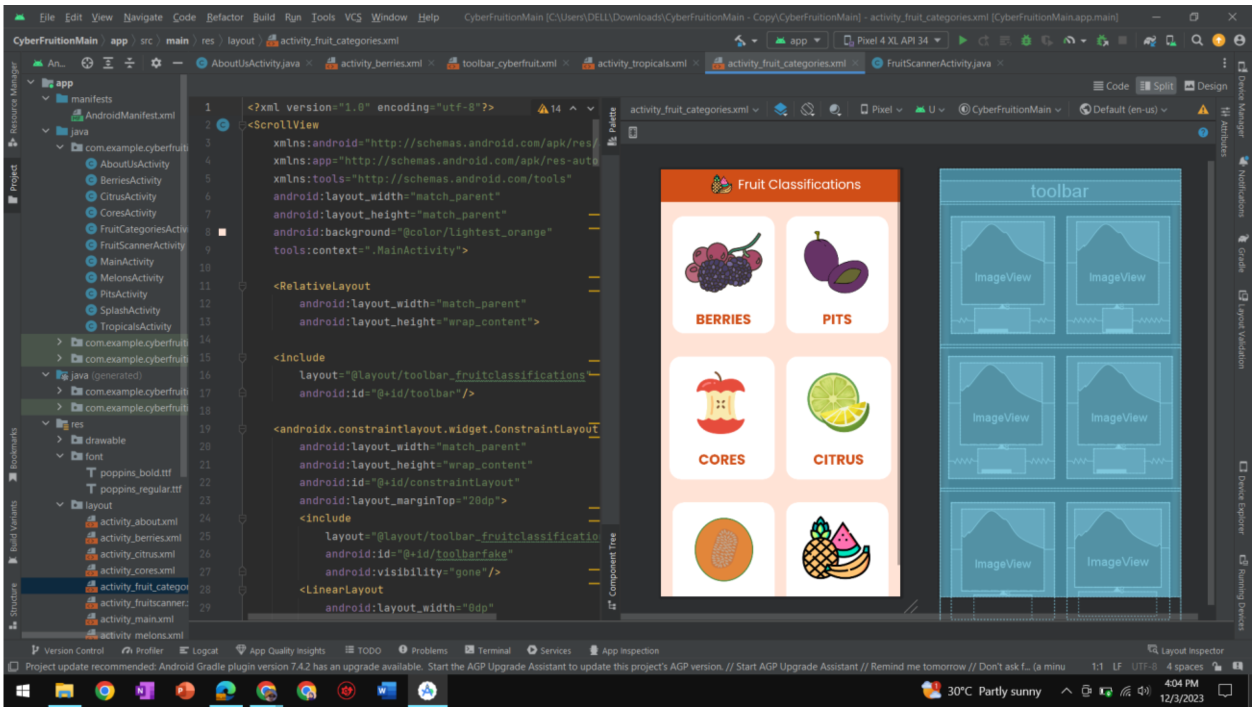
DEVELOPMENT AND TESTING

This chapter presents the graphical user interface, implementation plan, and the types of testing that the "CyberFruition: An AI-Based Fruit Classification and Recognition Framework" undergoes following the agile methodology.

5.1 Description of Prototype

The research team established a systematic process for building the prototype of the mobile application. Initially, ideas for mobile user interfaces were formed using wireframes, color schemes, typography, and images. Following this, the User Interface (UI) designer sketches out wireframes on paper, indicating the ideal placement of each feature within the system, its functionality, and the expected outcome.

The User Interface (UI) designer used Android Studio as the editing tool after conceptualizing, structuring, and sketching the prototype. All designs, including graphics, colors, contrast, and brightness, are digitally created to enhance every aspect of the mobile application. The completed prototype is then shared in the team's group chat for review, feedback, and assessment after being refined in Android Studio. Immediate modifications are sometimes necessary. After thorough evaluations, the rest of the team gave their approval.



```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/
    xmlns:app="http://schemas.android.com/apk/res-auto
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/lightest_orange"
    tools:context=".MainActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

    <include
        layout="@layout/toolbar_fruitclassifications"
        android:id="@+id/toolbar"/>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/constraintLayout"
        android:layout_marginTop="20dp">
        <include
            layout="@layout/toolbar_fruitclassificatio
            android:id="@+id/toolbarfake"
            android:visibility="gone"/>
        <LinearLayout
            android:layout_width="0dp"
```

Figure 6. UI Prototype

5.2 Integration of Visual Recognition

Teachable Machine presents an optimal solution for integrating AI-powered visual recognition into an application. This integration is specifically applied to the " CyberFruition: An AI-Based Fruit Classification and Recognition Framework ". Machine learning models are trained without the need for coding, utilizing Google's web-based tool, Teachable Machine. This tool simplifies the development of unique visual recognition models by providing a user-friendly interface.

The process to incorporate Teachable Machine's visual recognition into a mobile application are as follows:

1. **The model is trained using Teachable Machine:**
   a. A dataset of images for each class the app is expected to recognize is collected.
   b. These images are uploaded and labeled in Teachable Machine's web-based tool.
   c. The model is trained using Teachable Machine's interface, which employs transfer learning and pre-trained models to expedite training.
2. **The model is exported from Teachable Machine:**
   a. Once the model is trained and performing satisfactorily, it is exported in a format compatible with mobile applications.
   b. Teachable Machine offers options to export models in TensorFlow Lite format, which is suitable for mobile deployment.
3. **The mobile app development environment is set up:**
   a. The target platform for the mobile app (e.g., iOS or Android) is determined.
   b. The necessary development tools and frameworks for the chosen platform are installed. For Android, Android Studio and TensorFlow Lite may be used.
4. **A new mobile app project is created:**
   a. A new project is set up in the chosen mobile app development environment.
   b. The project settings and dependencies are configured based on the platform and frameworks being used.
5. **The Teachable Machine model is added to the mobile app project:**
   a. The exported Teachable Machine model file is imported into the mobile app project.
   b. The model file is placed in the appropriate location within the project's file structure.
6. **The model is integrated into the mobile app:**
   a. Code is written to load the Teachable Machine model into the mobile app.

b. The necessary preprocessing steps for the input images, such as resizing and normalization, are implemented.

c. The loaded model is used to make predictions based on the processed input images.

7. **Images are captured and processed in the mobile app:**

a. Features are implemented in the app to capture or select images for recognition.

b. The device's camera or image picker is utilized to obtain the input image data.

c. The image data is preprocessed based on the requirements of the Teachable Machine model.

8. **Predictions are made and actions are taken:**

a. The preprocessed image data is fed into the loaded Teachable Machine model.

b. Predictions are obtained from the model, which outputs probabilities or labels indicating the recognized object's class.

9. **The prediction results are used to perform specific actions within the mobile app, such as displaying information or triggering relevant functionalities. Testing and optimization:**

a. The mobile app is thoroughly tested to ensure the visual recognition works as expected.

b. The model is fine-tuned or the app's implementation is adjusted if necessary to improve accuracy or performance.

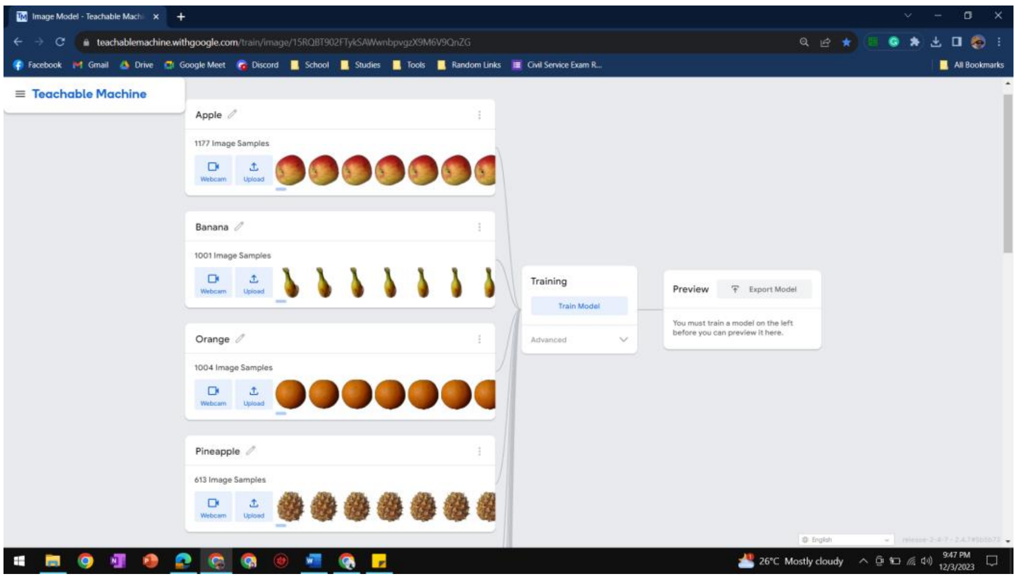


Figure 7. Train Image using Teachable Machine

5.3 Graphical User Interface (GUI)

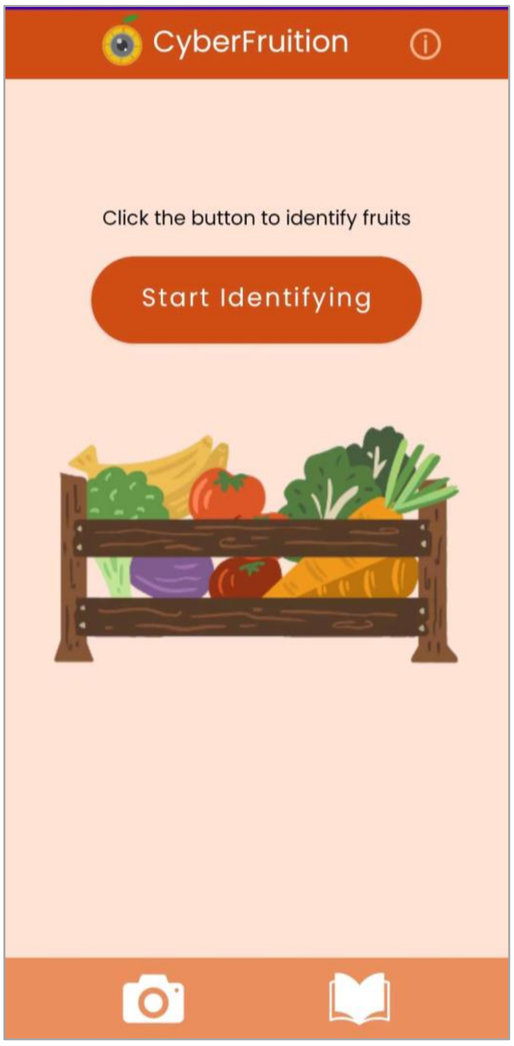


Figure 8. Home

Figure 8. The figure above is the home screen. This will pop-up after load screen when the users launch the application. There are four buttons which are the ‘start identifying’, camera, fruit classifications, and about. When the user presses ‘Start identifying’, this will head to the cameration options.

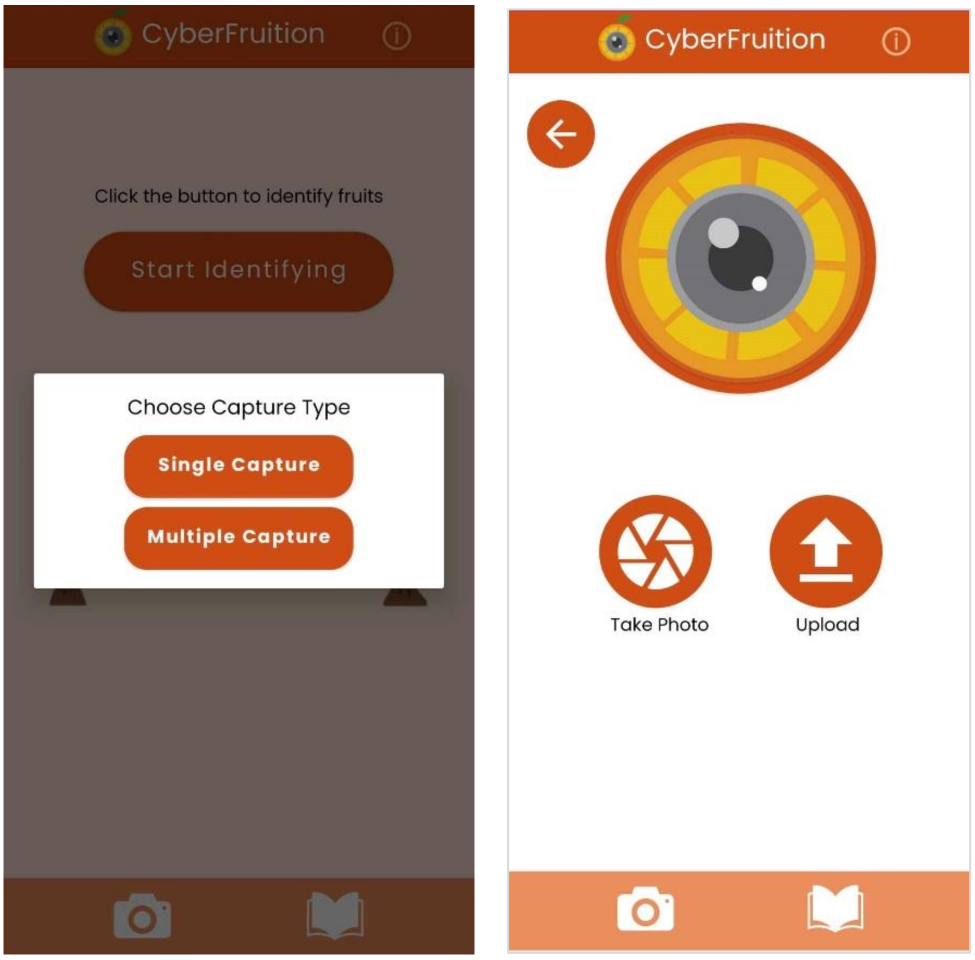
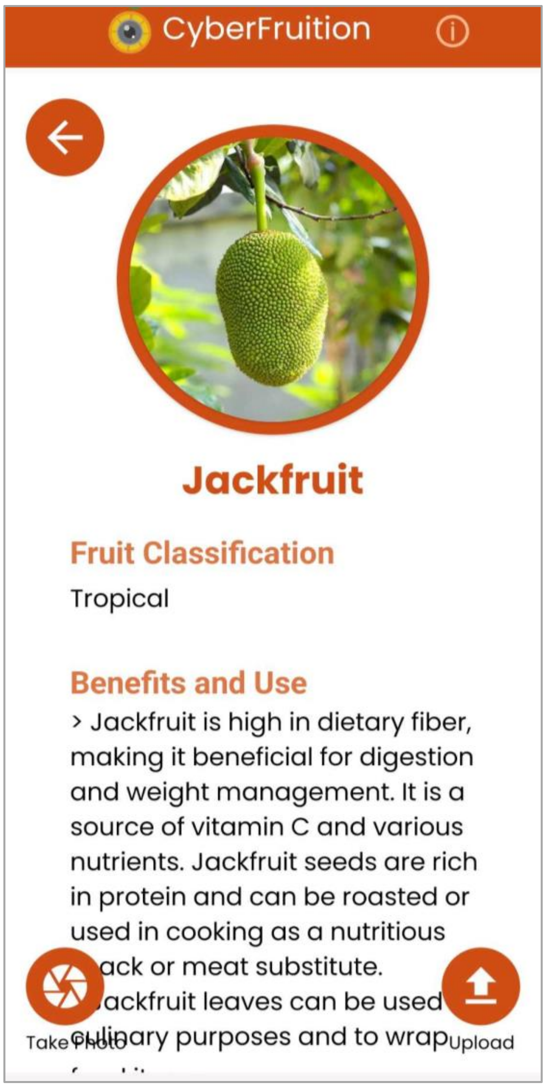


Figure 9. Fruit Camera Options

Figure 9. The figure above are the camera options. There are two capture type which are single and multiple capture. Once the user has chosen a capture type, the user may either take photo of the fruit or upload an image of it.



Jackfruit

Fruit Classification

Tropical

Benefits and Use

> Jackfruit is high in dietary fiber, making it beneficial for digestion and weight management. It is a source of vitamin C and various nutrients. Jackfruit seeds are rich in protein and can be roasted or used in cooking as a nutritious ack or meat substitute. ackfruit leaves can be used culinary purposes and to wrap

Figure 10. Fruit Result

Figure 10. This is the camera page. There are two icons on each side on the bottom, one for taking photo and the other for uploading a photo. The information that will be provided are the fruit classification, benefits, x and fun fact.

