Plant Care

MACHINE LEARNING BASED PLANT DISEASE DETECTION MOBILE APP FOR SRI LANKA

CSG3101.2 - Applied Project
Bachelor of Computer Science | U65
Edith Cowan University

Lecturer | Mr. Guhanathan Poravi

[redacted]

Table of Contents

Abstract

This document contains all documentation for a mobile app project created to diagnose plant diseases using Artificial intelligence & image recognition to help farmers in Sri Lanka to identify harmful pests and diseases, in the aim that it will help protect Sri Lankan farmlands & the Sri Lankan ecosystem.

CHAPTER 1: INTRODUCTION

2.1 Overview of Chapter

The main goal of this project is to design, develop and evaluate a machine learning based mobile application, how it was conceived, the problem at hand, and its background will all be explored in this chapter to delve deeper into the reason for the project and the relevance of its development. The backdrop of the project, its definition of the problem, its goals, its scope, and the requirements and tools required to make the project a reality are covered in detail.

2.2 Background

Since the economic crisis, Sri Lanka's agricultural production has significantly decreased, increasing the cost of fuel and many other necessities, making food unaffordable for most of the population. As a result, the government has predicted food shortages over the next few years since the gov failed to provide domestic production of organic pesticides and fertilizers as promised following the ban on chemical fertilizers and pesticides, a decision that ruined crop yields (How a Fertilizer Ban Became a Part of Sri Lanka's Crisis, 2022). As a result, Sri Lanka's once self-sufficient production of average rice yields (which is a staple grain) fell. Additionally, the UN World Food Programme had designated 2022 to be a year of "unprecedented hunger," which will be exacerbated by pandemics, socioeconomic situations, natural disasters, climate change, and most importantly, pests and illnesses that will affect developing nations the most. Internationally, the risk of famine is at an all-time high. Sri Lanka is working to alleviate its food deficit by maintaining stable prices, making sure farmers receive fertilizer on schedule, and requiring all households to engage in agriculture whenever possible (Food Crisis in Sri Lanka Likely to Worsen amid Poor Agricultural Production, Price Spikes and Ongoing Economic Crisis, FAO and WFP Warn | World Food Programme, 2022). However, more research and development in agriculture is needed to make it more exact and predictable, with better fertilizer and pesticides to cure pests and prevent diseases. Although the plant pathology department in Sri Lanka provides services to farmers in disease diagnosis and prevention, it still does not give farmers the tools to recognize and address a problem as soon as it arises.

Plant diseases are a significant problem in agriculture, the economics and growth of Sri Lanka are directly impacted by this agricultural problem. Plant diseases lower crop yields, raising the cost of living during a downturn in the country's economy. There are several variations of plant diseases and infections that unknowingly move throughout regions because of human activity and natural factors. Farmers can cure these diseases or remove the affected plant, depending on the type of plant disease, to stop it from spreading (Hosack & Miller, 2017). Human actions and natural causes both have the potential to cause plant disease. A new plant disease may spread and endanger the harvests without the farmers' knowledge thus, farmers should act more quickly and effectively to stop the spread of a disease by being able to immediately identify unknown plant diseases and the threat they pose to the farm.

Technology can safeguard the country's food supply and serve as a barrier to prevent infections from crossing international boundaries (World Food Programme, 2022). The Sri Lankan economy and its people can benefit from this technology by improving the yield's quality and quantity. Despite being a great and advantageous technological solution for Sri Lanka, this technology must be affordable for low-income farmers. Since many Sri Lankans own smartphones, this project intends to create a solution that can run on basic mobile devices that can diagnose diseases in time.

The agriculture sector in Sri Lanka is affected by a wide variety of plant diseases. Only a few important plant diseases affecting a narrow range of crops are the focus of this study (Cinnamon – Dept. Of Export Agriculture, n.d.). These include the following:

- **Tomato** (Tomato Bacterial spot, Tomato Early blight, Tomato Late blight, Tomato Leaf Mold & Tomato Septoria leaf spot)
- **Rice** (Bacterial leaf blight, Brown spot, Leaf smut, Hispa & LeafBlast)
- **Tea** (Red leaf spot, Algal leaf spot, Bird's eyespot, White spot & Anthracnose)
- **Coffee** (Rust & red spider mite)
- **Cinnamon** (Rough Bark & Stripe Canker)

These plant diseases have been selected based on their impact on the Sri Lankan agricultural industry.

2.3 Problem Definition

The process of identifying and separating a sick plant before the disease spreads and destroys an entire crop yield is crucial in agriculture. The time window when the disease affects a crop and it being spread to the next crop is very slim, thus requiring farmers to be well educated with every type of disease that could affect their crop is essential. It is not feasible for a person to be knowledgeable in every existing disease as well as identifying a specific disease as many diseases have the same symptom and treatments differ drastically. Involving a professional at a time when a crop is determined to be ill is time-consuming and time is of the essence. To overcome this problem efficiently so that the farmers themselves can identify and therefore act as fast as possible on the matter, this document proposes a solution that involves a mobile application that can identify the disease with accuracy that no human can achieve in the limited time window. The application self learns and corrects itself to be more accurate overtime and will get the results even faster. The application also includes a history of recent detections so that the farmer can take precautions to avoid a particular disease in retrospect.

2.4 Project Aim

The main goal of this project is to modernize Sri Lanka's agricultural sector by designing, developing, and evaluating a mobile application that can recognize and identify diseases which can impact different plants using a modern and future proofed technology that make identifying plant diseases easier, for low-income farmers who have little understanding of the diseases that can harm their crops.

The project aims to deliver a system that identifies plant diseases from input images using machine learning accurately and automatically. To achieve this, the best model and the best initial parameters will be chosen for the datasets and the trained data will be used against unseen data. The fundamental models will then be further improved by tuning hyperparameters to reach the required precision and accuracy. The outputs and the efficiency will be taken into consideration to choose the best version of the application that delivers the goal.

2.5 Project Scope

Make an application Identify the diseases from a photograph (Input). The user will then receive a brief explanation of the disease via the app. The app will provide a map with the locations of where the disease was previously discovered (locations learned from all the other user data). The purpose of this project is to create a mobile app that is more approachable and simpler for non-technical users to utilize. To diagnose the disorders, the program will employ image recognition. A Deep Neural Network can be used to create the image classifiers once the network has been trained on pictures of plants with diseases.

2.5.1 In-Scope Items

- Image classifier for Plant diseases detection using deep learning algorithm.
  - Source code
  - Trained model
  - Custom Data set for 5 crops
  - TensorFlow Lite
- App Designs
  - Low fidelity wireframes prototype
  - High fidelity prototype
- Flutter project developed for the Android Platform & Android App
  - Flutter Project/Source Code
  - Mapbox API integration
  - Firebase API integration
  - ML model integration
  - Follow the Material UI v3 guidelines to design and develop App UI
  - APK
- Integration of TensorFlow Lite to flutter project.
  - Use Image picker to select image from device or take an image from device.
  - Classify image and validate results.
- Backend Development & integration
  - Modelling relational data in NoSQL document database to easily store, sync, and query data for the flutter app.
  - Firebase Authentication (Continue with google Button on app): End-to-end identity solution to the app for easy user authentication and sign-in to the app in a secure manner.
  - Firebase Cloud Firestore: provide the app with a serverless apps solution that can scale globally, Sync data across devices, on or offline.
  - Firebase Cloud Storage to store user generated content.
- Documentation
  - Literature Review (LR)
  - Software requirements specification (SRS)
  - Software Architecture and Design / Tech stack
  - Test Report
  - User guide for app

2.5.2 Out-Of-Scope Items

- System will not support real time Image classifying and image recognition.
- Convolutional neural network algorithm used for deep learning image classification: system uses existing algorithm.
- Image classification is limited to plant and leaf recognition.
- Cloud based solutions to store User content.
- Multiple language support (Sinhala and Tamil).
- Usability testing will not be conducted with human test subjects.
- Dataset created from Secondary research.
- Custom backend with API, system will use existing technologies developed.
- Although app uses location the system is not location-based service (LBS)

2.6 Tools And Technical Requirements (Resource Requirement)

2.6.1 Software Requirements

- Operating System: Machine learning frequently uses one of the following operating systems: Windows, Linux, or MacOS.
- Python: Most machine learning libraries are developed in Python, a prominent programming language for this field.
- Python Libraries: NumPy, Pandas, Matplotlib, and Scikit-learn are a few well-known machine learning libraries for Python that offer a variety of tools for data manipulation, visualization, and model training.
- Deep learning libraries: Popular deep learning libraries that offer tools for creating, training, and analysing neural networks include TensorFlow, PyTorch, and Keras.
- Integrated Development Environment (IDE): For developing, testing, and debugging code, an IDE like Jupyter Notebook, Spyder, or PyCharm is helpful.
- Version Control System: Git is a widely used version management program that is helpful for monitoring code changes.
- Documentation: Microsoft Office/Google Docs to create reports and documentation with cloud sharing capabilities to work with the entire team.

2.6.1.1 Developer Environment

- Operating System (OS): Windows 10 or newer / macOS Big Sur or newer
- Android Studio (https://developer.android.com/studio)
- XCode 14 (https://developer.apple.com/xcode) | Optional
- Java 19 (https://www.oracle.com/java/technologies/javase/jdk19-archive-downloads.html)
- VS Code (https://code.visualstudio.com)
- Python 1.10 (https://www.python.org)
- NodeJS (https://nodejs.org/en)
- Flutter 3.3 (https://flutter.dev)
- gradle-7.6 (https://docs.gradle.org/7.6/release-notes.html)
- CUDA toolkit (https://developer.nvidia.com/cuda-toolkit)
- Anaconda Environment (https://www.anaconda.com/products/distribution)
- Jupyter Notebook (https://jupyter.org)
- Git (https://git-scm.com)

2.6.1.2 User Environment

- Operating System (OS): Android 8 or newer (https://developer.android.com/about/versions)

2.6.2 Hardware Requirements

The resources that are expected to complete the project are identified based on the objectives, anticipated solution, and deliverables.

- Central Processing Unit (CPU): Running machine learning algorithms requires a modern multicore CPU. AMD Ryzen or Intel i5 or i7 processors. Minimum Recommended CPU Intel 5th gen or newer or Ryzen, with multicore and multithreading support.
- Graphical Processing Unit (GPU): Deep learning methods require a graphics processing unit (GPU) to run because they are computationally demanding. Minimum GPU Requirements NVIDIA GeForce 10 Series Graphics Cards or higher to utilise CUDA framework for GPU accelerated parallel processing for ML model training.
- Random Access Memory (RAM): For most machine learning workloads, 8GB of RAM or more is advised. However, 16GB or more might be needed for larger datasets or more complicated models.
- Storage: For the dataset, mobile platform emulators, and any trained models, a machine learning system will need a sizable quantity of storage. It is advised to have at least 256GB of storage, but larger datasets or more complicated models can call for 1TB or more.
- Cooling: Due to the computationally demanding nature of machine learning algorithms, sufficient cooling is necessary to prevent overheating.
- Power supply: A dependable power source is necessary for a machine learning system to keep it steady throughout training.

2.6.2.1 Recommended Hardware Requirements

2.6.2.1.1 Developer environment

- CPU: Intel 12th gen i7 or i9 processors or newer /AMD Ryzen 9 5000 series or newer
- GPU: Nvidia GeForce RTX 30 series or newer
- Memory: 16GB or 32GB (DDR4 or DDR5)
- Storage: 512GB

2.6.2.1.2 User environment

- Android Device: Google Pixel 6 pro or newer / Samsung Galaxy S20 or newer
- Display size: 6.7 inch.
- Memory: 128GB (8GB RAM)
- Main camera: 12MP or higher

2.6.2.2 Minimum Hardware Requirements

2.6.2.2.1 Developer environment

- CPU: Intel 5th gen i5 or i7 processors or newer /AMD Ryzen 3 5000 series or newer
- GPU: Nvidia GeForce GTX 10 series or newer
- Memory: 8GB or higher
- Storage: 256GB

2.6.2.2.2 User environment

- Android device: Google Pixel 3 or newer / Samsung Galaxy S10 or newer
- Display size: 5.5 inch or larger.
- Memory: 32GB (4GB RAM)
- Main camera: 12MP or higher

2.6.3 Data Requirements

Name	Dataset
Rice	Leaf Rice Disease | Kaggle Rice Leaf Diseases Dataset | Kaggle Rice Diseases Image Dataset | Kaggle Dhan-Shomadhan: A Dataset of Rice Leaf Disease Classification for Bangladeshi Local Rice - Mendeley Data Rice Leaf Disease Image Samples - Mendeley Data New Bangladeshi Crop Disease | Kaggle
Tomato	Plant Disease Detection using Keras | Kaggle
Tea	Identifying Disease in Tea leaves | Kaggle
Coffee	Coffee plant disease | Kaggle https://data.mendeley.com/datasets/yy2k5y8mxg JMuBEN - Mendeley Data
Cinnamon	Cinnamon Plant Stem and Branch Disease Dataset | Kaggle

Notes:

- **Quality**: The information must be accurate, of a high standard, and relevant to the work at hand. An incomplete, inconsistent, or biased set of data might have a detrimental effect on the model's performance.
- **Quantity**: The complexity of the task and the model being used will determine how much data is needed for machine learning. In general, performance improves with increased data.
- **Labelled Data**: We need labelled data for our machine learning tasks, which means that the data is accompanied by details about the intended classification or result.
- **Data Format**: The format of the data must be appropriate for the machine learning algorithm or framework being utilized. The most popular formats are CSV, TSV, and JSON.
- **Pre-processing**: Before the data can be utilized to train a model, it may need to be cleaned, transformed, or normalized.
- **Data Splitting**: Datasets for training, validating, and testing must be separated. This is crucial to assess the model's performance and determine how well it generalizes to fresh, untested data.
- **Data augmentation**: The amount of data that is sometimes provided might not be enough. In these situations, training data can be artificially increased by using data augmentation techniques.

2.7 Document Structure

The documentation begins by providing background on the problem at hand, explaining why it is important, outlining the aim of the project, and stating the project scope, as well as the tools and technical requirements needed to replicate the research.

Chapter two consists of a cohesive literature review by exploring the existing machine learning modules and solutions for the problem at hand. A background on machine learning is provided, its capabilities in the past, present, and future, along with the process of choosing the correct machine learning model for a particular problem and the key aspects of the machine learning model used in the project is discussed. Additionally, the technologies used in the project is reviewed and justified for their selection.

Chapter three consists details of the methodologies employed in this project. It discusses the specific software development methods chosen, the project deliverables, any deviations from the plan, and the steps taken to mitigate potential risks.

Chapter four delves into the process of gathering system requirements by starting with stakeholder analysis. then the methodologies used for requirement elicitation, conducted SWOT analysis, and covers both functional and non-functional requirements. Additionally, it provides a context diagram, use case diagram, use case elaborations, design class diagram, and sequence diagram to give a comprehensive understanding of the system requirements gathering process.

Chapter five covers the design specification, outlining the design principles followed. High-level architecture diagram, data flow diagram, tiered architecture diagram, data flow diagrams are provided, along with UI wireframes to give a comprehensive understanding of the system's design.

Chapter six, consists detail of the project implementation, including the tools and languages used as well as the technology along with the reasoning behind our development decisions, implementation of core functionalities, the challenges encountered.

Chapter seven delves into the testing of the system, outlining the goals and objectives of the testing, including various functional, non-functional, integration and model tests performed. Additionally, competitive, functional, and internal benchmarking are provided to ensure that the system is functioning as intended.

The document provides a cohesive conclusion for the project by reflecting on the project aims and progress, discussing the existing skills of the authors, identifying any deviations from the plan, and providing justifications for them, and states future enhancements for the project. Additionally, the Appendix includes references, meeting minutes documentation, and a user guide for the application.

2.8 Chapter Summary

This chapter contains the high-level overview of the project for the machine learning based plant disease detection mobile app for Sri Lanka. The reason for this project to exist and the problem it aims to solve. The scope of the project was outlined and documented the necessary resources, required for the project.

CHAPTER 2: LITERATURE REVIEW

3.1 Overview Of Chapter

The next step in computer evolution is the automation aspect which automates tasks on its own with little to no human input or hardcoded programming modules, Machine learning (ML) is a branch of artificial Intelligence (AI) which makes applications more accurate and thus leaving less room for error or false predictions on its own. An algorithm is any step or technique used to perform a certain calculation; ML is performed by using past available data as input to a machine learning algorithm which intern will predict an output that is desired (Burns, 2021). As computers evolve into automation, ML has become a topic that will continue to arise hence this literature review will focus on the current state of machine learning.

3.2 The Literature on Machine Learning

3.2.1 What Is Machine Learning?

Humans learn and evolve by past actions and experiences, but machines have always followed instructions constructed by humans, even though this type of computation has revolutionized the entire world, machines have become to a point where it is also capable of learning from past data to improve the present and predict the future which is known as machine learning (NexusAdmistraIntegra, 2020). To understand ML, one must think of the way the human brain categorizes information which is subject to change based on future and past experiences thus is not a fixed state information, Machine learning follows a similar pattern, it learns and categorizes data in a way it is understandable and is flexible to change and improved upon. ML consists of many algorithms to predict and understand data, these algorithms can self-modify and improve over time, prediction becomes more accurate with more data since it learns from the data to build the prediction model and finetunes it so that when the time comes for prediction, it can predict accurately (NexusAdmistraIntegra, 2020). This prediction ability of ML can be used in any field in the real world, for instance writing an algorithm that can do more than one type of task is time consuming and not practical but using ML one can let the machine to learn and improve its algorithm in a way it is usable and adaptable to any environment. The best part of ML is its ability to learn from its mistakes and correct itself faster than any human.

3.2.2 The Importance of Machine Learning

Due to modern technological advancements and the continuous evolution of computation comes a demand and importance for data analysis. With the enormous variety and access for data, a high value prediction technology is much needed to make better decisions and better way of handling certain tasks in real time faster than any

human or human programmed application can handle automatically. ML has changed the way data is handled and interpreted and has replaced traditional techniques (Tavasoli, 2016). Machine learning allows computers to improve their performance on a specific task without being explicitly programmed. This is particularly useful in fields such as image and speech recognition, natural language processing, and decision making, where traditional programming methods have limitations. Additionally, it can be used to analyse and make predictions from large datasets, which can lead to new insights and improved decision making in a wide range of industries (Burns, 2021). Machine learning can be used in a plant disease detection app to analyse images of plants and identify symptoms of various diseases. The app can use a trained machine learning model to recognize patterns in the images that indicate the presence of a specific disease. This can help farmers and gardeners quickly diagnose and treat plant diseases, improving crop yields and reducing the spread of disease.

3.2.3 What Is Machine Learning Used For?

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns, and make decisions with minimal human intervention according to Sarker (2021), thus, machine learning is used in a variety of applications, such as:

- Image and speech recognition
- Natural language processing
- Anomaly detection
- Predictive modelling
- Self-driving cars
- Medical diagnosis
- Financial analysis
- Fraud detection
- Email filtering
- Recommendation systems

ML is used in almost all industry that involves a lot of data, such as ceramic industry which uses ML to predict the behaviour of materials under different environment, automotive industry which uses ML to predict component durability and identifying anomalies earlier, energy management industry which uses ML to analyse electricity consumption demands and supply power accordingly, food industry which uses ML to understand consumer trends and needs that are subject to change overtime (NexusAdmistraIntegra, 2020). ML can also be used to identify a potential problem before it occurs which can be applied to any sector that will benefit from regular maintenance.

3.2.4 Types Of Machine Learning

There are mainly three popular types of ML systems namely: Supervised, Unsupervised, Semi- and Reinforcement machine learning (What Is Machine Learning?, 2022). To understand these types, one must first get a clear understanding of data which is what all ML algorithms are dependent upon. The most common form of data includes words and numbers which are stored in a particular order of tables or data can include values of pixels, waveforms of images and audio, all of which is stored in relation and categorized in datasets. Such as a plant disease dataset that includes images of plants with diseases and the information of said disease. All datasets contain labelled examples and features. An example is equal to a row in a table, whereas features are the values stored in it which the training model uses to predict the label. A label is thus the output of the prediction model, for instance a weather model that predicts the amount of rain fall for a particular day has features such as temperature, humidity, wind direction and so on, in this model the label would be the rainfall amount that is predicted based on the data gathered and analysed to predict (Supervised Learning | Machine Learning, 2022). Dataset can be labelled, or unlabelled, labelled data are examples that have both the features and the respective label whereas unlabelled examples does not contain a label, the model predicts the label. Many examples of a particular subject are considered as a dataset of that subject, dataset is the amount of diversity or range of data coverage and size of the data. Datasets come in different diversity and size, thus depending on just the size or the diversity will not make the dataset suitable, a dataset must be both highly diverse with sufficient size of data to predict accurate predictions (Supervised Learning | Machine Learning, 2022). Additionally, datasets that include many features will help the model to predict better than a dataset with less features, but it is not always guaranteed as it is highly dependent on the relationship to the label to be predicted.

3.2.5 Machine Learning Model

ML model is any file that has been trained by an algorithm to recognize a certain pattern with the associated dataset where it uses the data to understand, learn and predict a label (QuinnRadich, 2021). A trained model can then be used to predict and analyse new data which the model will learn from to be more accurate.

3.2.6 Training A Model.

Machine learning model accuracy is highly based on its training, training a model consists of using datasets with labelled examples that way the model then works on predicting the labels from the available features and then compares the output with the labels actual value and the predicted value. The model self learns from its miscalculation based on the difference between predicted and actual values and updates the prediction model to be more accurate (Supervised Learning | Machine Learning, 2022). Furthermore, to make the model more

accurate, ML practitioners can adjust the configurations and features that the model uses to perform predictions. Once the training is complete an evaluation takes place to determine the model's learning, which is by using a label dataset with just the features from which the model is to predict the labels which are then crosschecked with the actual labels of the dataset.

3.2.7 Supervised Learning

Uses labelled data to train the model, the machine knows the features of an object or dataset and the labels associated. We gather the data which has been labelled with the features associated with it so that the machine understands the patterns and identifies each label accurately (Supervised Learning | Machine Learning, 2022). This is equal to a student learning from past examinations with questions and answers thus they are well prepared to take a new examination (What Is Machine Learning?, 2022). Supervised comes into play as it involves a human supervises and trains the model with the correct results.

3.2.8 Unsupervised Learning

Uses unlabelled data or incorrect answers, its goal is to identify and understand patterns among the data hence the model has no way to understand how to categorize the data, so it must draw conclusions on its own rules (What Is Machine Learning?, 2022). It employs a technique called clustering which is grouping related examples of data which a human can then supply meaning to if need be. For instance, this may include a cluster based on temperature in a weather dataset which reveals a segment that showcases different seasons based on temperature, which one can then name accordingly (What Is Machine Learning?, 2022).

3.2.9 Reinforcement Learning

Uses a reward system to learn based on the principal of feedback, for instance if one provides a picture to the machine of a plant with disease type A and ask it to predict the type of disease, the machine learns based on the output and the confirmation or correction given by the user, in this case if the machine predicts the disease is type B, the user will then tell its incorrect and the machine will learn from the feedback to classify the type of disease accurately next time (What Is Machine Learning?, 2022).

3.2.10 Simi-Supervised Learning

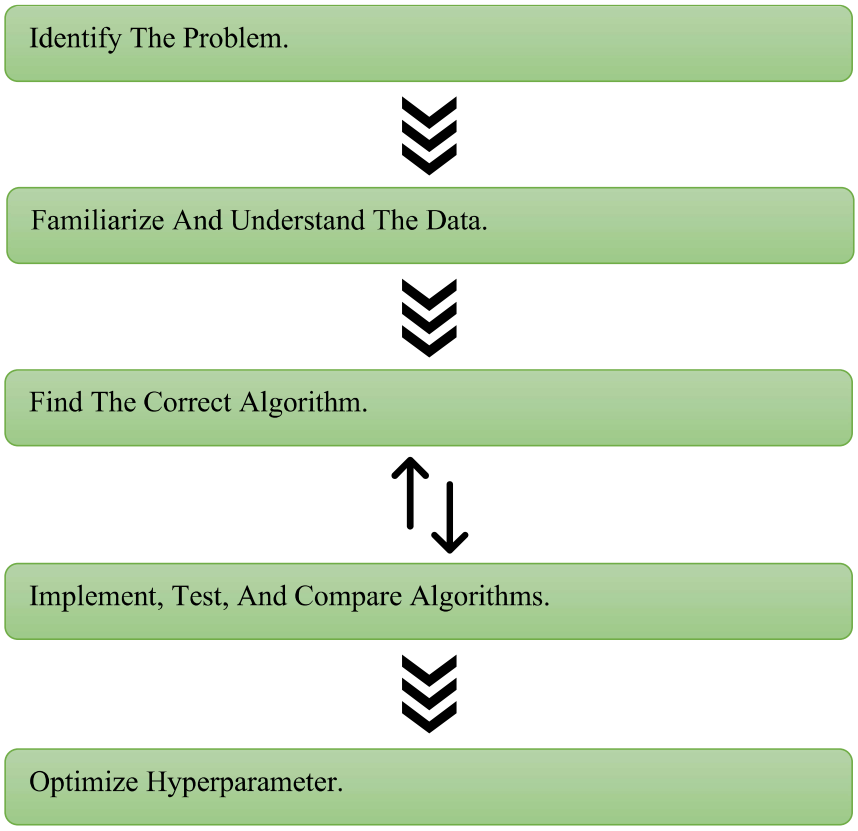
Another type of machine learning involves combining both supervised and unsupervised learning techniques together called semi-supervised learning, which uses a combination of both supervised and unsupervised

learning to overcome the disadvantages of both learning modules. It works by training a module with both labelled and unlabelled dataset, it mainly involves few labelled examples and many unlabelled examples. Some applications that use semi-supervised learning include speech analysis, web content classification and protein sequence classification (Importance of Machine Learning - Javatpoint, 2011). Thus, semi – supervised learning has the best of both supervised and semi supervised advantages.

3.2.11 Deep Learning

This type of machine learning involves training artificial neural networks on a large dataset. Deep learning algorithms can learn and make intelligent decisions on their own, without human intervention. It is designed to analyse data with logical structure similar to a human brain leading to a conclusion. Artificial neural networks, which are layered structures of algorithms used in analysis and are modelled after the biological network structure of the human brain, are much more effective at learning than traditional machine learning models (Grieve, 2020). It is intended to examine data using a structural model that resembles that of the human brain and leads to a conclusion.

3.3 Choosing The Correct Machine Learning Model.



Choosing the right type of ML model depends on various factors and it can be overwhelming with the abundance of choices available. Finding a solution to a problem includes several steps and each must be followed sequentially as shown.

Identify the problem – This is a crucial part in choosing the right ML learning model, if the data or target values are labelled then it comes under supervise learning, if its unlabelled and the sole purpose is to find patterns in the data then it belongs in the unsupervised category. Additionally, the type of output is more important. Regression is the problem if the output is going to be any kind of number-based output. In contrast, a classification problem is when the output is a class, and a clustering problem is when the output is a set of input groups (Zaid Alissa Almaliki, 2019). If one thinks the model will benefit from labelled and unlabelled data with pattern recognition, then Semi-supervised learning model will be a suitable path. For the type of problem that will benefit from human or environmental feedback to be more accurate or learn overtime, comes under reinforcement category (breton, 2021). Hence, identifying the problem is the first and the most crucial stage of choosing the correct machine learning model.

Familiarize and understand the data – Understanding the type and size of data is crucial for choosing a model as it decides the successful rate of the module. Data is referred to as the raw material in the analysis process and these raw materials give us insights that lead to a better decision and an overall better ML model. Both categorical and numerical algorithms work with different types and different amounts of data thus to make it more efficient, one must decide the requirements based on whether it will need a small or a large dataset. Then it's time to analyse the data which includes tasks that understand descriptive statistical data plus data that contains visualization and plots. Once the analysis is complete it is necessary to process the data that contains events such as data pre-processing, profiling, and cleansing from different sources (Zaid Alissa Almaliki, 2019). Finally, the data is to be transformed to a state that is usable and beneficial for the model, it involves a term that is known as feature engineering. According to Jason Brownlee, feature engineering is a way to make the predictive model more accurate of unseen data by transforming raw data in a way it contains features that suit the model being developed and the problem to be solved (Zaid Alissa Almaliki, 2019). Hence, familiarizing and understanding the data is the second step in most crucial stage of choosing the correct machine learning model after identifying the problem.

Find the correct algorithm – Once the problem and the data are understood and categorized according to the needs of the model, the next step is to identify the algorithms that are suitable to be implemented, to complete this stage certain things need to be considered such as, the accuracy, interpretability, complexity, and scalability of the model (Zaid Alissa Almaliki, 2019). The time it takes to build, train, and test the model. The performance of the model once it is implemented and finally, the model needs to meet the goal of the of the project. Finding the correct algorithm can make or break the entire project as the entire project is dependent on it.

Implement, test, and compare algorithms – The performance of each algorithm can be compared by setting up a ML pipeline or the same algorithm can be used in different groups of datasets and can be run whenever unseen data is added. According to Zaid Alissa Almaliki (2019) the best practice is to use a baseline algorithm model and build on top of it, rather than jumping into a complicated algorithm right away (Zaid Alissa Almaliki, 2019).

Optimize hyperparameter – Hyperparameter is a value that determines and controls the learning process of a machine learning model, one sets the hyperparameter values that the algorithm will use to learn before the training, since the model cannot change the values during the training process. Parameters reside inside the model, they are learned from the data during the learning process, where the algorithm maps the input features and the labels. Hyperparameters are not part of the model, once the learning is done, the parameters will be trained which intern is what we refer to as the model and the parameters that were learned can be known whereas the hyperparameter values which were used to train it will not be known by looking at the model (Nyuytiymbiy, 2022). There are ways to optimize hyperparameters such as grid search and random search optimization. Grid

search divides the hyperparameter domain into discrete grid then every combination of values is tried to calculate the performance using cross validation, which is a time-consuming process but will find the best point in the domain. Whereas random search, even though is similar to grid search, it tests randomly selected subset points, which means the smaller the points the faster the optimization but less accuracy and vice versa (Malato, 2021). It's also important to note that choosing the correct model is an iterative process and it may require one to try multiple models and evaluate their performance. It is good to use domain knowledge to guide the selection process and to make sure that the chosen model is practical and can be deployed in the real world.

3.3.1 ML Algorithms & Problem Types.

Table 1A table that consists of ML algorithms based on its problem type.

Unsupervised	Clustering	K-means
	Dimension Reduction	PCA
		SVD
Supervised	Classification	Logistic Regression
		LDA
		Kernel Support Vector Machines
		Random Forest
		Neural networks (CNN)
	Regression	Neural networks
		Linear Regression

3.3.2 Linear Regression

When it is necessary to predict a future value of a process that is currently running, a linear regression algorithm can be used. It is a supervised statistical method which uses a linear equation to find the best fit linear line in the dependent and independent variable. One variable is regarded as independent while the other is dependent, it uses the independent variable to predict the outcome of the dependent variable (Zaid Alissa Almaliki, 2019). For instance, it can predict obesity in relation to work life balance linearly (Spiceworks, 2022). In other words, it summarizes relationships between two continuous variables.

3.3.3 Logistic Regression

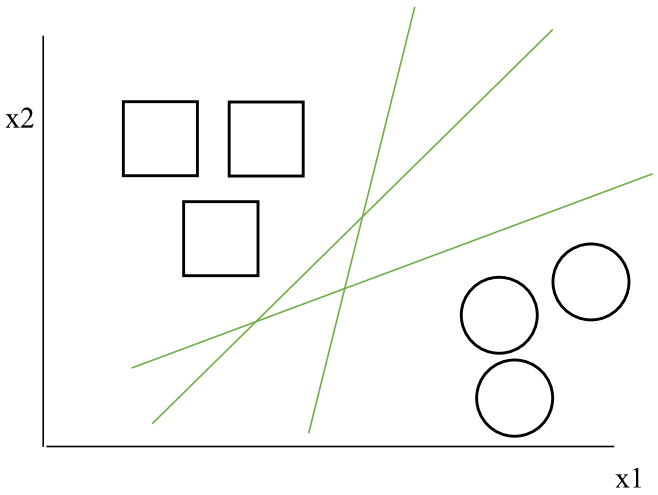
Despite its name, it is a classification algorithm which performs binary classification thus the label or output is in binary between 0 and 1, which models the probability of the event taking place using log odds or previous observations of a dataset. For instance, logistic regression can predict the stage of a cancer patient (Spiceworks, 2022). When the output is categorical it can be taken as a special case of linear regression where a log of odds is used as the dependent variable (Zaid Alissa Almaliki, 2019).

3.3.4 K-Means

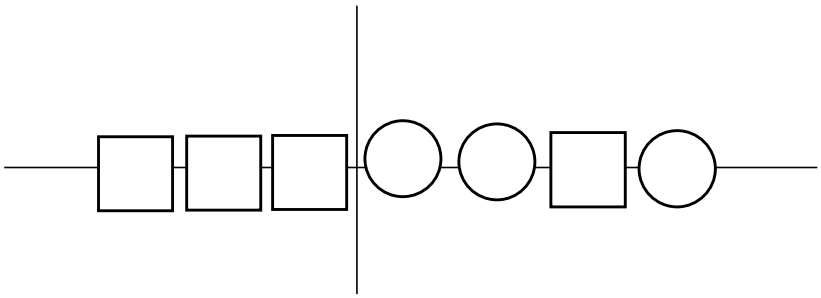
An unsupervised clustering algorithm used to separate data into groups in other words it is used to classify unlabelled data into a predetermined number of cluster/groups based on similarities, which is k. K-means clustering works by providing it the number of clusters (k) that needs to be generated. Then k data points are chosen at random, and each point is assigned to a cluster. After which the cluster centroids will be computed till the ideal assigning of data points to clusters that do not vary known as centroid is found (Sharma, 2021). For instance, if there are 10 data points which represent 5 grapefruits and 5 broccolis and one want to categorize them into two groups. According to Zaid Alissa Almaliki (2019), one of the ways to differentiate would be with the size or shape differences, thus, to categorize this data, one selects two random data points as the starting position then compares it with every other point to find the closest starting position. Since the data points were chosen randomly the first fluster will be inaccurate. Both grapefruit and broccoli will be in each cluster in different amounts, the next stage is to take the average of all points in one group and use that as the starting point for that group and follow the same technique for the other group. On the second round the grapefruits and broccolis will be in their own separate groups, since the average is a lot closer to the majority of each cluster and to confirm the success rate it is advised to calculate the average and run the grouping again to see if any points have changed the third time, if no points have changed then the grouping is done, if not the grouping process is run till there are no changes

3.3.5 Kernel Support Vector Machines (SVM)

Is a supervised algorithm best suited for classification, the main objective is to a hyperplane in an N-dimensional space to classify the datapoints. The dimensions depend on the number of features in the dataset. If there are only two features then only one hyperplane is used to separate it into two, whereas if the input features are more than 2 then it becomes a 2-D plane (Sasidharan, 2021). For instance, if there are two independent variables x1 and x2 that includes one dependent variable which is in this case is a square or a circle as follows.

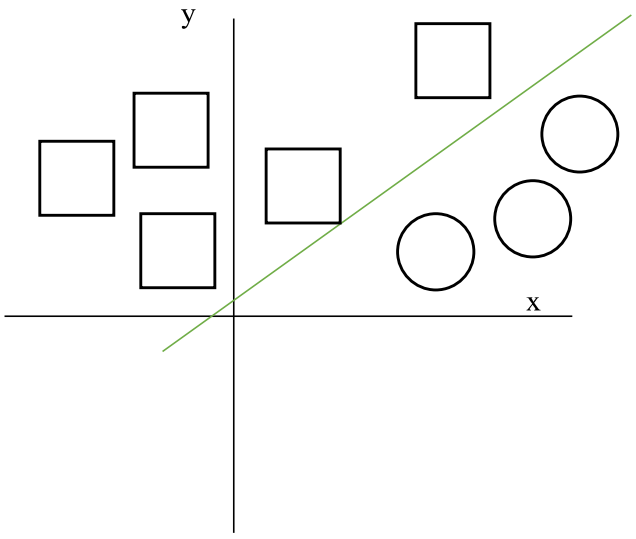


There can be multiple hyperplane (line) in t), situation with two input features which separates the data between squares and circles (linearly separable data), but since there can be multiple hyperplanes in multiple positions, one way to select the best hyperplane is the plane that denotes the largest separation between two classes, in other words the hyperplane with the maximized distance to the nearest data point on each side known as the maximum-margin hyperplane (Sasidharan, 2021). In case there's a circle close to a square and thus cannot be separated as all circles and all squares the SVM classifies the circle close to the square as an outlier of circles and ignores the outlier to find the best hyperplane. If the data is not linearly separable, for instance if the data is as follows, then the SVM solves this by creating new variables using a kernel.



