LivingGreen: A Botanical Identification Mobile Application with Barter System

---

A Thesis Project Presented to the Faculty of

the School of Computer Studies

University of San Jose - Recoletos

Cebu City, Philippines

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Information Technology

By

Rasmil Lloyd P. Augusto

Ronel John S. Tano

Mervyn B. Morales

Christian D. Lastimosa

ENDORSEMENT

This project entitled **"LivingGreen: A Botanical Identification Mobile Application with Barter System"** prepared by AUGUSTO, RASMIL LLOYD P., TANO, RONEL JOHN S., MORALES, MERVYN B., LASTIMOSA CHRISTIAN D. in partial fulfillment of the requirements for the Degree of BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY MAJOR IN MOBILE APPLICATION DEVELOPMENT has been endorsed and is recommended for the acceptance and approval for ORAL EXAMINATION.

MR. JOHN LEEROY A. GADIANE, MCS
Adviser Coordinator

ENGR. CARMEL M. TEJANA, MManE, MSIT
Member

ENGR. VINCENT PATALITA III
Coordinator

MR. RODERICK A. BANDALAN, MSIT
Member Chairperson, CS/IT Department

DR. JOVELYN C. CUIZON, DMHRM
Dean

APPROVAL

Approved by the Tribune for Oral Examination with the grade of **PASSED**.

MR. JOHN LEEROY A. GADIANE, MCS
Adviser Coordinator

ENGR. CARMEL M. TEJANA, MManE, MSIT
Member

ENGR. VINCENT PATALITA III
Coordinator

MR. RODERICK A. BANDALAN, MSIT
Member Chairperson, CS/IT Department

DR. JOVELYN C. CUIZON, DMHRM
Dean

ACKNOWLEDGEMENT

First and foremost, the researchers would like to thank God for giving us wisdom and good health to finish this study and guiding us in our everyday journey. To our friends

and families, despite the hardships and obstacles, our friends and family helped us; the researchers reached our objectives and finished our project through the moral, spiritual, and financial support that they have given us. Our study will not be possible without our advisers, professors, and capstone coordinators.

Special Mentions:

We want to thank Mr. Roderick Bandalan for offering her valuable time to help us with our study. We would also like to thank Mr. Vicente Patalita III for helping us with the technical aspect of our project. And lastly, to our adviser, Mr. John Leeroy Gadiane, who has been so patient with us, for giving us technical support for our study and motivating us whenever we feel like giving up. This study would not have been possible without our dearest adviser and capstone coordinator. Furthermore, we would like to extend our gratitude to the people who have been part of this study.

ABSTRACT

Plants are categorized into smaller groups according to their shared characteristics, which can be daunting given their complexity. While experts can quickly recognize familiar plants, identifying potentially harmful or toxic ones, particularly in medicine, can be challenging. Botanists possess the expertise to distinguish such plants, but with millions of species featuring similar parts (roots, stems, leaves), they must devise a system to classify them effectively. Living Green aims to expound botanical research through a Mobile Botanical Identifier mobile application for finding an unknown plant's captured or uploaded photo with a barter system. It is a mobile application that connects users with plant enthusiasts and plant experts to aid in identifying a plant name.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Rationale of the Study

Our world is primarily made up of vegetation, so it is called a "green planet." In addition to the relationship between plants and our "food," "medicines," and "furniture," it is essential to understand why we can conceive a world without the oxygen that plants supply. Because of this, we can argue that plants are the foundation of life. Plants are classified into smaller groups based on the characteristics that they share. Recognition of plants might be challenging since they are so complex. The process of recognizing familiar plants was simple for experts, but sometimes, particularly in medicine, we need to identify prejudiced or toxic plants. Botanists can quickly identify these plants, but they must find a way to classify the many different species since millions of other plant species are composed of similar parts (roots, stems, leaves).

Designing a plant recognition system is necessary to save time and money. Numerous studies have focused on leaves to identify plants because, in comparison to other plant parts, leaves are the most crucial in conveying a plant's characteristics. Fruits and flowers are also not always present because most plants are seasonal, and their size, shape, and color change as they develop. Numerous studies have employed leaves to categorize different plant types based on their shape, texture, venation, and color. Chemical approaches, instrumental methods, and another method called the optical method—more advantageous than other techniques—were all used in identifying leaves.

One of their main advantages is the complex composition of botanicals, which comprises groups of related compounds with overlapping activities that interact to produce a higher overall activity.

In a barter system, customers come together in a marketplace to trade goods or services for one another without exchanging money. In barter markets, manufactured items or supplied services are used to offset the cost of purchased goods or services. By utilizing the capabilities of computers and the Internet, electronic barter (also known as e-barter) systems offer a more reliable and beneficial exchange of supplies. Contrary to the conventional definition of e-commerce, transactions in e-barter systems only sometimes involve the exchange of money, even though they appear to be similar to well-known e-commerce systems.

In the midst of the pandemic, people felt isolated and suddenly cut off in the busy errands of the world. In fact, The review data, gathered and examined by researchers at Texas A&M University and published in the Journal of Environmental Horticulture, supports the idea that residing in or near green spaces and spending as much time as possible in both natural settings and cultivated gardens can enhance mood, lessen the adverse effects of stress, encourage physical activity and other positive behaviors, improve cognition, lessen aggression, and enhance memory. Hall and Knuth (2019). After all, it is not just to dissolve boredom or to suffice the need for mental health; this kind of hobby also offers a gateway for business opportunities. There are already 2.6 billion monthly active users on Facebook, which translates into potential customers for Facebook Marketplace. Additionally, more users mean more vendors and shopping categories to select from. Through this social media platform, plant businesses have been

thriving. With use of this mobile application, it binds the hobby with marketing at once for ease of access.

This paper aims to develop LivingGreen, a Mobile Botanical Identifier mobile application for finding the uploaded photo of an unknown plant with a barter system. It is a mobile application that connects users with plant enthusiasts and plant experts to aid in identifying a plant/s name.

REVIEW OF RELATED WORKS

Google develops Google Lens for image recognition. It is best described as a real-world search engine. It uses artificial intelligence to identify or determine text and objects in pictures and a live view from the user's camera on the phone. It then allows the user to learn about and interact with those elements in various interesting ways. Google Lens will also recognize shops, eating places, bars, snack bars, cafes, and clubs, too, and will present to the user with a pop-up window that will show the reviews and comments, address details, and opening hours. What strikes them as astounding is the capacity and capability to identify commonplace objects. Impressive is the capacity and capability to identify commonplace objects. If the user aims at the program at hand, it will identify it and propose a thumbs-up emoji, which is amusing and exciting, but if the user points it at a drink, it will attempt to identify and determine what it is. But while Lens's ability to speak, recognize a flower, search or look for a book, or provide the user with information on a landmark is certainly impressive, these are the most mundane productivity abilities of the system that are much more likely to find a place in the user's everyday life.

TapTapSee is a mobile camera application that is designed and developed specifically for blind and visually impaired users. Its purpose is to help the blind and visually impaired identify objects or materials they encounter daily. Powered by the CloudSight Image Recognition API, TapTapSee uses the mobile camera of the user's device and the VoiceOver functionality or feature to take a photograph or video of something and identify it out loud for the user. The application will indicate to not only the user that the user just took a photograph of a water bottle, for example but also the color of the water bottle and the brand of the water bottle as well.

LogoGrab is developed by Visual Technologies. The application's main feature is the identification or recognition of objects, logos, text, and commercial materials. However, the most outstanding feature of LogoGrab is the scene detection system feature. Although this photo identifier application offers limited usability, it can efficiently and accurately identify the specific companies operating in the market. This is why LogoGrab is considered the best tool for finding even the most hidden logos, photographs, and commercial or business stuff. In addition, the application's main purpose or core idea is to identify or recognize and give marketers information about the brand's assets. Furthermore, the innovative application of image identification technology in LogoGrab is praiseworthy.

Cam Find is an application that can uniquely identify objects by picture for the user. The application's most outstanding feature or functionality is the visual search engine through which the user can search the physical or real world. The simple interface of this image recognition application allows photography of an object. The visual search engine

behind the application will precisely tell the user what the object is. The application produces or generates multiple outcomes in the form of videos, images, and web content to help the user figure out what the user is looking for. These results also increase the knowledge of the user.

Plantix is a plant application for growing plants. The application offers information about growing various plants and other crops. The application also helps diagnose problems, including various diseases and other potential problems of the plants. Plantix has a global reach with many people helping with regional plant information and even local in some cases. Thus, it somewhat also serves as a sort of social media or forum space for farmers and growers as well. There are no ads or in-app purchases in the actual application, which is free to download.

OBJECTIVES OF THE STUDY

This study aims to create and develop a mobile application connecting botanical experts and plant enthusiasts. This application is equipped with an image recognition function that will assist plant enthusiasts in identifying plants that have yet to be discovered.

Aim of Researchers:

- To encourage botanical enthusiasts to contribute to botanical research.
- Provide a communicative and informative platform to link plant experts and enthusiasts.
- To integrate APIs for data in the backend, such as geolocation and plant identification data.

- Provide a plant identifier to identify plants quickly
- To integrate a forum feature if a plant is not recognized or identified by the plant identifier
- To integrate a barter feature for users to exchange their plant(s) with another user without exchanging money.

PROJECT SCOPE AND LIMITATION

The application requires the users to have a good internet connection and location services turned on. The application will only support Android devices running on Android 5.0 (Lollipop) or higher, which provides image recognition capabilities. Using a more recent version of Android may provide better results. Researchers utilize the plant.id API for the database of plants and plant identification to gather data and perform certain tasks. However, it is important to take note that we do not have the ability to train the API ourselves. This means that we relied on the API's pre-trained model and could not modify it to suit our specific needs. Despite this limitation, we could effectively leverage the API's functionality to achieve our project goals.

CHAPTER II

SOFTWARE REQUIREMENTS AND DESIGN SPECIFICATION

The researchers elaborate on the functional and non-functional needs for the "LivingGreen" mobile application. Included are the application's overview idea, the use case diagram, the use case narratives matching each use case of the Use Case Diagram (Use Case Diagram), the architectural diagram, the use case narratives, the activity diagram, the class diagram, the user interface, and the design.

APPLICATION OVERVIEW

LivingGreen is a mobile application connecting plant enthusiasts and experts. With the use of the plant identifier feature, this application will help the users or plant enthusiasts identify the plant that is unfamiliar to them. Plants not identified by the plant identifier can be posted in the feed for the plant experts to answer or identify that plant. With the in-app trading mechanism, users have the freedom to acquire plants from other users by setting up a trade request, and with the help of a built-in chat system, the parties can communicate easily.

ARCHITECTURAL DIAGRAM

The architectural diagram shows the application architecture view, which helps the reader identify the applications, database, services, and interactions.

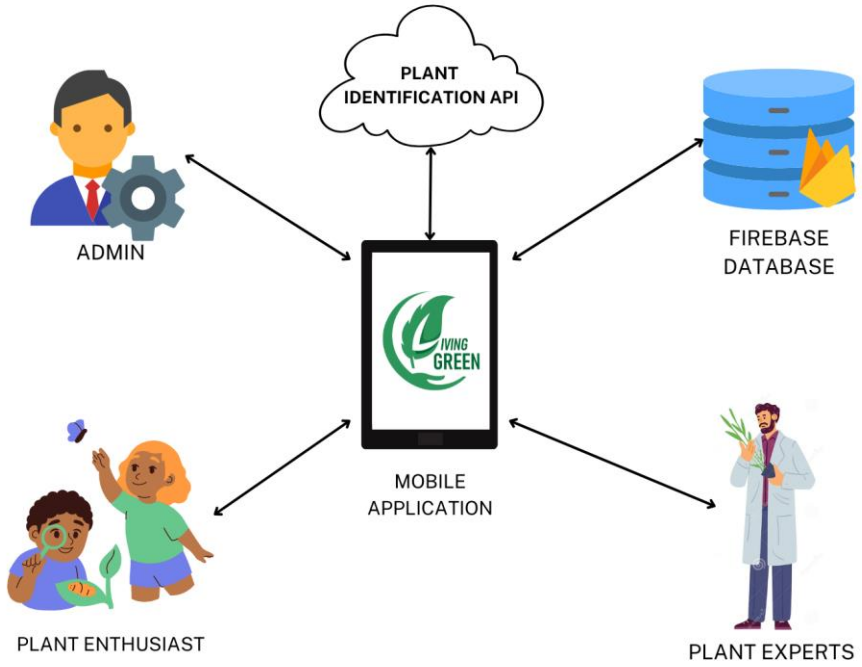


Figure 1: Architectural Diagram

Figure 1 explains the structure of the “Living Green” Application. The structure consists of clients, a mobile application, a plant identification API, and a database. The application has three users, namely the administrator, plant enthusiast, and plant experts. The mobile application handles the user’s requests. A mobile application is also responsible for the communication and connection of the users. The Firebase database contains and stores the data the users need for their system.

USE CASE DIAGRAM

The use case diagram depicts the relationship between actors and the various system use cases in which actors are involved.

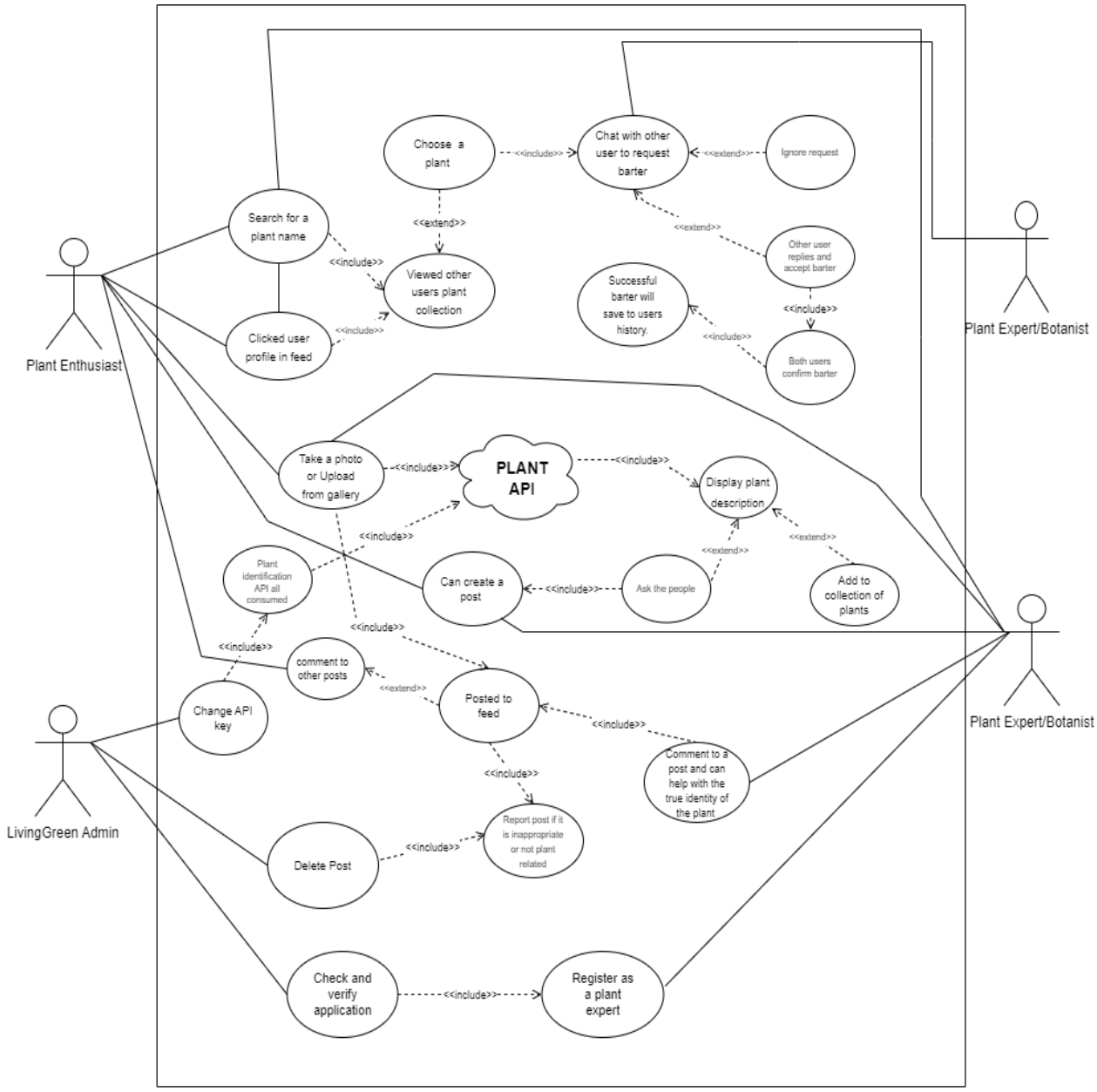


Figure 2: Use Case Diagram

Figure *2* depicts the functionalities of the application. The administrator can change the API key used by the application if the allotted number of identifications are all consumed, can verify plant expert registration, and can delete reported posts. The plant

enthusiasts/users can take a photo or upload from a device for plant identification, can post asking questions related to plants, can report posts, can request barter, can chat with others if someone requests barter, can save the identified plants to their plant collection, can comment to other posts and user can report other posts if it is inappropriate or not plant related. Plant enthusiasts can request barter from another plant enthusiast. Plant experts must register as plant experts/plant botanists, and the Admin will verify it. Plant experts can help plant enthusiasts identify plants in the posts through comments.

USE CASE NARRATIVES

The Use Case Narratives of LivingGreen show the specific flow mobile to achieve a specific goal. It explains how each uses case functions on the whole application.

Use Case Narrative: UC001 PLANT CAPTURE

Use Case: UC001	Plant Capture
Actors:	Plant Enthusiasts and Plant Experts
Purpose:	This is used to capture a plant for identification.
Overview:	The actor will take a photo or upload a photo.
Type:	Required
Pre-condition:	The actor has created an account.
Post-condition:	The actor will be able to display the taken or chosen picture of the actor.
Flow of Events	
Actor Action	**Systems Response**

1. The actor will sign-up.	2. The system will bring the actor to the home page.
3. The actor will go to the camera tab.	4. The system will bring the actor to the camera tab, where the actor can take a photo or upload a photo of a plant.
5. The actor will take a photo or upload a photo of a plant.	6. The system will display the taken or chosen picture of the actor on the image preview page.

Table 1: Use Case Narrative for Plant Capture

Use Case Narrative: UC002 PLANT IDENTIFICATION

Use Case: UC002	Plant Identification
Actors:	Plant Enthusiasts and Plant Experts
Purpose:	This is used to identify a plant.
Overview:	The system will identify the plant.
Type:	Required
Pre-condition:	The actor will take a photo or upload a photo.
Post-condition:	The actor will be able to display the identification result.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap the “Identify” button	2. The system will display suggestions of what is the plant’s identity.
3. The actor will tap the “Add to collection” button	4. The system will put the image of the identified or unidentified plant in the plant inventory of the user.
5. The actor will tap the “Ask the people” button	6. The system will redirect the actor to the home page to create a post with the potential name of the plant and the picture.

Table 2: Use Case Narrative for Plant Identification

Use Case Narrative: UC003 ASK OTHER PLANT ENTHUSIASTS AND PLANT EXPERTS FOR HELP

Use Case: UC003	Ask Other Plant Enthusiasts and Plant Experts for Help
Actors:	Plant Enthusiasts and Plant Experts
Purpose:	This is used to identify clearly the plants that are not identified well in the plant identifier.
Overview:	The system will bring the actor to a page that allows the actor to seek help or ask questions if a plant is not identified well.
Type:	Required
Pre-condition:	The actor used the plant identifier feature.
Post-condition:	The actor posted to the feed asking for help on what plant the actor has.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap the “Ask the People” button	2. The system will redirect the actor to the “create a post.“
3. The actor will input the title, description and etc. of a post.	4. Flashed on the screen, the post description is inputted by the actor, such as the Title, description, and plant image.
5. The actor will choose an image or choose another image.	6. Returns to the “Create a post” screen displaying the content of the post, such as its title, description, image, and the location of the actor.
9. Click the “Confirm” button.	10. Saves the data to the database.

Alternative Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap the “Ask the people” button.	2. The system will redirect the actor to the “create a post page.“

3. The actor will input the title, description and etc. of a post.	4. Flashed on the screen, the post description is inputted by the actor, such as the Title, description, and plant image.
5. Click the “Confirm” button.	6. Saves the data to the database. It will prompt a message “Post created successfully!”.

Table 3: Use Case Narrative for Ask Other PLant Enthusiasts and Experts for Help

Use Case Narrative: UC004 COMMENT TO A POST

Use Case: UC004	Comment to a Post
Actors:	Plant Enthusiasts and Plant Experts
Purpose:	This is used to help other plant enthusiasts and plant experts identify plants.
Overview:	The actor will comment on a post.
Type:	Required
Pre-condition:	The actor has created an account and logged in.
Post-condition:	Created a comment to a post of other plant enthusiasts and plant experts.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap a post in the feeds.	2. The system will redirect the actor to the page of the post that the actor chose.
3. The actor will input a comment on a post.	4. The text controller will display the imputed texts.
5. The actor will tap the send icon.	6. The database gets data from the comment.