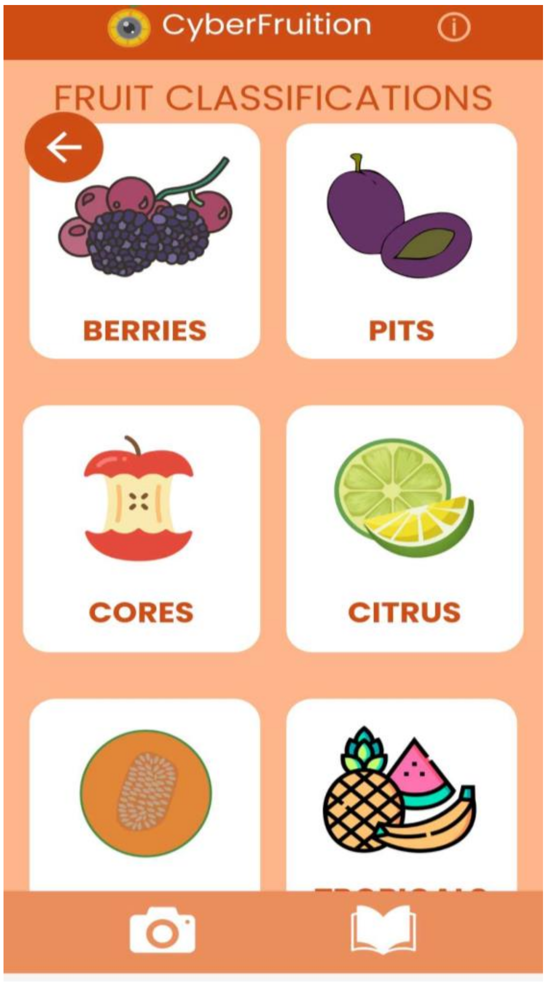
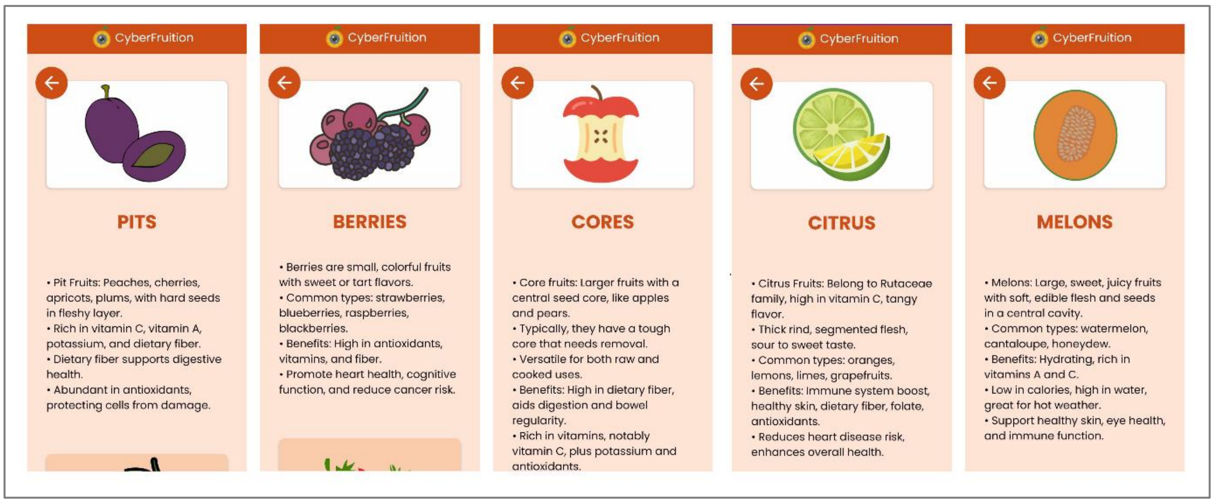


Figure 11. Fruit Classifications

Figure 11. This is the page for fruit classifications. The six fruit classifications are in the cards shown. Once opened, details and fruit examples of the classifications are displayed.



PITS

• Pit Fruits: Peaches, cherries, apricots, plums, with hard seeds in fleshy layer.
• Rich in vitamin C, vitamin A, potassium, and dietary fiber.
• Dietary fiber supports digestive health.
• Abundant in antioxidants, protecting cells from damage.

BERRIES

• Berries are small, colorful fruits with sweet or tart flavors.
• Common types: strawberries, blueberries, raspberries, blackberries.
• Benefits: High in antioxidants, vitamins, and fiber.
• Promote heart health, cognitive function, and reduce cancer risk.

CORES

• Core fruits: Larger fruits with a central seed core, like apples and pears.
• Typically, they have a tough core that needs removal.
• Versatile for both raw and cooked uses.
• Benefits: High in dietary fiber, aids digestion and bowel regularity.
• Rich in vitamins, notably vitamin C, plus potassium and antioxidants.

CITRUS

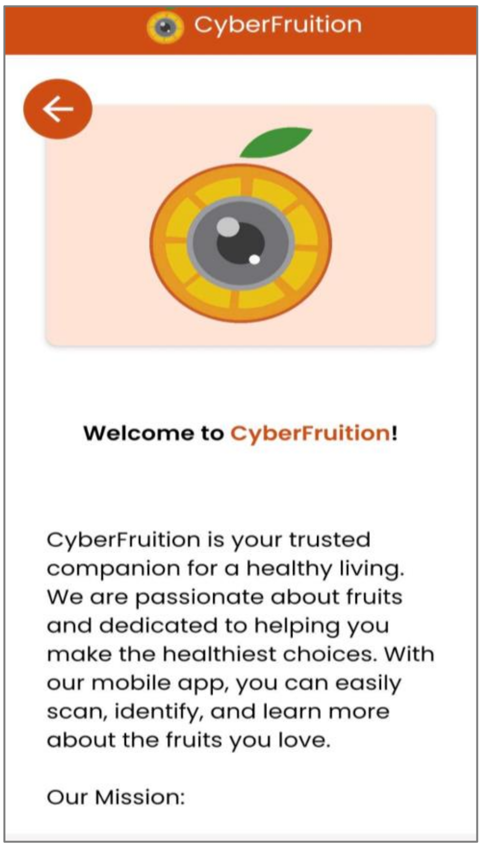
• Citrus Fruits: Belong to Rutaceae family, high in vitamin C, tangy flavor.
• Thick rind, segmented flesh, sour to sweet taste.
• Common types: oranges, lemons, limes, grapefruits.
• Benefits: Immune system boost, healthy skin, dietary fiber, folate, antioxidants.
• Reduces heart disease risk, enhances overall health.

MELONS

• Melons: Large, sweet, juicy fruits with soft, edible flesh and seeds in a central cavity.
• Common types: watermelon, cantaloupe, honeydew.
• Benefits: Hydrating, rich in vitamins A and C.
• Low in calories, high in water, great for hot weather.
• Support healthy skin, eye health, and immune function.

Figure 12. Inside Fruit Classifications

Figure 12. These are the pages of each fruit classification. Each page contains two fruit samples of the classification along with their brief descriptions.



Welcome to CyberFruition!

CyberFruition is your trusted companion for a healthy living. We are passionate about fruits and dedicated to helping you make the healthiest choices. With our mobile app, you can easily scan, identify, and learn more about the fruits you love.

Our Mission:

Figure 13. About

Figure 13. This page presents that information all about the system. Both the system purpose and the developer's goal.

5.4 Development

In the development phase, the front and back ends of the system were simultaneously created by the group. The programming language employed in this project is Java XML, Android Studio Code as the IDE. The members of the group identified functional features; Features must be 100% working. Once operational, it should be tested immediately, and whether the test fits the criteria of a working feature. Features are considered as successful.

The benefits of using Java XML are its flexibility and popularity as a standard for transferring and storing structured data. It is platform-independent, human-readable, adaptable, standardized, capable of data validation, and secure. These attributes make it an ideal choice for developers and companies seeking a reliable and flexible method to manage data.

5.5 Testing

To assess the precision of the mobile application, a series of tests were conducted by the researchers. These tests were designed to determine if the developer had fulfilled all the significant criteria to ensure the application's success.

5.5.1 Unit Testing

This testing process aids the developer in comprehending the code written throughout the development process. It allows for quicker modifications in case of an issue or error, enabling developers to verify the validity of the generated code and test the correct operation of each feature. The application consists of numerous modules, with several code scripts used for each feature. There is only one type of user, and the results of the unit testing are contained in Appendix F.

5.5.2 Alpha Testing

This testing assists the team in determining if the application meets the requirements and functions effectively. This testing is employed after all other tests have been completed and delivered. The findings from the Alpha testing are in Appendix E.

5.5.3 Beta Testing

The primary objective of this testing procedure is to ascertain if the actual product is functioning correctly. The team will undergo various testing stages, as mentioned above, to ensure that each module and component of the final product operates as expected. Additionally, this provides an excellent opportunity to help the team address any issues and bugs that may emerge during the process, enabling the delivery of a product with minimal bugs and stability.

5.6 Implementation Plan

From the development of the user interface to the completion of the development phase, the project had a timeline of 99 days. The agile methodology was planned to be used for iterative and incremental development. The main feature of the application was enhanced over time, with future versions expected to build upon it and add more features. The researchers' initial step was to submit a study proposal to the assigned topic advisor. Once the project was approved, the research commenced. The researchers developed a questionnaire for conducting a survey, a method of data collection. The respondents were any available students, including individuals from any household, and the project was a CyberFruition: An AI-Based Fruit Classification and Recognition Framework.

The researchers devised three Implementation Phases. Planning and analysis were to commence immediately. In the second Design Phase, the front-end developers chose the colors, typeface, font size, icons, buttons, and other design elements for the user interface (UI). The final step was Implementation, during which front-end and back-end engineers collaborated to execute all that had been planned and prepared. Each module successfully passed three (3) levels of testing after performing without error.

The initial test was unit testing, which verified the accuracy of each line of code created in each module of the application. Alpha testing was the second test, which encompassed all testing components. The final test was a beta test where actual application users evaluated whether the application modules passed or failed. After all the requirements were met and the project was successfully created, the CyberFruition: An AI-Based Fruit Classification and Recognition Framework was completed.

CHAPTER 6

RESULTS AND DISCUSSIONS

This chapter presents the results and discussion of the study “CyberFruition: AI-Based Fruit Classification and Recognition Framework”

6.1 Introduction

The team of researchers moved forward with the assessment of "CyberFruition: AI-Based Fruit Classification and Recognition Framework" by involving users to offer their feedback based on their experience with the mobile application. We collected a total of thirty-five (35) user responses.

After the app's development, the researchers initiated user testing, enabling users to share their feedback based on the app's functionality during their usage. We received ten responses from student users and twenty-five from regular individuals who acted as end users. All the respondents were residents of Zamboanga City, who evaluated the mobile application to confirm if all test cases were executed correctly.

Before commencing the beta test, we sought permission from our close friends, family, and relatives to participate as respondents for the end-user side. The researchers then sent an evaluation form to the selected respondents. We ensured that the respondents had our guidance and insights while filling out the evaluation and during their usage of the mobile application. They were all able to test the mobile application and record the outcomes of each test case in the evaluation form

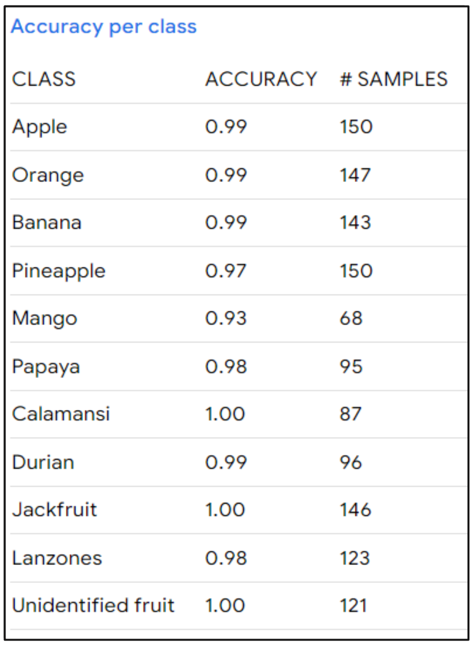
.

6.2 Result of Implementation

From the findings and results, specific objectives that the developers were able to achieve the following: to integrate image recognition feature to capture fruit using the device’s built-in camera, to scan and identify the fruits being detected, to perform solo or multiple recognition of fruits by pair, to create an interface of fruit classifications for simple navigation and browsing, to develop our own custom datasets of various fruits that will be utilized to train and assist in the creation of an image recognition model., to integrate image recognition feature to capture fruit using the device's built-in camera. Allow inputting and uploading of a minimum of 1 and a maximum of 2 fruits to be identified.

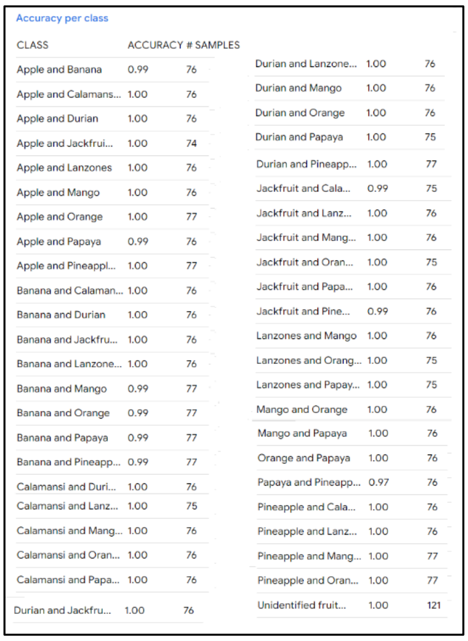
6.2.1 Result of Integration of Visual Recognition

The process of capturing and inputting fruits involves visual recognition. This is activated when users point their phone camera at actual fruits, initiating the image recognition feature. The developers of the application made use of a web-based tool, Teachable Machine from Google.com, to recognize images and objects. They also utilized TensorFlow, a machine learning framework that is open-source and developed by Google. This framework comprises the TensorFlow Object Identification API, which not only offers pre-trained models for object identification but also allows for the creation of custom models. The ten images used for training the visual recognition are displayed on the researcher's screen.



Accuracy per class

CLASS	ACCURACY	# SAMPLES
Apple	0.99	150
Orange	0.99	147
Banana	0.99	143
Pineapple	0.97	150
Mango	0.93	68
Papaya	0.98	95
Calamansi	1.00	87
Durian	0.99	96
Jackfruit	1.00	146
Lanzones	0.98	123
Unidentified fruit	1.00	121



Accuracy per class

CLASS	ACCURACY	# SAMPLES
Apple and Banana	0.99	76
Apple and Calamans...	1.00	76
Apple and Durian	1.00	76
Apple and Jackfrui...	1.00	74
Apple and Lanzones	1.00	76
Apple and Mango	1.00	76
Apple and Orange	1.00	77
Apple and Papaya	0.99	76
Apple and Pineappl...	1.00	77
Banana and Calaman...	1.00	76
Banana and Durian	1.00	76
Banana and Jackfru...	1.00	76
Banana and Lanzone...	1.00	76
Banana and Mango	0.99	77
Banana and Orange	0.99	77
Banana and Papaya	0.99	77
Banana and Pineapp...	0.99	77
Calamansi and Duri...	1.00	76
Calamansi and Lanz...	1.00	75
Calamansi and Mang...	1.00	76
Calamansi and Oran...	1.00	76
Calamansi and Papa...	1.00	76
Durian and Jackfru...	1.00	76
Durian and Lanzone...	1.00	76
Durian and Mango	1.00	76
Durian and Orange	1.00	76
Durian and Papaya	1.00	75
Durian and Pineapp...	1.00	77
Jackfruit and Cala...	0.99	75
Jackfruit and Lanz...	1.00	76
Jackfruit and Mang...	1.00	76
Jackfruit and Oran...	1.00	75
Jackfruit and Papa...	1.00	76
Jackfruit and Pine...	0.99	76
Lanzones and Mango	1.00	76
Lanzones and Orang...	1.00	75
Lanzones and Papay...	1.00	75
Mango and Orange	1.00	76
Mango and Papaya	1.00	76
Orange and Papaya	1.00	76
Papaya and Pineapp...	0.97	76
Pineapple and Cala...	1.00	76
Pineapple and Lanz...	1.00	76
Pineapple and Mang...	1.00	77
Pineapple and Oran...	1.00	77
Unidentified fruit...	1.00	121

Figure 14. Accuracy per class

The dataset given illustrates the precision of different fruit classifications, with the majority of classes exhibiting high accuracy levels ranging from 93% to 100%. However, certain classes like mango, papaya, and pineapple display noticeably lesser accuracy rates, which vary between 93% and 97%.

6.2.2 Usability Testing through End User Evaluation

An assessment of the app's usefulness and usability was conducted through an end-user evaluation. This involved 10 students from various universities studying different courses, and 25 respondents from different households. The System Usability Scale (SUS), a well-established and reliable survey that provides a consistent measure of a system's overall usability, was used to evaluate the system's usability.