Origin (o)

In this case it calls a point x_i on the line and creates a new variable y_i as a function of distance from the origin point o and get the following.



According to Sasidharan (2021), a new variable y is created as a function of distance from the origin and since it is a nonlinear function that creates a new variable it is referred to as a kernel and it converts non-separable into sparable based on labels.

3.3.6 Random Forest

Is a supervised algorithm that grows and combines different decision trees to create a forest of decisions that can be used for both classification problems. A decision tree is a type of algorithm that is used to classify data like a flowchart with different pathways to an outcome which starts at a single point and branches off into multiple directions with different possible outcomes (Meltzer, 2021). Random forest combines multiple decision trees to predict the class of a dataset and some trees may predict correct outputs and some do not. For a better forest classifier there should be actual values in the feature variables of the dataset to increase accuracy and the predictions must have low mutual relationships. The advantage of random forest is that it takes less training time, predicts outputs with high accuracy and most importantly it can maintain accuracy when a large portion of the data is unavailable. It works in two phases, phase one is to create the random forest with a combination of N decision tree and phase two is when the predictions for each created tree occur. The working process includes selecting random K data points from training set, then build the decision trees with the selected data points, finalize the number N for decision trees that needs to be built, repeat selection of random K points and building trees from the selected data (Machine Learning Random Forest Algorithm - Javatpoint, 2021). With the predictions of each decision tree, new data points are added to a category which has most of the votes.

3.3.7 Neural Networks

Inspired by the human brain, neural networks are a form of artificial intelligence. It simulates lots of densely interconnected brain cells inside a computer system to learn, recognize patterns and make decisions in a human-like manner, it learns by itself. According to Zaid Alissa Almaliki (2019), on one side of the neural network there are inputs which could be a picture of a plant or a type of data from a drone. On the other side there are outputs of what the network wants to do. And in between are nodes with connections determining what output makes sense based on the input.

3.3.8 Principal Component Analysis (PCA)

Machine learning uses the Principal Component Analysis (PCA) method to reduce dimensionality. It is mostly used for exploratory data analysis and data visualization because it is an unsupervised technique that disregards the labels of the data. Finding the key factors (principal components) responsible for the bulk of the data's variance is the aim of PCA. PCA can decrease the dimensionality of the data while keeping the most crucial information by finding these principal components. The initial step in PCA is to identify the eigenvectors (set of vectors associated with a linear system of equations) of the data's covariance matrix, or the directions in which the data has the maximum variance. The data is then projected onto a new, lower-dimensional space using these eigenvectors. The primary components, which are the new axes in this space, are ranked according

to how much variance they account for. It may also be used to plot high-dimensional data in 2D or 3D, which is helpful for examining and comprehending the data's structure (Biswal, 2023). By eliminating noise and redundant data from the data, PCA helps prevent overfitting in models, which is one of its key advantages.

3.3.9 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a supervised method for dimensionality reduction and feature extraction in classification tasks. It increases the distance between various classes in the condensed feature space. LDA identifies the linear combinations of the original features that best separate the various classes by assuming that the data within each class is normally distributed with equal covariance matrices thus, LDA is used for classification problems and focuses on maximizing the separation between classes (Meigarom Lopes, 2017).

3.3.10 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is an unsupervised method for dimensionality reduction, matrix factorization, and feature extraction. It breaks down a matrix into its singular values and singular vectors, which are its component pieces. Finding the low-rank approximation of a matrix or lowering the dimensionality of the data while keeping the most crucial details, is the aim of the SVD algorithm (Afham, 2020). Thus, is used for unsupervised learning and focuses on reducing dimensionality while retaining the most important information.

3.4 Evolution Of Machine Learning

The evolution of machine learning can be traced back to the 1940s and 1950s, with the development of early computer systems and the concept of artificial intelligence (Anyoha, 2017). However, the field of machine learning as we know it today began to take shape in the 1980s and 1990s. Some key milestones in the evolution of machine learning include:

- The development of decision trees in the 1960s and 1970s laid the foundation for many modern machine learning algorithms (Selman, 2022).
- The introduction of the backpropagation algorithm in the 1980s, which made it possible to train large neural networks more efficiently (Raitoharju, 2022).
- The emergence of support vector machines (SVMs) in the 1990s, which became a popular technique for classification and regression problems (Pisner & Schnyer, 2020).
- The advent of kernel methods and kernel tricks in the late 1990s, which allowed SVMs to be applied to non-linearly separable data (Wilimitis, 2019).
- The breakthroughs in deep learning in the 2010s, which led to significant improvements in image and speech recognition, natural language processing, and other applications of machine learning (Chai et al., 2021).

- The emergence of reinforcement learning in the 2010s, which has led to significant advances in control and decision-making in various fields such as robotics, gaming, and self-driving cars (Zhang & Mo, 2021).
- The development of Generative models, such as GANs, VAEs, etc. which are being used in various fields such as computer vision, audio processing, etc (Cong et al., 2022).
- The recent advancements in AutoML, which enables the automation of the machine learning pipeline, making it more accessible for non-experts (anton, 2022).

Overall, the evolution of machine learning has been marked by a steady increase in the complexity and capabilities of models, as well as a growing emphasis on big data and distributed computing.

3.5 Future Of Machine Learning.

Several well-established businesses are being completely transformed by machine learning thanks to its tremendous development potential. It consistently provides insights at speeds and precision no human has ever achieved, supports developers with creative solutions to issues, and aids businesses in making critical decisions thus, it is humanity's best creation yet. In the modern world the term big data alludes to the availability of information that is produced and accessed by every company. Thus, we cannot manage most data streams all around us, it needs a more sophisticated approach, one that can be done by superhuman speeds. To intercept data and analyse it, Machine learning is the best solution since it can automate generation, storage, retrieval, and the analysis of data to quickly derive significant insights from the dataset. Machine learning is the reason behind technological achievements such as robots that are constantly learning and improving at a continuous state as long as they are powered. Automation technologies such as MLOps and AutoML make machine learning automation possible which frees up every organization to focus on removing repetitive tasks such as maintenance and data integration. Data determines the ability to develop and learn aspects of machine learning and the more data the more accurate the performance of the algorithm, which is already evident in recommendation engines such as Netflix, YouTube, Amazon and so forth that uses machine learning algorithms to suggest based on user likes and dislikes history data to constantly learn and understand the user (Singh, 2022).

The future of machine learning however is largely anticipated to continue to advance in Deep learning, which has already significantly improved picture and speech recognition. The focus may also shift to generative models and reinforcement learning. Additionally, there might be a stronger focus on creating verifiable AI and applying machine learning into a wider variety of applications and sectors, it is also anticipated that edge devices and IoT would apply machine learning more frequently (Janiesch et al., 2021). We might also witness advancements in areas of AI including decision-making, autonomous systems, and natural language understanding because of the growing amounts of data and computer capacity.

Machine learning can be used to help a variety of industries, including:

- **Healthcare**: To improve disease detection and treatment, machine learning can be utilized for tasks including speech and picture identification, natural language processing, and predictive analytics (Huang et al., 2022).
- **Finance**: Fraud detection, risk management, and portfolio optimization can all be done using machine learning (Altexsoft, 2019).
- **Retail**: Demand forecasting, price optimization, and personalized recommendations are all possible with machine learning (Couto et al., 2020).
- **Manufacturing**: For quality assurance and predictive maintenance, machine learning can be employed (Alexandre Gonfalonieri, 2019).
- **Transportation**: Self-driving cars, traffic forecasting, and logistics efficiency can all benefit from machine learning (Media, 2018).
- **Energy**: In power plants and wind turbines, machine learning can be utilized for proactive maintenance and defect finding (Ng & Lim, 2022).
- **Agriculture**: Disease diagnosis (Mohanty et al., 2016), precision farming, soil analysis, and agricultural production prediction are all possible uses for machine learning (van Klompenburg et al., 2020).

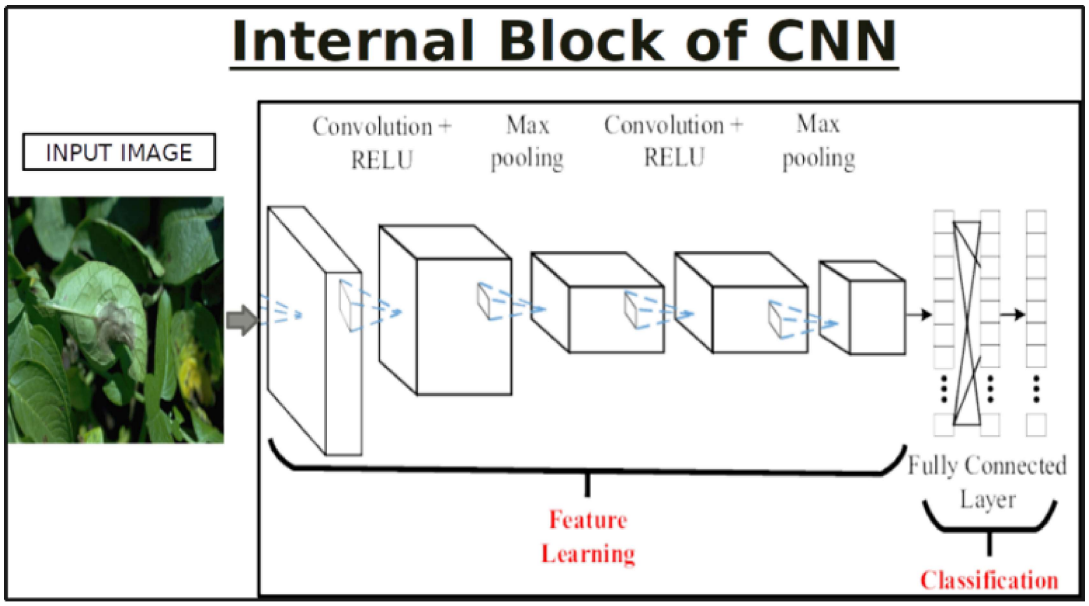
Thus, machine learning can be applied to many different industries to automate and enhance decision-making, streamline operations, and increase productivity, the future of machine learning has unlimited potential.

3.6 Key Aspects of Machine Learning Used in Our Application.

Convolutional Neural Network (CNN) is a supervised type of deep learning neural network used mostly in image recognition as it is capable of processing data in a structured array. It is used to classify images and to predict patterns among them with many features, namely a simple structure, less training parameters and the ability to adapt (Liu et al., n.d.). Reason for the selection of CNN learning model includes (Understanding Feed Forward Neural Networks in Deep Learning, 2022)

- Its ability to pick up the input image characteristics such as lines, circles, gradients, or any anomaly.
- The state of the image is of little importance as it can even process an imperfect image depending on its characteristics.
- CNN is a multilayer / feed forward neural network, which is an AI network where the nodes do not loop, and the data flow makes sure the input nodes receive the data first which then travels through hidden layers and finally to the output nodes.
- Since it is a convolutional layer which has many layers assembled on top of another where each layer can recognize more characteristics in the image as complex as handwriting or human face, the more layers the more sophisticated it becomes.

CNN Architecture diagram (Munnangi, 2019).



3.6.1 Steps To Build A CNN Learning Architecture.

- Convolution
- ReUL Layer
- Pooling Layer
- Flattening
- Fully connected layer.

3.6.2 Convolution

The very first layer that extracts the characteristic of the input image in our case the plant picture to learn the relationships of features, it does this process with kernel and filters.

3.6.3 ReUL Layer

Rectified Linear Unit is a common deep learning activation function it returns 0 if the input is negative whereas the positive input returns the value of the input (Chen, 2021), it essentially filters the images that contain a negative value with 0 when the node input is above a certain threshold (Challa, 2022). The output is $f(x) = \max(0,x)$ it is used to introduce non linearity to CNN model (Munnangi, 2019).

3.6.4 Pooling Layer

Used to only keep the valuable data which is to be further processed by reducing the parameters using down sampling. The type of pooling we choose is Max Pooling it reduces the dimensionality of images by carefully reducing the number of pixels in the output which it calculated from the previous convolutional layer (Max Pooling in Convolutional Neural Networks Explained, 2018).

3.6.5 Flattening

Used to convert 2 Dimensional arrays from pooled maps to a single continuous vector where it becomes a flattered matrix which is then inserted as input to the connected layers to classify the image (Biswal, 2022).

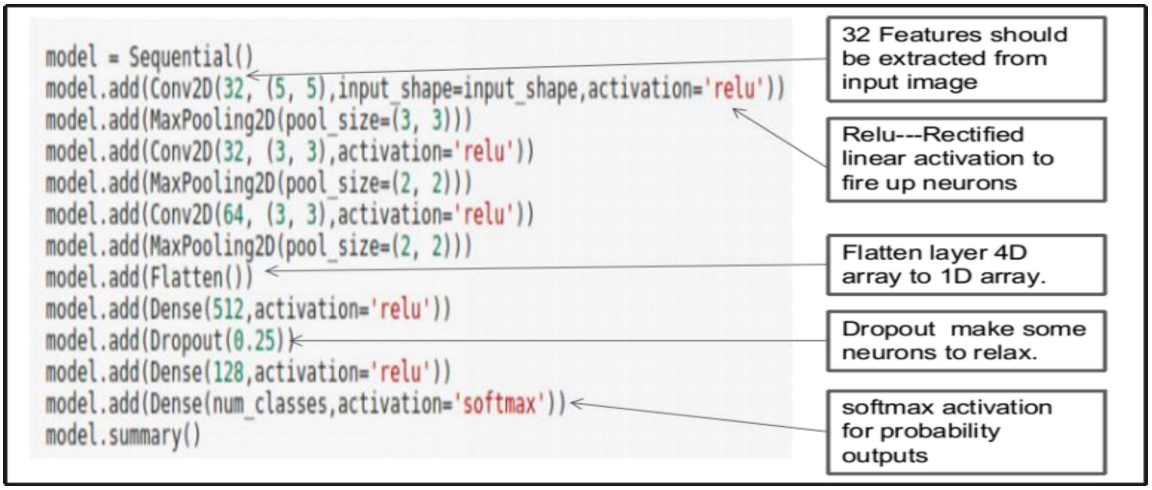
3.6.6 Fully Connected Layer.

The layer multiplies the input with a weight matrix then adds a bias vector, both convolutional and down sampling layers are followed with many connected layers (Fully Connected Layer - MATLAB, 2022). The flatten vector is passed into the input layer and is combined with features to create the model which then is activated via an activation function to classify the outputs using softmax or sigmoid.

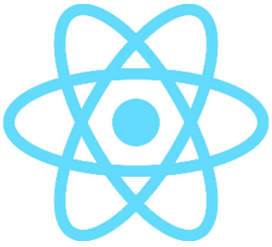
3.6.7 Computing Platform

Jupyter notebook has become is an iterative web-based platform that supports multiple programming languages, has markdown cells, easy formatting, and more detailed write ups (Goled, 2022). It allows users to create as well as share documents with live code, visualizations, and equations. It is based on numerical simulation, statistical modelling, data visualization, data cleaning and transformation as well as machine learning (Klingler, 2021).

Summary of layers diagram used to build CNN model (Munnangi, 2019).



3.7 Literature Review of The Technologies

App Development	
Native platform	Building an app exclusively for a given platform, such as iOS or Android, utilizing the platform's native programming language and development tools, is known as native app development. This strategy enables complete access to the features and capabilities of the device and can lead to a more refined and polished user experience. The software will need to be created and maintained independently for each platform (Gillis, 2022).
Cross platform	Cross-platform app development entails creating an app that can run on numerous platforms while utilizing a single codebase. By eliminating the need to create and maintain different codebases for each platform, this method can save time and money. Cross-platform development can be accomplished in a variety of methods by employing web technologies like HTML, CSS, and JavaScript or frameworks like React Native, Xamarin, and Flutter (Davidson, 2022).
 Flutter	Google developed the open-source, free mobile app development framework known as Flutter. Applications are created with it for Linux, Mac, Windows, Android, iOS, and the Web. It offers a reactive programming methodology for creating user interfaces and leverages the programming language Dart. It also comes with a large selection of pre-made widgets, enabling quick development and a native-like experience for iOS and Android (Flutter Development: Everything You Need to Know, 2015).
 React native	A well-known open-source framework called React Native realised by Facebook that enables programmers to create mobile apps using JavaScript and React.js. By using the same codebase for both the iOS and Android platforms, it enables developers to create mobile apps that have the same appearance and feel as native apps. The virtual DOM of React Native, which is comparable to React, optimizes updates and enhances the functionality of the app. Additionally, React Native makes it simple to integrate other native modules and third-party libraries (React Native · a Framework for Building Native Apps Using React, 2022).
 PWA - Progressive Web Apps	The user experience of a progressive web app (PWA) is comparable to that of a native mobile app. PWAs utilize cutting-edge web technologies to provide users with quick, dependable, and engaging experiences regardless of their network connection. PWAs can be downloaded and used offline on a user's device. PWAs are simple to find and share because they can be shared via URLs and found using search engines (Progressive Web Apps, 2022). Web technologies including HTML, CSS, and JavaScript are used to create PWAs. They don't need a separate app store and may be accessed through a web browser.

In summary, Flutter is a popular choice for app development due to its hot reload feature that allows for faster experimentation and bug fixing, the easy-to-learn Dart programming language, native-like performance using the Skia graphics engine, customizable widgets, and growing community support with resources such as tutorials and documentation. Whereas Progressive Web Apps (PWAs) are best suited for creating web

applications that provide a user experience similar to that of a native mobile app, PWAs use current web technologies to deliver quick, dependable, and engaging experiences to users regardless of their network connection and can be installed on a user's device. React Native, on the other hand, enables developers to create mobile apps that have the same look and feel as native apps by using the same codebase for both iOS and Android platforms.

Deep Learning	
 Tensorflow	TensorFlow is a comprehensive library for building and deploying machine learning models. It has a large community and is actively developed by Google, making it a popular choice for building complex, large-scale models. TensorFlow also provides a wide range of tools and libraries, such as TensorBoard, for visualizing and debugging models (BasuMallick, 2022).
 Pytorch	PyTorch is an open-source machine learning library developed by Facebook. It is known for its easy-to-use API and dynamic computational graph, which allows for more flexibility when building models. PyTorch also has strong support for GPU acceleration, making it a good choice for training large, complex models (Yasar & Lewis, 2022).
 Keras	Keras is a high-level neural network library written in Python and capable of running on top of TensorFlow. It is designed to be user-friendly and modular, allowing for fast experimentation, fast prototyping, supporting convolutional neural networks (CNNs) and easy model building. Keras can be used as a wrapper for other low-level libraries like TensorFlow and PyTorch, and it is known for its ease of use and readability of the code (Keras, 2019).

In summary, TensorFlow is a powerful, flexible, and production-ready library with a large community, lots of integrations and a rich ecosystem, making it a popular choice for machine learning and deep learning tasks. Where as Keras is a high-level neural network library that is best suited for fast experimentation and prototyping due to its simple and user-friendly API, it supports multiple backends, it's easy to create and train models. PyTorch, on the other hand, is best suited for research and development in deep learning. It offers a dynamic computational graph and built-in support for distributed training, which makes it easy to scale models to large datasets.

Platforms	
android	Google created the well-known open-source Android operating system for mobile devices. It is built on the Linux kernel and can run on a variety of gadgets, such as smartphones, tablets, televisions, and even automobiles (Chen, 2019).
iOS	Apple's iPhone, iPad, and iPod Touch are all powered by iOS, the company's mobile operating system. It is built to work with Apple hardware and is based on the Unix operating system (Posey, 2022).

Reasons to choose Android platform over IOS (Georgiou, 2022):

- A bigger market share means that there is a larger potential user base for Android apps than there is for iOS apps.
- Given that Android is an open-source operating system, developers have more freedom and flexibility in the technologies and tools they can employ, they may find it simpler to construct and test their apps as a result.
- Android can make it simpler to guarantee that an app is compatible with a variety of devices because it is used on a wide range of products from various manufacturers, with various screen sizes and resolutions.
- It is less expensive to create for Android than for iOS. For instance, while the development tools for iOS are proprietary and demand a paid developer account, the tools for Android are free and open source. Additionally, unlike iOS apps, which can only be released through the App Store, Android developers can distribute their apps through a variety of app shops.

There are other factors that might influence a developer's decision, such as the fact that iOS apps often have better income potential, that iOS devices offer a more consistent user experience, and that the rigorous app review process maintains a particular standard of quality. In the end, the platform selection depends on the requirements and objectives of the app development project. In our project since a larger potential user base is the focus to bring this project to as many farmers and agricultural sectors as possible, Android platform was the main choice to bring the project to all users interested in agriculture, cross platform is the ideal solution (Langholz, 2022).

3.8 Model And Feature Selection

Model and feature selection are two important tasks in machine learning, model selection is the process of deciding which model will work best to address a particular issue. The choice of model will rely on the nature of the problem, the quantity and quality of the dataset, and the available computational resources. Different models have different strengths and limitations. Decision trees, random forests, neural networks, and logistic and linear regression are examples of frequently used models. Whereas the process of selecting a subset of the available features to include in a model is referred to as feature selection. By eliminating unnecessary, redundant, or noisy features, feature selection aims to enhance the performance of the model. Additionally, it might make the data less dimensional and the model easier to understand. Techniques like mutual information, correlation-based feature selection, and recursive feature removal can be used for feature selection. Thus, model and feature selection must be done iteratively as part of a process called "model tuning" or "hyperparameter tuning" to create high-performing machine learning models (The Ultimate Guide to Evaluation and Selection of Models in Machine Learning, 2020).

Feature selection can assist prevent overfitting, which happens when a model is overly complicated and performs well on training data but badly on fresh, untested data, in addition to enhancing model performance. Feature selection can streamline the model and improve its ability to generalize to new data by deleting unneeded features. The trade-off between bias and variance is a crucial factor in model and feature selection. While a model with high variance will be more adaptable and is likely to overfit the data, a model with high bias will make strong assumptions about the data and is likely to underfit it. Model and feature selection both depend on finding the ideal balance between bias and variance. Finally, it's crucial to remember that model and feature selection should be carried out in the context of a particular issue and utilizing a particular set of assessment metrics. There isn't a universally applicable answer. For instance, accuracy is an excellent evaluation metric for a classification problem but mean squared error or R-squared value is a superior metric for a regression problem (D'Agostino, 2022).

3.9 Chapter Summary

The chapter is focused on finding the best approach and strategies to develop the plant disease application. To begin development, one must understand the basic concepts of machine learning, its history, importance, types, and future of ML to aid the decision of the type of ML to be used. Then understanding how ML is used in different sectors and environments and how it came to be with each stage of evolution is crucial. To understand the evolution of ML, key aspects to be used and the technologies that are used to develop the application are discussed with the review of the technologies that are used in the model.

CHAPTER 3: METHODOLOGIES

4.1 Chapter Overview

In this chapter, we describe the development methodology in the context of research, and project management. We also provided a work breakdown structure (WBS) in the form of a GANTT chart, which detailed the project's timeframe and deliverables. Furthermore, we have identified some of the major risks involved with the project and how we aim to address them.

4.2 Selected Software Development Methodology

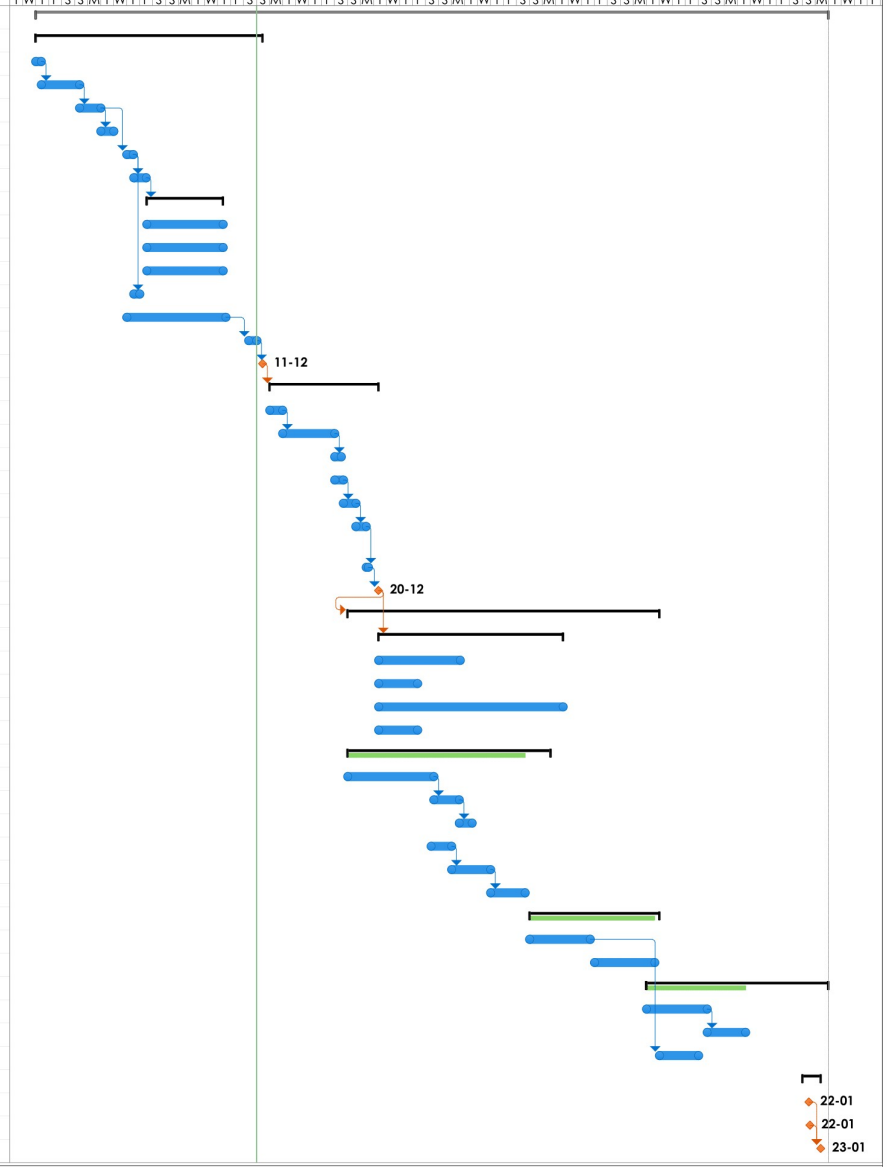
The Rational Unified Process (RUP) is an iterative software development process that we have chosen as our software lifecycle model. It is often referred to as the Unified Process Model. Rational Corporation built it, and it is planned and documented using UML (Unified Modelling Language). This procedure is part of the IBM Rational Method Composer (RMC) product. The unified process can be customized, designed, and personalized thanks to IBM (International Business Machine Corporation). Ivar Jacobson, Grady Bootch, and James Rambaugh proposed RUP. RUP is use-case driven, iterative (process repetition), and incremental (value gain) by nature, supplied online using web technology, and may be adjusted or tailored in modular and electronic form. Because multiple components, altered and portions of the cycle can be repeated until the program matches requirements and objectives, the process is agile (GeeksForGeeks, 2022).

RUP contains four major phases: Inception, Elaboration, Construction, and Transition. Each phase has its own set of goals, activities, and outputs.

- **Inception Phase**: The primary goal is to adequately scope the system as a basis for confirming initial costs and budgets. The business case, which comprises the business context, success factors.
  - Vision statement
  - First use case
  - Project plan
  - Risk Assessment
- **Elaboration Phase**: The major goal is to mitigate the key risk items identified by analysis up to the end of this phase. The elaboration phase is when the project begins to take shape. During this phase, the risk domain study is completed, and the project architecture takes shape (Agile Data Warehousing for the Enterprise, 2016).
  - A use-case model
  - A description of the software architecture used in the implementation of a software system.
  - Risk model
  - Development plan (GANTT)
  - Prototypes
- **Construction Phase**: Construction of the software system
  - Completion of the software system
- **Transition Phase**: Beta testing and project rollout

3.3 Gantt Chart Figure 1 Project Timeline in a GANTT chart

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
0		**Software Development**	**63.25 days**	**Thu 24-11-22**	**Mon 23-01-23**	
1		**1 Project Initiation**	**18 days**	**Thu 24-11-22**	**Sun 11-12-22**	
2		1.1 Team formation	2 hrs	Thu 24-11-22	Thu 24-11-22	
3		1.2 Selection of topic with research	3 days	Thu 24-11-22	Sun 27-11-22	2
4		1.3 Superviser approval	2 days	Sun 27-11-22	Mon 28-11-22	3
5		1.4 Familiarisation with topic	1 day	Tue 29-11-22	Tue 29-11-22	4
6		1.5 Determine project scope	4 hrs	Thu 01-12-22	Thu 01-12-22	4
7		1.6 Identify the goals and objectives	1 day	Thu 01-12-22	Fri 02-12-22	6
8		**1.7 Learning Techincal Material**	**6 days**	**Fri 02-12-22**	**Thu 08-12-22**	**7**
9		1.7.1 Firebase Backend Tutorial	6 days	Fri 02-12-22	Thu 08-12-22	
10		1.7.2 Machine Learning	6 days	Fri 02-12-22	Thu 08-12-22	
11		1.7.3 Flutter Tutorial	6 days	Fri 02-12-22	Thu 08-12-22	
12		1.8 Identify Requirements	4 hrs	Thu 01-12-22	Thu 01-12-22	6
13		1.9 Draft project proposal	8 days	Thu 01-12-22	Thu 08-12-22	
14		1.10 Feedback session with superviser	1 day	Sat 10-12-22	Sat 10-12-22	13
15		1.11 Project Proposal Submission	3 hrs	Sun 11-12-22	Sun 11-12-22	14
16		**2 Elaboration**	**8.38 days**	**Mon 12-12-22**	**Tue 20-12-22**	**15**
17		2.1 Further research	1 day	Mon 12-12-22	Mon 12-12-22	
18		2.2 Literature Review and Documentation	4 days	Tue 13-12-22	Fri 16-12-22	17
19		2.3 Iteration Plan	4 hrs	Sat 17-12-22	Sat 17-12-22	18
20		2.4 Systems architecture	1 day	Sat 17-12-22	Sat 17-12-22	
21		2.5 Detailed evaluation and development plan	1 day	Sat 17-12-22	Sun 18-12-22	20
22		2.6 Incorporate feedback on software specifications	4 hrs	Sun 18-12-22	Mon 19-12-22	21
23		2.7 Revise the use-case model	3 hrs	Mon 19-12-22	Mon 19-12-22	22
24		2.8 Presentation	4 hrs	Mon 19-12-22	Tue 20-12-22	23
25		**3 Construction/ Development**	**24.88 days**	**Sun 18-12-22**	**Tue 10-01-23**	**24**
26		**3.1 Development of mobile application**	**15 days**	**Tue 20-12-22**	**Tue 03-01-23**	**24**
27		3.1.1 Development of UI	7 days	Tue 20-12-22	Mon 26-12-22	
28		3.1.2 Development of Animations	3 days	Tue 20-12-22	Fri 23-12-22	
29		3.1.3 Development of core features	15 days	Tue 20-12-22	Tue 03-01-23	
30		3.1.4 API integration	3 days	Tue 20-12-22	Fri 23-12-22	
31		**3.2 Development of backend services**	**16.38 days**	**Sun 18-12-22**	**Mon 02-01-23**	
32		3.2.1 Develop Machine Learning Model	7 days	Sun 18-12-22	Sat 24-12-22	
33		3.2.2 Train Machine Learning Model	2 days	Sat 24-12-22	Mon 26-12-22	32
34		3.2.3 Testing ML Model accuracy	1 day	Mon 26-12-22	Tue 27-12-22	33
35		3.2.4 User Authentication system	2 days	Sat 24-12-22	Sun 25-12-22	
36		3.2.5 Firebase Security Rules	3 days	Mon 26-12-22	Wed 28-12-22	35
37		3.2.6 Firebase integration	3 days	Thu 29-12-22	Sat 31-12-22	36
38		**3.3 Testing**	**10.38 days**	**Sun 01-01-23**	**Tue 10-01-23**	
39		3.3.1 Blackbox Testing	5 days	Sun 01-01-23	Thu 05-01-23	
40		3.3.2 Unit Testing	5 days	Fri 06-01-23	Tue 10-01-23	
41		**4 Transition/ Documentation**	**14.5 days**	**Tue 10-01-23**	**Mon 23-01-23**	
42		4.1 Documenting Final Report	5 days	Tue 10-01-23	Sat 14-01-23	
43		4.2 Develop user guide	3 days	Sat 14-01-23	Tue 17-01-23	42
44		4.3 Testing Report	3 days	Wed 11-01-23	Fri 13-01-23	39
45		**5 Production/ Closure**	**1.38 days**	**Sun 22-01-23**	**Mon 23-01-23**	
46		5.1 Submission of the Final Report	4 hrs	Sun 22-01-23	Sun 22-01-23	
47		5.2 Submission of Source Code	6 hrs	Sun 22-01-23	Sun 22-01-23	
48		5.3 Demo Application	2 hrs	Mon 23-01-23	Mon 23-01-23	46,47



Project: Software Development
Date: Sun 11-12-22

Task, Split, Milestone, Summary, Project Summary, Inactive Task, Inactive Milestone, Inactive Summary, Manual Task, Duration-only, Manual Summary Rollup, Manual Summary, Start-only, Finish-only, External Tasks, External Milestone, Deadline, Progress, Manual Progress

4.3 Project Deliverables

Deliverables	Date
Project Proposal	12th December 2022
Literature Review Evaluating and analysing chosen modules and comparison	23rd January 2023
Software Requirement Specification The document specifying the requirements which is required to deliver the final product of the project	
System Design Document The document describes the proposed recommendation system's design and overviews of the algorithms to be built.	
Flutter Project App source code	
Deep Learning Model Saved .h5 files for the six models, Saved .tflite files for the six models & Saved optimized .tflite files for the six models.	
App UI Designs & Prototypes Low and High-fidelity wireframes and screenshots.	
Final Document Final Document refers to this Document, which contains all project documentation.	
Android App (Final Software Product) The functioning final mobile application with all the features specified in initial proposal functional requirements.	

Table 2 Project Deliverables

4.4 Deviations & Risk Mitigation

Any project with a significantly longer timeframe will confront several risks during its implementation. It is critical to assess the potential risks and devise mitigation strategies. The table below details the project's perceived risks and mitigation strategies.

Risk	Level	Frequency	Mitigation
Changing Requirements of the System When going through the development cycle, the core requirements and the sub requirements might occur, in which case going through each iteration of the prototype it is being implemented and addressed.	High	High	Utilizing an iterative and incremental development process, such as RUP, will enable requirement changes and the incorporation of new features. It is vital to develop the system in modules so that if requirements change, there are as few deferrals as possible.
Deficiency of datasets In this project we have used deep learning CNN algorithm, for it to predict the diseases, we need to train the DL model and there should sufficient data sets to train the models.	Low	Low	Prior to the project while researching about the concept we discovered that there are free datasets available even before committing to the project. To minimize relying on services or subscriptions, data should be gathered from the beginning. If data is still unavailable, less data-intensive alternatives such as ontologies may be considered.
Inadequate availability and support when it comes to software resources Software components such as APIs, relevant plugins sometimes is unavailable or expensive, therefore inadequate availability.	Low	High	The relevant IDEs, APIs, plugins, and software components are addressed and researched prior to the commencement of the project. Since some of the requirements are simple and composed the software components needed to fulfil can be managed. This needed to be addressed before the commencement of the project since lack of software tools may cause a drawback.
Inadequate availability and provisioning when it comes to hardware resources Deep learning projects can be resource hungry. Since training ML models needs a lot of hardware power, there can be a inadequate provisioning when it comes to hardware resources.	High	High	Hardware requirement analysis prior to the training to get the efficiency report of existing hardware, and if more resources are needed, cloud infrastructure such as Azure, Aws can be used.
Vast and wide variety of subject knowledge is needed with learning resources. Since we use deep learning CNN algorithms and tools such as TensorFlow, Jupyter Notebook,	High	Low	Learning resources are provided from credible sources and implementation support is provided from respective forums when it comes to TensorFlow and flutter.

Keras and several other tools, we are needed to gain variety and broad spectrum of knowledge.			

Table 3 Top risks and mitigation

4.5 Chapter Summary

Rational Unified Process (RUP) is an iterative software development process that we choose as our software lifecycle model. Because of its iterative nature and implementation flexibility, we hope to continuously improve and test features to fit our changing requirements over the course of the project. Following that, we presented a work breakdown structure (WBS) and its corresponding Gantt Chart. We also conducted a Risk Analysis, which is essential prior to the start of the project.