Table 4: Use Case Narrative for Comment to a Post

Use Case Narrative: UC005 CHAT

Use Case: UC005	Chat
Actors:	Plant Enthusiasts and Plant Experts
Purpose:	This is used for the users to communicate with each other.
Overview:	The actor will use chat to communicate with other users.
Type:	Not required
Pre-condition:	If the actor wants to trade a plant from another user, the chat will be available, and press the “Chat” button.
Post-condition:	The actor was able to chat with other users.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will ask for a trade with another user.	2. The system will redirect the actor to the pending trade page, and the chat will be available.
3. The actor will click the “chat” button.	4. The system will redirect the user to a chat page with the chosen use of the actor.
5. The actor will input a chat.	6. The system will create a chat thread.
6. The actor will send the chat to the other user.	7. The system will record it in the database and send the data to the other user’s chat page.

Table 5: Use Case Narrative for Chat

Use Case Narrative: UC006 VIEW OTHER USER’S PLANT INVENTORY

Use Case: UC006	View Other User’s Plant Inventory
Actors:	Plant Enthusiasts and Plant Experts

Purpose:	This is used to help other plant enthusiasts to view their collections of plants and can view other users.
Overview:	The actor will view their collections.
Type:	Not required
Pre-condition:	The actor will go to the news feed to find another user.
Post-condition:	The actor viewed the user's plant inventory.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap the profile picture of the user.	2. The system will redirect the actor to the other user plant collection.
3. The actor can then see available plants from that user and scroll.	4. The UI can scroll up and down.

Alternative Flow of Events	
Actor Action	**Systems Response**
1. The actor will search for a plant name.	2. The system will display the suggested plant with the same or similar name to what the user searched for.
3. The actor will tap a plant image.	4. The system will redirect the actor to have the option to trade a plant.

Table 6: Use Case Narrative for View Other User's Plant Inventory

Use Case Narrative: UC007 VIEW OWN PLANT INVENTORY

Use Case: UC007	View Own Plant Inventory
Actors:	Plant enthusiasts

Purpose:	This is used to help other plant enthusiasts to view their collections of plants.
Overview:	The actor will view their collections.
Type:	Not required
Pre-condition:	The actor will go to the profile tab, and there is a tab that has “my plants.”
Post-condition:	The actor has been able to view their own inventory.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap on the profile tab.	2. The system will redirect the actor to the plant collection of the user.
3.	4. The system will redirect the actor to his/her own plant inventory page and display his/her plants that are being identified.

Table 7: Use Case Narrative for View Own Plant Inventory

Use Case Narrative: UC008 VIEW PROFILE

Use Case: UC008	View Profile
Actors:	Plant enthusiasts and plant experts
Purpose:	This screen will display the necessary information about the user that includes: Location, Name, Role, and a button that will redirect to the user’s collection of plants.
Overview:	Actors or users can view and edit their information.
Type:	Not required

Pre-condition:	The actor will have to register first in order to be enlisted into the users' database.
Post-condition:	The actor has the option to change the picture of their avatar and can edit their basic information on the page.
Flow of Events	
Actor Action	**Systems Response**
1. The actor must press his/her profile on the bottom navigation bar.	2. The system will transfer the view to the profile page.
3. Users can edit their information by going to “Account Information.”	4. It will receive the new information about the user and save it to the database. The edited field will appear immediately after the submission of the form.

Table 8: Use Case Narrative for View Profile

Use Case Narrative: UC009 BARTER A PLAN

Use Case: UC009	Barter a Plant
Actors:	Plant enthusiasts and plant experts
Purpose:	This screen will accommodate plant enthusiasts for them to be able to barter plants.
Overview:	Actors can view trade requests from other users.
Type:	Not required
Pre-condition:	Actors must have a plant saved in his/her plant collection in order for the other user to inspect the plant and request barter.
Post-condition:	The actor has the option to ignore the barter request.
Flow of Events	
Actor Action	**Systems Response**

1. Actors will browse plants from another plant enthusiast and request a trade for that plant selected.	2. The system will generate a pending trade, request a barter, and send it to the target actor.
3. The target actor will have the option to accept or ignore the request.	4. The chat button will be available.
5. The target actor will choose a picture of the plant that they want to barter to the barter request.	6. The system will save the conversation between actors.

Table 9: Use Case Narrative for Bartering Plant

Use Case Narrative: UC0010 SEARCH FOR A USER OR PLANT

Use Case: UC0010	Search for a User or Plant
Actors:	Plant Enthusiasts and Plant Experts
Purpose:	This is used to search for a user or plant.
Overview:	The actor will search for a user or plant on the search bar.
Type:	Not required
Pre-condition:	There are existing names and plants of other users in the system.
Post-condition:	The actor can view the search results.
Flow of Events	
Actor Action	**Systems Response**
1. The actor clicks on the search bar.	2. The system will take the keyword to the search function and filter the collection of plants.
3. The actor inputs a name of a plant.	4. The system will display the search results.

Table 10: Use Case Narrative for Search for a User or Plant

Use Case Narrative: UC0011 VERIFICATION

Use Case: UC0011	Verification
Actors:	Admin

Purpose:	This is used to verify plant experts.
Overview:	The actor will view, verify or reject a user's request to be a plant expert.
Type:	Required
Pre-condition:	A user should have a request to be a plant expert.
Post-condition:	A normal user will now be a plant expert.
Flow of Events	
Actor Action	**Systems Response**
1. The actor clicks on the "experts request" tab.	2. The system will navigate the actor to a page where all the requests are displayed.
3. The actor will tap the "Accept proof" button.	4. That particular user who sent the request will now be a plant expert in the application.
5. The actor will tap the "Reject proof" button.	6. The request will be removed from the expert's request.
7. The actor will tap the "Display" button.	8. The proof will be displayed.
9. The actor will tap the "Download" button.	10. The proof will be downloaded to the device.

Table 11: Use Case Narrative for Verification

Use Case Narrative: UC0012 DELETE REPORTED POST

Use Case: UC0012	Delete Reported Post
Actors:	Admin
Purpose:	This is used to delete the reported posts.
Overview:	The actor can delete a post that is reported by a user or plant expert that is misleading or not related to plants.
Type:	Not required
Pre-condition:	A user should have a reported post for the admin to review and will decide to keep or delete the post.

Post-condition:	The post will be deleted from the database.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap the “Reported post.” tab	2. The system will display all the reported posts.
3. The actor will tap the “Delete” button.	4. The post will be deleted from the database.

Table 12: Use Case Narrative for Delete Reported Post

Use Case Narrative: UC0013 CHANGE API KEY

Use Case: UC0013	Change API Key
Actors:	Admin
Purpose:	This is used to change the plant identification API key.
Overview:	The actor will change the API Key by inputting a new API key.
Type:	Not required
Pre-condition:	The actor has logged in as an admin.
Post-condition:	A new API key will be used in the system.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap on the “Change API” tab.	2. The system will redirect the actor to the change API page.
3. The actor will input a new API key and then hit “Change.”	4. The plant identification system will use a new API key.

Table 13: Use Case Narrative for Change API Key

Use Case Narrative: UC0014 VIEW ALL USERS

Use Case: UC0014	View All Users
Actors:	Admin
Purpose:	This is used to view all users registered in the application.
Overview:	The actor will view all the registered users.
Type:	Not required
Pre-condition:	The actor has logged in as an admin.
Post-condition:	The actor can view all the registered users.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will tap the “All users” tab.	2. The system will redirect the actor to a page where all users will be displayed.

Table 14: Use Case Narrative for View All Users

Use Case Narrative: UC0015 COMMENT VOTING

Use Case: UC0015	Comment Voting
Actors:	Plant enthusiasts and plant experts
Purpose:	This is used to upvote or downvote a comment.
Overview:	If the actor finds a comment that is helpful, the actor will upvote a comment, and if an actor sees a comment that is misleading or irrelevant to the post, the actor will downvote that comment.
Type:	Not required
Pre-condition:	The actor has logged in to the system.

Post-condition:	The actor has voted a comment.
Flow of Events	
Actor Action	**Systems Response**
1. The actor will click the “arrow up” button in the comment.	2. The system will add one vote to a comment.
3. The actor will click the “arrow down” button in the comment.	4. The system will subtract one vote from a comment.

Table 15: Use Case Narrative for Comment Voting

ACTIVITY DIAGRAM

The activity diagram presents the series of actions and flows in the application’s features.

Upload or take a picture of a plant.

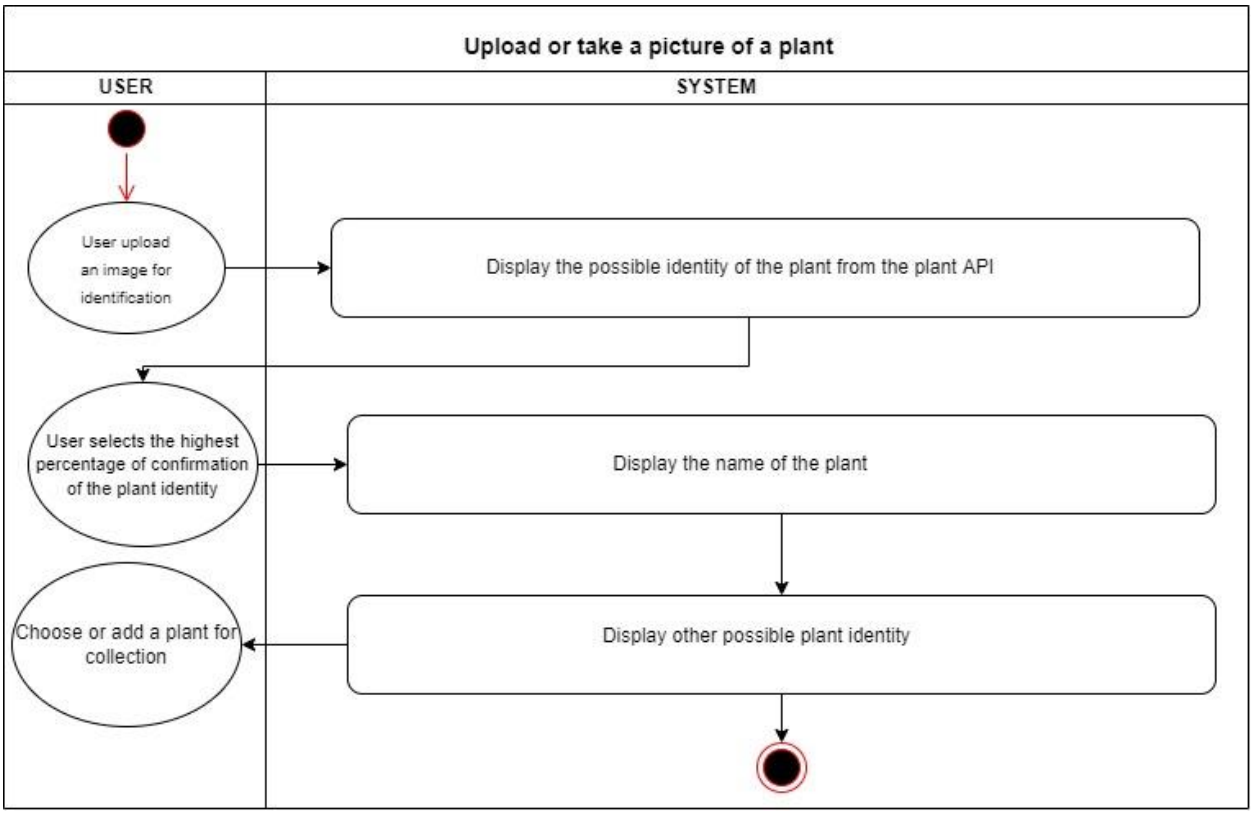


Figure 3.1 Activity Diagram for Search Rental Property

Figure 3.1 demonstrates that when a user uploads an image of a plant for plant identification, the application will then display all the possible identities of the plant that is being uploaded. Then the application will rank the result of the possible identity of the plant based on its percentage of confirmation or possibility.

Chat with another User.

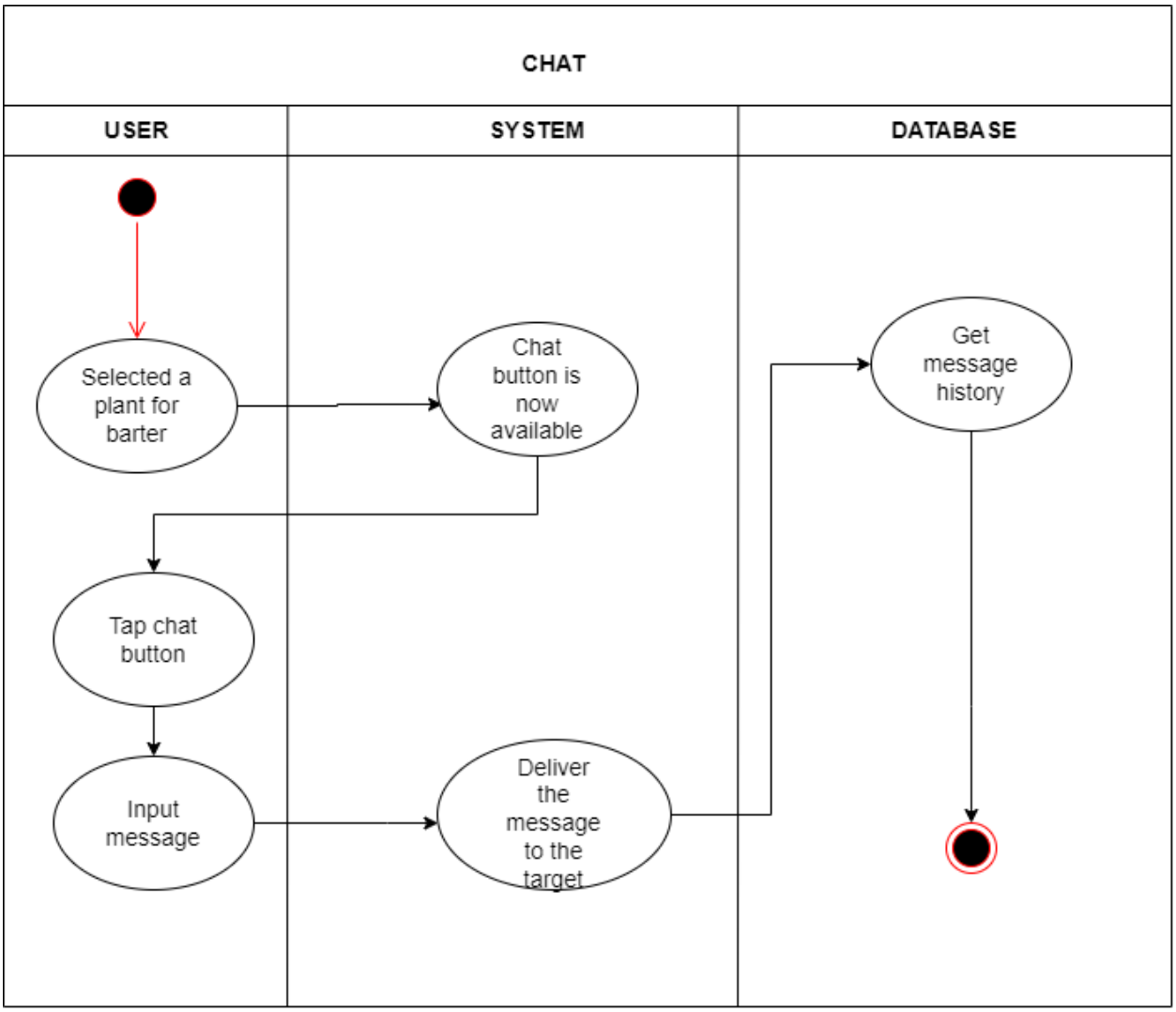


Figure 3.2 Activity Diagram for Chat with another user

Figure 3.2 demonstrates how the application allows plant enthusiasts and other plant enthusiasts to communicate and talk about bartering their plants.

Update Account Information

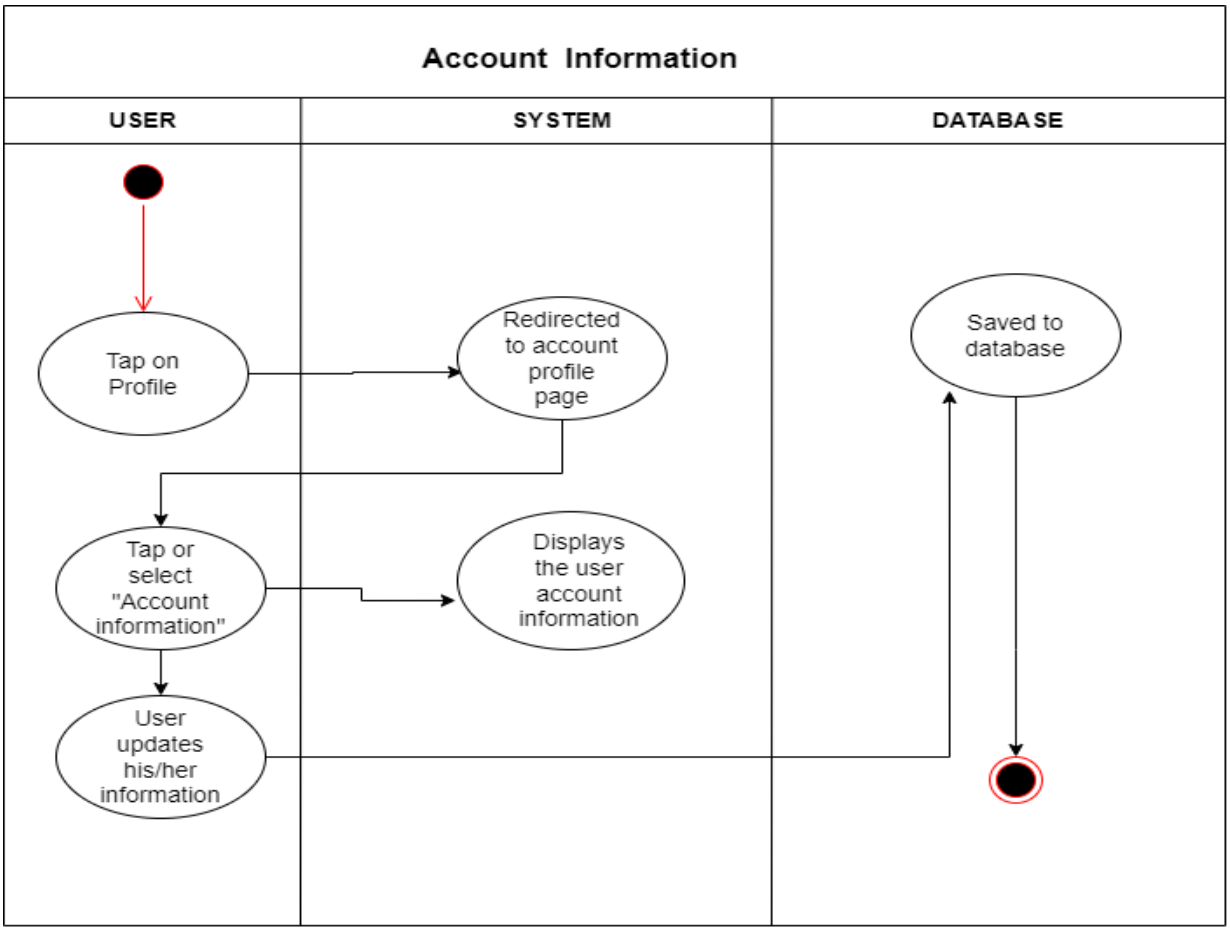


Figure 3.3 Activity diagram for Update Profile

Figure 3.3 allows the user to update their profile or their account information. When the user selects “Profile” from the bottom navigation bar, then “Account Information ", the system will display their personal information, and the user’s account information can be edited or updated.

