

**Project LYF: Development of Mobile Application for Plant Recognition and Location Documentation**

Project Documentation to be submitted

to the Faculty of School of

Computing and Information Technology

under Asia Pacific College

In partial fulfillment of the Requirements for the Subject

Structure Systems Analysis and Design (SYSADD1)

By:

|  |  |
| --- | --- |
| **Jamie Therese Y. Gahallon** | **John David L. Solomon** |
| Project Manager, System Analyst and Documenter | System Analyst, Project Researcher and  Documenter |

|  |  |
| --- | --- |
| Ms. Marielet Guillermo | Mr. Manuel Sebastian S. Sanchez |
| Project Adviser | Subject Professor |

April 23, 2019

Table of Contents

Executive Summary6

Introduction7

Project Context7

Purpose and Description7

Objectives8

Scope and Limitations9 Scope9 Limitations9

Review of Related Literature11

PlantNet11

Garden Answers11

PlantFinder11

PlantSnap11

Technical Background12

Relevant Theory and Concepts12

Image Recognition12

Approach to Image Recognition12

Neural Network Architecture13

Convolutional Network Architecture14

Tools and Utilities17

TensorFlow17

Google Maps API18

Methodology19

Requirements Analysis19

Event Table19

Requirements Documentation21

Use Case Full Description21

User Registers21

Login22

Captures Plant Image23

Process Image24

Create Unidentified Plant Report25

Create Incorrect Location Report26

Create Add/Edit Information Request27

Validate Report and Request28

Edits Plant Glossary29

Generate System Reports30

Gap Analysis/Needs Assessment31

Design of Software, Systems, Product and Processess32

Android32

Web45

Development51

Appendices52

Use Case Diagram52

Context Diagram53

Data Flow Diagram Level 054

Entity Relationship Diagram55

Data Dictionary56

Activity Diagram60

Login/Register60

Process Image62

Add Information63

Edit Information64

Browse Location65

Admin Add Information66

Admin Edit Information67

System Sequence Diagram68

Process Image68

User Add/Edit Information69

Report Incorrect Location70

Verify Incorrect Location71

Admin Verify Requests72 Admin Add/Edit Information73

Communication Diagram74

Location and Results74

Incorrect Location75

Plant76

Plant Match Results77

Object Diagram78

Class Diagram79

Timing Diagram81

Package Diagram82

Component Diagram83

State Machine Diagram84

Account Object84

Plant Object84

Plant Match Result Object85

Deployment Diagram86

Interaction Diagram87

Process Image87

Add/Edit Information88

Report Incorrect Location89

Survey Results90

Survey Questions96

**Executive Summary**

Philippine Institute of Traditional and Alternative Health Care (PITAHC) is a government branch created under the Department of Health. Their vision is described as “People’s health through traditional and alternative health care” which states their desire to inject traditional and alternative medicines to the conventional health care system in the country. PITAHC aims to promote and advocate the use of traditional, alternative, preventive, and curative health care modalities that have been proven safe, effective, cost effective and consistent with government standards on medical practice.

According to an interview with PITAHC's researcher, it is difficult for PITAHC to advocate the use of medicinal plants Filipinos find difficult to recognize. Plants are officially identified by the botanists from Bureau of Plant Industry through observation of its physical characteristics. It is also difficult for them to gather necessary information such as plant abundance and location. To accomplish this task, PITAHC sends representatives to every location and manually keep track of medicinal plant population and their locations.

The project aims to support PITAHC in their mandated functions by creating image classifier that could help users identify medicinal plants through image capture of its leaf and shares location to other users upon recognition. Also, the project will aid promotion by providing results of their researches, which includes traditional medicine and their clinically approved usage, to users.

**I. Introduction**

**1.1 Project Context**

The government created Philippine Institute of Traditional and Alternative Health Care as mandated by the Republic Act 8423 “to improve the quality and delivery of health care services to the Filipino people through the development of traditional and alternative health care and its integration into the national health care delivery system”. Philippine Institute of Traditional and Alternative Health Care, or PITAHC, is working under the Department of Health towards this mandated goal. PITAHC envisions itself to lead research, development, promotion and development of standards on traditional and complementary medicines to ensure its accessibility, availability, sustainability and integration into the national health care system.

The project members have interviewed Ms. Ma. Teresa M. Torres, a Science Research Specialist II for PITAHC. According to her, it is difficult for PITAHC to advocate use of medicinal plants because some Filipinos can’t recognize them. During the interview, Ms. Torres has explained various physical characteristics of the plants that the general public more often misidentify or how some of these beneficial plants are treated like weeds despite its medicinal benefits. Ms. Torres also mentions the danger of wrongly classifying a medicinal plant, and consuming poisonous or dangerous plants.

The researchers conducted a survey wherein respondents are presented pictures of the 10 common medicinal plants approved by the Department of Health and asked to choose its corresponding name. Well-known plants such as the Lagundi and Bayabas receives high rating of correct answers, both 34%, but other common plants on the list receives low rating. This indicates that there is a difficulty in identifying these plants.

PITAHC also performs regular surveying in various locations at the Philippines to manually keep track of traditional and complementary medicinal plant abundance. They perform this by sending representatives to these areas who manually search and survey the land. In addition, PITAHC has performed numerous studies proving their effectivity and safety but this information hasn’t reached the general public.

These problems cripple the operation of PITAHC. The team addresses these problems in support of PITAHC towards their mandated function through Lyf.

**1.2 Purpose and Description**

Philippine Institute of Traditional and Alternative Healthcare, or PITAHC, is facing problems on regarding their advocacy and promotion of traditional and contemporary medicine because of Filipinos’ difficulty in recognizing and identifying medicinal plants.

To address these, the team developed Lyf. Lyf is a system that includes an Android application for users and website for the admin. The android application allows users to register and login. Once account has been created, the user can now access the application’s functions. These includes image recognition trained on the Department of Health’s recommended medicinal plants. The feature helps users to identify plants and explore their medicinal benefits. Users will have to take a photo of its leaf for the system to recognize. Users can also contribute information through the app.

Ms. Ma. Teresa M. Torres, a Science Research Specialist II for PITAHC, has provided the team with the results of their studies to include in the system. Once identified by the app, the location is shared among all users which can be seen through map. Using this function, users can view all shared plant location, or filter the results through plant search. Upon viewing a certain plant location, the user can also view that plant’s details.

The app also contains a plant glossary which users can browse through to access all plant details derived from the studies.

The system’s website allows admin to respond to incorrect location reports and requests to add/edit information. These changes are visible to the users through the android app. The admin can also view all results of image recognition.

**1.3 Objectives**

The project is intended to address the problems PITAHC has encountered in performing their mandate and functions. These functions are associated with ensuring accessibility, encouraging research and information sharing, promotion of use of traditional and contemporary medicinal plant. It has been difficult for PITAHC to achieve their mission because of problems on the department’s implementation, such as manual gathering of researches and location surveys, information dissemination and plant identification on the user’s end.

General Objectives

* To create a system that will allow identification, information exchange, and location sharing of traditional and contemporary medicinal plant and generate reports for the use of PITAHC.

Specific Objectives

* To classify plant by its leaf using an image classifier developed on TensorFlow
* To create a collection of data on medicinal plants and its benefits
* To gather data on location and abundance of plants

**1.4 Scope and Limitations**

**Scope**

* The Admin can:
  + Create an account by providing:
    - Name
    - Username
    - Password
    - Email
    - Position
  + Access the Lyf Web Admin Panel
  + Assign Admin role to registered account.
  + Edit account.
  + View all result of image recognition, both successful and failed results.
  + Address and remove plant location from Maps.
  + View all plant information shown in plant glossary.
  + Add/Edit plant information in plant glossary.
  + View all requests for add and edit plant information.
  + Change request status to Verified.
  + Delete requests.
* The Users can
  + Create an account by providing:
    - Name
    - Username
    - Password
    - Email
  + Access the Lyf App.
  + Edit account.
  + View all locations of successful image recognitions through Maps.
  + Report incorrect location of plants in Maps.
  + Identify plants by isolating the leaf and capturing an image of it.
  + View the plant glossary and information verified by the admin.
  + Create requests to add information/plant in the plant glossary.
  + Create requests to edit information in the plant glossary.

**Limitations**

* The Android app is developed using SDK for Android 8.1.
* The image quality is dependent on the camera specifications of the device and its effect on the result of image recognition is not considered.
* Researchers used TensorFlow for image recognition and training.
* Training model are limited to classification of *Bayabas, Pansit-pansitan.*
* The Lyf Web Admin Panel is the responsibility of the client, PITAHC. Admin roles are granted by PITAHC.
* System can only recognized plants which has undergone match training.
* Plants can only be recognized by capturing an image of its leaf.

**II. Review of Related Literature/Systems**

This section discusses other projects related to Lyf. The related projects identified may not have the same implementation of Lyf but are guided by the same concept and background.

**2.1 PlantNet**

PlantNet Plant Idenfication. Retrieved from <https://www.educationalappstore.com/app/plantnet-plant-identification>. This is an image sharing and retrieval application for plants. PlantNet identifies plant through image recognition software. The species and images used to identify plants evolve as contributions from end users increases. The app provides the scientific and English name upon plant recognition.

**2.2 Garden Answers**

Garden Answers Plant Identification. Retrieved from <http://www.gardenanswers.com>. Garden Answers identifies plant through image recognition software. Upon identification, the app provides information on the plant provided by garden and horticulture experts. It also contains a database to find out more information about a plant without prior identification.

**2.3 PlantFinder**

PlantFinder – Plant Identifier. Retrieved from <https://itunes.apple.com/us/app/plantfinder-plant-identifier/id1437376141?mt=8>. Main features of PlantFinder includes plant identification with the camera and access to their flora database. Information provided on the app includes basic information such as scientific and common name, and plant care. This app focuses on plant identification and providing plant care suited for the specific plant.

**2.4 PlantSnap**

PlantSnap. Retrieved from <https://www.plantsnap.com/>. PlantSnap also identifies plant through image recognition. Information provided by PlantSnap focuses more on plant’s classification taxonomy. Upon recognition, PlantSnap also retrieve the location which other users can also see through a map. Users of the free version are limited to a number of ‘snaps’ or image identification per day.

**III. Technical Background**

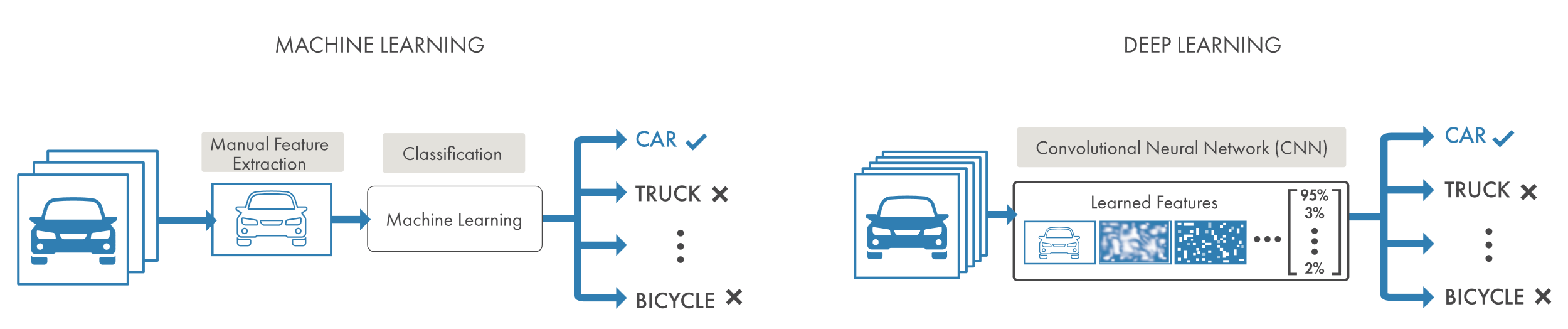
**3.1 Relevant Theory and Concepts**

**IMAGE RECOGNITION**

Image Recognition is a term used for technologies that can detect and recognize subjects such as people, animals or objects through algorithms and machine learning. [1]Image recognition is a subset of computer vision, referring to the ability of a computer to “see”, to decipher, and understand an image. To teach computers to process visual data, computers must be taught how to recognize pattern first. [2]

Typical image recognition algorithms include optical character recognition, pattern matching and gradient matching, face recognition, scene identification and scene change detection.

Figure 1 Comparing Machine Learning and Deep Learning

**APPROACH TO IMAGE RECOGNITION**

***Machine Learning***

Image recognition through machine learning means identifying and extracting key features from images and using them as input to a machine learning model. [3] To simplify the machine learning approach, it takes in data, pushes it through algorithms, and make a prediction, giving the impression that a computer “thinks” and arrives in a conclusion.

***Deep Learning***

Another approach is through the deep learning method. Deep learning is concerned with algorithms inspired by brains neural networks. It is a technique that teaches computers to do what comes naturally to humans: learn by example. A computer model learns classification tasks directly from images. These models are trained using a large set of labeled data and neural network architectures that layers as many as 150.

Most deep learning methods uses neural network architectures. Models are trained used large set of labeled data and features are learned directly from data without manual feature extraction. [4]

Computers learn from image by extraction features such as edges, texture, and color to form a reduced representation that easily describes an object. These are used by a machine learning classifier to “learn” a task.

Deep learning has surged over the recent years because of its improving accuracy, availability of GPUs to train deep networks faster, and large amounts of labeled data.

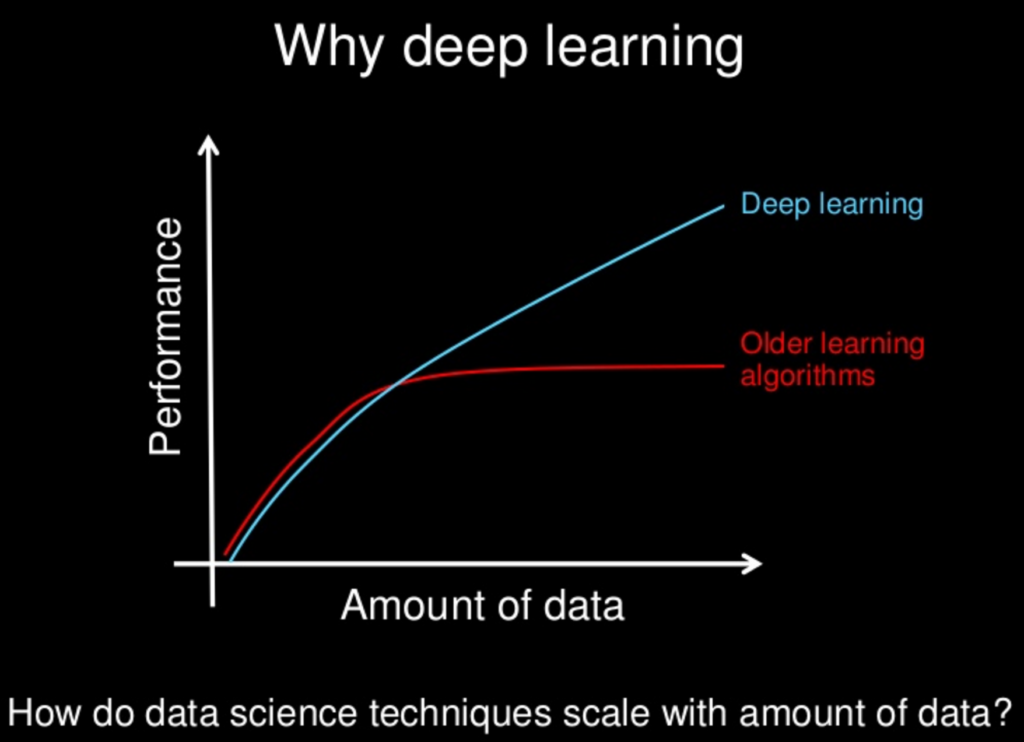


Figure 2 Comparison of deep learning

**NEURAL NETWORK ARCHITECTURE**

Neural networks are algorithms, loosely based on the human brain, designed to recognize patterns. Data are interpreted through labeling or clustering raw input. The patterns recognized are numerical, contained in vectors. Neural networks helped to group unlabeled data based on similarities on the example inputs and classify the data when a there is a labeled dataset.

Layers in the networks are composed of nodes. A node is where computation happens. It combines input from the data with a set of coefficient or *weights*, which assigns significance to the input. This input combined with weights are summed. They are passed on the activation function, which determine if the signal should progress further to the network.

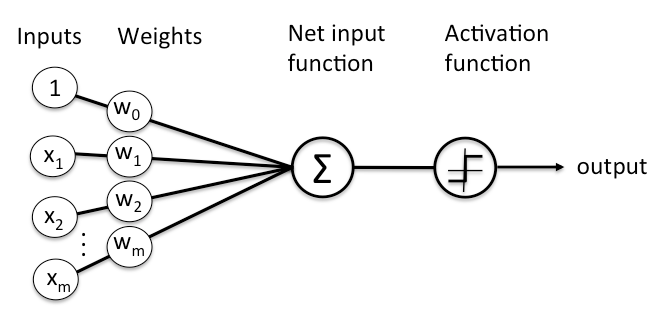


Figure 3 Node Representation

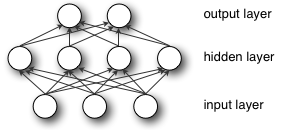


Figure 4 Layer Representation

Depth refers to the number of node layers the data must pass in a multistep process of pattern recognition. More than 3 layers is needed to be classified as “deep learning”. Each layer trains on a distinct feature based on the previous layer’s output. Thus, the further the data advances, the more complex the features the nodes can recognize. [5]

**Convolutional Neural Network**

Convolutional neural networks, or CNN, are primarily used to classify images, cluster by similarity, and perform recognition within scenes. These networks process images as tensor. Tensors are matrices of numbers with dimensions. [6]

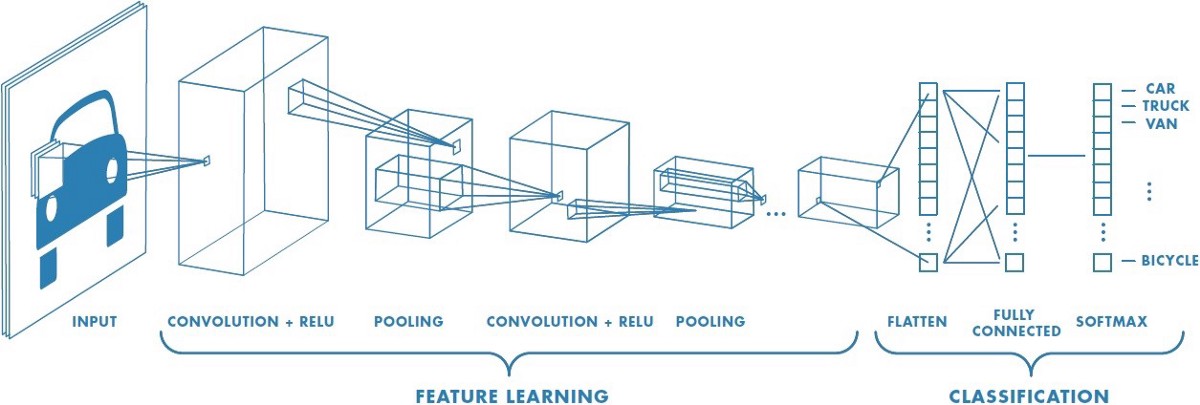


Figure 5 Convolutional Neural Network

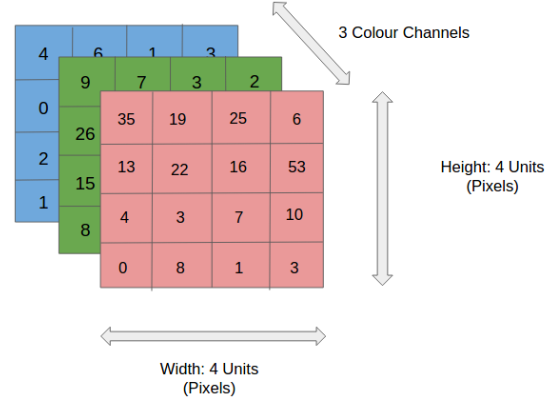
Using an RGB image as an example, it can be separated by its three color panes – Red, Green and Blue. CNN reduces the images into forms easier to process, without losing critical features.

Figure 6 4x4x3 Image

A close up of a computer

Description automatically generated

*Figure 7 Convolution operation on a 4x4x3 image matrix with a 2x2 kernel with stride length of 1*

The CNN layers consist of learnable filters or kernels. During the forward pass, the filter slides across the with and height of the input volume, computing dot products between the filter and input. An activation map is produced, which gives the responses of the filter at every spatial position. All results are summed with the bias to give a squashed one-depth channel Convoluted Feature Output.

The pooling layer is responsible for reducing the spatial size of the convolved feature, decreasing the computation power required to process the data. It extracts dominant features.



Figure 8 Pooling Layer

The image is converted to a suitable form for Multi-Level Pereceptron. Image is flattened into a column vector fed to a feed-forward neural network. Over time, the model can distinguish between dominating and certain low-level features and classify them using the Softmax Classification technique. [7]

**3.2 Tools and Utilities**

**TENSORFLOW**

TensorFlow is an open source library created by the Google Brain Team released in 2015. It is used for numerical computation and large-scale machine learning. It can train and run deep neural networks for applications such as handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and partial differential equation-based simulations.

It allows developers to create dataflow graphs that describes the movement of data using a graph or a series of processing nodes. Each node represents a mathematical operation and each connection is a tensor.

TensorFlow provides abstraction to developers. Instead of dealing with the details of implementing algorithms, developers can focus on the overall logic of the application while it deals with the details behind the scene.

**GOOGLE MAPS API**

On today's Web and applications, mapping solutions are a dependency. We use them to see the location of things, to search for the position of an address, to get driving directions, and to do numerous other things. Most information has a location, and if something has a location, it can be displayed on a map.

There are several mapping solutions including Yahoo! Maps and Bing Maps, but the most popular one is Google Maps. In fact, according to Programmableweb.com, it's the most popular API on the Internet.

Applications, mobile applications and web sites that are combining data or functionality from two or more sources are commonly referred to as mashups. Mashups are becoming increasingly popular and have revolutionized the way information is being used and visualized.

Mapping solutions are one important ingredient in a lot of these mashups. The Google Maps API lets you harness the power of Google Maps to use in your own applications to display user’s data (or others') data in an efficient and usable manner.

Google Map’s API is a powerful tool that can be used to create a custom map, a searchable map, check-in functions, display live data synching with location, plan routes, or create a mashup just to name a few. An even more powerful is the mashup of more than one API. For example, one cool mashup idea blends Google Maps and Uber bookings and pickups.

Google Places API allows web and mobile app developers to do customize their search by:

(i) apply a keyword search and/or filters;

(ii) rank by distance and popularity.

**IV. Methodology, Results and Discussion**

**4.1 Requirements Analysis**

**4.1.1 Event Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event** | **Trigger** | **Source** | **Use Case** | **Response** | **Destination** |
| A registration form will be shown if the user hasn’t created an account for the app yet (or wants to create a new account for the app) | User registration | User | User Registers | Display Register form | User/System |
| User has photographed a plant for identification | User wants to identify a certain plant | User | Captures Plant Image | System prepares image for identification | System |
| System compares image with training data set | System receives photo | System | Process Image | System displays identified plant match | User |
| System creates report if plant match is not found | Plant cannot be identified by system | System | Create Unidentified Plant report | System generates Unidentified Plant report | User/System |
| User reports incorrect/missing location | Report from user | User | Create Incorrect Location report | System generates Incorrect Location report | User/System |
| User wants to add or edit plant information in the plant glossary | User adds/edits plant information in the plant glossary | User | Create Add/Edit Plant Information Request | System generates Add/Edit Plant Information Request for the admin | Admin/System |
| Admin responds to requests and reports | Requests and reports from user | Admin | Validate Report and Request | Admin validates reports and requests | User/System |
| Admin updates plant glossary | Admin updates plant details in the system | Admin | Update Plant Glossary | System updates plant glossary | System |
| System generates reports | Admin requests for System Reports | Admin | Generate System Reports | System displays reports | Admin/System |

Figure 9 Event Table

# **4.2 Requirements Documentation**

**4.2.1 Use Case Full Description**

**4.2.1.1 User Registers**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | User Registers | |
| Scenario: | A registration form will be shown if the user hasn’t created an account for the app yet (or wants to create a new account for the app) | |
| Triggering Event: | User registration | |
| Brief Description: | If the user hasn’t created an account yet or wants to create a new one, then the user will be directed to the registration screen | |
| Actor/s | * User * System | |
| Related Use Case: | * Login | |
| Stakeholders: | * User * System | |
| Preconditions: | * User must have internet access * User must give permission to internet access on their devices | |
| Post Conditions: | * User account created | |
| Flow of Activities: | User | System |
| 1.0 User selects Register  2.0 User fills out the registration form | 1.1 Directs to Registration  2.1 System creates user account |
| Exception Conditions: | 1. User must fill out all required fields of the registration for,   2.1 User must enter an email and username that doesn’t belong to other registered user. | |

Figure 10 Use Case Full Description - User Registers

**4.2.1.2 Login**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Login | |
| Scenario: | Upon launching the app, the first screen the user will see is the login form | |
| Triggering Event: | App launched | |
| Brief Description: | Launching the app will directly go to the Login screen | |
| Actor/s | * Users * System | |
| Related Use Case: | - | |
| Stakeholders: | * Users * System | |
| Preconditions: | * User must have internet access * User must have an existing Lyf account * User must give permission to internet access on their devices | |
| Post Conditions: | * User is logged in the app * User can access app functions | |
| Flow of Activities: | User | System |
| 1. Launched the app 2. User enters account credentials 3. User logged in | * 1. Directs to Login Screen   2. System verifies credentials   3.1 Display Lyf Home |
| Exception Conditions: | 1. App is not launched.   1.1 No account/create new account  2.0 User entered incorrect credentials. User is redirected to the Login Screen again. | |

Figure 11 Use Case Full Description - Login

**4.2.1.3 Captures Plant Image**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Captures Plant Image | |
| Scenario: | User has photographed a plant for identification | |
| Triggering Event: | User wants to identify a certain plant | |
| Brief Description: | User can identify plant by capturing a photo of its leaf. | |
| Actor/s | * Users * System | |
| Related Use Case: | * Log In * User Registers | |
| Stakeholders: | * User * System | |
| Preconditions: | * User’s device must have a camera * User must give permission to camera and internet access. | |
| Post Conditions: | * Photo uploaded for processing | |
| Flow of Activities: | User | System |
| 1.0 User captures photo | 1.1 Upload photo to system |
| Exception Conditions: | 1.0 Permission to access camera not granted by the user.  1.1 Permission to access internet not granted by the user. | |

Figure 12 Use Case Full Description – Captures Plant Image

**4.2.1.4 Process Image**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Process Image | |
| Scenario: | System compares image with training data set | |
| Triggering Event: | System receives photo | |
| Brief Description: | Once the image is captured, the system will try to find a match for the captured photo. | |
| Actor/s | * System | |
| Related Use Case: | * Captures Plant Image | |
| Stakeholders: | * User * System | |
| Preconditions: | * User must have photographed a plant | |
| Post Conditions: | * Display result | |
| Flow of Activities: | User | System |
| 1.0 User submits photo for recognition | 1.1 System displays result of image recognition |
| Exception Conditions: | 1.0 Permission to access device location not granted  1.1 Permission to access internet not granted | |

Figure 13 Use Case Full Description – Process Image

**4.2.1.5 Create Unidentified Plant Report**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Creates Unidentified Plant report | |
| Scenario: | System creates report if plant match is not found | |
| Triggering Event: | Plant cannot be identified by system | |
| Brief Description: | In cases of no matches in the system’s data set , the system will generate a not found report. | |
| Actor/s | * System | |
| Related Use Case: | * Captures Plant Image * Process Image | |
| Stakeholders: | * User * System | |
| Preconditions: | * System is not able to find a match from the data set | |
| Post Conditions: | * Not found report is created | |
| Flow of Activities: | User | System |
| 1.0 User submits photo for recognition | 1.1 System generates a Unidentified Plant report |
| Exception Conditions: | 1.0 Plant is identified by the system | |