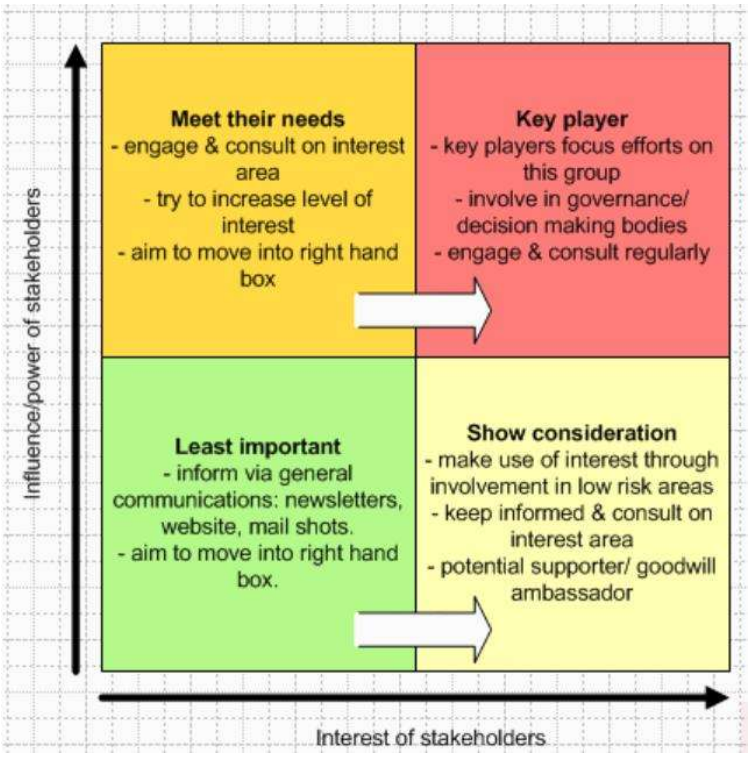
CHAPTER 4: SOFTWARE REQUIREMENTS SPECIFICATION

5.1 Chapter Overview

A System Requirement Specification (SRS) is a document that describes the system requirements for a software or hardware project. It outlines the functional and non-functional requirements, constraints, and any other information needed to develop and maintain the system. The SRS serves as a contract between the development team and the client and is used as a reference throughout the development process to ensure that the final product meets the client's needs. The SRS is typically created at the beginning of the project and is reviewed and approved by all stakeholders before development begins.

5.2 Stakeholder Analysis

Stakeholder analysis is the process of identifying and assessing the interests, needs and impact of individuals or groups who have an interest in a project or organization. Stakeholder analysis is the process of identifying and assessing the interests, needs and impact of individuals or groups who have an interest in a project or organization.



5.3 Requirement Elicitation Methodologies

The goal of requirement elicitation is to gather complete, accurate, and consistent information about the stakeholders' needs and requirements for the system. It is important to involve a diverse group of stakeholders in the requirement elicitation process to ensure that all perspectives and needs are considered.

5.3.1 Brainstorming

This involves bringing stakeholders together in a group setting to generate ideas and gather information about their needs and requirements for the system.

Findings: As citizen and as software engineering our team was able to brainstorm idea and features the system can offer farmers to resolve the problem. Requirements and goals were collected by the team during the preliminary research phase of the project. Brainstorming was ideal solution to gather requirements without ethical clearance. To identify all the essential requirements for the application, we were also able to combine brainstorming with other elicitation techniques.

5.3.2 Observation

This involves observing the stakeholders using similar systems or performing tasks related to the system being developed, to gather information about their needs and requirements.

By conducting research on existing plant disease diagnostic apps, we were able to identify key features and functionality that users find valuable. We identified the important areas for improvement in these existing applications and technology and used that information to develop our application ensuring that it offers a more comprehensive and user-friendly experience for the stakeholders. The research also provided an insight and opportunity to identify latest trends and technologies used in similar apps to help us stay relevant and up to date with the current technology. Overall, observation was a valuable tool in gathering requirements and improvements to the functionality of our application.

5.3.3 Reverse Engineering

This involves analysing existing systems or similar systems that are in use by the stakeholders to gather information about their needs and requirements.

Findings: There are some current systems to discuss, but none of them are user-friendly or capable of learning from the needs and interests of a user to make recommendations. And we analysed similar systems that used CNN algorithm for deep learning.

5.4 SWOT Analysis

Strengths	Weaknesses
• The use of deep learning algorithms allows for high accuracy in detecting plant diseases. • The app can process large amounts of data quickly and efficiently. • The app can be easily integrated with existing systems and devices, such as smartphones or drones.	• The app may require a significant amount of training data to achieve high accuracy. • The app may be affected by a lack of data diversity, leading to lower accuracy for certain plant species or disease types. • The app may be affected by variability in the quality of the images used for detection.
Opportunities	**Threats**
• The app can be used in various sectors, such as agriculture and horticulture, to improve crop yield and reduce losses due to disease. • The app can be integrated with precision agriculture technology to provide real-time disease detection and treatment. • The app can be used to monitor and track the spread of plant diseases, which can help to prevent outbreaks and improve crop management.	• Limited awareness of the app among potential users. • Difficulty in obtaining accurate data. • Lack of funding to support further development and integration of the app. • Limited acceptance of the app by farmers and agricultural experts. • Competition from other similar apps and technologies.

5.5 Functional & Non-Functional Requirements

The MoSCoW method is a prioritization technique used to categorize requirements into four categories: Must-have, Should-have, Could-have, and Would-like-to-have. The MoSCoW principle can be used to prioritize the requirements and ensure that the most important requirements are addressed first.

Priority Level	Description
Must Have (**M**)	Requirements that are necessary for the successful completion of the project.
Should Have (**S**)	Requirements that are important for project completion but not necessary.
Could Have (**C**)	Requirements that are nice to have but have a very small impact when left out of the project.
Will not have (**W**)	Requirements that the system may not have and that are not considered a top priority at this time.

5.5.1 Functional Requirements

FR ID	Requirement	Priority Level
FR1	The user could be able to register for an account.	M
FR2	The user should be able to authenticate himself/herself.	M
FR3	The user must be able to select the type of plant.	M
FR4	The user must be able to take/insert a picture using the mobile app.	M
FR5	The app must analyse the given image using the trained model.	M
FR6	The app must be able to predict and display the disease type using ML.	M
FR7	The app should be able to show a description of the predicted disease.	S
FR8	The app could let the user input the location or get the live location from the user's mobile when a disease has been found.	C
FR9	The app could display a map with all the locations where diseases have been found. (Using the data collected by all users)	C
FR10	The app could display the disease detection history of the user.	C
FR11	The user could be able to export the history of the detection results as a PDF document.	C
FR12	The app will not have a notification system.	W
FR13	The app will not diagnose the diseases which are not included in the datasets.	W
FR14	The app will not permit video sources as input.	W

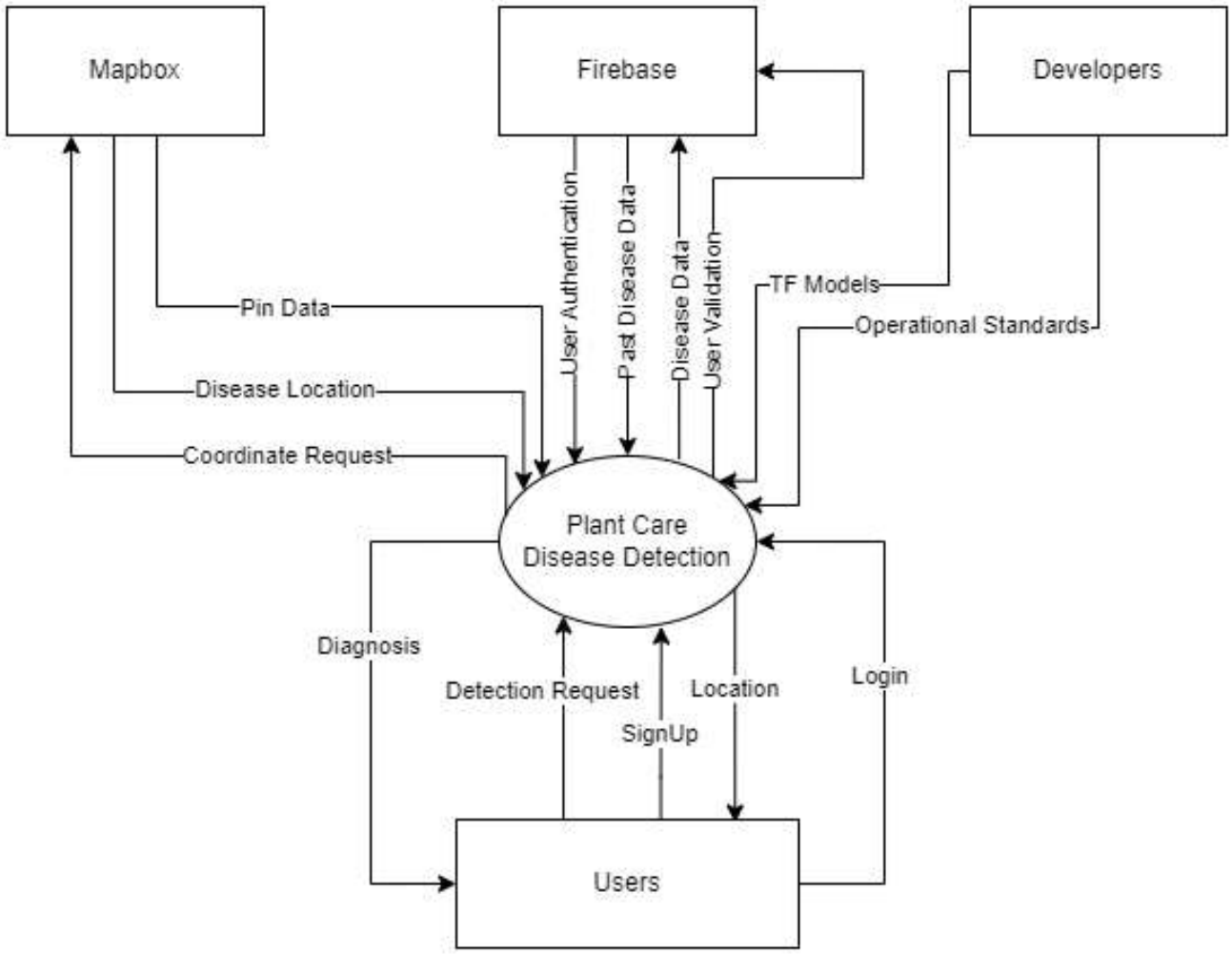
5.5.2 Non-Functional Requirements

NFR ID	Requirement	Specification	Priority Level
NFR1	The user must be able to access the app on the android platform.	Accessibility	Important
NFR2	The user should have a unique ID number to locate identity in the system.	Security	Desirable
NFR3	The diagnosed diseases should have an accuracy of over 70%.	Accuracy	Important
NFR4	The app should diagnose the disease within 30 seconds.	Performance	Important
NFR5	The system UI should be easy to understand and use.	Usability	Desirable
NFR6	The ML model used for the app should be able to improve over time. (Can train the model with an improved/larger dataset to detect new diseases)	Scalability	Desirable

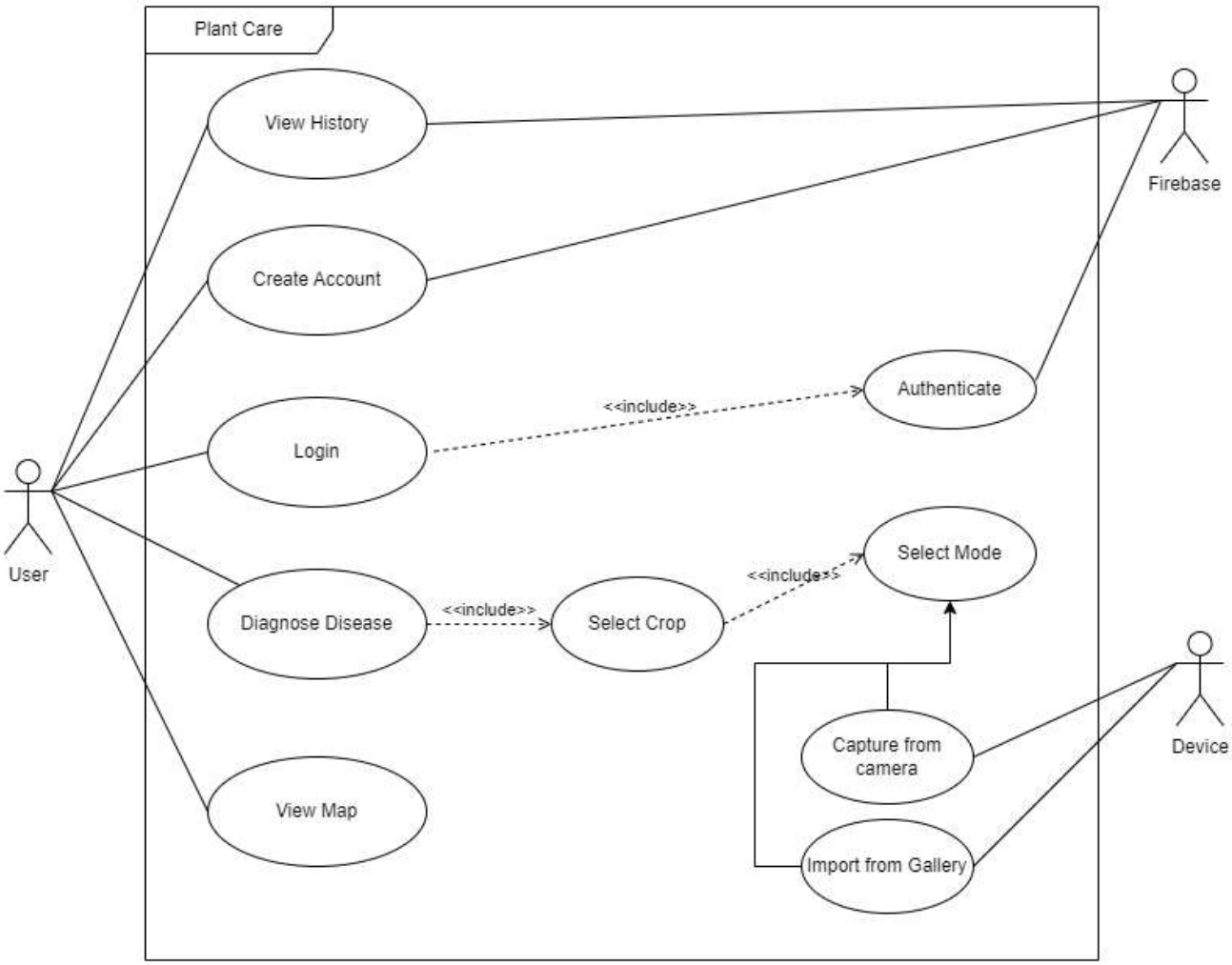
Additionally:

- **Usability**: The system should be easy to navigate with clear instructions held together with a modernized interface.
- **Scalability**: The system should handle and process the load of many end users with many plant species without a reduction in performance.
- **Performance**: The system should identify plant diseases less than 5 seconds accurately along with useful disease descriptions.
- **Security**: The system should endure user data is secure and protected with security measures.
- **Reliability**: The system should repeat the same functionalities without anomalies.
- **Availability**: The system should be in a functional state 99% of the time with measures taken to reduce total system failure by ensuring parts of the modules are on different servers so that in case one fails, the other remain functional.
- **Compatibility**: mobile application should work across many android devices with varying display sizes and hardware
- **Maintainability**: The system should be easy to maintain and update with comprehensive design and documentation. App should also be expandable in terms of its plant dataset to detect many types of plants over time with the appropriate diseases.
- **Testability**: The system should be tested thoroughly with a complete test plan and test cases that cover the application's functionality.
- **Accuracy**: The system should provide accurate readings with 99% accuracy and confirm the probability of the disease type to the end user.

5.6 Context Diagram



5.7 Use Case Diagram



5.8 Use Case Elaborations

5.8.1 Login

USE CASE	Login	
Use Case ID	1	
Brief Description	User can access their account and do the actions	
Primary Actor	User	
Level	Primary	
Trigger	User selects “login” prompt	
Secondary Actor	Firebase	
Pre-conditions	User must have an existing registered account	
Minimal guarantees	The system prompts the user to reenter the username and password again.	
Post-condition	User will gain the access to the account	
Main Flow	Step	Action
	1	The user clicks on the Login feature
	2	The user grants fields to enter email address and the password
	3	User enters the credentials
	4	The system authenticates the data
	5	The system redirect to the user’s profile
Alternative Flow	Step	Branching Action
	1.1 1.2	If a user does not have an existing account, the system will prompt the user to create an account. If user forgot his password, User can select the “forgot password” option. The system allows user to answer his security questions.

5.8.2 Create Account

USE CASE	Create Account	
Use Case ID	2	
Brief Description	User can create an account to become a registered user.	
Primary Actor	User	
Level	Primary	
Trigger	User selects “Sign up” prompt	
Secondary Actor	Firebase	
Pre-conditions	User should have an existing email address	
Minimal guarantees	A pop-up message will appear to enter the required fields.	
Post-condition	User will be given the access to the application.	
Main Flow	Step	Action
	1	The user clicks on create account feature
	2	The user enters required data
	3	The system validates the information
	4	Information is stored in the user account
	5	System notifies that the account has been created

5.8.3 View History

USE CASE	View History	
Use Case ID	3	
Brief Description	Allows the user to view their plant disease history which is stored in the user’s account.	
Primary Actor	User	
Level	Primary	
Trigger	User selects “view history”	
Secondary Actor	Firebase	
Pre-conditions	User must login to the application	
Minimal guarantees	The system will display the history page of the user account.	
Post-condition	The user can view all historical records consisting of plant diseases that were diagnosed.	
Main Flow	Step	Action
	1	The user clicks on View History feature.
	2	The system retrieves user historical data from the database
	3	The system displays all historical data

5.8.4 Diagnose Disease

USE CASE	Diagnose Disease	
Use Case ID	4	
Brief Description	User can select the crop and upload a photo from the device gallery or take a photo from the camera for the system to diagnose the disease.	
Primary Actor	User	
Level	Primary	
Trigger	User clicks on “Diagnose Disease”	
Secondary Actor	Device	
Pre-conditions	User should be logged in to the app.	
Minimal guarantees	The system will prompt the user to login. The system will upload the image file to diagnose the disease.	
Post-condition	The system will display the diagnosed disease.	
Main Flow	Step	Action
	1	User clicks on Diagnose Disease feature
	2	User uploads an image file
	3	The system runs ML algorithm to process and diagnose the disease.
	4	User prompt “Diagnose Disease”
Alternative Flow	Step	Branching Action
	1.1 1.2	The user can select crop type before uploading the image. The user can select upload mode. Import from gallery. Capture from camera

5.8.5 View Map

USE CASE	View Map	
Use Case ID	5	
Brief Description	All the plant disease location data gathered via the app is stored in the system. Thus, letting the app users view a Map consisting of the locations of a specific disease.	
Primary Actor	User	
Level	Primary	
Trigger	User clicks on “View Map” feature.	
Secondary Actor	N/A	
Pre-conditions	1. Gathered data should be processed and stored with location labels. 2. The user accessing map data must be authenticated.	
Minimal guarantees	The system will display a map	
Post-condition	The system will view a map with accurate disease locations based on user data.	
Main Flow	Step	Action
	1	User clicks on View Map feature.
	2	The system shows the respective plant disease locations on map.

5.9 Design Class Diagram

The following Design Class Diagram displays the objects or classes that makes the system, as well as the navigability between interacting classes. It also demonstrates the public and private characteristics of a class. These are the classes defined in the database, their public and private properties, and the activities required to implement the system's main functionalities.

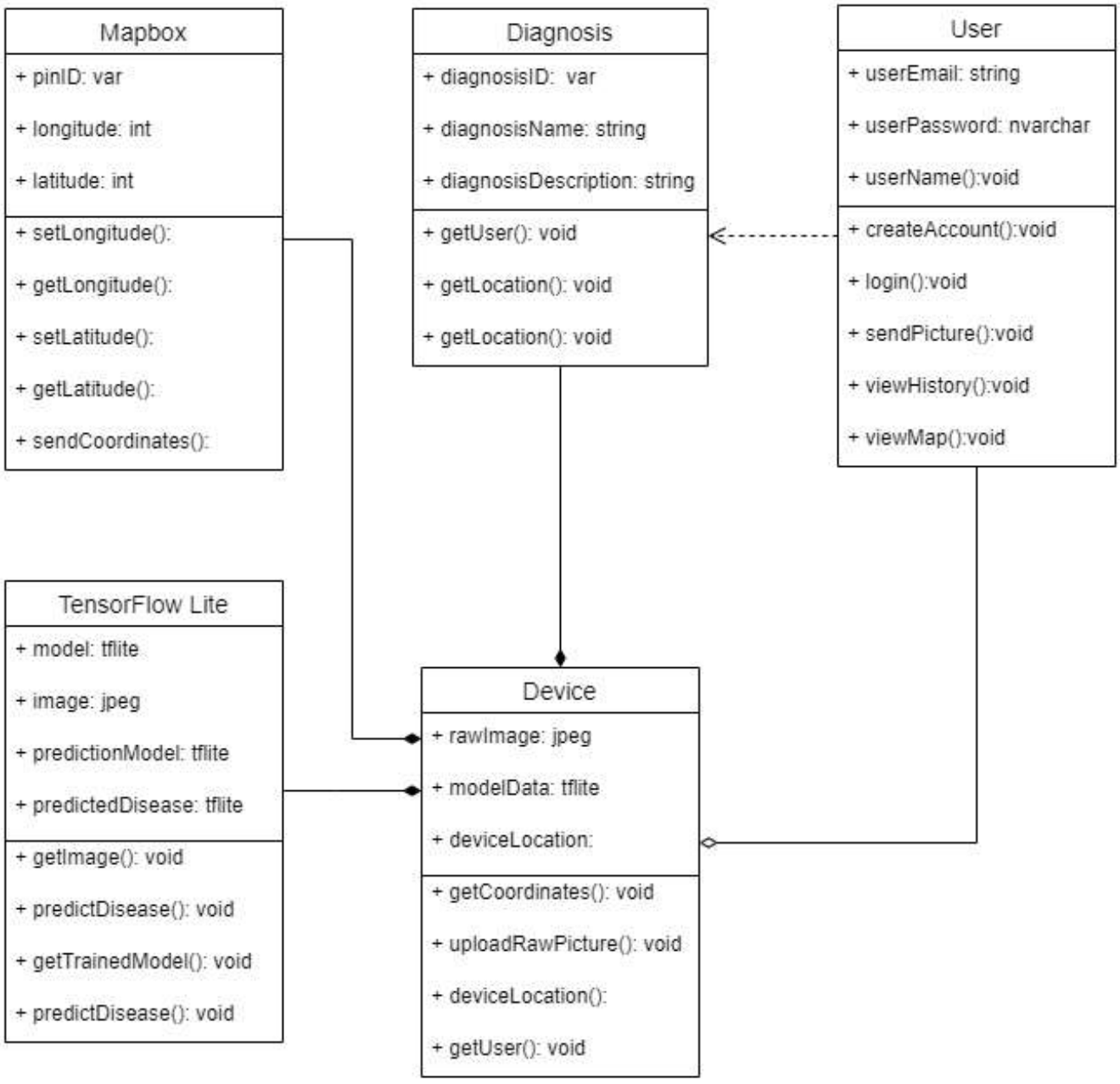
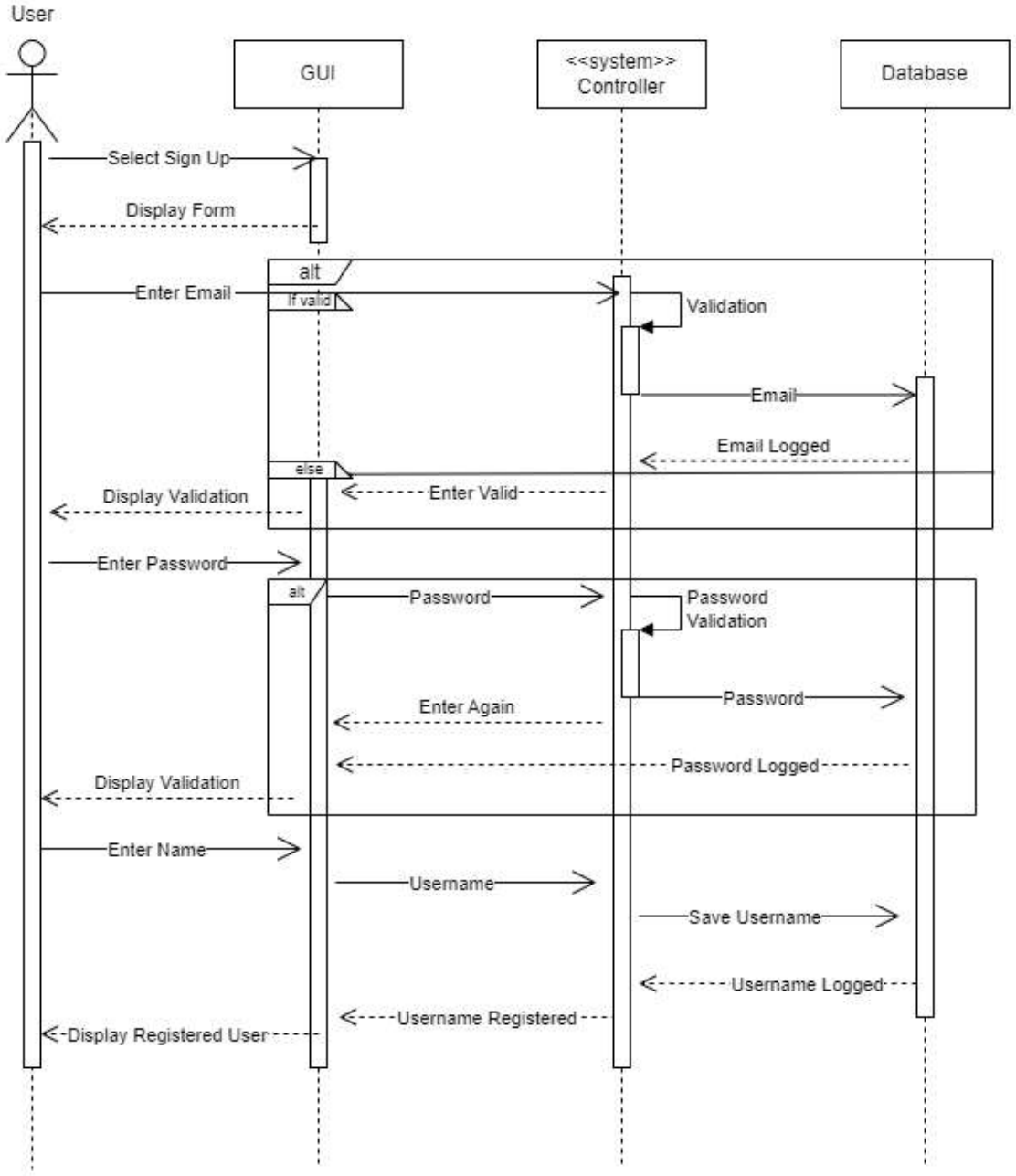


Figure 13: Design Class Diagram

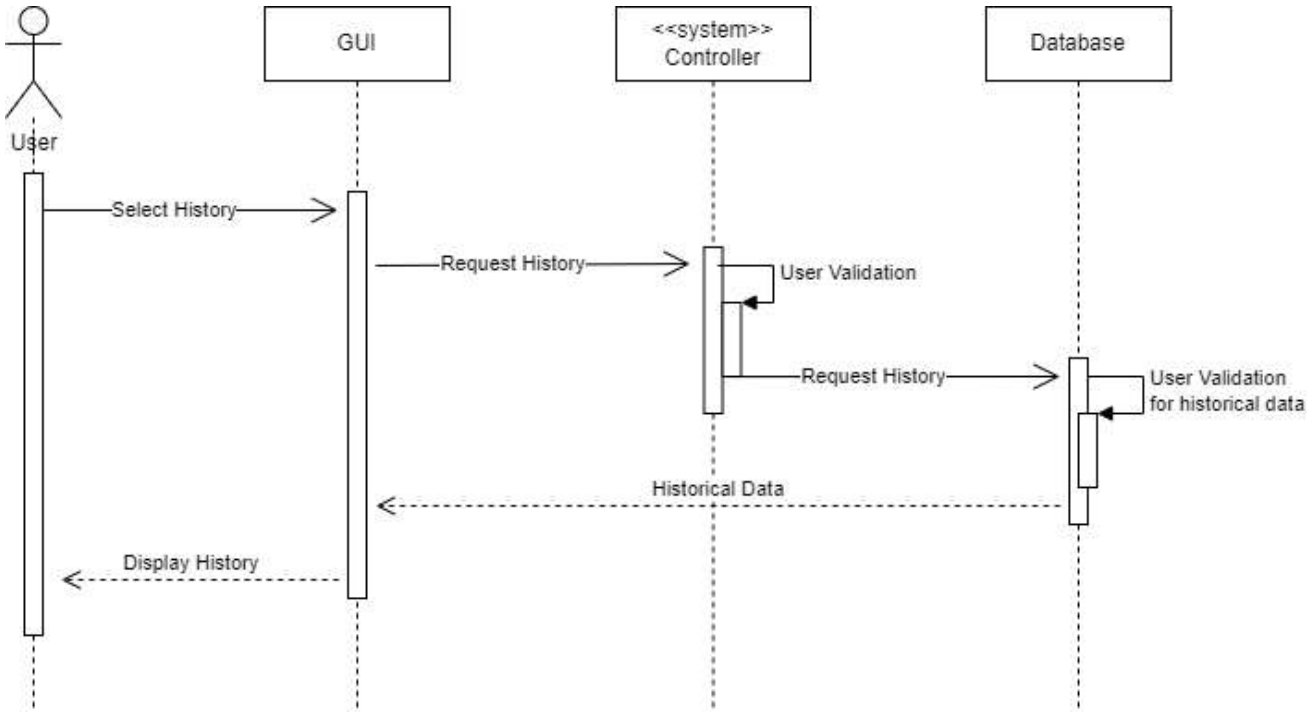
5.10 Sequence Diagrams

Here are the sequence diagrams which presents the structured order sequence of diagnosis and the various functions or use cases of the system upon the use case elaboration. This is to present the interactive behaviour of a system, behaviour of the classes and how the functions are performing in the system. It is illustrating how the order in which the interactions occur when a particular use case is executed. Lifeline represents a class instance (object) and the consequent cases that follow as well as certain operators.

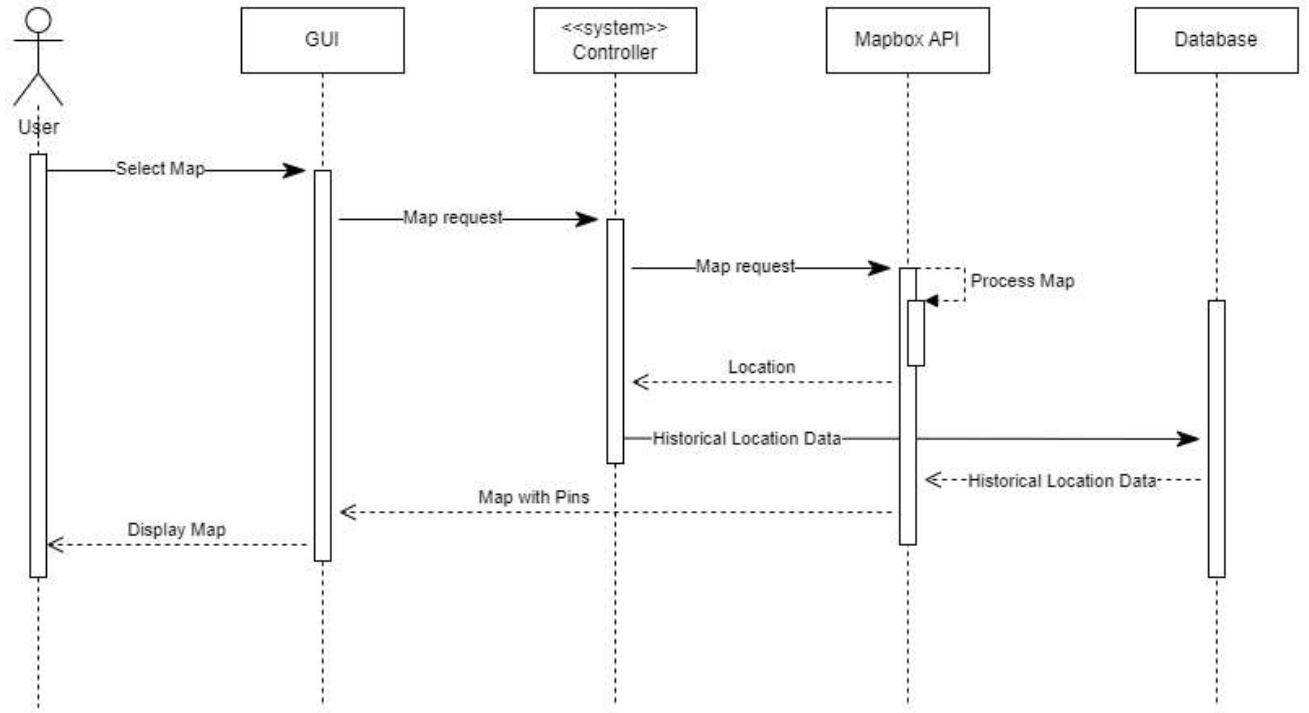
5.10.1 Sequence Diagram for Create Account



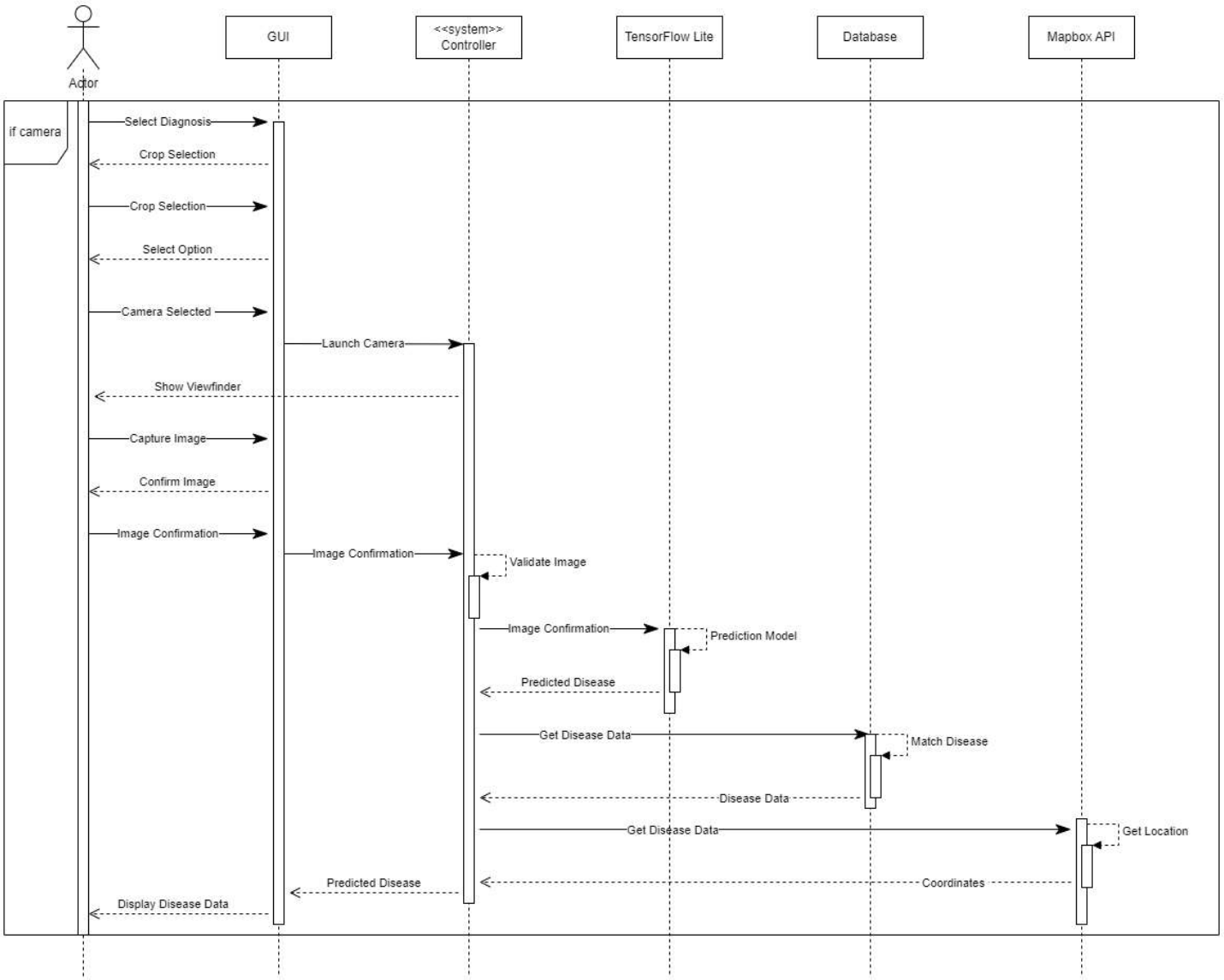
5.10.2 Sequence Diagram for View History