Add a Post

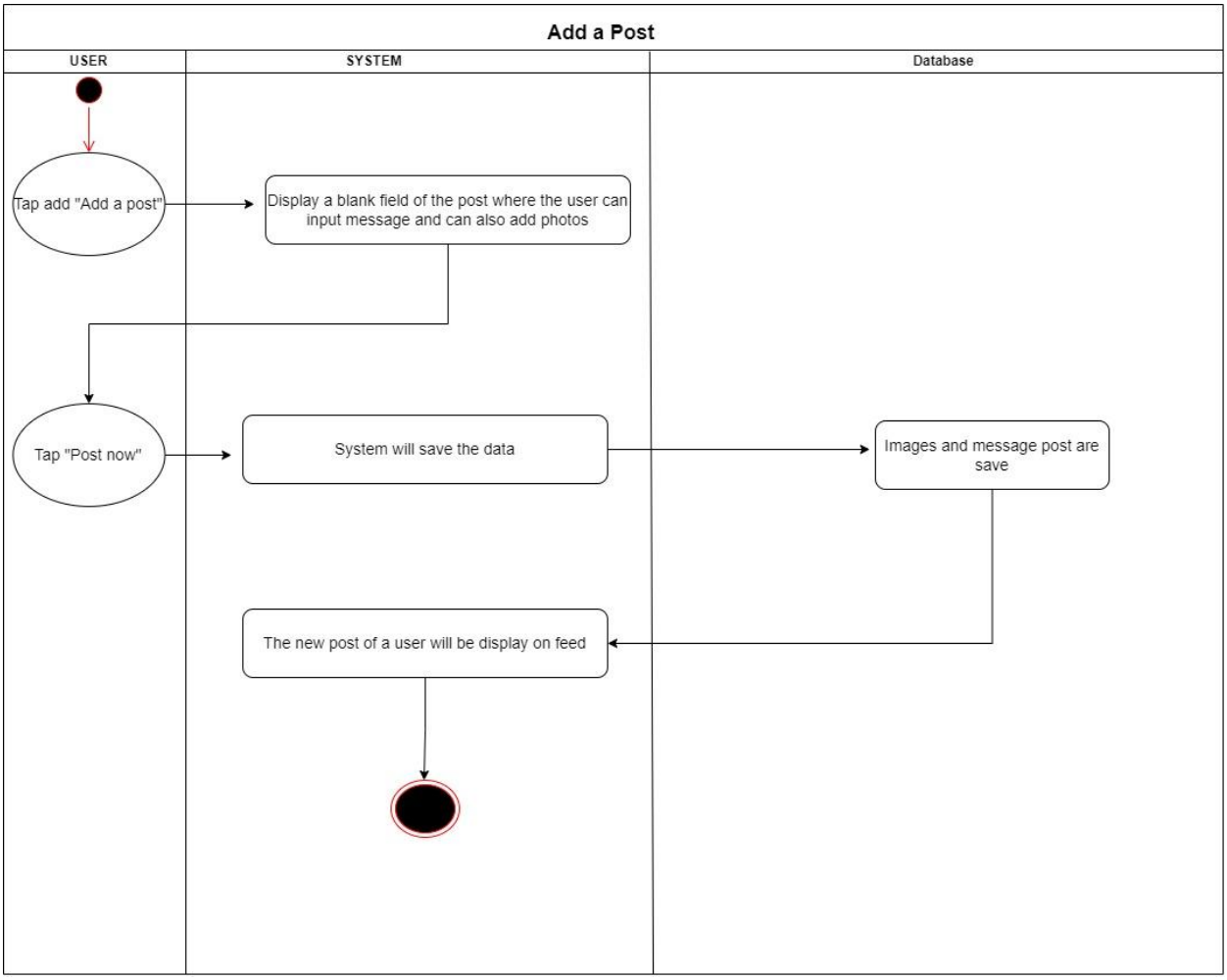


Figure 3.4 Activity Diagram for Add a Post

Figure 3.4 depicts how a plant enthusiast's posts help identify the plant. Posting on the feed happens when the database does not find and identify the plant. When the user writes a post, the user must add a picture of the plant and a message.

User Search for a Barter

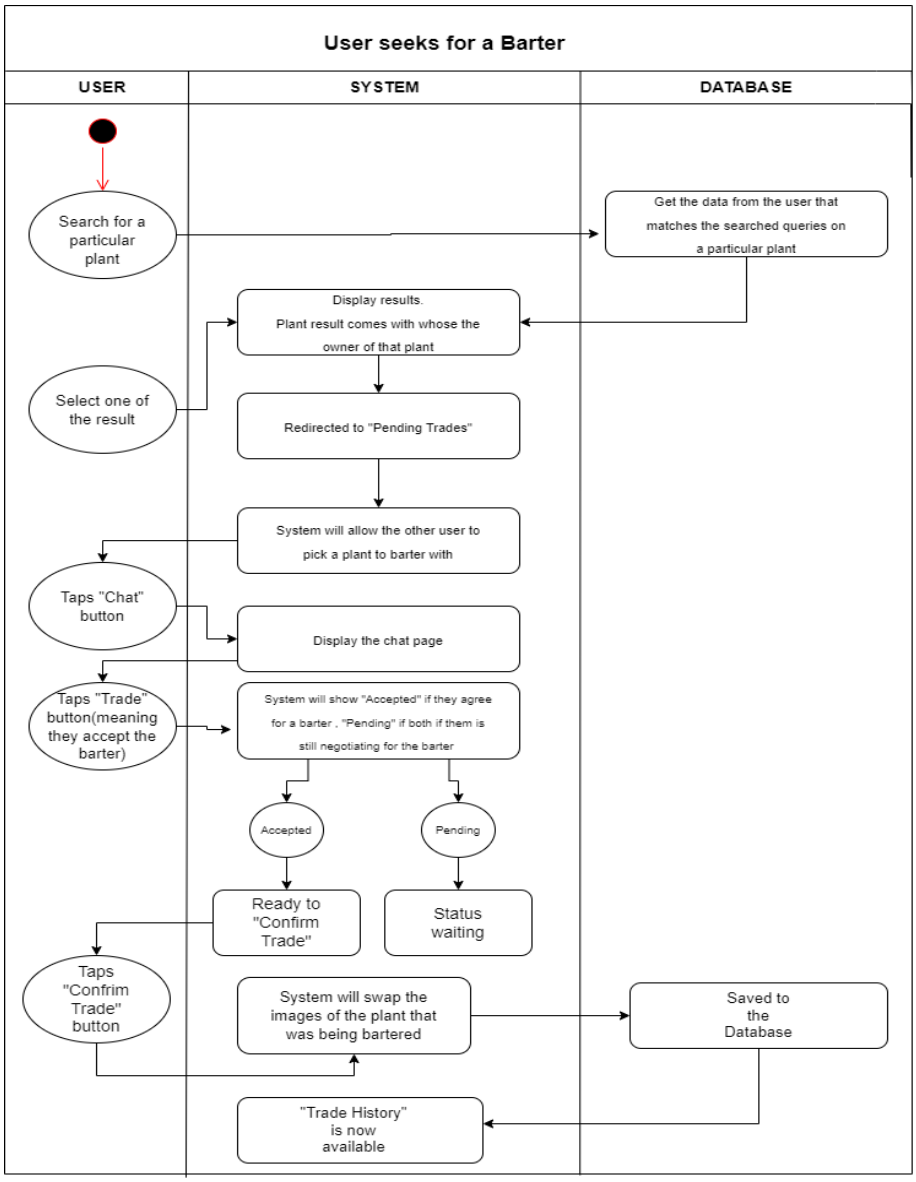


Figure 3.5 Activity Diagram for User Search for Barter of Plant

Figure 3.5 Activity Diagram above shows how users barter their plants to another user. If the seeker wants to view other user inventory, the system will let the seeker view other users' inventory. If the seeker selects one from the other users' inventory system, they will then proceed to the chat system to negotiate how they will barter their plants.

Admin Verifying Plant Expert Registration

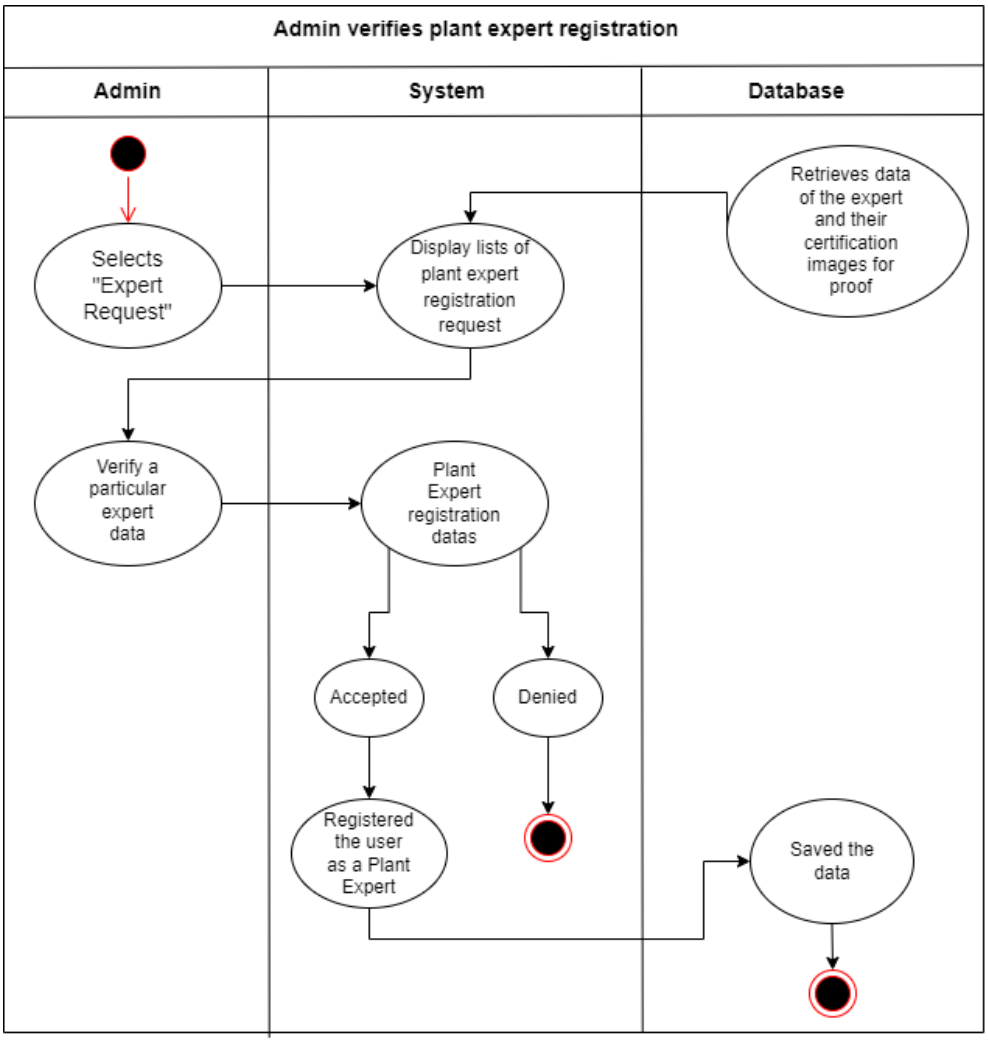


Figure 3.6 Activity Diagram for Admin Verifying Plant Expert Registration

Figure 3.6 Activity Diagram above shows how the Admin verifies a particular expert registration data. The administrator can only verify plant expert registration. When a user wants to register as a “Plant Expert,” he/she must submit a copy of his/her certification that will stand as proof.

Admin Verifying Reported Posts

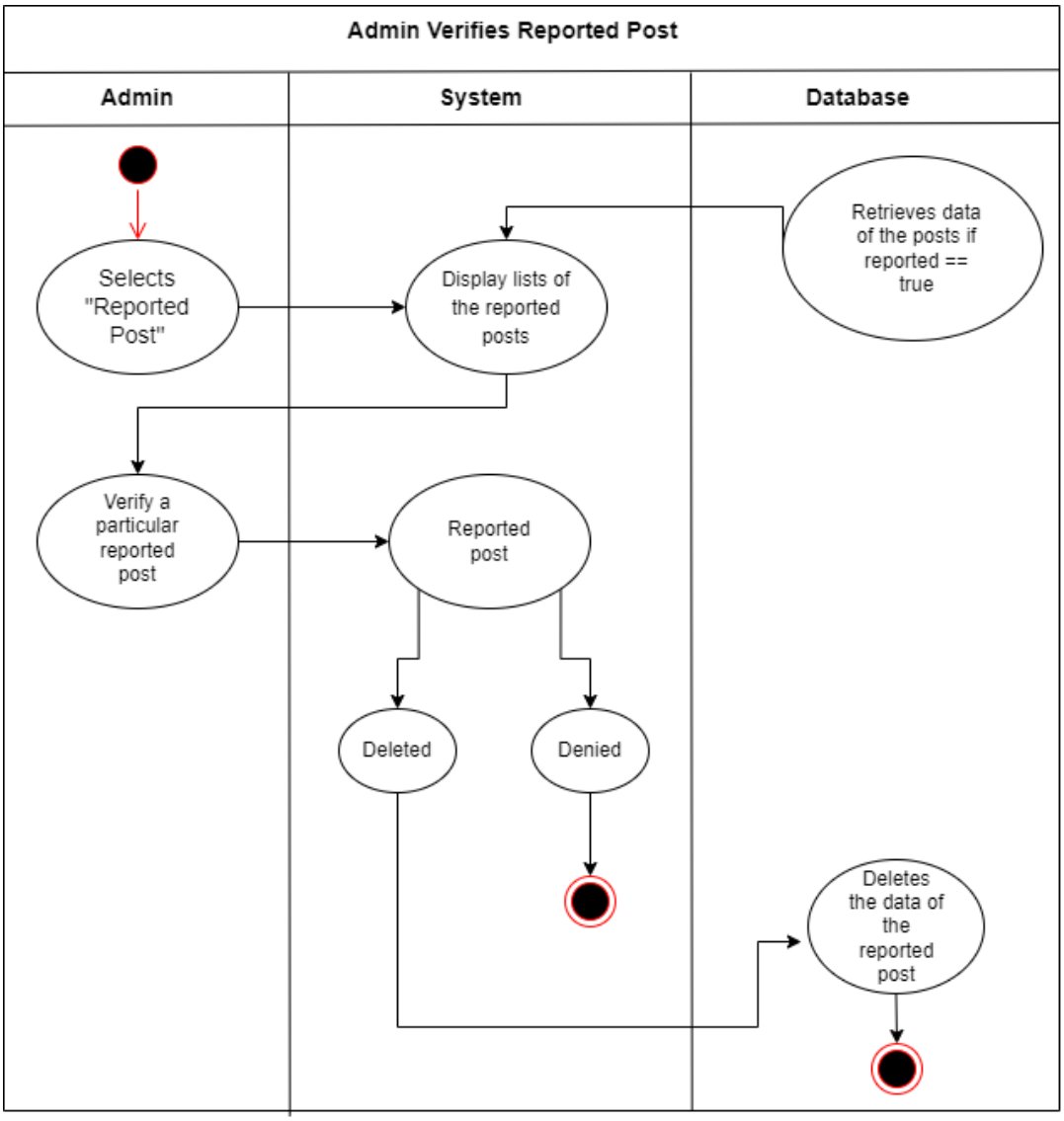


Figure 3.7 Activity Diagram for Admin Verifying Reported Posts

Figure 3.7 Activity Diagram above shows the procedure of how the Admin verifies a particular reported post. Users can report posts if they think it is inappropriate; the administrator's job is to check and verify them. The Admin will delete the data of the reported post if it is inappropriate.

Admin Changing API Key

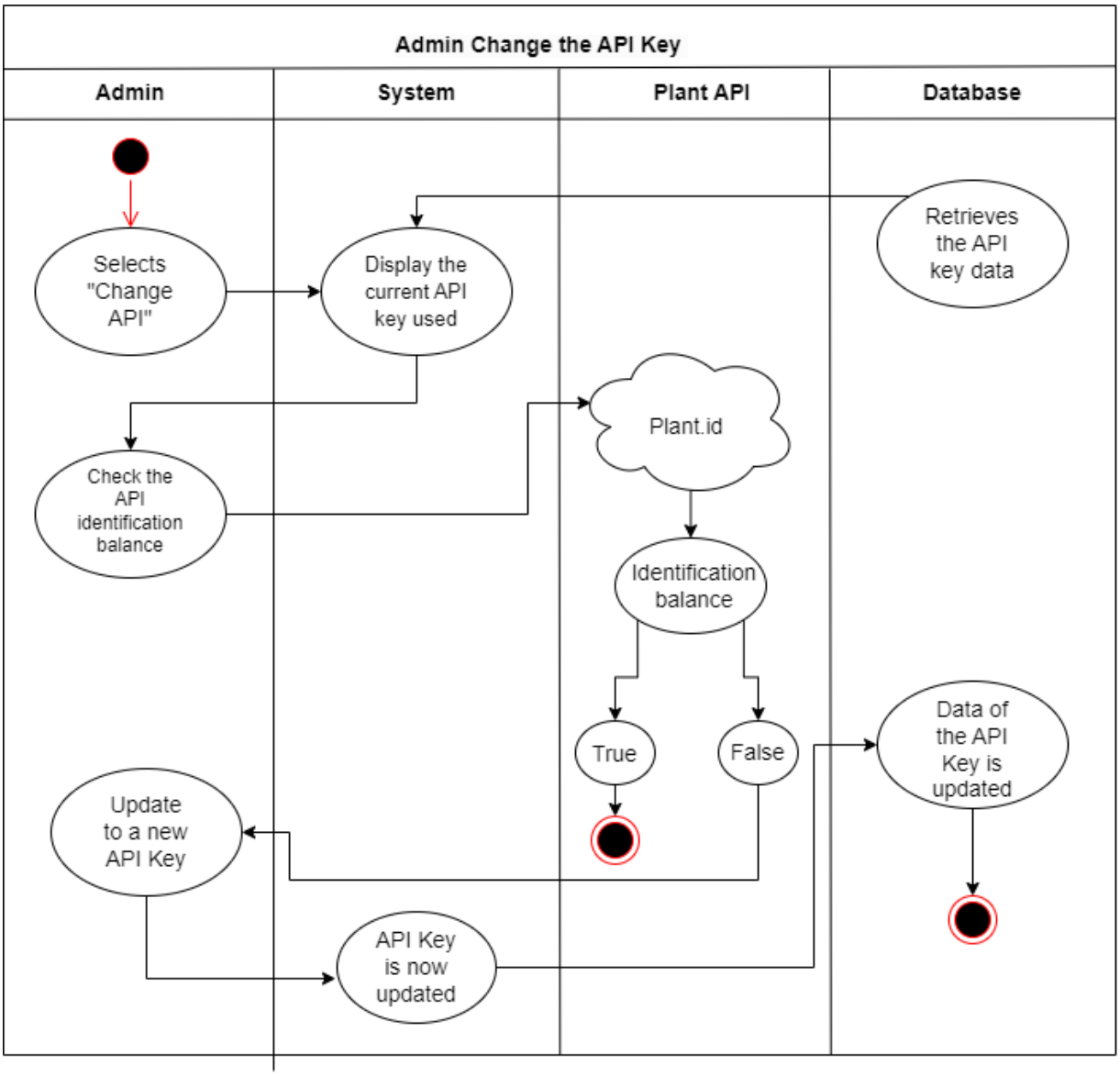


Figure 3.8 Activity Diagram for Admin Changing API Key

Figure 3.8 Activity Diagram above shows how the Admin checks the API identification balance. If the API identification balance is fully consumed, the administrator will change the API key into a new one.

CLASS DIAGRAM

The Entity Relationship Diagram shows all the relationships of the tables. It contains our data's entity relationship diagram.

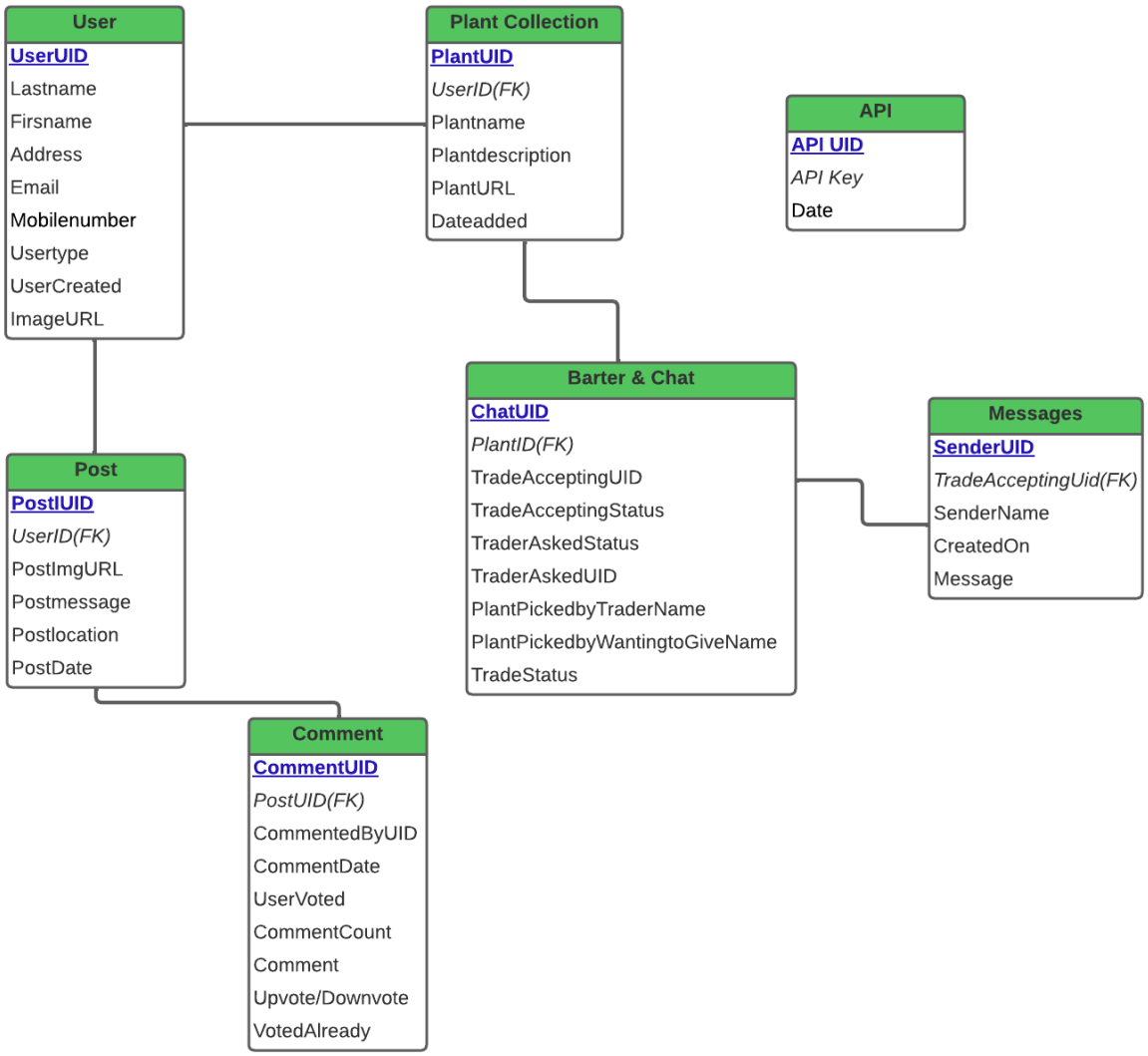


Figure 3.7: Class Diagram

Users

The entity is the user, categorizing them as plant enthusiasts or plant experts/botanists. This class stores the user's basic information. This entity is legally associated with the post entity they will post on the feed.