Number of Student Respondents	Number of Household Respondents	Total Number of Respondents
10	25	35

Table 3. Total number of respondents

6.3 Summary of Findings

This section aims to provide a concise summary of the results discussed in Chapter 6. The information was collected using qualitative descriptive techniques, with Google Form survey questionnaires distributed to a total of thirty-five (35) participants.

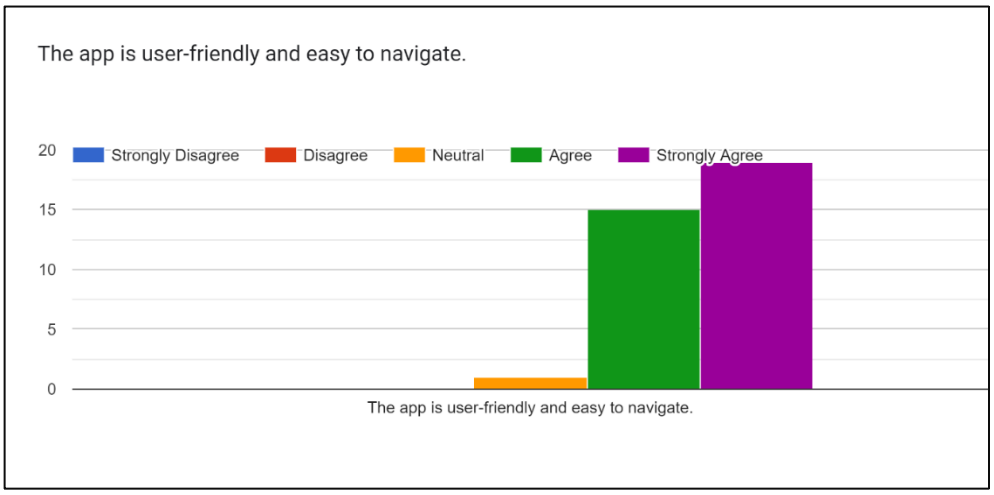


Figure 15. The app is user-friendly and easy to navigate.

Figure 15 shows that out of the total 35 respondents, the majority, 19 to be exact, strongly agree that the application is straightforward to navigate and use. In addition, 15 respondents agree with this view, while one respondent remains neutral.

These results suggest that a large segment of the respondents positively evaluated the application's usability and navigation, with most strongly affirming its user-friendliness.

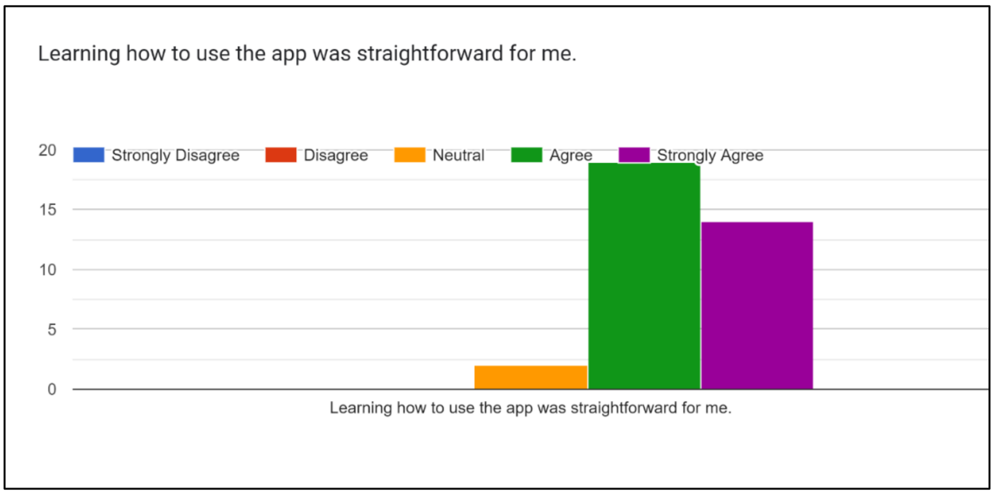


Figure 16. Learning how to use the app was straightforward for me.

As depicted in Figure 16 above, the majority of respondents, 14 out of 35, strongly agree that learning to use the application was easy for them. The remaining 19 respondents merely agree with this statement, while 2 others remain neutral.

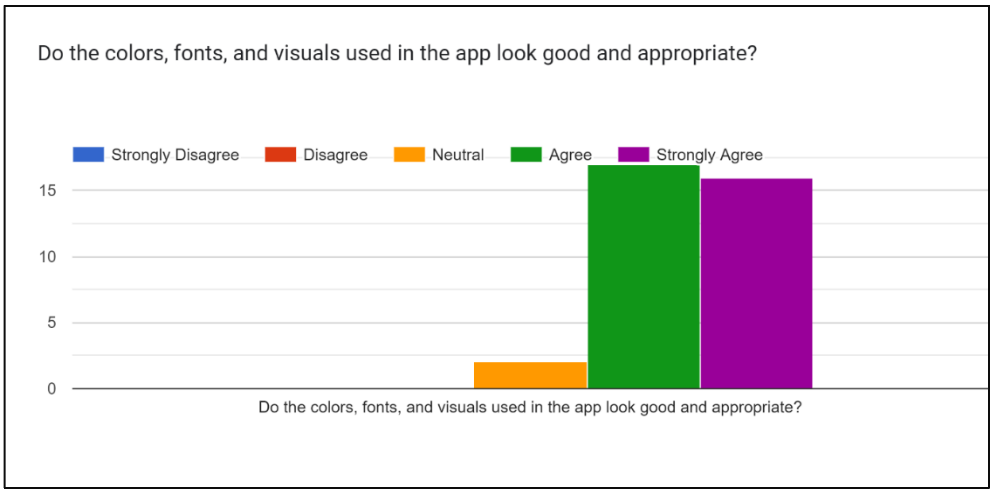


Figure 17. Do the colors, fonts, and visuals used in the app look good and appropriate?

As shown in Figure 17 above, out of the total 35 respondents, 16 strongly agree that the application's interface is consistent. In addition, 17 respondents also agree with this statement, and 2 others remain neutral.

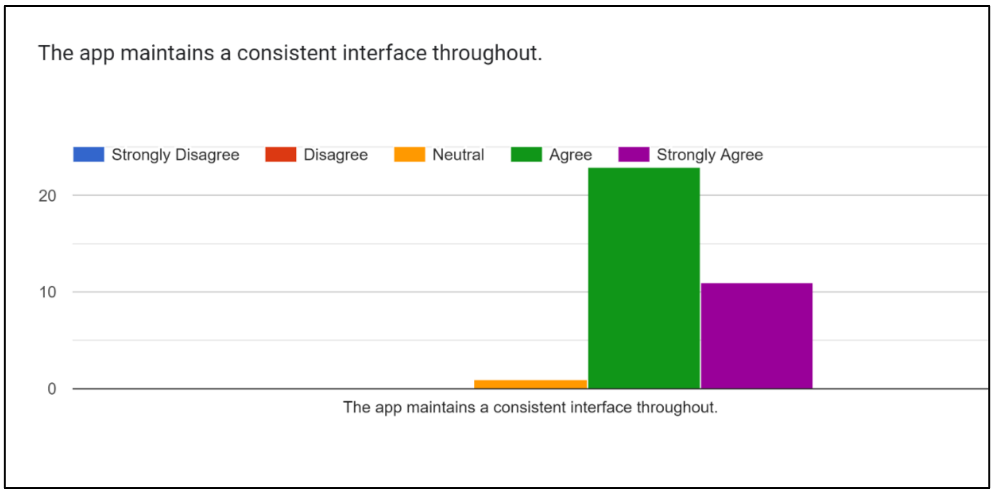


Figure 18. The app maintains a consistent interface 50 *throughout.*

Figure 18 above shows the results indicate that out of the total 35 respondents, 11 Strongly Agree with the consistency of the application’s interface, Additionally, 23 respondents Agree with it but perhaps not as strongly and 1 neutral.

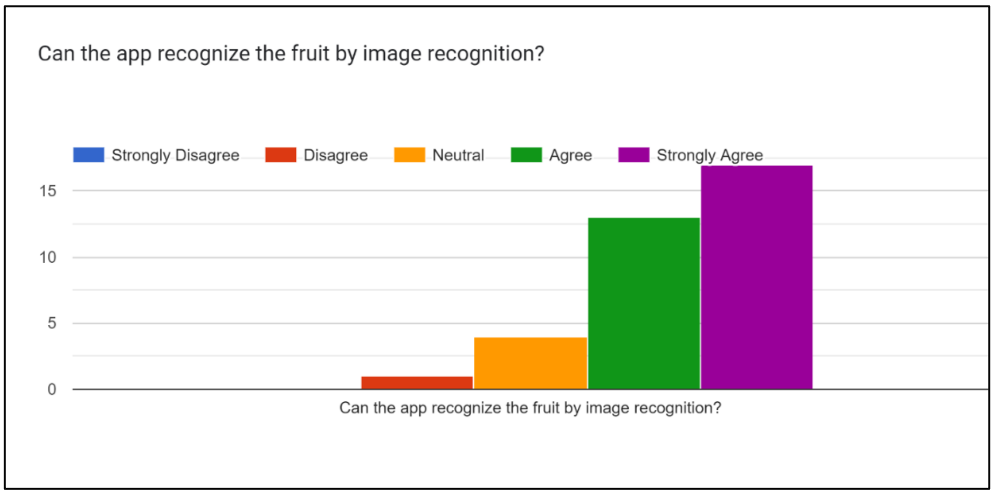


Figure 19. Can the app recognize the fruit by image recognition?

As depicted in Figure 19 above, out of the total 35 respondents, 11 strongly agree that the application's interface is consistent. Furthermore, 23 respondents also agree with this statement, and 1 respondent remains neutral.

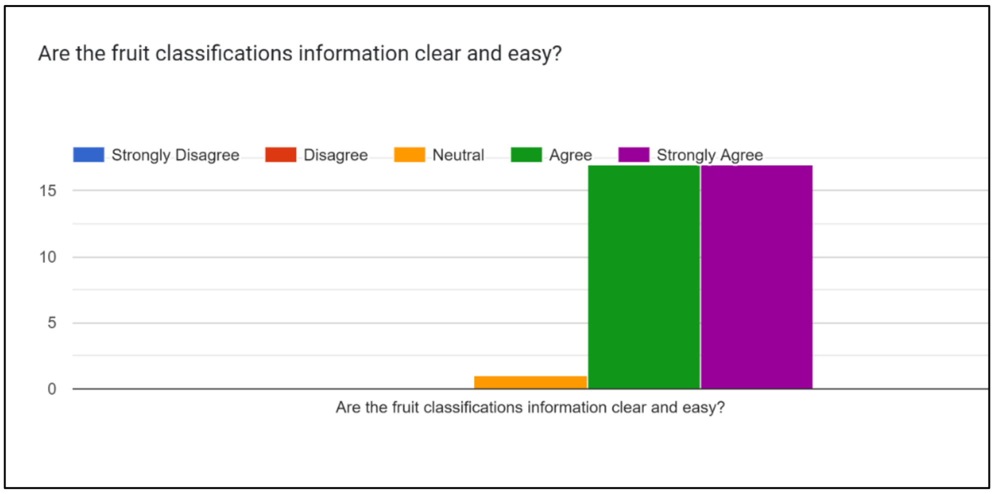


Figure 20. Are the fruit classifications information clear and easy?

Figure 20 above shows the results indicate that out of the total 35 respondents, 17 Strongly Agree that the generated recipes included clear and easy-to-follow instructions. Additionally, 17 respondents Agree with this statement, while 1 respondent remains neutral.

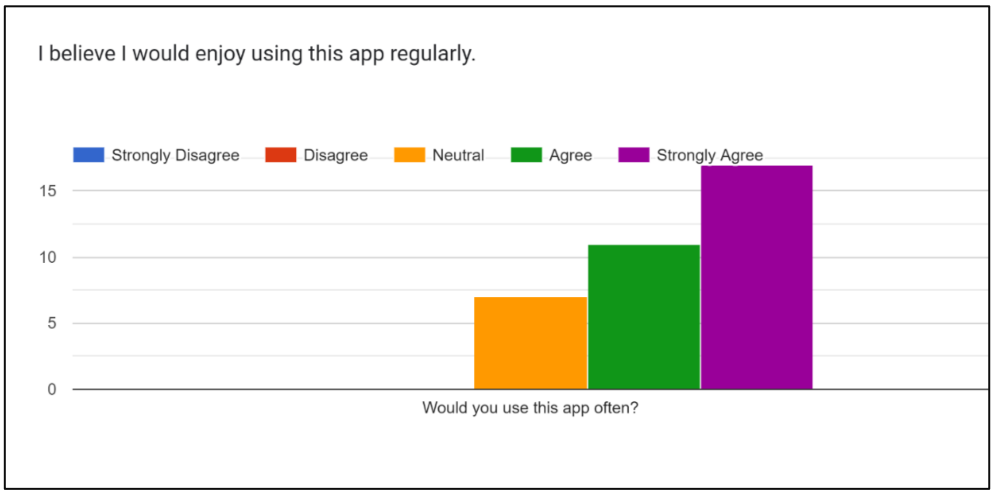


Figure 21. I believe I would enjoy using this app regularly.

As shown in Figure 21 above, the majority of respondents, 17 out of 35, strongly agree that they would like to use this application frequently. In addition, 11 respondents also agree, while 7 respondents remain neutral.

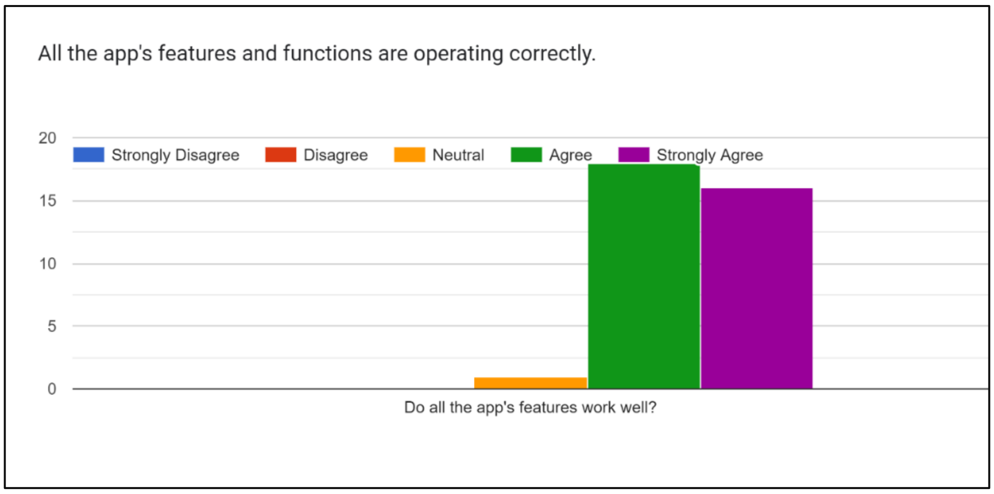


Figure 22. All the app's features and functions are operating correctly.

As depicted in Figure 22 above, out of the total 35 respondents, 16 strongly agree that all the features and functions of the application are working properly. Furthermore, 18 respondents also agree with this view, while one respondent remains neutral.

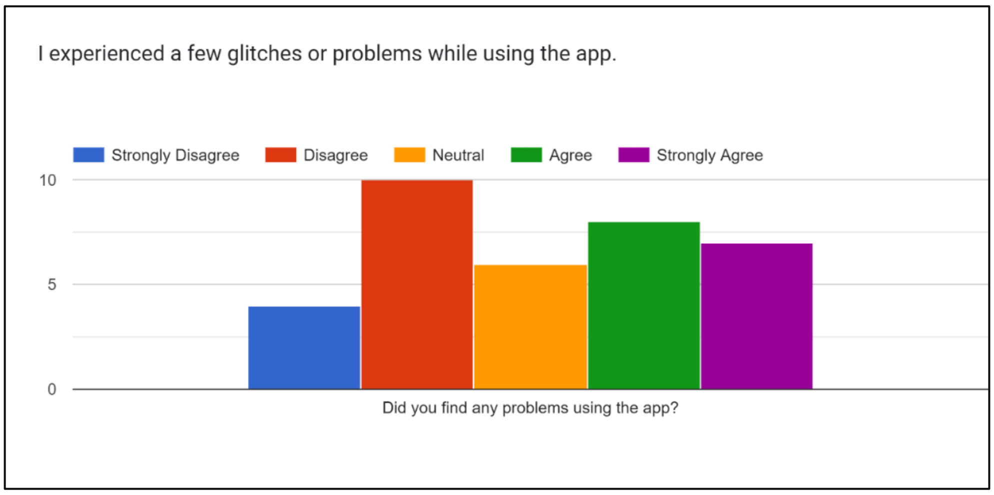


Figure 23. I experienced a few glitches or problems while using the app.