Figure 14 Use Case Full Description – Create Unidentified Plant Report

**4.2.1.6 Create Incorrect Location Report**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create Incorrect Location report | |
| Scenario: | User reports incorrect/missing location | |
| Triggering Event: | Report from user | |
| Brief Description: | User can create incorrect location report if locations of plant matches are incorrect or missing. | |
| Actor/s | * System * User | |
| Related Use Case: | * Captures Plant Image * Process Image | |
| Stakeholders: | * User * System * Admin | |
| Preconditions: | * The user must have reported for either incorrect match or incorrect location report. | |
| Post Conditions: | * Incorrect location report generated | |
| Flow of Activities: | User | System |
| 1. User selects a plant location on the map 2. User report the location | * 1. System display plant match result, plant information and location details   2.1 System generates an incorrect location report |
| Exception Conditions: |  | |

Figure 15 Use Case Full Description – Create Incorrect Location Report

**4.2.1.7 Create Add/Edit Information Request**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Create Add/Edit Plant Information Request | |
| Scenario: | User wants to add or edit plant information in the plant glossary | |
| Triggering Event: | User adds/edits plant information in the plant glossary | |
| Brief Description: | Users can submit requests to add information or edit existing information on the plant glossary | |
| Actor/s | * System * User | |
| Related Use Case: | * Login * Create unidentified plant report * Create incorrect location report | |
| Stakeholders: | * System * Admin | |
| Preconditions: | * The user wants to add or edit information on the plant glossary | |
| Post Conditions: | * An add/edit information request is submitted for the admin to validate. | |
| Flow of Activities: | User | System |
| 1. User selects Add/Edit Information 2. User fills out form | * 1. System displays Add/Edit Information Form   2.1 System submits request for the admin to validate. |
| Exception Conditions: | 1.0 User did not select Add/Edit Information and Cancelled | |

Figure 16 Use Case Full Description – Create Add/Edit Information Request

**4.2.1.8 Validate Report and Request**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Validate Report and Request | |
| Scenario: | Admin responds to requests and reports | |
| Triggering Event: | Validation from Admin | |
| Brief Description: | Admin checks reports and requests. Admin can update locations as a response to incorrect location reports and verify add/edit requests to update the plant glossary. | |
| Actor/s | * Admin * System | |
| Related Use Case: | * Create unidentified plant report * Create incorrect location report * Create Add/Edit Plant Information Request | |
| Stakeholders: | * System | |
| Preconditions: | * Admin must be logged in | |
| Post Conditions: | * System records updated. | |
| Flow of Activities: | Admin | System |
| 1.0 Admin selects Reports/Requests  2.0 Admin selects record to be edited  3.0 Admin make changes to record | 1.1 Display all Reports/Requests  2.1 Display edit record form  3.1 System updates records |
| Exception Conditions: | 3.1 Admin makes no changes to records. | |

Figure 17 Use Case Full Description – Validate Report and Request

**4.1.2.9 Edits Plant Glossary**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Edits Plant Glossary | |
| Scenario: | Admin updates plant glossary | |
| Triggering Event: | Admin updates plant details in the system | |
| Brief Description: | Admin can add new findings/information originating from the admin or users to make sure it is up-to-date. | |
| Actor/s | * Admin * System | |
| Related Use Case: | * Login | |
| Stakeholders: | * System | |
| Preconditions: | * Admin must be logged in | |
| Post Conditions: | * Plant Glossary updated | |
| Flow of Activities: | Admin | System |
| 1.0 Admin selects the plant glossary  2.0 Admin selects record to be edited  or creates new plant record  3.03 3.0 Admin make changes to record/  submit new record | 1.0 Display all plant records  2.1 Display edit/create record form  3.1 System updates records |
| Exception Conditions: | 3.0 No information will be added. | |

Figure 18 Use Case Full Description – Edits Plant Glossary

**4.2.1.10 Generate System Reports**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Generate System Reports | |
| Scenario: | System generates reports | |
| Triggering Event: | Admin requests for System Reports | |
| Brief Description: | System can generate reports on summary of requests, reports, identification results and plant location. | |
| Actor/s | * Admin * System | |
| Related Use Case: | * Login | |
| Stakeholders: | * Admin | |
| Preconditions: | * Admin must be logged in | |
| Post Conditions: | * System generates reports | |
| Flow of Activities: | Admin | System |
| 1.0 Admin selects the type of report to request | 1.1 System display the summary/report. |
| Exception Conditions: | 1.0 Admin has not made any selection | |

Figure 19 Use Case Full Description – Generate System Reports

**4.3 Gap Analysis/Needs Assessment**

|  |  |  |
| --- | --- | --- |
| User Requirement | Current Standing | Proposed Action |
| Allow users to identify plants easily | Specimen/plant has to be brought to PITAHC or Bureau of Plant Industry for identiciation | Specimen/plant is identified through the Image Recognition feature of the Lyf app. |
| Improve information dissemination by accessing more platforms to reach users. | Information is currently just being posted on PITAHC’s site | Aside from the web, info can now be viewed, accessed via Lyf app. |
| Encourage sharing of information with PITAHC and other citizens/researchers through easy and uncomplicated process of submission | People who want to share their research and have them released through PITAHC has to schedule a meeting with a PITAHC representative. | Information can be submitted through the Lyf app. Users are notified upon verification and approval of PITAHC. Information will then be disseminated and available on the Lyf app. |
| Gather information on plant population or abundance on locations | PITAHC performs field research, sends representatives to location, and manually keep track of plant population | Location of the identified plants are automatically recorded, available to other Lyf app users and data source of system report for the admin. |

Figure 20 Gap Analysis/Needs Assessment

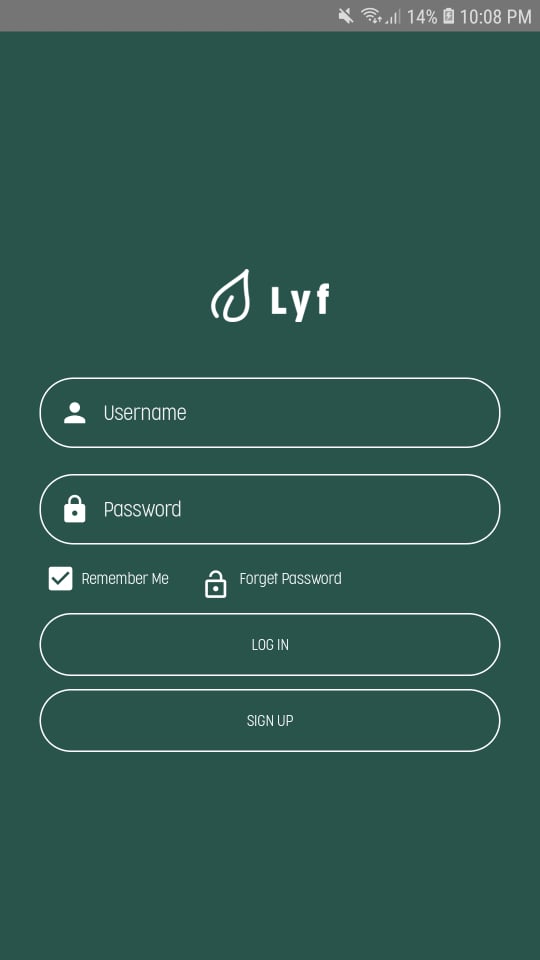
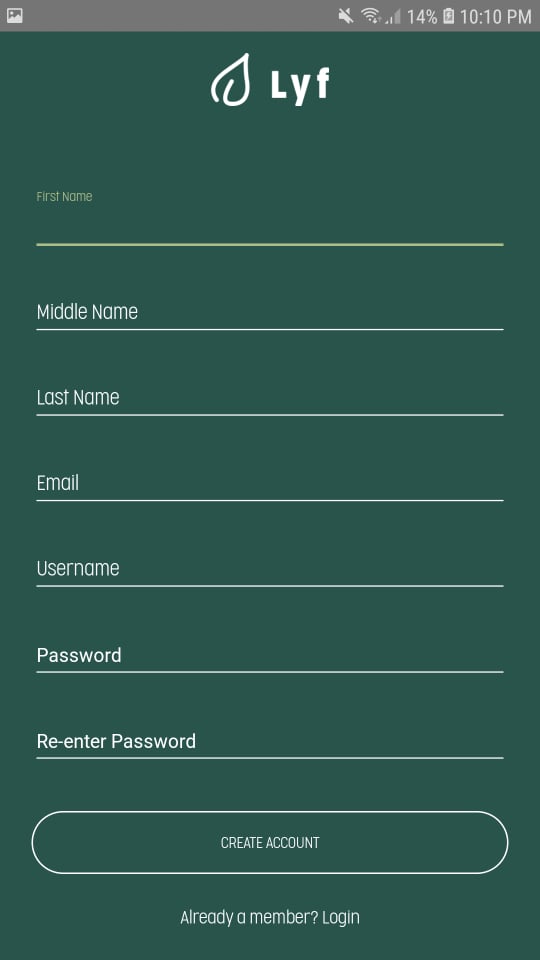
**4.4 Design of Software, Systems, Product, and/or Processes**

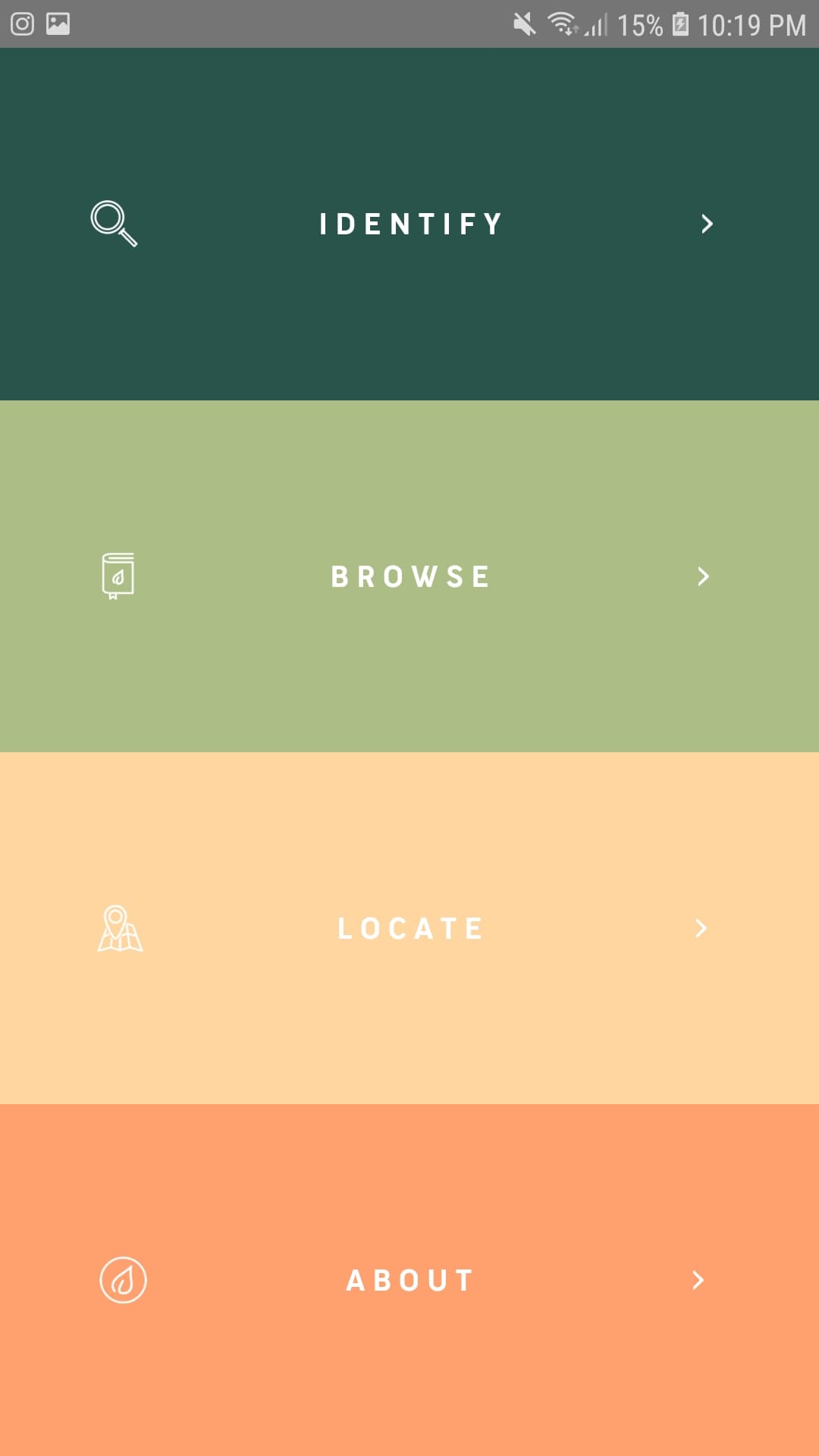
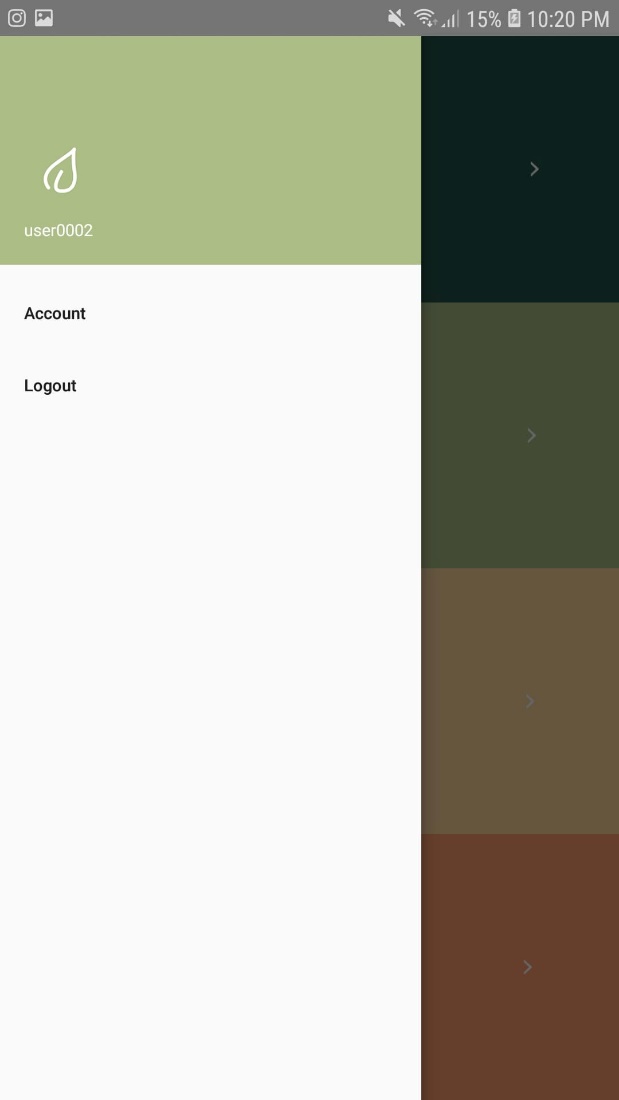
Figure 20 Android - LoginFigure 21 Android - Register

Figure 22 Android – Main MenuFigure 23 Android – Navigation Drawer

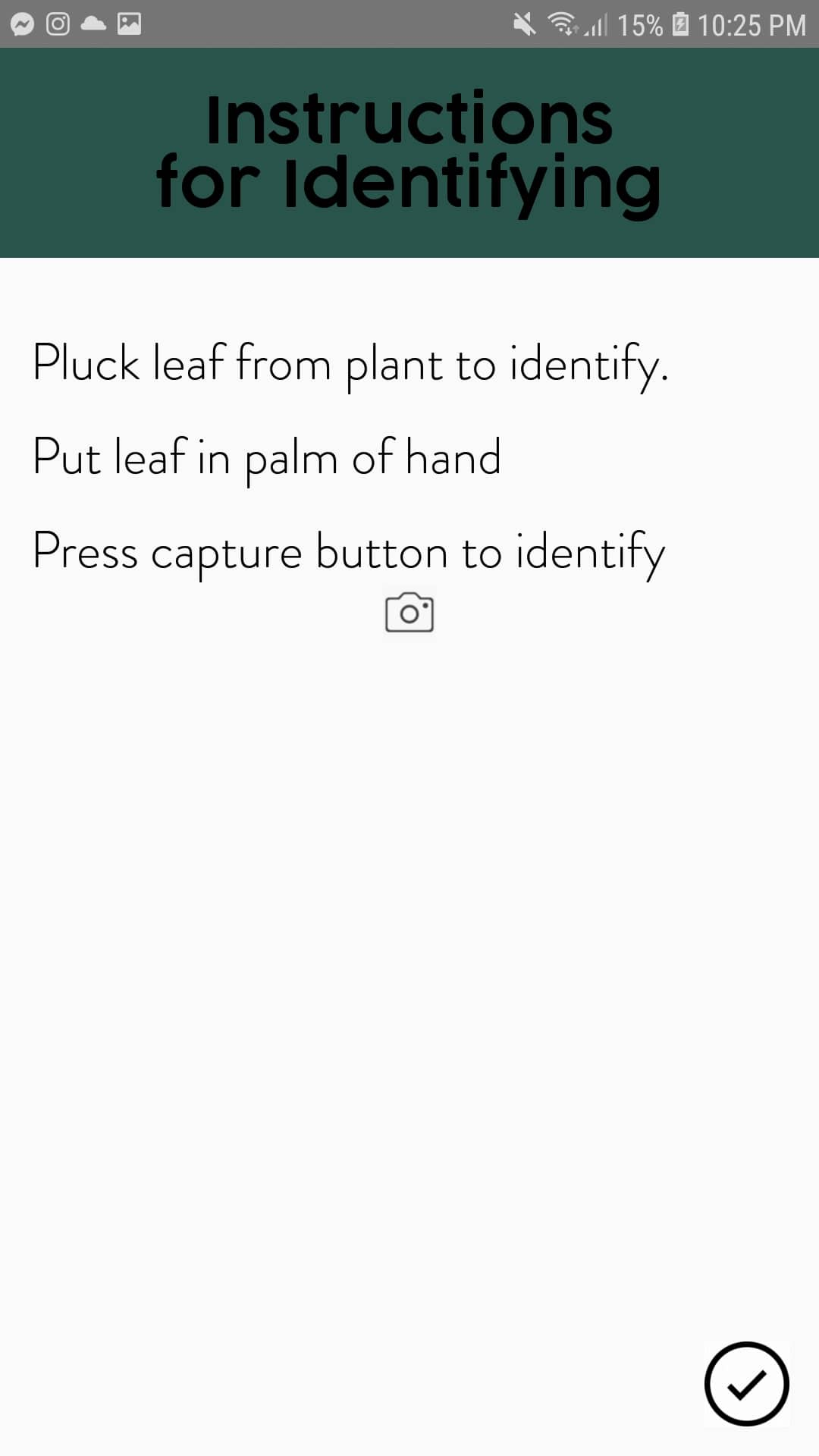


Figure 24 Android – Camera ViewFigure 25 Android – Camera Tutorial

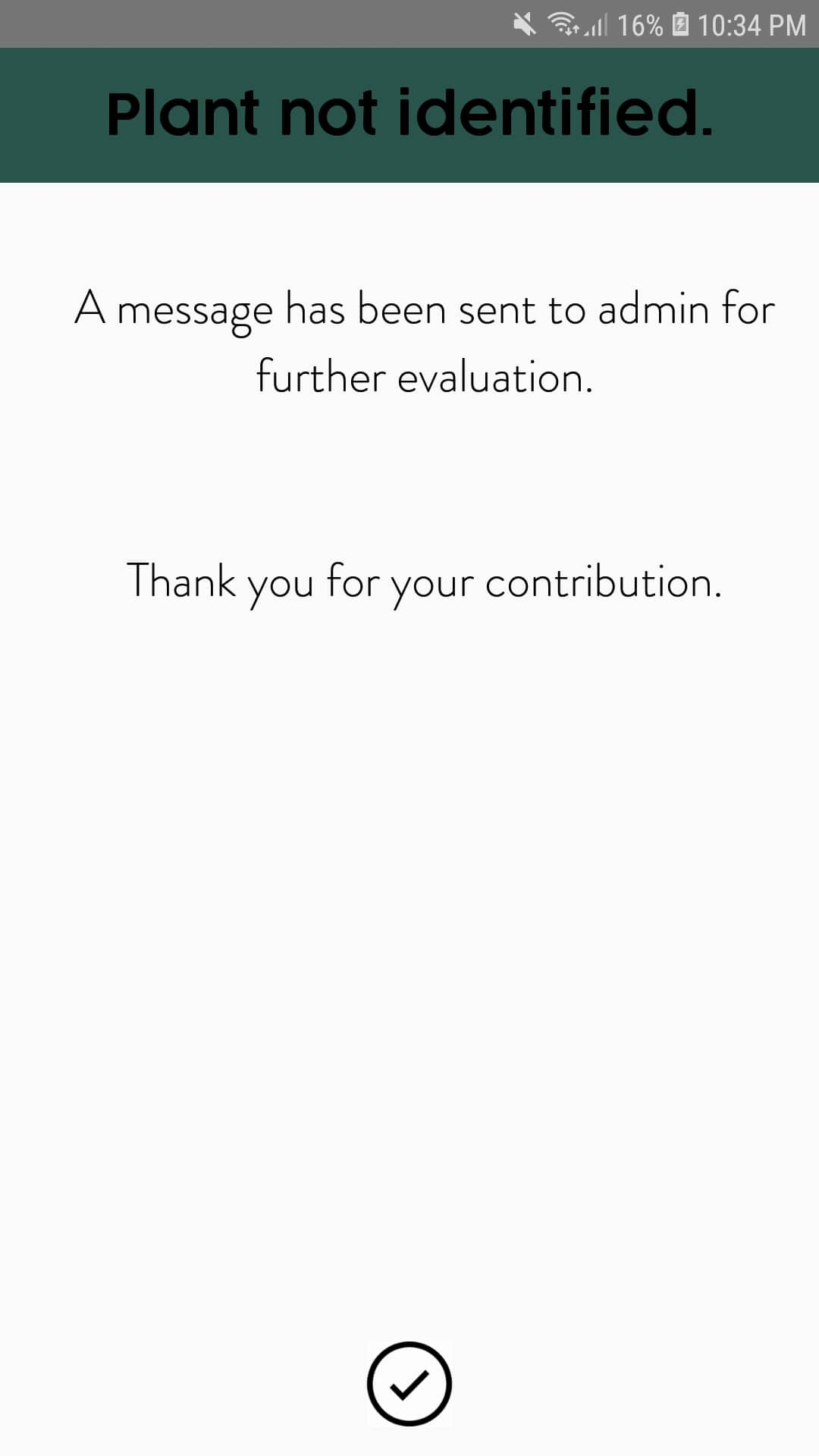


Figure 26 Android – Image Recognition ResultsFigure 27 Android – Plant not identified result