Plant Collection

After the user activates the identifier function and the API supplies the necessary data, this entity will conjoin the user's capital information and the payloads from the API to create the user's data to the captured plant. Users will have the option to save the identified plant to his/her plant collection.

Post

The user can post to the user feed, and they can ask questions regarding the identity of a plant to other users, especially the plant experts/botanist.

Comment

All users can comment on anybody's post. The post's author can verify whether the comments about their questions are reliable or not through the downvote and upvote buttons.

Barter & Chat

The swap and chat entity monitors the successful barter between users, storing the plant's name and persons involved in the transaction. Once the users are satisfied

with the trade, a single button will call the barter log to document the current transaction. The users can also chat with the other user during the swap.

USER INTERFACE SPECIFICATION

This section provides the user interface and user experience design for LivingGreen, exhibiting all of the application's screens for the prospective user involvement if the program.

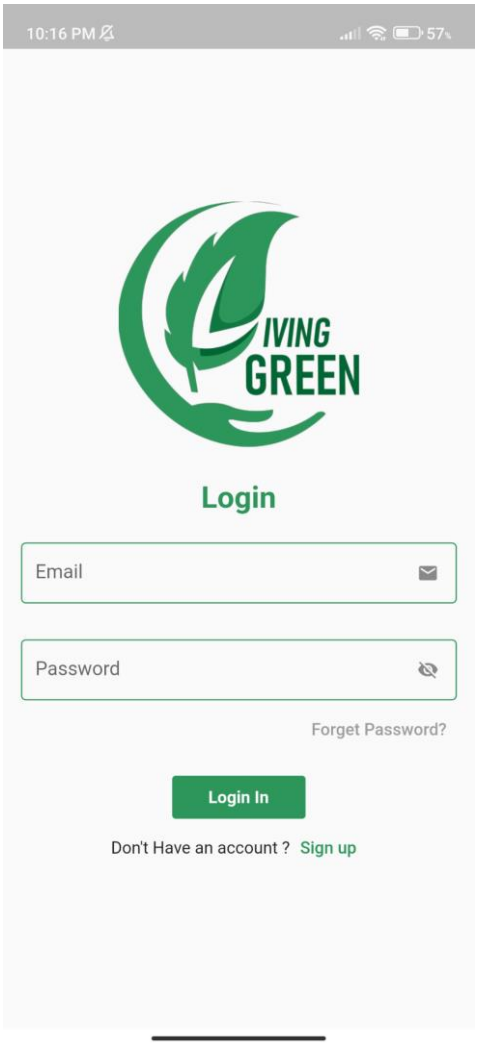


Figure 4.1: User Interface Design for Login Screen

Enables Plant Enthusiasts and Plant Experts to log in to the mobile platform. This necessitates the submission of the user's login and password. The user establishes an account by clicking the "Sign Up" button if they do not already have one.

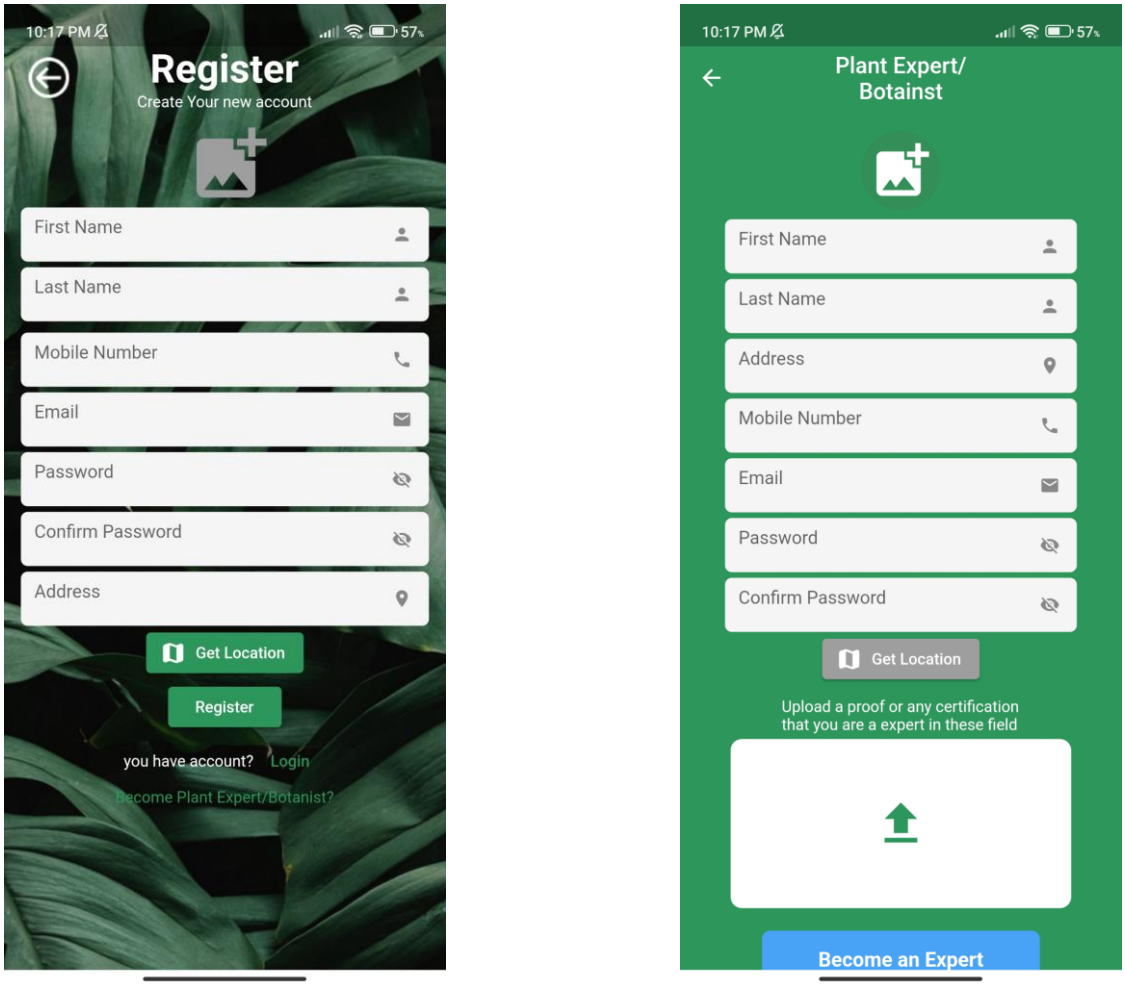


Figure 4.2: User Interface Design for Registration Page

Displays the user's required information fields. The user must submit the required information to complete the job on the page. There are types of information that the user must input; First, we will utilize personal information such as first and last names, phone

numbers, and addresses. The same in the users' registration page. The second section of account information is for the plant experts/botanist, where the botanist will upload a certification or any proof of legitimacy.

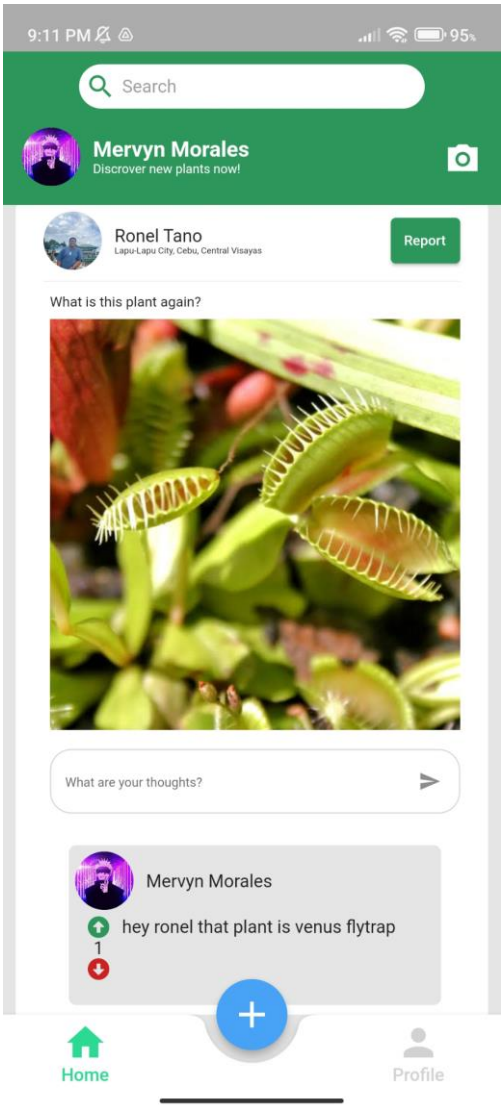


Figure 4.3: User Interface Design for Home Page

Depicts the user’s homepage. This is to handle all post data of the user’s and other users’ posts. Users can comment on a post.

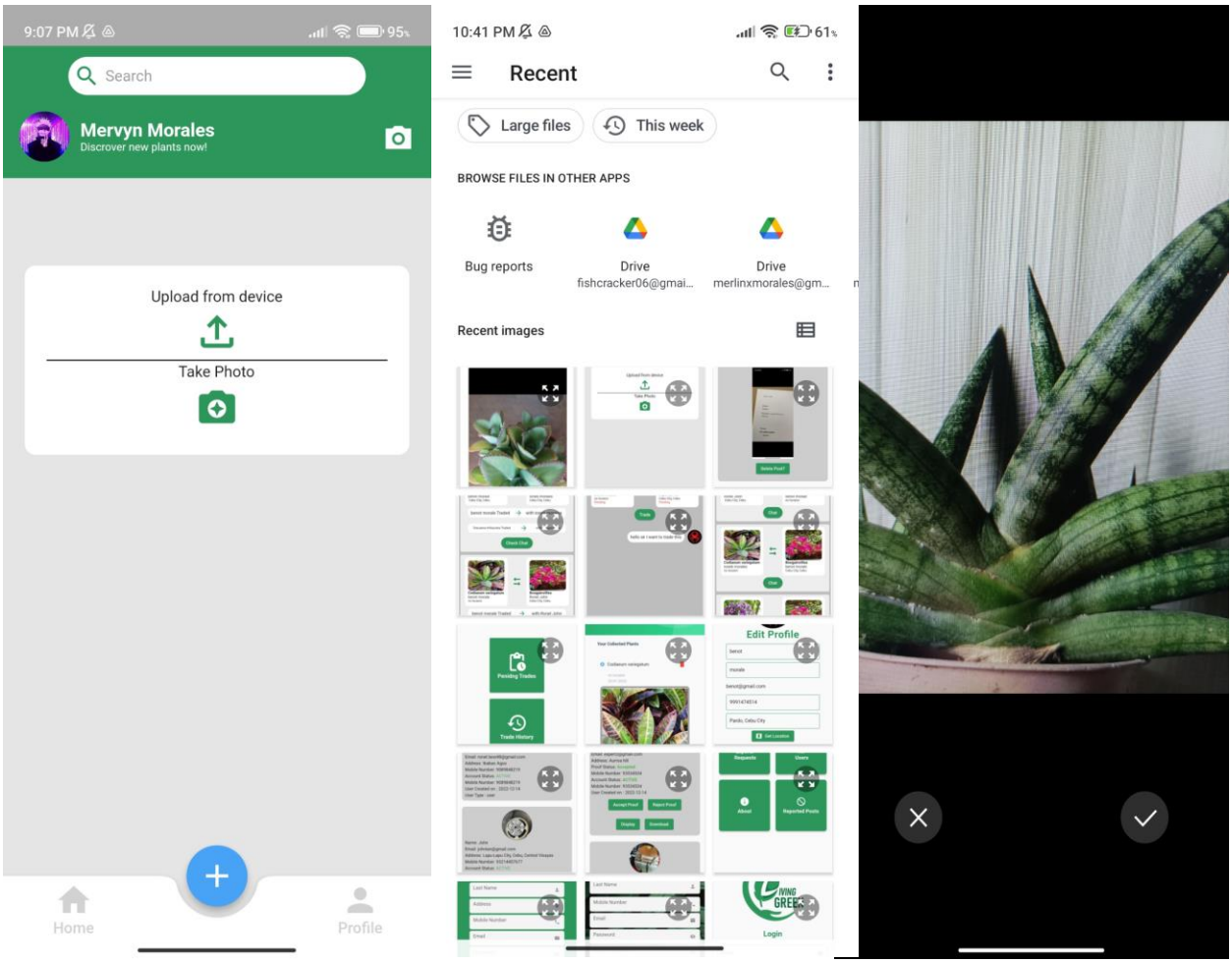


Figure 4.4: User Interface Design for Camera Page

Depicts the user's camera page. The user can choose whether to upload a photo or take a photo. If the user chooses the upload from the device button, the system will redirect the user to his/her phone gallery. If the user chooses to take a picture, the system will open the user's phone camera.

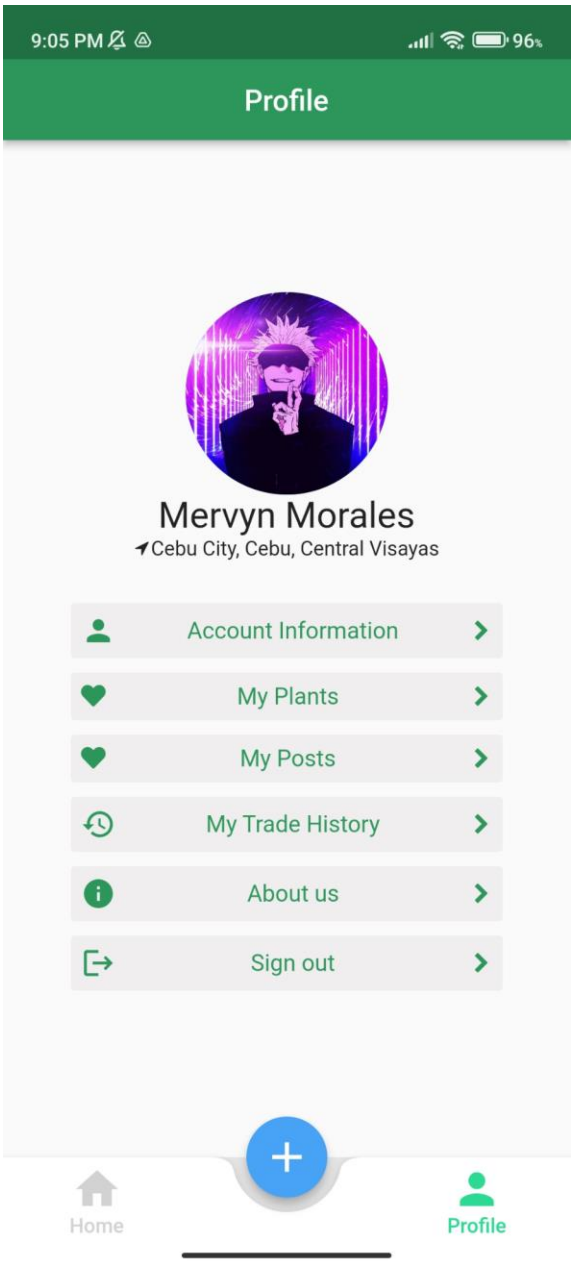


Figure 4.5: User Interface Design for Profile Page(the current user)

Displays the user some options to choose from. The users can modify their account information. The user can see his/her plant collection on the profile page. The user also can click the “My trade history” button to see his/her trade or barter request. The user can also click the “My posts” button to view his/her post.

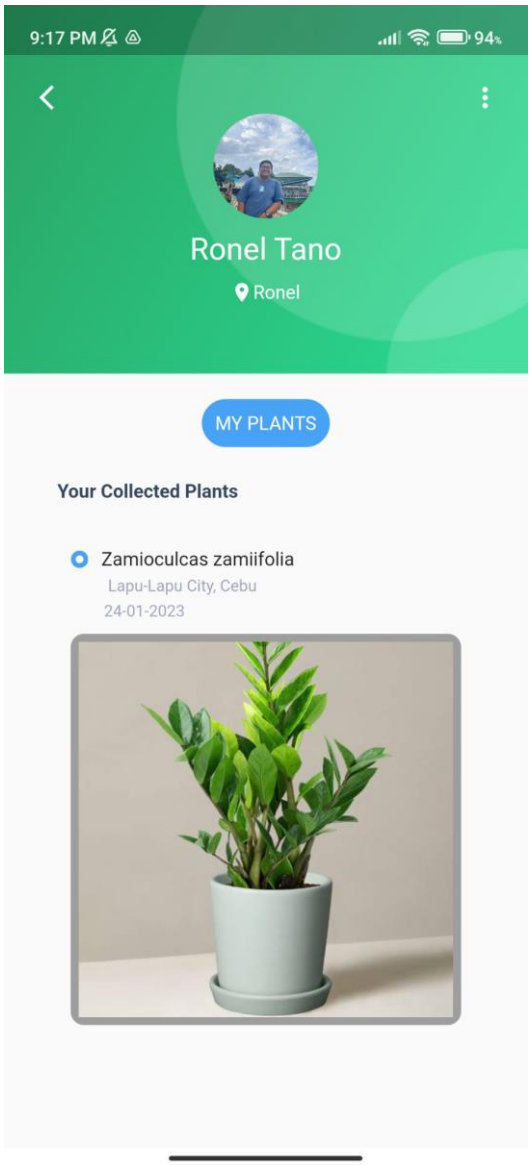


Figure 4.6: User Interface Design for Profile Page(other users)

Displays the other user's information. The users can view the plant inventory of the other user, and they can also click the image of the plant to ask for barter.

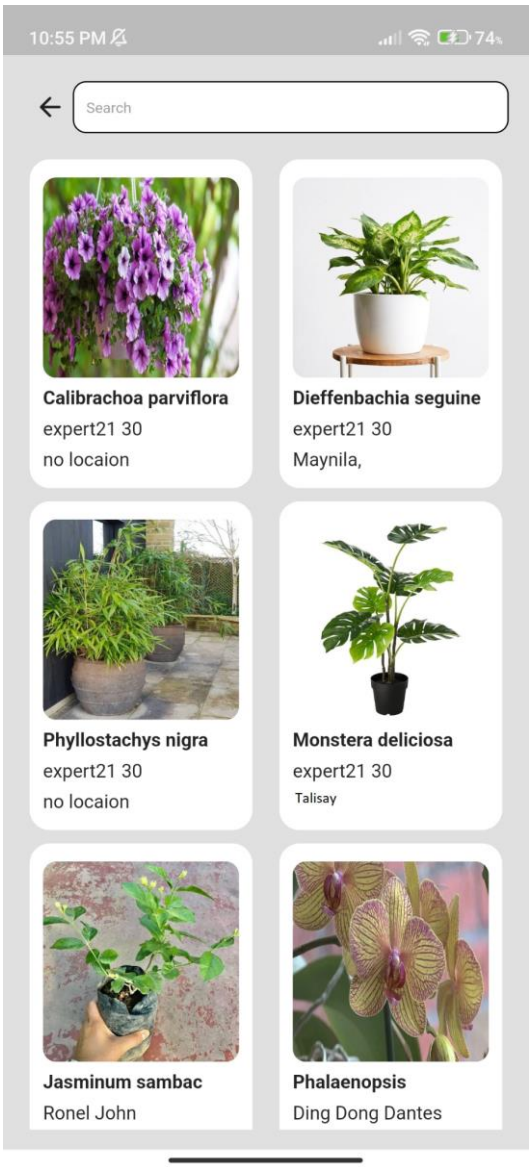


Figure 4.7: User Interface Design for Search

Depicts the user's search result. The user can search for the name of a plant.