As illustrated in Figure 23 above, out of the total 35 respondents, 4 strongly disagree that they encountered any errors or issues while using the application. In addition, 10 respondents also disagree with this statement. On the other hand, 7 respondents strongly agree and 8 respondents agree that they did encounter a few errors or issues, while 6 respondents remain neutral.

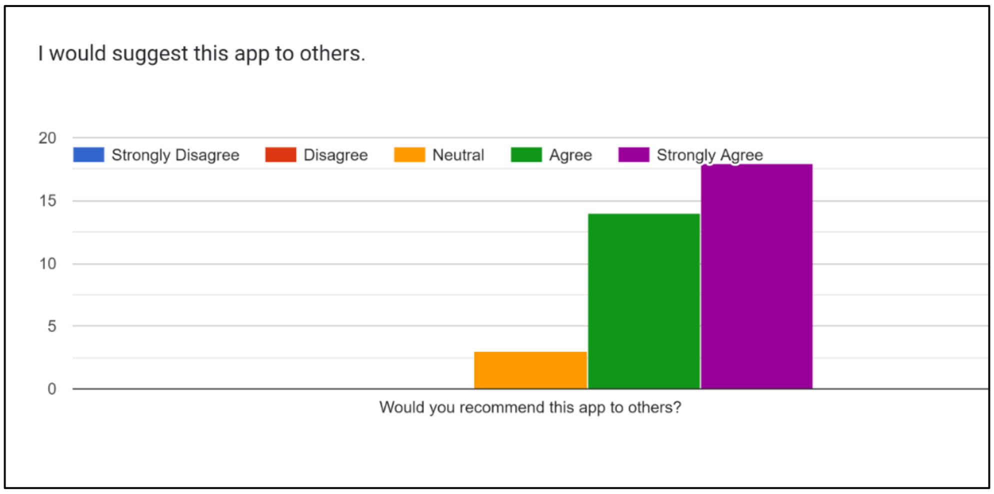


Figure 24. I would suggest this app to others.

As shown in Figure 24 above, among the 35 respondents, 18 strongly agree that they would recommend this application to others. Furthermore, 14 respondents also agree, while 3 respondents remain neutral.

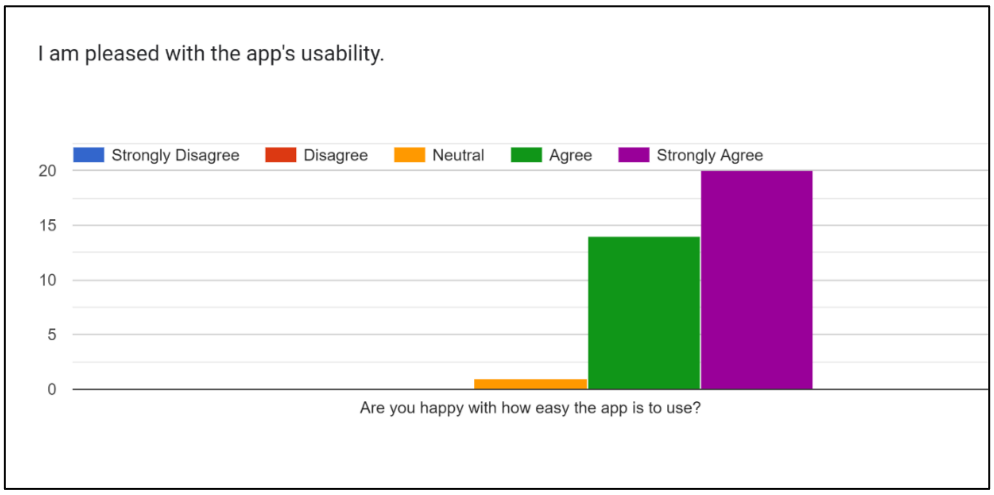


Figure 25. I am pleased with the app's usability.

As depicted in Figure 25 above, out of the total 35 respondents, 20 strongly agree that they are satisfied with the application's usability. In addition, 14 respondents also agree with this view, while one respondent remains neutral.

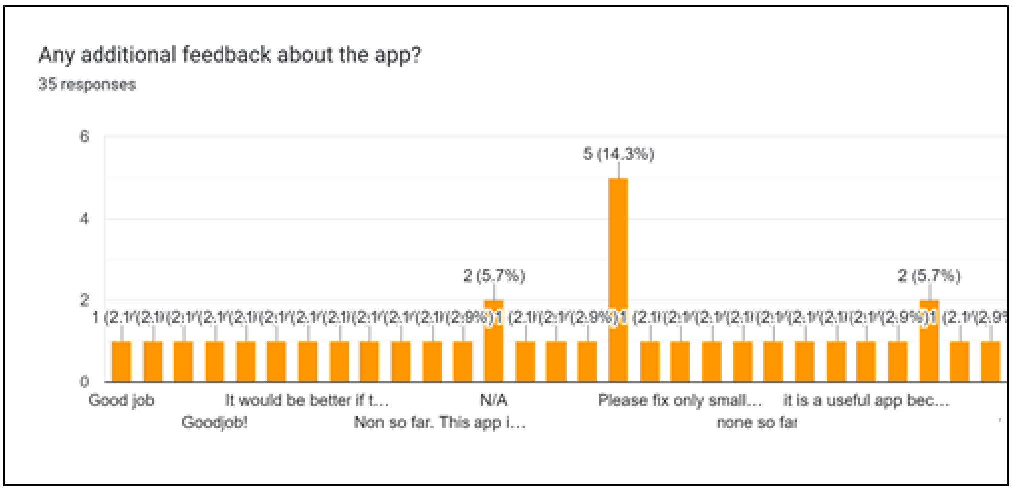


Figure 26. Any additional feedback about the app?

As presented in Figure 26 a survey involving thirty-five (35) respondents revealed that they recommend incorporating more information about fruits, including their various uses. Additionally, some respondents suggested including vegetables and expanding the fruit selection.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

This chapter presents the conclusion and recommendations made results of this study, “CyberFruition: AI-Based Fruit Classification and Recognition Framework".

7.1 Conclusions

The study, "CyberFruition: AI-Based Fruit Classification and Recognition Framework," centers on the creation of a mobile application designed to assist users in identifying and learning about fruits they have available. The research showcases the efficacy of visual recognition technology in a fruit identification app, addressing the common issue individuals encounter when needing to research and gather information about fruits.

In this study, researchers and tech enthusiasts highlight that a mobile application serves as an excellent platform for gaining more knowledge about fruits. The app's accessibility and portability, in comparison to traditional websites, provide users with a more convenient way to explore and learn about fruits.

The research team has successfully developed a mobile application that caters to the needs of fruit enthusiasts. The application presents a practical solution for anyone seeking information about a fruit based on their available components, by leveraging visual recognition technology.

7.2 Recommendations

Based on the summary of findings, conclusion and evaluation of the CyberFruition: AI-Based Fruit Classification and Recognition Framework that have been presented, the following recommendations are suggested

User Experience:

The application should integrate additional information about fruits, specifically their varied uses, incorporate vegetables, and increase in the variety of fruits to augment the application's utility for users.

REFERENCES

Lima, D., Liu, F. and Snetkov, L. (2017) Summary on fruit identification methods: A literature review, ResearchGate.

O, R. (2020) Top 50 must-try fruits in the Philippines, Pinoy Search.

Group, D.C. (2020) Agri chief: 'buy, promote local fruits', Official Portal of the Department of Agriculture.

Slavin, J. L., & Lloyd, B. (2012). Health benefits of fruits and vegetables. Advances in nutrition, 3(4), 506-516.

Alkahlout, M., Abu-Naser, S. S., Alsaqqa, A. H., & Abu-Jamie, T. N. (2022). Classification of A few Fruits Using Deep Learning.

Joseph, J. L., Kumar, V. A., & Mathew, S. P. (2021). Fruit classification using deep learning. In Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2021 (pp. 807-817). Springer Singapore.

Barbedo, J. G. A., Koenigkan, L. V., & Santos, T. T. (2016). Identifying multiple plant diseases using digital image processing. Biosystems engineering, 147, 104-116.

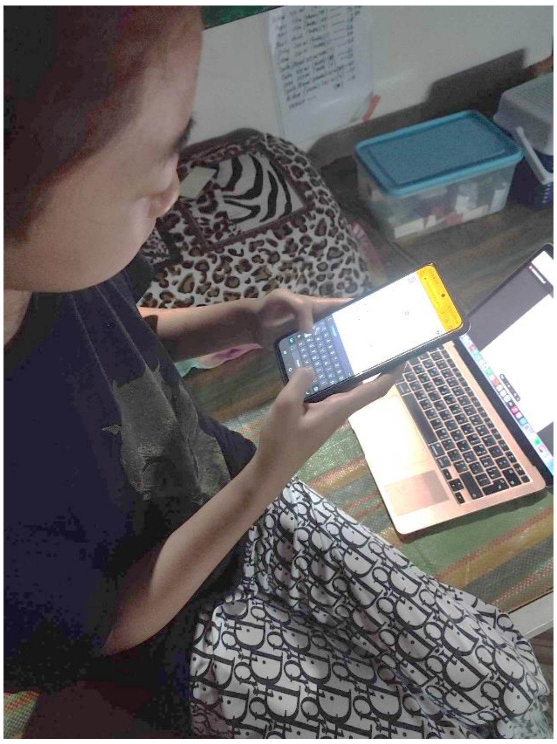
Patel, H. N., Jain, R. K., & Joshi, M. V. (2011). Fruit detection using improved multiple features-based algorithm. International journal of computer applications, 13(2), 1-5.

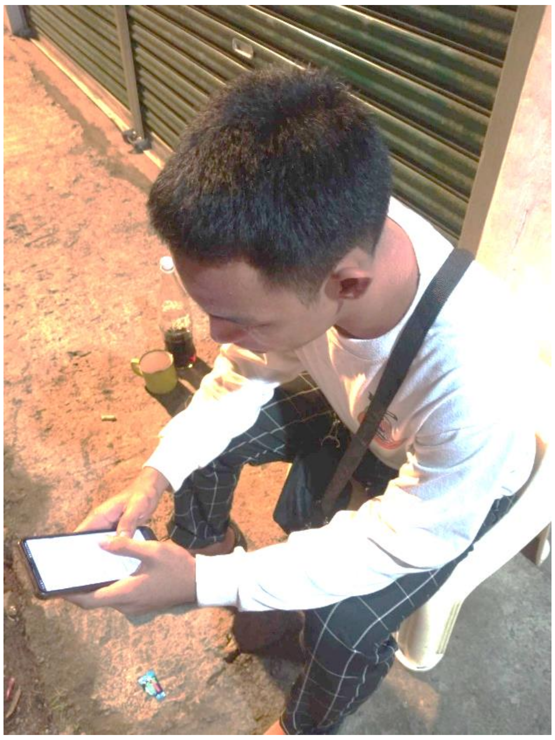
Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A review of convolutional neural network applied to fruit image processing. Applied Sciences, 10(10), 3443.

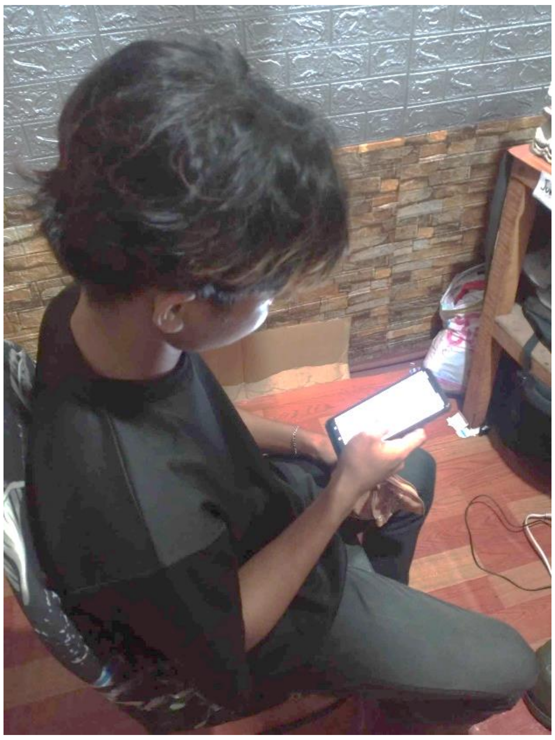
Pym, D., Spring, J. M., &amp; O&amp;rsquo;Hearn, P. (2018, May 22). Why separation logic works - philosophy &amp; technology. SpringerLink.

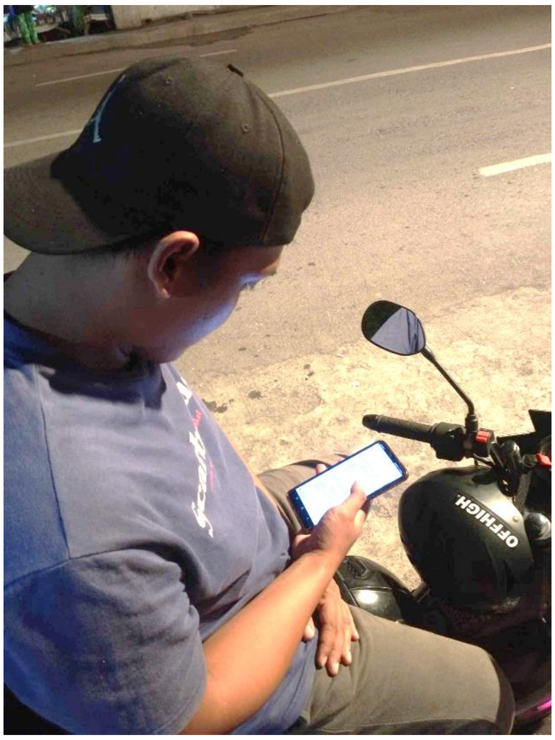
Mostafa Kabolizadeh, AbstractTimeliness, Gheyas, I. A., Gitelson, A. A., He, Y., Li, W.-G., Li, Z., Mandal, D., Maponya, M. G., Mercier, A., Pal, S., Peñuelas, J., Pudil, P., Rahmati, A., Rajoub, B., Rondeaux, G., Shafiee, S., Song, X.-P., Talema, T., … Chong, L. (2023, September 22). Improving classification accuracy for separation of area under crops based on feature selection from multi-temporal images and machine learning algorithms. Advances in Space Research.