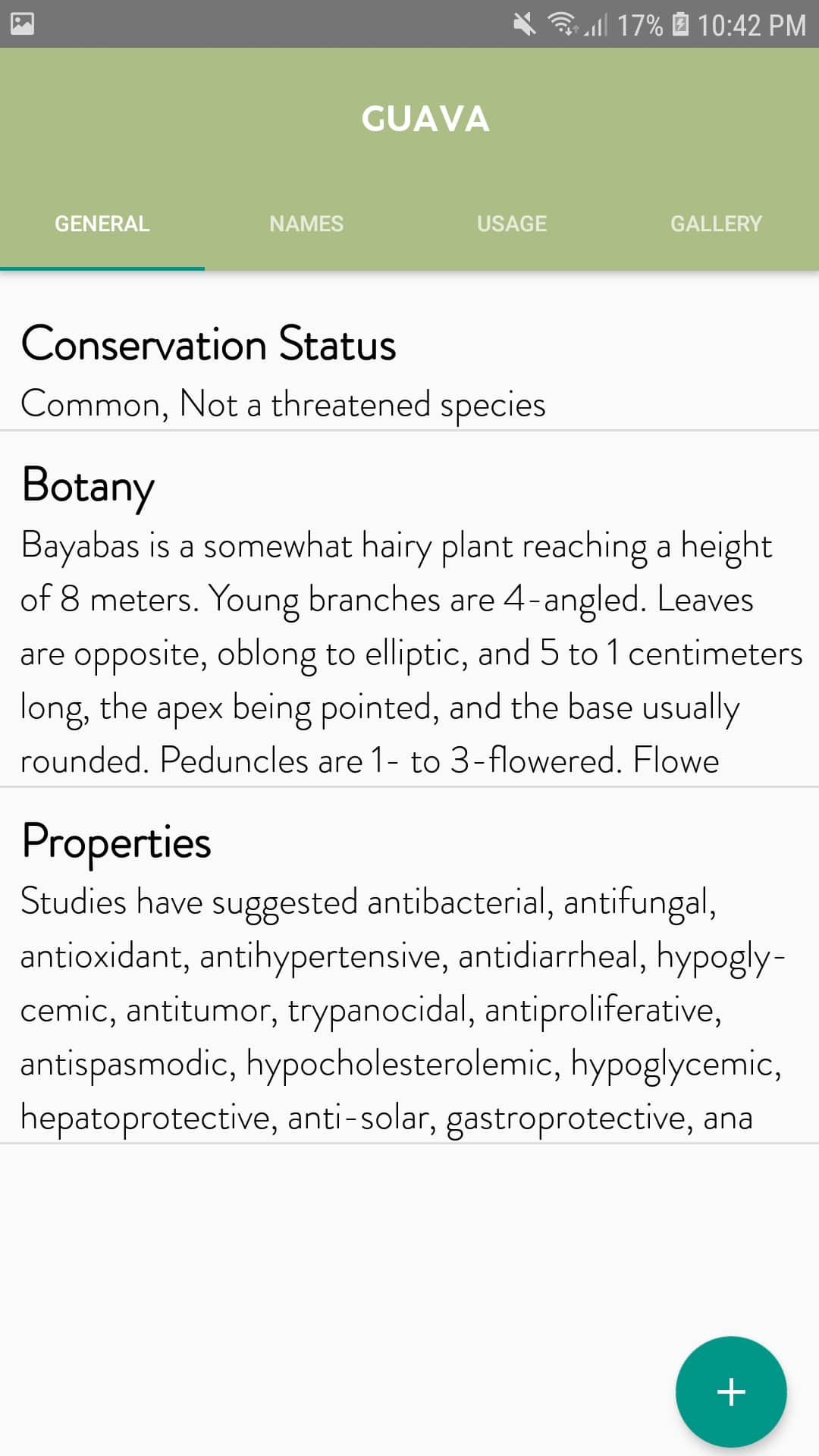


Figure 28 Android –Browse Figure 29 Android – Tab: General Information

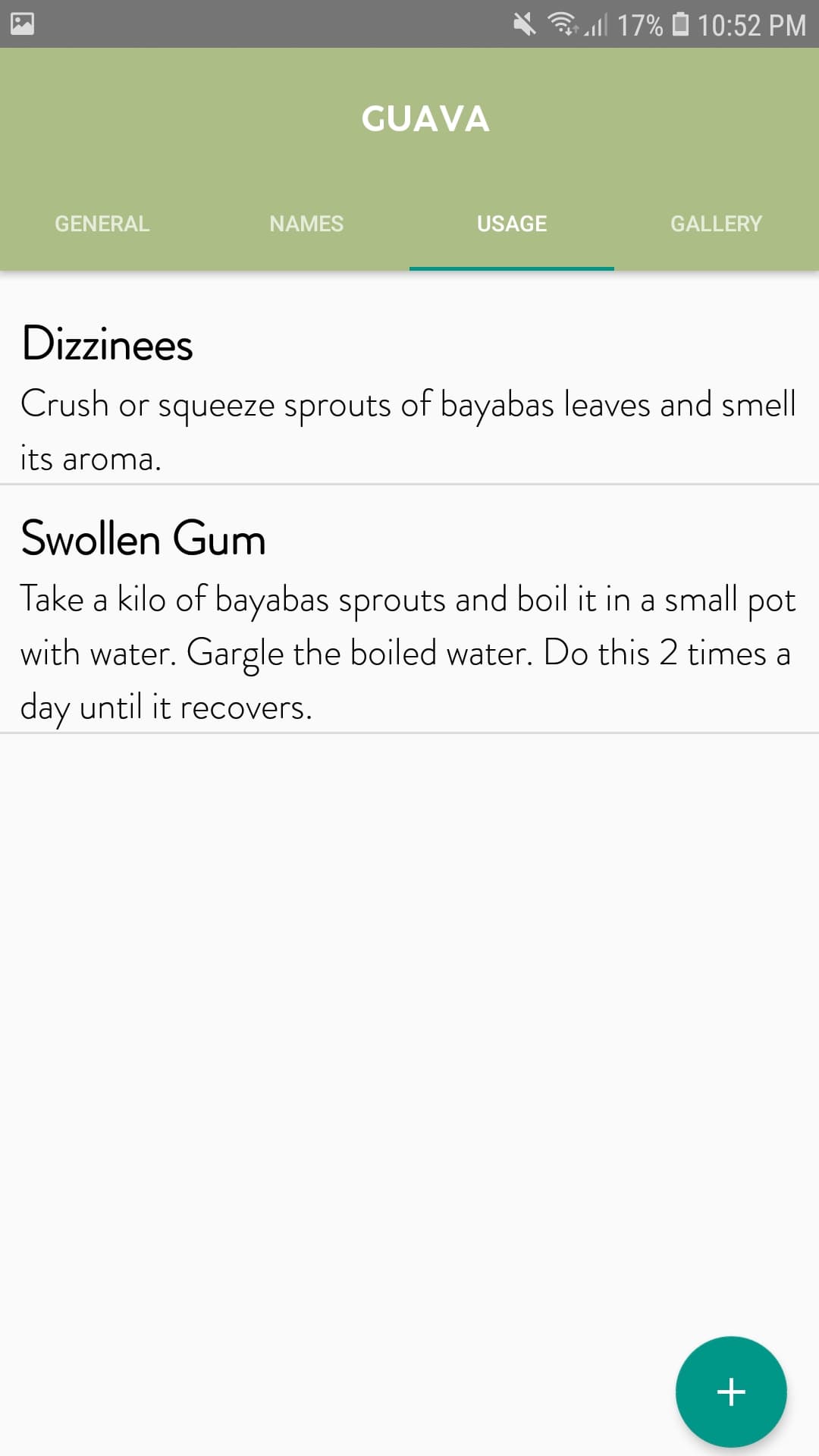
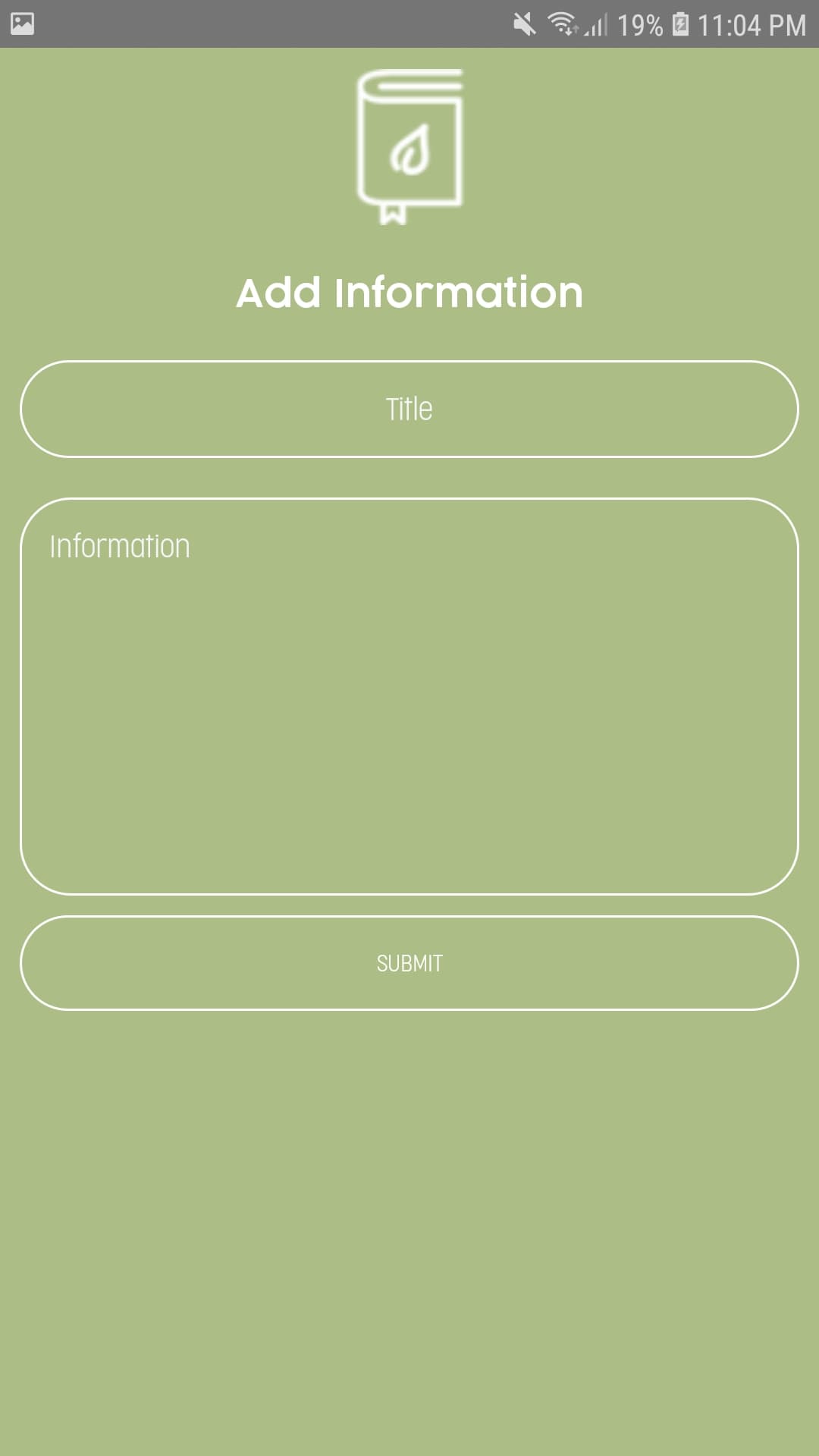


Figure 30 Android –Tab: Names Figure 31 Android – Tab: Usage



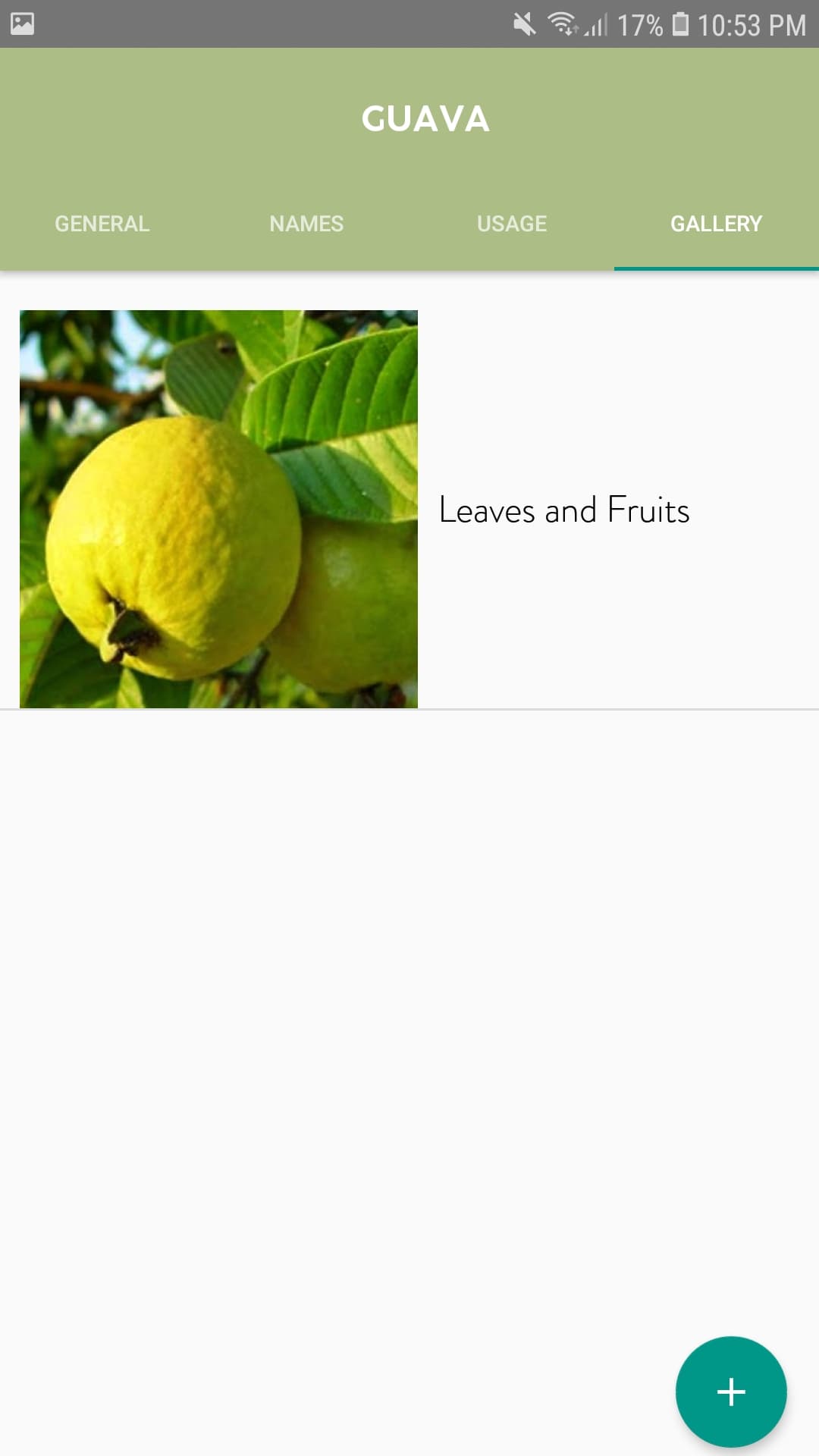


Figure 32 Android –Tab: Gallery Figure 33 Android – Add Information

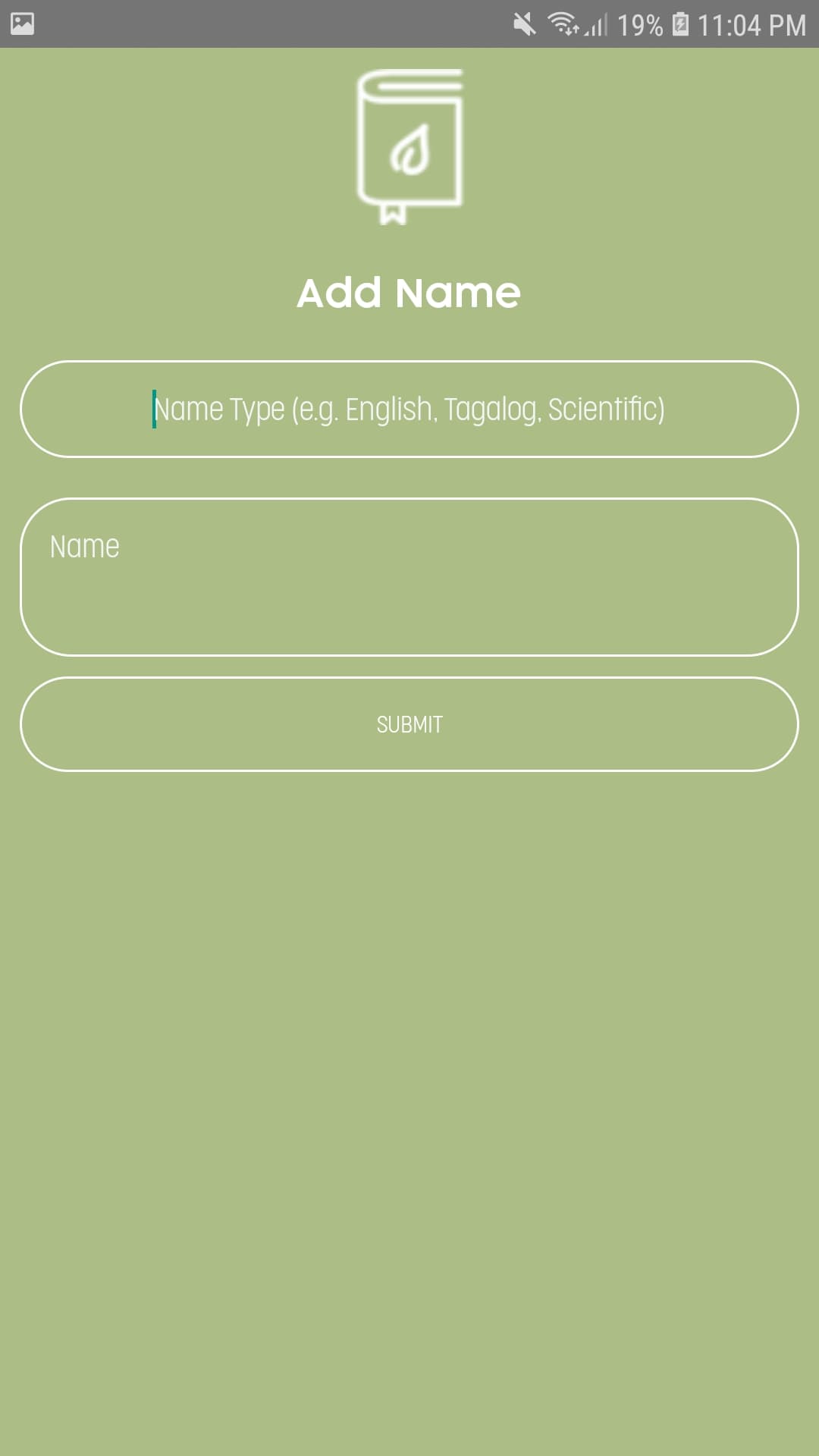
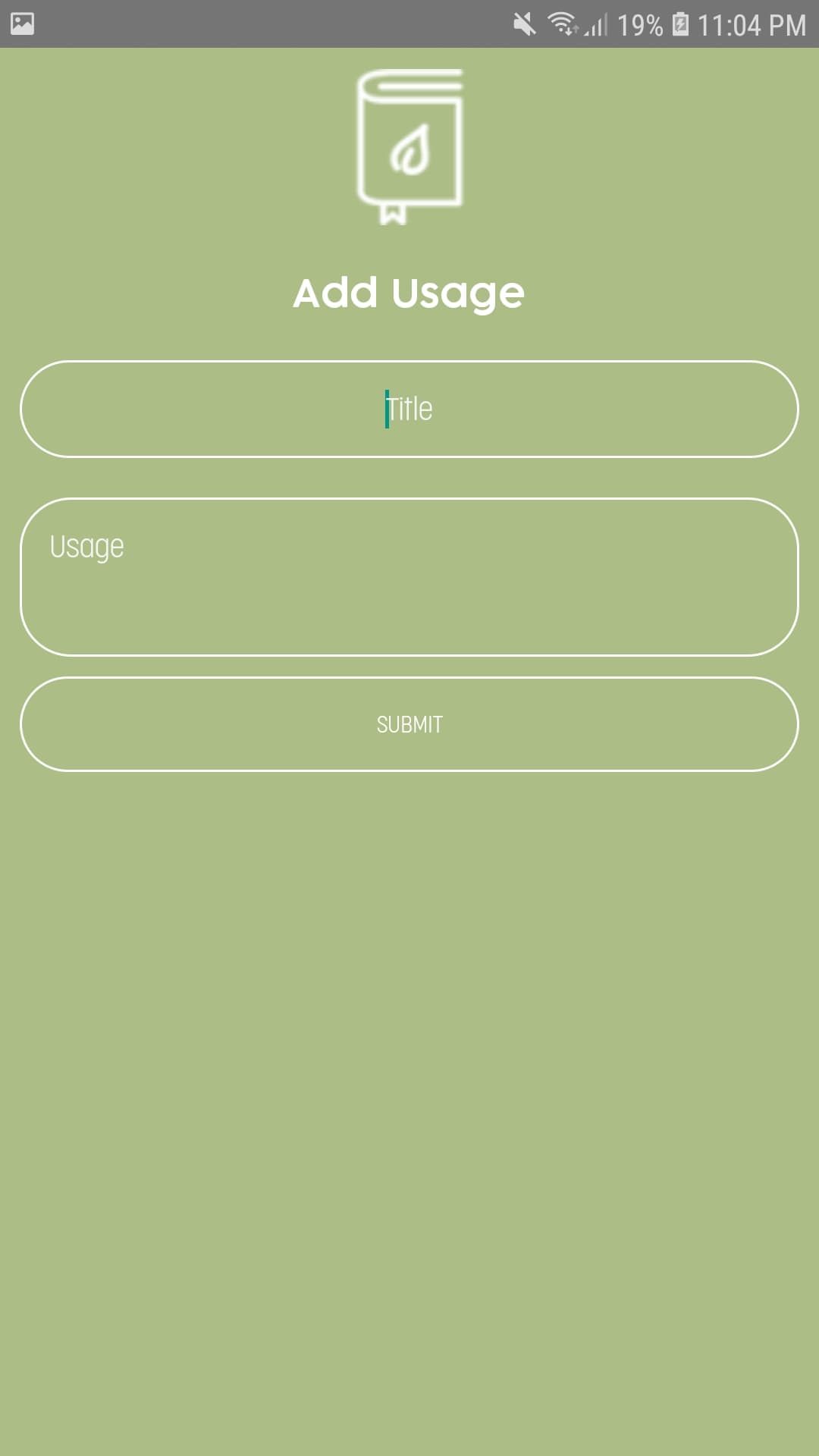


Figure 34 Android – Add Name Figure 35 Android – Add Usage

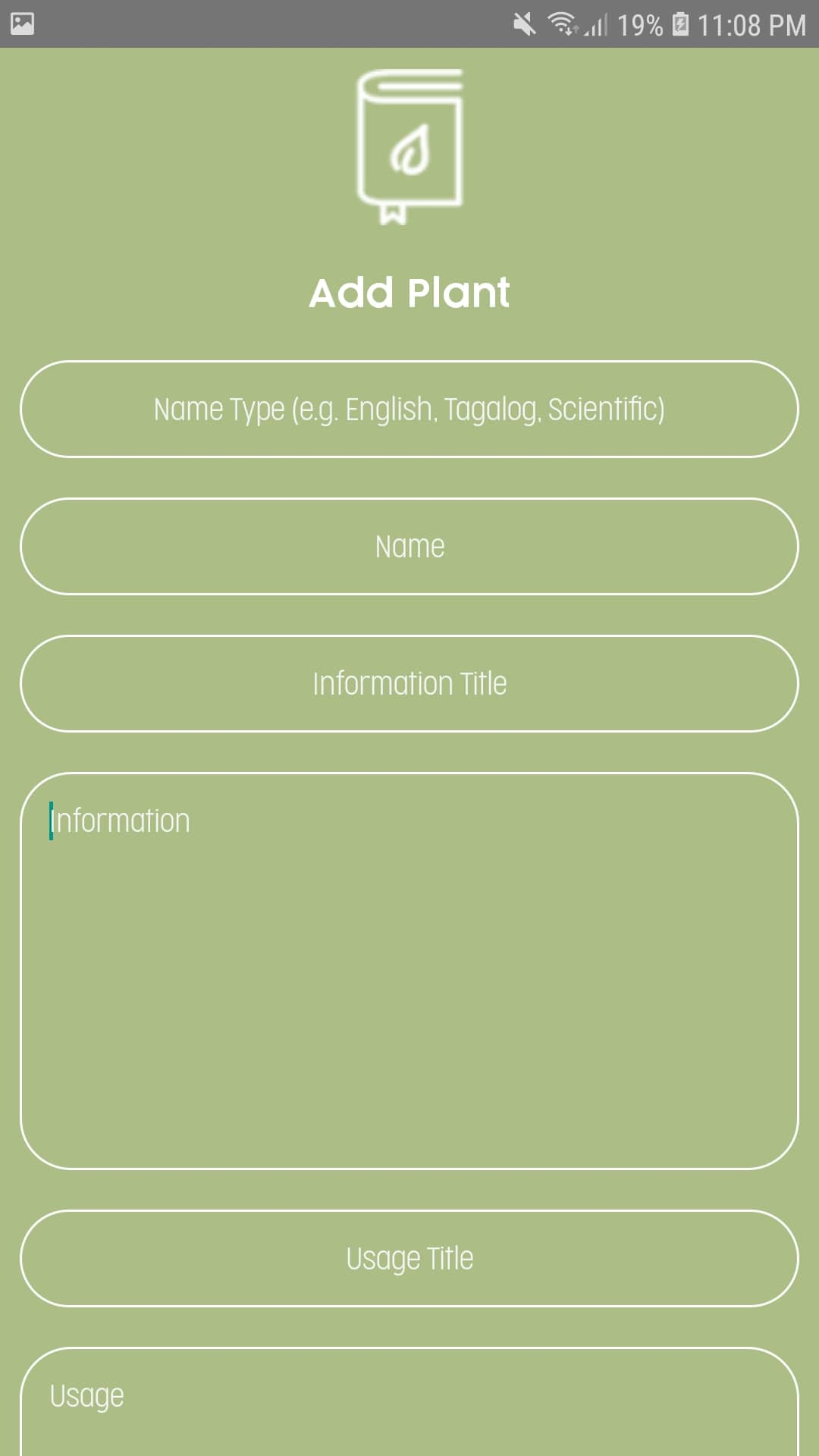


Figure 36 Android – Add Image Figure 37 Android – Add Plant

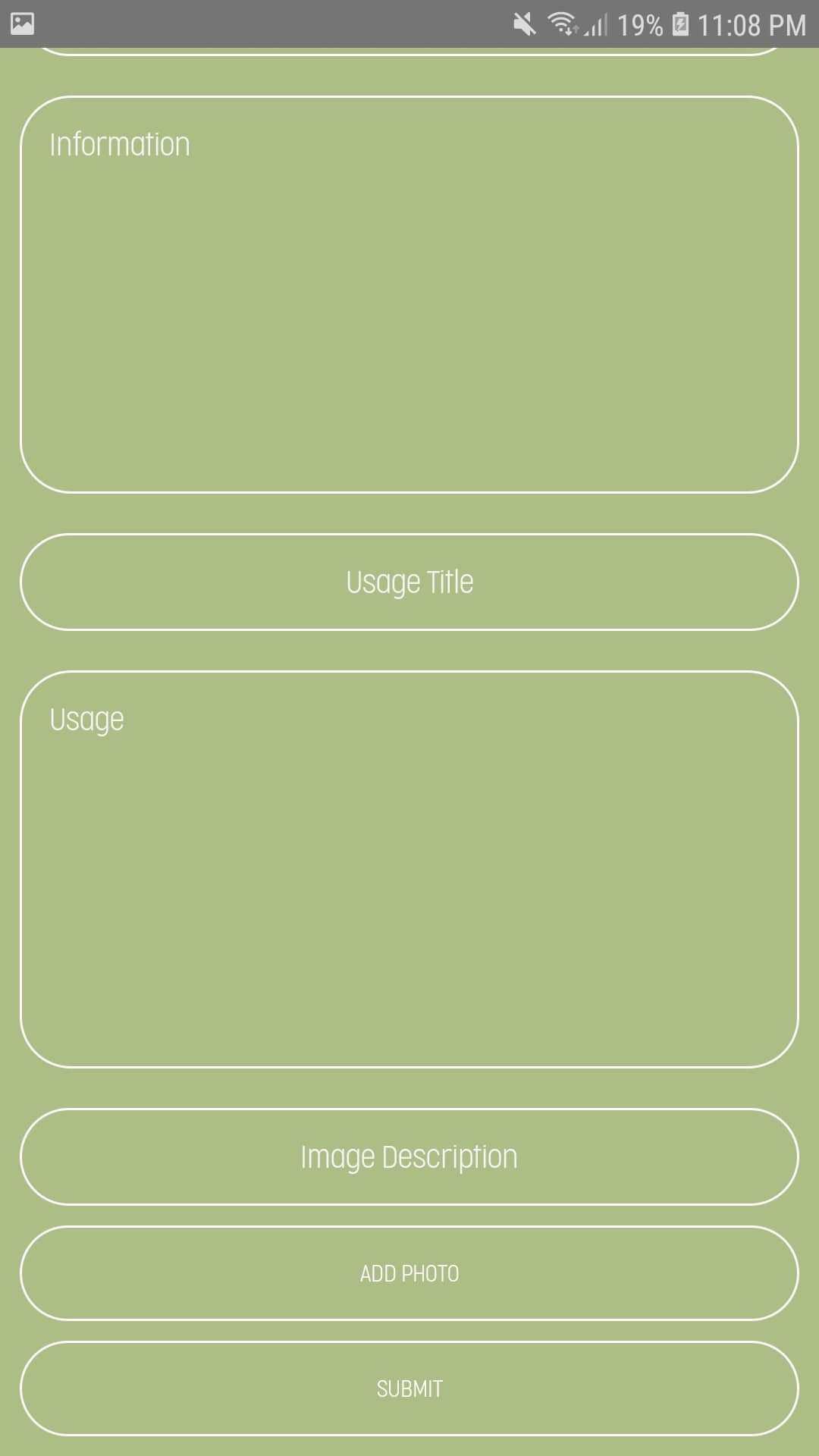
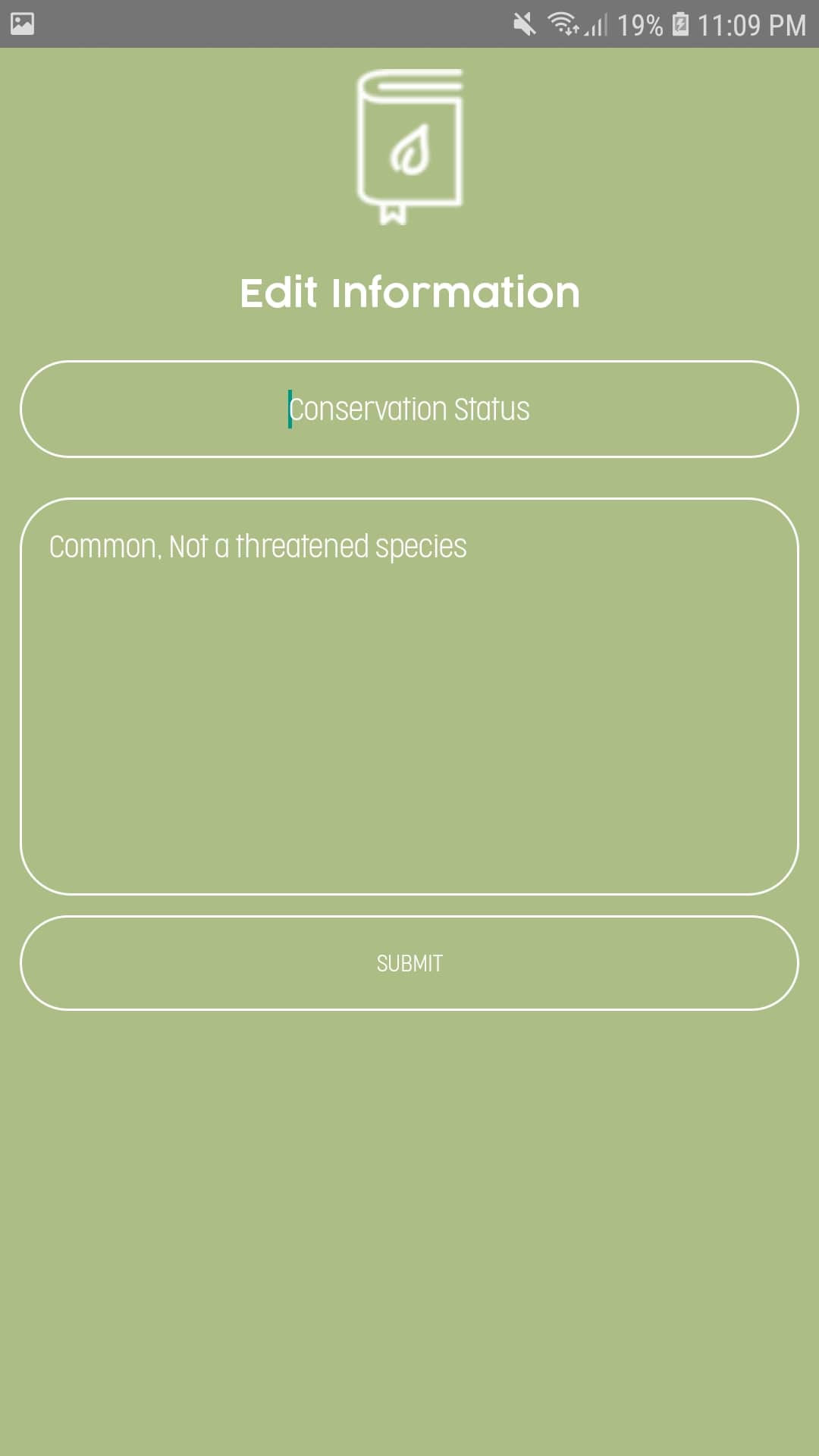