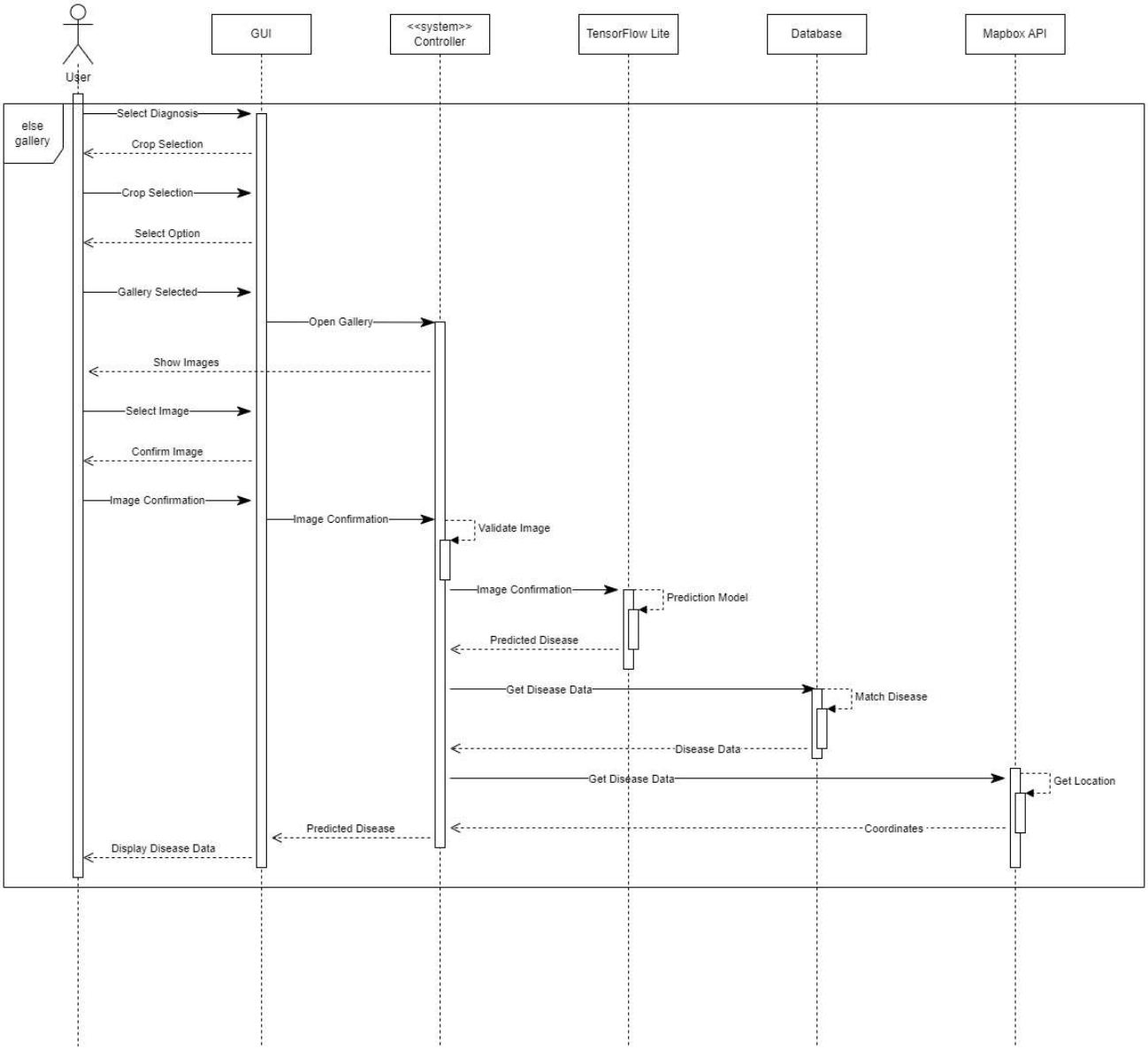
5.10.3 Sequence Diagram for View Map



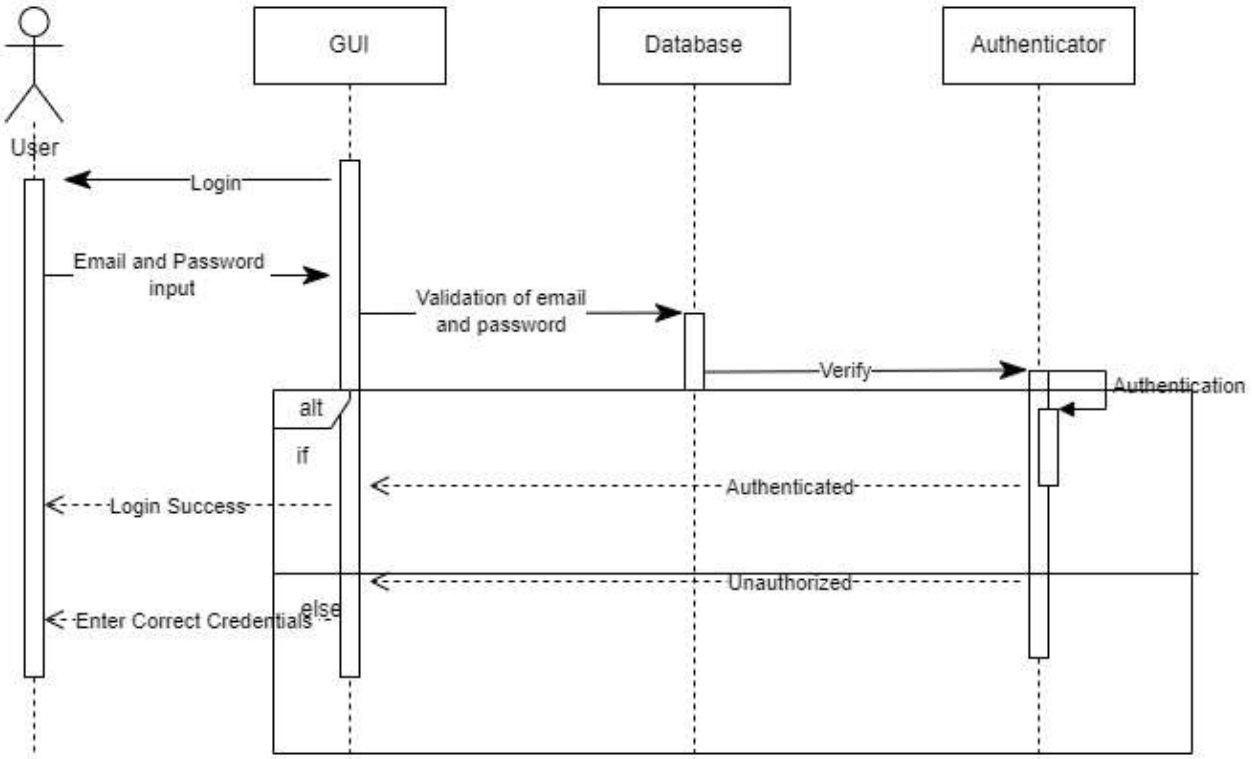
5.10.4 Sequence Diagram for Diagnose Disease with Camera



5.10.5 Sequence Diagram for Diagnose Disease choosing with gallery



5.10.6 Sequence Diagram for Login



5.11 Chapter Summary

This chapter covered the process of requirement gathering, classification of stakeholders and their engagement with the project. To collect requirements, many methods of requirement elicitation were studied and applied. Functional and non-functional requirements were stated. Reverse Engineering, SWOT analysis, overall use case diagram, use case elaborations and diagrams such as context diagram were presented in this chapter. All observations, findings, conclusions, and summarizations were included.

CHAPTER 5: DESIGN SPECIFICATION

6.1 Overview

Here we will go through the design principles and specification required for software development and output. High-level representation diagram that represents the system's architecture and implementation, the three-tiered architecture which shows the system divided into three tiers which are the presentation tier, logic tier and data tier, and low-level designs such as the Design Class Diagram and the Component Diagram. The Sequence Diagrams are shown for specified Use Cases in the Use Case Specification. We also show the UI Wireframe, which will display the system's navigation in a snapshot. All of information is displayed alongside rationalizations for the resolutions made.

6.2 Design Principles

Reusability - The ability to expand or add features, as well as extend properties and characteristics from one class to another, is a vital facet of our program architecture. One important factor is that the reuse of one component must be possible without the subsequent reuse of other unrelated components.

Correctness - Our application's goal was to create a recommender engine that proposes tailored events to consumers based on their specific tastes. As a result, correctness would attempt to make sure that the system behaves as described.

Scalability - This refers to the ability of components to be flexible and adjust to inevitable changing needs, as well as the ability of components to be used again independently. One more important point to remember is that reducing one component of the system should not cause the entire system to crash. As a result, we must allow for individual freedom within the operating environment.

Adaptability - This refers to components' ability to be flexible and adjust to changing needs, as well as with their ability to be used independently again. Some other aspect to keep in mind is that reducing one component of the system should not cause the entire system to crash. As a result, individual freedom within the operating environment must be allowed.

Maintainability - In this section, the recommender system's overall architecture is described. The architecture diagram that follows illustrates the relationships, restrictions, and boundaries between components as well as the overall shape of the suggested solution. The user's symbol stands for any users who interact with the developed app, including app users, administrators, and organizers. A bidirectional interface between the machine learning engine and the firebase database enables the system to recommend events to active users based on their previous utilization information from the database.

6.3 High-Level System Architecture Diagram

The overall architecture of the recommender system is described in this section. The architecture diagram that follows abstracts the general structure of the proposed solution, as well as the relationships, restrictions, and boundaries between components. The user's symbol represents users who interact with the produced app, such as app users, organizers, and administrators. The machine learning engine and the firebase database have a bi-directional interaction that allows the system to recommend events to active users based on their previous user data from the database.

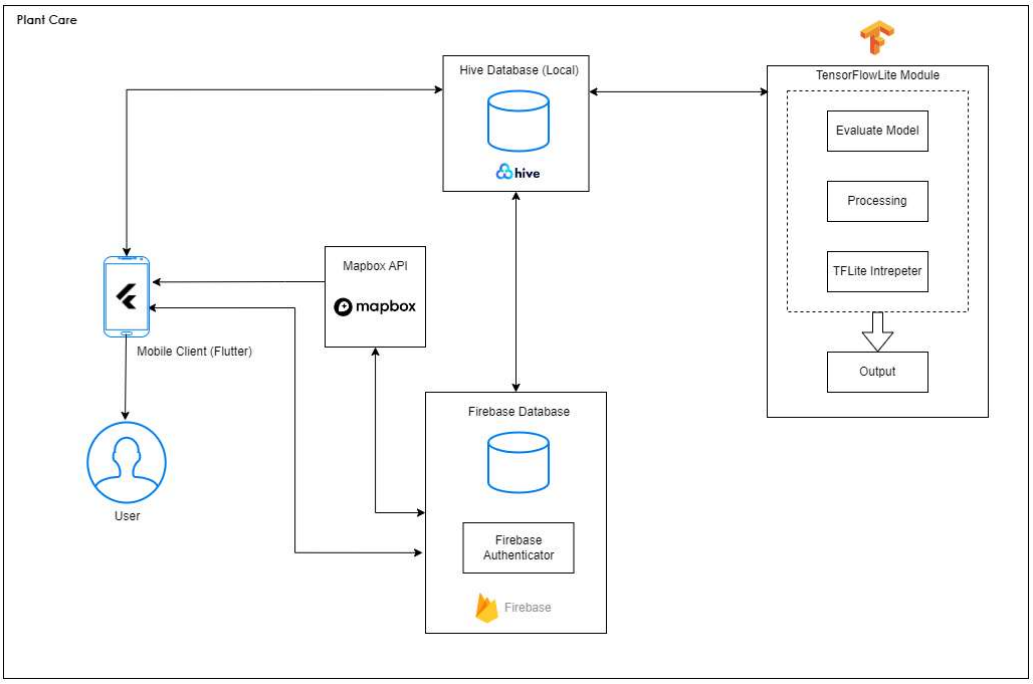


Figure 2 High-Level System Architecture Diagram

6.4 Data Flow Diagram

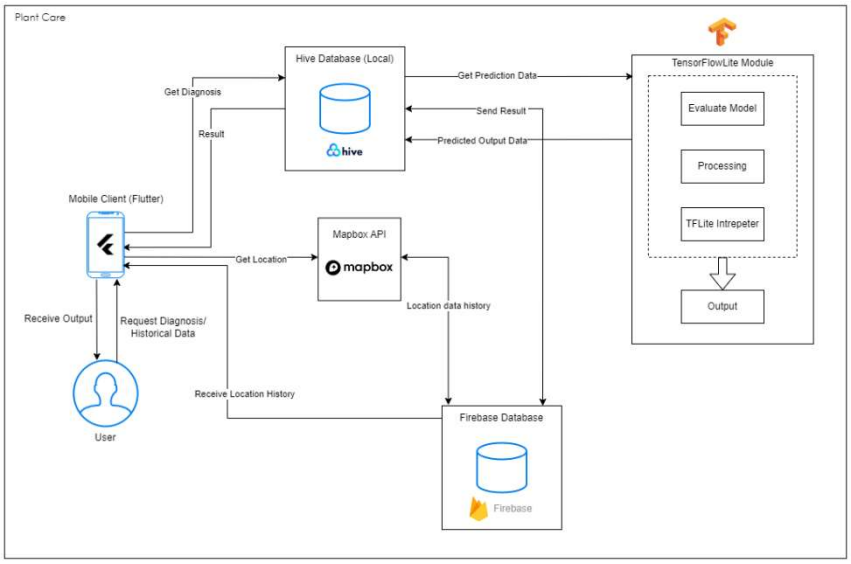


Figure 12: Dataflow Diagram

6.5 Three-Tiered Architecture Diagram

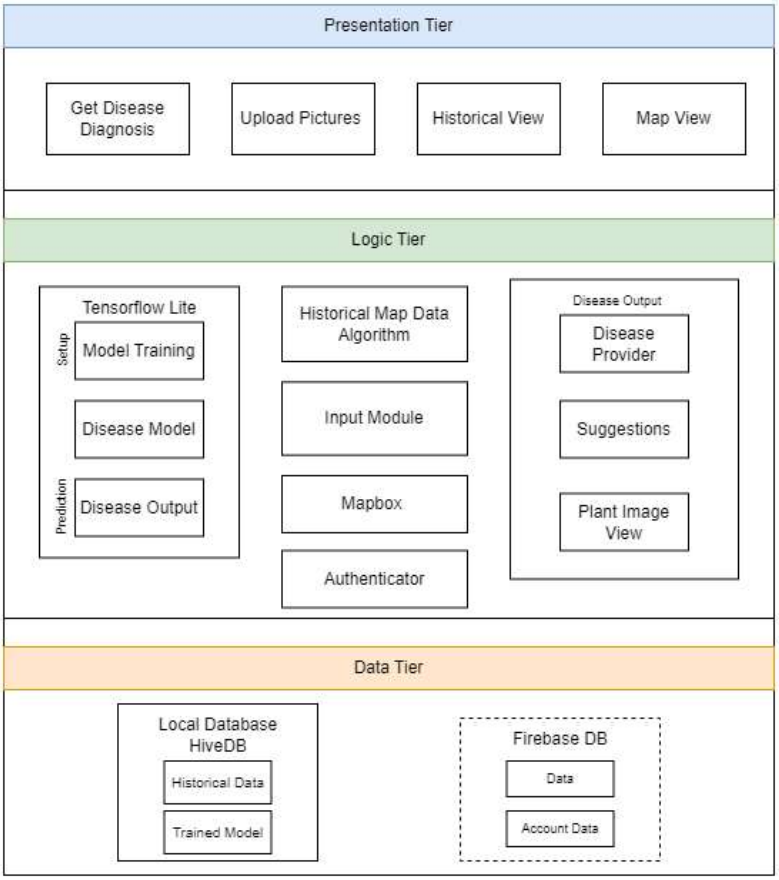


Figure 3: Three-tiered Architecture

6.6 UI Wireframe

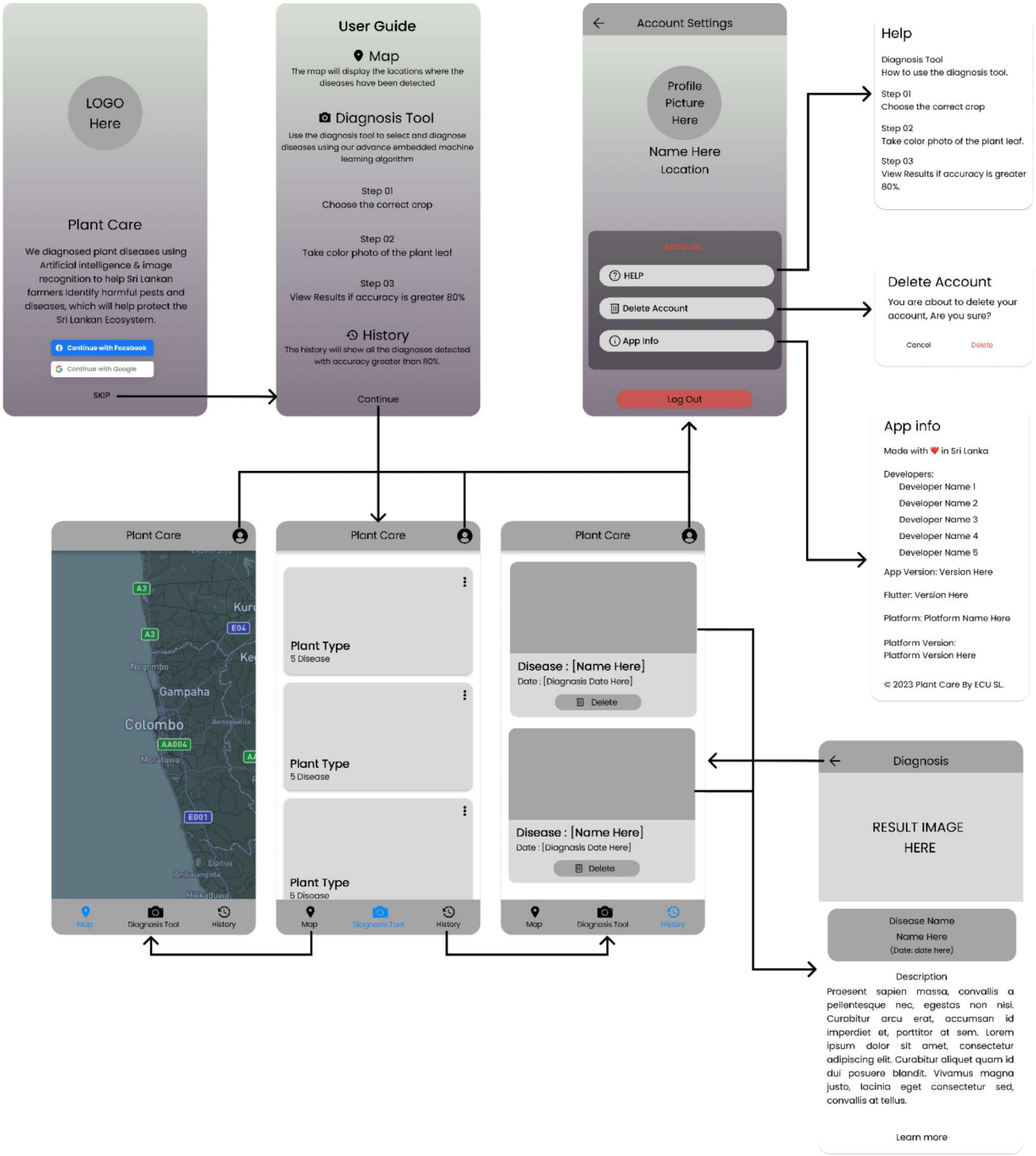


Figure 4 UI Wireframe

There are four main screens which the user will go through, which are Home, Diagnosis, Map and History screens. Initially user is navigated to the home page which the user is given the option to Login with an existing account, sign up with google or skip the login or sign up, since if the user does not want to proceed with the sign up or login.

After a successful login, signup or otherwise skip, the user is directed to the diagnosis screen directly, which show cases the main functionalities of the system. Five types of crops which are cinnamon, coffee, tea, tomato, and rice are shown as widgets for the user to select depending on the crop, after selection of the crop, user is prompted with two options to select the mode of uploading the picture of the crop selected. The two options are Take a Picture or Choose a photo. If the user selects the camera the user will be connected to the camera API, then after confirmation of the image captured, the image will be processed, and the user will be directed to the results of the diagnosis. Same as the Take a Picture module, if the user chooses to upload an existing photo for the diagnosis the user can go to the file explorer and choose the image and it will be processed.

The diagnosis results screen will showcase the results upon processing the users uploaded picture. The results page will display the disease name, processed date, image uploaded by the user, description of the disease and a learn more button which will direct the user to the source of information.

The history screen will display all of the historical data or results as a list which was processed for the user in a widget manner. The user can select one result which will direct the user to the selected result screen. It will display the past diagnosis result with the date of the result.

In the Map module, the user will be able to view a navigable and interactable map which spans the map screen. It will display pins which it shows case the locations of diagnoses.

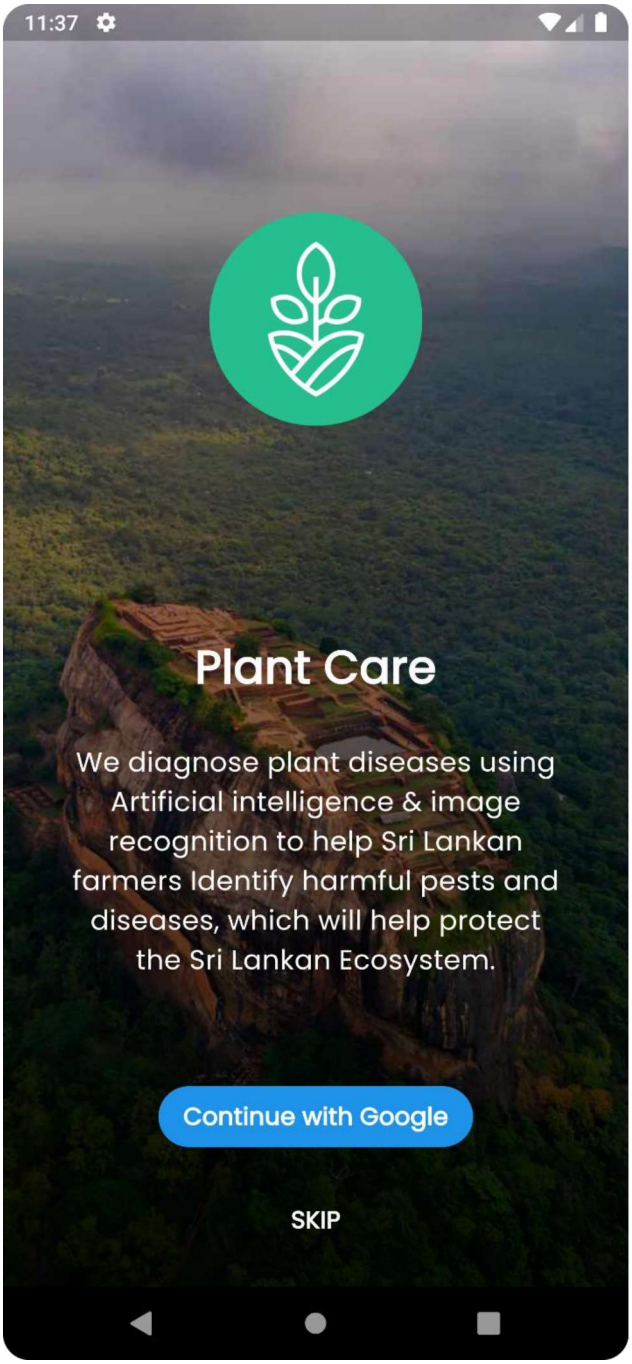
The UI will also consist with the header which will display the application name Plant Care and the Account button which the user can select and view the user details.

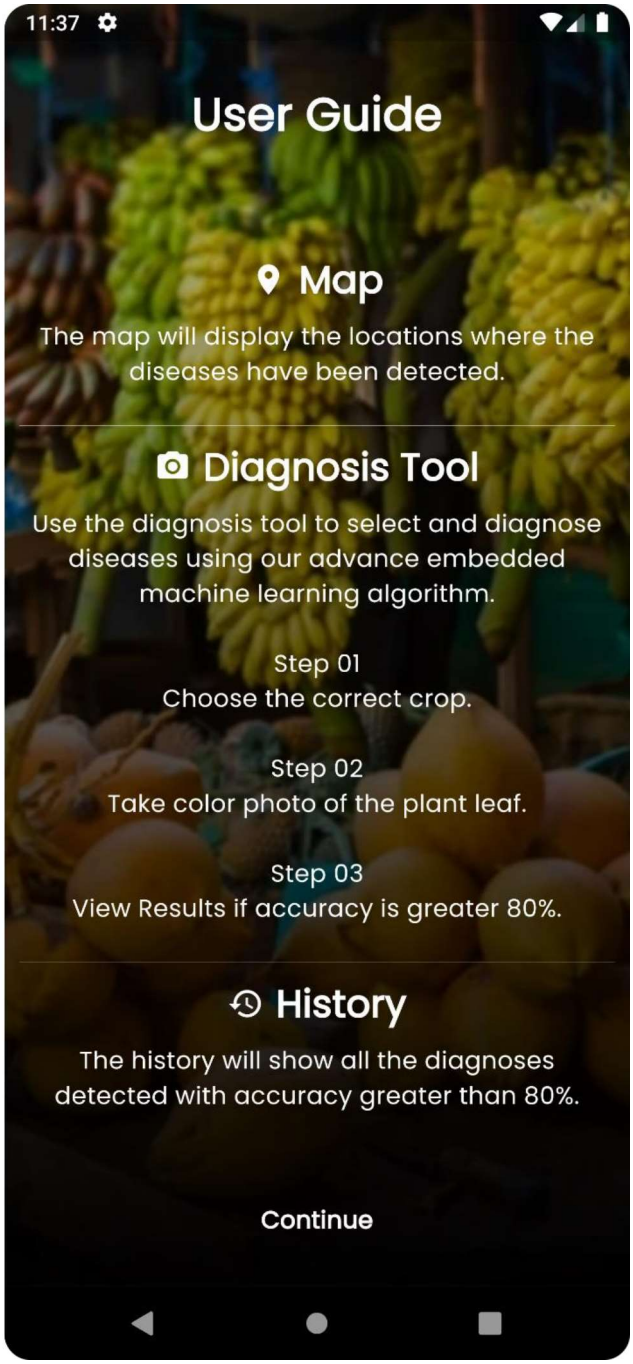
6.7 Chapter Summary

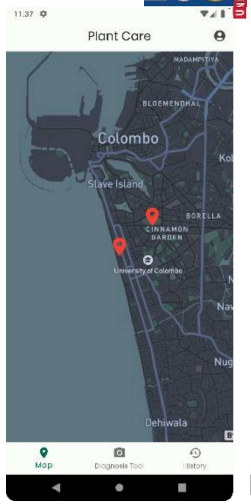
This chapter discussed the most important features of system design. This chapter also includes the design diagrams which are high-level and low-level of the system and architecture.

6.8 Screenshot

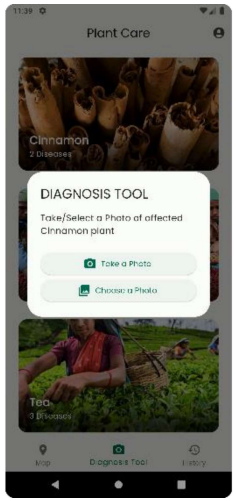
Screenshot for Light mode and Dark mode.

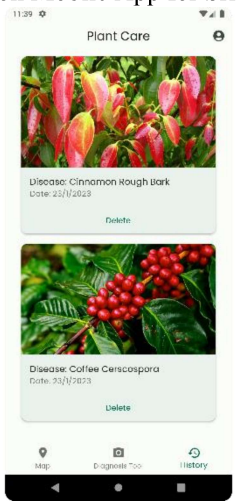




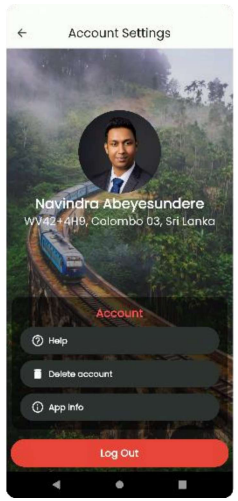

Plant Care
Colombo
Slave Island
Dehiwala
Map
Diagnosis Tool
History

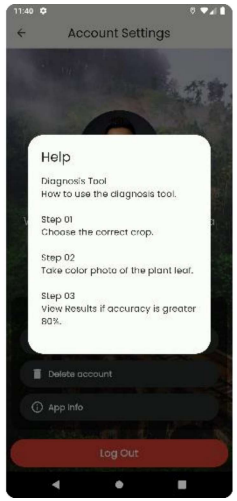

Plant Care
Cinnamon
2 Diseases
Coffee
3 Diseases
Tea
3 Diseases
Map
Diagnosis Tool
History

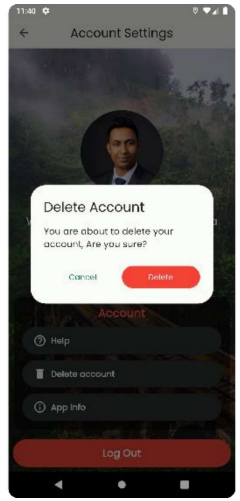

Plant Care
DIAGNOSIS TOOL
Take/Select a Photo of affected Cinnamon plant
Take a Photo
Choose a Photo
Map
Diagnosis Tool
History

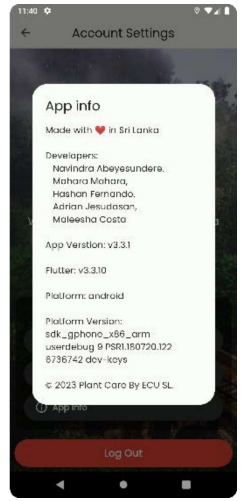

Plant Care
Disease: Cinnamon Rough Bark
Date: 23/1/2023
Delete
Disease: Coffee Cerscospora
Date: 23/1/2023
Delete
Map
Diagnosis Tool
History

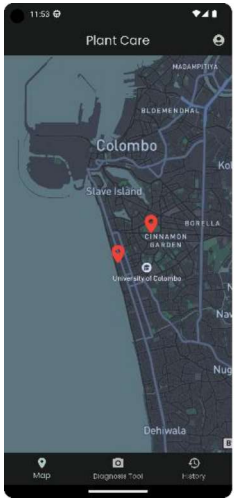

Diagnosis
Disease Name
Cinnamon Rough Bark
(Date: 23/1/2023)
Description
Rough bark disease is the most common disease of cinnamon which affects on young bark of immature shoots as brown spots and spread gradually throughout the bark. Leaves of the infested plants show clorosis and under severe conditions infected immature plants will die. Diseased bark can not be peeled. Disease can be controlled by destroying diseased plant and through adoption of correct cultural practices. Harvesting should be done at correct intervals and excess lateral branches to be removed. As a chemical treatment 1% Bordeaux


Account Settings
Navindra Abeyesundere
WV42+4H9, Colombo 03, Sri Lanka
Account
Help
Delete account
App info
Log Out

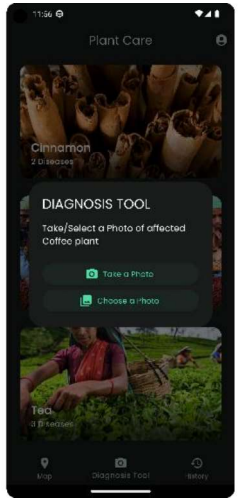

Account Settings
Help
Diagnosis Tool
How to use the diagnosis tool.
Step 01
Choose the correct crop.
Step 02
Take color photo of the plant leaf.
Step 03
View Results if accuracy is greater 80%.
Delete account
App info
Log Out

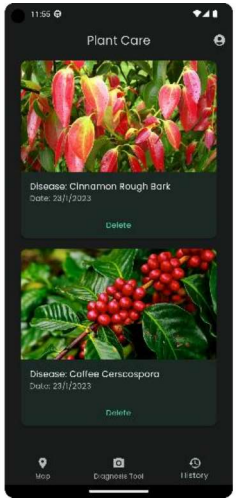

Account Settings
Delete Account
You are about to delete your account, Are you sure?
Cancel
Delete
Account
Help
Delete account
App info
Log Out


Account Settings
App info
Made with ❤ in Sri Lanka
Developers:
Navindra Abeyesundere,
Mahara Mahara,
Hashan Fernando,
Adrian Jesudasan,
Maleesha Costa
App Version: v3.3.1
Flutter: v3.3.10
Platform: android
Platform Version:
sdk_gphone_x86_arm
userdebug 9 PSR1.180720.122
8736742 dev-keys
© 2023 Plant Care By ECU SL.
App info
Log Out

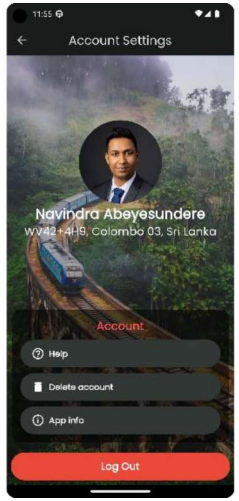

Plant Care
Colombo
Slave Island
Dehiwala
Map
Diagnosis Tool
History

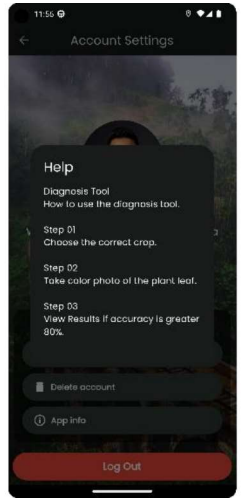

Plant Care
Cinnamon
2 Diseases
Coffee
3 Diseases
Tea
3 Diseases
Map
Diagnosis Tool
History

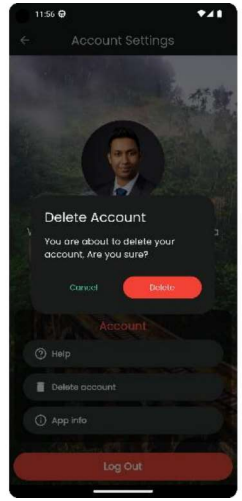

Plant Care
DIAGNOSIS TOOL
Take/Select a Photo of affected Coffee plant
Take a Photo
Choose a Photo
Map
Diagnosis Tool
History

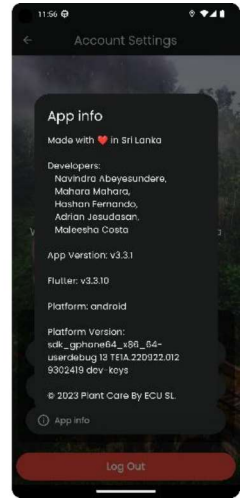

Plant Care
Disease: Cinnamon Rough Bark
Date: 23/1/2023
Delete
Disease: Coffee Cerscospora
Date: 23/1/2023
Delete
Map
Diagnosis Tool
History


Diagnosis
Disease Name
Coffee Cerscospora
(Date: 23/1/2023)
Description
This disease, also called Iron Spot, is caused by the fungal pathogen, Cercospora coffeicola and tends to present itself on coffee plants grown in areas of higher moisture and rainfall and on plants that are stressed. Common symptoms of this disease are 'brown eye-spots' with a slight yellow halo on the leaves and brown, sunken, oval-ish blotches with a slight purplish halo on the cherry. These blotches can resemble sunburn.
Learn more


Account Settings
Navindra Abeyesundere
WV42+4H9, Colombo 03, Sri Lanka
Account
Help
Delete account
App info
Log Out


Account Settings
Help
Diagnosis Tool
How to use the diagnosis tool.
Step 01
Choose the correct crop.
Step 02
Take color photo of the plant leaf.
Step 03
View Results if accuracy is greater 80%.
Delete account
App info
Log Out


Account Settings
Delete Account
You are about to delete your account, Are you sure?
Cancel
Delete
Account
Help
Delete account
App info
Log Out

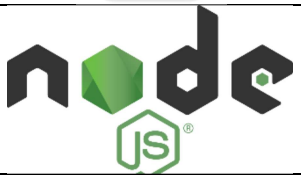
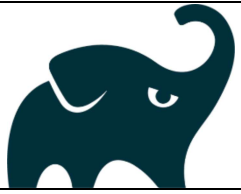

Account Settings
App info
Made with ❤ in Sri Lanka
Developers:
Navindra Abeyesundere,
Mahara Mahara,
Hashan Fernando,
Adrian Jesudasan,
Maleesha Costa
App Version: v3.3.1
Flutter: v3.3.10
Platform: android
Platform Version:
sdk_gphone64_x86_64-
userdebug 13 TE1A.220922.012
9302419 dev-keys
© 2023 Plant Care By ECU SL.
App info
Log Out

CHAPTER 6: IMPLEMENTATION

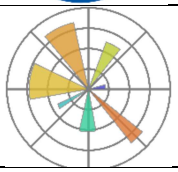
7.1 Chapter Overview

This chapter delves into the implementation of the project, including information on the technology, tools, and languages used, the justification for the development choices, the implementation of core features, and any difficulties encountered during the implementation process.