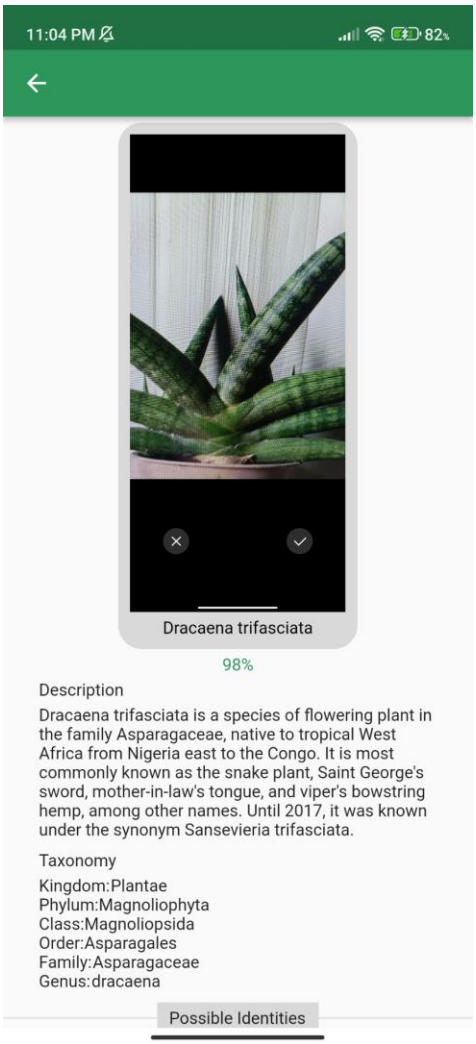


Figure 4.8: User Interface Design for Plant Identification

Displays the plant identification of a plant being fed to the API for plant identification. The user can see the plant name and description of the plant they want to be identified.

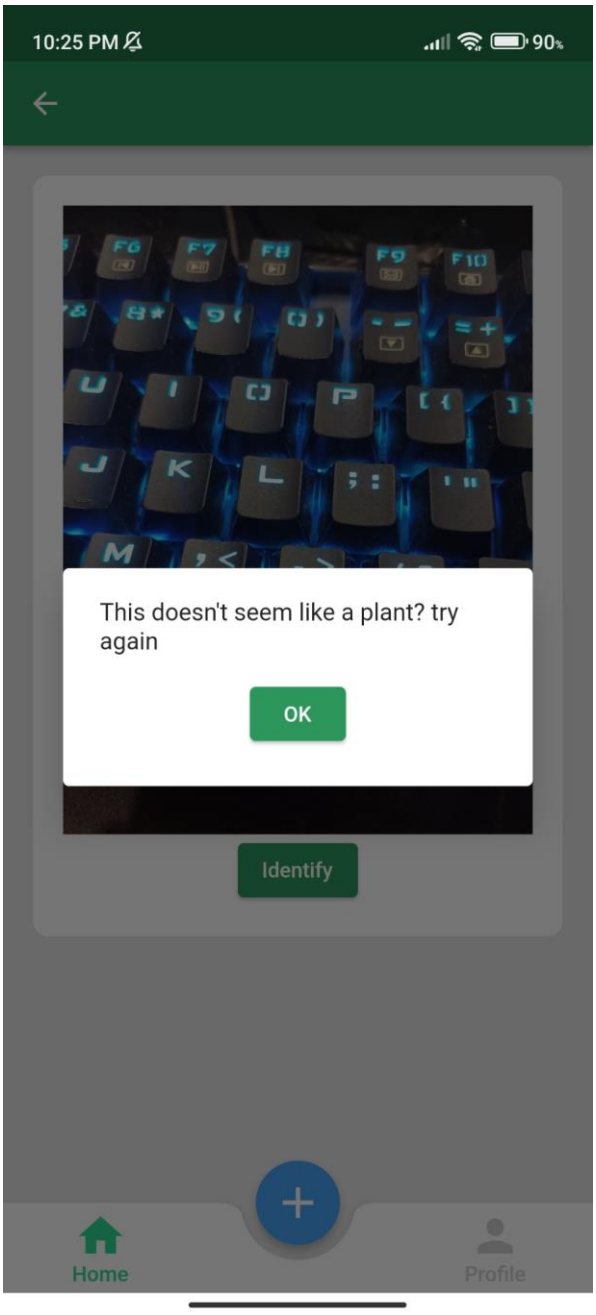


Figure 4.9: User Interface Design for uploaded or taken a photo which is not a plant

The Prompt message is deleted if the uploaded/taken picture is unrelated to plant/s.

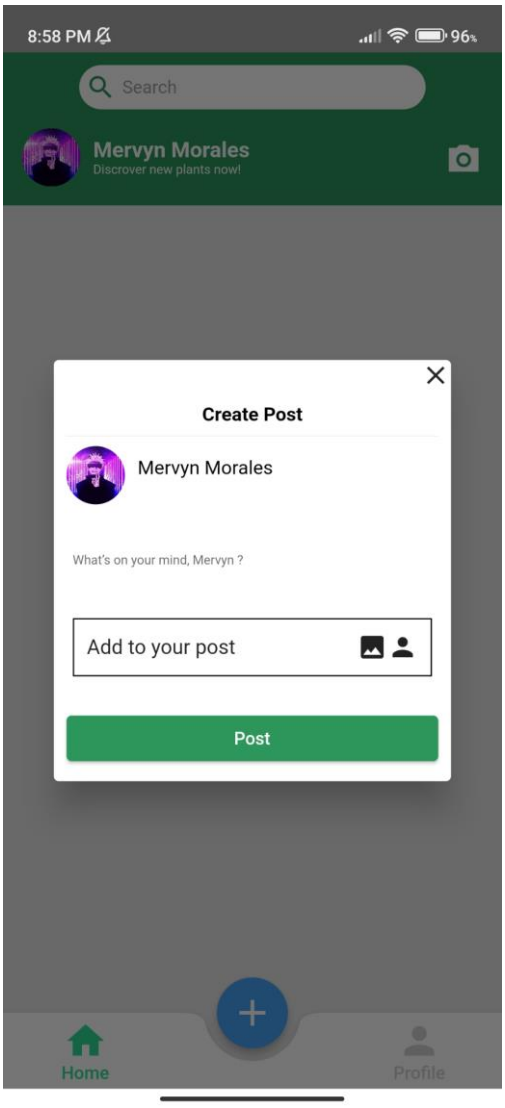


Figure 4.10: User Interface Design for Creating a Post

The user can create a post if he/she wants to clarify if the identification of his/her plant is accurate or not identified at all. The user can ask for help from other plant enthusiasts or plant experts.

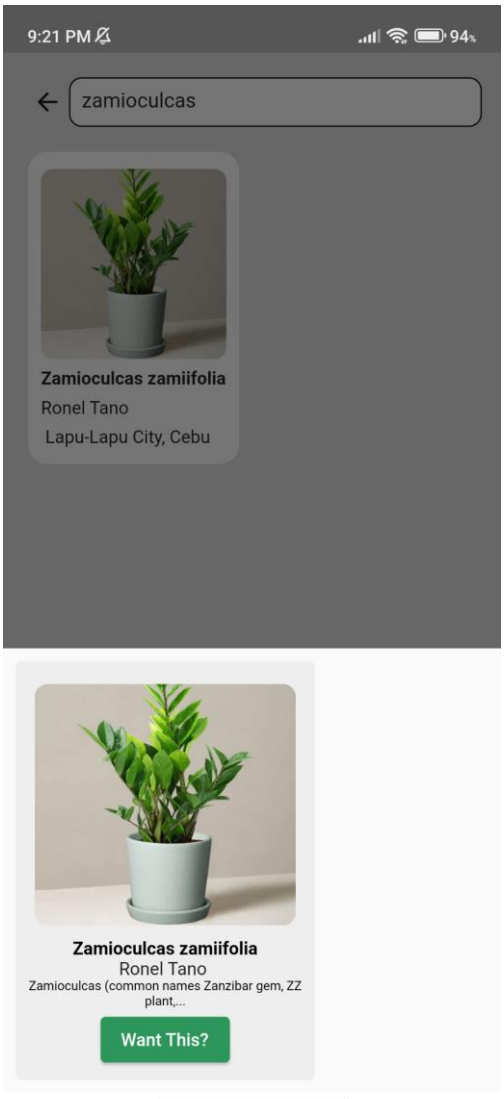


Figure 4.11: User Interface Design for viewing the searched plant name

Displays the user's brief information - name and location. The users can view the plant being searched and the plants the other users have.

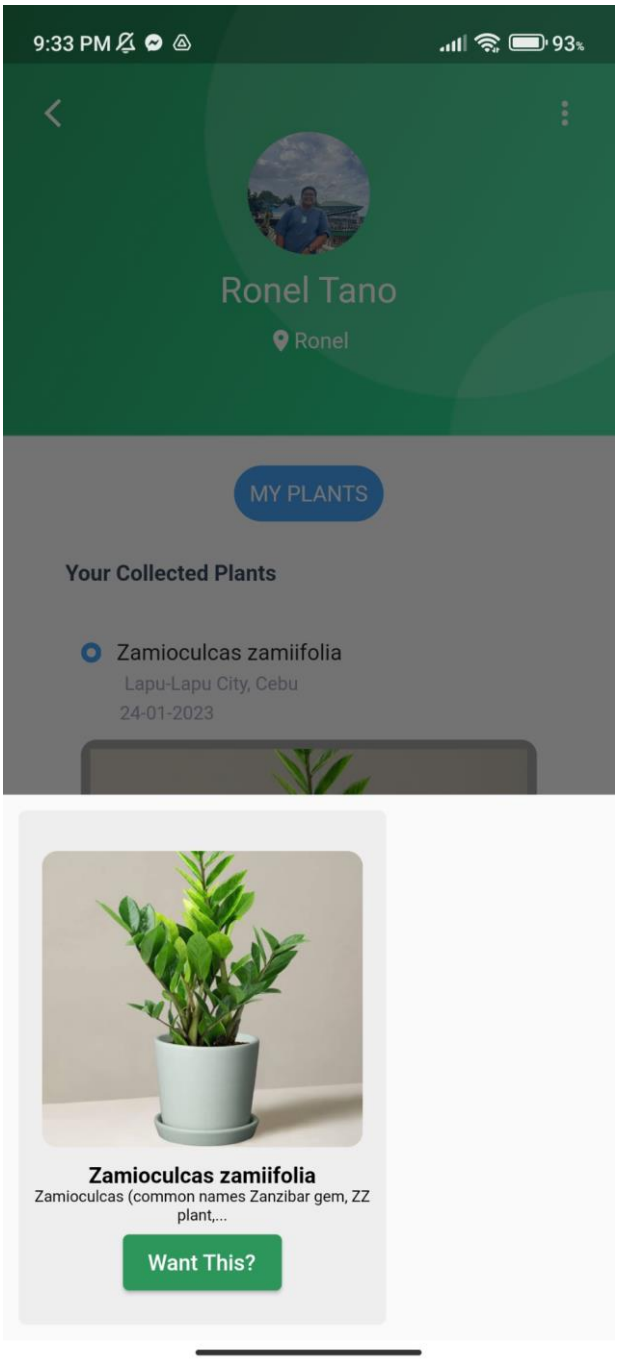


Figure 4.12: User Interface for Viewing Other Users' Inventory

Displays the “Want This?” button. The user will have the option to choose the plant to be bartered. If the user clicks “Want This”? It will be redirected to pending trades.

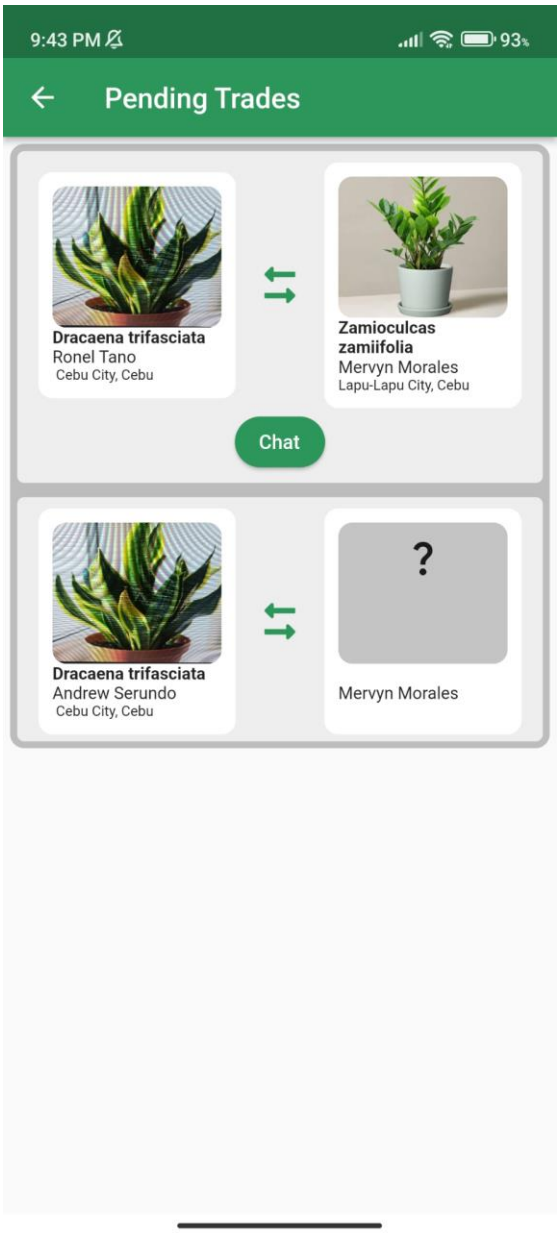


Figure 4.13: User Interface Design when a User received a trade request

Depicts the user's trade request. This also shows the images of the plants that they want to barter.

Figure 4.14: User Interface Design for Chat with other users

Depicts the user's chat box. Users can chat with other users and request barter on a certain plant they do not have in their homes. Users can also ask the barterer to arrange a meet-up for their swap.

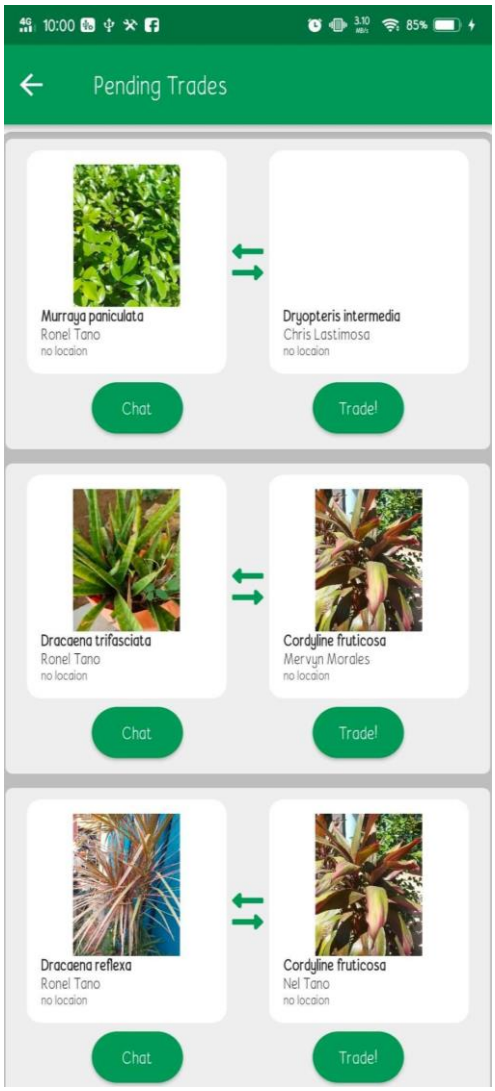


Figure 4.15: User Interface Design for Confirming Trades

Depicts the user's "Confirm Trade" button. If both users agree to the trade, it will indicate "Accepted," and the "Confirm Trade" button will appear.

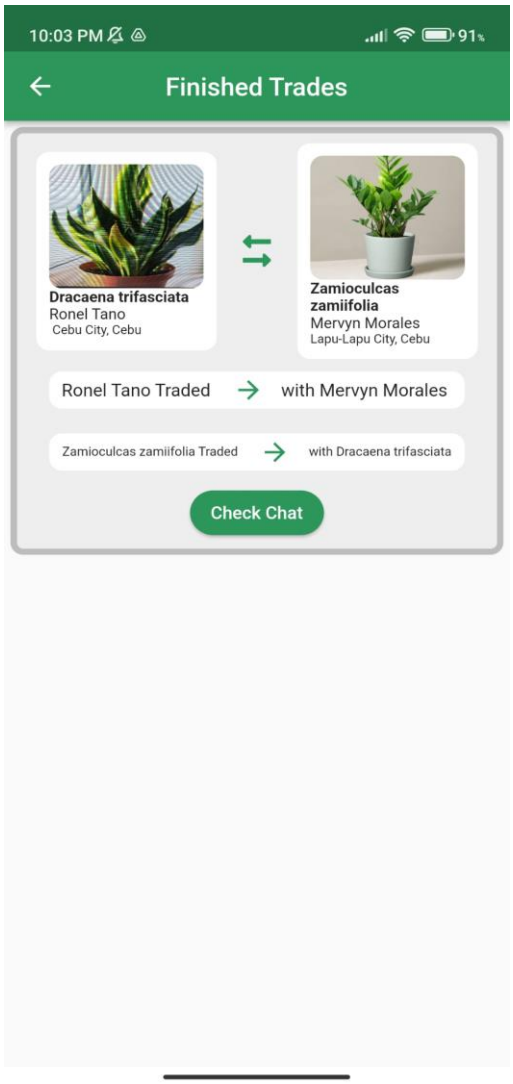


Figure 4.16: User Interface Design for the User's successful trade(Trade History)

Depicts the user's successful trades. Every user's successful trades will have a trade history, and it also saves their chat conversations.

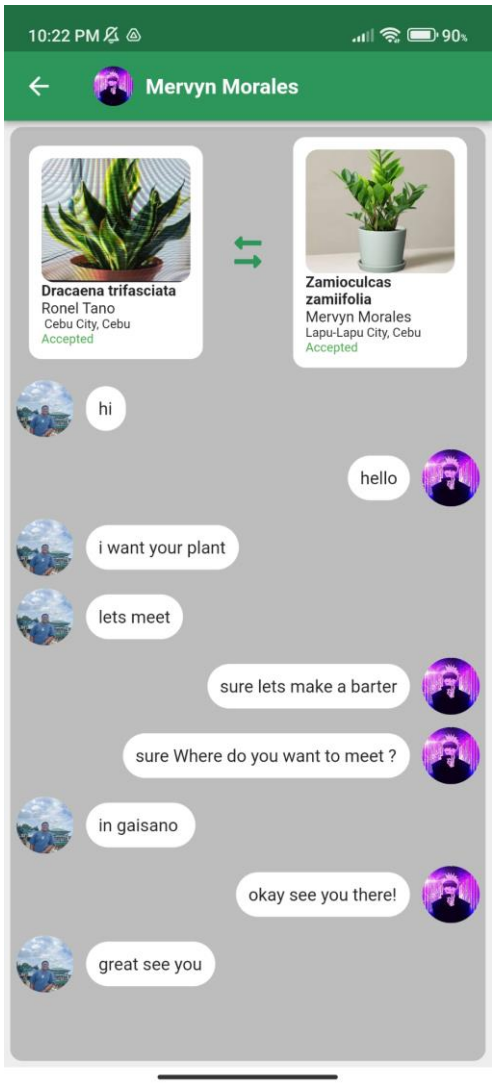


Figure 4.17: User Interface Design for User's Chat History

Depicts the user's chat history. For every successful trade, the chat history of every user will also be saved.

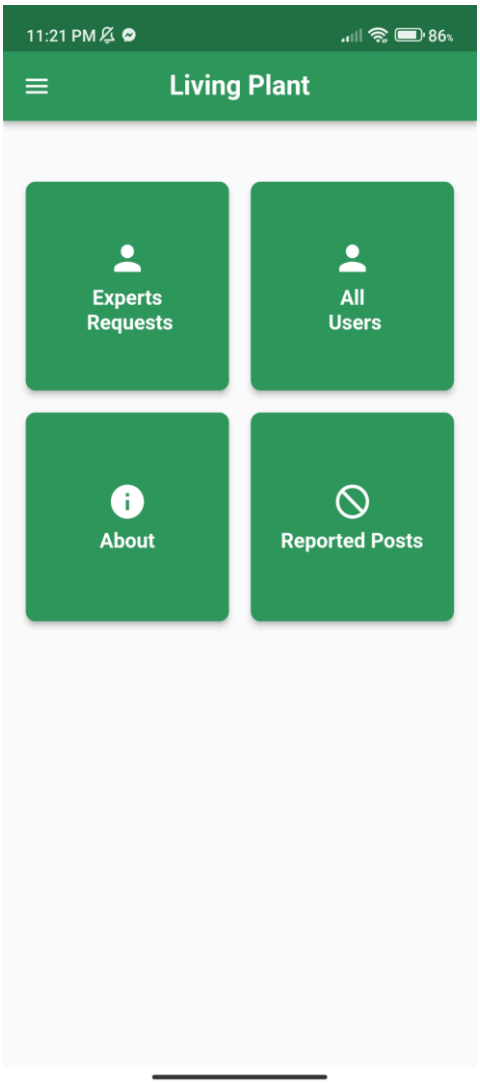
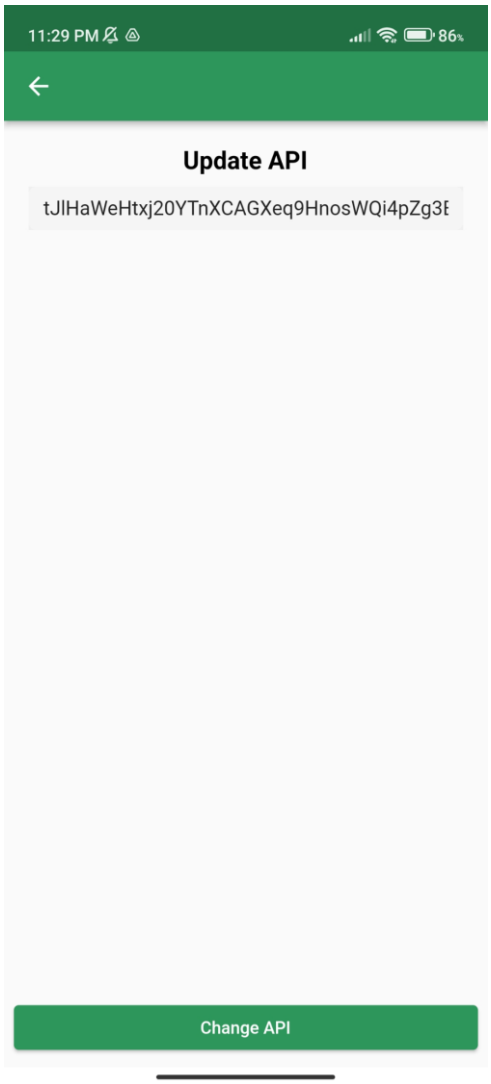


Figure 4.18 User Interface Design for the Administrator

Depicts the options available for administrators. The administrator can access all the options that create and maintain most systems.