Figure 38 Android – Add Plant(cont) Figure 39 Android – Edit Information

Figure 40 Android – Edit Name Figure 41 Android – Edit Usage

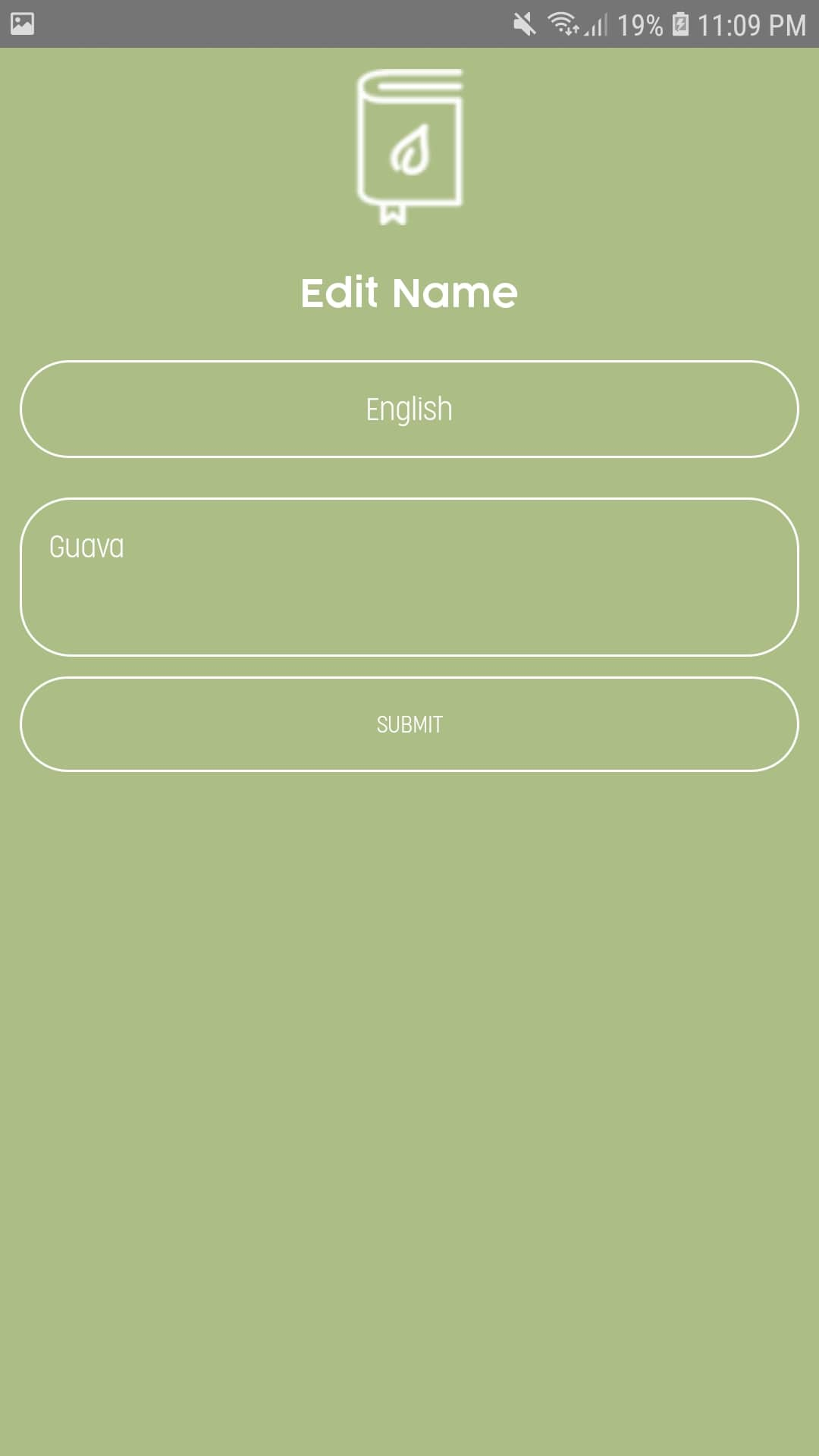
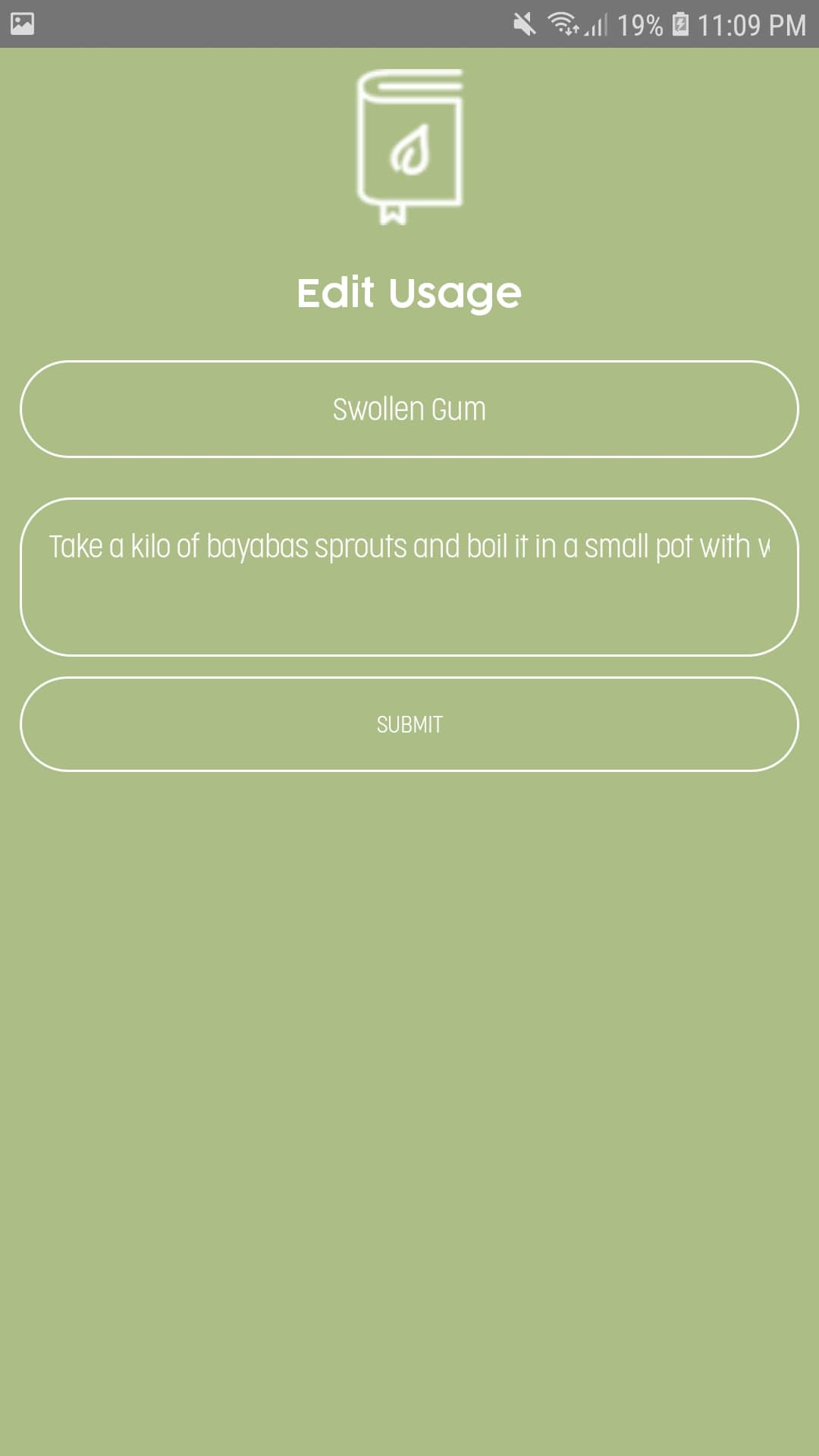
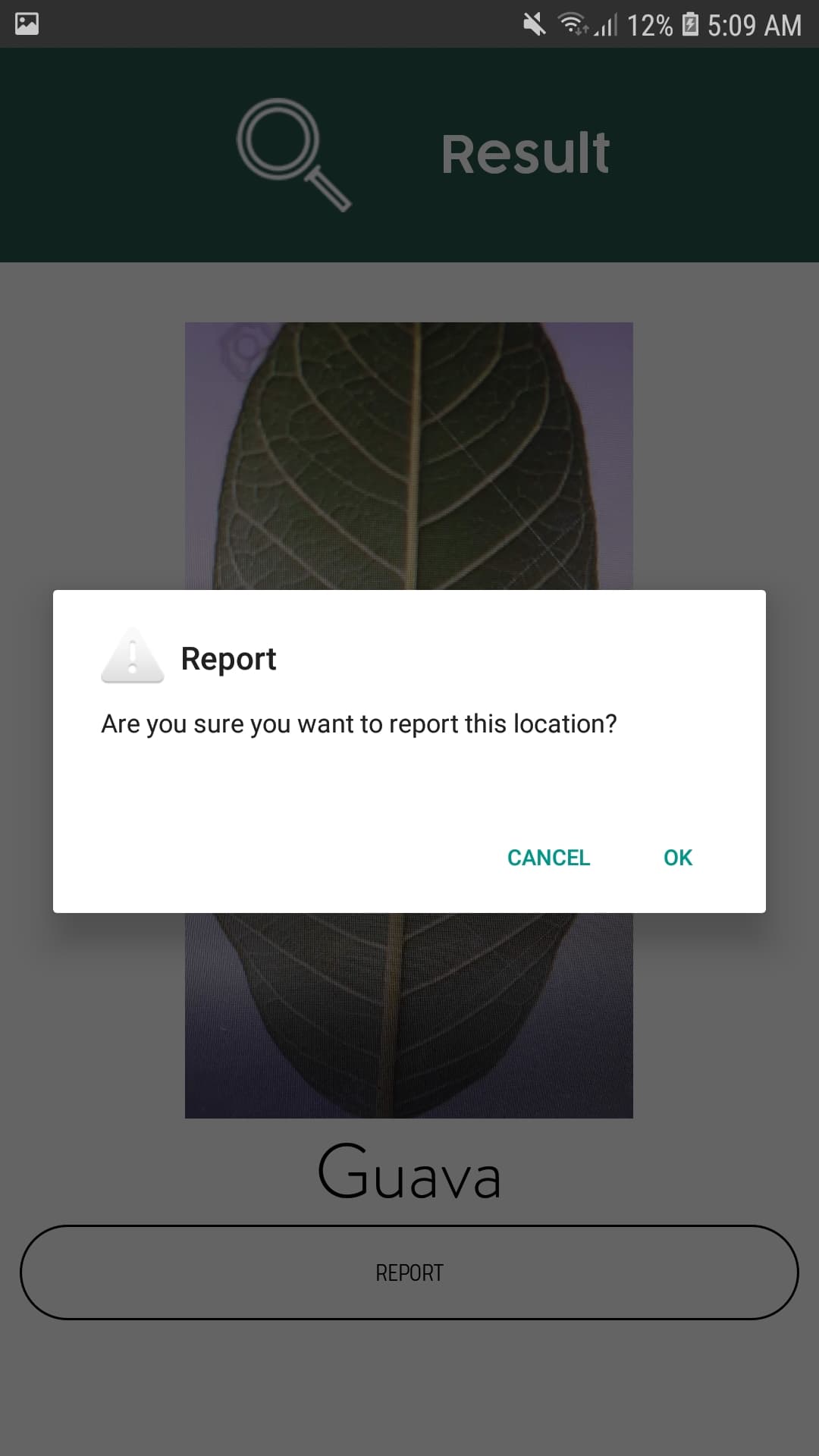
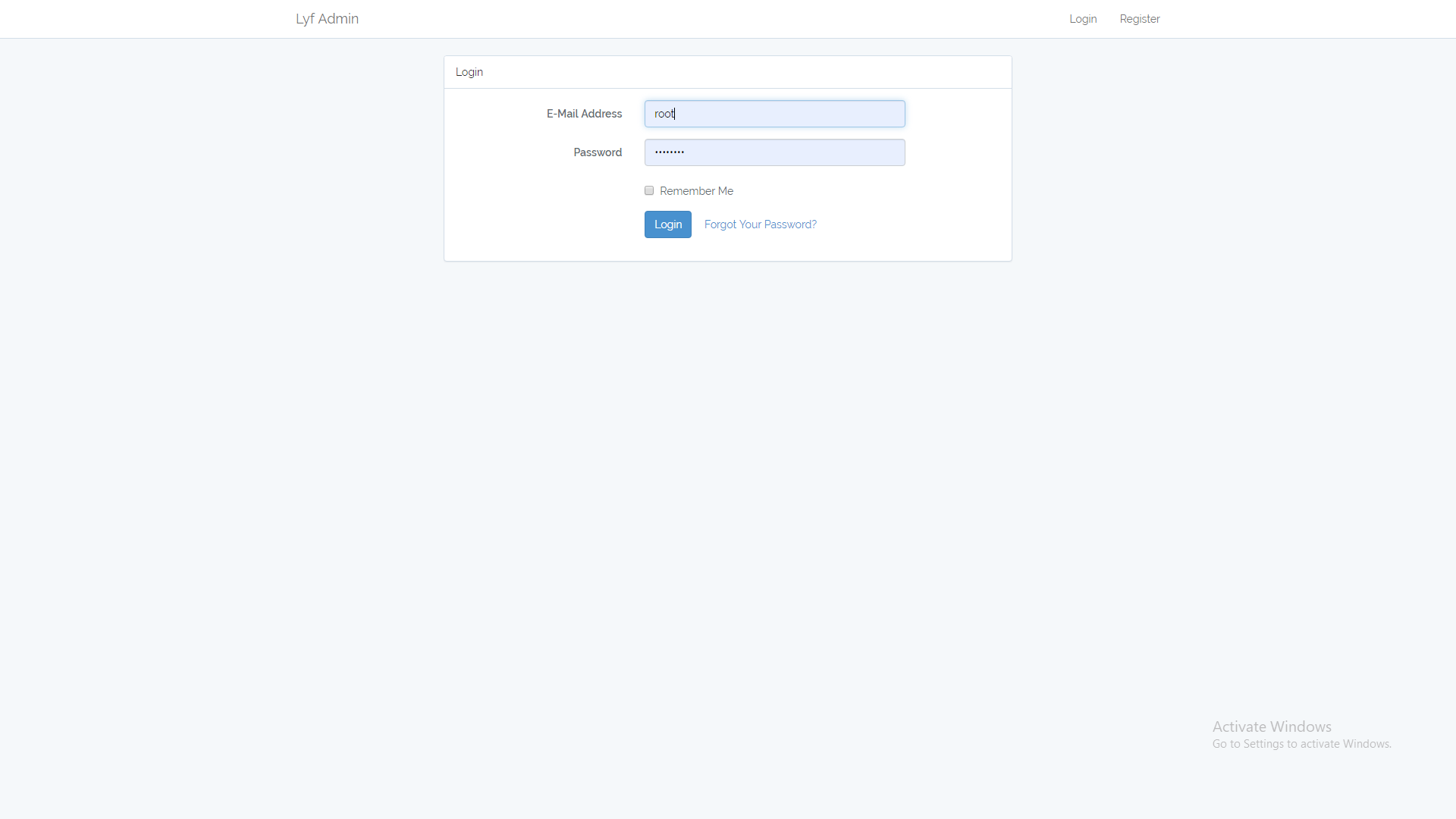




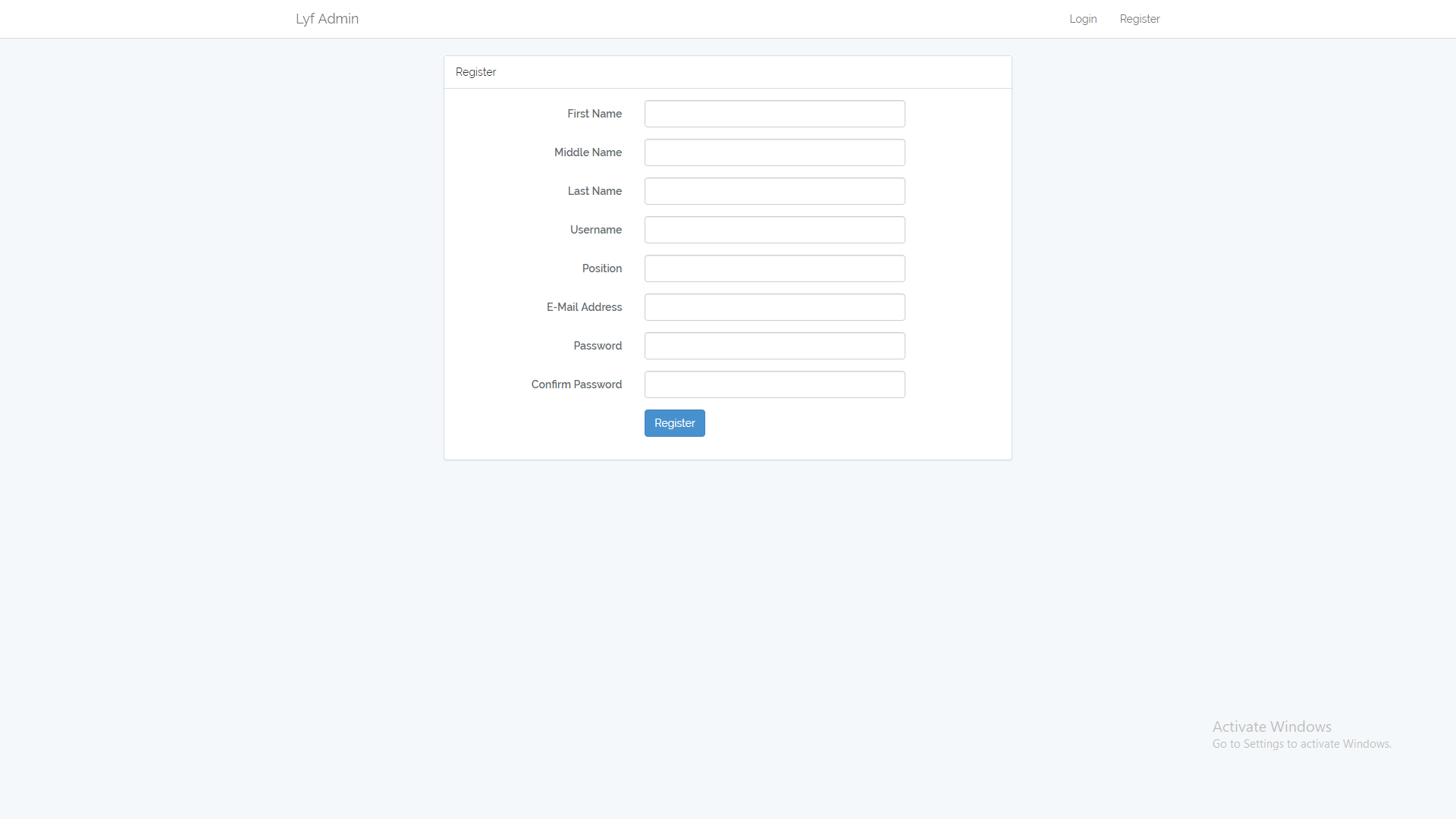
Figure 42 Android – Map View Figure 43 Android – View of Result



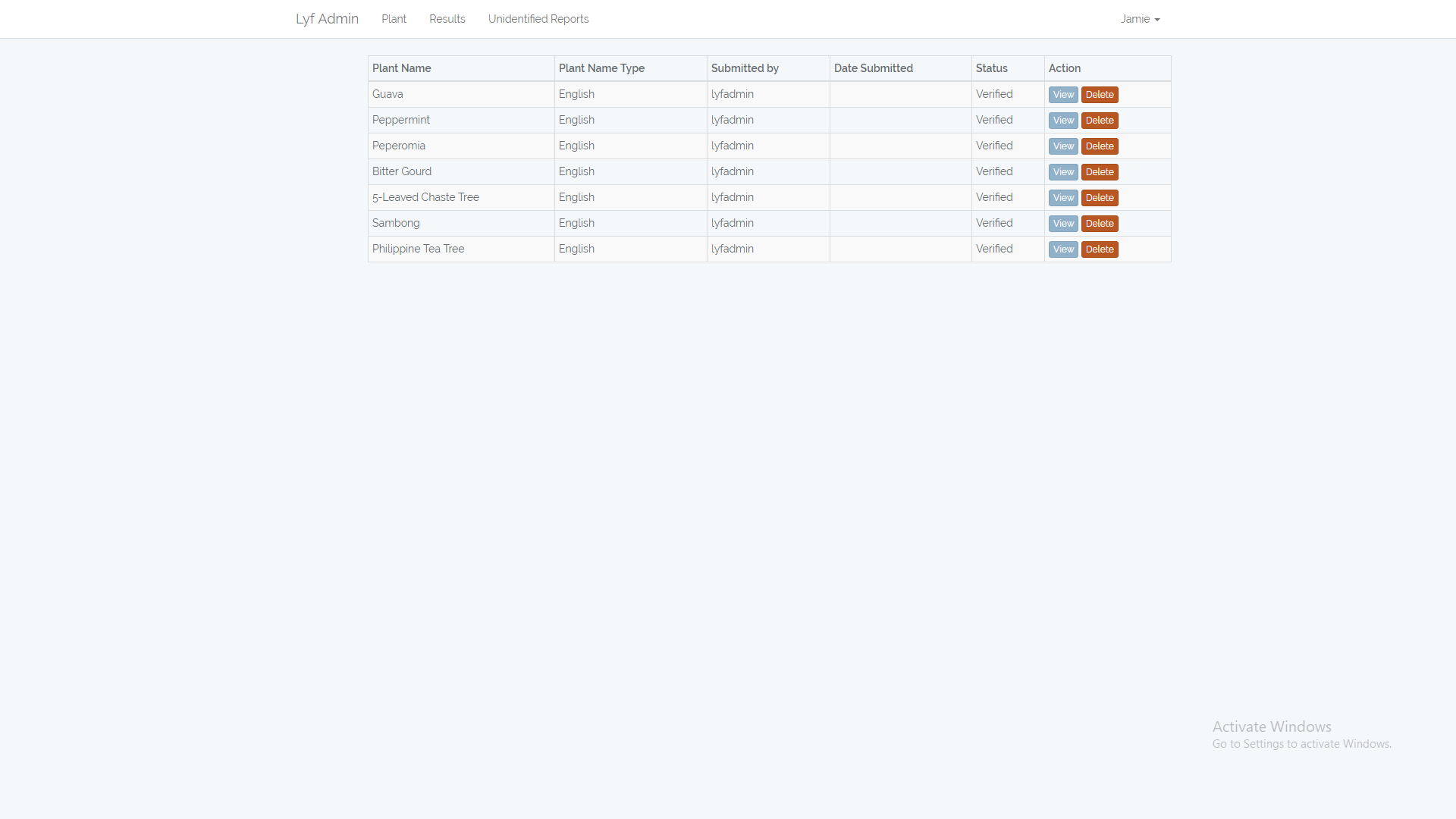
*Figure 44 Android – Reporting Location*