7.2 Selection of Languages & Tools

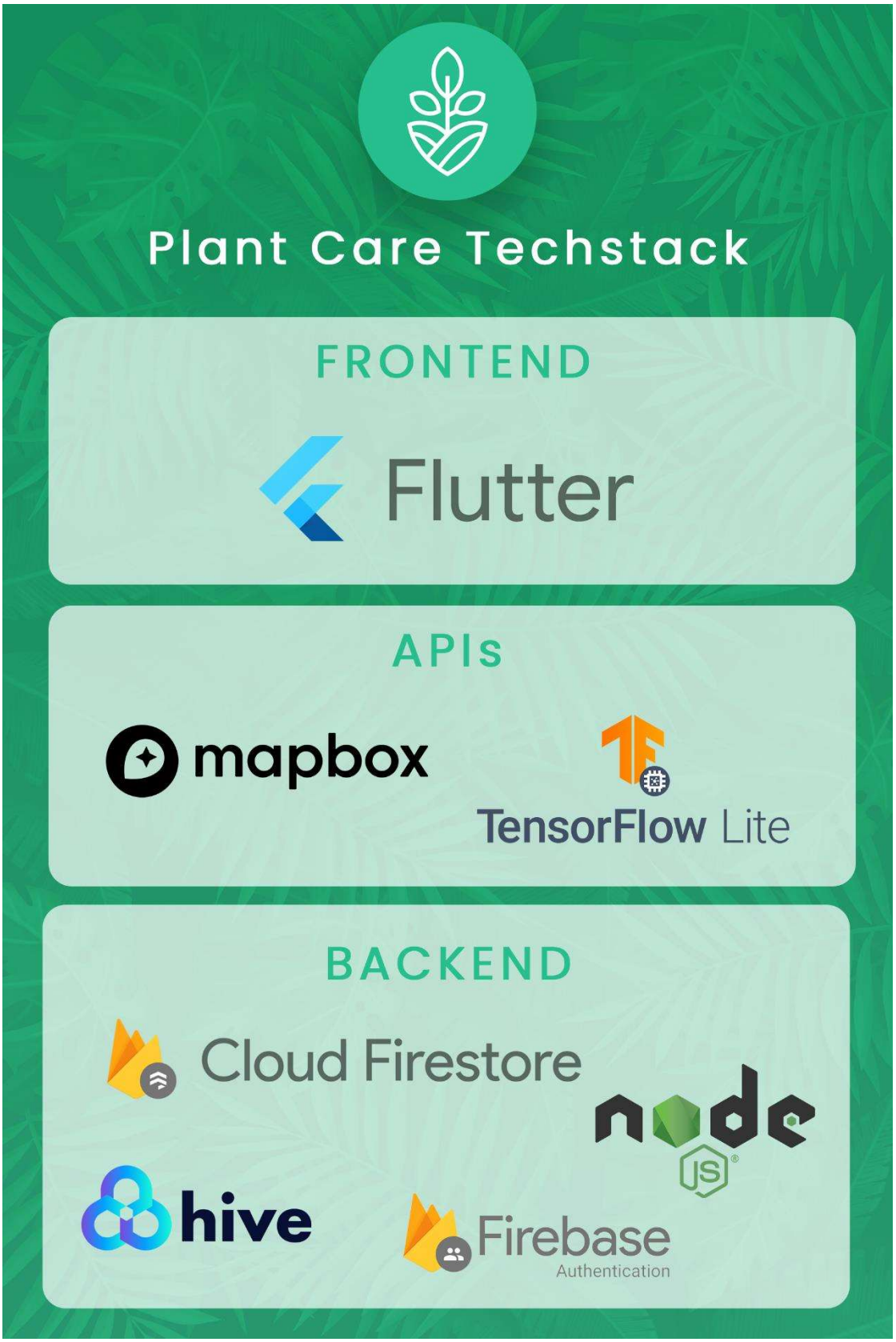
Languages		
	**Python**	Used for data analysis, building, and deploying models in machine learning.
	**Dart**	Client optimized programming language used to build the mobile application.
	**Java 19**	A programming language that is used in machine learning primarily for building and deploying models.
	**NodeJS**	Used for training and deploying models using JavaScript libraries like TensorFlow.js in machine learning.
	**Flutter**	Cross platform mobile app development framework used for machine learning tasks, such as image classification, using libraries such as TensorFlow Lite.
	**Gradle**	Used to manage dependencies and build machine learning models using libraries such as TensorFlow and PyTorch.

Tools		
	**VS Code**	A code editor mainly used for software development, but it can also be used for machine learning tasks, it has various extensions and plugins that support machine learning development.
	**Android Studio**	An IDE mainly used for developing Android apps, but it can also be used for machine learning tasks such as building and deploying models on Android using TensorFlow Lite.
	**Anaconda Environment**	A distribution that provides a convenient environment for managing packages, dependencies and running Jupyter Notebook for machine learning tasks.
	**Jupyter Notebook**	An interactive web-based platform that allows writing, executing and visualizing code, used in machine learning for prototyping, development, documentation of models and experiments.
	**Git**	A version control system that allows for tracking changes, collaborating, and managing the source code of the project.
	**XCode**	Used for developing iOS, macOS apps, and can also be used for machine learning tasks such as building and deploying models on iOS and macOS using Core ML.
Toolkit		
	**CUDA toolkit**	A parallel computing platform developed by NVIDIA, used to accelerate the training and inference of deep learning models using libraries such as TensorFlow and PyTorch.
Deep Learning Libraries		
	**TensorFlow**	An open-source machine learning library that provides a flexible ecosystem for building, training, and deploying a wide range of models,

		used for deep learning, image recognition, and natural language processing.
	**Karas**	An open-source neural network library written in Python used in machine learning as an interface to run Tensorflow and other backends.
	**SciPy**	A Python library that provides functions for optimization, signal and image processing, and other tasks that are used in machine learning.
	**Matplotlib**	A plotting library for Python that is used in machine learning to visualize and analyse data and results of models.
	**scikit-learn**	A machine learning library for Python that provides a consistent interface for various algorithms and tools for tasks such as classification, regression, clustering and model selection.
	**seaborn**	A data visualization library in Python that is used in machine learning to explore the data and detect patterns and outliers.
Flutter Dependencies		
image_picker	Picking images from the image library or using the device camera.	
tflite	Accessing TensorFlow Lite API, Supports image classification, object detection.	
provider	Used for managing a piece of data around the app	
hive	Local database for the app	
hive_flutter	Makes it easier to use with flutter	
url_launcher	Allows to open the web browser, map applications as well as email, etc.	
flutter_map	Provides the mapping interface for the application.	
latlong2	A lightweight library for common latitude and longitude calculation	

geolocator	This plugin provides a cross-platform (iOS, Android) API to get the generic location (GPS)	
geocoding	A Flutter Geocoding plugin which provides easy geocoding and reverse-geocoding features	
firebase_core	Responsible for connecting the Flutter app to the Firebase project.	
firebase_auth	Enables Android and iOS authentication using passwords, phone numbers for user identity.	
cloud_firestore	A NoSQL document database that can easily store, sync, and query data for the mobile app.	
API		
	Mapbox	Used for adding location-based features to application, it has machine learning capabilities for geospatial data analysis.
Database		
	Firebase	A mobile application development platform that provides backend services, used in machine learning for deploying models and serving predictions in real-time using Firebase ML Kit.
	Hive	Offers pre trained models and custom model development services, is better at analysing complex datasets.

7.3 Technology Selection



7.4 Implementation of Core Functionalities

7.4.1 Deep Learning Model Training & Implementation

7.4.1.1 Gathering Data.

Data collection is a vital stage in developing a Deep learning model since the model's quality depends on the training data quality. The issue or task the model is being developed to tackle should be represented in the datasets used to train the model. A model will only produce precise predictions or judgments if trained on a wide and varied data collection. Furthermore, the data must be error-free, consistent, and devoid of outliers. A Deep learning model cannot learn and produce accurate predictions without a significant quantity of high-quality data.

7.4.1.2 Building CNN Model by Adding Necessary Layers

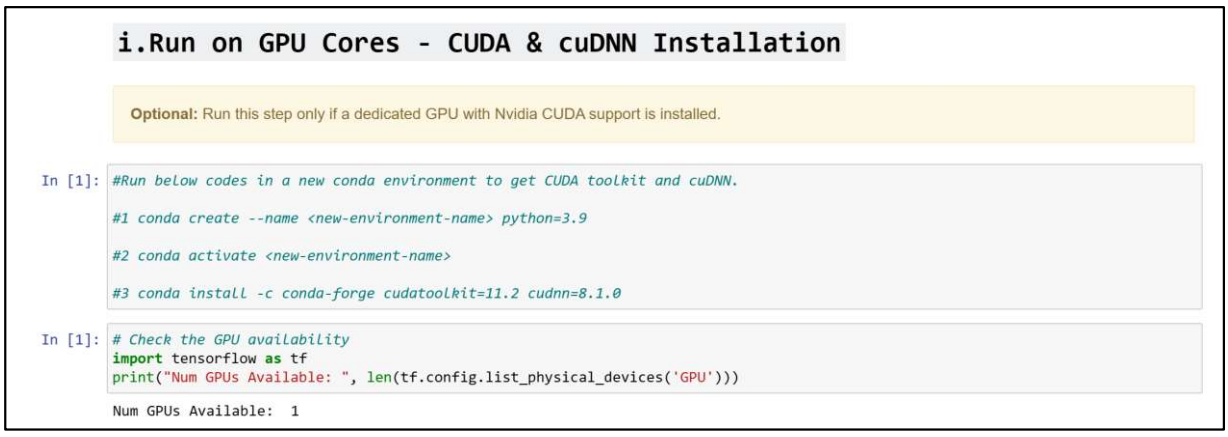
Wide-used open-source libraries like TensorFlow and Keras were used to construct the convolutional neural network (CNN) model, and five layers have been added.

a. Convolution Layer
b. ReLU Layer
c. Pooling Layer
d. Flattening Layer
e. Fully Connected Layer

These layers were thoroughly explained in a previous chapter.

7.4.1.3 Training The Models

PREREQUISITE - i



Before training the models, a new anaconda environment needs to be created with python 3.9 support using the #1 code line since it removes irrelevant installed libraries that might cause conflicts/errors in the code.

Then the CUDA and cuDNN libraries can be installed into the conda environment to use GPU and speed up the model training process only if a dedicated GPU is available with CUDA support.

The code line in the second identifies any available GPUs which can be used to run the training process.

PREREQUISITE - ii

```
ii.Installing Packages

In [3]: # !pip install tensorflow==2.6
        # !pip install keras=2.6
        # !pip install scipy
        # !pip install matplotlib
```

These code lines install the necessary packages into the environment; TensorFlow and Keras are used to build the model. Scipy is used for technical and scientific computing(BrainStation®, 2022), and Matplotlib is used for plotting accuracy graphs.

STEP 1

```
1.Importing Libraries

In [1]: # Import Libraries
        import warnings
        warnings.filterwarnings("ignore")

        import tensorflow as tf
        import os
        import glob
        import matplotlib.pyplot as plt

        # Keras API
        import keras
        from keras.models import Sequential
        from keras.layers import Dense,Dropout,Flatten
        from keras.layers import Conv2D,MaxPooling2D,Activation,AveragePooling2D,BatchNormalization
        from keras.preprocessing.image import ImageDataGenerator
```

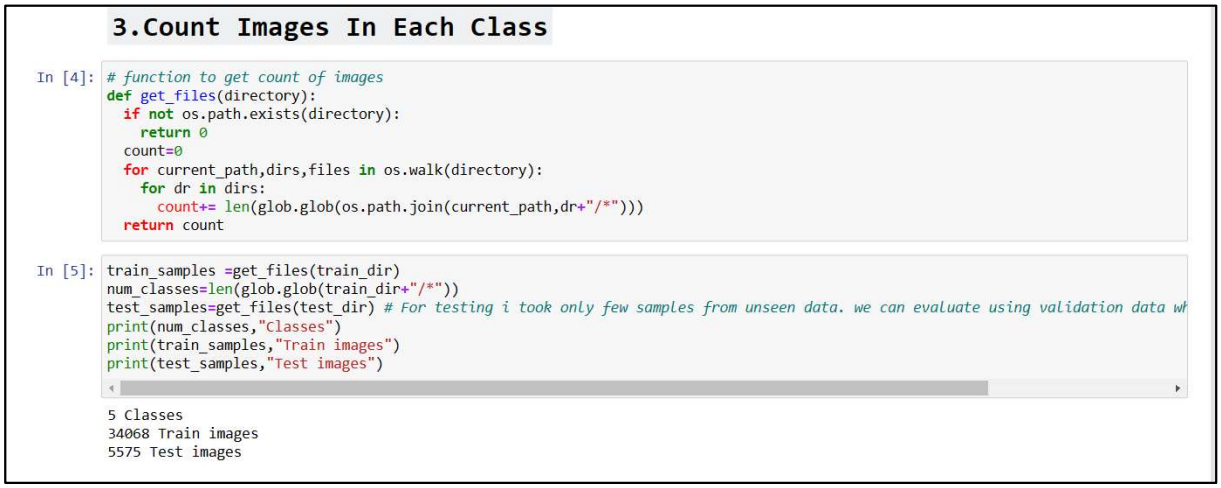
The first code cell imports all the necessary libraries to the Jupyter notebook's .ipynb file.

STEP 2

```
2.Loading the Dataset into Train and Test Variables

In [2]: train_dir = "Dataset/Train0/"
        test_dir  = "Dataset/Test0/"
```

This step sets the train_dir and test_dir variables with train and test dataset paths, respectively.

STEP 3



3.Count Images In Each Class

In [4]:
```
# function to get count of images
def get_files(directory):
  if not os.path.exists(directory):
    return 0
  count=0
  for current_path,dirs,files in os.walk(directory):
    for dr in dirs:
      count+= len(glob.glob(os.path.join(current_path,dr+"/*")))
  return count
```

In [5]:
```
train_samples =get_files(train_dir)
num_classes=len(glob.glob(train_dir+"/*"))
test_samples=get_files(test_dir) # For testing i took only few samples from unseen data. we can evaluate using validation data wh
print(num_classes,"Classes")
print(train_samples,"Train images")
print(test_samples,"Test images")
```

```
5 Classes
34068 Train images
5575 Test images
```

The first cell code defines a function that can count the train and test dataset images. The second cell code uses that function and prints the results. According to that, the dataset includes 34068 Train Images and 5575 Test images.

STEP 4

4.Preprocessing the Data

In [6]:
```
# Preprocessing data.
train_datagen=ImageDataGenerator(rescale=1./255,
                                 shear_range=0.2,
                                 zoom_range=0.2,
                                 validation_split=0.2, # validation split 20%.
                                 horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [7]:
```
# set height and width and color of input image.
img_width,img_height =256,256
input_shape=(img_width,img_height,3)
batch_size =32

train_generator =train_datagen.flow_from_directory(train_dir,
                                                   target_size=(img_width,img_height),
                                                   batch_size=batch_size)
test_generator=test_datagen.flow_from_directory(test_dir,shuffle=True,
                                                   target_size=(img_width,img_height),
                                                   batch_size=batch_size)
```

```
Found 34068 images belonging to 5 classes.
Found 5575 images belonging to 5 classes.
```

Data pre-processing converts and arranges data into an appropriate format to train the deep-learning model. It aims to eliminate noises and inconsistencies in the dataset. Further, it is an essential stage in constructing a deep learning model because of the enormous influence it may have on the model's performance.

Here in the first cell,

The "**rescale**" property is used to normalize the pixel values of the images by dividing all pixel values by 255. It scales the pixel values to the range of 0 to 1.

The "**shear_range**" and "**zoom_range**" properties randomly apply, shearing and zooming to the images. This helps to increase the diversity of the training set and prevent overfitting by providing the model with more variations of the same image.

The "**validation_split**" is used to split a portion of the training data, 20% in this case, for validation. It assists in evaluating the performance of the model during training.

The "**horizontal_flip**" is used to randomly flip the images horizontally, which increases the diversity of the training set and helps the model generalize better.

In the second cell,

"**img_width,img_height**" is used to set the width and height to 256px.

"**input_shape"** is used to set the number of channels of the input images.

"**batch_size**" sets the number of images to 32 to be processed in one training step.

STEP 5

5.Build CNN Model

In [9]:
```
# CNN building.
model = Sequential()
model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(32, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128,activation='relu'))
model.add(Dense(num_classes,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 252, 252, 32)      2432
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 84, 84, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 82, 82, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 41, 41, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 39, 39, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 19, 19, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 23104)             0
_________________________________________________________________
dense (Dense)                (None, 512)               11829760
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 645
=================================================================
Total params: 11,926,245
Trainable params: 11,926,245
Non-trainable params: 0
_________________________________________________________________
```

This code creates a CNN model by stacking layers on top of each other. The architecture is defined by adding different types of layers to the model in sequence.

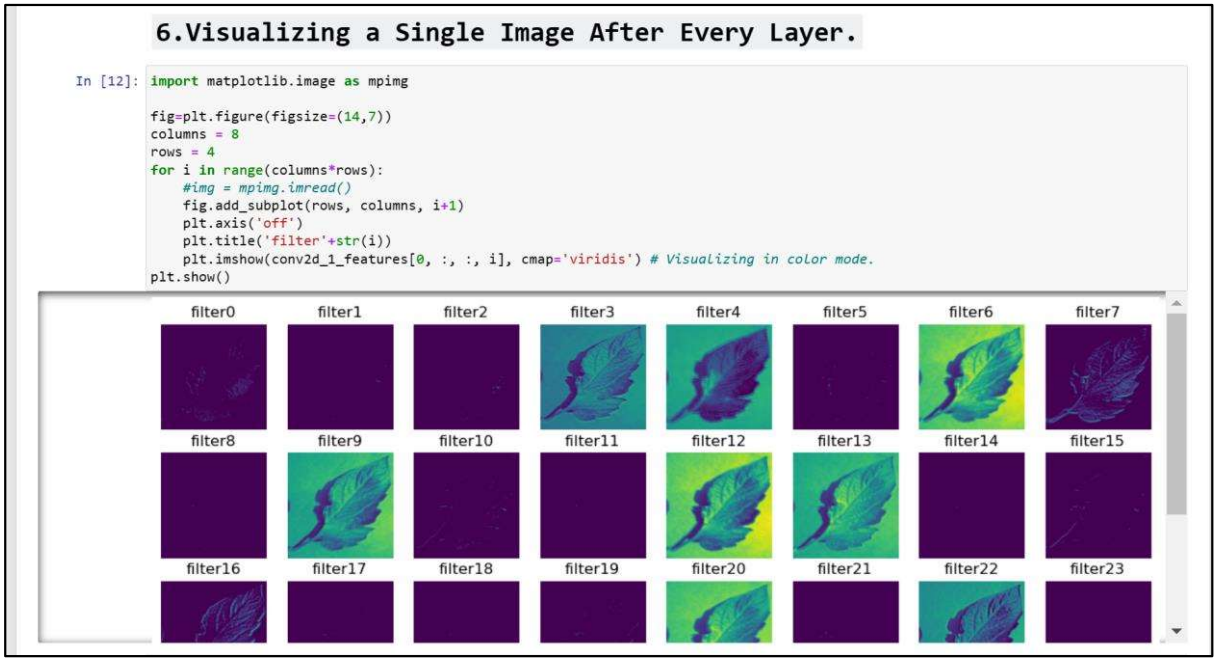
The first step is creating an instance of the Sequential class called model(variable), which is used for creating a linear stack of layers. Then, it adds a 2D convolutional layer to the model using the "Conv2D" layer. The layer has 32 filters with a kernel size of 5x5, the variable input_shape defines the input shape, and the activation function used is ReLU. Next, it adds a Maxpooling layer to the model using the "MaxPooling2D" layer. The layer has a pool size of (3, 3) which help to reduce the spatial dimensions of the data while keeping the essential features. Likewise, another two convolutional and Maxpooling layers are added.

After that, it adds a fully connected layer to the model using the "Dense" layer. The layer has 512 neurons and an activation function, ReLU. Then, it adds a Dropout layer to the model, which helps to prevent overfitting by randomly dropping some neurons during the training process. Another dense layer is added to the model with 128 neurons and activation function ReLU. Lastly, it adds the output layer to the model using the "Dense" layer with "num_classes" neurons and activation function "softmax".

The "model.summary()" function displays a summary of the model's architecture, including the number of parameters for each layer in the model. It gives a quick overview of the model's architecture and the number of parameters that need to be trained, which can help understand the model's complexity.

This architecture is a standard structure for CNN models, inspired by the LeNet architecture. It includes a series of convolutional layers to extract features from the images, followed by a series of fully connected layers to classify the images based on the extracted features. The architecture can be changed and fine-tuned based on the specific problem and dataset.

STEP 6



In this step, the code selects a specific image from the test dataset and visualizes what happens to that, in each layer explained in the above step.

STEP 7

7.Training The Model.

In [11]:
```
# validation data.
validation_generator = train_datagen.flow_from_directory(
    train_dir, # same directory as training data
    target_size=(img_height, img_width),
    batch_size=batch_size)
```

```
Found 34068 images belonging to 5 classes.
```

In [12]:
```
# Model building to get trained with parameters.
import tensorflow as tf

opt=tf.keras.optimizers.Adam(lr=0.001)
model.compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
train=model.fit_generator(train_generator,
                          epochs=10,
                          steps_per_epoch=train_generator.samples // batch_size,
                          validation_data=validation_generator,
                          validation_steps= validation_generator.samples// batch_size,verbose=1)
```

```
Epoch 1/10
1064/1064 [==============================] - 665s 621ms/step - loss: 0.2569 - accuracy: 0.9098 - val_loss: 0.0799 - val_accurac
y: 0.9754
Epoch 2/10
1064/1064 [==============================] - 658s 618ms/step - loss: 0.0857 - accuracy: 0.9730 - val_loss: 0.0723 - val_accurac
y: 0.9767
Epoch 3/10
1064/1064 [==============================] - 678s 637ms/step - loss: 0.0628 - accuracy: 0.9809 - val_loss: 0.0470 - val_accurac
y: 0.9847
Epoch 4/10
1064/1064 [==============================] - 665s 625ms/step - loss: 0.0536 - accuracy: 0.9845 - val_loss: 0.0220 - val_accurac
y: 0.9936
Epoch 5/10
1064/1064 [==============================] - 663s 624ms/step - loss: 0.0399 - accuracy: 0.9884 - val_loss: 0.0458 - val_accurac
y: 0.9848
Epoch 6/10
1064/1064 [==============================] - 652s 613ms/step - loss: 0.0385 - accuracy: 0.9885 - val_loss: 0.0261 - val_accurac
y: 0.9920
Epoch 7/10
1064/1064 [==============================] - 661s 621ms/step - loss: 0.0243 - accuracy: 0.9929 - val_loss: 0.0265 - val_accurac
y: 0.9925
Epoch 8/10
1064/1064 [==============================] - 663s 624ms/step - loss: 0.0322 - accuracy: 0.9907 - val_loss: 0.0306 - val_accurac
y: 0.9910
Epoch 9/10
1064/1064 [==============================] - 673s 632ms/step - loss: 0.0328 - accuracy: 0.9910 - val_loss: 0.0473 - val_accurac
y: 0.9895
Epoch 10/10
1064/1064 [==============================] - 671s 631ms/step - loss: 0.0252 - accuracy: 0.9935 - val_loss: 0.0125 - val_accurac
y: 0.9963
```

This code uses the data generators and the previously defined CNN model to train the model with the specified parameters.

The first step is importing the TensorFlow library. Then, it creates an instance of the Adam optimizer, which adapts the learning rate for each parameter with a rate of 0.001. Next, it compiles the model by specifying the optimizer, loss function, and metrics.

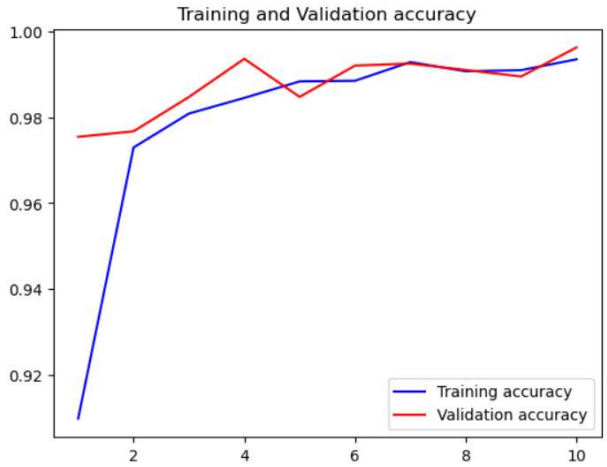
Lastly, it trains the model using the fit_generator() method, which uses the data generators defined earlier for training and validation data. The training process runs for 10 epochs. Further, the number of batches per training cycle is calculated by dividing the number of samples by the batch size, and the same is done for validation data. The training process prints out the progress to the user.

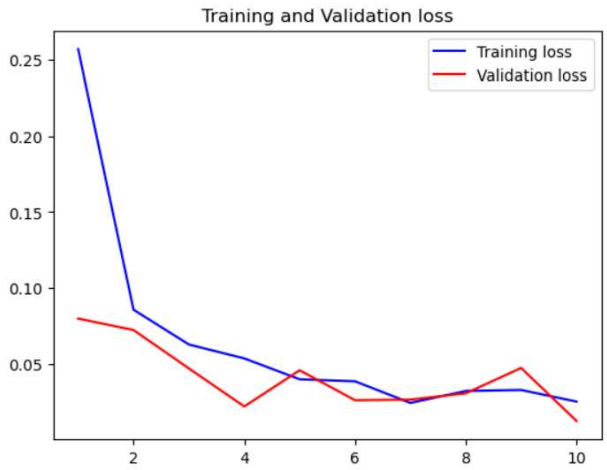
The model's parameters are updated using the Adam optimizer after each batch of training data. The loss and accuracy are calculated using the specified loss function and metrics, respectively, and the model's performance is evaluated using the validation data.

STEP 8

8.Plot For Accuracy And Losses

In [13]:
```
acc = train.history['accuracy']
val_acc = train.history['val_accuracy']
loss = train.history['loss']
val_loss = train.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```





This code visually represents the model's accuracy and loss during training and validation. It gets the accuracy and loss data from the training history, creates an array of the number of epochs, and plots the training and validation data on separate graphs for accuracy and loss. The training data is represented by the blue line and the validation data by the red line. The goal is to see the accuracy increasing over time and the loss decreasing over time, and also the validation data should be similar to the training data. If the validation data is lower than the training data, this suggests the model is overfitting, performing well on training data but not on unseen data.

STEP 9

9.Evaluating the model using Test data

```
In [14]: # from tensorflow import keras
         # model = keras.models.load_model('Models_h5/model2.h5')
         score,accuracy =model.evaluate(test_generator,verbose=1)
         print("Test score is {}".format(score))
         print("Test accuracy is {}".format(accuracy))

         175/175 [==============================] - 21s 121ms/step - loss: 0.0951 - accuracy: 0.9840
         Test score is 0.09506388008594513
         Test accuracy is 0.984035849571228
```

This step evaluates the model by using the test dataset. Here, the model has got a 0.9840 accuracy.

STEP 10

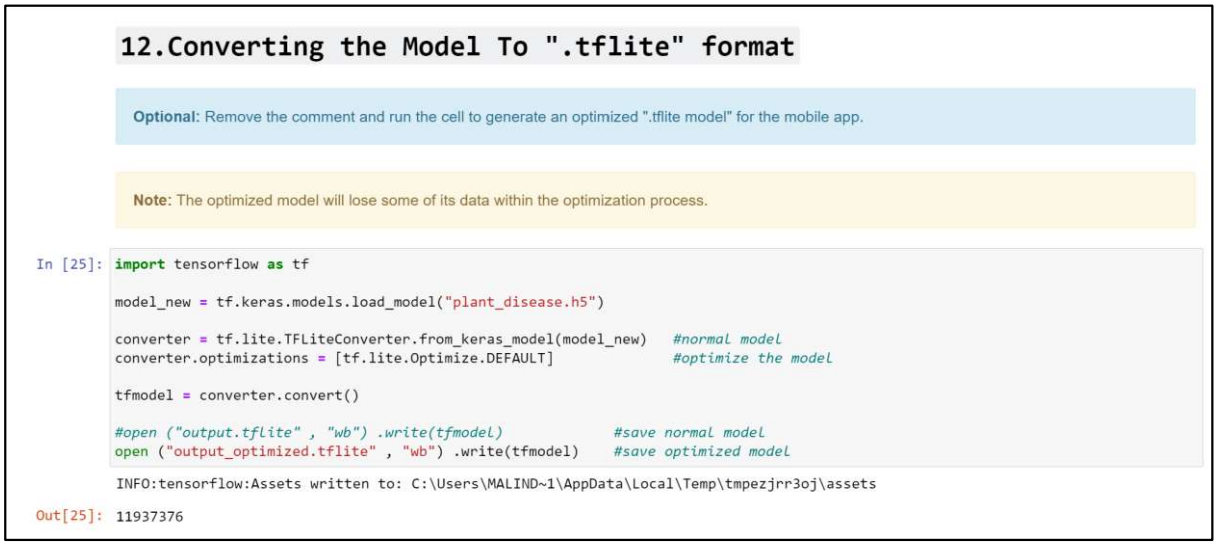
10.Saving the Model as a ".h5" file

```
In [15]: # Save entire model with optimizer, architecture, weights and training configuration.
         from keras.models import load_model
         model.save('Models_h5/modelAll.h5')

In [16]: # Save model weights.
         from keras.models import load_model
         model.save_weights('Models_Weights/modelAll_weights.h5')

In [17]: # Get classes of model trained on
         classes = train_generator.class_indices
         classes

Out[17]: {'Cinnamon': 0, 'Coffee': 1, 'Rice': 2, 'Tea': 3, 'Tomato': 4}
```

Finally, the trained model and the model weights are saved as a .h5 file.

7.4.1.4 Convert The Model into A Mobile-Friendly Version

Converting a model from the .h5 format to the .tflite(TensorFlow Lite) format is done to optimize the model for deployment on resource-constrained devices such as mobile devices. The .h5 format is mainly used to save the model architecture, weights, and optimizer state, while the .tflite format is designed to be smaller in size and more efficient in terms of memory and computational resources. This makes it more appropriate for deployment on mobile devices, which often have limited resources compared to desktop or cloud-based systems.



12.Converting the Model To ".tflite" format

Optional: Remove the comment and run the cell to generate an optimized ".tflite model" for the mobile app.

Note: The optimized model will lose some of its data within the optimization process.

In [25]:
```
import tensorflow as tf

model_new = tf.keras.models.load_model("plant_disease.h5")

converter = tf.lite.TFLiteConverter.from_keras_model(model_new)   #normal model
converter.optimizations = [tf.lite.Optimize.DEFAULT]              #optimize the model

tfmodel = converter.convert()

#open ("output.tflite" , "wb") .write(tfmodel)             #save normal model
open ("output_optimized.tflite" , "wb") .write(tfmodel)    #save optimized model
```

```
INFO:tensorflow:Assets written to: C:\Users\MALIND~1\AppData\Local\Temp\tmpezjrr3oj\assets
```

Out[25]:
```
11937376
```

As this project uses six models, the .tflit files have been further reduced by their sizes using model optimization.

The Plant Care Mobile Application's six models are as follows,

7.4.1.4.1 Validation Model

This model works as the first layer of the disease detection process. It has five classes, one separate class for each crop. It identifies the input image crop and outputs the results.

Classes: Cinnamon, Coffee, Rice, tea, Tea and Tomato

7.4.1.4.2 Cinnamon Diseases Model

The model identifies **Cinnamon** diseases in a cinnamon plant.

Classes: Cinnamon Rough Bark, Cinnamon Stripe Canker and Cinnamon Healthy.

7.4.1.4.3 Rice Diseases Model

The model identifies **Rice** diseases in a **Rice** plant.

Classes: Rice Bacterial Blight, Rice Blast, Rice Brown Spot, Rice Tungro, Rice Healthy

7.4.1.4.4 Coffee Diseases Model

The model identifies **Coffee** diseases in a **Coffee** plant.

Classes: Coffee Cerscospora, Coffee Phoma, Coffee Leaf Rust and Coffee Healthy.

7.4.1.4.5 Tomato Diseases Model

The model identifies **Tomato** diseases in a **Tomato** plant.

Classes: Tomato Late Blight, Tomato Leaf Mold, Tomato Septoria Leaf Spot,

7.4.1.4.6 Tea Diseases Model

The model identifies **Tea** diseases in a **Tea** plant.

Classes: Tea Algal Leaf, Tea Anthracnose, Tea Bird Eye Spot, Tea Healthy

Ex: Cinnamon Rough Bark crop is given to the DL model for identification.

The first (Validation) model identifies the type of crop as Cinnamon. Then the app will redirect the image to the cinnamon model. Finally cinnamon model identifies the specific disease.

7.4.2 Flutter Project & Build

7.4.2.1 Main.dart

The `main()` function is a top-level function in Dart that initiates the execution of the program. the `main()` function can only be used once.


```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:hive_flutter/hive_flutter.dart';

import 'services/disease_provider.dart';
import 'src/welcome_screen.dart';
import 'src/user_guide_screen.dart';
import 'src/account_screen.dart';
import 'src/diagnosis_details.dart';
import 'src/home_page/models/disease_model.dart';
import 'src/home_page/home_screen.dart';

Future main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Hive.initFlutter();
  Hive.registerAdapter(DiseaseAdapter());
  await Hive.openBox<Disease>('plant_care');
  runApp(PlantCareApp());
}

class PlantCareApp extends StatefulWidget {
  PlantCareApp({Key? key}) : super(key: key);

  @override
  State<PlantCareApp> createState() => _PlantCareAppState();
}

class _PlantCareAppState extends State<PlantCareApp> {
  bool isLoggedIn = true;

  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider<DiseaseProvider>(
      create: (context) => DiseaseProvider(),
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Plant Care',
        darkTheme: ThemeData(
            useMaterial3: true,
            colorSchemeSeed: const Color(0xff26bf8d),
            brightness: Brightness.dark,
            fontFamily: 'Poppins-Regular'),
        theme: ThemeData(
            useMaterial3: true,
            colorSchemeSeed: const Color(0xff26bf8d),
            fontFamily: 'Poppins-Regular'),
        initialRoute: WelcomeScreen.id,
        routes: {
          WelcomeScreen.id: (context) => WelcomeScreen(),
          UserGuide.id: (context) => UserGuide(),
          HomeScreen.id: (context) => HomeScreen(),
          DiagnosisDetailsScreen.id: (context) => DiagnosisDetailsScreen(),
          Settings.id: (context) => Settings(),
        },
        onGenerateRoute: (RouteSettings routeSettings) {
          return MaterialPageRoute<void>(
            settings: routeSettings,
            builder: (BuildContext context) {
              switch (routeSettings.name) {
                case WelcomeScreen.routeName:
                  return WelcomeScreen();
                case UserGuide.routeName:
                  return UserGuide();
                case HomeScreen.routeName:
                  return HomeScreen();
                case DiagnosisDetailsScreen.routeName:
                  return DiagnosisDetailsScreen();
                case Settings.routeName:
                  return Settings();
                default:
                  return HomeScreen();
              }
            },
          );
        },
      ),
    );
  }
}
```

Section 1 – Imports all the necessary dependency and package required. Material is used as the application is developed for android and application follows material UI 3. Since the application uses Hive database to store user generated content locally, its imported here (at the highest level). The provider package provides an easy way to a wrapper around InheritedWidget to make them easier to use and more reusable.

Section 2 – contains the main code of the application and return `ChangeNotifierProvider<DiseaseProvider>` this declares the provider for the application.

Section 3 – contains the main android app `MaterialApp();` if app is developed for iOS then the `CupertinoApp();` can be called.

Section 4 – sets the app navigation and set the welcome screen as the `initialRoute`.