APPENDIX A
DATA GATHERING









APPENDIX B
SOURCE CODE

CaptureMultipleActivity.java

```
public class CaptureMultipleActivity extends AppCompatActivity {

    TextView result;
    TextView emptyInfoTextView, bananaInfoTextView, orangeInfoTextView, appleInfoTextView,
pineappleInfoTextView, mangoInfoTextView, lanzonesInfoTextView, jackfruitInfoTextView,
appleAndJackfruitInfoTextView, appleAndLanzonesInfoTextView,
appleAndMangoInfoTextView, appleAndOrangeInfoTextView, papayaInfoTextView,
pineapplesInfoTextView, calamansiInfoTextView,
            appleAndPapayaInfoTextView, appleAndCalamansiInfoTextView,
appleAndPineappleInfoTextView, appleAndBananaInfoTextView,
bananaAndCalamansiInfoTextView, bananaAndPineappleInfoTextView,
bananaAndPapayaInfoTextView, bananaAndOrangeInfoTextView,
bananaAndMangoInfoTextView, bananaAndLansonesInfoTextView,
bananaAndJackfruitInfoTextView, jackfruitAndPapayaInfoTextView,
jackfruitAndMangoInfoTextView, jackfruitAndCalamansiInfoTextView,
jackfruitAndOrangeInfoTextView, jackfruitAndLanzonesInfoTextView,
            durianInfoTextView, bananaAndDurianInfoTextView,  durianAndJackfruitInfoTextView,
calamansiAndDurianInfoTextView, durianAndPineappleInfoTextView,
durianAndPapayaInfoTextView, durianAndMangoInfoTextView, durianAndOrangeInfoTextView,
durianAndLanzonesInfoTextView, pineappleAndCalamansiInfoTextView,
pineappleAndLanzonesInfoTextView, pineappleAndMangoInfoTextView,
pineappleAndOrangeInfoTextView, papayaAndPineappleInfoTextView,
appleAndDurianInfoTextView, lanzonesAndPapayaInfoTextView,
mangoAndPapayaInfoTextView, orangeAndPapayaInfoTextView,
calamansiAndPapayaInfoTextView, calamansiAndLanzonesInfoTextView,
calamansiAndMangoInfoTextView, calamansiAndOrangeInfoTextView,
lanzonesAndMangoInfoTextView, lanzonesAndOrangeInfoTextView,
mangoAndOrangeInfoTextView, jackfruitAndPineappleInfoTextView;

    ImageView imageView;
    int imageSize = 224;
    ImageButton buttonCaptureCamera, buttonSelectImage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setContentView(R.layout.activity_capture_multiple);

        result = findViewById(R.id.result);
        imageView = findViewById(R.id.imageView);

        // Informations
        emptyInfoTextView = findViewById(R.id.emptyInfoTextView);
        appleInfoTextView = findViewById(R.id.appleInfoTextView);
        bananaInfoTextView = findViewById(R.id.bananaInfoTextView);
        orangeInfoTextView = findViewById(R.id.orangeInfoTextView);
        pineappleInfoTextView = findViewById(R.id.pineappleInfoTextView);
        mangoInfoTextView = findViewById(R.id.mangoInfoTextView);
        lanzonesInfoTextView = findViewById(R.id.lanzonesInfoTextView);
        jackfruitInfoTextView = findViewById(R.id.jackfruitInfoTextView);
        appleAndJackfruitInfoTextView = findViewById(R.id.appleAndJackfruitInfoTextView);
        appleAndLanzonesInfoTextView = findViewById(R.id.appleAndLanzonesInfoTextView);
        appleAndMangoInfoTextView = findViewById(R.id.appleAndMangoInfoTextView);
        appleAndOrangeInfoTextView = findViewById(R.id.appleAndOrangeInfoTextView);
        papayaInfoTextView = findViewById(R.id.papayaInfoTextView);
        pineapplesInfoTextView = findViewById(R.id.pineapplesInfoTextView);
        calamansiInfoTextView = findViewById(R.id.calamansiInfoTextView);
        appleAndPapayaInfoTextView = findViewById(R.id.appleAndPapayaInfoTextView);
        appleAndCalamansiInfoTextView = findViewById(R.id.appleAndCalamansiInfoTextView);
        appleAndPineappleInfoTextView = findViewById(R.id.appleAndPineappleInfoTextView);
        appleAndBananaInfoTextView = findViewById(R.id.appleAndBananaInfoTextView);
        bananaAndCalamansiInfoTextView =
findViewById(R.id.bananaAndCalamansiInfoTextView);
        bananaAndPineappleInfoTextView =
findViewById(R.id.bananaAndPineappleInfoTextView);
        bananaAndPapayaInfoTextView = findViewById(R.id.bananaAndPapayaInfoTextView);
        bananaAndOrangeInfoTextView = findViewById(R.id.bananaAndOrangeInfoTextView);
        bananaAndMangoInfoTextView = findViewById(R.id.bananaAndMangoInfoTextView);
        bananaAndLansonesInfoTextView = findViewById(R.id.bananaAndLansonesInfoTextView);
        bananaAndJackfruitInfoTextView = findViewById(R.id.bananaAndJackfruitInfoTextView);
        jackfruitAndPapayaInfoTextView = findViewById(R.id.jackfruitAndPapayaInfoTextView);
        jackfruitAndMangoInfoTextView = findViewById(R.id.jackfruitAndMangoInfoTextView);
        jackfruitAndCalamansiInfoTextView =
findViewById(R.id.jackfruitAndCalamansiInfoTextView);
        jackfruitAndOrangeInfoTextView = findViewById(R.id.jackfruitAndOrangeInfoTextView);
        jackfruitAndLanzonesInfoTextView =
findViewById(R.id.jackfruitAndLanzonesInfoTextView);
        jackfruitAndPineappleInfoTextView =
findViewById(R.id.jackfruitAndPineappleInfoTextView);
        appleAndDurianInfoTextView = findViewById(R.id.appleAndDurianInfoTextView);
        durianInfoTextView = findViewById(R.id.durianInfoTextView);
        bananaAndDurianInfoTextView = findViewById(R.id.bananaAndDurianInfoTextView);
        bananaAndJackfruitInfoTextView = findViewById(R.id.bananaAndJackfruitInfoTextView);
        durianAndJackfruitInfoTextView = findViewById(R.id.durianAndJackfruitInfoTextView);
        calamansiAndDurianInfoTextView =
findViewById(R.id.calamansiAndDurianInfoTextView);
        durianAndPineappleInfoTextView = findViewById(R.id.durianAndPineappleInfoTextView);
        durianAndPapayaInfoTextView = findViewById(R.id.durianAndPapayaInfoTextView);
        durianAndMangoInfoTextView = findViewById(R.id.durianAndMangoInfoTextView);
        durianAndOrangeInfoTextView = findViewById(R.id.durianAndOrangeInfoTextView);
        durianAndLanzonesInfoTextView = findViewById(R.id.durianAndLanzonesInfoTextView);
        pineappleAndCalamansiInfoTextView =
findViewById(R.id.pineappleAndCalamansiInfoTextView);
        pineappleAndLanzonesInfoTextView =
findViewById(R.id.pineappleAndLanzonesInfoTextView);
        pineappleAndMangoInfoTextView = findViewById(R.id.pineappleAndMangoInfoTextView);
        pineappleAndOrangeInfoTextView =
findViewById(R.id.pineappleAndOrangeInfoTextView);
        papayaAndPineappleInfoTextView =
findViewById(R.id.papayaAndPineappleInfoTextView);
        appleInfoTextView = findViewById(R.id.appleInfoTextView);
        lanzonesAndPapayaInfoTextView = findViewById(R.id.lanzonesAndPapayaInfoTextView);
        mangoAndPapayaInfoTextView = findViewById(R.id.mangoAndPapayaInfoTextView);
        orangeAndPapayaInfoTextView = findViewById(R.id.orangeAndPapayaInfoTextView);
        calamansiAndPapayaInfoTextView =
findViewById(R.id.calamansiAndPapayaInfoTextView);
        calamansiAndLanzonesInfoTextView =
findViewById(R.id.calamansiAndLanzonesInfoTextView);
        calamansiAndMangoInfoTextView =
findViewById(R.id.calamansiAndMangoInfoTextView);
        calamansiAndOrangeInfoTextView =
findViewById(R.id.calamansiAndOrangeInfoTextView);
        lanzonesAndMangoInfoTextView = findViewById(R.id.lanzonesAndMangoInfoTextView);
        lanzonesAndOrangeInfoTextView = findViewById(R.id.lanzonesAndOrangeInfoTextView);
        mangoAndOrangeInfoTextView = findViewById(R.id.mangoAndOrangeInfoTextView);

        ImageButton buttonCaptureCamera = findViewById(R.id.buttonCaptureCamera);
        ImageButton buttonCaptureCameraBigger =
findViewById(R.id.buttonCaptureCameraBigger);
        ImageButton buttonSelectImage = findViewById(R.id.buttonSelectImage);
        ImageButton buttonSelectImageBigger = findViewById(R.id.buttonSelectImageBigger);

        buttonCaptureCamera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                captureImage();
            }
        });

        buttonCaptureCameraBigger.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                captureImage();
            }
        });

        buttonSelectImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                selectImage();
            }
        });

        buttonSelectImageBigger.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                selectImage();
            }
        });

        ImageButton buttonAbout = findViewById(R.id.aboutButton);
        ImageButton buttonBack = findViewById(R.id.buttonBack);

        buttonAbout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Navigate to Page 1
                Intent intent = new Intent(CaptureMultipleActivity.this,
AboutUsActivity_MultipleCapture.class);
```

```
                startActivity(intent);
                overridePendingTransition(0, 0);
            }
        });

        buttonBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Navigate to Page 2
                Intent intent = new Intent(CaptureMultipleActivity.this, HomeActivity.class);
                startActivity(intent);
                overridePendingTransition(0, 0);
            }
        });
        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);

        bottomNavigationView.setOnItemSelectedListener(item -> {
            switch (item.getItemId()){
                case R.id.camera:
                    showCustomDialog();
                    return true;

                case R.id.fruitClass:
                    startActivity(new Intent(getApplicationContext(), FruitCategoriesActivity.class));
                    Toast.makeText(CaptureMultipleActivity.this, "Fruit Classifications",
Toast.LENGTH_SHORT).show();
                    overridePendingTransition(0, 0);
                    finish();
                    return true;
            }
            return false;
        });
    }
    private void showItemInfo(String itemName, String itemInfo, TextView itemInfoTextView) {
        // Update the TextView with item information
        itemInfoTextView.setText(itemInfo);
        CharSequence styledText = HtmlCompat.fromHtml(itemInfo,
HtmlCompat.FROM_HTML_MODE_LEGACY);
        itemInfoTextView.setText(styledText);
        itemInfoTextView.setVisibility(View.VISIBLE);
    }


    public void classifyImage(Bitmap image){
        try {
            Model2 model2 = Model2.newInstance(getApplicationContext());

            // Creates inputs for reference.
            TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 224, 224, 3},
DataType.FLOAT32);
            ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * imageSize * imageSize * 3);
            byteBuffer.order(ByteOrder.nativeOrder());

            // get 1D array of 224 * 224 pixels in image
            int [] intValues = new int[imageSize * imageSize];
            image.getPixels(intValues, 0, image.getWidth(), 0, 0, image.getWidth(),
image.getHeight());

            // iterate over pixels and extract R, G, and B values. Add to bytebuffer.
            int pixel = 0;
            for(int i = 0; i < imageSize; i++){
                for(int j = 0; j < imageSize; j++){
                    int val = intValues[pixel++]; // RGB
                    byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f / 255.f));
                    byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f / 255.f));
                    byteBuffer.putFloat((val & 0xFF) * (1.f / 255.f));
                }
            }
            // Reset information TextViews before updating
            emptyInfoTextView.setText("");
            bananaInfoTextView.setText("");
            orangeInfoTextView.setText("");
            appleInfoTextView.setText("");
            pineappleInfoTextView.setText("");
            mangoInfoTextView.setText("");
            lanzonesInfoTextView.setText("");
            jackfruitInfoTextView.setText("");
            durianInfoTextView.setText("");
            appleAndJackfruitInfoTextView.setText("");
            appleAndLanzonesInfoTextView.setText("");
            appleAndMangoInfoTextView.setText("");
            appleAndOrangeInfoTextView.setText("");
            papayaInfoTextView.setText("");
            pineapplesInfoTextView.setText("");
            calamansiInfoTextView.setText("");
            appleAndPapayaInfoTextView.setText("");
            appleAndCalamansiInfoTextView.setText("");
            appleAndPineappleInfoTextView.setText("");
            appleAndBananaInfoTextView.setText("");
            bananaAndCalamansiInfoTextView.setText("");
            bananaAndPineappleInfoTextView.setText("");
            bananaAndPapayaInfoTextView.setText("");
            bananaAndOrangeInfoTextView.setText("");
            bananaAndMangoInfoTextView.setText("");
            bananaAndLansonesInfoTextView.setText("");
            bananaAndJackfruitInfoTextView.setText("");
            jackfruitAndPapayaInfoTextView.setText("");
            jackfruitAndMangoInfoTextView.setText("");
            jackfruitAndCalamansiInfoTextView.setText("");
            jackfruitAndOrangeInfoTextView.setText("");
            jackfruitAndLanzonesInfoTextView.setText("");
            jackfruitAndPineappleInfoTextView.setText("");
            durianInfoTextView.setText("");
            bananaAndDurianInfoTextView.setText("");
            bananaAndJackfruitInfoTextView.setText("");
            durianAndJackfruitInfoTextView.setText("");
            calamansiAndDurianInfoTextView.setText("");
            durianAndPineappleInfoTextView.setText("");
            durianAndPapayaInfoTextView.setText("");
            durianAndMangoInfoTextView.setText("");
            durianAndOrangeInfoTextView.setText("");
            durianAndLanzonesInfoTextView.setText("");
            pineappleAndCalamansiInfoTextView.setText("");
            pineappleAndLanzonesInfoTextView.setText("");
            pineappleAndMangoInfoTextView.setText("");
            pineappleAndOrangeInfoTextView.setText("");
            papayaAndPineappleInfoTextView.setText("");
            appleInfoTextView.setText("");
            lanzonesAndPapayaInfoTextView.setText("");
            mangoAndPapayaInfoTextView.setText("");
            orangeAndPapayaInfoTextView.setText("");
            calamansiAndPapayaInfoTextView.setText("");
            calamansiAndLanzonesInfoTextView.setText("");
            calamansiAndMangoInfoTextView.setText("");
            calamansiAndOrangeInfoTextView.setText("");
            lanzonesAndMangoInfoTextView.setText("");
            lanzonesAndOrangeInfoTextView.setText("");
            mangoAndOrangeInfoTextView.setText("");


            inputFeature0.loadBuffer(byteBuffer);


            // Runs model inference and gets result.
            Model2.Outputs outputs = model2.process(inputFeature0);
            TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

            float[] confidences = outputFeature0.getFloatArray();
            // find the index of the class with the biggest confidence.
            int maxPos = 0;
            float maxConfidence = 0;
            for(int i = 0; i < confidences.length; i++){
                if(confidences[i] > maxConfidence){
                    maxConfidence = confidences[i];
                    maxPos = i;
                }
            }
            String[] classes = {
                    "Apple and Banana",
                    "Apple and Calamansi",
                    "Apple and Durian",
                    "Apple and Jackfruit",
                    "Apple and Lanzones",
                    "Apple and Mango",
                    "Apple and Orange",
                    "Apple and Papaya",
                    "Apple and Pineapple",
                    "Banana and Calamansi",
                    "Banana and Durian",
                    "Banana and Jackfruit",
                    "Banana and Lanzones",
                    "Banana and Mango",
                    "Banana and Orange",
                    "Banana and Papaya",
                    "Banana and Pineapple",
                    "Calamansi and Durian",
                    "Calamansi and Lanzones",
                    "Calamansi and Mango",
                    "Calamansi and Orange",
                    "Calamansi and Papaya",
                    "Durian and Jackfruit",
                    "Durian and Lanzones",
                    "Durian and Mango",
                    "Durian and Orange",
                    "Durian and Papaya",
                    "Durian and Pineapple",
                    "Jackfruit and Calamansi",
                    "Jackfruit and Lanzones",
                    "Jackfruit and Mango",
                    "Jackfruit and Orange",
                    "Jackfruit and Papaya",
                    "Jackfruit and Pineapple",
                    "Lanzones and Mango",
                    "Lanzones and Orange",
                    "Lanzones and Papaya",
                    "Mango and Orange",
                    "Mango and Papaya",
                    "Orange and Papaya",
                    "Papaya and Pineapple",
                    "Pineapple and Calamansi",
                    "Pineapple and Lanzones",
                    "Pineapple and Mango",
                    "Pineapple and Orange",
                    "Unidentified fruits"
            };
            CardView cardView = findViewById(R.id.imageCard);
            result.setText(classes[maxPos]);
            switch (classes[maxPos]) {
                case "Apple and Jackfruit":
                    showItemInfo(getString(R.string.apple_and_jackfruit),
    getString(R.string.apple_and_jackfruit_info), appleAndJackfruitInfoTextView);
                    break;
                case "Apple and Lanzones":
                    showItemInfo(getString(R.string.apple_and_lanzones),
    getString(R.string.apple_and_lanzones_info), appleAndLanzonesInfoTextView);
                     break;
                case "Apple and Mango":
                    showItemInfo(getString(R.string.apple_and_mango),
    getString(R.string.apple_and_mango_info), appleAndMangoInfoTextView);
                    break;
                case "Apple and Orange":
                    showItemInfo(getString(R.string.apple_and_orange),
    getString(R.string.apple_and_orange_info), appleAndOrangeInfoTextView);
                    break;
                case "Apple and Papaya":
                    showItemInfo(getString(R.string.apple_and_papaya),
    getString(R.string.apple_and_papaya_info), appleAndPapayaInfoTextView);
                    break;
                case "Apple and Calamansi":
                    showItemInfo(getString(R.string.apple_and_calamansi),
    getString(R.string.apple_and_calamansi_info), appleAndCalamansiInfoTextView);
                    break;
                case "Apple and Pineapple":
                    showItemInfo(getString(R.string.apple_and_pineapple),
    getString(R.string.apple_and_pineapple_info), appleAndPineappleInfoTextView);
                    break;
                case "Apple and Banana":
                    showItemInfo(getString(R.string.apple_and_banana),
    getString(R.string.apple_and_banana_info), appleAndBananaInfoTextView);
                    break;
                case "Banana and Calamansi":
                    showItemInfo(getString(R.string.banana_and_calamansi),
    getString(R.string.banana_and_calamansi_info), bananaAndCalamansiInfoTextView);
                    break;
                case "Banana and Pineapple":
                    showItemInfo(getString(R.string.banana_and_pineapple),
    getString(R.string.banana_and_pineapple_info), bananaAndPineappleInfoTextView);
                    break;
                case "Banana and Papaya":
                    showItemInfo(getString(R.string.banana_and_papaya),
    getString(R.string.banana_and_papaya_info), bananaAndPapayaInfoTextView);
                    break;
                case "Banana and Orange":
                    showItemInfo(getString(R.string.banana_and_orange),
    getString(R.string.banana_and_orange_info), bananaAndOrangeInfoTextView);
                    break;
                case "Banana and Mango":
                    showItemInfo(getString(R.string.banana_and_mango),
    getString(R.string.banana_and_mango_info), bananaAndMangoInfoTextView);
                    break;
                case "Banana and Lanzones":
                    showItemInfo(getString(R.string.banana_and_lansones),
    getString(R.string.banana_and_lansones_info), bananaAndLansonesInfoTextView);
                    break;
                case "Banana and Jackfruit":
                    showItemInfo(getString(R.string.banana_and_jackfruit),
    getString(R.string.banana_and_jackfruit_info), bananaAndJackfruitInfoTextView);
                    break;
                case "Jackfruit and Papaya":
                    showItemInfo(getString(R.string.jackfruit_and_papaya),
    getString(R.string.jackfruit_and_papaya_info), jackfruitAndPapayaInfoTextView);
                    break;
                case "Jackfruit and Mango":
                    showItemInfo(getString(R.string.jackfruit_and_mango),
    getString(R.string.jackfruit_and_mango_info), jackfruitAndMangoInfoTextView);
                    break;
                case "Jackfruit and Calamansi":
                    showItemInfo(getString(R.string.jackfruit_and_calamansi),
    getString(R.string.jackfruit_and_calamansi_info), jackfruitAndCalamansiInfoTextView);
                    break;
                case "Jackfruit and Orange":
                    showItemInfo(getString(R.string.jackfruit_and_orange),
    getString(R.string.jackfruit_and_orange_info), jackfruitAndOrangeInfoTextView);
                    break;
                case "Jackfruit and Lanzones":
                  showItemInfo(getString(R.string.jackfruit_and_lanzones),
    getString(R.string.jackfruit_and_lanzones_info), jackfruitAndLanzonesInfoTextView);
                  break;
                case "Jackfruit and Pineapple":
                  showItemInfo(getString(R.string.jackfruit_and_pineapple),
    getString(R.string.jackfruit_and_pineapple_info), jackfruitAndPineappleInfoTextView);
                  break;
                case "Apple and Durian":
                  showItemInfo(getString(R.string.apple_and_durian),
    getString(R.string.apple_and_durian_info), appleAndDurianInfoTextView);
                  break;
                case "Banana and Durian":
                  showItemInfo(getString(R.string.banana_and_durian),
    getString(R.string.banana_and_durian_info), bananaAndDurianInfoTextView);
                  break;
                case "Durian and Jackfruit":
                  showItemInfo(getString(R.string.durian_and_jackfruit),
    getString(R.string.durian_and_jackfruit_info), durianAndJackfruitInfoTextView);
                  break;
                case "Calamansi and Durian":
                  showItemInfo(getString(R.string.durian_and_jackfruit),
    getString(R.string.durian_and_jackfruit_info), durianAndJackfruitInfoTextView);
                  break;
                case "Durian and Pineapple":
                  showItemInfo(getString(R.string.durian_and_pineapple),
    getString(R.string.durian_and_pineapple_info), durianAndPineappleInfoTextView);
                  break;
                case "Durian and Papaya":
                  showItemInfo(getString(R.string.durian_and_papaya),
    getString(R.string.durian_and_papaya_info), durianAndPapayaInfoTextView);
                  break;
                case "Durian and Mango":
                  showItemInfo(getString(R.string.durian_and_mango),
    getString(R.string.durian_and_mango_info), durianAndMangoInfoTextView);
                  break;
                case "Durian and Orange":
                  showItemInfo(getString(R.string.durian_and_orange),
    getString(R.string.durian_and_orange_info), durianAndOrangeInfoTextView);
                  break;
                case "Durian and Lanzones":
                  showItemInfo(getString(R.string.durian_and_lanzones),
    getString(R.string.durian_and_lanzones_info), durianAndLanzonesInfoTextView);
                  break;
                case "Pineapple and Calamansi":
                  showItemInfo(getString(R.string.pineapple_and_calamansi),
    getString(R.string.pineapple_and_calamansi_info), pineappleAndCalamansiInfoTextView);
                  break;
                case "Pineapple and Lanzones":
                  showItemInfo(getString(R.string.pineapple_and_lanzones),
    getString(R.string.pineapple_and_lanzones_info), pineappleAndCalamansiInfoTextView);
                  break;
                case "Pineapple and Mango":
                  showItemInfo(getString(R.string.pineapple_and_mango),
    getString(R.string.pineapple_and_mango_info), pineappleAndMangoInfoTextView);
```

```
                break;
            case "Pineapple and Orange":
                showItemInfo(getString(R.string.pineapple_and_orange),
getString(R.string.pineapple_and_orange_info), pineappleAndOrangeInfoTextView);
                break;
            case "Papaya and Pineapple":
                showItemInfo(getString(R.string.papaya_and_pineapple),
getString(R.string.papaya_and_pineapple_info), papayaAndPineappleInfoTextView);
                break;
            case "Lanzones and Papaya":
                showItemInfo(getString(R.string.lanzones_and_papaya),
getString(R.string.lanzones_and_papaya_info), lanzonesAndPapayaInfoTextView);
                break;
            case "Mango and Papaya":
                showItemInfo(getString(R.string.mango_and_papaya),
getString(R.string.mango_and_papaya_info), mangoAndPapayaInfoTextView);
                break;
            case "Orange and Papaya":
                showItemInfo(getString(R.string.orange_and_papaya),
getString(R.string.orange_and_papaya_info), orangeAndPapayaInfoTextView);
                break;
            case "Calamansi and Papaya":
                showItemInfo(getString(R.string.calamansi_and_papaya),
getString(R.string.calamansi_and_papaya_info), calamansiAndPapayaInfoTextView);
                break;
            case "Calamansi and Lanzones":
                showItemInfo(getString(R.string.calamansi_and_lanzones),
getString(R.string.calamansi_and_lanzones_info), calamansiAndLanzonesInfoTextView);
                break;
            case "Calamansi and Mango":
                showItemInfo(getString(R.string.calamansi_and_mango),
getString(R.string.calamansi_and_mango_info), calamansiAndMangoInfoTextView);
                break;
            case "Calamansi and Orange":
                showItemInfo(getString(R.string.calamansi_and_orange),
getString(R.string.calamansi_and_orange_info), calamansiAndOrangeInfoTextView);
                break;
            case "Lanzones and Mango":
                showItemInfo(getString(R.string.lanzones_and_mango),
getString(R.string.lanzones_and_mango_info), lanzonesAndMangoInfoTextView);
                break;
            case "Lanzones and Orange":
                showItemInfo(getString(R.string.lanzones_and_orange),
getString(R.string.lanzones_and_orange_info), lanzonesAndOrangeInfoTextView);
                break;
            case "Mango and Orange":
                showItemInfo(getString(R.string.mango_and_orange),
getString(R.string.mango_and_orange_info), mangoAndOrangeInfoTextView);
                break;
            case "Unidentified fruits":
                cardView.setCardBackgroundColor(Color.RED);
                result.setTextColor(Color.RED);
                break;
            default:
                findViewById(R.id.emptyInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.orangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.pineappleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.mangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.lanzonesInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.jackfruitInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndJackfruitInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndLanzonesInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndMangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndOrangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.papayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.pineapplesInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.calamansiInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndCalamansiInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndPineappleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndBananaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndCalamansiInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndPineappleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndOrangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndMangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndLansonesInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndJackfruitInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.jackfruitAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.jackfruitAndMangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.jackfruitAndCalamansiInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.jackfruitAndOrangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.jackfruitAndLanzonesInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.jackfruitAndPineappleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleAndDurianInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndDurianInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.bananaAndJackfruitInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianAndJackfruitInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.calamansiAndDurianInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianAndPineappleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianAndMangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianAndOrangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.durianAndLanzonesInfoTextView).setVisibility(View.GONE);

findViewById(R.id.pineappleAndCalamansiInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.pineappleAndLanzonesInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.pineappleAndMangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.pineappleAndOrangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.papayaAndPineappleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.appleInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.lanzonesAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.mangoAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.orangeAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.calamansiAndPapayaInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.calamansiAndLanzonesInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.calamansiAndMangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.calamansiAndOrangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.lanzonesAndMangoInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.lanzonesAndOrangeInfoTextView).setVisibility(View.GONE);
                findViewById(R.id.mangoAndOrangeInfoTextView).setVisibility(View.GONE);
                cardView.setCardBackgroundColor(ContextCompat.getColor(this, R.color.orange));

                break;
        }

        findViewById(R.id.buttonCaptureCamera).setVisibility(View.VISIBLE);
        findViewById(R.id.buttonSelectImage).setVisibility(View.VISIBLE);
        findViewById(R.id.textViewCaptureCamera).setVisibility(View.VISIBLE);
        findViewById(R.id.textViewSelectImage).setVisibility(View.VISIBLE);
        findViewById(R.id.textViewCaptureCameraBigger).setVisibility(View.GONE);
        findViewById(R.id.textViewSelectImageBigger).setVisibility(View.GONE);
        findViewById(R.id.bottom_navigation).setVisibility(View.GONE);
        hideBiggerImageButtons();

        // Releases model resources if no longer used.
        model2.close();
    } catch (IOException e) {
        // TODO Handle the exception
    }
}


@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if (requestCode == 1 && resultCode == RESULT_OK) {
        // File picker result
        try {
            // Get the selected image from the file picker
            InputStream inputStream = getContentResolver().openInputStream(data.getData());
            Bitmap image = BitmapFactory.decodeStream(inputStream);

            // Resize the image
            int dimension = Math.min(image.getWidth(), image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);

            // Display the image
            imageView.setImageBitmap(image);

            // Resize the image to the required input size for the model
            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);

            // Perform image classification
            classifyImage(image);

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    } else if (requestCode == 2 && resultCode == RESULT_OK) {
        // Camera capture result
        Bitmap image = (Bitmap) data.getExtras().get("data");
```

```
            int dimension = Math.min(image.getWidth(), image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage(image);
        }

        super.onActivityResult(requestCode, resultCode, data);
    }
    private void captureImage() {
        // Launch camera if we have permission
        if (checkSelfPermission(Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED) {
            Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, 2);
        } else {
            // Request camera permission if we don't have it.
            requestPermissions(new String[]{Manifest.permission.CAMERA}, 200);
        }


    }

    private void selectImage() {
        // Launch the file picker for image selection
        Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        intent.setType("image/*");
        startActivityForResult(intent, 1);

    }

    // Function to hide the bigger image buttons
    private void hideBiggerImageButtons() {
        findViewById(R.id.buttonCaptureCameraBigger).setVisibility(View.GONE);
        findViewById(R.id.buttonSelectImageBigger).setVisibility(View.GONE);
    }
    private void showCustomDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        LayoutInflater inflater = getLayoutInflater();
        View dialogView = inflater.inflate(R.layout.dialog_capture_options, null);

        builder.setView(dialogView)
                .setCancelable(true);

        AlertDialog dialog = builder.create();

        Button btnSingleCapture = dialogView.findViewById(R.id.btnSingleCapture);
        Button btnMultipleCapture = dialogView.findViewById(R.id.btnMultipleCapture);

        btnSingleCapture.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(CaptureMultipleActivity.this, CaptureSingleActivity.class);
            Toast.makeText(CaptureMultipleActivity.this, "Single Capture",
Toast.LENGTH_SHORT).show();
            startActivity(intent);
            overridePendingTransition(0, 0);
            dialog.dismiss();
         }
      });

      btnMultipleCapture.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Toast.makeText(CaptureMultipleActivity.this, "Multiple Capture",
Toast.LENGTH_SHORT).show();
            dialog.dismiss();
         }
      });

      dialog.show();
   }
}
```