4.19 User Interface Design for Changing API Key

Depicts the change of API key. The administrator can change the API key every time the allotted number of plant identification in the API is all consumed.

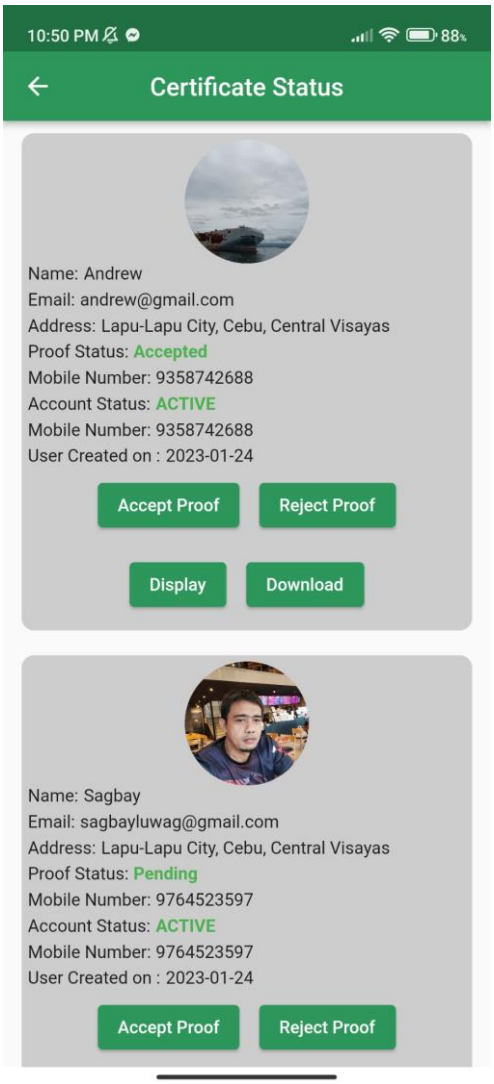


Figure 4.20: User Interface Design for Plant Expert Verification

Depicts the user's request to be registered as a plant expert. Before a user is approved to be a plant expert, the administrator will verify their request.

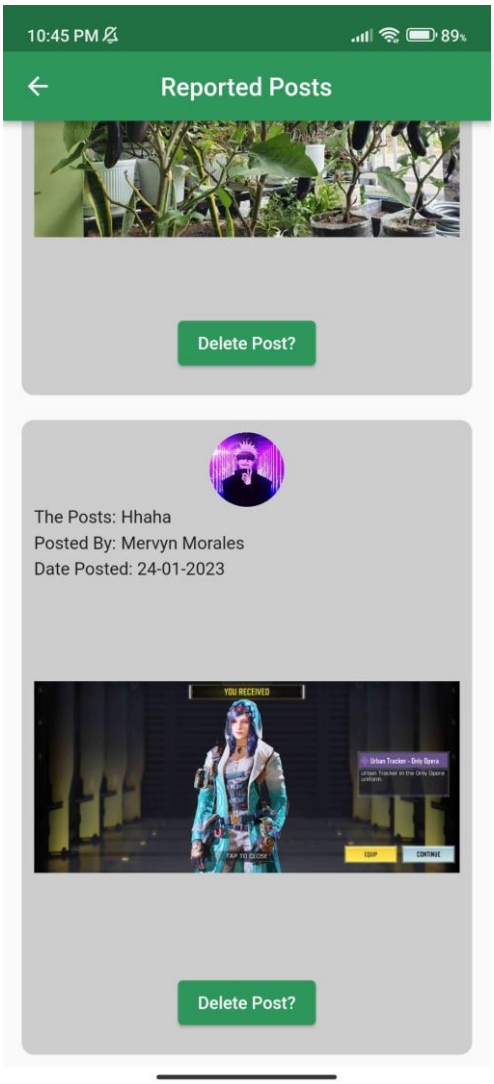


Figure 4.21: User Interface Design for Reported Post

Depicts the reported posts from the feed. The administrator can verify or delete the reported post whether the reported post is plant related or not.

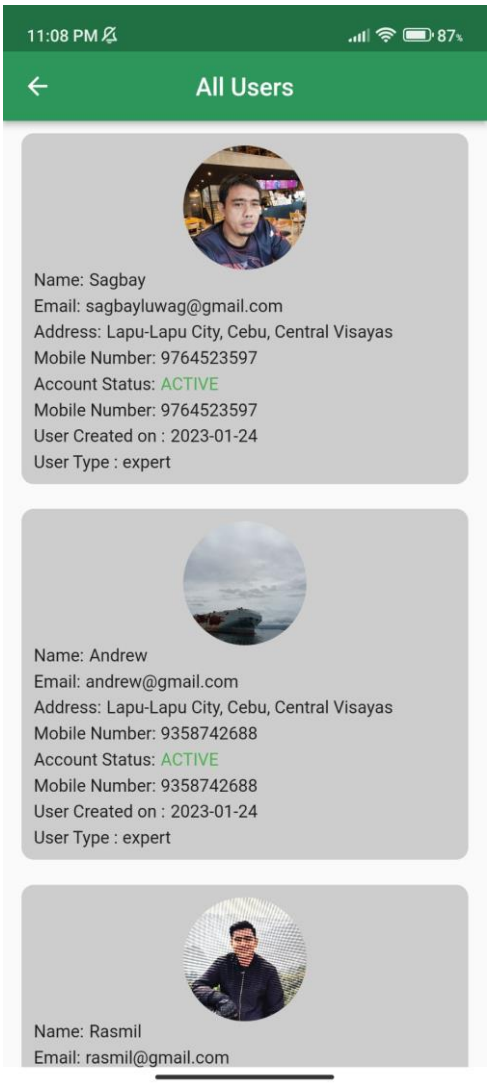


Figure 4.22: User Interface Design for all Registered Users

Depicts the list of registered users. The administrator can view all the registered users, and the Admin can identify each user, whether an expert or a user.

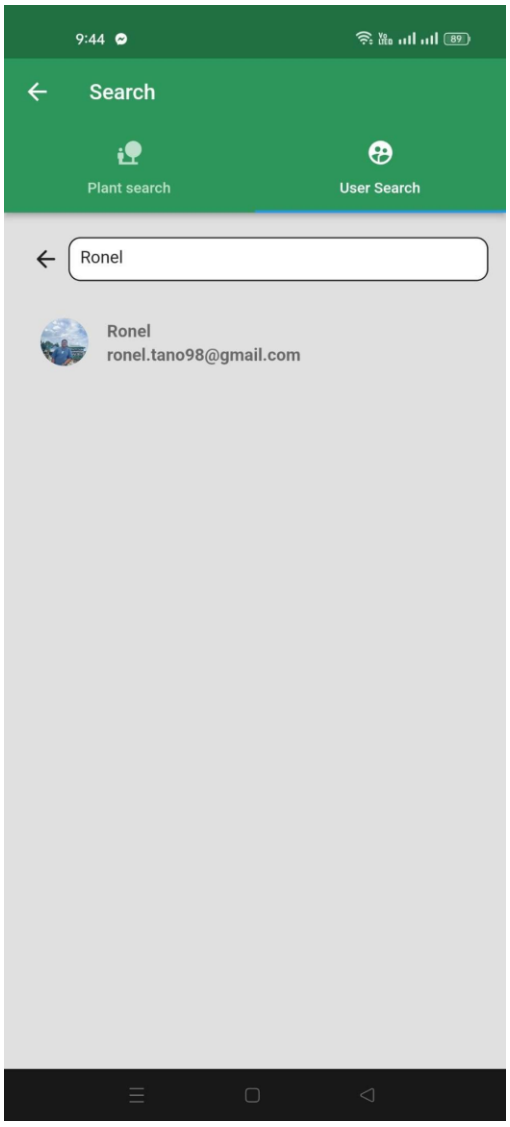


Figure 4.23: User Interface Design for the searched user's name

Depicts the list of registered users. The users can view the searched name.

CHAPTER III

SOFTWARE DEVELOPMENT AND TESTING

This chapter summarizes how the researchers developed the platform, integrated different environments, used management tools for development, and tested the process. It breaks down the research's importance by identifying the application's root. It also explains the functionalities and process of development of how the researchers implemented such a development process.

DEVELOPMENT SOFTWARE PLATFORMS, DEVELOPMENT ENVIRONMENTS, AND PROJECT MANAGEMENT TOOLS

The researchers utilized Visual Studio Code and Android Studio for mobile development. In mobile, the researchers use Visual Studio Code and Android Studio as their primary programming source-code editors for mobile. Moreover, Android Studio has a Gradle-based build system, emulator, code templates, and GitHub integration. Every Android Studio project has at least one mode containing source code and resource files. The operating system that is being used for the development is Windows 10 operating system.

The researchers used Flutter as their primary programming language for mobile and web development. Google developed Flutter in May 2017 and released it as a free, open-source software development kit. It is also known as a user interface framework, a collection of changeable user interface components that comprise an application modifiable during development. The programming language Dart is the foundation for

Flutter. We implemented the provider package because it is easier to understand, does not use much code, and uses concepts that apply to every other approach.

The database used for LivingGreen is a NoSQL database called Firebase. This serverless document database can be easily scalable. The security rules could also be customizable to ensure that data is always protected in the LivingGreen system.

DEVELOPMENT PROCESS

In this section, both the project development and testing processes are discussed and explained to better understand what's happening in the application.

Database

The researcher used the Firestore database, a NoSQL database for the backend part, which is free and efficient. Cloud Firestore is fast and provides real-time updates to the database changes. This is where the data of users are stored securely.

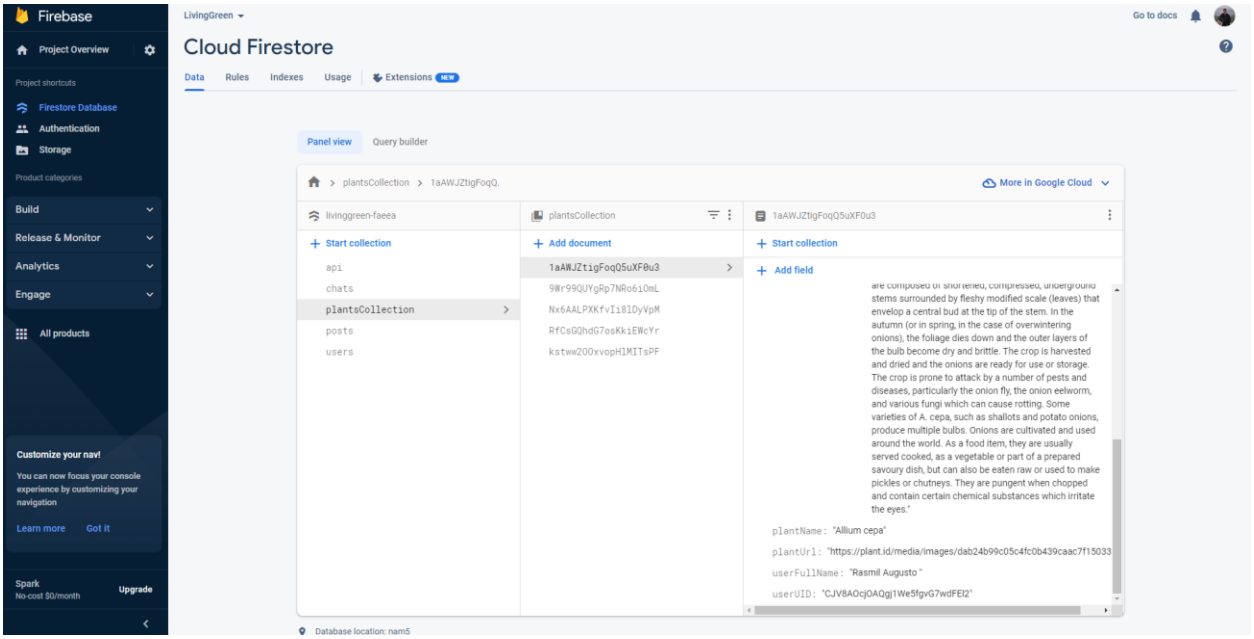


Figure 5: Firestore Database

Image Management System

The researcher used the Cloud Storage for Firebase to handle the images of LivingGreen, as shown in Figure 5 above. Cloud Storage is a powerful, cost-effective object storage service that Google builds. This is where the images of users are stored and secured. Cloud Storage is also scalable, which is suitable for the users of LivingGreen.

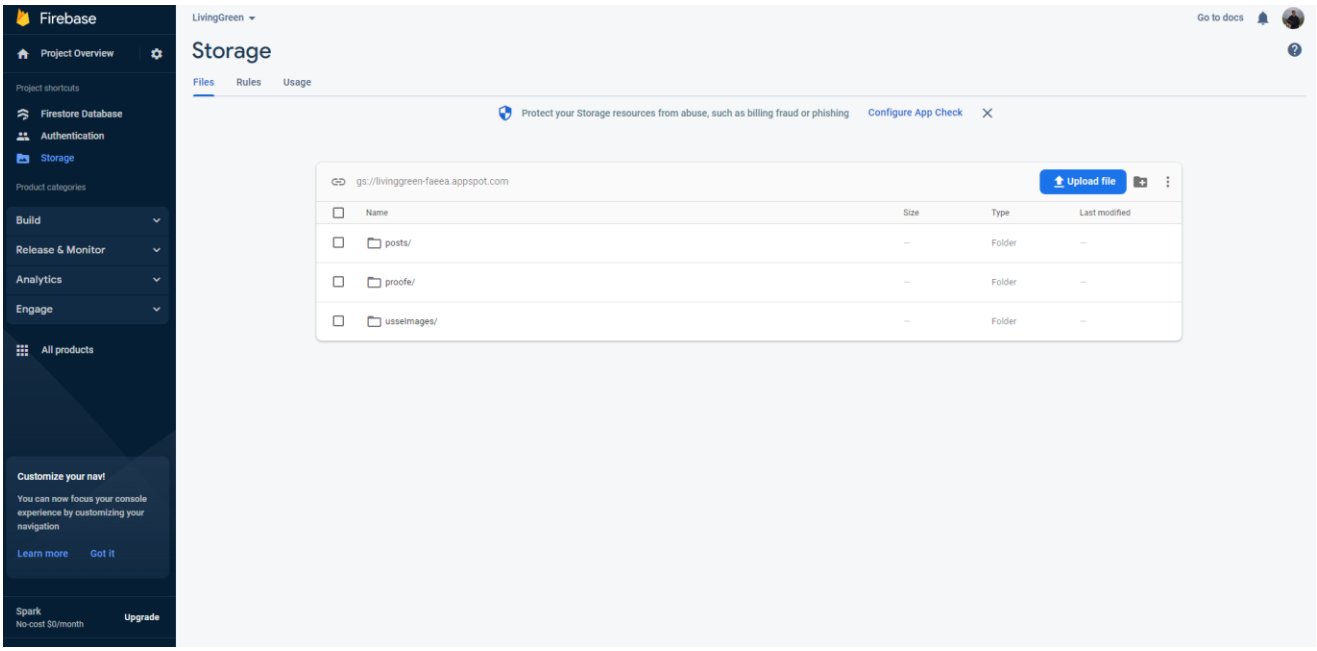


Figure 5.1: Storage for Firebase

State Management

When developing applications using Flutter, state management often comes into the discussion because there should be a plan on how to efficiently retrieved or passed data from one another.

The “state” in Flutter refers to the data stored inside a widget that can be modified depending on the current operation. The state of an app can be updated or completely

changed at the start of an application or when a page reloads. That means everything widgets do requires handling the data retrieved from the user and passing it among themselves to perform one or more operations. Flutter can also use the state to display information to the user.

The provider is the state management used for the development of LivingGreen. The Provider package, created by Remi Rousselet, aims to handle the state as cleanly as possible. In the provider, widgets listen to changes in the state and update as soon as they are notified.

Therefore, instead of the entire widget tree rebuilding when there is a state change, only the affected widget is changed, thus reducing the amount of work and making the app run faster and more smoothly.

```
runApp(
  MultiProvider(
    providers: [
      ListenableProvider<postPageProvider>(
        create: (_) => postPageProvider(),
      ), // ListenableProvider
      ListenableProvider<imageRecogntionProvider>(
        create: (_) => imageRecogntionProvider(),
      ), // ListenableProvider
      ListenableProvider<traderProvider>(
        create: (_) => traderProvider(),
      ),         // ListenableProvider
    ],
    child: const MyApp(),
  ), // MultiProvider
);
```

Figure 6: Code Snippet of ListenableProviders

Accept or decline a request to be a plant expert.

```
acceptCertficate(String certificateID) async {
  showDialog(
    context: context,
    builder: (_) => loadingDialog(message: "Accepting Certificate"),    Pre
  );
  await LivingPlant.firebaseFirestore!
      .collection("users")
      .doc(certificateID)
      .update({
    "proofeStatus": "Accepted",
  });
  Navigator.pop(context);    Do not use BuildContexts across async gaps.
}

// accept functoin
rejectCertficate(String certificateID) async {
  showDialog(
    context: context,
    builder: (_) => loadingDialog(message: "Rejecting Certificate"),    Pre
  );
  await LivingPlant.firebaseFirestore!
      .collection("users")
      .doc(certificateID)
      .update({
    "proofeStatus": "Rejected",
  });
  Navigator.pop(context);    Do not use BuildContexts across async gaps.
}
}
```

Figure 6.1: Code Snippet of Accept or decline a request to be a plant expert

Figure 6.1 shows how the Admin can accept or decline a user that wants to be a plant expert in the app. If the user has legitimate proof that he is a plant expert, the Admin will view it first, check the proof, and later on, will accept the request. Otherwise, the Admin will reject it, and it will be marked as “rejected,” and the user cannot be granted to be a plant expert user in the application.

Plant Identification Feature

```
identifyPlant() async {
  showDialog(
    context: context,
    barrierDismissible: false,
    builder: (_) =>
        const loadingDialog(message: "Identifying plant... Please wait."),
  );
  requestingData();
}
```

```
var client = http.Client();
var url = Uri.parse("https://api.plant.id/v2/identify");
// Getting Data
var gettingAPI = await LivingPlant.firebaseFirestore!
    .collection("api")
    .doc("HEmMnpsEKMje2B6TUSYU")
    .get();

print(gettingAPI.data()!['api']);    Avoid `print` calls i
String api = gettingAPI.data()!['api'];
var headers = {
  "Content-Type": "application/json",
  "Api-Key": api,
};

final Map body = {
  "images": [base64Image],
  "modifiers": ["similar_images"],
  "plant_details": [
    "common_names",
    "url",
    "wiki_description",
    "taxonomy",
    "synonyms"
  ],
  "suggestions": ["synonyms", "taxonomy"]
};
```

Figure 6.2: Code Snippet of Plant Identification

Figure 6.2 shows the plant identification feature code that uses a Plant Identification API to identify different types of plants based on their features. The code sends an image of the plant to the API, which then processes the image using machine-learning algorithms to determine the plant species.

The API we are using is plant. Id, which uses a database of plant images and their respective plant species to make the identification. When the API receives an image, it compares it with the images in the database and returns the plant species that best matches it.

The code then displays the name and other information about the identified plant species to the user. To use the Plant Identification Feature, you need access to the API and a proper API key. The API key is used to authenticate the code and ensure only authorized users can access the plant identification service.