Functions & Features:

- Provider (Hive DB)
- Routes and navigation

7.4.2.2 Welcome screen & User guide screen.

```dart
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'user_guide_screen.dart';

class WelcomeScreen extends StatefulWidget {
  WelcomeScreen({Key? key}) : super(key: key);
  static String id = "Welcome";
  static const routeName = '/Welcome';

  @override
  State<WelcomeScreen> createState() => _WelcomeScreenState();
}

class _WelcomeScreenState extends State<WelcomeScreen> {
  Color gradientStart = Colors.transparent;
  Color gradientEnd = Color.fromARGB(200, 0, 0, 0);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: <Widget>[
          ShaderMask(
            shaderCallback: (rect) {
              return LinearGradient(
                begin: Alignment.topCenter,
                end: Alignment.bottomCenter,
                colors: [gradientStart, gradientEnd],
              ).createShader(
                  Rect.fromLTRB(0, 140, rect.width, rect.height - 20));
            },
            blendMode: BlendMode.darken,
            child: Container(
              decoration: BoxDecoration(
                image: DecorationImage(
                  image: AssetImage("assets/images/Welcome_BG_3.jpg"),
                  fit: BoxFit.cover,
                ),
              ),
            ),
          ),
          Column(
            children: [
              Expanded(
                child: Container(
                  child: Align(
                    alignment: Alignment.center,
                    child: Container(
                      width: 140,
                      height: 140,
                      decoration: BoxDecoration(
                        image: DecorationImage(
                          image: AssetImage("assets/images/App Logo.png"),
                          fit: BoxFit.cover,
                        ),
                      ),
                    ),
                  ),
                ),
                flex: 1,
              ),
              Expanded(
                child: Container(
                    margin: EdgeInsets.only(bottom: 10.0),
                    child: Text(
                      'Plant Care',
                      style: TextStyle(
                          fontSize: 30,
                          fontWeight: FontWeight.bold,
                          color: Colors.white),
                      textAlign: TextAlign.center,
                    )),
                flex: 0,
              ),
              Expanded(
                child: Container(
                  margin: EdgeInsets.only(bottom: 10.0),
                  child: Text(
                    'We diagnose plant diseases using Artificial intelligence & image recognition to help Sri Lankan farmers identify harmful pests and diseases, which will help protect the Sri Lankan Ecosystem.',
                    style: TextStyle(fontSize: 18.0, color: Colors.white),
                    textAlign: TextAlign.center,
                  ),
                  padding: EdgeInsets.symmetric(vertical: 10.0),
                  constraints: BoxConstraints(
                    maxWidth: 330.0,
                  ),
                ),
                flex: 0,
              ),
              Expanded(
                child: Padding(
                  padding: const EdgeInsets.all(20),
                  child: ButtonTheme(
                    minWidth: 320.0,
                    height: 50.0,
                    child: ElevatedButton(
                      onPressed: () {},
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.blue, // Background color
                        foregroundColor:
                            Colors.white, // Text Color (Foreground color)
                      ),
                      child: Container(
                        child: Text(
                          'Continue with Google',
                          style: TextStyle(
                            fontSize: 16,
                            fontWeight: FontWeight.bold,
                          ),
                          textAlign: TextAlign.center,
                        ),
                      ),
                    ),
                  ),
                ),
                flex: 0,
              ),
              Expanded(
                child: Container(
                  margin: EdgeInsets.only(bottom: 20.0),
                  child: ButtonTheme(
                    minWidth: 200.0,
                    height: 50.0,
                    child: TextButton(
                      onPressed: () {
                        HapticFeedback.selectionClick();
                        Navigator.pushReplacementNamed(context, UserGuide.id);
                      },
                      // textColor: Colors.white,
                      child: Container(
                        child: Text(
                          'SKIP',
                          style: TextStyle(
                            fontSize: 16,
                            fontWeight: FontWeight.bold,
                            color: Colors.white,
                          ),
                          textAlign: TextAlign.center,
                        ),
                      ),
                    ),
                  ),
                ),
                flex: 0,
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

Provides UI for the first screens when app is launched. Welcome screen will allow the user to login using Google account or continue without login. User guide screen provides a quick start guide for the user to use the app.

Section 1 – first layer of stack background image layer, followed by Gradient to make text more visible to user.

Section 2 – App logo from app assets.

Section 3 – App name

Section 4 – App description

Section 5 – Firebase auth service, for user to login with Google Account.

Section 6 – "Skip" button for user to use app without authenticating. (data store locally)

Functions & Features:

- User Authentication

Note:
User guide screen only contains UI and does not contain any major Functions or Features. Therefore, is not included in this documentation.

7.4.2.3 Home screen

Home screen of the app is the default focal point of navigation system. The home screen is a placeholder for Three tabs.

- Map
- Diagnosis Tool
- History

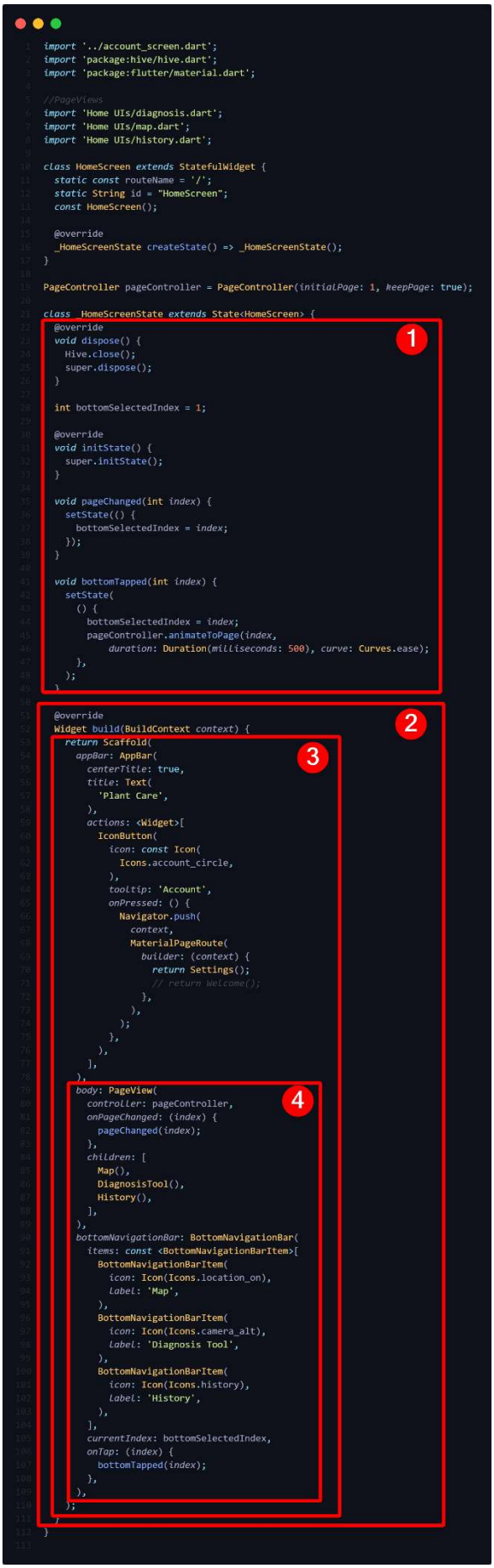
Section 1 - opens the hive DB and declares necessary variable to store navigation selection information.

Section 2 – is the main Return widget. Containing section 3 The Scaffold.

Section 4 - Bottom Navigation bar with sliding functionalities for ease of use.

Functions & Features:

- Main app navigation:
- Navigate to Map
- Navigate to Diagnosis Tool
- Navigate to History
- Navigate to Account Settings



7.4.2.3.1 Diagnosis Tool

This screen is the default tab of the home screen as it gives quick access to the core function of the app. The user can engage in the first step of using the diagnosis tool selecting desired crop type to diagnose. A pop up will then let the user select diagnosis method (take photo or select photo from gallery). This screen contains an advance 3d effect on the cards called Parallax to give the app aesthetic depth to the UI.

Code for Parallax effect (https://docs.flutter.dev/cookbook/effects/parallax-scrolling) project code not included in this document because there more than 400 lines of code. Please follow the link above to get more information.

List of cards for screen include the following =>

```dart
class Disease_List {
  const Disease_List({
    required this.name,
    required this.types,
    required this.imageUrl,
  });

  final String name;
  final String types;
  final String imageUrl;
}

const locations = [
  Disease_List(
    name: 'Cinnamon',
    types: '2 Diseases',
    imageUrl: 'assets/images/cinnamon.png',
  ),
  Disease_List(
    name: 'Coffee',
    types: '3 Diseases',
    imageUrl: 'assets/images/Coffee.jpg',
  ),
  Disease_List(
    name: 'Tea',
    types: '3 Diseases',
    imageUrl: 'assets/images/Tea.jpg',
  ),
  Disease_List(
    name: 'Tomato',
    types: '3 Diseases',
    imageUrl: 'assets/images/Tomatoes.jpg',
  ),
  Disease_List(
    name: 'Rice',
    types: '4 Diseases',
    imageUrl: 'assets/images/Rice.jpg',
  ),
];
```

Cards: a for loop was used to build a scrollable list of all the diseases.

```
Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(
        children: [
          for (final location in locations)
            DiseaseListItem(
              imageUrl: location.imageUrl,
              name: location.name,
              country: location.types,
            ),
        ],
      ),
    );
  }
```

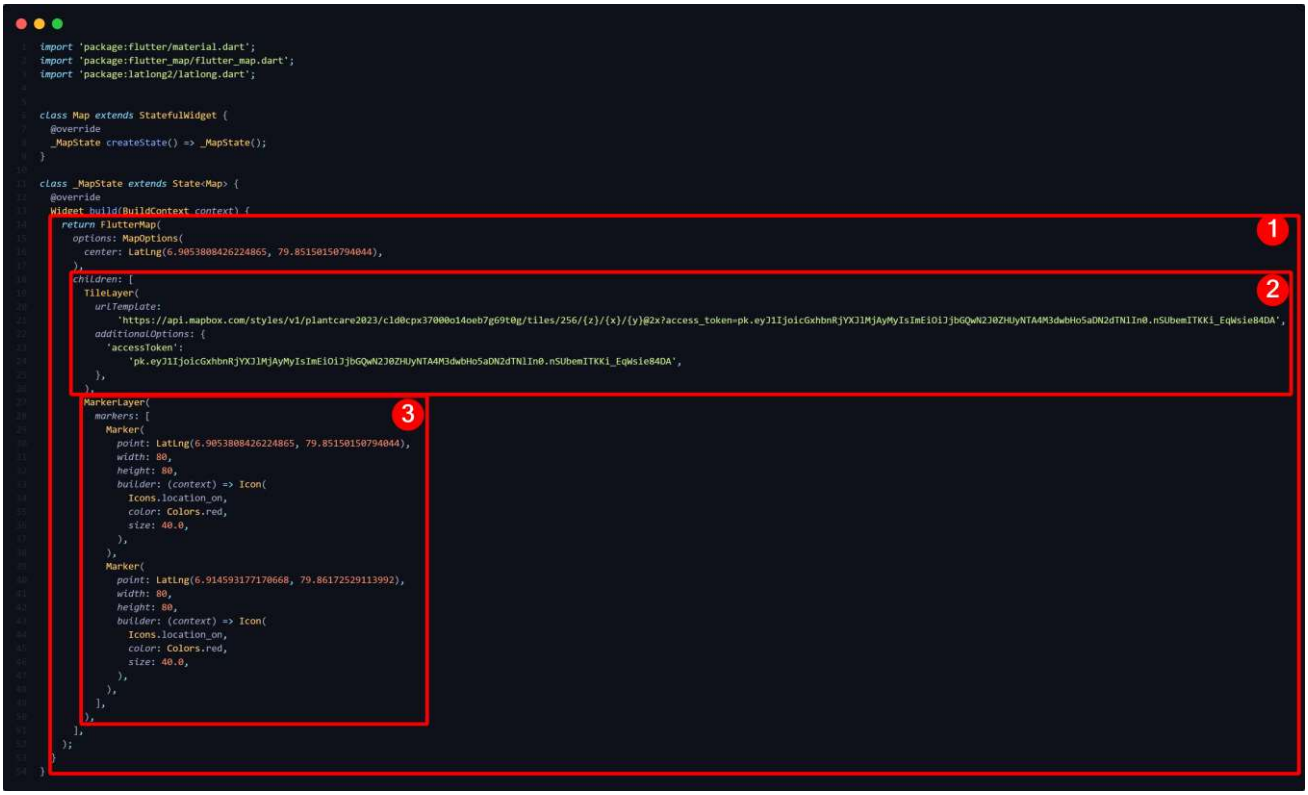
A Gesture detector was used to react to user selection. `OnTap()` opened a dialog box.

```
ElevatedButton.icon(
  onPressed: () async {
    late double _co    await classifier
        .getDisease(ImageSource.camera, widget.name)
        .then(
      (value) {
        _disease = Disease(
            name: value![0]["label"],
            imagePath: classifier.imageFile.path);
        _confidence = value[0]['confidence'];
        print(value);
      },
    );
    // Check confidence
    if (_confidence > 0.8) {
      // Set disease for Disease Service
      _diseaseService.setDiseaseValue(_disease);
      // Save disease
      _hiveService.addDisease(_disease);
      Navigator.restorablePushNamed(
        context,
        DiagnosisDetailsScreen.routeName,
      );
    } else {
      Navigator.pop(context);
    }
  },
  icon: Icon(Icons.camera_alt_rounded),
  label: const Text('Take a Photo'),
),
```

Functions & Features:

- Select crop.
- Pass crop type to image classifier.
- DialogBox

7.4.2.3.1 Map



Map show the user the locations where disease have been identified.

Section 1 – Uses the flutter_map package to render a map onto the screen.

Section 2 - Integration of the Mapbox API and map style

Section 3 – list of makers on the map

Functions & Features:

- Display all location where disease have been identified form data reived from firebase and hive.
- Map navigation

7.4.2.3.2 History

The history screen will display all passed diagnosis with accuracy over 80%

Functions & Features:

- Stream Builder for hive list of objects
- On press open diagnosis details screen

```dart
import 'dart:io';
import '/services/hive_database.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '/services/disease_provider.dart';
import 'package:hive_flutter/hive_flutter.dart';
import '/src/home_page/models/disease_model.dart';
import '../../diagnosis_details.dart';

class History extends StatefulWidget {
  @override
  _HistoryState createState() => _HistoryState();
}

class _HistoryState extends State<History> {
  @override
  Widget build(BuildContext context) {
    return ValueListenableBuilder<Box<Disease>>(
      valueListenable: Boxes.getDiseases().listenable(),
      builder: (context, box, _) {
        final diseases = box.values.toList().cast<Disease>();
        if (diseases.isNotEmpty) {
          return ListView.builder(
            itemCount: diseases.length,
            itemBuilder: (context, index) {
              return _returnHistoryContainer(
                diseases[index],
                context,
                Provider.of<DiseaseProvider>(context),
              );
            },
          );
        } else {
          return _returnNothingToShow();
        }
      },
    );
  }
}

Widget _returnHistoryContainer(
  Disease disease,
  BuildContext context,
  DiseaseProvider diseaseService,
) {
  return Padding(
    padding: const EdgeInsets.symmetric(horizontal: 25.0, vertical: 5.0),
    child: Card(
      elevation: 2.5,
      clipBehavior: Clip.antiAlias,
      child: GestureDetector(
        onTap: () {
          diseaseService.setDiseaseValue(disease);
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) {
                return DiagnosisDetailsScreen();
              },
            ),
          );
        },
        child: Column(
          children: [
            Container(
              width: MediaQuery.of(context).size.width,
              height: 200,
              child: Image(
                fit: BoxFit.fitWidth,
                image: FileImage(
                  File(disease.imagePath),
                ),
              ),
            ),
            ListTile(
              title: Text('Disease: ${disease.name}'),
              subtitle: Text(
                'Date: ${disease.dateTime.day}/${disease.dateTime.month}/${disease.dateTime.year}',
              ),
            ),
            TextButton(
              onPressed: () {
                disease.delete();
              },
              child: const Text('Delete'),
            ),
          ],
        ),
      ),
    ),
  );
}

Widget _returnNothingToShow() {
  return Container(
    child: const Center(
      child: Text(
        'Nothing to show, history is empty',
        style: TextStyle(fontSize: 20),
      ),
    ),
  );
}
```

7.4.2.4 Diagnosis details screen

Diagnosis details screen displays the results after model diagnoses. User can view the result and get a description of Diagnosis including a source of the information.

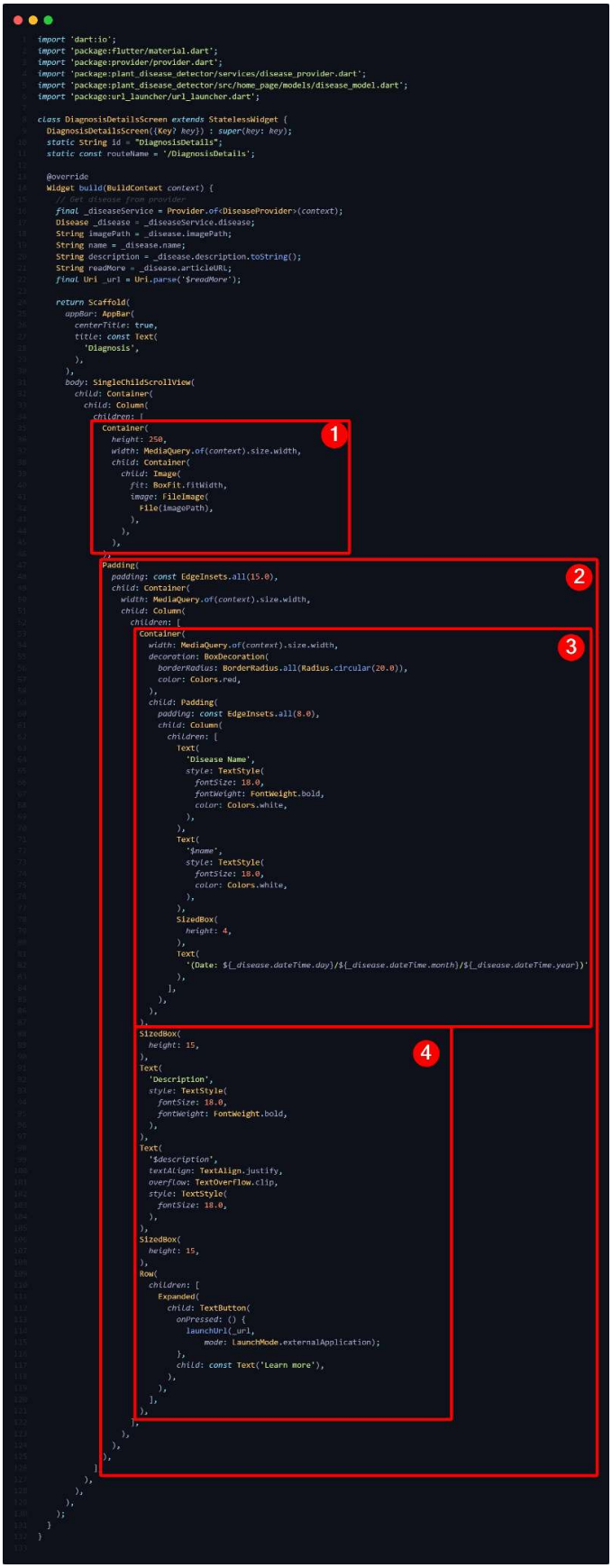
- Map
- Diagnosis Tool
- History

Section 1 – Gets the relevant image for the selected query from the provider. Image is set to fix the space.

Section 2 – is the main body of content Section 3 contains a quick view while section 4 contains description.

Functions & Features:

- Display result of Diagnosis:
- Use `url_launcher` to open in browser.



7.4.2.5 Account Settings

The account screen allows the user to view his/her account, additionally this screen allows the user to get help to use the app, delete account, get app info and logout of account if logged in. Additionally, this screen can get the users current location. location data is then used to identify where a disease was detected.

Functions & Features:

- Logout from Account
- Delete Account
- Get user location

GetLocation() =>

```dart
Future<bool> _handleLocationPermission() async {
  bool serviceEnabled;
  LocationPermission permission;

  serviceEnabled = await Geolocator.isLocationServiceEnabled();
  if (!serviceEnabled) {
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text(
            'Location services are disabled. Please enable the services')));
    return false;
  }
  permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
    permission = await Geolocator.requestPermission();
    if (permission == LocationPermission.denied) {
      ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Location permissions are denied')));
      return false;
    }
  }
  if (permission == LocationPermission.deniedForever) {
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text(
            'Location permissions are permanently denied, we cannot request permissions.')));
    return false;
  }
  return true;
}

Future<void> _getCurrentPosition() async {
  final hasPermission = await _handleLocationPermission();

  if (!hasPermission) return;
  await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high)
      .then((Position position) {
    setState(() => _currentPosition = position);
    _getAddressFromLatLng(_currentPosition!);
  }).catchError((e) {
    debugPrint(e);
  });
}

Future<void> _getAddressFromLatLng(Position position) async {
  await placemarkFromCoordinates(
          _currentPosition!.latitude, _currentPosition!.longitude)
      .then(
    (List<Placemark> placemarks) {
      Placemark place = placemarks[0];
      setState(
        () {
          _currentAddress =
              '${place.street}, ${place.subLocality}, ${place.country}';
        },
      );
    },
  ).catchError(
    (e) {
      debugPrint(e);
    },
  );
}
```

7.4.2.6 Classify.dart

Classify class is used to classify image provided and provide a result.

Line 11: specifies the source of the image, options include the camera or device file directory.

Line 14: `if` condition check if image path is valid. Exception handling.

Line 16: `switch` statement loads the relevant model for user selected crop type.

Functions & Features:

- Classify image based on provide DL model.
- Check if accuracy is over 80%

```dart
import 'dart:io';
import 'package:image_picker/image_picker.dart';
import 'package:tflite/tflite.dart';
import 'dart:developer';

class Classifier {
  late File imageFile;
  late List outputs;

  Future<List?> getDisease(ImageSource imageSource, String plantType) async {
    var image = await ImagePicker().pickImage(source: imageSource);
    var imagePath = image?.path;

    if (imagePath != null) {
      imageFile = File(imagePath);
      switch (plantType) {
        case 'Cinnamon':
          log('plantType: $plantType');
          await Tflite.loadModel(
            model: "assets/model/model_cinnamon.tflite",
            labels: "assets/model/labels_cinnamon.txt",
            numThreads: 1,
          );
          break;
        case 'Coffee':
          log('plantType: $plantType');
          await Tflite.loadModel(
            model: "assets/model/model_coffee.tflite",
            labels: "assets/model/labels_coffee.txt",
            numThreads: 1,
          );
          break;
        case 'Rice':
          log('plantType: $plantType');
          await Tflite.loadModel(
            model: "assets/model/model_rice.tflite",
            labels: "assets/model/labels_rice.txt",
            numThreads: 1,
          );
          break;
        case 'Tea':
          log('plantType: $plantType');
          await Tflite.loadModel(
            model: "assets/model/model_tea.tflite",
            labels: "assets/model/labels_tea.txt",
            numThreads: 1,
          );
          break;
        case 'Tomato':
          log('plantType: $plantType');
          await Tflite.loadModel(
            model: "assets/model/model_coffee.tflite",
            labels: "assets/model/labels_coffee.txt",
            numThreads: 1,
          );
          break;

        default:
          log('$plantType');
          break;
      }
      var result = await classifyImage(imageFile);
      Tflite.close();
      log('Tflite.close');
      return result;
    } else {
      log('image is null');
      return [null];
    }
  }

  Future<List?> classifyImage(File image) async {
    log('classifyImage');
    var output = await Tflite.runModelOnImage(
        path: image.path,
        imageMean: 0.0, // defaults to 117.0
        imageStd: 255.0, // defaults to 1.0
        numResults: 2, // defaults to 5
        threshold: 0.2, // defaults to 0.1
        asynch: true);
    return output;
  }
}
```

7.5 Chapter Summary

This chapter covered all the aspects when it comes to implementation of the project. Starting from the front-end, the user interface (UI) was built with Flutter combining few of the powerful features and tools available in Flutter. Together with Hive which is a fast key-value database written in pure dart which was used to store local data from the diagnosis functionality. For training the machine learning models with deep learning CNN algorithms using TensorFlow which is a free and open-source software library for machine learning and AI. By gathering and combining multiple data sets which contains more than 20000 images were used to train the models. For the cloud-based applications, firebase was utilized, which are for the firebase authenticator which was used for the account authentication functionalities and firebase store for cloud storage, storage utilities used for diagnosis data under account as such. Mapbox was used as the map API utilizing services such as the atlas and pin module. All implementation aspects such as online processes involved and offline processes used to build up the system were documented in this chapter. All problems and challenges faced while implementing the product was documented in this chapter.

CHAPTER 7: TESTING

8.1 Chapter Overview

This chapter describes the testing performed on the system to ensure it is functioning properly and meets specified requirements. The testing includes both functional and non-functional testing methods, using black box techniques. The test plan, including the objectives and criteria followed, is outlined. Additionally, unit testing and integration testing done on the codebase are documented. The chapter concludes with the presentation of benchmarking result.

8.2 Goals and Objectives of Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. The goals and objectives of testing can include:

- To identify defects or bugs in the system.
- To ensure that the system meets the specified requirements.
- To evaluate the system's compliance with industry standards and regulations.
- To measure the system's performance and scalability.
- To improve the overall quality of the system.
- To provide information to stakeholders on the system's readiness for release.
- To support the maintenance and evolution of the system.
- To detect any security vulnerabilities in the system.

The main objective of testing is to identify any defects or issues in the system as early as possible in the development process, so that they can be fixed before the system is released. This can help to reduce the overall time of development and increase the chances of delivering a high-quality product.

8.3 Functional Testing

The test plan of the black box testing and its results are as follows,

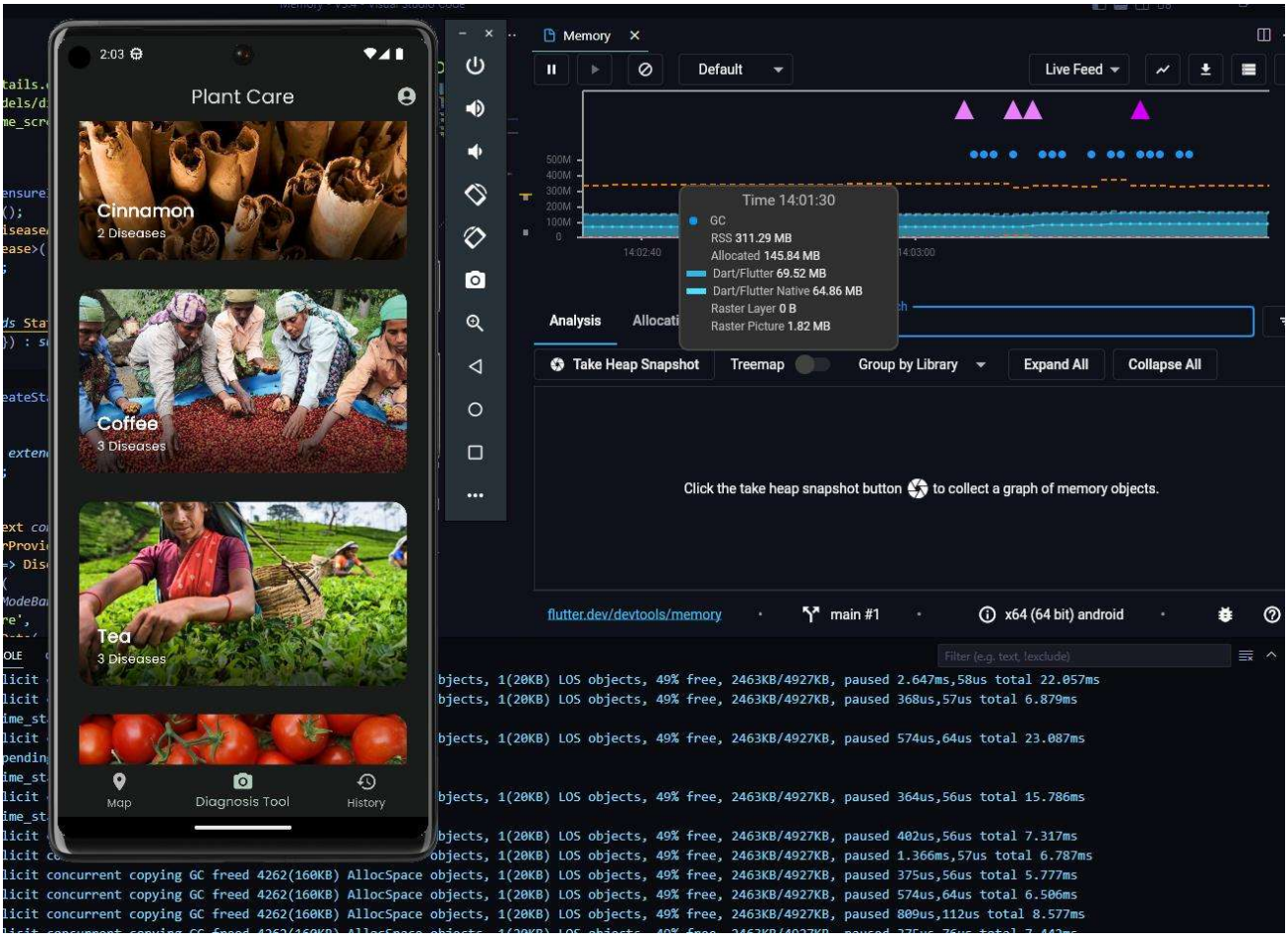
Testcase#	Description	Input Data /User action	Expectedoutcome	Actual outcome	Status
Testing base workflows					
1	Login with Google Account	Clicks Continue with Google Account button	User is directed to the Firebase auth service Google account	User is directed to the Firebase auth service Google account	Pass
2	Pressing skip button should direct user to User guide page without authenticating	Clicks Skip button	User is directed to the User guide page	User is directed to the User guide page	Pass
3	Pressing continue button will direct user to home page	Clicks Continue button in User guide page	User is directed to home page	User is directed to home page	Pass
4	Pressing on Cinnamon widget button	Clicks the Cinnamon widget button	User is made a prompt to select the mode of uploading the image	User is made a prompt to select the mode of uploading the image	Pass
5	Pressing on Coffee widget button	Clicks theCoffee widget button	User is made a prompt to select the mode of uploading the image	User is made a prompt to select the mode of uploading the image	Pass
6	Pressing on Tea widget button	Clicks theTea widget button	User is made a prompt to select the mode of uploading the image	User is made a prompt to select the mode of uploading the image	Pass
7	Pressing on Rice widget button	Clicks theRice widget button	User is made a prompt to select the mode of uploading the image	User is made a prompt to select the mode of uploading the image	Pass
8	Pressing on Tomato widget button	Clicks theTomato widget button	User is made a prompt to select the mode of uploading the image	User is made a prompt to select the mode of uploading the image	Pass
9	Selecting Take a Photo from prompted options	Clicks the Take a Photo from prompted options	User is redirected to the camera application	User is redirected to the camera application	Pass
10	Selecting Choose a Photo from prompted options	Clicks the Choose a Photo from prompted options	User redirected to the image picker in the files	User redirected to the image picker in the files	Pass
11	User selects the image from imager picker	Selects the image from imager picker	User is redirected to the diagnosis results screen and displayed with the disease name, date and description	User is redirected to the diagnosis results screen and displayed with the disease name, date and description	Pass

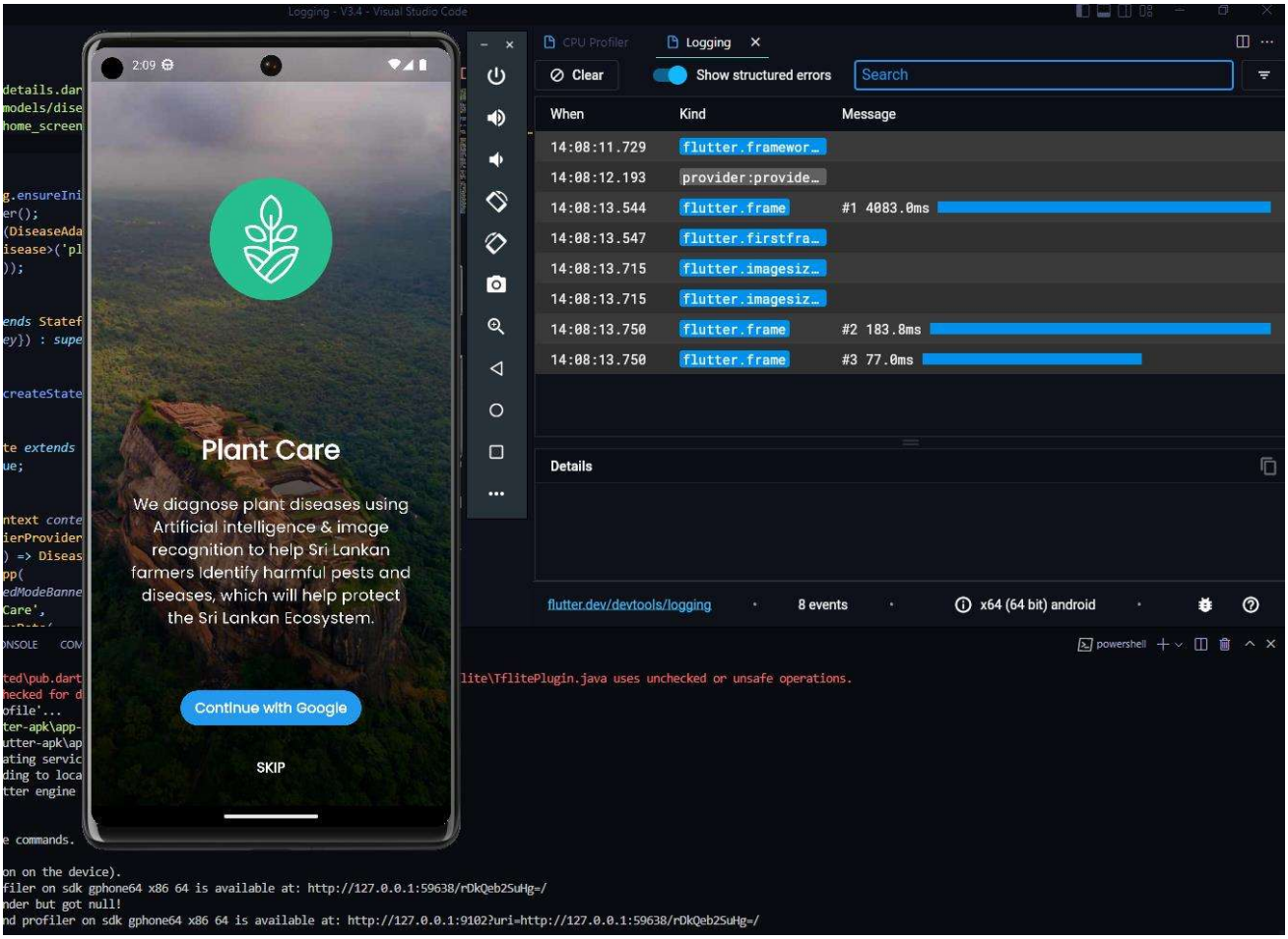
12	Pressing on Learn more in the results screen	Clicks the Learn more button in the results screen	Web browser is opened redirecting to the result source or information	Web browser is opened redirecting to the result source or information	Pass
13	User selects a historical result from History screen	Clicks a historical result from History screen	User redirected to specific result displaying disease name, date and description	User redirected to specific result displaying disease name, date and description	Pass
14	Pressing Delete button in History screen	Clicking Delete button in History screen	The historical record is deleted	The historical record is deleted	Pass
15	Selects the Map option	Clicks the Map option	The Mapbox API loads the navigable map with pins	The Mapbox API loads the navigable map with pins	Pass
16	Selects the Account Settings	Clicks the Account Settings	User is displayed the Account settings	User is displayed the Account settings	Pass
17	Selecting Help button in Account Settings	Clicking Help button in Account Settings	Prompted with Help prompt	Prompted with Help prompt	Pass
18	Selecting Delete Account button	Clicking Delete Account button	User account is deleted	User account is deleted	Pass
19	Selecting the App info button	Clicking the App info button	User is prompted with the version information	User is prompted with the version information	Pass

Table: Functional Testing Plan

Functional test pass rate = Number of tests passed/ Number of tests performed = 19/19 = Pass

8.4 Non-Functional Testing




Logging - V3.4 - Visual Studio Code
2:09
Plant Care
We diagnose plant diseases using Artificial intelligence & image recognition to help Sri Lankan farmers identify harmful pests and diseases, which will help protect the Sri Lankan Ecosystem.
Continue with Google
SKIP
CPU Profiler
Logging
Clear
Show structured errors
Search
When
Kind
Message
14:08:11.729 flutter.framewor…
14:08:12.193 provider:provide…
14:08:13.544 flutter.frame #1 4083.0ms
14:08:13.547 flutter.firstfra…
14:08:13.715 flutter.imagesiz…
14:08:13.715 flutter.imagesiz…
14:08:13.750 flutter.frame #2 183.8ms
14:08:13.750 flutter.frame #3 77.0ms
Details
flutter.dev/devtools/logging
8 events
x64 (64 bit) android
powershell
lite\TflitePlugin.java uses unchecked or unsafe operations.
on the device).
filer on sdk gphone64 x86 64 is available at: http://127.0.0.1:59638/rDkQeb2SuHg=/
nder but got null!
nd profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9102?uri=http://127.0.0.1:59638/rDkQeb2SuHg=/

8.5 Model Testing

Deep Learning Models Testing

In order to determine a deep learning model's accuracy in classifying, predicting, or creating new data, performance on data that has never been seen before is used to test the model. Typically, this stage comes after the model's training process but before it is used in production. By contrasting the model's predictions with the real values of the test data, testing aims to determine how accurate the model is. The effectiveness of the model can be assessed in this manner using a variety of measures, including accuracy, precision, recall, f1 score, and confusion matrix. Before the model is implemented, this might assist in finding potential improvement areas.

A model's accuracy is the percentage of right predictions it makes from all the ones produced. Although it is a straightforward and often used statistic, an unbalanced dataset might cause it to be inaccurate. A model's precision is, from all correct predictions made, the percentage of genuine positive predictions produced. It evaluates the model's ability to correctly identify positive instances among all positive predictions. The recall is the percentage of correct predictions a model produced from all real positive events. It assesses the model's capacity to find every occurrence of positivity. The f1-score is the harmonic mean of recall and accuracy. This testing method often applies to unbalanced datasets and balances accuracy and recall (Korstanje, 2021).

The confusion matrix is a table that illustrates how well a classification method performs. For every class, it lists the proportion of accurate forecasts and inaccurate guesses. Values on the crosswise indicate the times when the forecasted class matches the actual class, while those on the diagonal reflect instances where the classifier made an incorrect labelling decision (Narkhede, 2018). The f1-score, recall, and other performance indicators are computed using this confusion matrix.

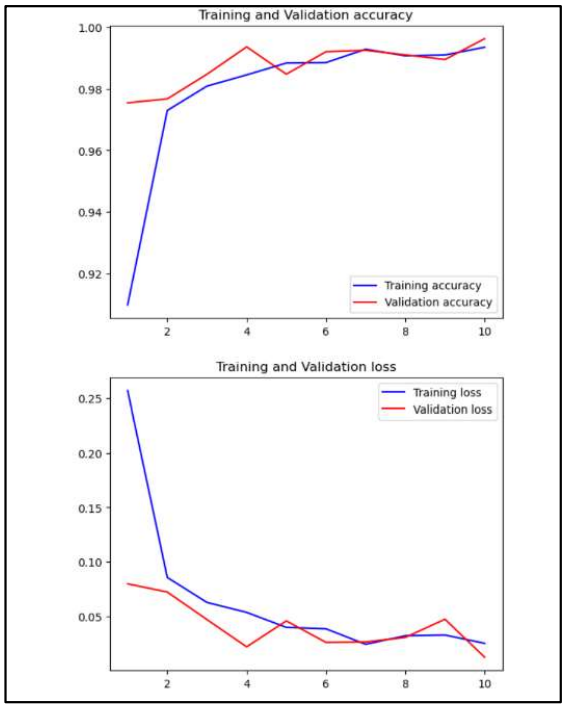
The primary purpose of the accuracy vs epochs graph is to show the model's performance during the training time duration. It demonstrates how the model's accuracy rises and lowers with the number of training iterations. This graph can spot trends like overfitting or underfitting and determine how many epochs are best for a particular model.

For each model, accuracy vs epochs graph, model evaluation, confusion matrix, precision, recall and f1 score indicators were generated, and the screen grabs are added below.

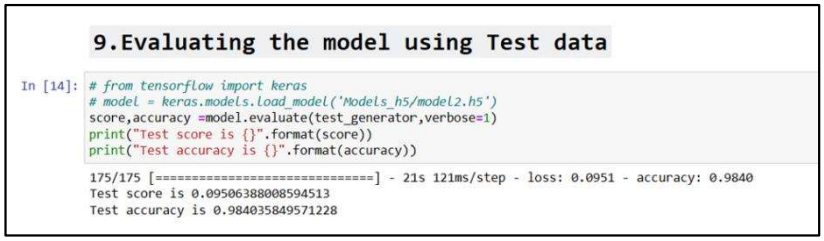
1st model – Crop Validation Model

(Cinnamon, Coffee, Rice, Tea and Tomato**)**

Accuracy vs epochs graph



Model Accuracy

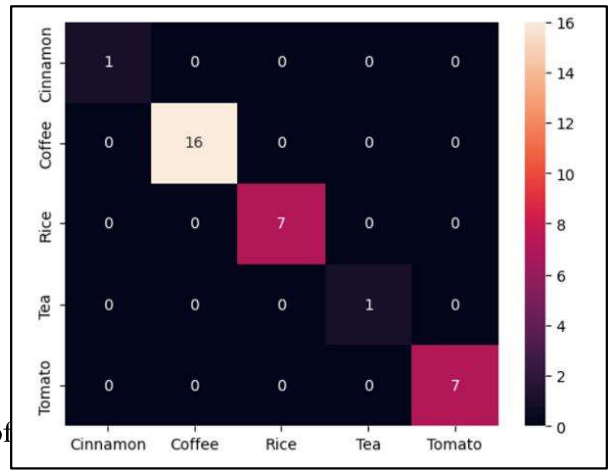


9.Evaluating the model using Test data