CaptureSingleActivity.java

```
public class CaptureSingleActivity extends AppCompatActivity {

   TextView result;
   TextView emptyInfoTextView, bananaInfoTextView, orangeInfoTextView, appleInfoTextView,
pineappleInfoTextView, mangoInfoTextView, lanzonesInfoTextView, jackfruitInfoTextView,
papayaInfoTextView, calamansiInfoTextView, durianInfoTextView;

   ImageView imageView;
   int imageSize = 224;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_capture_single);

      result = findViewById(R.id.result);
      imageView = findViewById(R.id.imageView);

      // Informations
      emptyInfoTextView = findViewById(R.id.emptyInfoTextView);
      bananaInfoTextView = findViewById(R.id.bananaInfoTextView);
      orangeInfoTextView = findViewById(R.id.orangeInfoTextView);
      appleInfoTextView = findViewById(R.id.appleInfoTextView);
      pineappleInfoTextView = findViewById(R.id.pineappleInfoTextView);
      mangoInfoTextView = findViewById(R.id.mangoInfoTextView);
      lanzonesInfoTextView = findViewById(R.id.lanzonesInfoTextView);
      jackfruitInfoTextView = findViewById(R.id.jackfruitInfoTextView);
      papayaInfoTextView = findViewById(R.id.papayaInfoTextView);
      calamansiInfoTextView = findViewById(R.id.calamansiInfoTextView);
      durianInfoTextView = findViewById(R.id.durianInfoTextView);

      ImageButton buttonCaptureCamera = findViewById(R.id.buttonCaptureCamera);
```

```
        ImageButton buttonCaptureCameraBigger =
findViewById(R.id.buttonCaptureCameraBigger);
        ImageButton buttonSelectImage = findViewById(R.id.buttonSelectImage);
        ImageButton buttonSelectImageBigger = findViewById(R.id.buttonSelectImageBigger);

        buttonCaptureCamera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                captureImage();
            }
        });

        buttonCaptureCameraBigger.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                captureImage();
            }
        });

        buttonSelectImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                selectImage();
            }
        });

        buttonSelectImageBigger.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                selectImage();
            }
        });

        ImageButton buttonAbout = findViewById(R.id.aboutButton);
        ImageButton buttonBack = findViewById(R.id.buttonBack);

        buttonAbout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Navigate to Page 1
            Intent intent = new Intent(CaptureSingleActivity.this,
AboutUsActivity_SingleCapture.class);
            startActivity(intent);
            overridePendingTransition(0, 0);
            }
        });

        buttonBack.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Navigate to Page 2
            Intent intent = new Intent(CaptureSingleActivity.this, HomeActivity.class);
            startActivity(intent);
            overridePendingTransition(0, 0);
        }
```

```
    });
        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);

        bottomNavigationView.setOnItemSelectedListener(item -> {
            switch (item.getItemId()){
                case R.id.camera:
                    showCustomDialog();
                    return true;

                case R.id.fruitClass:
                    startActivity(new Intent(getApplicationContext(), FruitCategoriesActivity.class));
                    Toast.makeText(CaptureSingleActivity.this, "Fruit Classifications",
Toast.LENGTH_SHORT).show();
                    overridePendingTransition(0, 0);
                    finish();
                    return true;
            }
            return false;
        });
    }
    private void showItemInfo(String itemName, String itemInfo, TextView itemInfoTextView) {
        // Update the TextView with item information
        itemInfoTextView.setText(itemInfo);
        CharSequence styledText = HtmlCompat.fromHtml(itemInfo,
HtmlCompat.FROM_HTML_MODE_LEGACY);
        itemInfoTextView.setText(styledText);
        itemInfoTextView.setVisibility(View.VISIBLE);
    }


    public void classifyImage(Bitmap image){
        try {
            Model model = Model.newInstance(getApplicationContext());

            // Creates inputs for reference.
            TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 224, 224, 3},
DataType.FLOAT32);
            ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * imageSize * imageSize * 3);
            byteBuffer.order(ByteOrder.nativeOrder());

            // get 1D array of 224 * 224 pixels in image
            int [] intValues = new int[imageSize * imageSize];
            image.getPixels(intValues, 0, image.getWidth(), 0, 0, image.getWidth(),
image.getHeight());

            // iterate over pixels and extract R, G, and B values. Add to bytebuffer.
            int pixel = 0;
            for(int i = 0; i < imageSize; i++){
                for(int j = 0; j < imageSize; j++){
                    int val = intValues[pixel++]; // RGB
                    byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f / 255.f));
                    byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f / 255.f));
                    byteBuffer.putFloat((val & 0xFF) * (1.f / 255.f));
                }
            }
            // Reset information TextViews before updating
            emptyInfoTextView.setText("");
            bananaInfoTextView.setText("");
            orangeInfoTextView.setText("");
            appleInfoTextView.setText("");
            pineappleInfoTextView.setText("");
            mangoInfoTextView.setText("");
            lanzonesInfoTextView.setText("");
            jackfruitInfoTextView.setText("");
            papayaInfoTextView.setText("");
            calamansiInfoTextView.setText("");
            durianInfoTextView.setText("");


            inputFeature0.loadBuffer(byteBuffer);


            // Runs model inference and gets result.
            Model.Outputs outputs = model.process(inputFeature0);
            TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

            float[] confidences = outputFeature0.getFloatArray();
            // find the index of the class with the biggest confidence.
            int maxPos = 0;
            float maxConfidence = 0;
            for(int i = 0; i < confidences.length; i++){
                if(confidences[i] > maxConfidence){
                    maxConfidence = confidences[i];
                    maxPos = i;
                }
            }
            String[] classes = {
                    "Apple",
                    "Orange",
                    "Banana",
                    "Pineapple",
                    "Mango",
                    "Papaya",
                    "Calamansi",
                    "Durian",
                    "Jackfruit",
                    "Lanzones",
                    "Unidentified fruit"
            };
            CardView cardView = findViewById(R.id.imageCard);
            result.setText(classes[maxPos]);
            switch (classes[maxPos]) {
                case "Apple":
                    showItemInfo(getString(R.string.apple), getString(R.string.apple_info),
appleInfoTextView);
                    break;
                case "Banana":
                    showItemInfo(getString(R.string.banana), getString(R.string.banana_info),
bananaInfoTextView);
                    break;
                case "Orange":
                    showItemInfo(getString(R.string.orange), getString(R.string.orange_info),
orangeInfoTextView);
                    break;
                case "Pineapple":
                    showItemInfo(getString(R.string.pineapple), getString(R.string.pineapple_info),
pineappleInfoTextView);
                    break;
                case "Mango":
                    showItemInfo(getString(R.string.mango), getString(R.string.mango_info),
mangoInfoTextView);
                    break;
                case "Lanzones":
                    showItemInfo(getString(R.string.lanzones), getString(R.string.lanzones_info),
lanzonesInfoTextView);
                    break;
                case "Jackfruit":
                    showItemInfo(getString(R.string.jackfruit), getString(R.string.jackfruit_info),
jackfruitInfoTextView);
                    break;
                case "Papaya":
                    showItemInfo(getString(R.string.papaya), getString(R.string.papaya_info),
papayaInfoTextView);
                    break;
                case "Calamansi":
                    showItemInfo(getString(R.string.calamansi), getString(R.string.calamansi_info),
calamansiInfoTextView);
                    break;
                case "Durian":
                    showItemInfo(getString(R.string.durian), getString(R.string.durian_info),
durianInfoTextView);
                    break;
                case "Unidentified fruit":
                    cardView.setCardBackgroundColor(Color.RED);
                    result.setTextColor(Color.RED);
                    break;
                default:
                    findViewById(R.id.emptyInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.bananaInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.orangeInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.appleInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.pineappleInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.mangoInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.lanzonesInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.jackfruitInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.papayaInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.calamansiInfoTextView).setVisibility(View.GONE);
                    findViewById(R.id.durianInfoTextView).setVisibility(View.GONE);
                    cardView.setCardBackgroundColor(ContextCompat.getColor(this, R.color.orange));
                    break;
            }


            findViewById(R.id.buttonCaptureCamera).setVisibility(View.VISIBLE);
            findViewById(R.id.buttonSelectImage).setVisibility(View.VISIBLE);
            findViewById(R.id.textViewCaptureCamera).setVisibility(View.VISIBLE);
            findViewById(R.id.textViewSelectImage).setVisibility(View.VISIBLE);
            findViewById(R.id.textViewCaptureCameraBigger).setVisibility(View.GONE);
            findViewById(R.id.textViewSelectImageBigger).setVisibility(View.GONE);
            findViewById(R.id.bottom_navigation).setVisibility(View.GONE);
            hideBiggerImageButtons();


            // Releases model resources if no longer used.
            model.close();
        } catch (IOException e) {
            // TODO Handle the exception
        }
    }


    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        if (requestCode == 1 && resultCode == RESULT_OK) {
            // File picker result
            try {
                // Get the selected image from the file picker
                InputStream inputStream = getContentResolver().openInputStream(data.getData());
                Bitmap image = BitmapFactory.decodeStream(inputStream);

                // Resize the image
                int dimension = Math.min(image.getWidth(), image.getHeight());
                image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);

                // Display the image
                imageView.setImageBitmap(image);

                // Resize the image to the required input size for the model
                image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);

                // Perform image classification
                classifyImage(image);

                } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
        } else if (requestCode == 2 && resultCode == RESULT_OK) {
            // Camera capture result
            Bitmap image = (Bitmap) data.getExtras().get("data");
            int dimension = Math.min(image.getWidth(), image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage(image);

             }

        super.onActivityResult(requestCode, resultCode, data);
    }
```

```
    private void captureImage() {
        // Launch camera if we have permission
        if (checkSelfPermission(Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED) {
            Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, 2);
        } else {
            // Request camera permission if we don't have it.
            requestPermissions(new String[]{Manifest.permission.CAMERA}, 200);
        }


    }

    private void selectImage() {
        // Launch the file picker for image selection
        Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        intent.setType("image/*");
        startActivityForResult(intent, 1);

    }

    // Function to hide the bigger image buttons
    private void hideBiggerImageButtons() {
        findViewById(R.id.buttonCaptureCameraBigger).setVisibility(View.GONE);
        findViewById(R.id.buttonSelectImageBigger).setVisibility(View.GONE);
    }
    private void showCustomDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        LayoutInflater inflater = getLayoutInflater();
        View dialogView = inflater.inflate(R.layout.dialog_capture_options, null);

        builder.setView(dialogView)
                .setCancelable(true);

        AlertDialog dialog = builder.create();

        Button btnSingleCapture = dialogView.findViewById(R.id.btnSingleCapture);
        Button btnMultipleCapture = dialogView.findViewById(R.id.btnMultipleCapture);

        btnSingleCapture.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(CaptureSingleActivity.this, "Single Capture",
Toast.LENGTH_SHORT).show();
                dialog.dismiss();
            }
        });

        btnMultipleCapture.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(CaptureSingleActivity.this, CaptureMultipleActivity.class);
                Toast.makeText(CaptureSingleActivity.this, "Multiple Capture",
Toast.LENGTH_SHORT).show();
                startActivity(intent);
                overridePendingTransition(0, 0);
                dialog.dismiss();
            }
        });

        dialog.show();
    }
}
```