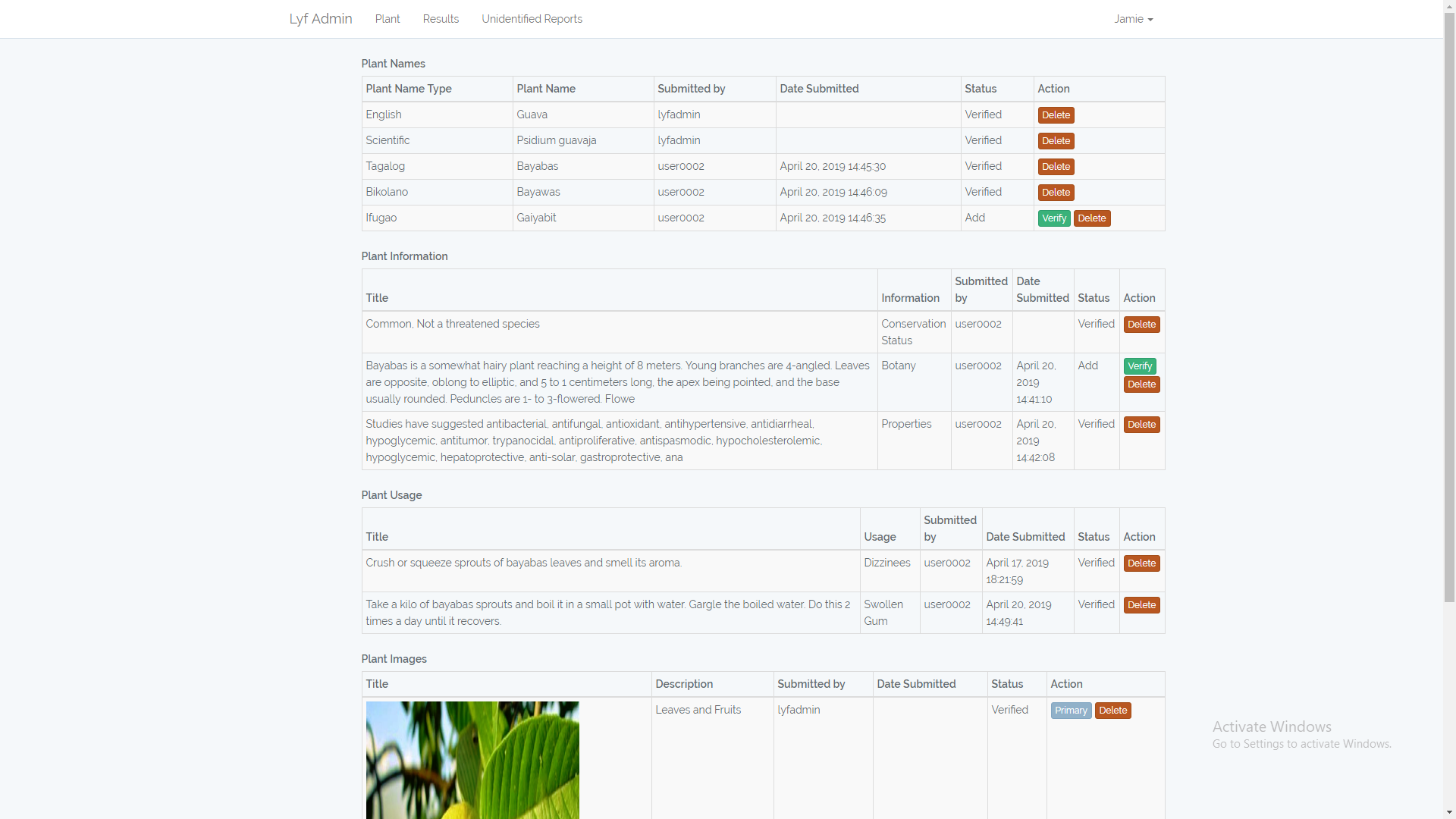
*Figure 45 Web – Login*



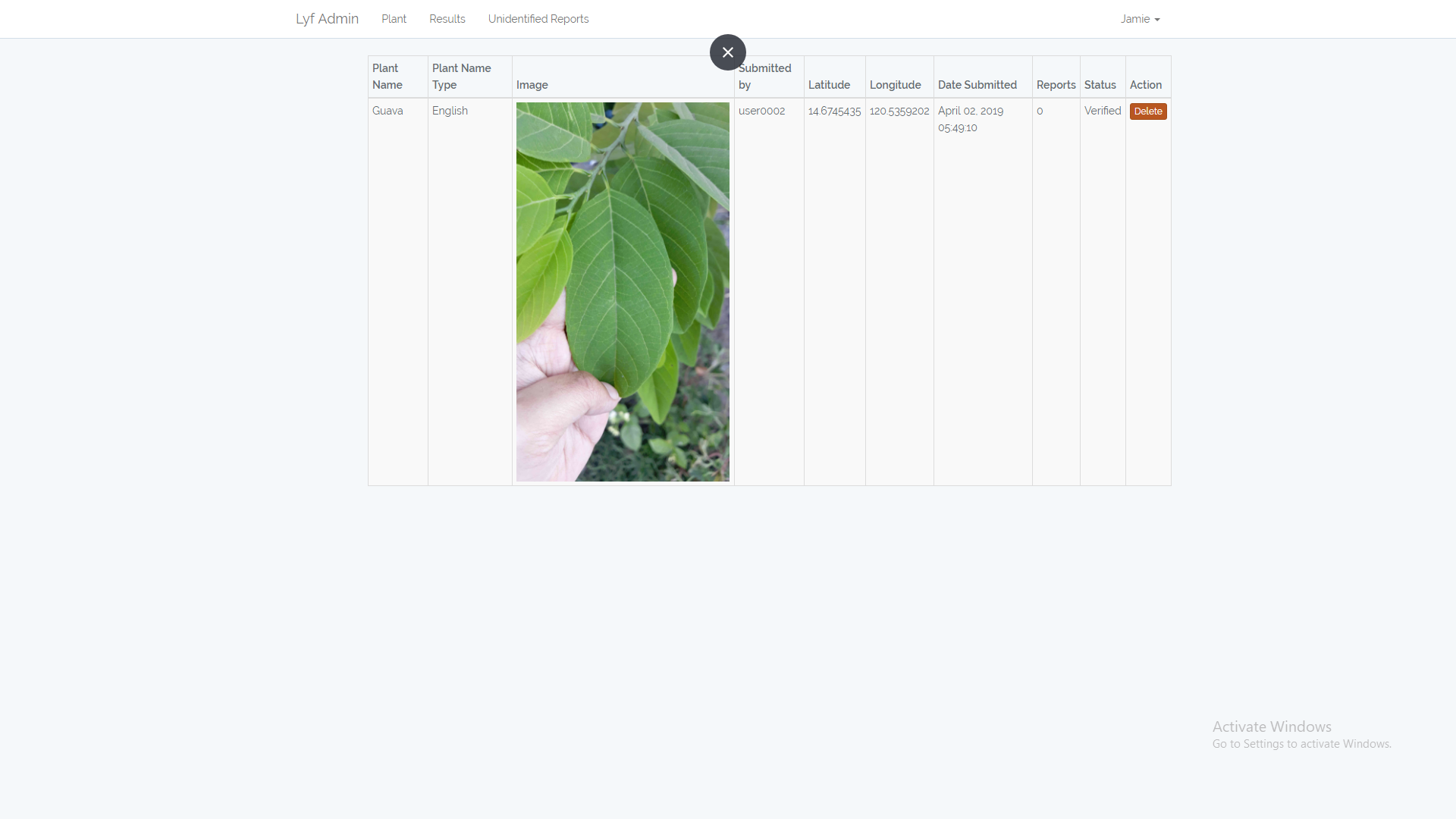
*Figure 46 Web – Register*



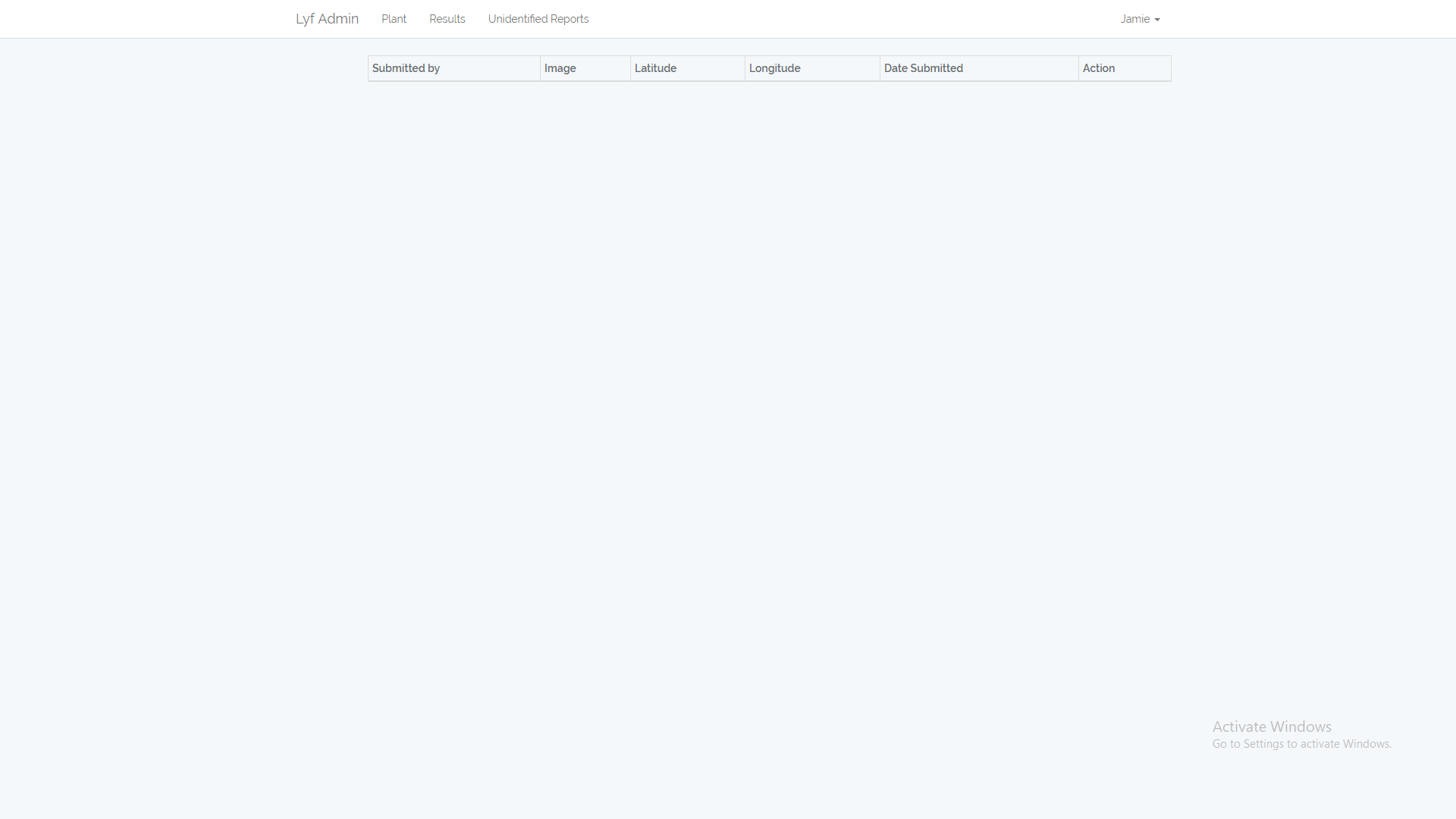
*Figure 47 Web – Plant Glossary*



*Figure 48 Web – Specific Plant Glossary including Requests*



*Figure 49 Web – Results*



*Figure 50 Web – Unidentified Reports*

**4.5 Development**

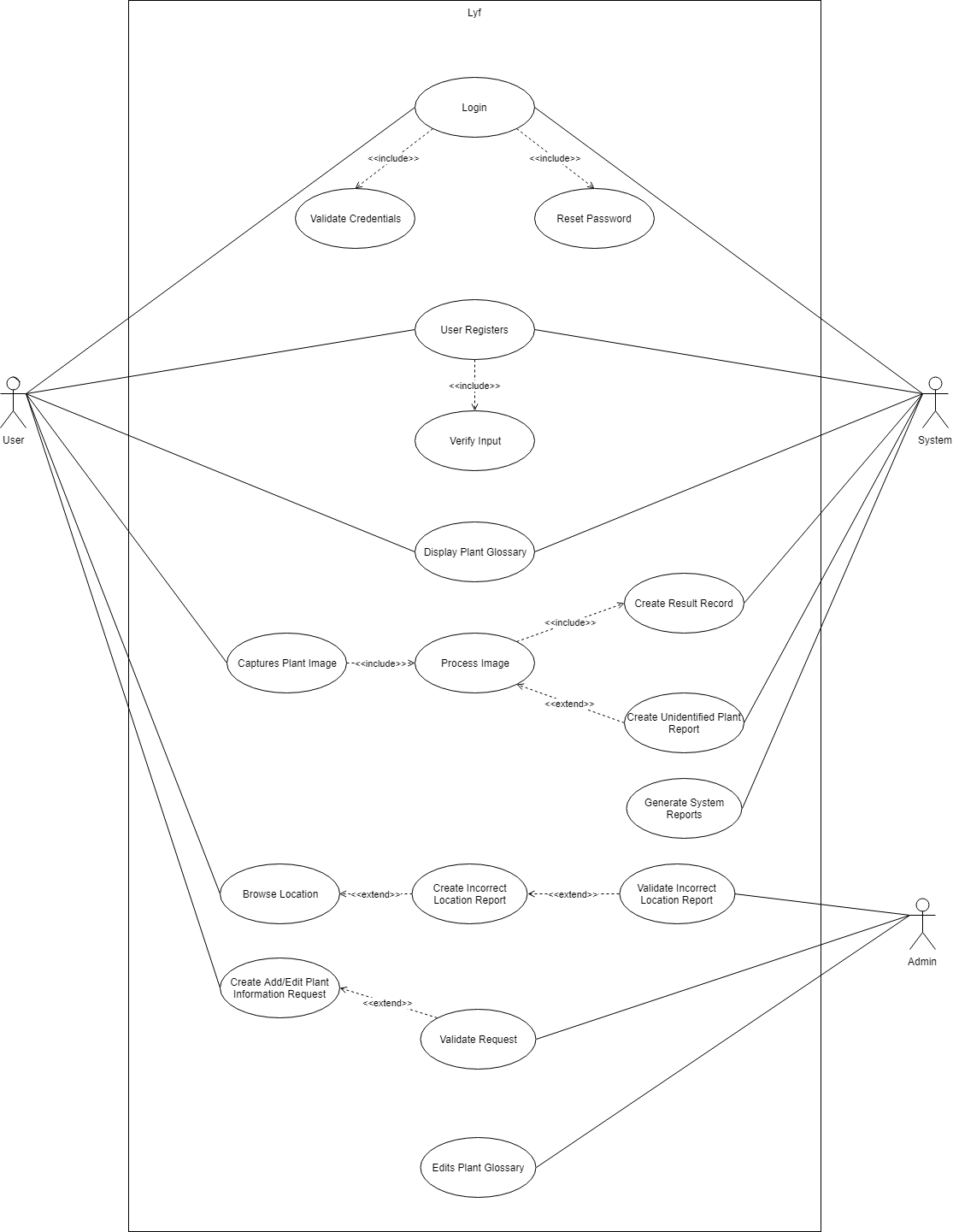
The team used TensorFlow to build the image recognition model imported in the Android application. To create the model, the team captures videos of holding the leaf in different angle and lighting. FFmpeg is used to generate images from each frames of the videos. The images are labeled in a folder. TensorFlow uses MobileNet for training. Input image resolution is 224 px. MobileNet performs same calculations at each location in the image. TensorFlow will generate a .pb and .txt file. The .pb files contains a version of the selected network with final layer retained for categories. The .txt file contains the labels. These two files are imported as assets on Android Studio.

The team opted to build the Android application on Android Studio because of the extensive support, especially from Google Developers. Also, integration of TensorFlow is easier using the said IDE. Google Maps API is used for functions related to locations such as getting device location and displaying plant match results in Google Maps. CameraView is used to call on camera functions such as camera view, flash, and capture. The Android application make and assess HTTP requests using the Volley library. It also uses API built on Lumen, a php microframework based on Laravel. The web application is built using Laravel. Layout of the current prototype are defaults of Laravel and will need better design.

The project is hosted on DigitalOcean on a Ubuntu 16.04 LAMP stack.

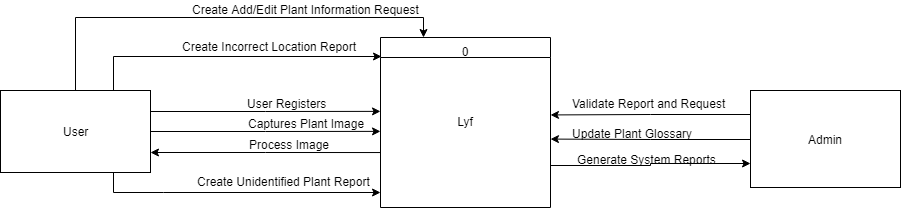
**APPENDICES**

**Use Case Diagram**

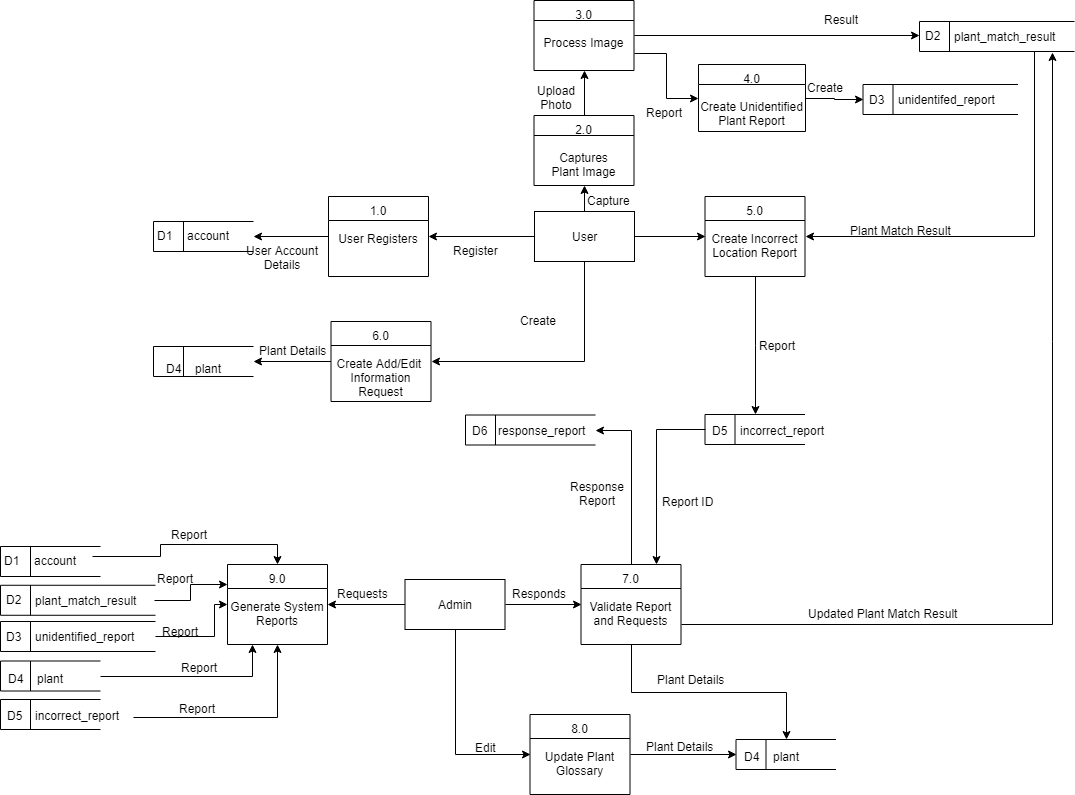
****

*Figure 51 Use Case Diagram*

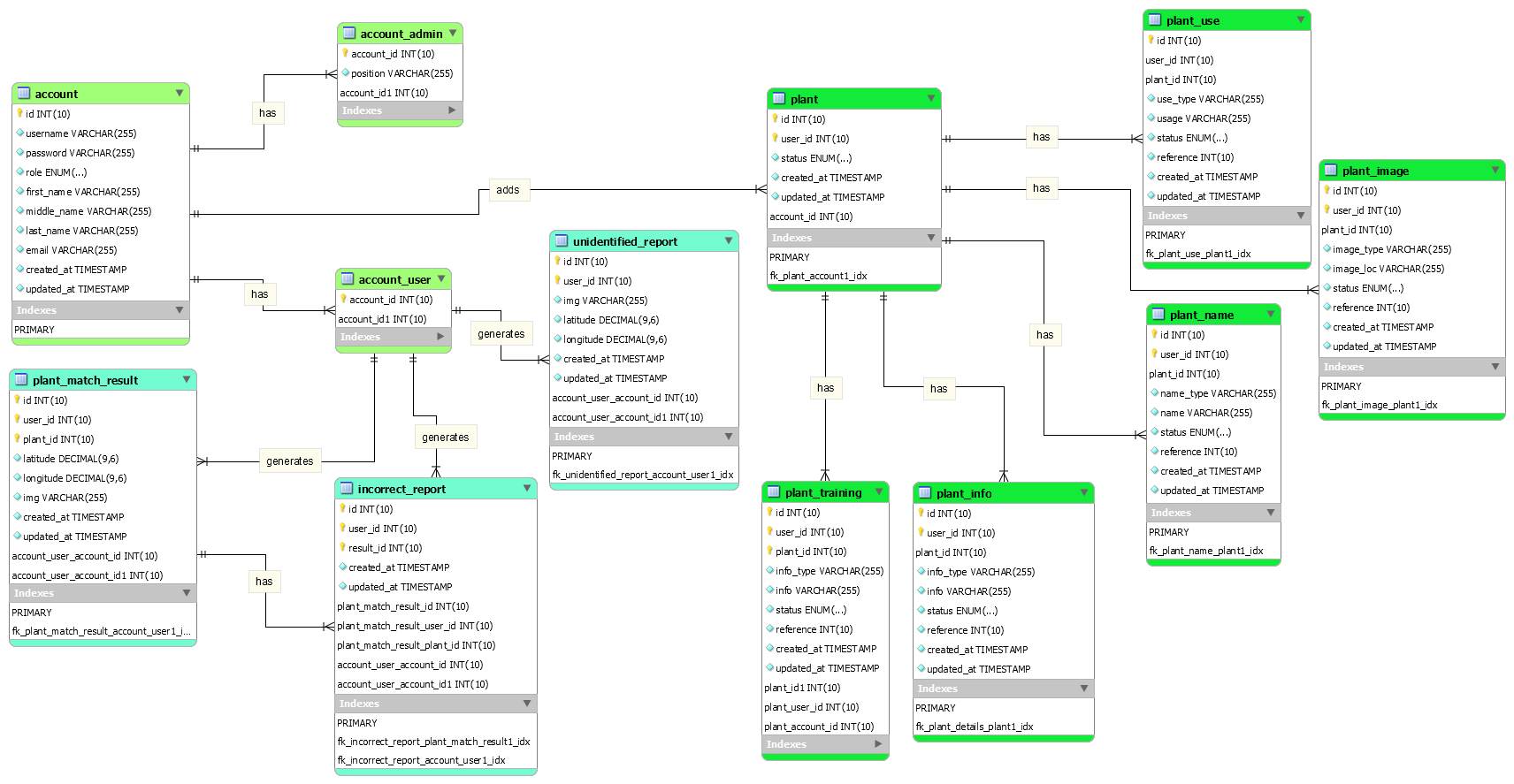
**Context Diagram**

****

*Figure 52 Context Diagram*

**Data Flow Diagram Level 0**

*Figure 53 Data Flow Diagram Level 0*

**Entity Relationship Diagram**

*Figure 54 Entity Relationship Diagram*

**Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| username | varchar | 255 | NN | Yes |  |
| password | varchar | 255 | NN | Yes |  |
| role | enum | ‘Admin’, ‘User’ | NN | Yes |  |
| first\_name | varchar | 255 | NN | Yes |  |
| middle\_name | varchar | 255 |  |  |  |
| last\_name | varchar | 255 | NN | Yes |  |
| email | varchar | 255 | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | Timestamp |  |  |  |  |

**Table: account**

*Figure 55 Data Dictionary – account table*

**Table: account\_user**

*Figure 56 Data Dictionary – account\_user table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | FieldLength/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| position | varchar | 255 | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

**Table: account\_admin**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | FieldLength/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

*Figure 57 Data Dictionary – account\_admin table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| status | enum | ‘Add’,’Verified’,’Deleted’ | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

**Table: plant**

*Figure 58 Data Dictionary –plant table*

**Table: plant\_info**

*Figure 59 Data Dictionary –plant\_info table*

**Table: plant\_image**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| plant\_id | Int | 10 | NN | Yes | plant |
| image\_type | varchar | 255 | NN | Yes |  |
| image\_loc | varchar | 255 | NN | Yes |  |
| status | Enum | ‘Add’,’Verified’,’Deleted’ | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

*Figure 60 Data Dictionary –plant\_image table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| plant\_id | Int | 10 | NN | Yes | plant |
| info\_type | varchar | 255 | NN | Yes |  |
| Info | varchar | 255 | NN | Yes |  |
| status | Enum | ‘Add’,’Edit’,’Verified’,’Deleted’ | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| plant\_id | Int | 10 | NN | Yes | plant |
| name\_type | varchar | 255 | NN | Yes |  |
| name | varchar | 255 | NN | Yes |  |
| status | Enum | ‘Add’,’Edit’,’Verified’,’Deleted’ | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

**Table: plant\_name**

*Figure 61 Data Dictionary –plant\_name table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| plant\_id | Int | 10 | NN | Yes | plant |
| latitude | decimal | 9,6 | NN | Yes |  |
| longitude | decimal | 9,6 | NN | Yes |  |
| img | varchar | 255 | NN | Yes |  |
| status | enum | ‘Verified’,’Deleted’ | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

**Table: plant\_match\_result**

*Figure 62 Data Dictionary –plant\_match\_result table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| latitude | decimal | 9,6 | NN | Yes |  |
| longitude | decimal | 9,6 | NN | Yes |  |
| img | varchar | 255 | NN | Yes |  |
| status | Enum | ’Verified’,’Deleted’ | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

**Table: unidentified\_report**

*Figure 63 Data Dictionary –unidentified\_report table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| user\_id | Int | 10 | NN | Yes | account |
| result\_id | Int | 10 | NN | Yes | plant\_match\_result |
| status | Enum | ’Verified’,’Deleted’ | NN | Yes |  |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

**Table: incorrect\_report**

*Figure 64 Data Dictionary –incorrect\_report table*

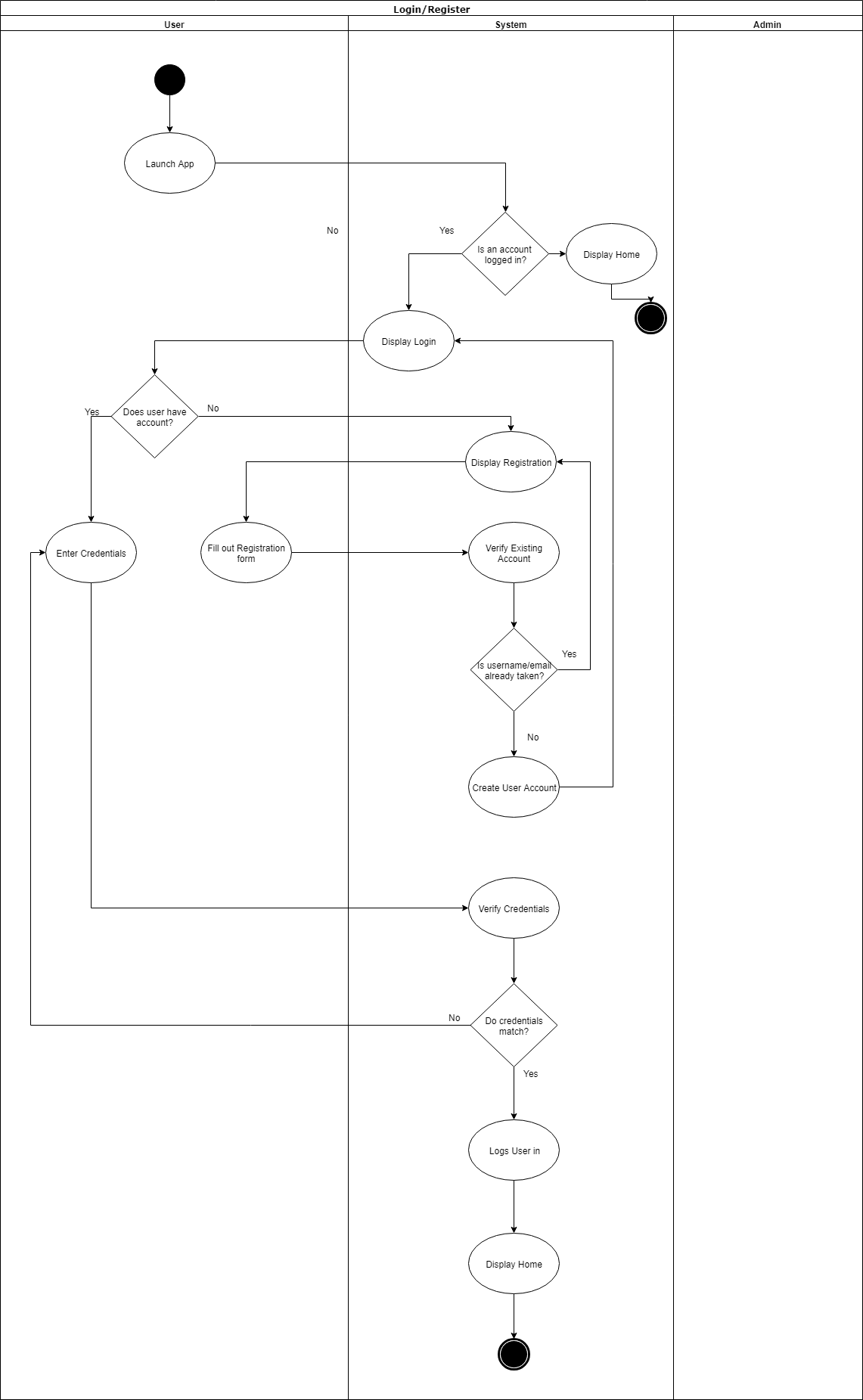
**Table: plant\_training**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Field Length/Values | Constraint | Required | Reference Table |
| id | Int | 10 | PK/NN | Yes |  |
| plant\_id | Int | 10 | NN | Yes | account |
| training\_id | Int | 10 | NN | Yes | plant\_match\_result |
| created\_at | timestamp |  | NN | Yes |  |
| updated\_at | timestamp |  |  |  |  |

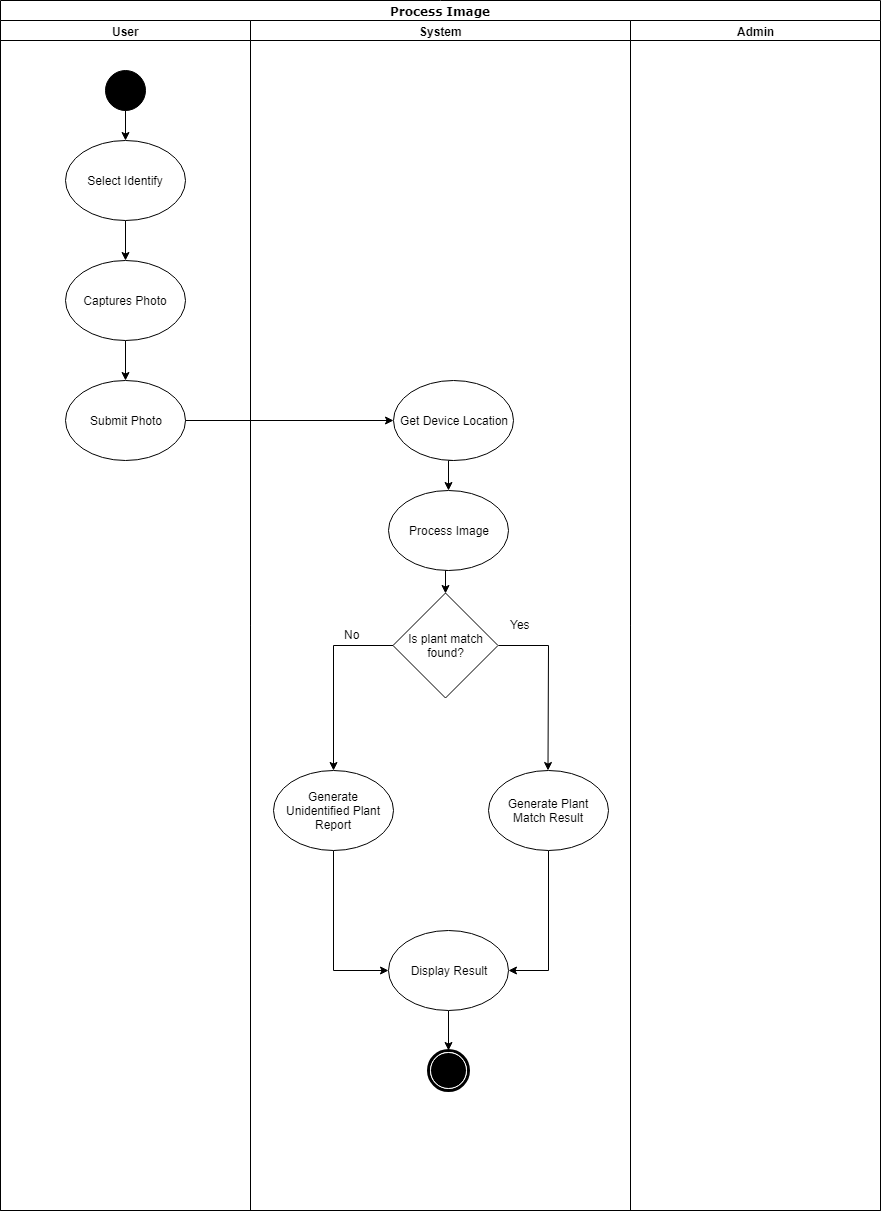
*Figure 65 Data Dictionary –plant\_training table*