Ask the People Function

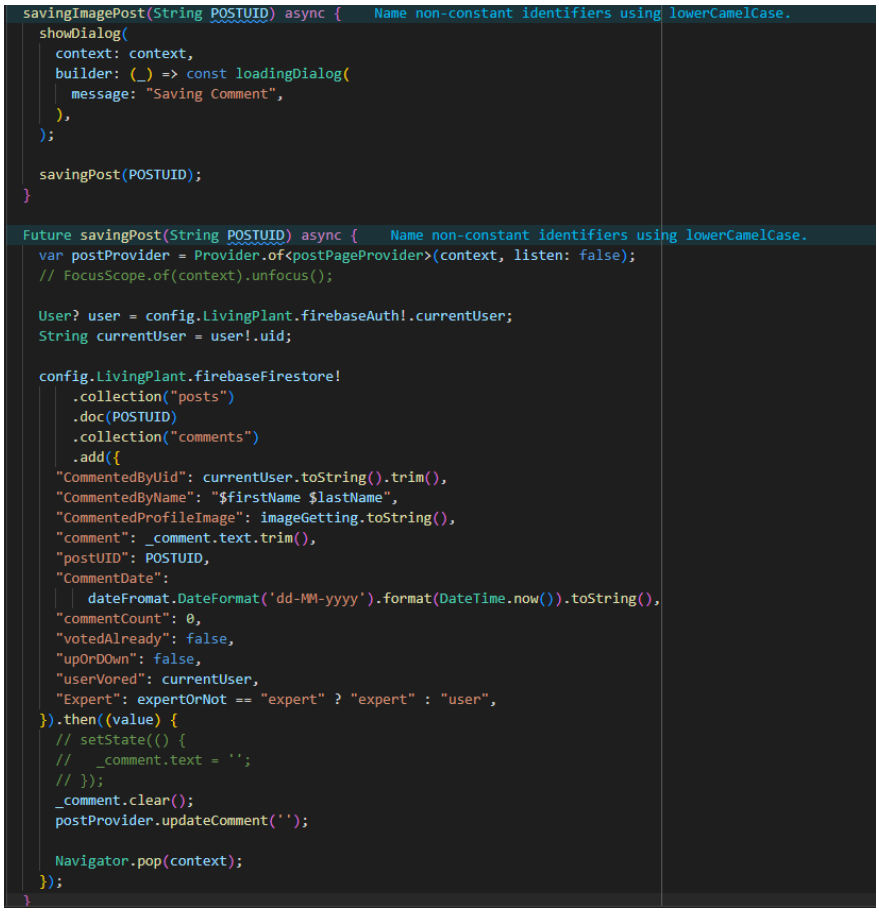
```
askThePropleFunction() async {
  var gettingDataFromPriovider =
      Provider.of<imageRecogntionProvider>(context, listen: false);

  module.Welcome model =
      module.welcomeFromJson(gettingDataFromPriovider.gettingResponseData);
  var providerPage = Provider.of<postPageProvider>(context, listen: false);

  providerPage.updateingImage(XFile(model.images![0].url!));
  providerPage.updatePostMessage(model.suggestions![0].plantName.toString());

  print("This is the XFile ${providerPage.xFileGetting!.path}");    Avoid `print` calls in production code.
  print("This is the XFile ${providerPage.plantName}");    Avoid `print` calls in production code.

  Navigator.pop(context);
  Route route = MaterialPageRoute(builder: (_) => const bottomAndUp());
  Navigator.pushReplacement(context, route);
 }
}
```

Figure 6.3: Code Snippet of Ask the People Function

Figure 6.3 shows the Ask the People function in Flutter using provider state management code that allows users to create and upload new posts to the application while managing the state of the application. State management in Flutter refers to the process of managing the data that is shared and updated throughout the application.

In this implementation, the provider package is used to manage the state of the application, specifically the state of the post data. The provider package provides a way to store and access data throughout the application, making it easier to update the UI in real-time when the state changes.

Saving Post



```
savingImagePost(String POSTUID) async {    Name non-constant identifiers using lowerCamelCase.
  showDialog(
    context: context,
    builder: (_) => const loadingDialog(
      message: "Saving Comment",
    ),
  );

  savingPost(POSTUID);
}

Future savingPost(String POSTUID) async {    Name non-constant identifiers using lowerCamelCase.
  var postProvider = Provider.of<postPageProvider>(context, listen: false);
  // FocusScope.of(context).unfocus();

  User? user = config.LivingPlant.firebaseAuth!.currentUser;
  String currentUser = user!.uid;

  config.LivingPlant.firebaseFirestore!
      .collection("posts")
      .doc(POSTUID)
      .collection("comments")
      .add({
    "CommentedByUid": currentUser.toString().trim(),
    "CommentedByName": "$firstName $lastName",
    "CommentedProfileImage": imageGetting.toString(),
    "comment": _comment.text.trim(),
    "postUID": POSTUID,
    "CommentDate":
        dateFromat.DateFormat('dd-MM-yyyy').format(DateTime.now()).toString(),
    "commentCount": 0,
    "votedAlready": false,
    "upOrDOwn": false,
    "userVored": currentUser,
    "Expert": expertOrNot == "expert" ? "expert" : "user",
  }).then((value) {
    // setState(() {
    //   _comment.text = '';
    // });
    _comment.clear();
    postProvider.updateComment('');

    Navigator.pop(context);
  });
}
```

Figure 6.4: Code Snippet of Saving Post

Figure 6.4 shows the saving post function, which uses a provider state management and Firebase, allowing the user to store and manage the post data in the application in real time. This approach allows the user to easily share the post data with multiple users and update it in real time as it changes. It changes when other users comment on it.

Upvote and Downvote a comment

```
if (userVored != currentUser) {
  showDialog(
    context: context,
    builder: (_) => const loadingDialog(message: ""),
  );
  await config.LivingPlant.firebaseFirestore!.collection("posts").doc(postUID).collection("comments").doc(CommentUID).update({
    "commentCount": CommentCount! + 1,
    "votedAlready": true,
    "userVored": currentUser,
  }).then((value) {
    Navigator.pop(context);
  });
}
```

```
if (userVored != currentUser) {
  showDialog(
    context: context,
    builder: (_) => const loadingDialog(message: ""),
  );
  await config.LivingPlant.firebaseFirestore!.collection("posts").doc(postUID).collection("comments").doc(CommentUID).update({
    "commentCount": CommentCount! - 1,
    "votedAlready": true,
    "userVored": currentUser,
  }).then((value) {
    Navigator.pop(context);
  });
}
```

Figure 6.5: Code Snippet of Upvote and Downvote a comment

Figure 6.5 shows how a user can upvote or downvote a comment. If a user finds a comment helpful, the user can upvote it so that the comment will be bumped upward, and if a user finds a comment irrelevant or unhelpful, the user can downvote it for it to be bumped down.

```
var votedAlready =
    commentSnapshot.data!
                .docs[
            CommentIndex]
        [
        'votedAlready'];
```

Figure 6.6: Code Snippet of you can use only Upvote and Downvote a comment once per comment

If a user has already voted on a comment, the user can no longer upvote or downvote again because a user can only vote once per comment.

Report a Post

```
onPressed: () async {
  showDialog(
    context: context,
    builder: (_) =>
        const loadingDialog(
            message: ""), // loadingDialog
  );
  await config
      .LivingPlant.firebaseFirestore!
      .collection("posts")
      .doc(snapshot
          .data!.docs[index].id)
      .update({
    "reported": true,
  }).then(
    (value) {
      Navigator.pop(context);
      showDialog(
        context: context,
        builder: (_) => const errorDialog(
            message:
                "Post was reported, admin will check"),
      );
    },
  );
},
```

Figure 6.7: Code Snippet of Report a Post

Figure 6.8 shows how a user can report a post. If a user finds a post that is inappropriate or misleading and irrelevant to the application, a user can report a post of

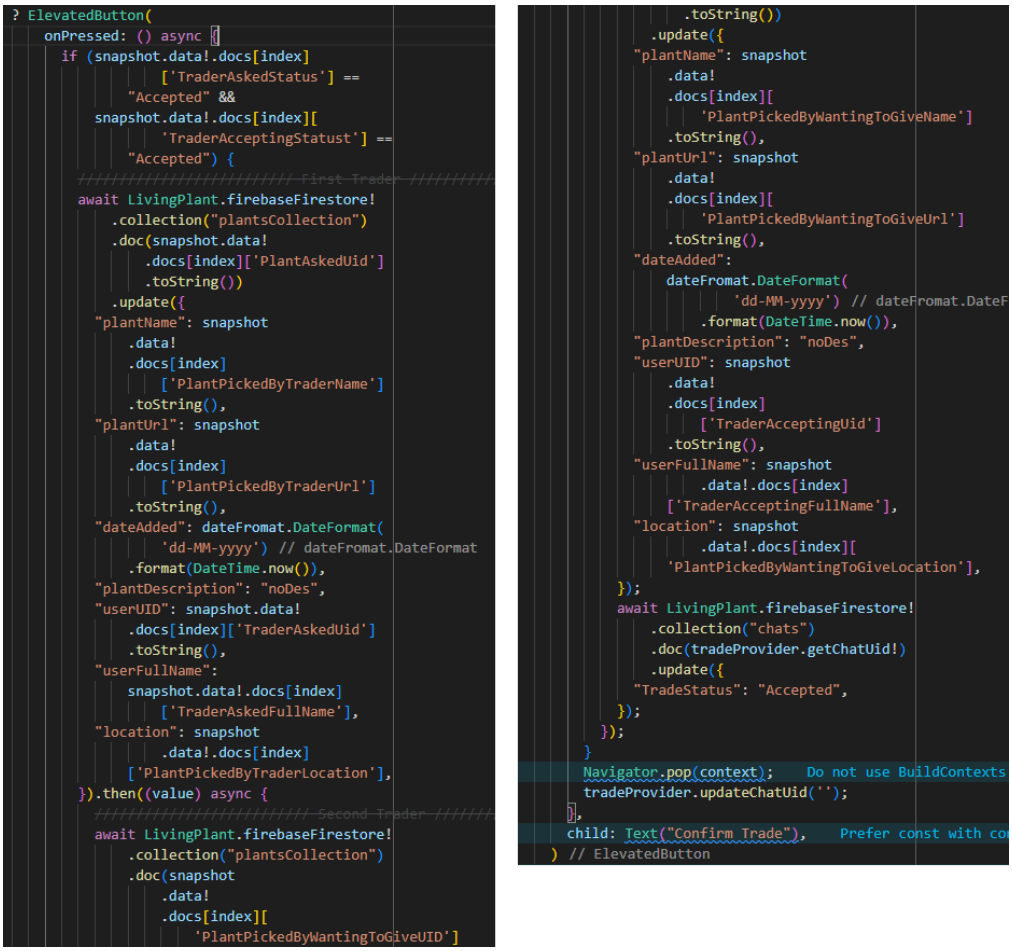
other users. There is a “report” button in every post, and if a user pushes that button, that post will be tagged as a reported post, and the Admin will check it.

Plant Barter

```
ElevatedButton(
  onPressed: () async {
    showDialog(
      context: context,
      builder: (_) => const loadingDialog(
          message: "Trading Plant, Please wait"), //
    );
    isSender(snapshot.data!.docs[index]
        ['TraderAskedUid'])
      ? await LivingPlant.firebaseFirestore!
        .collection("chats")
        .doc(tradeProvider.getChatUid)
        .update(
            {"TraderAskedStatus": "Accepted"})
      : await LivingPlant.firebaseFirestore!
        .collection("chats")
        .doc(tradeProvider.getChatUid)
        .update({
        "TraderAcceptingStatust": "Accepted"
        });
  },
  style: ElevatedButton.styleFrom(
    shape: const StadiumBorder(),
  ),
  child: const Text("Trade"),
), // ElevatedButton
```

Figure 6.9: Code Snippet of asking another user to barter a plant

Figure 6.9 shows how the barter system works. First, the user must offer a swap to another user and click the “trade,” then the other user, after checking if it’s okay for him/her to trade that plant, the other user must also click the ”trade” button for the trade to be successful.



```
? ElevatedButton(
    onPressed: () async {
      if (snapshot.data!.docs[index]
              ['TraderAskedStatus'] ==
          "Accepted" &&
      snapshot.data!.docs[index][
              'TraderAcceptingStatust'] ==
          "Accepted") {
      ////////////////////// First Trader ////////
      await LivingPlant.firebaseFirestore!
          .collection("plantsCollection")
          .doc(snapshot.data!
              .docs[index]['PlantAskedUid']
              .toString())
          .update({
        "plantName": snapshot
            .data!
            .docs[index]
                ['PlantPickedByTraderName']
            .toString(),
        "plantUrl": snapshot
            .data!
            .docs[index]
                ['PlantPickedByTraderUrl']
            .toString(),
        "dateAdded": dateFromat.DateFormat(
                'dd-MM-yyyy') // dateFromat.DateFormat
            .format(DateTime.now()),
        "plantDescription": "noDes",
        "userUID": snapshot.data!
            .docs[index]['TraderAskedUid']
            .toString(),
        "userFullName":
            snapshot.data!.docs[index]
                ['TraderAskedFullName'],
        "location": snapshot
                .data!.docs[index]
            ['PlantPickedByTraderLocation'],
      }).then((value) async {
        ////////////////////// Second Trader ///////
        await LivingPlant.firebaseFirestore!
            .collection("plantsCollection")
            .doc(snapshot
                .data!
                .docs[index][
                    'PlantPickedByWantingToGiveUID']
                .toString())
            .update({
          "plantName": snapshot
              .data!
              .docs[index][
                  'PlantPickedByWantingToGiveName']
              .toString(),
          "plantUrl": snapshot
              .data!
              .docs[index][
                  'PlantPickedByWantingToGiveUrl']
              .toString(),
          "dateAdded":
              dateFromat.DateFormat(
                      'dd-MM-yyyy') // dateFromat.DateF
                  .format(DateTime.now()),
          "plantDescription": "noDes",
          "userUID": snapshot
              .data!
              .docs[index]
                  ['TraderAcceptingUid']
              .toString(),
          "userFullName": snapshot
                  .data!.docs[index]
              ['TraderAcceptingFullName'],
          "location": snapshot
                  .data!.docs[index][
              'PlantPickedByWantingToGiveLocation'],
        });
        await LivingPlant.firebaseFirestore!
            .collection("chats")
            .doc(tradeProvider.getChatUid!)
            .update({
          "TradeStatus": "Accepted",
        });
      });
      }
      Navigator.pop(context);    Do not use BuildContexts
      tradeProvider.updateChatUid('');
    },
    child: Text("Confirm Trade"),    Prefer const with co
  ) // ElevatedButton
```

Figure 6.10: Code Snippet of Plant Barter Confirmed

Figure 6.10 shows the code when after two users agree to barter their plants, the system will swap, update and add the additional data in the database to reflect the trade and then put it in the trade history.

TESTING PROCESS

Development Testing

The developer's testing segment depicts that all the features of the application are working correctly.

Test Cases

TEST CASE 1: ADMIN

Test Module	Test Scenario	Expected Result	Actual Result	Status
View plant expert requests	Admin will tap the “experts request” tab.	It must display the requests of a user to be a plant expert.	Displays the requests of the user that wants to be a plant expert.	**PASSED**
Accept plant expert request	Admin will tap the “Accept proof” button.	The user whom the Admin accepts will become a plant expert.	The user whom the Admin accepts will become a plant expert.	**PASSED**
Decline plant expert request	Admin will tap the “Reject proof” button.	The request will be removed from the expert’s request.	The request was removed from the expert’s request.	**PASSED**
View the plant expert’s proof	Admin will tap the “Display” button.	The proof will be displayed.	The proof was displayed.	**PASSED**
Download the plant expert’s proof	Admin will tap the “Download” button.	The proof will be downloaded.	The proof was downloaded.	**PASSED**
View all users	Admin will tap the “All users” Tab.	It must display all the users that are registered.	Displays all the users that are registered.	**PASSED**
View reported posts	Admin will tap the “reported post” tab.	It must display all the reported posts.	Displays all the reported posts.	**PASSED**
Delete reported post	Admin will tap the “Delete post” button.	The post will be deleted from the database.	The post was deleted from the database.	**PASSED**
Change the plant identification API key	Admin will tap the “Change API” tab and input a new API key.	The API key of the plant identification API will be changed.	The API key of the plant identification API was changed.	**PASSED**

Table 16: TEST CASE 1: ADMIN

This testing process shows where the Admin views, accepts, and declines a request to be a plant expert, views and downloads the proofs, views all the users, views and deletes reported posts and changes the API key.

TEST CASE 2: PLANT EXPERTS

Test Module	Test Scenario	Expected Result	Actual Result	Status
Register	Users enter the information in the registration form.	Users cannot leave any form empty after inputting all the details.	Users cannot leave any form empty after inputting all the details.	**PASSED**
Acquire an expert's badge	Users that desire to be an expert will fill out the required fields.	Upon admin's approval, a check badge will appear beside the username of the new expert.	When the admin grants the user an expert role, the badge will appear beside the name.	**PASSED**
Logging In	Users will input their email and password.	Users will be logged in.	Users can use the application.	**PASSED**
Forgot Password	Users can change their password.	Users will be prompted an email from Firebase where they will change their passwords there.	Users will be prompted an email from Firebase where they will change their passwords there.	**PASSED**
Create Post	Users can post about anything about plants.	The post will be shown publicly on the homepage or in the feed.	The post will be shown publicly on the homepage or in the feed.	**PASSED**
Add comment	Users can comment on any post in the feed.	Comments are displayed and can be seen by other users.	Comments are displayed and can be seen by other users.	**PASSED**

Upvote a comment	Users can upvote a comment.	Users can upvote a comment from other users.	Users can upvote a comment from other users.	**PASSED**
Downvote a comment	Users can downvote a comment.	Users can downvote a comment from other users.	Users can downvote a comment from other users.	**PASSED**
Report post	If the post is not plant related, any users can report the post.	Admins can view, delete and verify the post that the user has reported.	Admins can view, delete and verify the post that the user has reported.	**PASSED**
Search	Users can search for plants.	When a keyword is entered in the search, the system will query the results and will be displayed.	When a keyword is entered in the search, the system will query the results and will be displayed.	**PASSED**
Plant Identification (Upload a photo from the gallery)	Users can upload a plant photo from their gallery.	The user clicks the “Identify” button, then the uploaded photo will be identified by the API and then displayed.	The uploaded photo will be identified by the API and then displayed.	**PASSED**
Plant Identification (Take a photo)	Users can take a photo of a plant from their camera.	The photo taken will be identified by the API they display.	The photo taken will be identified by the API they display.	**PASSED**
Save to collection	Once a plant is identified, users can choose to save that plant's details.	The plant details will be saved in the user's plant collection.	The plant details will be saved in the user's plant collection.	**PASSED**
Ask the people	If the user is not satisfied	When the “Ask the People” button is	When the “Ask the People”	**PASSED**

	with the plant identification result.	clicked, it will redirect to the create post section.	button is clicked, it will redirect to the create post section.	
View Profile	Viewing of user's information.	Users can view their account information, plant inventories, my posts, trade history, and about us.	Users can view their account information, plant inventories, my posts, trade history, and about us.	**PASSED**
Account Information	The user wants to view or edit his/her account information.	The user can view and edit his/her account information.	The user can view and edit his/her account information.	**PASSED**
My Plants	The user wants to view or delete his/her plant collection.	The user can view and can delete his/her plant.	The user can view and can delete his/her plant.	**PASSED**
My Posts	The user wants to view his/her recent post.	The user can view all his/her posts.	The user can view all his/her posts.	**PASSED**
Pending trades	When the user clicks the "Want this" button on someone's profile or results in the search query.	On the pending screen, the desired plant will be displayed, and the other user can browse a plant in return.	On the pending screen, the desired plant will be displayed, and the other user can browse a plant in return.	**PASSED**
Trade History	After successful trades between parties, the "trade" button has been clicked.	On the Trade history screen, The chat between parties will be recorded, and the trade details.	On the Trade history screen, The chat between parties will be recorded, and the trade details.	**PASSED**

Table 17: TEST CASE 2: Plant Experts

TEST CASE 3: NORMAL USERS

Test Module	Test Scenario	Expected Result	Actual Result	Status
Register	Users enter the information in the registration form.	Users cannot leave any form empty after inputting all the details.	Users cannot leave any form empty after inputting all the details.	**PASSED**
Logging In	Users will input their email and password.	Users will be logged in.	Users can use the application.	**PASSED**
Forgot Password	Users can change their password.	Users will be prompted an email from Firebase where they will change their passwords there.	Users will be prompted an email from Firebase where they will change their passwords there.	**PASSED**
Create Post	Users can post about anything about plants.	The post will be shown publicly on the homepage or in the feed.	The post will be shown publicly on the homepage or in the feed.	**PASSED**
Add comment	Users can comment on any post in the feed.	Comments are displayed and can be seen by other users.	Comments are displayed and can be seen by other users.	**PASSED**
Upvote a comment	Users can upvote a comment.	Users can upvote a comment from other users.	Users can upvote a comment from other users.	**PASSED**
Downvote a comment	Users can downvote a comment.	Users can downvote a comment from other users.	Users can downvote a comment from other users.	**PASSED**
Report post	If the post is not plant	Admins can view, delete and verify	Admins can view, delete and	**PASSED**

	related, any users can report the post.	the post that the user has reported.	verify the post that the user has reported.	
Search	Users can search for plants.	When a keyword is entered in the search, the system will query the results and will be displayed.	When a keyword is entered in the search, the system will query the results and will be displayed.	**PASSED**
Plant Identification (Upload a photo from the gallery)	Users can upload a plant photo from their gallery.	The user clicks the “Identify” button, and then the API will identify the uploaded photo and then displayed.	The API will identify the uploaded photo and then displayed.	**PASSED**
Plant Identification (Take a photo)	Users can take a photo of a plant from their camera.	The photo taken will be identified by the API they display.	The photo taken will be identified by the API they display.	**PASSED**
Save to collection	Once a plant is identified, the users can choose to save that plant's details.	The plant details will be saved in the user's plant collection.	The plant details will be saved in the user's plant collection.	**PASSED**
Ask the people	If the user is not satisfied with the plant identification result.	When the “Ask the People” button is clicked, it will redirect to the create post section.	When the “Ask the People” button is clicked, it will redirect to the create post section.	**PASSED**
View Profile	Viewing of user’s information.	Users can view their account information, plant inventories, my posts, trade history, and about us.	Users can view their account information, plant inventories, my posts, trade history, and	**PASSED**

			about us.	
Account Information	The user wants to view or edit his/her account information.	The user can view and edit his/her account information.	The user can view and edit his/her account information.	**PASSED**
My Plants	The user wants to view or delete his/her plant collection.	The user can view and can delete his/her plant.	The user can view and can delete his/her plant.	**PASSED**
My Posts	The user wants to view his/her recent post.	The user can view all his/her posts.	The user can view all his/her posts.	**PASSED**
Pending trades	When the user clicks the “Want this” button on someone’s profile or results in the search query.	On the pending screen, the desired plant will be displayed, and the other user can browse a plant in return.	On the pending screen, the desired plant will be displayed, and the other user can browse a plant in return.	**PASSED**
Trade History	After successful trades between parties, the “trade” button has been clicked.	On the Trade history screen, the chat between parties will be recorded, and the trade details.	On the Trade history screen, the chat between parties will be recorded, and the trade details.	**PASSED**