FruitCategoriesActivity.java

```
public class FruitCategoriesActivity extends AppCompatActivity {

  CardView berries,pits,cores,citrus,melons,tropicals;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    setContentView(R.layout.activity_fruit_categories);

    berries = findViewById(R.id.berriesCard);
    pits = findViewById(R.id.pitsCard);
    cores = findViewById(R.id.coresCard);
    citrus = findViewById(R.id.citrusCard);
    melons = findViewById(R.id.melonsCard);
    tropicals = findViewById(R.id.tropicalsCard);
    berries.setOnClickListener(view -> {
      Intent intent = new Intent(FruitCategoriesActivity.this, BerriesActivity.class);
      Toast.makeText(this, "Berries Classification", Toast.LENGTH_SHORT).show();
      startActivity(intent);
      overridePendingTransition(0, 0);
    });
    pits.setOnClickListener(view -> {
      Intent intent = new Intent(FruitCategoriesActivity.this, PitsActivity.class);
      Toast.makeText(this, "Pits Classification", Toast.LENGTH_SHORT).show();
      startActivity(intent);
      overridePendingTransition(0, 0);
    });
    cores.setOnClickListener(view -> {
      Intent intent = new Intent(FruitCategoriesActivity.this, CoresActivity.class);
      Toast.makeText(this, "Cores Classification", Toast.LENGTH_SHORT).show();
      startActivity(intent);
      overridePendingTransition(0, 0);
    });
    citrus.setOnClickListener(view -> {
      Intent intent = new Intent(FruitCategoriesActivity.this, CitrusActivity.class);
      Toast.makeText(this, "Citrus Classification", Toast.LENGTH_SHORT).show();
      startActivity(intent);
      overridePendingTransition(0, 0);
    });
```

```
        melons.setOnClickListener(view -> {
            Intent intent = new Intent(FruitCategoriesActivity.this, MelonsActivity.class);
            Toast.makeText(this, "Melons Classification", Toast.LENGTH_SHORT).show();
            startActivity(intent);
            overridePendingTransition(0, 0);
        });
        tropicals.setOnClickListener(view -> {
            Intent intent = new Intent(FruitCategoriesActivity.this, TropicalsActivity.class);
            Toast.makeText(this, "Tropicals Classification", Toast.LENGTH_SHORT).show();
            startActivity(intent);
            overridePendingTransition(0, 0);
        });

        ImageButton buttonAbout = findViewById(R.id.aboutButton);

        buttonAbout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Navigate to Page 1
                Intent intent = new Intent(FruitCategoriesActivity.this,
AboutUsActivity_FruitClassifications.class);
                Toast.makeText(FruitCategoriesActivity.this, "About Us",
Toast.LENGTH_SHORT).show();
                startActivity(intent);
                overridePendingTransition(0, 0);
            }
        });
        ImageButton buttonBack = findViewById(R.id.buttonBack);

        buttonBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Navigate to Page 2
                Intent intent = new Intent(FruitCategoriesActivity.this, HomeActivity.class);
                startActivity(intent);
                overridePendingTransition(0, 0);
            }
        });
        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);

        bottomNavigationView.setOnItemSelectedListener(item -> {
            switch (item.getItemId()){
                case R.id.camera:
                    showCustomDialog();
                    return true;

                case R.id.fruitClass:
                    Toast.makeText(FruitCategoriesActivity.this, "Fruit Classifications",
Toast.LENGTH_SHORT).show();
                    return true;
            }
            return false;
        });
    }
    private void showCustomDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        LayoutInflater inflater = getLayoutInflater();
        View dialogView = inflater.inflate(R.layout.dialog_capture_options, null);

        builder.setView(dialogView)
                .setCancelable(true);

        AlertDialog dialog = builder.create();

        Button btnSingleCapture = dialogView.findViewById(R.id.btnSingleCapture);
        Button btnMultipleCapture = dialogView.findViewById(R.id.btnMultipleCapture);

        btnSingleCapture.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(FruitCategoriesActivity.this, CaptureSingleActivity.class);
                Toast.makeText(FruitCategoriesActivity.this, "Single Capture",
Toast.LENGTH_SHORT).show();
                startActivity(intent);
                overridePendingTransition(0, 0);
                dialog.dismiss();
            }
        });

        btnMultipleCapture.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(FruitCategoriesActivity.this, CaptureMultipleActivity.class);
                Toast.makeText(FruitCategoriesActivity.this, "Multiple Capture",
Toast.LENGTH_SHORT).show();
                startActivity(intent);
                overridePendingTransition(0, 0);
                dialog.dismiss();
            }
        });

        dialog.show();
    }

}
```

HomeActivity.java

```
public class HomeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setContentView(R.layout.activity_home);

        ImageButton aboutUs = findViewById(R.id.aboutButton);
        Button homeButton = findViewById(R.id.centeredButton); // Replace with your actual button
ID

        aboutUs.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(HomeActivity.this, AboutUsActivity.class);
            Toast.makeText(HomeActivity.this, "About Us", Toast.LENGTH_SHORT).show();
            startActivity(intent);
            overridePendingTransition(0, 0);
        }
    });

    homeButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showCustomDialog();
        }
    });

    BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);

    bottomNavigationView.setOnItemSelectedListener(item -> {
        switch (item.getItemId()){
            case R.id.camera:
                showCustomDialog();
                return true;

            case R.id.fruitClass:
                startActivity(new Intent(getApplicationContext(), FruitCategoriesActivity.class));
                Toast.makeText(HomeActivity.this, "Fruit Classifications",
Toast.LENGTH_SHORT).show();
                overridePendingTransition(0, 0);
                finish();
                return true;
        }
        return false;
    });
}

private void showCustomDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    LayoutInflater inflater = getLayoutInflater();
    View dialogView = inflater.inflate(R.layout.dialog_capture_options, null);

    builder.setView(dialogView)
            .setCancelable(true);

    AlertDialog dialog = builder.create();

    Button btnSingleCapture = dialogView.findViewById(R.id.btnSingleCapture);
    Button btnMultipleCapture = dialogView.findViewById(R.id.btnMultipleCapture);

    btnSingleCapture.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(HomeActivity.this, CaptureSingleActivity.class);
            Toast.makeText(HomeActivity.this, "Single Capture", Toast.LENGTH_SHORT).show();
            startActivity(intent);
            overridePendingTransition(0, 0);
            dialog.dismiss();
        }
    });

    btnMultipleCapture.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(HomeActivity.this, CaptureMultipleActivity.class);
            Toast.makeText(HomeActivity.this, "Multiple Capture",
Toast.LENGTH_SHORT).show();
            startActivity(intent);
            overridePendingTransition(0, 0);
            dialog.dismiss();
        }
    });

    dialog.show();
  }

}
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    setContentView(R.layout.activity_main);
  }

  public void openFruitScannerActivity(View view) {
    // Build a dialog to let the user choose between Model 1, Model 2, and Model 3
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Choose a Model")
        .setItems(new CharSequence[]{"Single Capture", "Multiple Capture", "Model 3"}, new
DialogInterface.OnClickListener() {
          @Override
          public void onClick(DialogInterface dialog, int which) {
            // Handle the chosen model
            Class<?> selectedActivityClass;
            switch (which) {
              case 0:
                // Model 1 selected
                selectedActivityClass = CaptureSingleActivity.class;
                break;
              case 1:
                // Model 2 selected
                selectedActivityClass = CaptureMultipleActivity.class;
                break;
              case 2:
                // Model 3 selected
                selectedActivityClass = FruitScanner3Activity.class;
                break;
```

```
                    default:
                        // Default to Model 1
                        selectedActivityClass = CaptureSingleActivity.class;
                }

                // Start the selected scanner activity
                startFruitScannerActivity(selectedActivityClass);
            }
        });

        // Show the dialog
        builder.create().show();
    }

    private void startFruitScannerActivity(Class<?> activityClass) {
        // Start the selected scanner activity
        startActivity(new Intent(this, activityClass));
        showToast("Fruit Scanner");
        overridePendingTransition(0, 0);
    }

    public void openFruitCategoriesActivity(View view) {
        startActivity(new Intent(this, FruitCategoriesActivity.class));
        showToast("Fruit Categories");
        overridePendingTransition(0, 0);
    }

    public void openAboutUsActivity(View view) {
        startActivity(new Intent(this, AboutUsActivity.class));
        showToast("About Us");
        overridePendingTransition(0, 0);
    }
    private void showToast(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }
}
```

SpashActivity.java

```
public class SplashActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setContentView(R.layout.activity_splash);
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                startActivity(new Intent(SplashActivity.this, HomeActivity.class));
                finish();
            }
        }, 3000);
    }
}
```

APPENDIX C
TESTING

ID	TEST CASE	EXPECTED RESULT	PASS/FAILED
1	The user starts the Application	The user is brought to the Splash Screen, and it will direct to the Start Screen of the application	**PASS**
2	The user selects the" Start" button	The user is brought to the Dashboard Screen	**PASS**
3	The user will have a selection of different fruit classifications	The user will be brought to a certain fruit classification depending on what the user selects	**PASS**
4	The user selects the camera or "Start Identifying" button	The user is brought to the Main Function Screen	**PASS**
5	The user will capture or input 1 to 2 fruits	The user's captured or inputted fruits with its information will be shown on screen.	**PASS**

Project Name:	CyberFruition: AI-Based Fruit Classification and Recognition Framework
Module Name:	Start Screen
Reference Document:	Functional Requirement
Created By:	Tester
Date of Creation:	
Date of Review/Execution:	

Test Case ID	**Test Scenario**	**Test Case**	**Pre-Condition**	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Status (Pass/Fail)**
Start_001	Verify if the app wont crash	Press Start Button	Directs to Dashboard Screen	Start button	Start button	Succesfully started	Dashboard screen is shown	Same as expected	PASS
Start_002	Verify if the app wont crash	Press Start Button	Directs to Dashboard Screen	Start button	The app has crashed	Start App Failed	Stays on start screen	Stays on start screen	FAILED
Start_003	Verify if the app wont crash	Press Start Button	Directs to Dashboard Screen	Start button	Start button	Succesfully started	Dashboard screen is shown	Same as expected	PASS

Project Name:	CyberFruition: AI-Based Fruit Classification and Recognition Framework
Module Name:	Fruit Classifications (Dashboard Screen)
Reference Document:	Functional Requirement
Created By:	Tester
Date of Creation:	
Date of Review/Execution:	