**Activity Diagram**

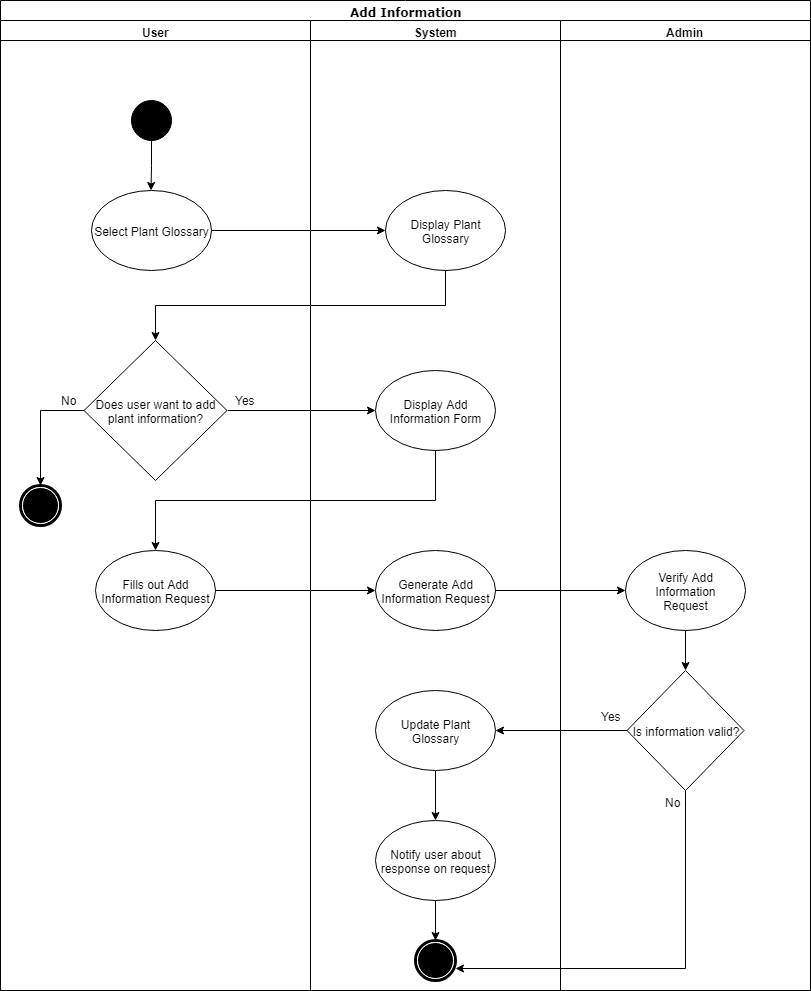
**Login/Register**

****

*Figure 66 Activity Diagram – Login/Register*

**Process Image**

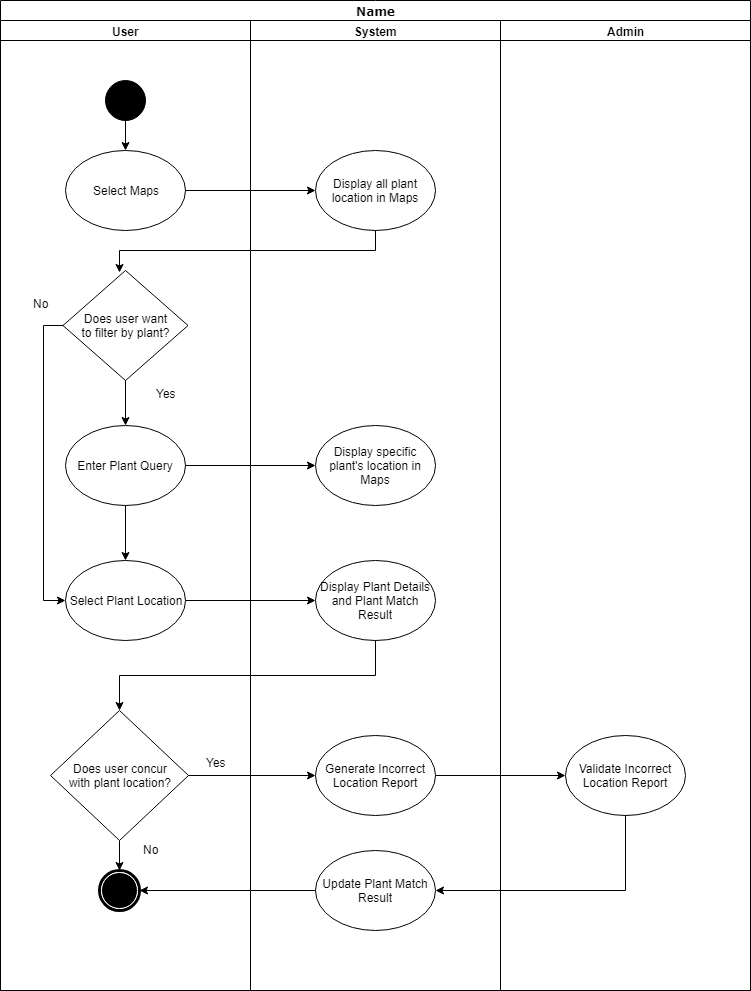
*Figure 67 Activity Diagram – Process Image*

**Add Information**

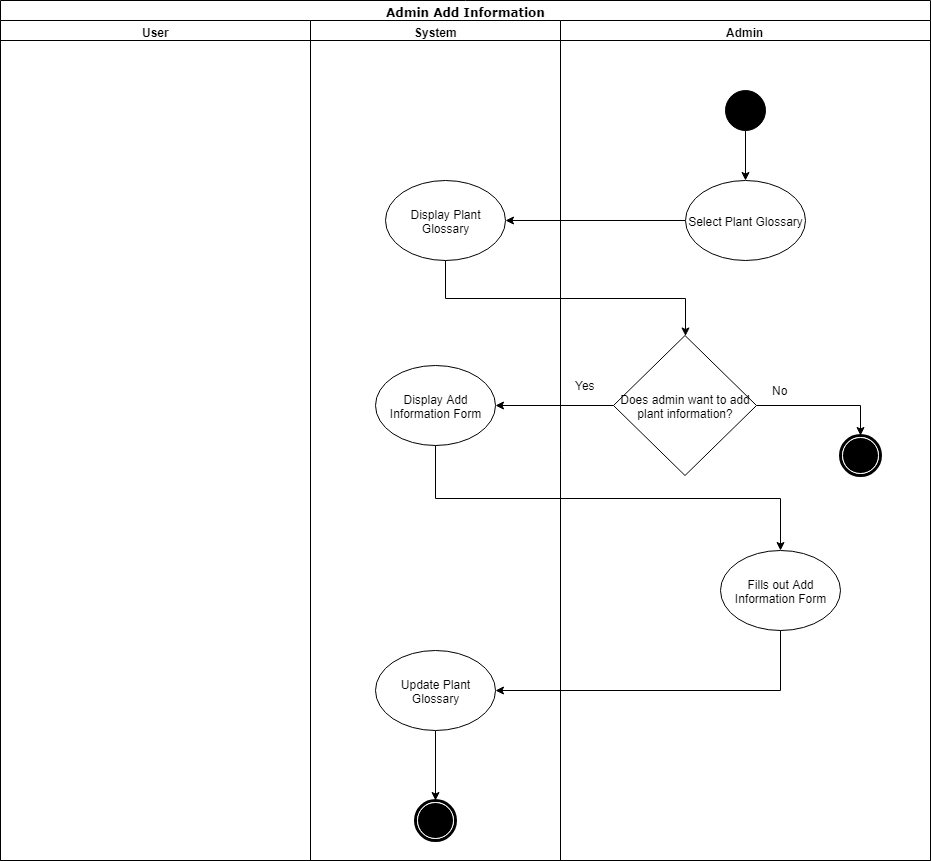
*Figure 68 Activity Diagram - Add Information*

**Edit Information**

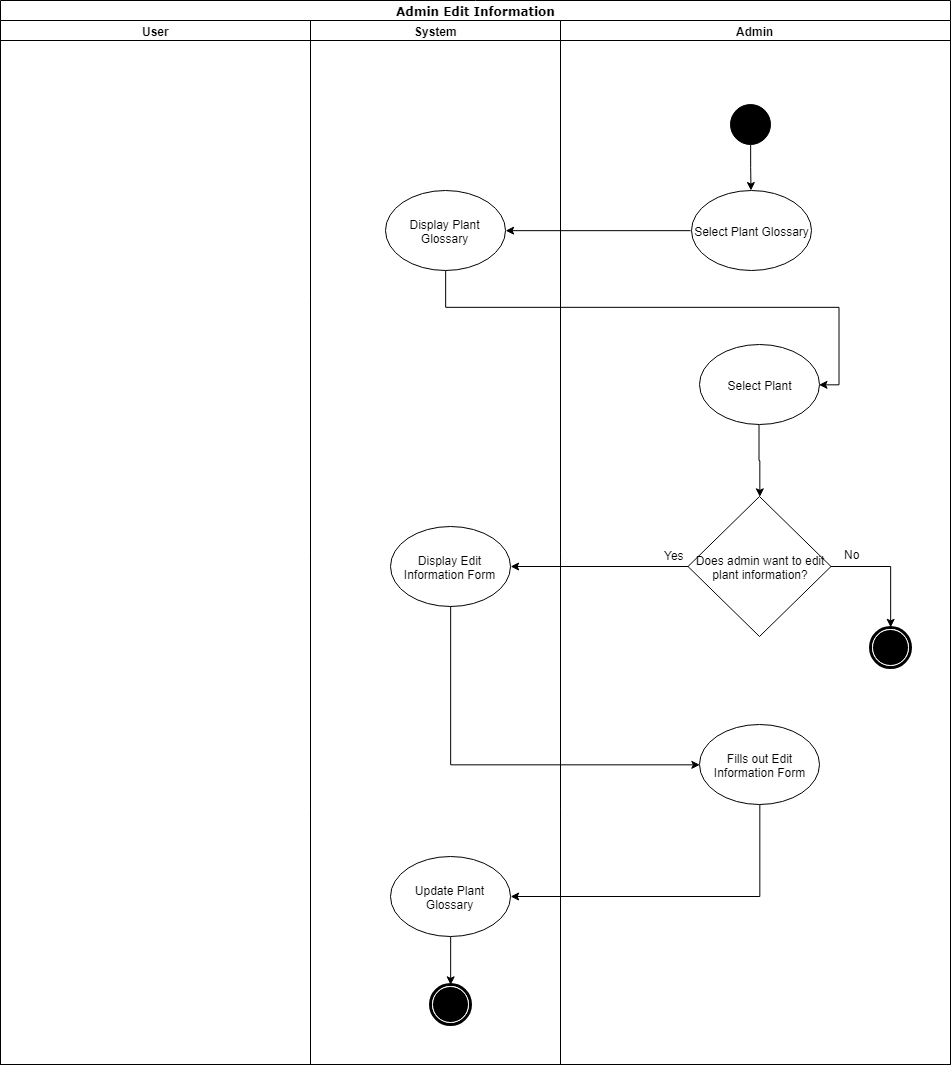
*Figure 69 Activity Diagram - Edit Information*

**Browse Location**

*Figure 70 Activity Diagram – Browse Location*

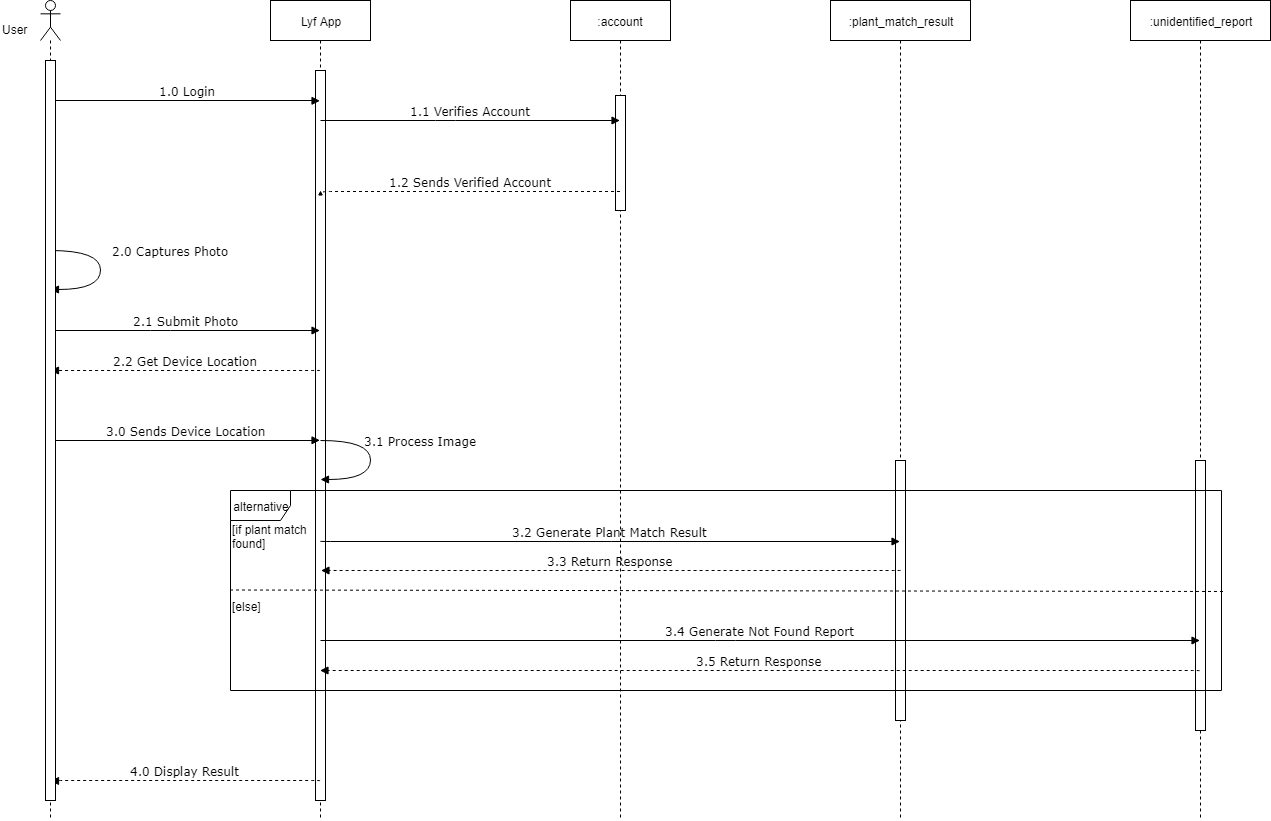
**Admin Add Information**

*Figure 71 Activity Diagram – Admin Add Information*

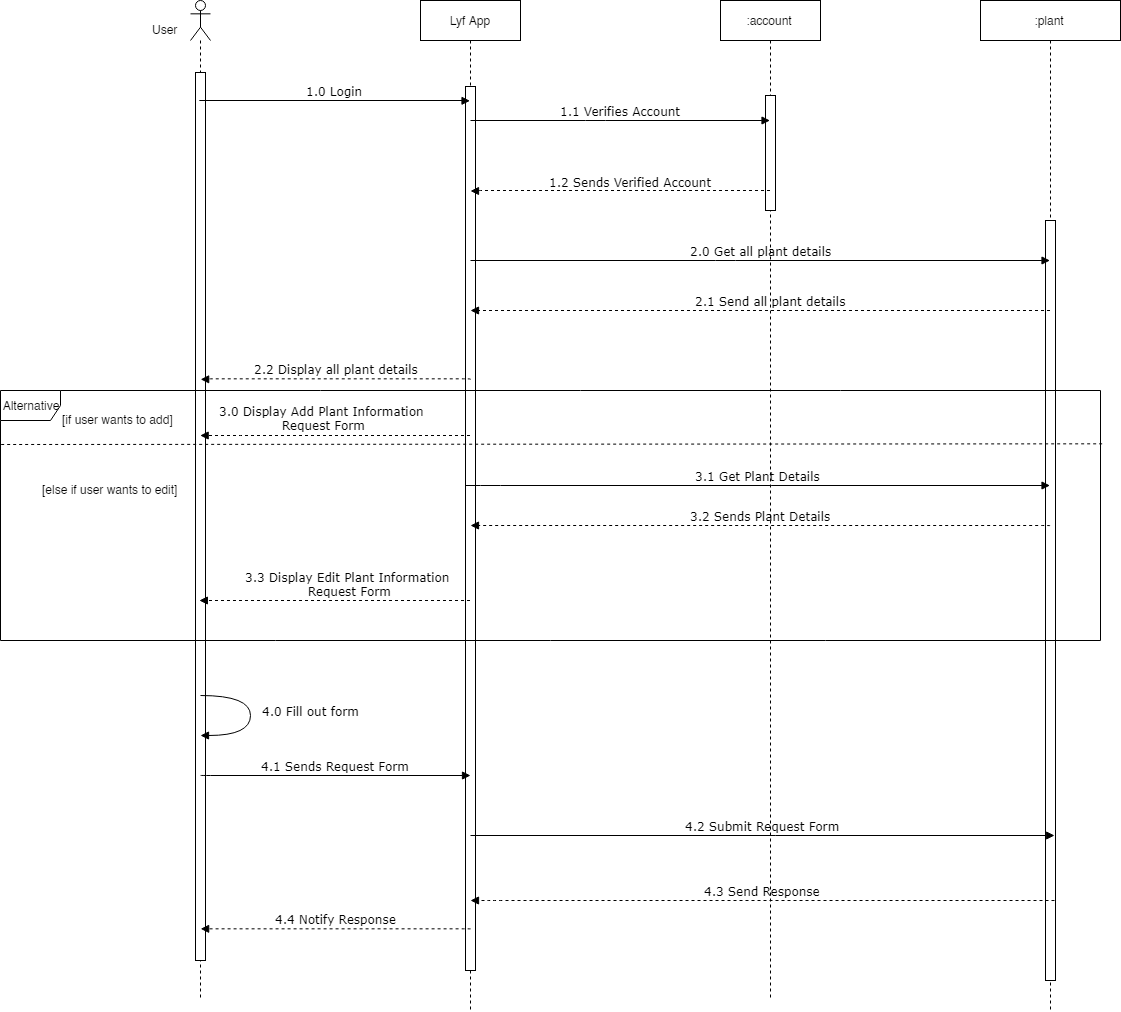
**Admin Edit Information**

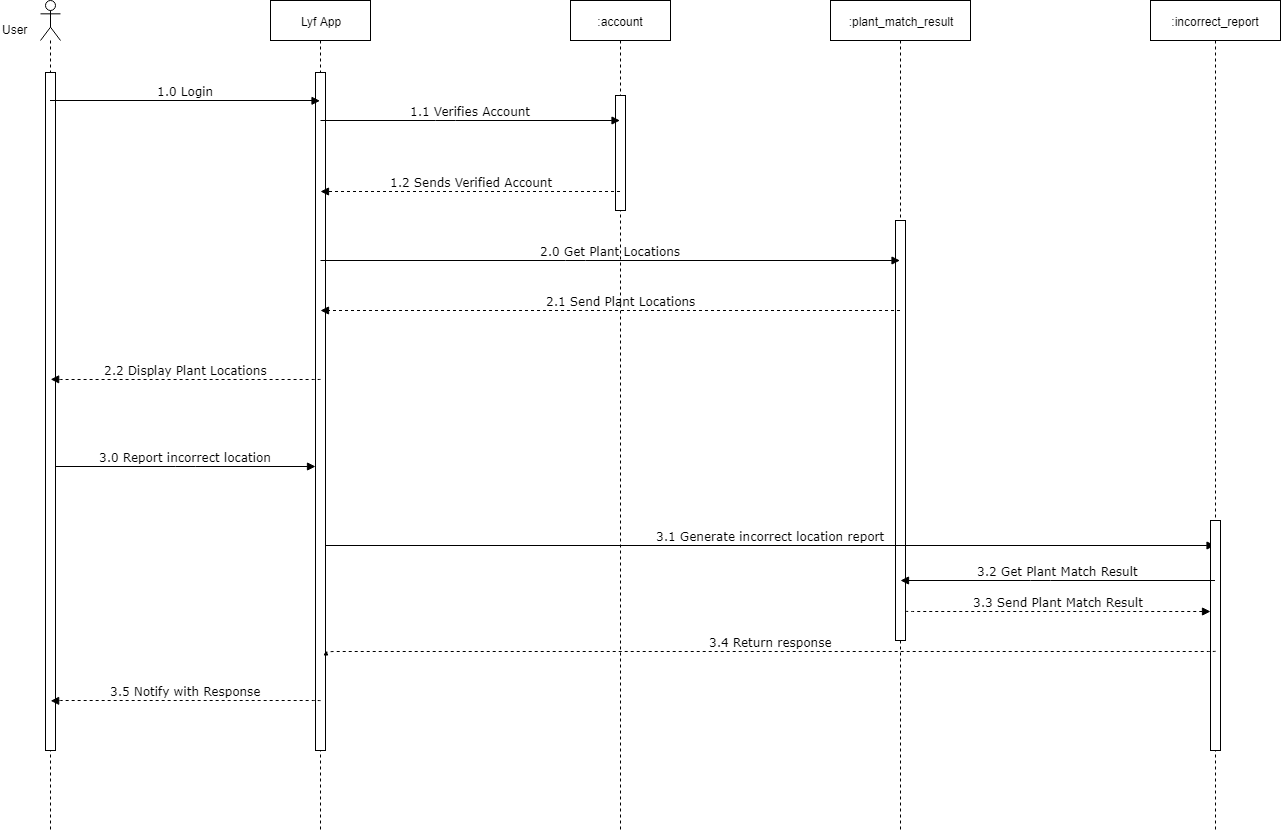
*Figure 72 Activity Diagram – Admin Edit Information*

**System Sequence Diagram**

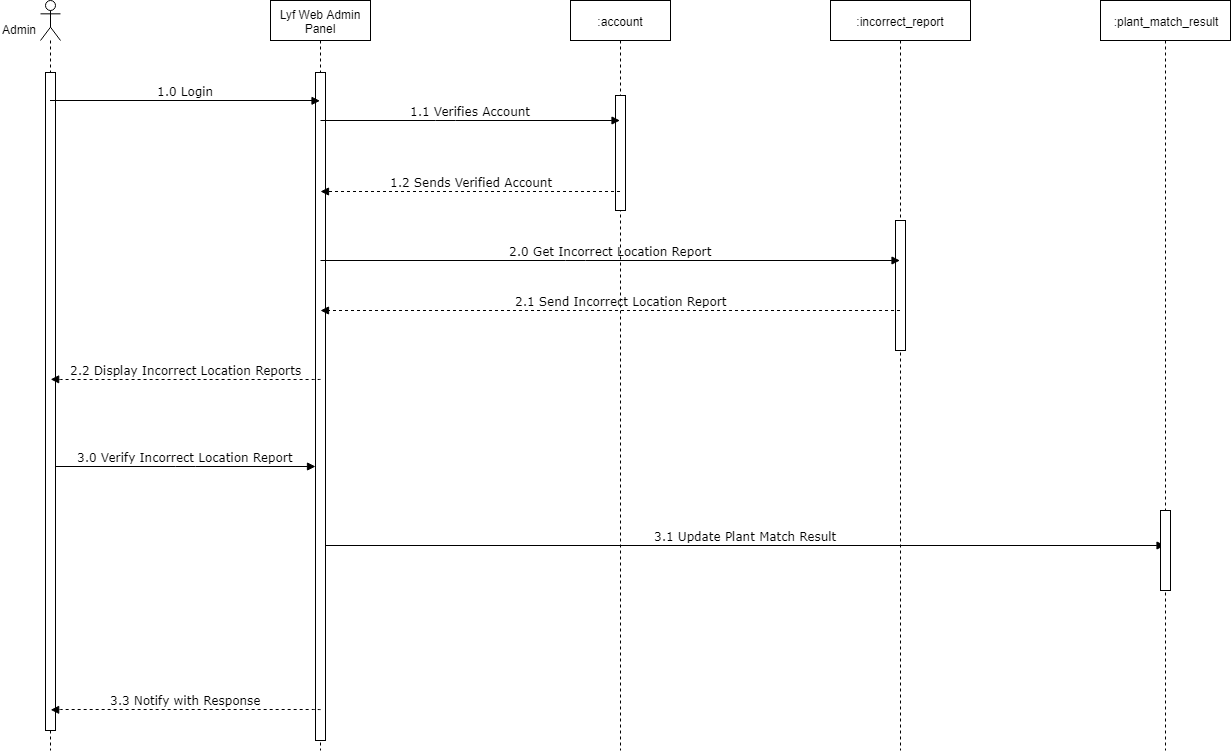
**Process Image**

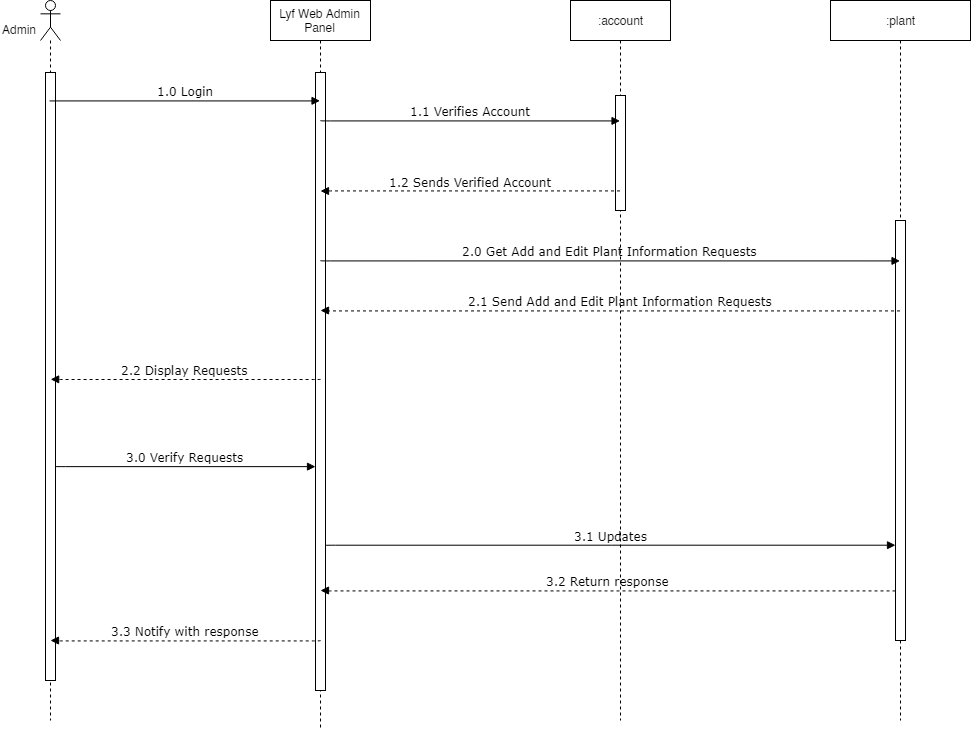
*Figure 73 System Sequence Diagram – Process Image*