```
In [14]: # from tensorflow import keras
         # model = keras.models.load_model('Models_h5/model2.h5')
         score,accuracy =model.evaluate(test_generator,verbose=1)
         print("Test score is {}".format(score))
         print("Test accuracy is {}".format(accuracy))

         175/175 [==============================] - 21s 121ms/step - loss: 0.0951 - accuracy: 0.9840
         Test score is 0.09506388008594513
         Test accuracy is 0.984035849571228
```

Confusion Matrix

```
In [12]: x_test, y_test = next(train_generator)

In [13]: import sklearn
         from sklearn.metrics import confusion_matrix
         import numpy as np

         model = tf.keras.models.load_model("Models_h5/modelAll.h5")

         # Generate predictions for the test set
         score, y_pred = model.evaluate_generator(test_generator)

         y_test = np.argmax(y_test, axis=1)
         y_pred = np.argmax(model.predict(x_test), axis=1)


         # Compute the confusion matrix
         cm = confusion_matrix(y_test, y_pred)

         # Print the confusion matrix
         print(cm)

         [[ 1  0  0  0  0]
          [ 0 16  0  0  0]
          [ 0  0  7  0  0]
          [ 0  0  0  1  0]
          [ 0  0  0  0  7]]

In [14]: #Get the class names
         class_names = test_generator.class_indices

In [15]: import seaborn as sns
         annot_kws = {"ha": 'center',"va": 'center'}
         sns.heatmap(cm, annot=True, annot_kws=annot_kws, xticklabels=class_names, yticklabels=class_names, fmt='g')
Out[15]: <AxesSubplot: >
```



Classification Report – Precision | Recall | f1-score

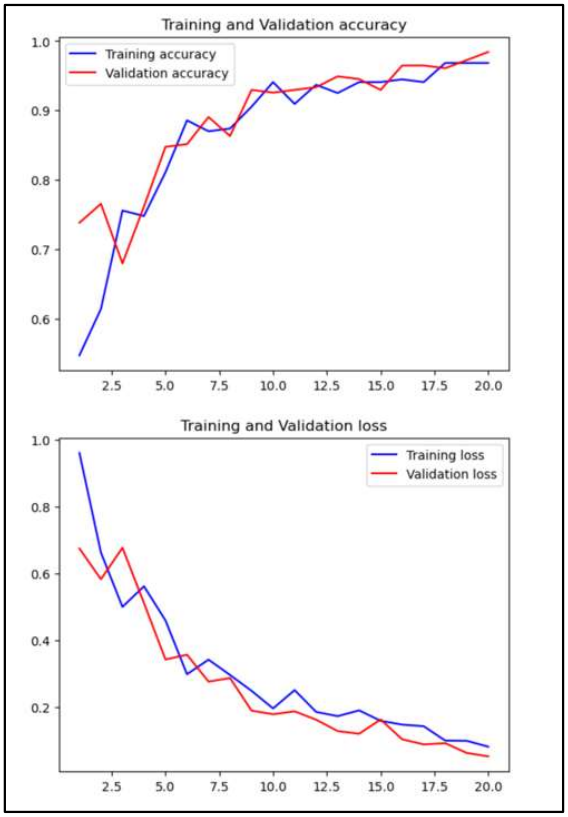
```
In [17]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred, target_names=class_names))
```

	precision	recall	f1-score	support
Cinnamon	1.00	1.00	1.00	1
Coffee	1.00	1.00	1.00	16
Rice	1.00	1.00	1.00	7
Tea	1.00	1.00	1.00	1
Tomato	1.00	1.00	1.00	7
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

2nd Model - Cinnamon Diseases Model

(Cinnamon Rough Bark, Cinnamon Stripe Canker and Cinnamon Healthy)

Accuracy vs epochs graph

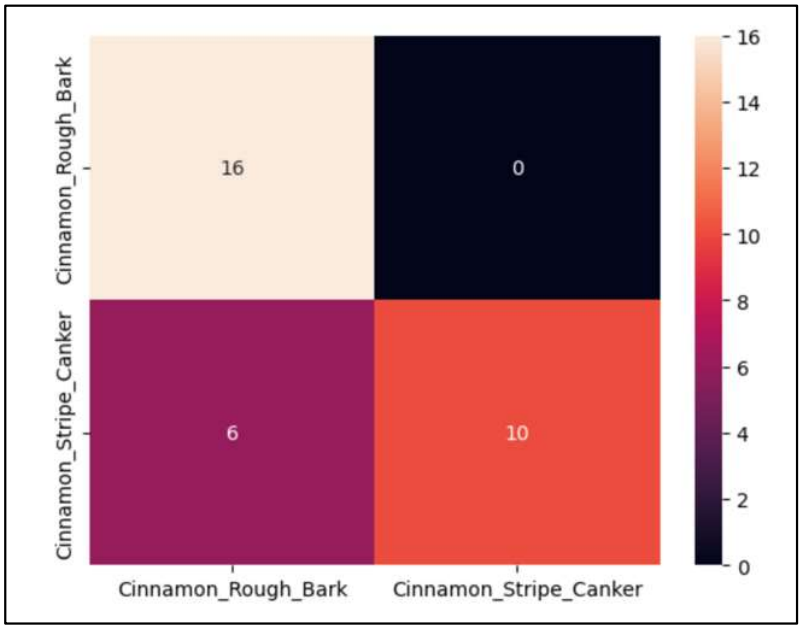


Model Accuracy

9.Evaluating the model using Test data

```
In [13]: # from tensorflow import keras
         # model = keras.models.load_model('Models_h5/model2.h5')
         score,accuracy =model.evaluate(test_generator,verbose=1)
         print("Test score is {}".format(score))
         print("Test accuracy is {}".format(accuracy))

         2/2 [==============================] - 3s 840ms/step - loss: 0.3840 - accuracy: 0.8095
         Test score is 0.3839763104915619
         Test accuracy is 0.8095238208770752
```

Confusion Matrix



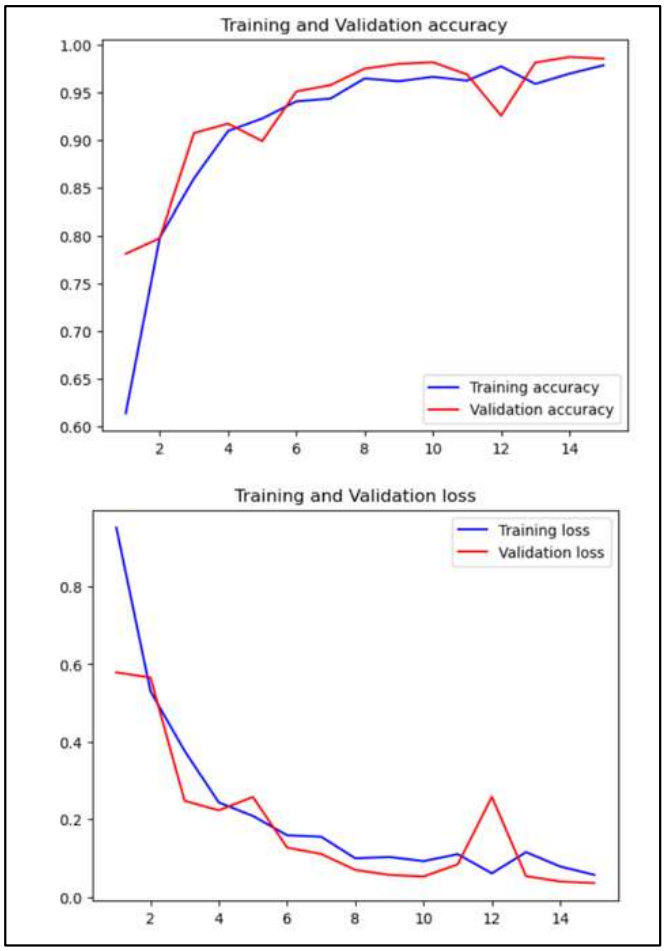
Classification Report – Precision | Recall | f1-score

```
In [27]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred, target_names=class_names))

                                precision    recall  f1-score   support

            Cinnamon_Rough_Bark       0.73      1.00      0.84        16
          Cinnamon_Stripe_Canker       1.00      0.62      0.77        16

                        accuracy                           0.81        32
                       macro avg       0.86      0.81      0.81        32
                    weighted avg       0.86      0.81      0.81        32
```

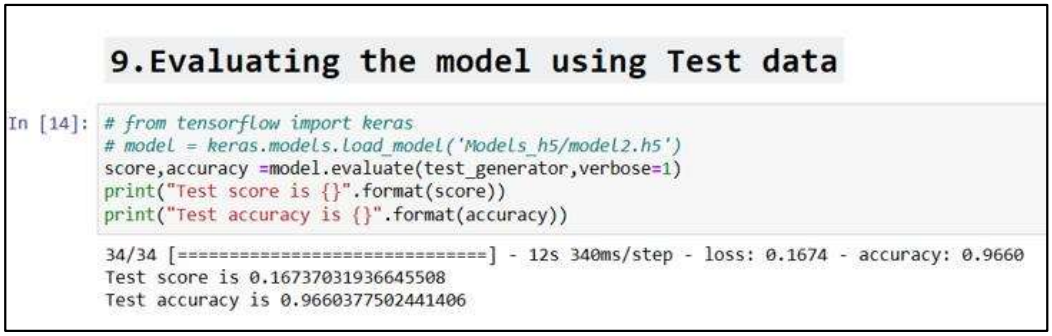
3rd Model - Rice Diseases Model

(Rice Bacterial Blight, Rice Blast, Rice Brown Spot, Rice Tungro, Rice Healthy)

Accuracy vs epochs graph



Model Accuracy

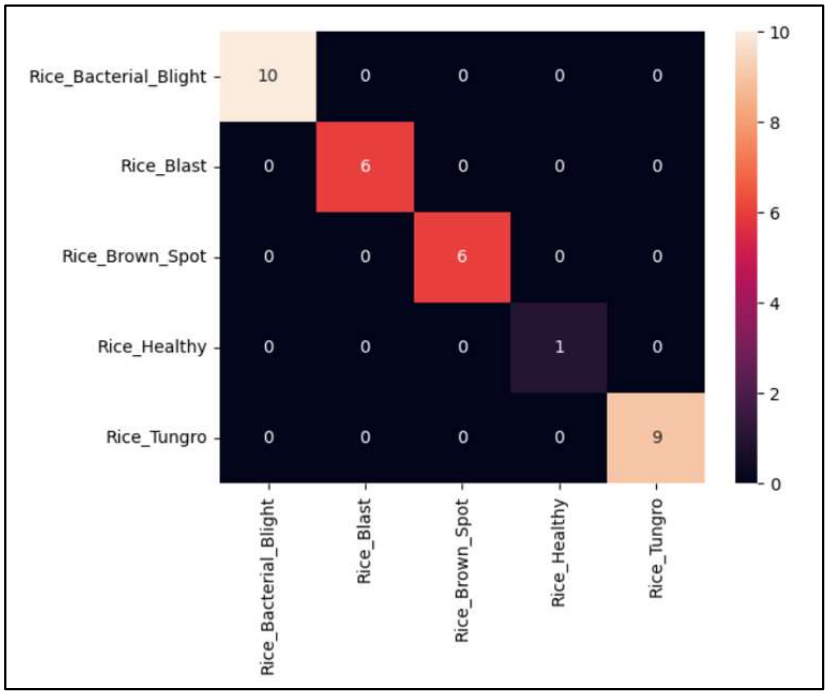


9.Evaluating the model using Test data

In [14]:
```
# from tensorflow import keras
# model = keras.models.load_model('Models_h5/model2.h5')
score,accuracy =model.evaluate(test_generator,verbose=1)
print("Test score is {}".format(score))
print("Test accuracy is {}".format(accuracy))
```

```
34/34 [==============================] - 12s 340ms/step - loss: 0.1674 - accuracy: 0.9660
Test score is 0.16737031936645508
Test accuracy is 0.9660377502441406
```

Confusion Matrix



Classification Report – Precision | Recall | f1-score

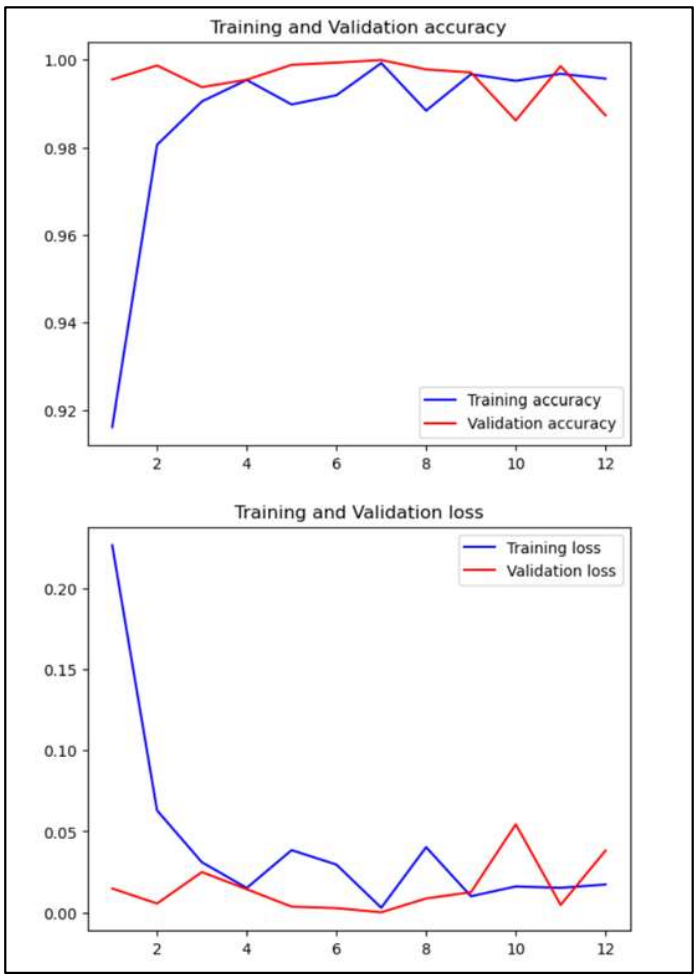
```
In [27]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred, target_names=class_names))
```

	precision	recall	f1-score	support
Rice_Bacterial_Blight	1.00	1.00	1.00	10
Rice_Blast	1.00	1.00	1.00	6
Rice_Brown_Spot	1.00	1.00	1.00	6
Rice_Healthy	1.00	1.00	1.00	1
Rice_Tungro	1.00	1.00	1.00	9
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

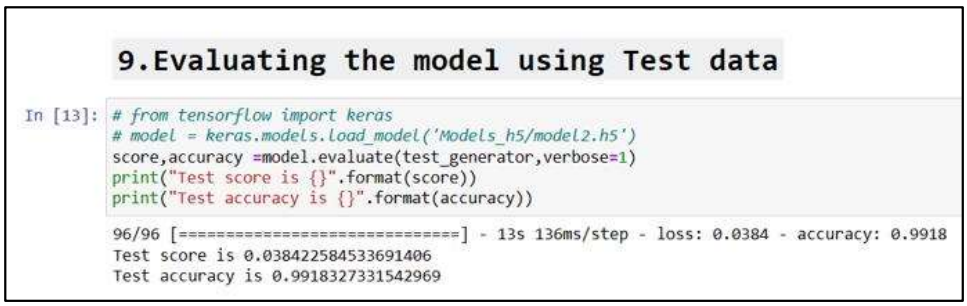
4th Model - Coffee Diseases Model

(Coffee Cerscospora, Coffee Phoma, Coffee Leaf Rust and Coffee Healthy)

Accuracy vs epochs graph



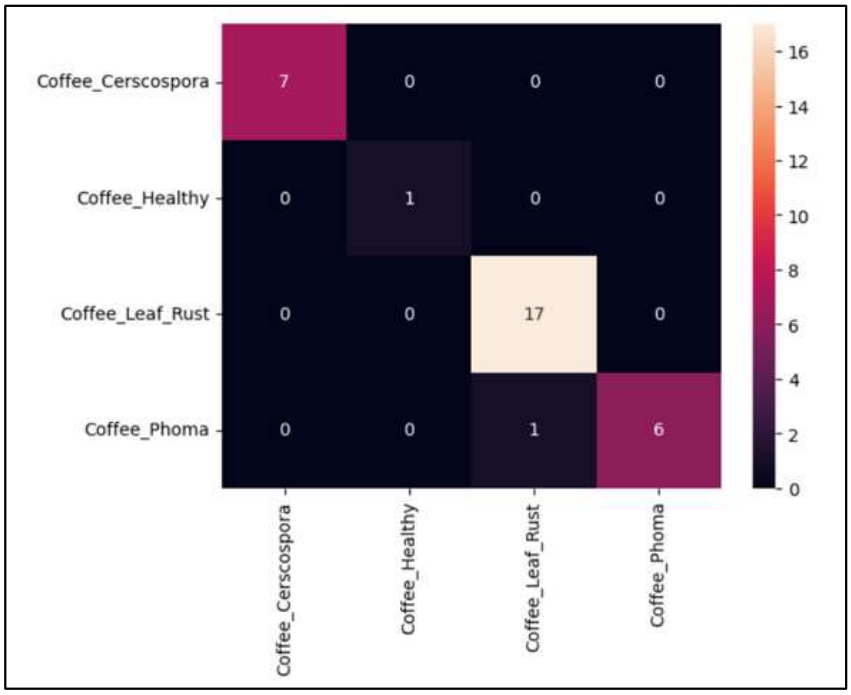
Model Accuracy



9.Evaluating the model using Test data

```
In [13]: # from tensorflow import keras
         # model = keras.models.load_model('Models_h5/model2.h5')
         score,accuracy =model.evaluate(test_generator,verbose=1)
         print("Test score is {}".format(score))
         print("Test accuracy is {}".format(accuracy))

         96/96 [==============================] - 13s 136ms/step - loss: 0.0384 - accuracy: 0.9918
         Test score is 0.038422584533691406
         Test accuracy is 0.9918327331542969
```

Confusion Matrix



Classification Report – Precision | Recall | f1-score

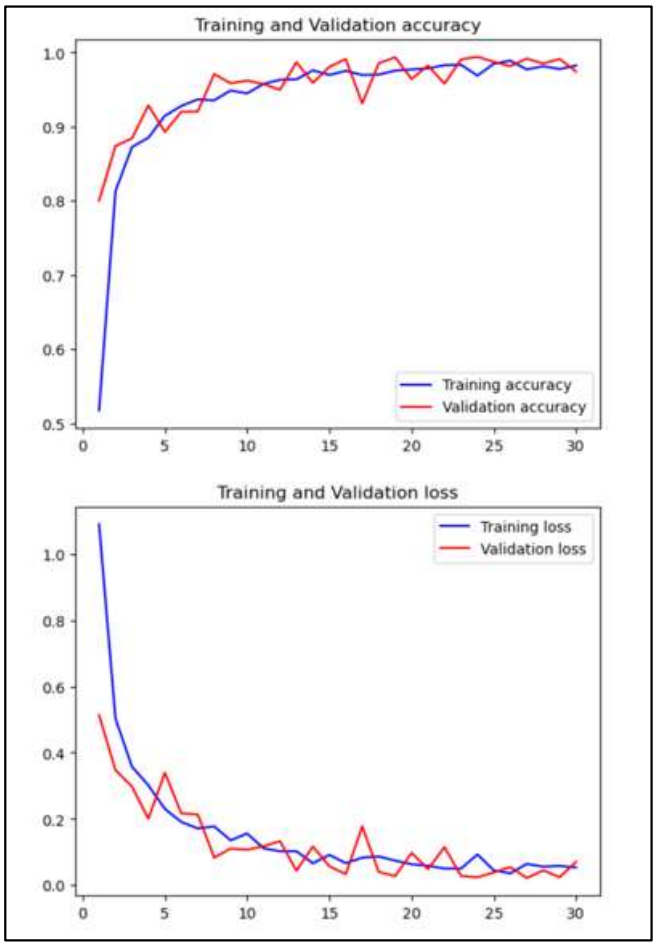
```
In [49]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred, target_names=class_names))
```

	precision	recall	f1-score	support
Coffee_Cerscospora	1.00	1.00	1.00	7
Coffee_Healthy	1.00	1.00	1.00	1
Coffee_Leaf_Rust	0.94	1.00	0.97	17
Coffee_Phoma	1.00	0.86	0.92	7
accuracy			0.97	32
macro avg	0.99	0.96	0.97	32
weighted avg	0.97	0.97	0.97	32

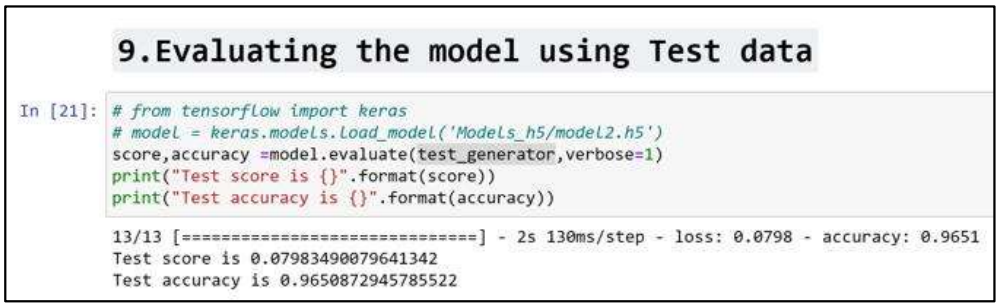
5th Model - Tomato Diseases Model

(Tomato Late Blight, Tomato Leaf Mold, Tomato Septoria Leaf Spot, Tomato Healthy)

Accuracy vs epochs graph



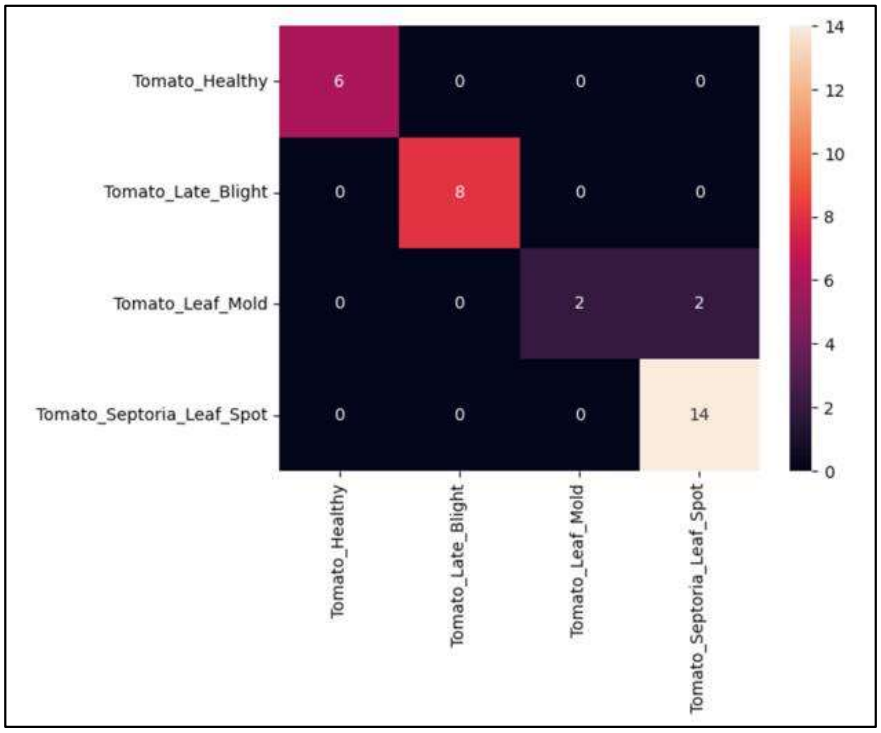
Model Accuracy



9.Evaluating the model using Test data

```
In [21]: # from tensorflow import keras
         # model = keras.models.load_model('Models_h5/model2.h5')
         score,accuracy =model.evaluate(test_generator,verbose=1)
         print("Test score is {}".format(score))
         print("Test accuracy is {}".format(accuracy))

         13/13 [==============================] - 2s 130ms/step - loss: 0.0798 - accuracy: 0.9651
         Test score is 0.07983490079641342
         Test accuracy is 0.9650872945785522
```

Confusion Matrix



Classification Report – Precision | Recall | f1-score

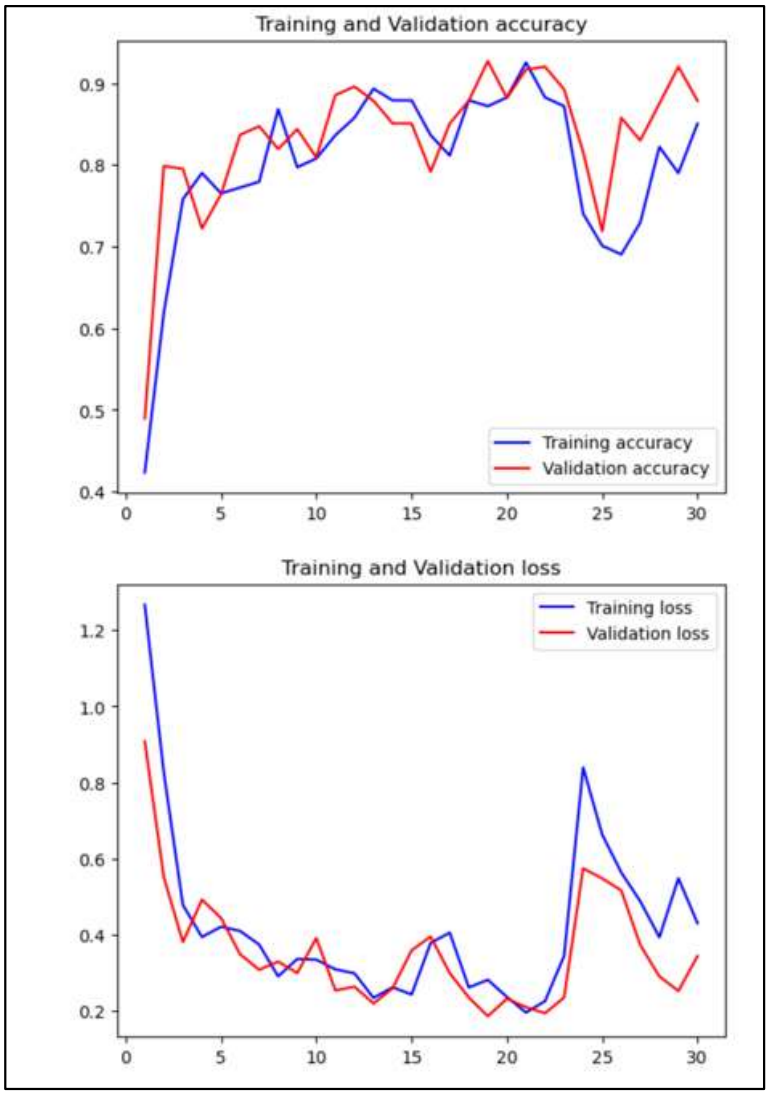
```
In [73]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred, target_names=class_names))
```

	precision	recall	f1-score	support
Tomato_Healthy	1.00	1.00	1.00	6
Tomato_Late_Blight	1.00	1.00	1.00	8
Tomato_Leaf_Mold	1.00	0.50	0.67	4
Tomato_Septoria_Leaf_Spot	0.88	1.00	0.93	14
accuracy			0.94	32
macro avg	0.97	0.88	0.90	32
weighted avg	0.95	0.94	0.93	32

6th Model - Tea Diseases Model

(Tea Algal Leaf, Tea Anthracnose, Tea Bird Eye Spot, Tea Healthy)

Accuracy vs epochs graph



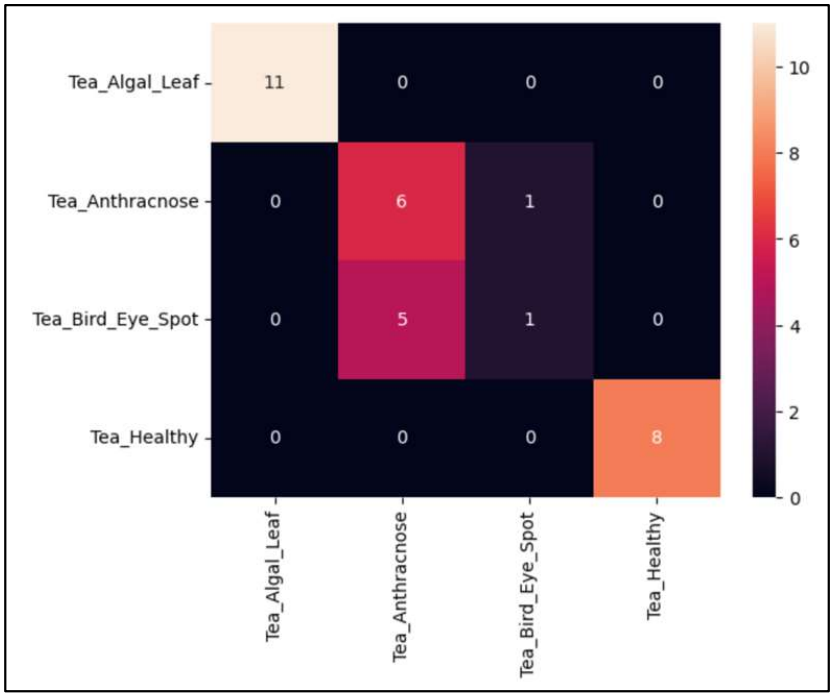
Model Accuracy

9.Evaluating the model using Test data

```
In [13]: # from tensorflow import keras
         # model = keras.models.load_model('Models_h5/model2.h5')
         score,accuracy =model.evaluate(test_generator,verbose=1)
         print("Test score is {}".format(score))
         print("Test accuracy is {}".format(accuracy))
```

```
3/3 [==============================] - 5s 1s/step - loss: 0.6123 - accuracy: 0.7703
Test score is 0.6123449206352234
Test accuracy is 0.7702702879905701
```

Confusion Matrix



Classification Report – Precision | Recall | f1-score

In [29]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred, target_names=class_names))
```

```
                    precision    recall  f1-score   support

     Tea_Algal_Leaf       1.00      1.00      1.00        11
    Tea_Anthracnose       0.55      0.86      0.67         7
Tea_Bird_Eye_Spot         0.50      0.17      0.25         6
       Tea_Healthy        1.00      1.00      1.00         8

          accuracy                           0.81        32
         macro avg       0.76      0.76      0.73        32
      weighted avg       0.81      0.81      0.79        32
```

8.6 Chapter Summary

This chapter initially documents the goals and objectives of testing. The testing of the application was done using different testing techniques such as confusion matrix in which aspects such as accuracy, precision recall and f1 score was considered, Functional Testing, Non-Functional Testing, Integration Testing was performed to check the extensibility or modularity and Model Testing. Finally benchmarking techniques such as Competitive Benchmarking, Functional Benchmarking, and Internal Benchmarking were used against existing systems.

CONCLUSION

9.1 Chapter Overview

The target audience and domain experts' ratings of the project's various components are covered in this chapter. The system's development, usability, and other crucial elements. After their evaluation, authors can also self-evaluate. It was also recorded whether the system's original goals were attained.

9.2 Project Aims and Progress

9.2.1 Aim of project

To design, develop and evaluate a mobile application that can identify diseases which can affect various plants**.** To provide any farmer or agriculturist with smartphone access to a cutting-edge plant protection system.

9.2.2 Completion of Deliverables

Deliverables	Date
Project Proposal	**Completed**
Literature Review Evaluating and analyzing chosen modules and comparison	
Software Requirement Specification The document specifying the requirements which is required to deliver the final product of the project	
System Design Document The document describes the proposed recommendation system's design and overviews of the algorithms to be built.	
Flutter Project App source code	
Deep Learning Model Saved .h5 files for the six models, Saved .tflite files for the six models & Saved optimized .tflite files for the six models.	
App UI Designs & prototypes Low and High-fidelity wireframes and screenshots.	
Final Document Final Document refers to this Document, which contains all project documentation.	
Android App (Final Software Product) The functioning final mobile application with all the features specified in initial proposal functional requirements.	