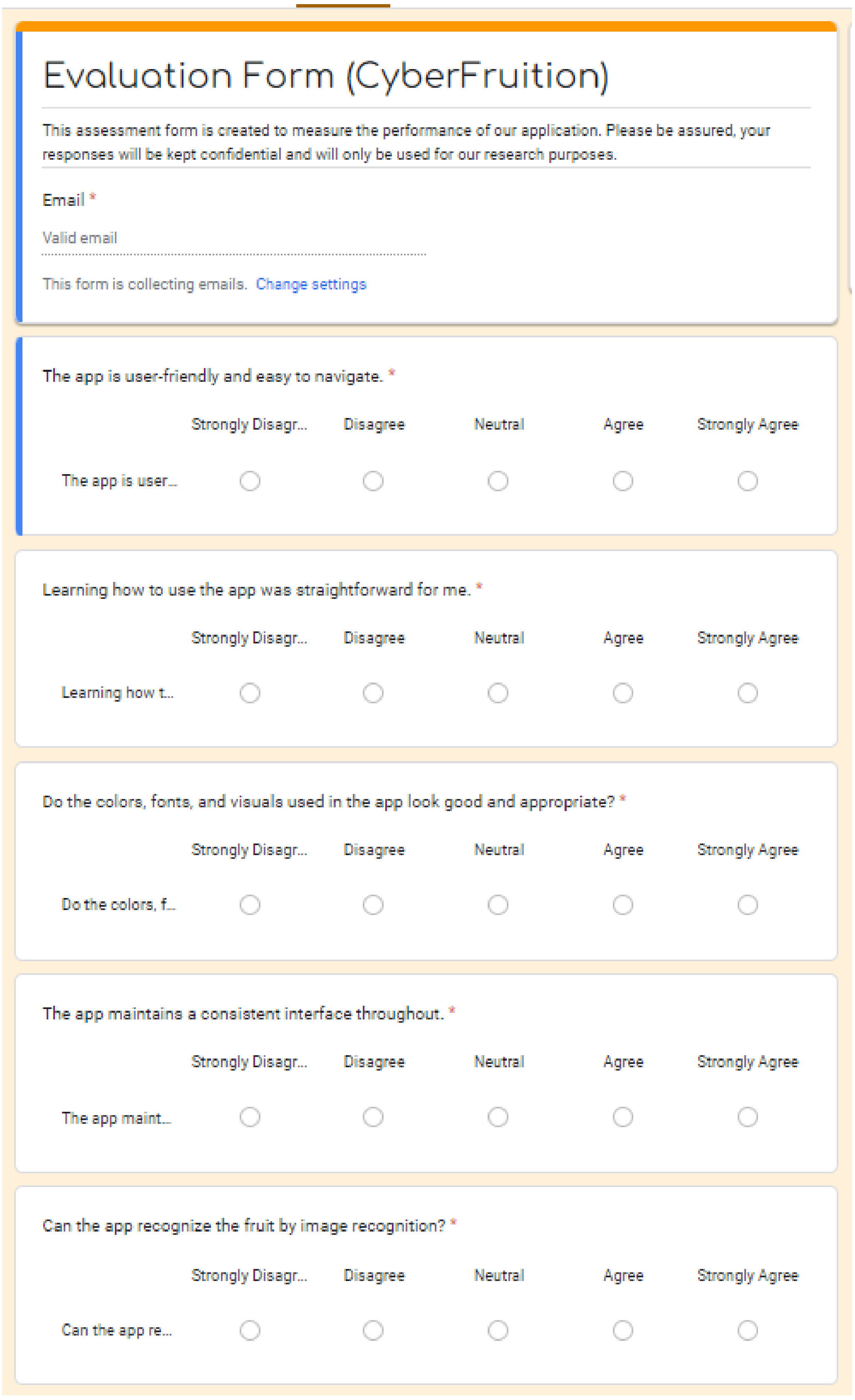
Test Case ID	**Test Scenario**	**Test Case**	**Pre-Condition**	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Status (Pass/ Fail)**
FC_001	Show berries fruit classification	Press berries fruit classification button	Directs to berries fruit classification screen	Berries Fruit classification button	Berries Fruit classification button	Successful	Berries Fruit Classification is now shown	Same as expected	PASS
FC_002	Show pits fruit classification	Press pits classification button	Directs to pits fruit classification screen	Pits Fruit classification button	Pits Fruit classification button	Successful	Pits Fruit Classification is now shown	Same as expected	PASS
FC_003	Show cores fruit classification	Press cores fruit classification button	Directs to cores fruit classification screen	Cores Fruit classification button	Cores Fruit classification button	Successful	Cores Fruit Classification is now shown	Same as expected	PASS
FC_004	Show citrus fruit classification	Press citrus fruit classification button	Directs to fruit classification screen	Citrus Fruit classification button	Citrus Fruit classification button	Successful	Fruit Classification is now shown	Same as expected	PASS
FC_005	Show melons fruit classification	Press melons fruit classification button	Directs to melons fruit classification screen	Melons Fruit classification button	Melons Fruit classification button	Successful	Fruit Classification is now shown	Same as expected	PASS
FC_005	Show tropical fruit classification	Press tropical fruit classification button	Directs to tropical fruit classification screen	Tropical Fruit classification button	Tropical Fruit classification button	Successful	Fruit Classification is now shown	Same as expected	PASS

Project Name:	CyberFruition: AI-Based Fruit Classification and Recognition Framework
Module Name:	Capture and Input Fruits
Reference Document:	Functional Requirement
Created By:	Tester
Date of Creation:	
Date of Review/Execution:	

Test Case ID	**Test Scenario**	**Test Case**	**Pre-Condition**	**Test Steps**	**Test Data**	**Expected Result**	**Post Condition**	**Actual Result**	**Status (Pass/Fail)**
CI_001	Capture fruit	Press Take Photo button	Directs to camera screen	Camera button	Camera button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS
CI_002	Capture two fruits	Press Take Photo button	Directs to camera screen	Camera button	Camera button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS
CI_003	Upload fruit image	Press Upload button	Directs to gallery screen	Upload button	Upload button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS
CI_004	Upload fruit image	Press Upload button	Directs to gallery screen	Upload button	Upload button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS
CI_005	Capture two fruits	Press Take Photo button	Directs to camera screen	Camera button	Camera button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS
CI_06	Capture two fruits	Press Take Photo button	Directs to camera screen	Camera button	Camera button	Incorrect Result	Stays on screen	Same as expected	FAILED
CI_07	Upload fruit image	Press Upload button	Directs to gallery screen	Upload button	Upload button	Incorrect Result	Stays on screen	Same as expected	FAILED
CI_08	Capture two fruits	Press Upload	Directs to	Upload button	Upload button	Incorrect Result	Stays on screen	Same as	FAILED

		button	gallery screen					expected	
CI_09	Capture fruit	Press Take Photo button	Directs to camera screen	Camera button	Camera button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS
CI_10	Capture two fruits	Press Take Photo button	Directs to camera screen	Camera button	Camera button	Incorrect Result	Stays on screen	Same as expected	FAILED
CI_11	Upload fruit image	Press Upload button	Directs to gallery screen	Upload button	Upload button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS
CI_12	Upload fruit image	Press Upload button	Directs to gallery screen	Upload button	Upload button	Successfully Generated	Fruit Result is shown.	Same as expected	PASS

APPENDIX D
EVALUATION FORM



Evaluation Form (CyberFruition)

This assessment form is created to measure the performance of our application. Please be assured, your responses will be kept confidential and will only be used for our research purposes.

Email *

Valid email

This form is collecting emails. Change settings

The app is user-friendly and easy to navigate. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
The app is user...	○	○	○	○	○

Learning how to use the app was straightforward for me. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Learning how t...	○	○	○	○	○

Do the colors, fonts, and visuals used in the app look good and appropriate? *

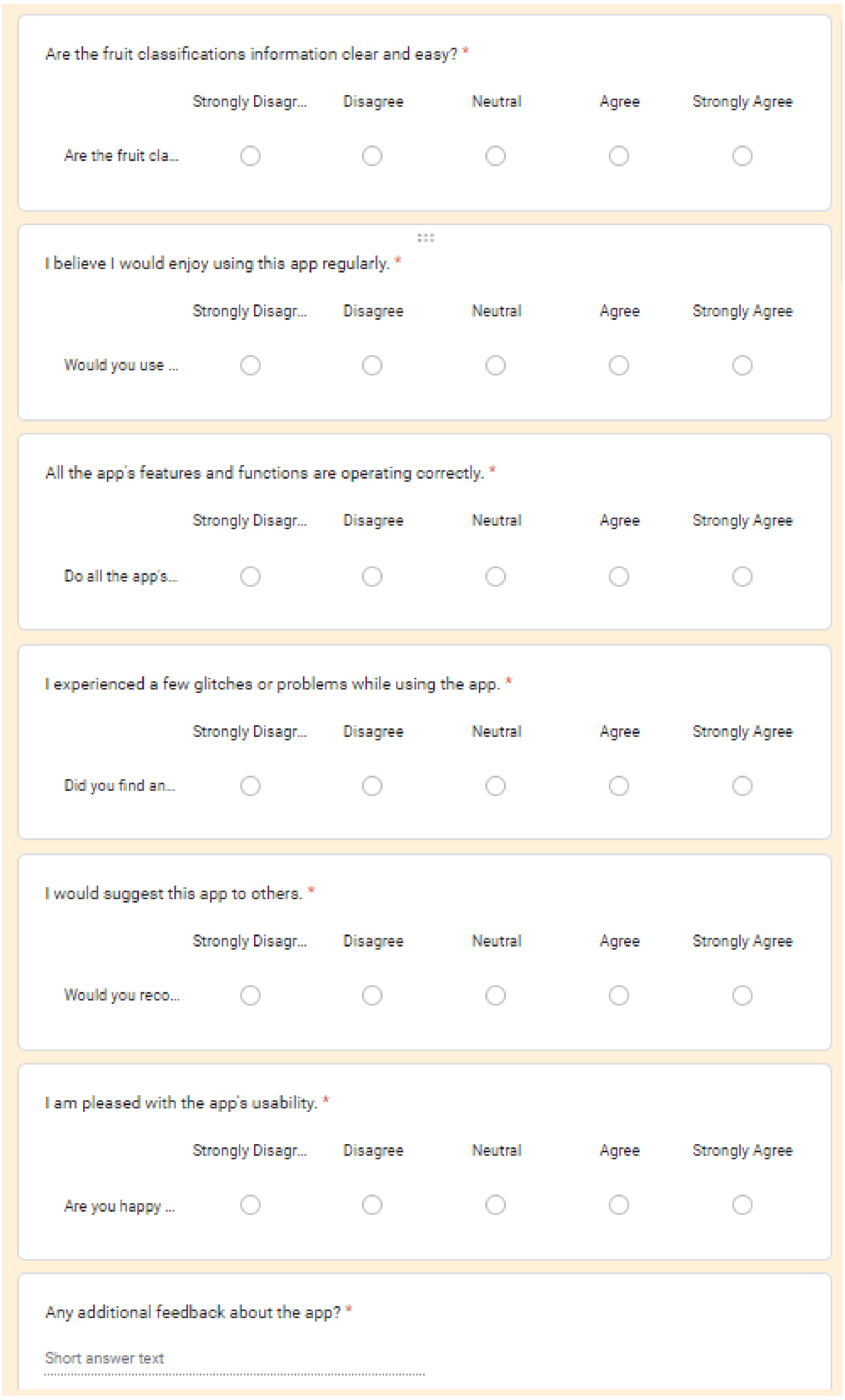
	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Do the colors, f...	○	○	○	○	○

The app maintains a consistent interface throughout. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
The app maint...	○	○	○	○	○

Can the app recognize the fruit by image recognition? *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Can the app re...	○	○	○	○	○


Are the fruit classifications information clear and easy? *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Are the fruit cla...	○	○	○	○	○

I believe I would enjoy using this app regularly. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Would you use ...	○	○	○	○	○

All the app's features and functions are operating correctly. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Do all the app's...	○	○	○	○	○

I experienced a few glitches or problems while using the app. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Did you find an...	○	○	○	○	○

I would suggest this app to others. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Would you reco...	○	○	○	○	○

I am pleased with the app's usability. *

	Strongly Disagr...	Disagree	Neutral	Agree	Strongly Agree
Are you happy ...	○	○	○	○	○

Any additional feedback about the app? *

Short answer text

APPENDIX E
GANTT CHART

Gantt chart

Task ID	Task Description	Task Duration	Start Date	End Date
1	Capstone Project Title Submissions (Individual)	**6**	May 26, 2023	May 31, 2023
2	Initial Title Defense	**1**	July 5, 2023	July 5, 2023
3	Chapter 1-4 revisions	12	July 6, 2023	July 17, 2023
4	Submission of Files (approved title)	1	July 20, 2023	July 20, 2023
5	Distribution of Tasks (Documentation, Front End, Back End)	1	August 9, 2023	August 9, 2023
6	Create project prototype	3	August 29, 2023	August 31, 2023
7	Finalization of Design GUI	7	September 13, 2023	September 19, 2023
8	Gathering Materials for Data sets	1	September 29, 2023	September 29, 2023
9	Training of Models (Data set)	13	October 5, 2023	October 17, 2023
10	Home Page Module	6	October 21, 2023	October 26, 2023
11	Camera Feature Module	11	October 27, 2023	November 6, 2023

12	Fruit Classifications Module	5	November 9, 2023	November 13, 2023
13	Finalization of Main Functions	4	November 14, 2023	November 17, 2023
14	Test Planning	1	November 18, 2023	November 18, 2023
15	Unit Testing	1	November 19,2023	November 19, 2023
16	System Testing	1	November 20, 2023	November 20, 2023
17	Finalization Documentation for Finals			
18	Deploy System for Finals			
19	Final Defense			
20	Rebuilding Some Modules			
21	Implementation the Modifications			
22	Deploy the Project			
23	Confirm the Documentation and Deliverables are up to date			
24	Release Sources			

APPENDIX F
USE CASE

Use Case #1	**Fruit Classifications**
Description	Develop a module that allow users to select different fruit classifications.
Actors	Users
Goal	Be able to navigate the different fruit classifications
Pre-Condition	It will direct to a certain fruit classification
Post-Condition	Information of the fruit classification chosen is shown
Basic Flow	Users must select a fruit classification and it will be directed to the fruit classification.
Alternate Path	Users will remain on the dashboard screen if they stay idle,
Use Case #2	**Camera**
Description	Develop a module that allow users to take photo or upload a photo then showing the result afterwards.
Actors	Users
Goal	Can be able to identify or recognize the fruit
Pre-Condition	Need to capture an image or input an image
Post-Condition	The detected or recognized fruit is shown
Basic Flow	Users will proceed to the camera option then will capture or upload an image. After capturing, the app will display the result.
Alternate Path	Users will still proceed to result page. An ‘unidentified fruit’ will be shown if no fruits are detected.

CURRICULUM VITAE

JOSE KAIRAN MEMORACION

Bachelor of Science in Information Technology

CONTACT

kairanmemoracion79@gmail.com

Tugbungan, Purok 7,
Zamboanga City,
Philippines
7000

PERSONAL PROFILE

Birthday:	April 5, 1999
Place of birth:	Zamboanga City
Age:	24
Sex:	Male
Civil status:	Single
Nationality:	Filipino
Religion:	Roman Catholic

EDUCATIONAL BACKGROUND

2019-2024

Bachelor of Science in Informa Technology

Western Mindanao State University

Normal Rd. Baliwasan

2016-2018

Philippine Christian University Of Manila

2012-2016

Immaculate Conception Archdiocesan School

2006-2012

Tugbungan Elementary School

AMEERANUR SARAEL

Bachelor of Science in Information Technology

CONTACT

asarael24@gmail.com

Sinunuc, Purok 10,
Zamboanga City,
Philippines
7000

PERSONAL PROFILE

Birthday:	February 24, 2002
Place of birth:	Zamboanga City
Age:	21
Sex:	Female
Civil status:	Single
Nationality:	Filipino
Religion:	IslM

EDUCATIONAL BACKGROUND

2020-2024
Bachelor of Science in Informa Technology
Western Mindanao State University
Normal Rd. Baliwasan

2018-2020
Baliwasan Senior High School - Stand Alone
Atty. Baban Drive, San Jose Rd.

2017-2018
Basilan National High School
Isabela City, Basilan

2007-2017
International Philippine School in Riyadh
Olaya Street, Haymalik Fahad Rd. Riyadh, KSA

AHMAD KHALIQ TUTTUH

Bachelor of Science in Information Technology

CONTACT

Theamh0021@gmail.com

Pagkasilasa Drive,
Talon-Talon,
Zamboanga City, Philippines
7000

PERSONAL PROFILE

Birthday:	November 20, 1999
Place of Birth:	Zamboanga city
Age:	24
Sex:	Male
Civil Status:	Single
Nationality:	Filipino
Religion:	Islam

EDUCATIONAL BACKGROUND

2018-2024

Bachelor of Science in Informa Technology

Western Mindanao State University

Normal Rd. Baliwasan

2016-2018

Pilar College Zamboanga City

R.T. Lim Boulevard

2012-2016

Pilar College Zamboanga City

R.T. Lim Boulevard

2006-2012

Western Mindanao State University

Normal Rd. Baliwasan