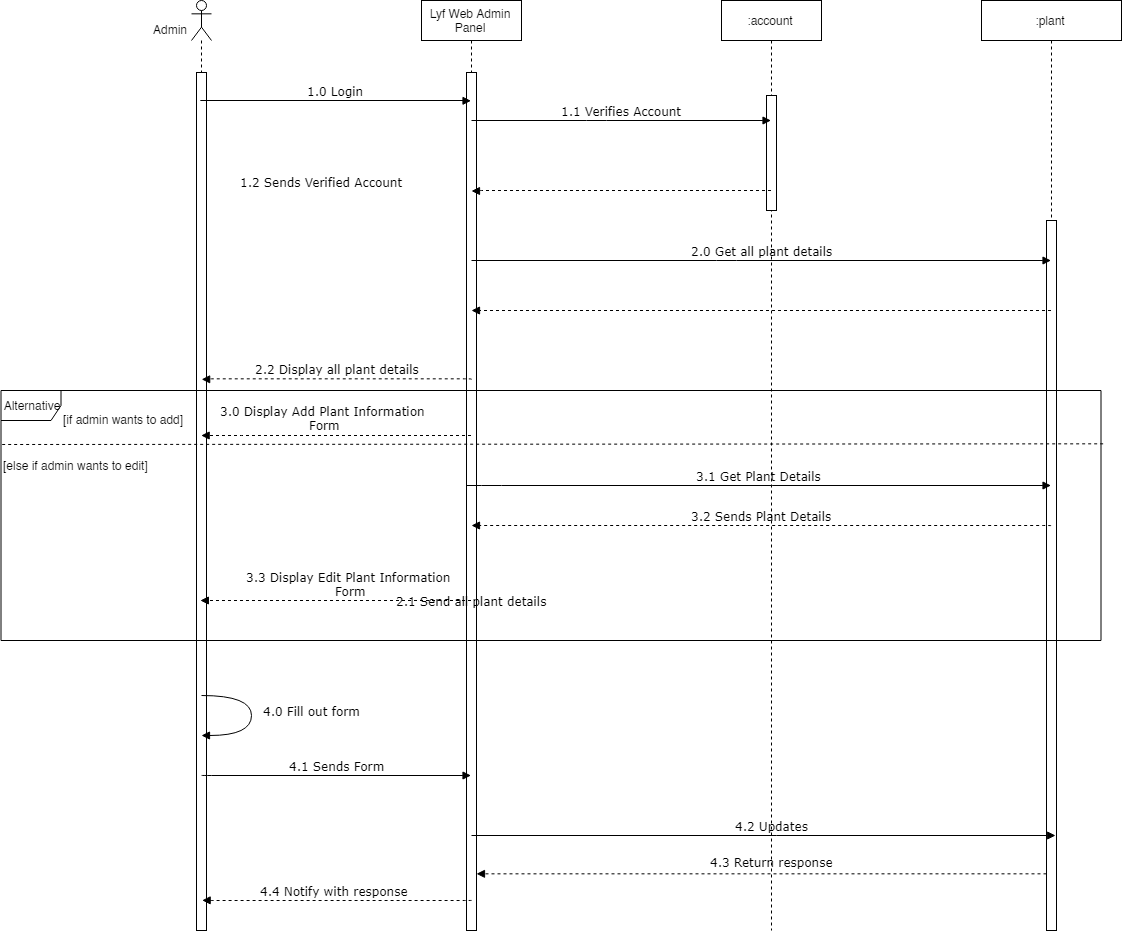
**User Add/Edit Information**

**Report Incorrect Location**

**Verify Incorrect Location**

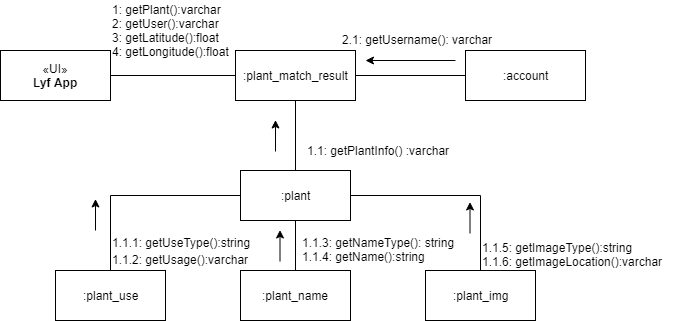
****

**Admin Verify Requests**

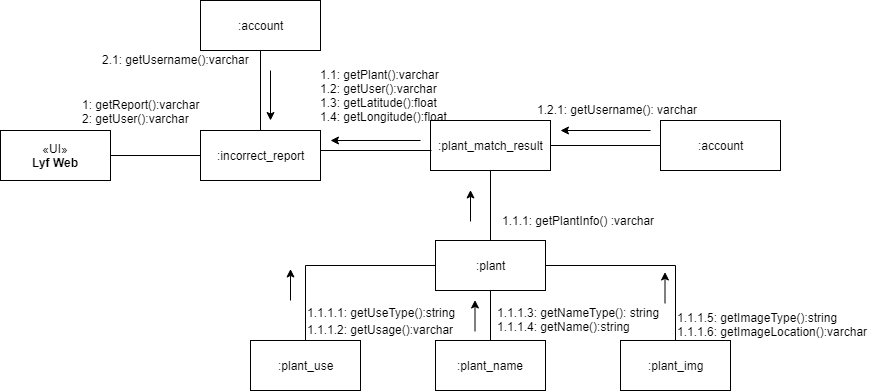
**Admin Add/Edit Information**

**Communication Diagram**

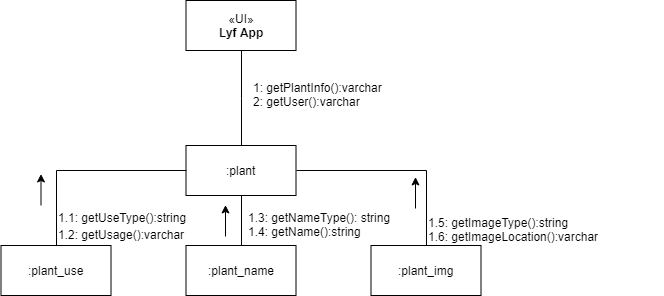
**Location and Results**

****

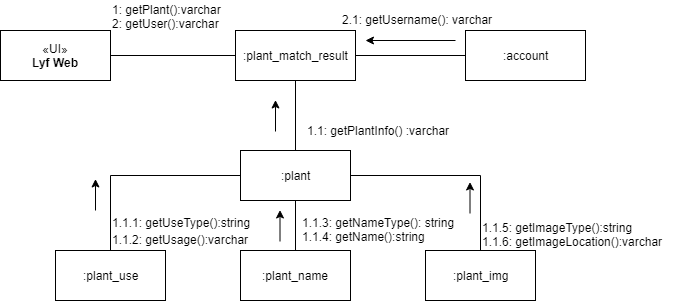
*Figure 79 Communication Diagram – Location and Results*

****Incorrect Location**

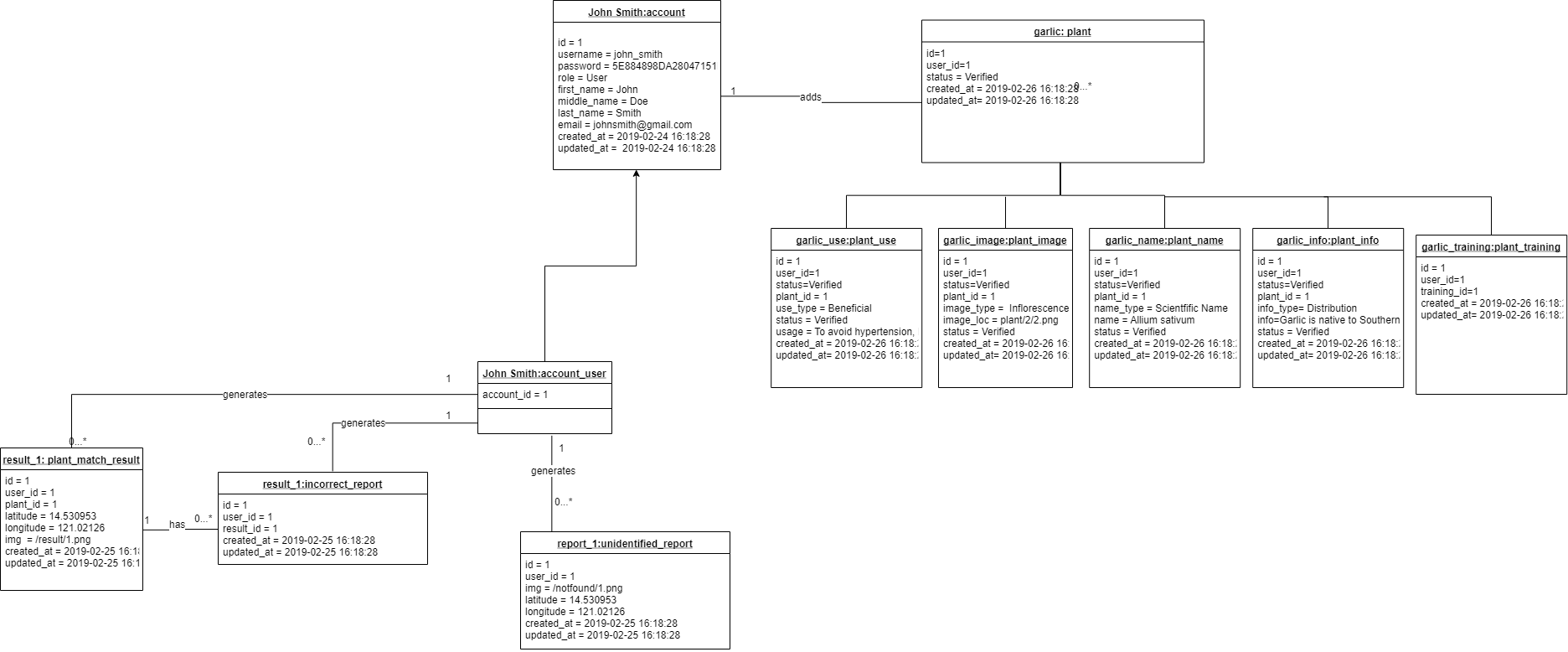
*Figure 80 Communication Diagram – Incorrect Location*

****Plant**

*Figure 81 Communication Diagram – Plant*

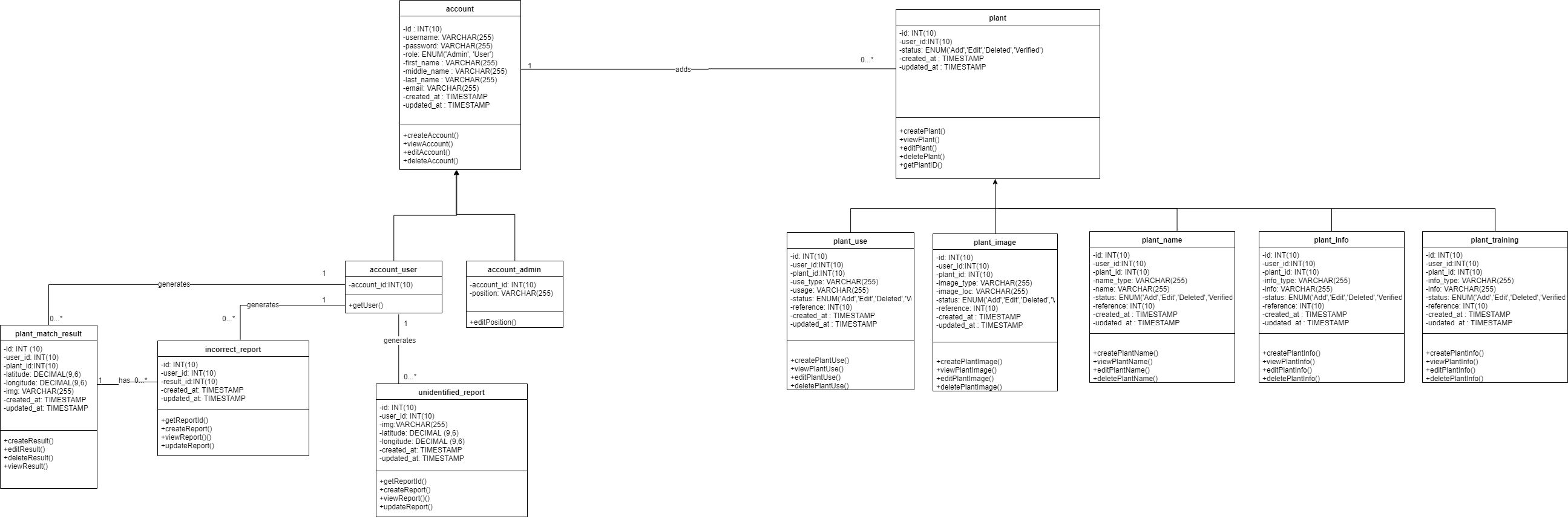
**Plant Match Results**

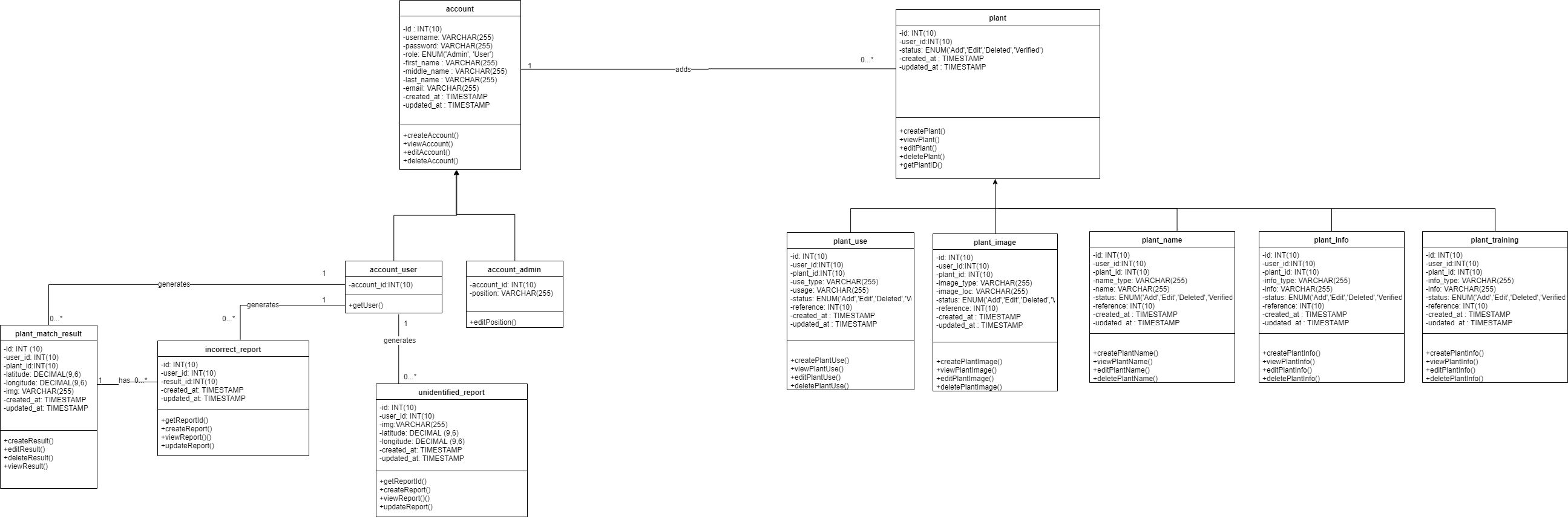
*Figure 82 Communication Diagram – Plant Match Results*

**Object Diagram**

*Figure 83 Object Diagram*

**Class Diagram**

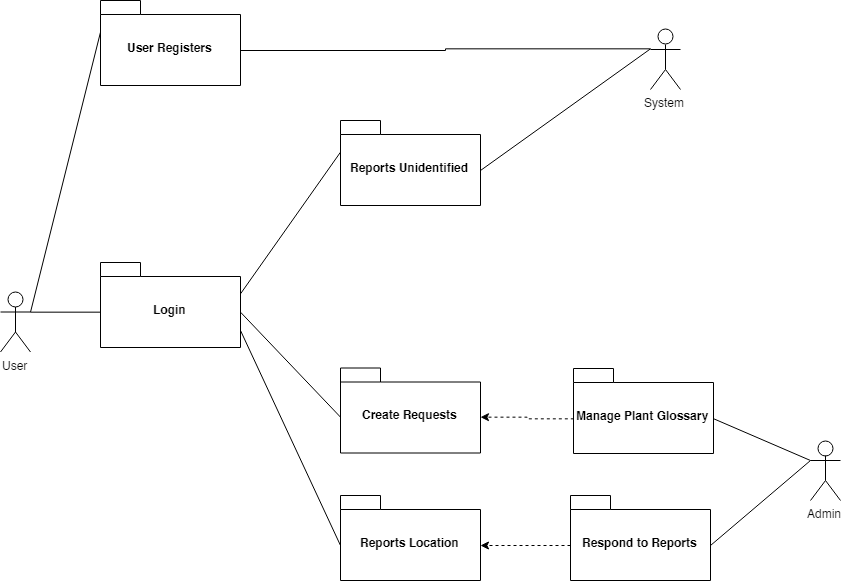
****



*Figure 83 Class Diagram*

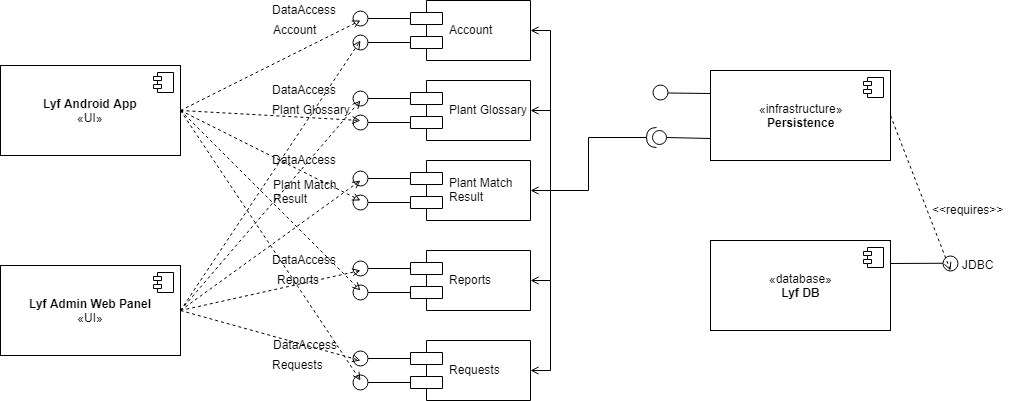
**Timing Diagram**

**Package Diagram**

****

*Figure 85 Package Diagram*

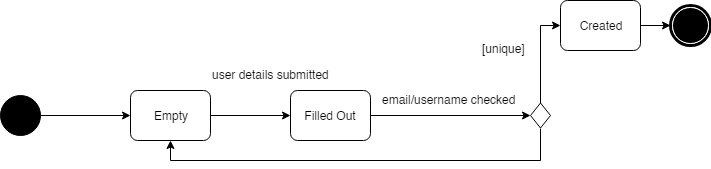
**Component Diagram**

****

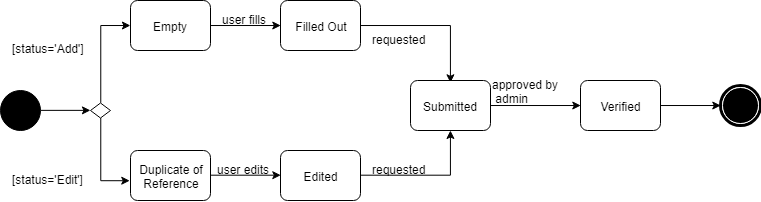
*Figure 86 Component Diagram*

**State Machine Diagram**

**Account**

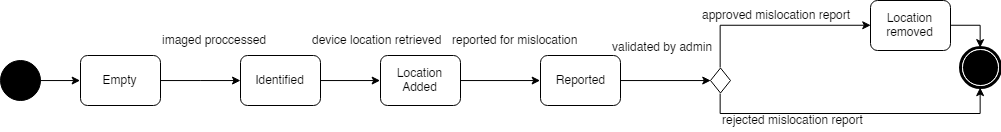
****

*Figure 87 State Machine Diagram – Account*

**Plant**

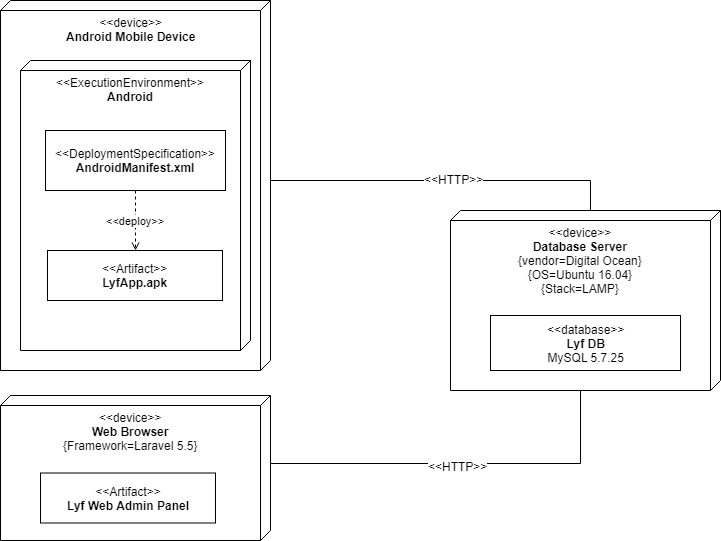
*Figure 88 State Machine Diagram – Plant*

**Plant Match Result**

****

*Figure 89 State Machine Diagram – Plant Match Result*

**Deployment Diagram**

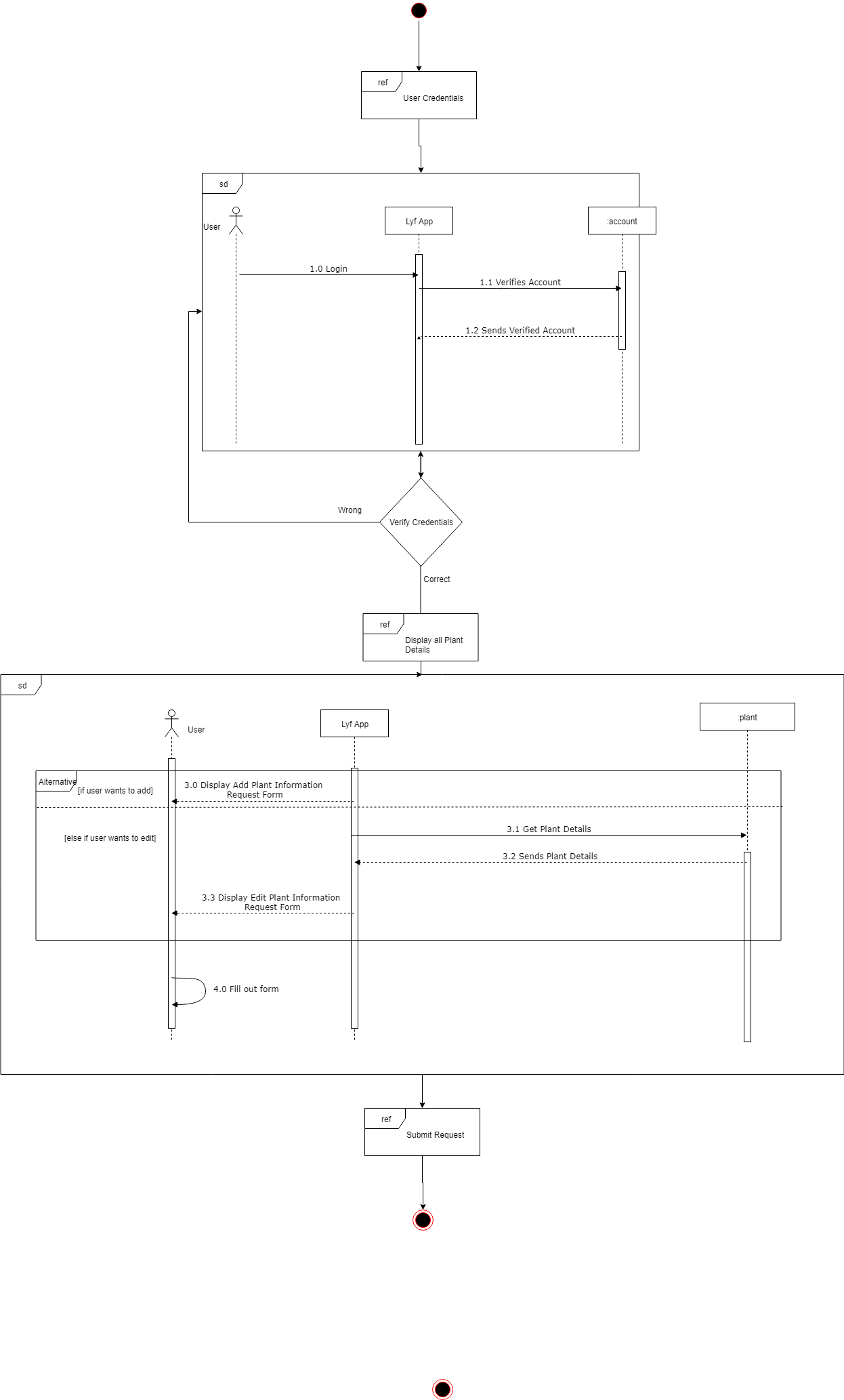
****

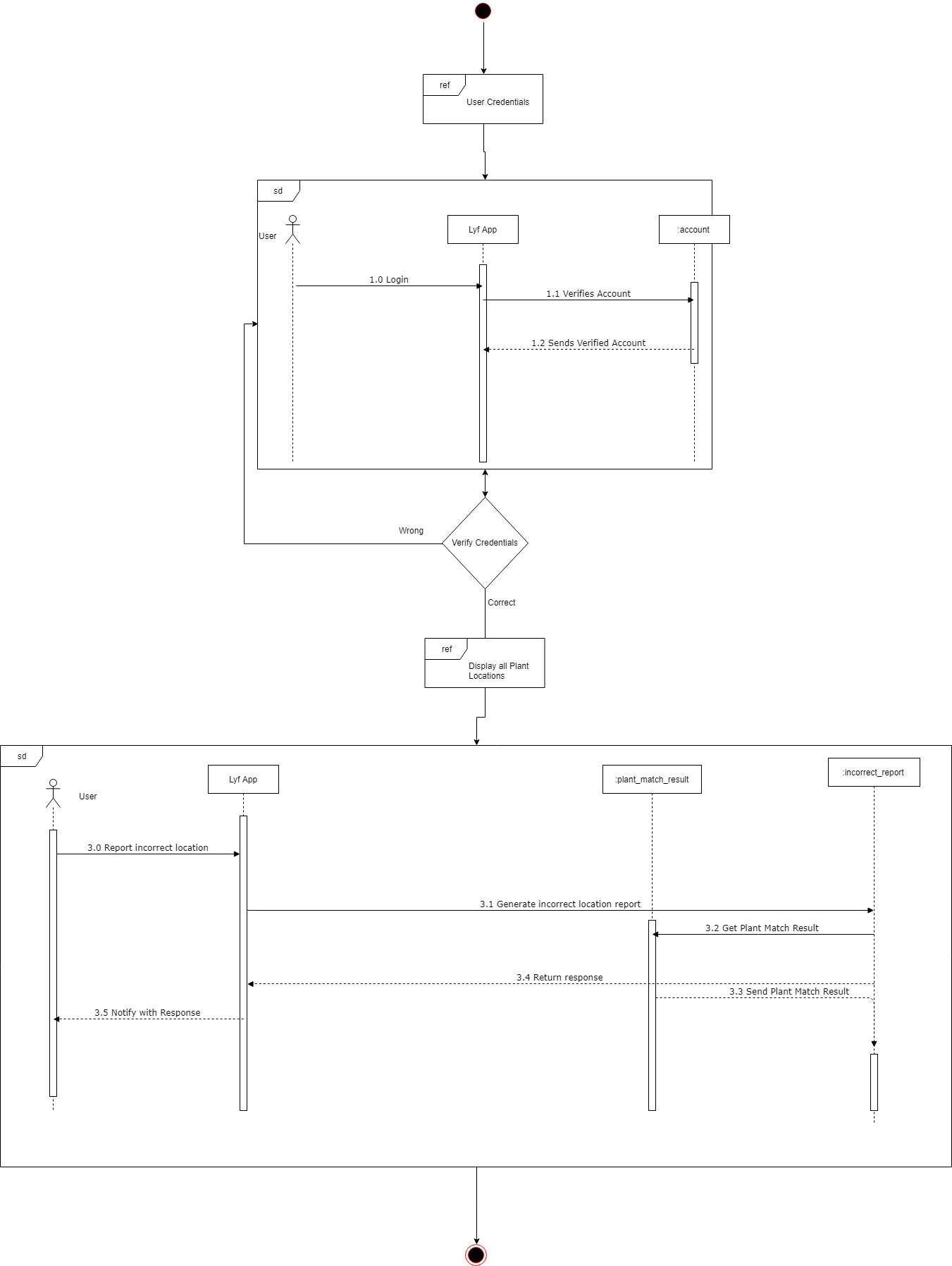
*Figure 90 Deployment Diagram*

**Interaction Diagram**

**A close up of a map

Description automatically generatedProcess Image**

**Add/Edit Information**

**Report Incorrect Location**

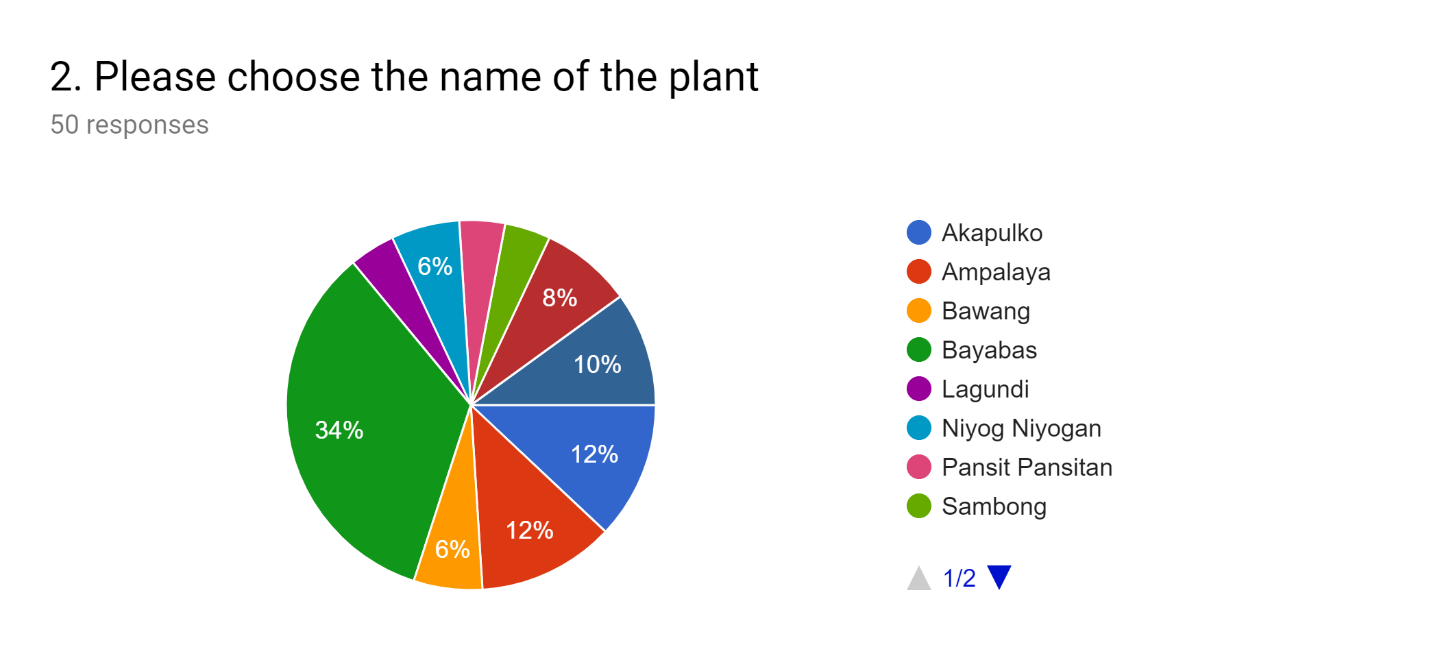
*Figure 93 Interaction Diagram – Report Incorrect Location*