Table 18: TEST CASE 3: Normal Users

Usability Testing

This test is intended to identify and evaluate the user experience while utilizing the app in its completed form. This test includes all functions of the app, including implemented system features. The table below shows the satisfaction rating of 20 users who have used the application. The ratings used in the survey were as follows:

5–Strongly Agree, 4–Agree, 3–Neutral, 2 –Disagree, 1 –Strongly Disagree

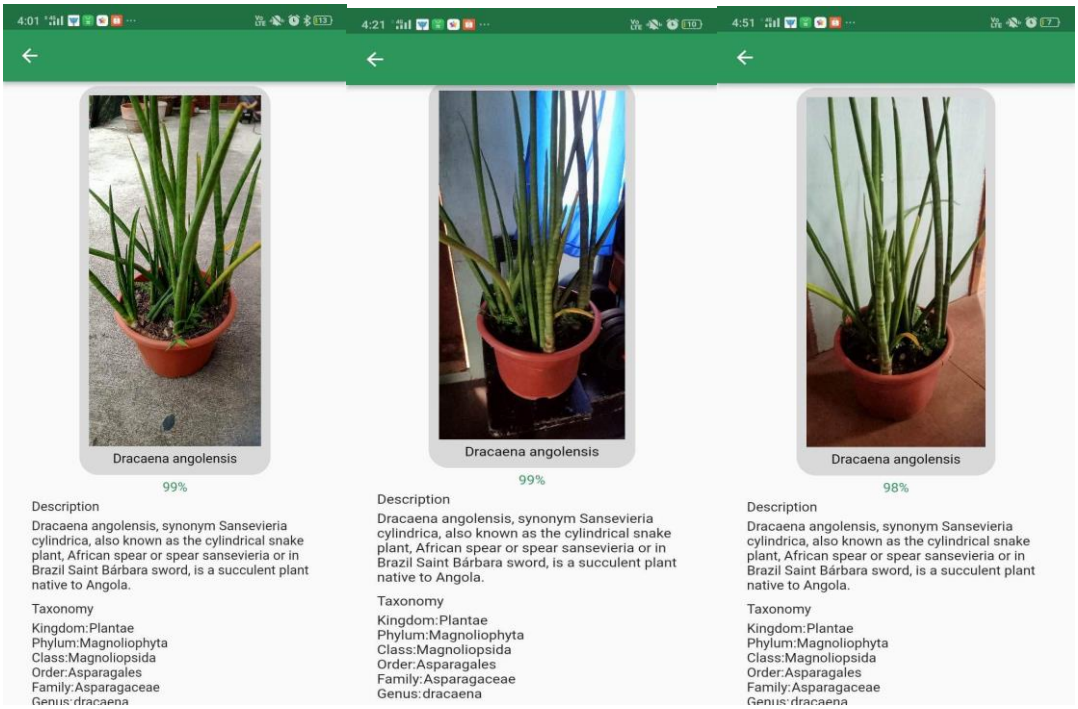
Questions							
	Strongly Agree	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**	**Total**	**Avg.**
The application doesn't crash.	14	4	1	1	0	20	**4.55**
It is easy to navigate between different pages.	13	5	1	1	0	20	**4.5**
The application's features work.	13	5	1	1	0	20	**4.5**
The application was not interrupted when using other apps and vice-versa.	13	5	1	1	0	20	**4.5**
The system gave error messages that clearly told me how to fix problems.	12	5	2	1	0	20	**4.4**

I was able to complete tasks and scenarios using this system.	14	4	1	1	0	20	**4.55**
It was easy to use this system.	13	5	1	1	0	20	**4.5**
The interface of this system is pleasant.	14	4	1	1	0	20	**4.55**
Would you likely recommend the LivingGreen application to a friend?	15	3	1	1	0	20	**4.6**

Table 19: Usability Testing

Performance and Network Testing

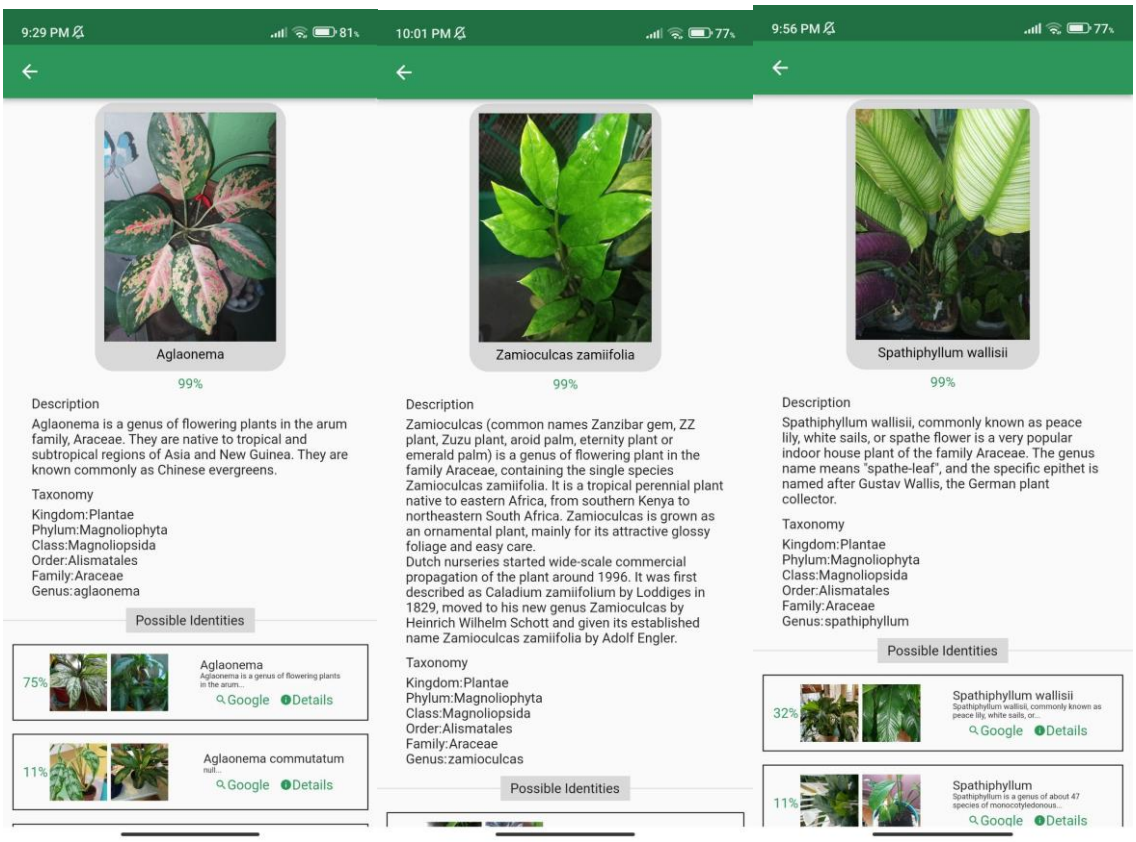
These tests evaluate the speed, responsiveness, network bandwidth, and latency. These tests aim to identify potential network-related issues and performance that could impact the performance of the system and the user experience.



Take a photo

	Test 1	Test 2	Test 3	Average Accuracy
Plant 1	99%	99%	98%	98.66%
Plant 2	99%	99%	99%	99%
Plant 3	95%	96%	99%	96.66%
Plant 4	99%	99%	99%	99%
Plant 5	99%	99%	99%	99%
			Average	**98.46%**

Table 20: Take a photo



Upload a photo

	Test 1	Test 2	Test 3	Average Accuracy
Plant 1	99%	99%	99%	99%
Plant 2	99%	99%	99%	99%
Plant 3	99%	99%	99%	99%
Plant 4	99%	99%	99%	99%
Plant 5	99%	99%	99%	99%
			Average	**99%**

Table 21: Upload a photo

The researchers conduct a performance task using the “upload a photo” and “take a photo” on our devices with different constraints; angles and lighting. To determine the plant identification accuracy, the researchers took five pictures for “take a photo” and then calculated its average accuracy. After, another five pictures for “upload a photo,” then calculated its average accuracy.

Network Response Bandwidth (Globe WiFi Telecom - ABUNO PAJAC, LLC)

	Response Time
Test 1	1m 18s 88
Test 2	51s 06
Test 3	1m 15s 03
Test 4	1m 08s 62
Test 5*	1m 32s 27
Average Response	**1m 12s 84**

Table 22: Network Response Bandwidth(Globe WiFi)

Network Response Bandwidth (PLDT WiFi Telecom - USJR BASAK)

	Response Time
Test 1	2m 51s
Test 2	1m 47s
Test 3	3m 21s
Test 4	2m 02s
Test 5*	1m 25s

Average Response	**2m 17s 00**

Table 23: Network Response Bandwidth(PLDT WiFi)

Network Response Bandwidth (Converge WiFi Telecom - Kinasang-an, Pardo)

	Response Time
Test 1	7s 06
Test 2	18s 57
Test 3	20s 48
Test 4	16s 18
Test 5	11s 45
Average Response	**14s 43**

Table 24: Network Response Bandwidth(Converge WiFi)

Network Response Bandwidth (Globe Mobile Data - ABUNO PAJAC, LLC)

	Response Time
Test 1	11 sec
Test 2	17 sec
Test 3	21 sec
Test 4	41 sec
Test 5	1m 15s
Average Response	**33 secs**

Table 25: Network Response Bandwidth(Globe Mobile Data)

Network Response Bandwidth (SUN Mobile Data - USJR BASAK CAMPUS)

	Response Time
Test 1	2m 51s
Test 2	1m 47s
Test 3	3m 21s
Test 4	2m 02s
Test 5	1m 25s
Average Response	**2m 17s**

Table 26: Network Response Bandwidth(SUN Mobile Data)

The researchers conduct a network response bandwidth task for WiFi and Mobile Data on our devices. To determine the response time, the plant identification can display the information of the identified plant.

Summary of Testing

Upon undergoing the testing, the researchers' project objectives were achieved. The application's features were evaluated and passed the testing phase. The survey result shows that each statement's average rating is above 4.0. This result indicates that the testers are satisfied with their app experience. The results also show that the relevant features of the app are working correctly. To attain a positive result, the researchers evaluated using different test cases. With all these, the system requirements were fully realized. After analyzing the network-based performance results of the main modes of connection, namely through WiFi and mobile data, the outcome is considerably less efficient, for it averages over a minute of response time. Several constraints should be

considered, such as the user's location or the client's distance from the nearest service provider. Another limitation observed during the testing of the application is when the application is connected under a public access point or any connection with a large volume of clients being handled at a given time. One example of this scenario is when the application was connected to the university's network.

CHAPTER IV

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

This chapter describes the issues and conclusions encountered by the creators during its development, which addresses hardships and how project plans and methods were adjusted.

SUMMARY OF FINDINGS

LivingGreen, a plant identification mobile application, utilizes a plant identification API to accurately identify plants based on images submitted by users via a mobile device camera. The application also includes a social media-like feature, allowing users to share information and connect with others interested in plants, as well as to share pictures of their plants and ask for identification help from other users. Additionally, the application includes a trading system, allowing users to trade plants with one another, using the application as a trading place. The findings suggest that combining these features provides a comprehensive and user-friendly experience for those interested in identifying and learning about plants and those looking to trade them, making it very convenient for people who love gardening and want to expand their plant collection.

CONCLUSION

The study has achieved its goal of providing a communicative and informative platform as a linkage between plant experts and plant enthusiasts where they can make a post, make a comment, vote a comment, use the plant identification feature, chat, and trade. Researchers utilize the plant.id API for the database of plants and plant identification. The application's backend and database are Firebase.

RECOMMENDATIONS

To further improve LivingGreen, future researchers should add or integrate more features like plant disease identification, plant health assessment, and plant condition recognition to help plant enthusiasts and experts with their plants. A prominent feature, like a marketplace for plants, can help improve the plant community.

To also improve the performance of the entire application, future researchers are encouraged to use another state management with a faster performance like GetX. As compared to other different state management libraries, GetX is a better state management, and it is because GetX just consumes minimum resources and it also provides better performance. Productivity: GetX's syntax is way easier, so it is productive. It saves the developers a massive amount of time and increases the application's speed because it does not use other extra resources. The researchers can also integrate machine learning to help the users if a marketplace feature is soon available.

CHAPTER V

BIBLIOGRAPHY

[1] Flutter SDK release notes | Flutter 3.0.0 release notes. Retrieved from: https://docs.flutter.dev/development/tools/sdk/release-notes/release-notes-3.0.0

[2] Postman Tutorial for Beginners to Perform API Testing - Encora. Retrieved from https://www.encora.com/insights/what-is-postman-api-test

[3] Provider State Management in Flutter | by Maaz Aftab - Medium. Retrieved from https://medium.com/codechai/provider-state-management-in-flutter-d453e73537c5

[4] What is the purpose of an upvote/downvote system and why is it considered preferable to having only upvotes? Retrieved from https://www.reddit.com/r/TheoryOfReddit/comments/czej4i/what_is_the_purpose_of_an_upvotedownvote_system/

[5] What is Google Lens? What are the ways of Google Lens save your time. Retrieved from https://www.computerworld.com/article/3572639/google-lens-android.html

[6] andrew fulmer. 2021. What is Flutter in a Nutshell? - Surf. Surf. Retrieved from https://surf.dev/what-you-should-know-about-flutter/

[7] The Benefits of Having Firebase for Mobile App Development. Retrieved from https://www.mindinventory.com/blog/benefits-of-firebase-in-mobile-app-development/

[8] TapTapsee

https://taptapseeapp.com/

[9] LogoGrab

https://www.tekrevol.com/blogs/best-image-recognition-apps/

[10] How to Get User's Current Location Address in Flutter — Geolocator & Geocoding | How to Get User's Current Location Address in Flutter - Medium. Retrieved from https://medium.com/@fernnandoptr/how-to-get-users-current-location-address-in-flutter-geolocator-geocoding-be563ad6f66a

[11] Plant Identification: Is It Worth the Effort? Retrieved from https://www.noble.org/news/publications/ag-news-and-views/2001/may/plant-identification-is-it-worth-the-effort/#:~:text=The%20ability%20to%20know%2C%20or,trend%2C%20either%20upward%20or%20downward.

[12] Plant.id - Plant identification app https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwibxoGmrd38AhVRIIgKHTRyBDAQFnoECA8QAQ&url=https%3A%2F%2Fplant.id%2F&usg=AOvVaw3qGkxu1LfBypHhOWWrixtP

[13] Gaël Thomas. 2019. What is Flutter and Why You Should Learn it in 2020. freeCodeCamp.org. Retrieved from https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/

[14] Visual Studio 2022 version 17.2 Release Notes | Microsoft Docs. Visual Studio 2022 version 17.2 Release Notes | Microsoft Docs. Retrieved from https://docs.microsoft.com/en-us/visualstudio/releases/2022/release-notes

[15] Cynthia Harvey. 2021. 11 Best Android IDEs for Developers of 2022 | Developer.com. Developer.com. Retrieved from https://www.developer.com/mobile/top-android-ides-fordevelopers/#:~:text=Android%20Studio%3A%20Google's%20official%20IDE, IDE%20for%20Java%20desktop%20apps!

[16] Fetch data from the internet - Flutter documentation. Retrieved from https://docs.flutter.dev/cookbook/networking/fetch-data

[17] Using SharedPreferences in Flutter | by Ashish Rawat. Retrieved from https://medium.flutterdevs.com/using-sharedpreferences-in-flutter-251755f07127

[18] How To Get a User's Location with the Geolocator Plugin in Flutter. Retrieved from https://www.digitalocean.com/community/tutorials/flutter-geolocator-plugin

[19] Charles Hall, Melinda Knuth; An Update of the Literature Supporting the Well-Being Benefits of Plants: A Review of the Emotional and Mental Health Benefits of Plants. *Journal of Environmental Horticulture* 1 March 2019; 37 (1): 30–38. doi: https://doi.org/10.24266/0738-2898-37.1.30

[20] *Facebook Marketplace Review: Easy, Convenience & Fast Selling*. The Best Singapore. (n.d.). Retrieved January 26, 2023, Retrieved from https://www.thebestsingapore.com/best-service/facebook-marketplace-review/

Curriculum Vitae

PERSONAL INFORMATION

Name : **Rasmil Lloyd P. Augusto**
Nickname : **Ras/Fluvia**
Date of Birth : **December 13, 1998**
Address : **Abuno, Pajac, Lapu-Lapu City, Cebu**
Status : **Single**
Citizenship : **Filipino**
Religion : **Roman Catholic**
Email Address : **augustorasmil@gmail.com**
Contact Number : **09479722174**

ACADEMIC INFORMATION

Tertiary : **University of San Jose Recoletos**
Secondary : **Saint Alphonsus Catholic School**
Primary : **Saint Alphonsus Catholic School**

PROFESSIONAL SKILLS

Language : **C, Java, HTML, CSS, JavaScript, Python, Dart, Flutter, Php, Angular, Typescript, MySQL**
Software : **Microsoft Office, Visual Studio, VSCode, Netbeans, Android Studio**
OS : **Windows**

PERSONAL INFORMATION

Name: **Ronel John S. Tano**
Nickname: **Nel/Tan**
Date of Birth: **November 7, 1998**
Address: **Deca Homes Mactan 1, Ibabao-Gis Agus Rd, Lapu Lapu City**
Status: **Single**
Citizenship: **Filipino**
Religion: **Roman Catholic**
Email Address: **ronel.tano98@gmail.com**
Contact Number: **09322712083**

ACADEMIC INFORMATION

Tertiary: **University of San Jose Recoletos**
Secondary: **Saint Alphonsus Catholic School**
Primary: **Saint Alphonsus Catholic School**

PROFESSIONAL SKILLS

Language: **C, Java, HTML, CSS, JavaScript, Dart, R, MySQL, TypeScript.**
Software: **Microsoft Office, Visual Studio, VSCode, Netbeans, Android Studio**
OS: **Windows**

PERSONAL INFORMATION

Name: **Christian D. Lastimosa**
Nickname: **Chan**
Date of Birth: **September 23, 1999**
Address: **24-35, Dawis Street, Purok Narra, San Roque, Talisay City, Cebu, 6045**
Status: **Single**
Citizenship: **Filipino**
Religion: **Roman Catholic**
Email Address: **chrislastimosa34@gmail.com**
Contact Number: **09166645842**

ACADEMIC INFORMATION

Tertiary: **University of San Jose Recoletos**
Secondary: **Talisay City National High School**
Primary: **Tabunoc Central School**

PROFESSIONAL SKILLS

Language: **C, Java, HTML, CSS, JavaScript, R, Dart, React Native, AngularJS, MySQL, TypeScript.**
Software: **Microsoft Office, Visual Studio, VSCode, Netbeans, Android Studio**
OS: **Windows**

PERSONAL INFORMATION

Name: **Mervyn B. Morales**
Nickname: **Benot/Ben**
Date of Birth: **April 10, 1999**
Address: **Sabellano Street, Lower Suran Quiot Pardo, Cebu**
Status: **Single**
Citizenship: **Filipino**
Religion: **Roman Catholic**
Email Address: **mervynmorales10@gmail.com**
Contact Number: **09991891404**

ACADEMIC INFORMATION

Tertiary: **University of San Jose Recoletos**
Secondary: **Academia De Nuestra Señora**
Primary: **Sta. Cruz Elementary School**

PROFESSIONAL SKILLS

Language: **C, Java, HTML, CSS, Php, R, MySQL, AngularJS, Dart**
Software: **Microsoft Office, Visual Studio, VSCode, Netbeans, Android Studio**
OS: **Windows**