9.3 Problems Faced

There are several limitations and problems that can be faced while developing a plant-based disease detection application with machine learning, such as:

- Lack of sufficient and high-quality data: Collecting and labelling large amounts of plant disease data can be time-consuming and expensive.
- Difficulty in obtaining accurate and consistent labels: The process of labelling images of plant diseases can be subjective and prone to human error.
- Limited generalizability: Models trained on data from one region or climate may not perform well when applied to other regions or climates.
- Complexity of plant diseases: Plant diseases can manifest in a variety of ways and can be caused by multiple factors, making it difficult to develop a model that can accurately detect all types of diseases.
- Technical issues: Building, training, and deploying machine learning models can require specialized knowledge and resources, which can be a barrier for some developers.
- Limited interpretability of the models: Some models are hard to interpret, which can make it hard to understand why the model is making certain predictions.
- Limited application or scalability: Model performance may not be robust enough to be used as a commercial product.
- Flutter map integration using mapbox package: cused a issue and had to use flutter_map package.
- Some datasets were not available to download directly from the Kaggle dataset. –
- The dataset was downloaded to Google Colab using the Kaggle API.
- Some libraries were not installed into the anaconda environment. Some were not imported.
- Installed libraries.
- -version incompatibility issues. – downgrade the library version
- Data availability and quality: A lack of high-quality and sufficient data can make it difficult to train accurate models. Data availability and quality are major challenges when it comes to developing machine learning-based plant disease detection applications. One reason for this is that plant disease data is often scattered and unorganized, making it difficult to collect and use for training models. Additionally, the quality of the data may be poor, with issues such as inconsistencies, missing values, and errors that can negatively impact the performance of the models. Furthermore, it is difficult to get high-resolution images or labelled data for many plant diseases, which is essential for training machine learning models. Additionally, there is a lack of publicly available datasets that are specifically tailored to plant disease detection, which further compounds the challenge of obtaining high-quality data.
- Data annotation: Data annotation is the process of labelling and categorizing data for use in machine learning. When developing a plant-based disease detection application with machine learning, data annotation is an important step in creating a dataset that can be used to train the model. This process can be time-consuming and expensive because it typically involves manually reviewing and labelling large amounts of data, such as images of plants, to identify specific features or characteristics that may indicate the presence of a disease. Additionally, it is important to have a diverse dataset with different

variations of the disease to improve the model's performance, which increases the time and cost required for data annotation.

- Variability in plant appearance: Plants can vary greatly in appearance, making it difficult to develop models that can accurately detect diseases in all cases.
- Intra-species variability: Even within a single species of plant, there can be significant variations in the appearance of different individuals. This makes developing a machine learning model that can quickly detect a disease based on visual attributes difficult.
- Inter-species variability: Different species of plants can have vastly different appearances, making it challenging to develop a single machine learning model that can accurately identify diseases across multiple species.
- External conditions: Such as lighting, temperature, and humidity can all have an impact on the appearance of a plant. Because of these differences, developing a machine learning model that can accurately identify a disease in different settings can be difficult.
- Overfitting: With limited data, models may be overfit to the training data, which can lead to poor performance on new, unseen data.
- Domain expertise: Developing a plant-based disease detection application requires domain expertise in both machine learning and plant pathology.
- Choosing the right model: Choosing the right model for the task can be challenging as different models may work better for different types of data and tasks.
- Real-time detection: Some applications require real-time detection, which can be difficult to achieve with machine learning models.
- Privacy and security: Collecting and using images of plants for disease detection may raise privacy and security concerns.
- Deployment: Deploying machine learning models in the field can be challenging, especially if the device or infrastructure is limited.

9.4 Existing Skills & New Skills

- **Testing**: During the system tests for deep learning, some of the group members' prior testing experience from their studies in software testing units came in handy.
- **Modelling** Relational Data In NoSQL

- **Machine Learning**: Members of the team had a comprehensive understanding of how machine learning and deep learning operate and how to use them effectively. Our previous degree-related projects gave us a deep understanding of the fundamentals of programming. We were able to clear up any confusions and identify the kind of machine learning that best correlated to our requirements.

- **Analysis**: To make conclusions, analysts look at data from a variety of sources. They carry out background research on approaches to ensure and solutions that are comparable to ours. To assess the advantages, disadvantages, risks, and opportunities presented by the creation of such an application, they also conduct market research.

- Agronomy or Plant Pathology: Understanding the issue and coming up with precise labels require knowledge of plant biology and plant illnesses.Mobile app development: To create a user-friendly and accessible app, you must have knowledge about mobile app development.User experience design: To develop an intuitive and user-friendly interface, one must be familiar with the concepts of user experience design.Data management and storage: For the purpose of storing and retrieving data, knowledge of data management and storage strategies, such as SQL and NoSQL databases, is required.

- **Security**: To ensure the security and privacy of the app's data, knowledge of security principles and technologies, such as encryption and authentication, is required.

9.5 Deviations and Justifications

FR ID	Requirement	Completion Status (and reason)	Priority Level
FR1	The user could be able to register for an account.	Complete	M
FR2	The user should be able to authenticate himself/herself.	Complete	M
FR3	The user must be able to select the type of plant.	Complete	M
FR4	The user must be able to take/insert a picture using the mobile app.	Complete	M
FR5	The app must analyse the given image using the trained model.	Complete	M
FR6	The app must be able to predict and display the disease type using ML.	Complete	M
FR7	The app should be able to show a description of the predicted disease.	Complete	S
FR8	The app could let the user input the location or get the live location from the user's mobile when a disease has been found.	Complete	C
FR9	The app could display a map with all the locations where diseases have been found. (Using the data collected by all users)	Complete	C
FR10	The app could display the disease detection history of the user.	Complete	C
FR11	The user could be able to export the history of the detection results as a PDF document.	Complete	C
FR12	The app will not have a notification system.	Complete	W
FR13	The app will not diagnose the diseases which are not included in the datasets.	Complete	W
FR14	The app will not permit video sources as input.	Complete	W

9.6 Future Enhancements

- Utilizing a variety of imaging techniques, including thermal and hyperspectral imaging, to provide more thorough details regarding the state of the plant.
- Incorporating sensor data as well as information from other sources, such as meteorological data, to give a more thorough picture of the environment around the plant.
- Building a more diverse dataset to improve the accuracy and the amount of plant types in model.
- Using AI methods that can be easily explained to increase comprehension and transparency of the model's decision-making process to the user.
- To offer real-time and remote monitoring of plants in the field, the application could be integrated with other systems, such as the Internet of Things (IoT) and precision agriculture technologies.
- Integrating blockchain technology into the system to enable secure and impenetrable data and result recording.

APPENDIX 1: REFERENCES

Altexsoft. (2019). Fraud Detection: How Machine Learning Systems Help Reveal Scams in Fintech, Healthcare, and eCommerce. AltexSoft. https://www.altexsoft.com/whitepapers/fraud-detection-how-machine-learning-systems-help-reveal-scams-in-fintech-healthcare-and-ecommerce/

Chai, J., Zeng, H., Li, A., & Ngai, E. W. T. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications, 100134. https://doi.org/10.1016/j.mlwa.2021.100134

Chen, J. (2019). Android Operating System: What You Need to Know. Investopedia. https://www.investopedia.com/terms/a/android-operating-system.asp

Cinnamon – Dept. of Export Agriculture. (n.d.). Dea.gov.lk. http://dea.gov.lk/cinnamon/

Couto, J., Brenner, M., Marin, G., Rios, B., Toscano, M., & Fagioli, M. (2020, April 16). How Machine Learning is reshaping Price Optimization. Tryolabs. https://tryolabs.com/blog/price-optimization-machine-learning

Facebook. (2022). React – A JavaScript library for building user interfaces. Reactjs.org. https://reactjs.org/

Food crisis in Sri Lanka likely to worsen amid poor agricultural production, price spikes and ongoing economic crisis, FAO and WFP warn | World Food Programme. (2022, September 12). Www.wfp.org. https://www.wfp.org/news/food-crisis-sri-lanka-likely-worsen-amid-poor-agricultural-production-price-spikes-and-ongoing

How a fertilizer ban became a part of Sri Lanka's Crisis. (2022, May 29). Sri Lanka News - Newsfirst. https://www.newsfirst.lk/2022/05/29/how-a-fertilizer-ban-became-a-part-of-sri-lankas-crisis/

Huang, J., Li, J., Li, Z., Zhu, Z., Shen, C., Qi, G., & Yu, G. (2022). Detection of Diseases Using Machine Learning Image Recognition Technology in Artificial Intelligence. Computational Intelligence and Neuroscience, 2022, 1–14. https://doi.org/10.1155/2022/5658641

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. Electronic Markets, 31. https://doi.org/10.1007/s12525-021-00475-2

Keras. (2019). Home - Keras Documentation. Keras.io. https://keras.io/

Liu, T., Fang, S., Zhao, Y., Wang, P., & Zhang, J. (n.d.). Implementation of Training Convolutional Neural Networks. Retrieved January 17, 2023, from https://arxiv.org/ftp/arxiv/papers/1506/1506.01195.pdf#:~:text=CNN%20is%20an%20efficient%20recognition

Max Pooling in Convolutional Neural Networks explained. (2018, February 9). Deeplizard.com. https://deeplizard.com/learn/video/ZjM_XQa5s6s#:~:text=Max%20pooling%20is%20a%20type

Media, O. (2018, June 31). From logistics regression to self-driving cars: Chances and challenges for machine learning in highly automated driving. Embedded Computing Design. https://embeddedcomputing.com/application/automotive/from-logistics-regression-to-self-driving-cars-chances-and-challenges-for-machine-learning-in-highly-automated-driving

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science, 7. https://doi.org/10.3389/fpls.2016.01419

Ng, E. Y.-K., & Lim, J. T. (2022). Machine Learning on Fault Diagnosis in Wind Turbines. Fluids, 7(12), 371. https://doi.org/10.3390/fluids7120371

React Native · A framework for building native apps using React. (2022). Reactnative.dev. https://reactnative.dev/

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, *2*(3). Springer. https://doi.org/10.1007/s42979-021-00592-x

The Ultimate Guide to Evaluation and Selection of Models in Machine Learning. (2020, November 2). Neptune.ai. https://neptune.ai/blog/the-ultimate-guide-to-evaluation-and-selection-of-models-in-machine-learning

Understanding Feed Forward Neural Networks in Deep Learning. (2022, October 14). Www.turing.com. https://www.turing.com/kb/mathematical-formulation-of-feed-forward-neural-network

van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. Computers and Electronics in Agriculture, 177, 105709. https://doi.org/10.1016/j.compag.2020.105709

World Food Programme. (2022). A global food crisis | World Food Programme. Www.wfp.org. https://www.wfp.org/global-hunger-crisis

Zhang, T., & Mo, H. (2021). Reinforcement learning for robot research: A comprehensive review and open issues. International Journal of Advanced Robotic Systems, 18(3), 172988142110073. https://doi.org/10.1177/17298814211007305

Importance of Machine Learning - Javatpoint. (2011, May 31). Www.javatpoint.com. https://www.javatpoint.com/importance-of-machine-learning

Flutter Development: Everything You Need to Know. (2015, June 12). Radixweb. https://radixweb.com/flutter-development

Tavasoli, S. (2016, December 24). *The Importance of Machine Learning for Data Scientists | Simplilearn*. Simplilearn.com. https://www.simplilearn.com/importance-of-machine-learning-for-data-scientists-article

Meigarom Lopes. (2017, March 28). Is LDA a dimensionality reduction technique or a classifier algorithm? Medium; Towards Data Science. https://towardsdatascience.com/is-lda-a-dimensionality-reduction-technique-or-a-classifier-algorithm-eeed4de9953a

Hosack, P., & Miller, L. (2017, June). Preventing and Managing Plant Diseases. Extension.missouri.edu. https://extension.missouri.edu/publications/mg13

Anyoha, R. (2017, August 28). The History of Artificial Intelligence. Science in the News; Harvard University. https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

Narkhede, S. (2018, May 9). Understanding Confusion Matrix - Towards Data Science. Medium. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Wilimitis, D. (2019, February 21). The Kernel Trick. Medium. https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

Zaid Alissa Almaliki. (2019, March 19). *Do you know how to choose the right machine learning algorithm among 7 different types?* Towards Data Science; Towards Data Science. https://towardsdatascience.com/do-you-know-how-to-choose-the-right-machine-learning-algorithm-among-7-different-types-295d0b0c7f60

Alexandre Gonfalonieri. (2019, November 7). How to Implement Machine Learning For Predictive Maintenance. Medium; Towards Data Science. https://towardsdatascience.com/how-to-implement-machine-learning-for-predictive-maintenance-4633cdbe4860

Munnangi, M. (2019, November 21). Crop: Plant Disease Identification Using Mobile App. Medium. https://towardsdatascience.com/crop-plant-disease-identification-using-mobile-app-aef821d1a9bc

Pisner, D. A., & Schnyer, D. M. (2020, January 1). Chapter 6 - Support vector machine (A. Mechelli & S. Vieira, Eds.). ScienceDirect; Academic Press. https://www.sciencedirect.com/science/article/pii/B9780128157398000067

Grieve, P. (2020, January 23). *Deep learning vs. machine learning: What's the difference?* Zendesk. https://www.zendesk.com/blog/machine-learning-and-deep-learning/#:~:text=Deep%20learning%20definition%3A%20A%20subfield

NexusAdmistraIntegra. (2020, March 6). *The 4 industries that benefit the most from machine learning*. Nexus Integra EN. https://nexusintegra.io/the-4-industries-that-benefit-the-most-from-machine-learning/

Afham, M. (2020, May 19). Singular Value Decomposition and its applications in Principal Component Analysis. Medium. https://towardsdatascience.com/singular-value-decomposition-and-its-applications-in-principal-component-analysis-5b7a5f08d0bd

Sasidharan, A. (2021, January 20). *Support Vector Machine Algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/support-vector-machine-algorithm/

Burns, E. (2021, March). *What Is Machine Learning and Why Is It Important?* SearchEnterpriseAI. https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

Malato, G. (2021, May 19). *Hyperparameter tuning. Grid search and random search*. Your Data Teacher. https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/#:~:text=Grid%20search%20is%20the%20simplest

Machine Learning Random Forest Algorithm - Javatpoint. (2021, July 23). Www.javatpoint.com. https://www.javatpoint.com/machine-learning-random-forest-algorithm

Meltzer, R. (2021, July 15). *What is Random Forest? [Beginner's Guide + Examples]*. Careerfoundry.com. https://careerfoundry.com/en/blog/data-analytics/what-is-random-forest/#:~:text=Random%20Forest%20is%20a%20powerful

breton. (2021, August 1). *A Full Guide on Choosing the Right Machine Learning Algorithm*. Medium. https://medium.com/@davidbreton03/a-full-guide-on-choosing-the-right-machine-learning-algorithm-5fa282a0b2a1

Korstanje, J. (2021, August 31). The F1 score | Towards Data Science. Medium. https://towardsdatascience.com/the-f1-score-bec2bbc38aa6

Chen, B. (2021, November 5). Why Rectified Linear Unit (ReLU) in Deep Learning and the best practice to use it with TensorFlow. Medium. https://towardsdatascience.com/why-rectified-linear-unit-relu-in-deep-learning-and-the-best-practice-to-use-it-with-tensorflow-e9880933b7ef#:~:text=The%20Rectified%20Linear%20Unit%20(ReLU)%20is%20the%20most%20commonly%20used

Klingler, N. (2021, November 17). Jupyter Notebook for Machine Learning - A Gentle Introduction. Viso.ai. https://viso.ai/deep-learning/jupyter-notebook-for-machine-learning/#:~:text=The%20platform%20is%20based%20on

Sharma, P. (2021, November 24). *Understanding K-Means Clustering Algorithm*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/11/understanding-k-means-clustering-in-machine-learningwith-examples/#:~:text=k%2Dmeans%20is%20a%20technique

QuinnRadich. (2021, December 30). *What is a machine learning model?* Learn.microsoft.com. https://learn.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model

Raitoharju, J. (2022, January 1). Chapter 3 - Convolutional neural networks (A. Iosifidis & A. Tefas, Eds.). ScienceDirect; Academic Press. https://www.sciencedirect.com/science/article/pii/B9780323857871000087

Nyuytiymbiy, K. (2022, January 15). *Parameters and Hyperparameters in Machine Learning and Deep Learning*. Medium. https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac#:~:text=Hyperparameters%20are%20parameters%20whose%20values

Goled, S. (2022, April 22). An argument against using Jupyter Notebook for Machine Learning. Analytics India Magazine. https://analyticsindiamag.com/an-argument-against-using-jupyter-notebook-for-machine-learning/#:~:text=Computing%20platforms%20like%20Jupyter%20Notebook

Selman, H. (2022, April 27). The history of machine learning - Dataconomy. Dataconomy. https://dataconomy.com/2022/04/the-history-of-machine-learning/

Davidson, T. (2022, May 15). Cross Platform App Development: Developing an Application for Maximum Exposure | Clean Commit. Cleancommit.io. https://cleancommit.io/blog/cross-platform-app-development-developing-an-application-for-maximum-exposure/

Spiceworks. (2022, May 20). *Linear Regression vs. Logistic Regression: Understanding 13 Key Differences | Spiceworks*. Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/linear-regression-vs-logistic-regression/#:~:text=Linear%20regression%20is%20utilized%20for

Singh, P. (2022, May 30). What is the Future of Machine Learning? Naukri. https://www.naukri.com/learning/articles/future-of-machine-learning/

Supervised Learning | Machine Learning. (2022, July 18). Google Developers. https://developers.google.com/machine-learning/intro-to-ml/supervised

What is Machine Learning? (2022, July 18). Google Developers. https://developers.google.com/machine-learning/intro-to-ml/what-is-ml

anton. (2022, July 24). AutoML Current Uses and Approaches - PostIndustria. PostIndustria. https://postindustria.com/automl-current-uses-and-approaches/

Challa, A. (2022, July 31). What is a ReLU layer? Educative: Interactive Courses for Software Developers. https://www.educative.io/answers/what-is-a-relu-layer

Fully connected layer - MATLAB. (2022, August 31). Www.mathworks.com. https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html

Cong, Y., Tao, M., & Leung, H. (2022, September 3). Generative Models in Signal Processing | Frontiers Research Topic. Www.frontiersin.org. https://www.frontiersin.org/research-topics/32911/generative-models-in-signal-processing

Angular. (2022, September 19). Angular.io. https://angular.io/guide/what-is-angular

Gillis, Alexander. S. (2022, October 18). What is native app? - Definition from WhatIs.com. SearchSoftwareQuality. https://www.techtarget.com/searchsoftwarequality/definition/native-application-native-app

Langholz, S. (2022, October 18). The Benefits and Challenges of iOS vs Android App Development. Customer Engagement Blog. https://onesignal.com/blog/the-benefits-and-challenges-of-ios-vs-android-app-development/

Posey, B. (2022, October 18). Apple iOS. SearchMobileComputing. https://www.techtarget.com/searchmobilecomputing/definition/iOS

Yasar, K., & Lewis, S. (2022, October 18). PyTorch. SearchEnterpriseAI. https://www.techtarget.com/searchenterpriseai/definition/PyTorch

D'Agostino, A. (2022, October 21). Why having many features can hinder your model's performance. Medium. https://towardsdatascience.com/why-having-many-features-can-hinder-your-models-performance-865369b6b8b1

Georgiou, M. (2022, November 2). Cross Platform App Development - Popular Frameworks and Trends in 2023. Imaginovation | Top Web & Mobile App Development Company Raleigh. https://imaginovation.net/blog/cross-platform-app-development-popular-frameworks-and-trends/

Progressive Web Apps. (2022, December 13). Web.dev. https://web.dev/progressive-web-apps/

Biswal, A. (2022, December 14). Convolutional Neural Network Tutorial [Update]. Simplilearn.com. https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network#:~:text=Flattening%20is%20used%20to%20convert

BasuMallick, C. (2022, December 27). What Is TensorFlow? Meaning, Working, and Importance |. Spiceworks. https://www.spiceworks.com/tech/devops/articles/what-is-tensorflow/

Biswal, A. (2023, January 6). Principal Component Analysis in Machine Learning | Simplilearn. Simplilearn.com. https://www.simplilearn.com/tutorials/machine-learning-tutorial/principal-component-analysis

APPENDIX 2: MINUTES OF MEETINGS

Minute No: 001
Date: 23 November 2022
Time: 9:00-10:30 pm

Venue: Google Meet
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
Action List
Items on the action list were completed or progressing as follows:
Item 1: Discussed Assignment Criteria. Status: Open
Item 2: Discussed a suitable problem domain for the project topic. Status: Open

General

Where To From Here
What you are going to do next.
Actions:
Item 1: Choose a Problem Domain.

ID	Action	Who	Due Date	Status
1	Set Up Team Communication Channel in Google Meets.	Malindu	Next Meet	Closed
2	Team Coordination.	Navindra	Next Meet	Closed
3	Brainstormed possible topic ideas.	ALL	Next Meet	Open
4	Set up a timetable.	ALL	Next Meet	Closed
5	Discussed the team contract and agreed on the meeting times.	ALL	Next Meet	Open

Minute No: 002
Date: 26 November 2022
Time: 8:00-9:30 pm
Venue: Google Meet
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
Action List
Items on the action list were completed or progressing as follows:
Item 1: Discussed possible problems in the chosen domain. Status: Closed
Item 2: Discussed possible solutions(Project Topic). Status Open

General

Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Create a shared word document for the proposal.	Hashan	Next Meet	Closed
2	Drafted the topics.	Adrian	Next Meet	Open
3	Selected a supervisor.	ALL	Next Meet	Closed
4	Choose a Domain.	ALL	Next Meet	Closed
5	Brainstorm ideas.	ALL	Next Meet	Open

Minute No: 003
Date: 29 November 2022
Time: 8:00-9:00 pm
Venue: Google Meet
Participants: Ms Inoshi, Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
Action List
Items on the action list were completed or progressing as follows:
Item 1: Selected a Project Topic. Status: Closed
Item 2: Requested for supervisor approval. Status: Open

General

Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Choose a Problem Domain.	ALL	Next Meet	Closed
2	Note down the discussed points.	Maleesha	Next Meet	Closed
3	Start researching.	ALL	Next Meet	Open
4	Get approval from the Supervisor about the topic.	ALL	Next Meet	Closed

Minute No: 004
Date: 2 December 2022
Time: 8.00 -12.00 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Started the Research. Status: Open
Item 2: Started learning Modules needed to develop the project app. Status: Open

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Set Up Team Communication Channel in MS Teams.	ALL	Next Meet	Open
2	Machine learning Tutorial.	Malindu	Next Meet	Closed
3	Flutter Tutorial.	Navindra	Next Meet	Closed
4	Firebase Tutorial.	Hashan	Next Meet	Closed
5	Topic Research.	ALL	Next Meet	Open

Minute No: 005
Date: 6 December 2022
Time: 8:30-12.30 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Discussed Project Proposal structures. Status: Open
Item 2: Continued Research. Status: Open

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Draft the project proposal.	Navindra	Next Meet	Open
2	Research on Possible ML models.	Malindu	Next Meet	Closed

Minute No: 006
Date: 8 December 2022
Time: 7:30-2.00 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Completed the Project Proposal. Status: Open

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Background	Navindra	Next Meet	Closed
2	Assumptions	Malindu	Next Meet	Closed
3	Gantt chart	Hashan	Next Meet	Closed
4	Tools and Technicals	Adrian	Next Meet	Closed
5	Reference	Maleehsa	Next Meet	Closed

Minute No: 007
Date: 9 December 2022
Time: 3:30-10.00 pm
Venue: MS Teams
Participants: Ms Inoshi, Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Completed the Project Proposal. Status: Closed
Item 2: Got Feedback from the Supervisor. Status: Closed

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Feedback Session.	ALL	Next Meet	Closed
2	Made Changes to the Document.	ALL	Next Meet	Closed

Minute No: 008
Date: 11 December 2022
Time: 5:30-2.00 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Reviewed the Project Proposal. Status: Closed
Item 2: Submission of the Project Proposal. Status: Closed

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Fine-tune the Project Document content.	Malindu	Next Meet	Closed
2	Submission of the Document.	Navindra	Next Meet	Closed

Minute No: 009
Date: 16 December 2022
Time: 5.30 pm-7.00 pm
Venue: MS Teams
Participants: Ms Inoshi, Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Discussed what to do next with the Supervisor. Status: Closed
Item 2: Further Research. Status: Closed

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Further Research	ALL	Next Meet	Closed
2	Discussed ideas with the Supervisor.	ALL	Next Meet	Closed

Minute No: 010
Date: 17 December 2022
Time: 7.30 pm-11.30 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Drafted the Literature Review. Status: Closed
Item 2: Drafted the Presentation Slides. Status: Closed
Item 3: Researched machine learning Models. Status: Closed

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Discussed ideas with the Supervisor.	ALL	Next Meet	Closed
2	Researched machine learning Models	Malindu	Next Meet	Closed
3	Drafted Presentation Slides	Navindra	Next Meet	Open
4	Drafted Literature Review	Adrian	Next Meet	Open
5	Took Notes	Maleesha	Next Meet	Closed

Minute No: 011
Date: 18 December 2022
Time: 8.30 pm-11.30 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Drawn High/Low-Fidelity Wireframes. Status: Closed
Item 2: Gather ideas for Flutter App User Interface. Status: Closed

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Wireframes	Navindra	Next Meet	Closed
2	Wireframes	Malindu	Next Meet	Closed
3	Gathering UI Ideas	Hashan	Next Meet	Open
4	Completed Presentation Slides	Navindra	Next Meet	Closed
5	Datasets Gathering for ML Model	Malindu	Next Meet	Open

Minute No: 012
Date: 18 December 2022
Time: 1 pm - 10 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Created Machine Learning Model. Status: Closed
Item 2: Gather Datasets to train the Models. Status: Closed
Item 3: Drawn High-Fidelity UI. Status: Closed

2. General

3. Where To From Here
What you are going to do next.
Actions:

ID	Action	Who	Due Date	Status
1	Datasets Fine Tuning.	Malindu, Navindra	Next Meet	Closed
2	Review the Datasets.	ALL	Next Meet	Open
3	ML Model Creation using Jupiter Notebook.	Malindu	Next Meet	Closed
4	ML Model Training.	Malindu	Next Meet	Open
5	ML Model Training.	Navindra	Next Meet	Open
6	Review The Models' accuracies and give Suggestions.	ALL	Next Meet	Open

Minute No: 013
Date: 19 December 2022
Time: 8 am - 2 pm
Venue: MS Teams
Participants: Navindra Abeyesundere, Malindu Mahara, Hashan Fernando, Adrian Jesudasan, Maleesha Gurunansleage
Apologies: None
Absent: None

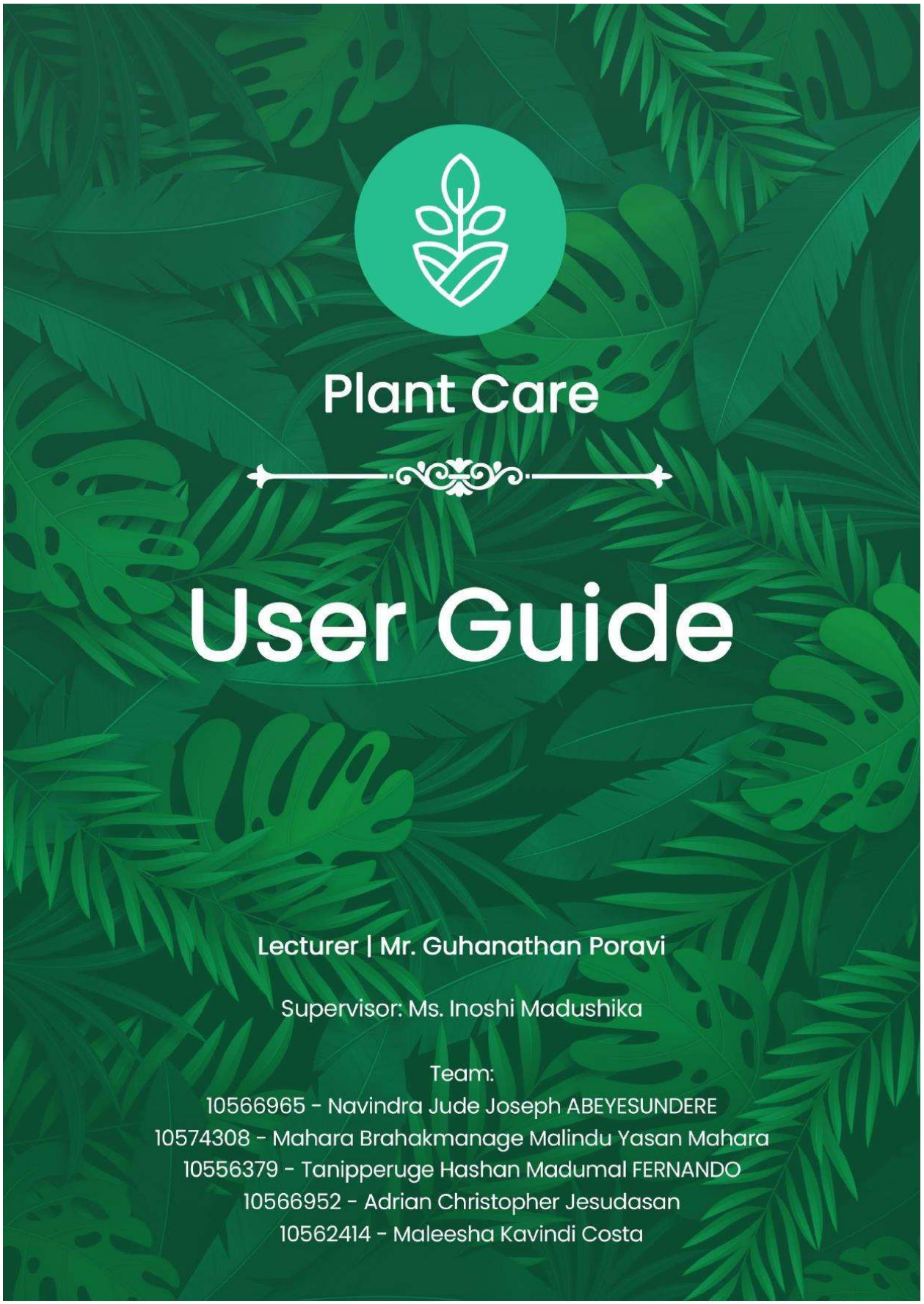
Discussion:
1. Action List
Items on the action list were completed or progressing as follows:
Item 1: Reviewed the Presentation. Status: Closed
Item 2: Preparation for the Presentation. Status: Closed

2. General

3. Where To From Here
What you are going to do next.
Actions:

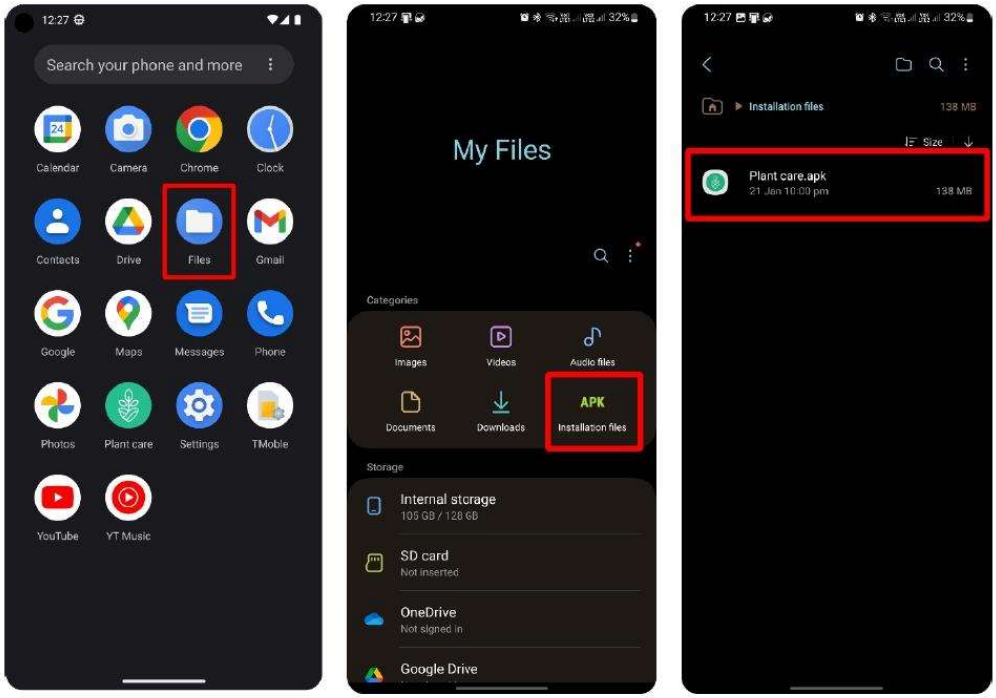
ID	Action	Who	Due Date	Status
1	Present the Presentation.	ALL	Next Meet	Closed
2	Video Editing and Submission of the Presentation.	Navindra	Next Meet	Closed
3	Review the Video.	Malindu	Next Meet	Closed

APPENDIX 3: USER GUIDE FOR APP

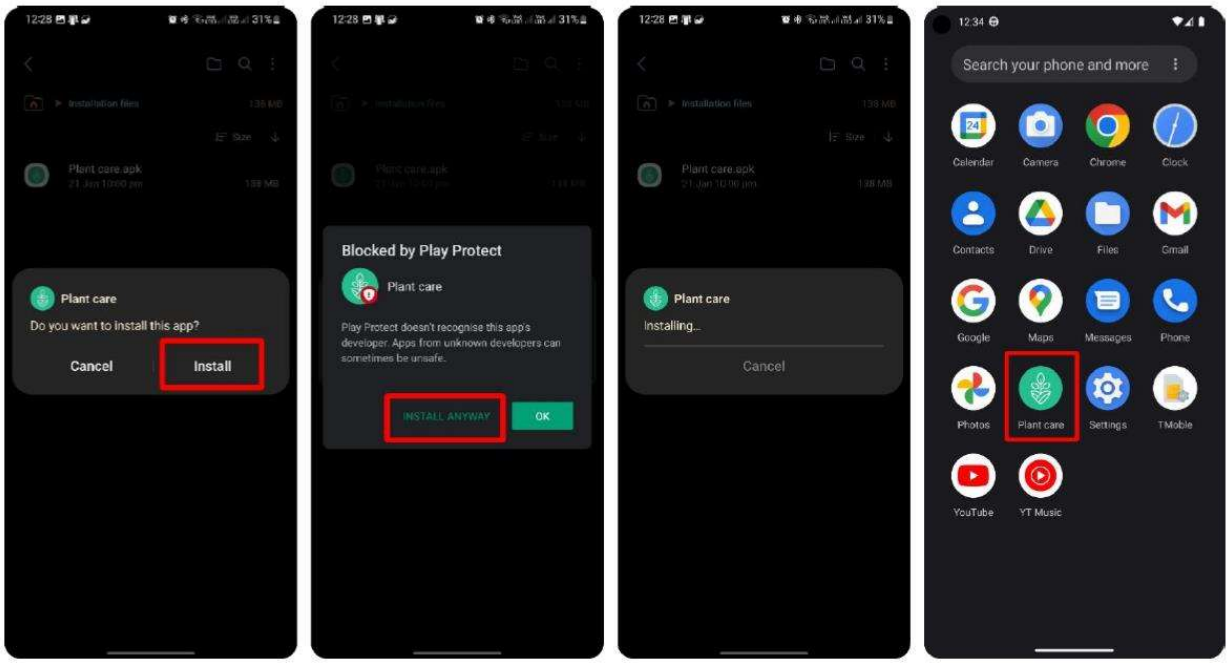


Install App on to device.

Step 1: Download App to device and locate download file in My Files as “Plant care.apk”.



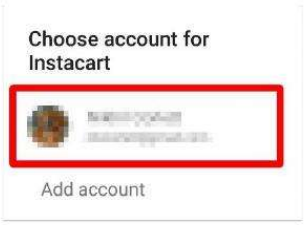
Step 2: Tap on file “Plant care.apk” and tap on “Install”. If your device tries to block installation tap on “INSTALL ANYWAY”. The app will install and then prompt to open or cancel. Tap on “open” to open app or locate app on device and tap to open.



Use app.

Step 1: Open app

Step 2: User can login to app by tapping on "Continue with Google" indicated by 1 then select account to continue. Or user can tap "SKIP" to skip logging into app.



Step 3: Quick guide will be displayed for the user read. User can tap "Continue" on the bottom centre.

Step 4: main navigations include the following.

1 – Main navigator of app. User can tap on icon or slide left or right to change the tab.

2 – Tap on user icon to navigate to Account settings.

3 – Main body of tab. Slide down to scroll down.

4 – Slide left for map

5 – Slide right for History