**Survey Results**

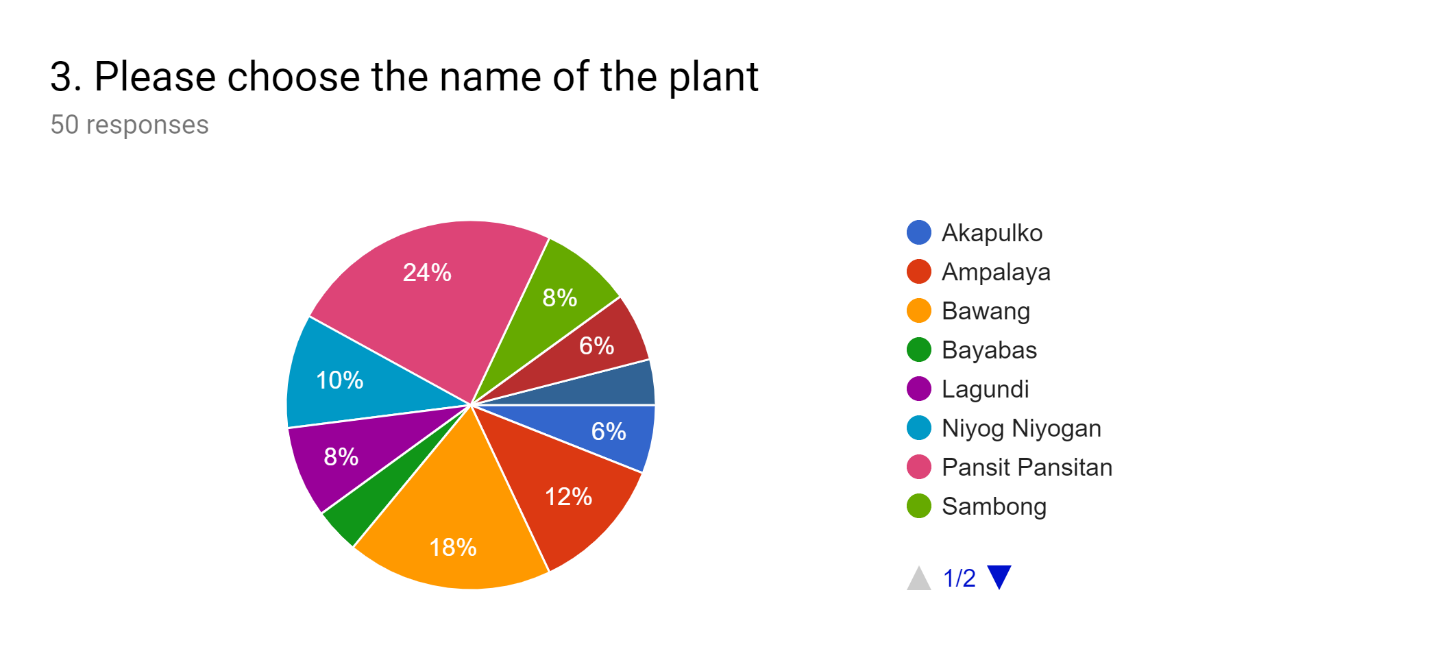
**Correct Answer: Akapulko 32%**

****

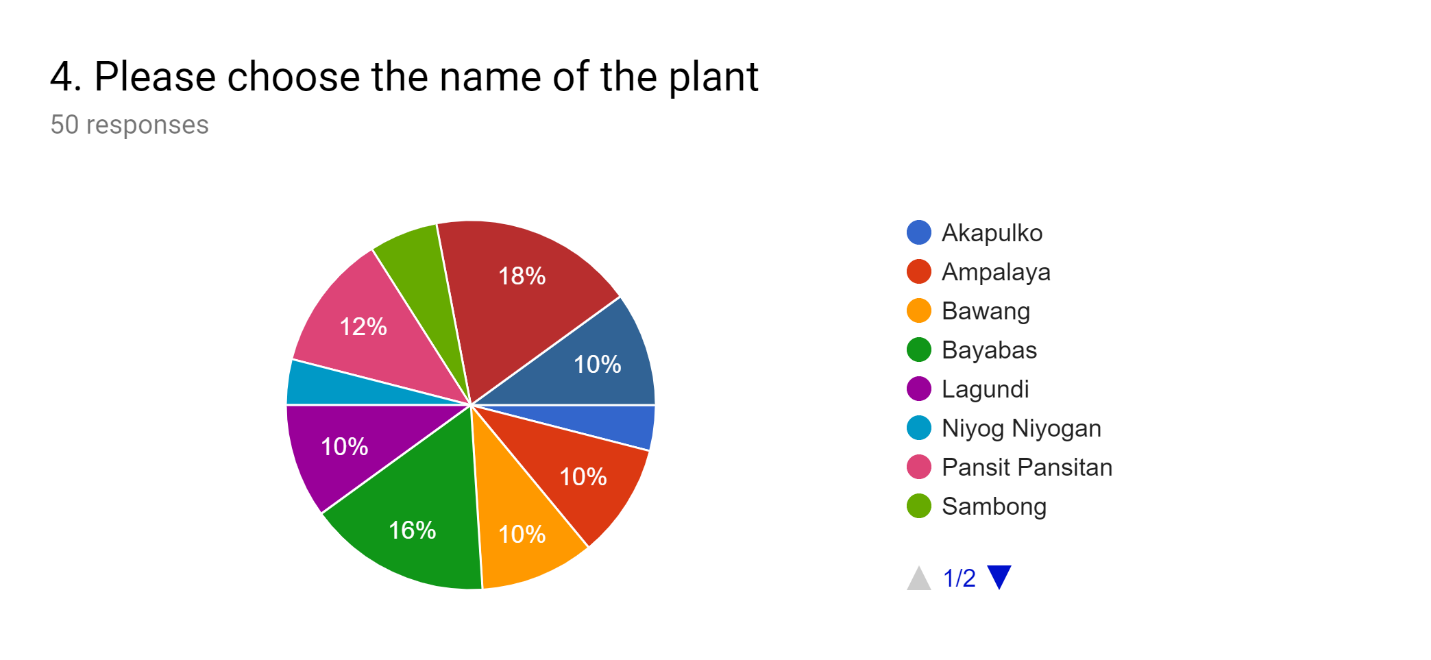
**Correct Answer: Bayabas 34%**

****

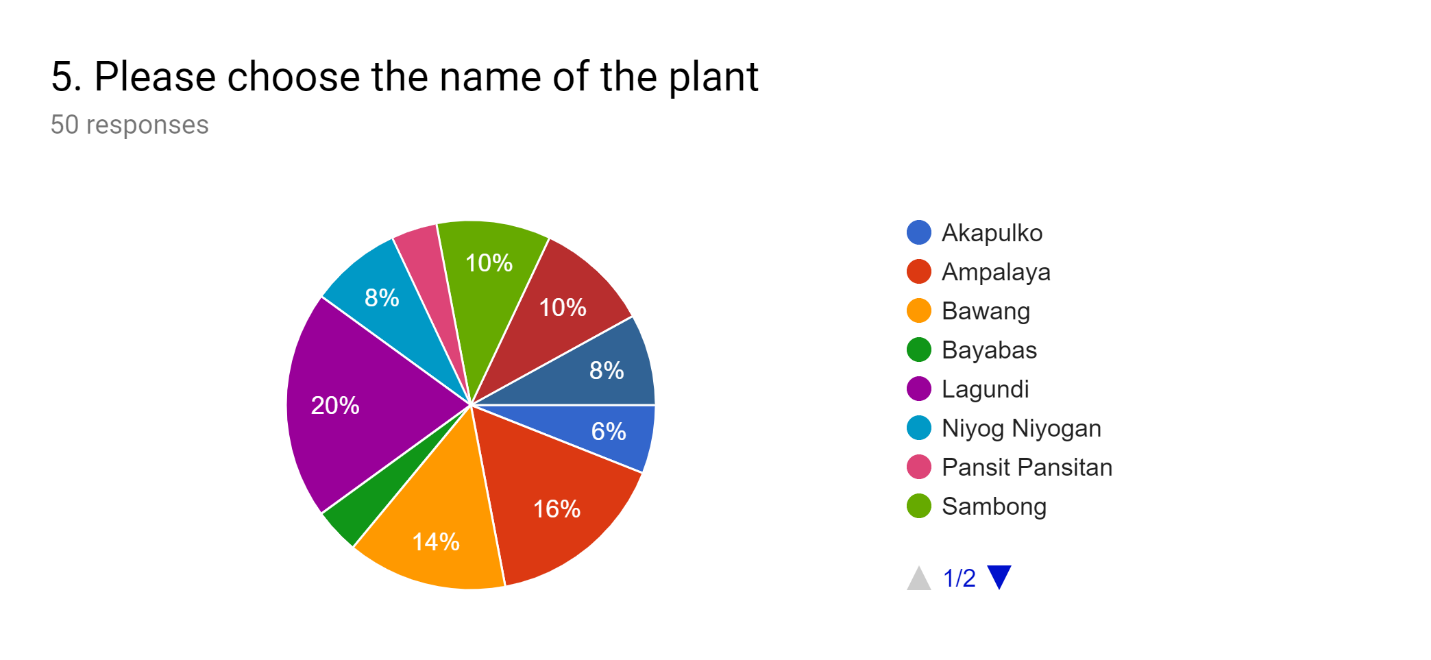
**Correct Answer: Pansit Pansitan 24%**

****

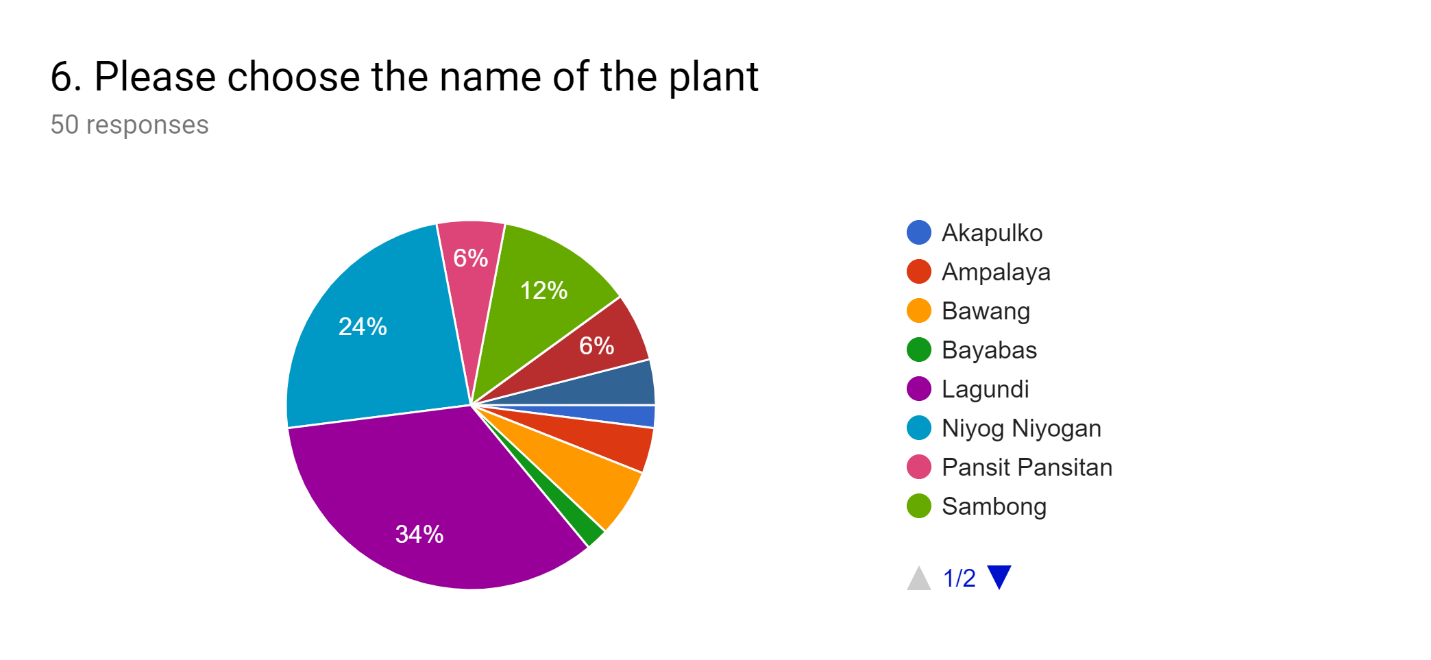
**Correct Answer: Sambong 16%**

****

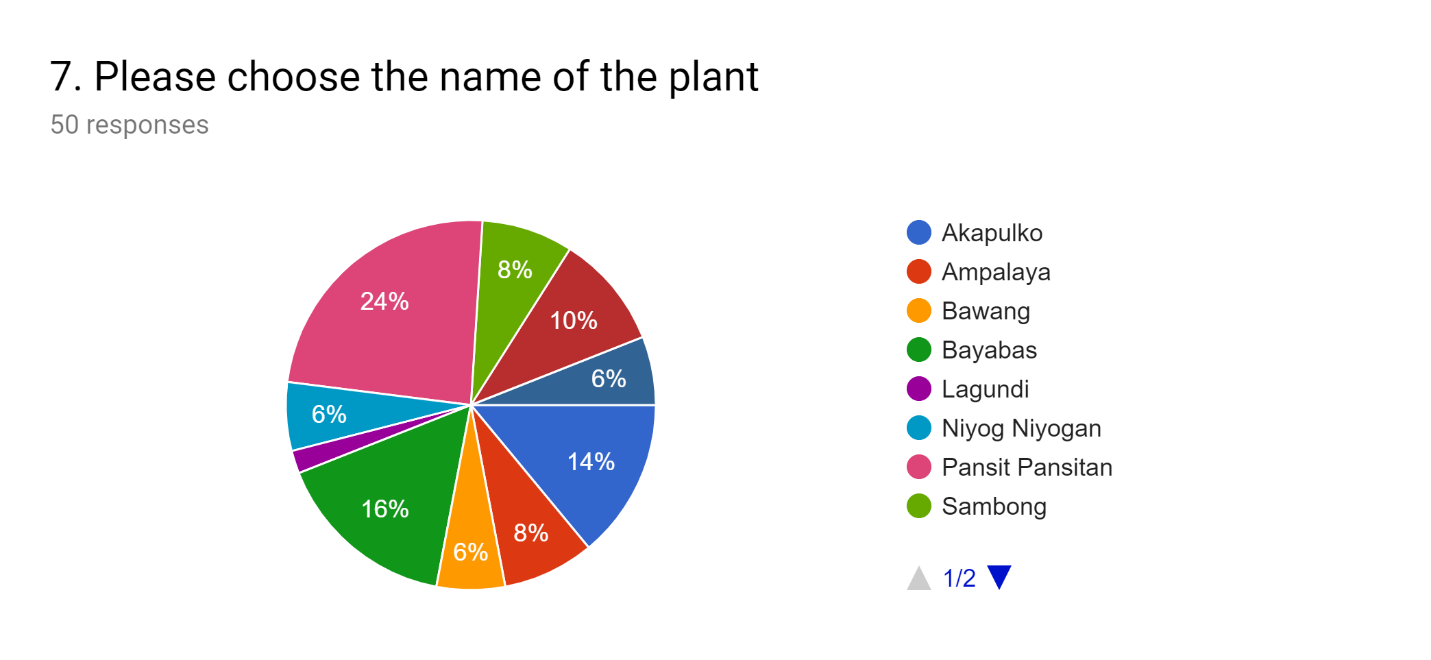
**Correct Answer: Ampalaya 16%**

****

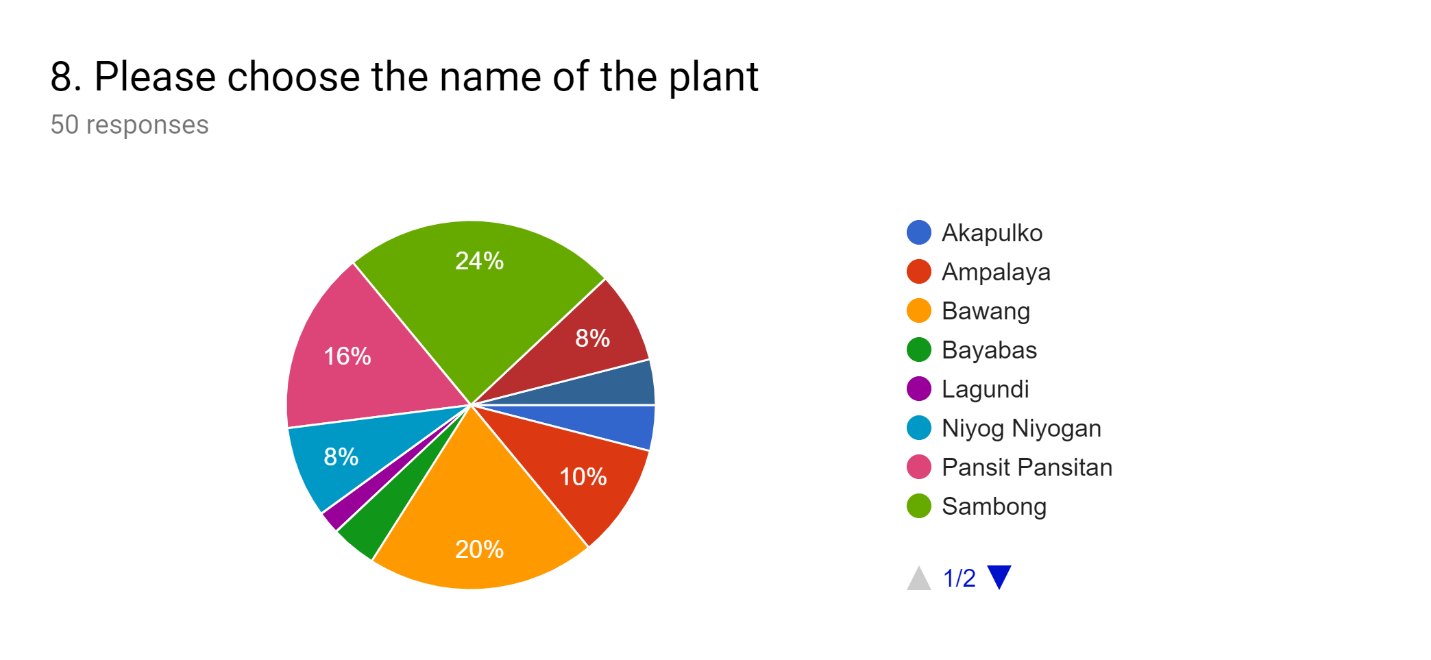
**Correct Answer: Lagundi 34%**

****

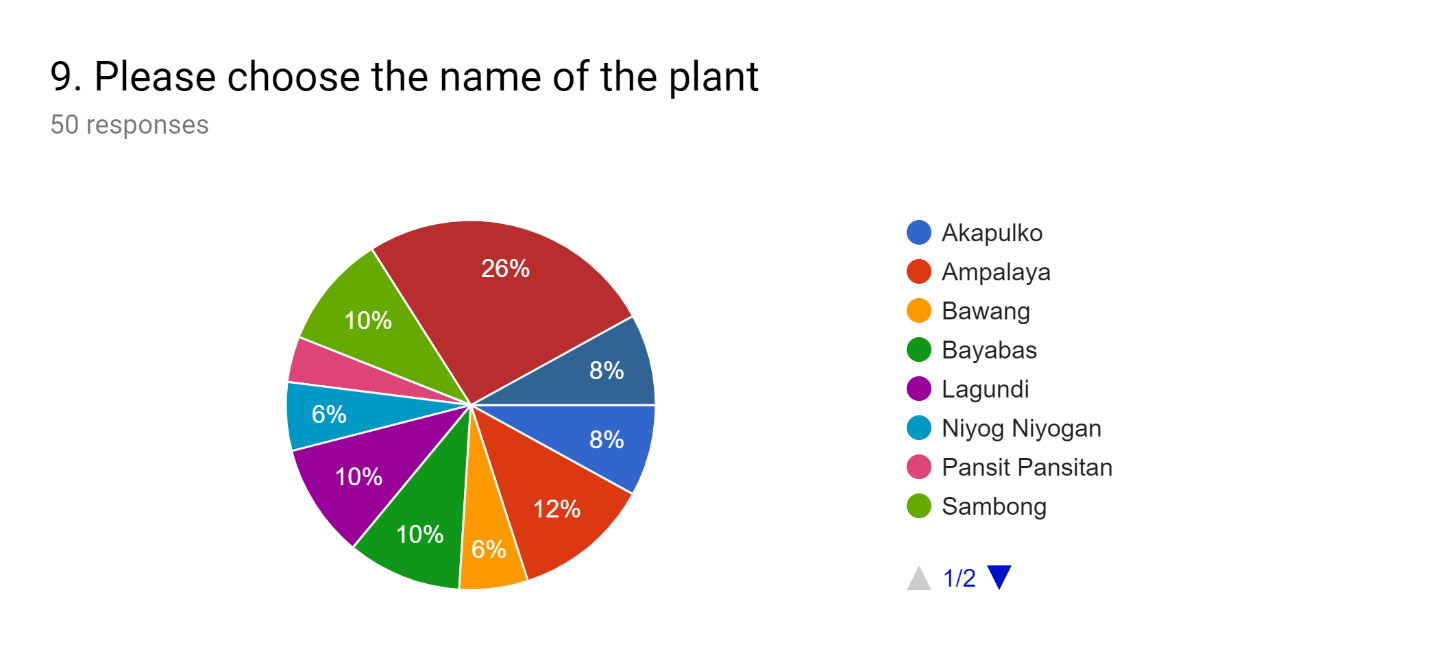
**Correct Answer: Tsaang Gubat 10%**

****

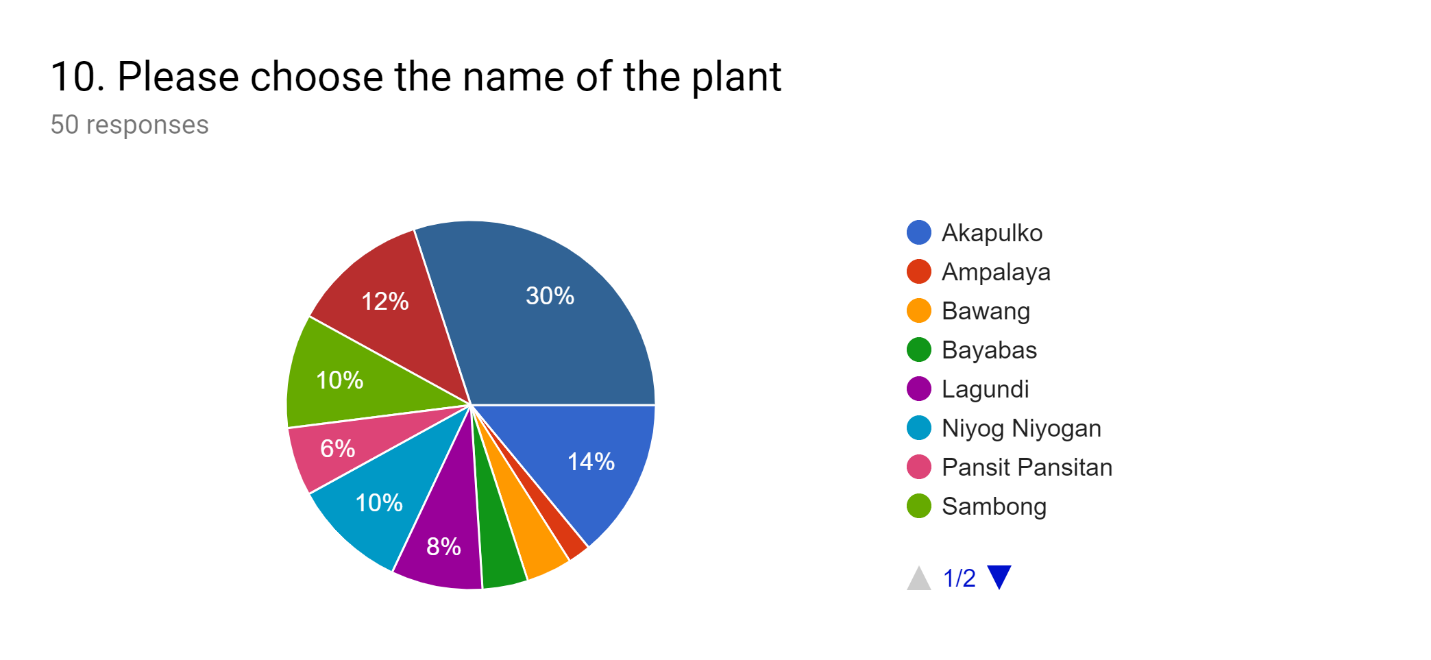
**Correct Answer: Bawang 20%**

****

**Correct Answer: Yerba Buena 8%**

****

**Correct Answer: Niyog Niyogan 5%**

****