Plante Diseases

(MobileApplication)

Prepared By:

Omar Mahmoud Muhammad	**CS**
Khalil Ahmed Muhammad Ali	**CS**
Youssef Mohammad ElSayed	**CS**
Mahmoud Mohammad Ibrahim	**CS**
Asel Tarek Ali Ahmed	**CS**
Omnia Attia Ali	**CS**

Under the supervision of

Dr. Zahraa Youssef

2023/2024

Acknowledgment

First of all, thanks to "Allah" for whom I always pray, under the light of his holly face asking him to give us the ability to do our work kindly and faithfully.

Second, we thank our families for their patience, help, and support. From the depth of our hearts, special thanks and gratitude are offered to our supervisor:

Dr. Zahraa Youssef

Contents

1	Abstract	5
1.1	**Chapter 1**	6
1.2	Introduction	6
1.3	Flowchart	9
2	**Chapter 2 TOOLS**	10
2.1	Adobe XD	10
2.2	Git	13
2.3	Github	14
2.4	CNN model	15
2.5	Python	17
2.6	Jupyter	18
2.7	Flask API	19
2.8	MongoDB	20
2.9	VSCode	21
2.10	postman	22
2.11	Dart	23
2.12	Flutter	24
2.13	Comparison between Ionic Vs React Native Vs Flutter Vs Xamar	26
2.14	Install Flutter & Dart on Windows	27
2.15	Adding a Dart package to the IDE	27
2.16	Android Studio	28
2.17	Dio	29
3	**Chapter 3 Code**	30

3.1	AI	30
3.2	API “Flask”	41
3.3	Back end	43
3.4	Database	45
3.5	Flutter app	49
3.6	Output	55
4	**CONCLUSION**	66
5	**REFERENCES**	67

1 Abstract:

The innovative mobile application we present aims to revolutionize plant disease detection, serving agriculture enthusiasts, farmers, and researchers. Developed cross-platform, it seamlessly operates on Android and iOS devices. The app's core functionality involves image analysis of plant leaves using cutting-edge artificial intelligence algorithms, enabling accurate disease identification. The carefully crafted user interface ensures intuitive navigation, providing a seamless experience for users with diverse technical backgrounds. This pioneering technology offers timely and precise diagnoses, marking a significant advancement in plant health monitoring.

The goal of the application is:

1. Identify the disease and enable users to take and upload photos of plant leaves

2. Accurate and rapid diagnostics to prevent the spread of diseases

3. Determine the appropriate treatment for each disease according to valid scientific information

How does the application work?

1. Image Capture and Upload:
    - Users capture images of plant leaves through the application.
    - Uploaded images undergo processing through artificial intelligence algorithms.
2. Disease Analysis and Identification:
    - The application analyzes images to identify patterns indicative of plant diseases.
3. Mitigation Recommendations:
    - Based on disease type and severity, the application suggests effective mitigation strategies to manage and control the spread of diseases

Application advantages:

- Simple design
- Professional and simple user interface
- Accurate identification of the disease
- Effective and timely response
- Structured information
- Easy access to treatment of diseases
- The administrator can constantly update the treatments

1.1 Chapter 1

1.2 INTRODUCTION:

Mobile applications, now integral to our daily lives, have prompted our team to develop a solution using Flutter, a cross-platform framework for Android and iOS devices. This application specifically tackles a pressing concern for farmers and agricultural professionals – plant disease detection.

Traditionally, identifying plant diseases involves delays and relies on human expertise. Our application revolutionizes this process by integrating artificial intelligence (AI) and machine learning (ML). By capturing plant leaf images through the mobile app, users receive instant disease identification and treatment recommendations.

The ML functionality analyzes images through deep learning, trained on a diverse dataset of plant diseases. This enables real-time disease identification, providing timely insights crucial for effective disease management. The continuous learning aspect enhances diagnostic capabilities over time, positioning our solution at the forefront of agricultural technology innovation.

In essence, the fusion of Flutter's cross-platform capabilities and machine learning transforms our application into a powerful tool, offering users swift, accurate, and informed decisions for crop health.

Importance of plants in our life:

Plants are truly the unsung heroes of our planet, playing a vital role in our lives in countless ways. Here are some of the most important reasons why plants are so crucial:

1. Air production and purification: Plants are the lungs of the Earth. Through photosynthesis, they take in carbon dioxide, a greenhouse gas, and release oxygen, the life-giving gas we need to breathe. They also filter out pollutants and dust, contributing to cleaner air and better health.

2. Food source: All food chains begin with plants. We directly consume fruits, vegetables, grains, and nuts, and they indirectly provide sustenance for herbivores, which we then eat as meat. Without plants, our food system would collapse.

3. Medicine and healthcare: Plants have been used for medicinal purposes for centuries, and many modern pharmaceuticals are derived from plant extracts. From

painkillers and antibiotics to cancer treatments, plants continue to offer valuable resources for healthcare.

4. Material resources: We rely on plants for a wide range of materials, including wood for furniture and construction, paper for writing and packaging, cotton for clothing, and even rubber for tires. Plants provide us with sustainable alternatives to many non-renewable resources.

5. Ecosystem stability: Plants play a crucial role in maintaining healthy ecosystems. They prevent soil erosion, regulate water flow, provide habitat for wildlife, and contribute to biodiversity. Their presence ensures the stability of the natural world, which is essential for our survival.

Threats currently facing plants:

Despite their crucial role in our lives, plants face many threats that put their survival and the balance of our ecosystems at risk. Here are some of the most pressing:

1. Habitat loss and fragmentation: Land conversion for agriculture, urban expansion, and infrastructure development leads to the destruction and fragmentation of natural habitats, leading to the loss of species and disruption of vital ecological functions.

2. Climate change: Rising temperatures, changing rainfall patterns, and extreme weather events pose major challenges to plants, causing stress, changing habitats, and disrupting their ability to grow.

3. Invasive species: Exotic plants introduced into new environments often outcompete native species for resources, leading to displacement and loss of biodiversity.

4. Overexploitation: Unsustainable harvesting of plants for food, medicine, timber, and other resources can push species towards extinction and upset the ecological balance.

5. Pollution: Air and water pollution resulting from chemicals, fertilizers, and industrial waste can harm plants, impair their growth and reproduction, and disrupt the entire food chain.

Idea of our application and solution:

The Problem:

Crops are the backbone of our food supply, but diseases constantly threaten them. Early detection of plant diseases is crucial for preventing crop losses and

ensuring food security. Traditional methods often rely on visual inspection by experts, which can be time-consuming, subjective, and inaccurate.

The Solution:

A mobile application that leverages the power of computer vision and machine learning to diagnose plant diseases through the analysis of plant leaves. This app would empower farmers, gardeners, and anyone concerned with plant health to:

- Easily capture and upload images of diseased leaves.
- Receive instant and accurate diagnoses of the disease.
- Access information on disease symptoms, causes, and treatment options.
- Connect with experts for further guidance.

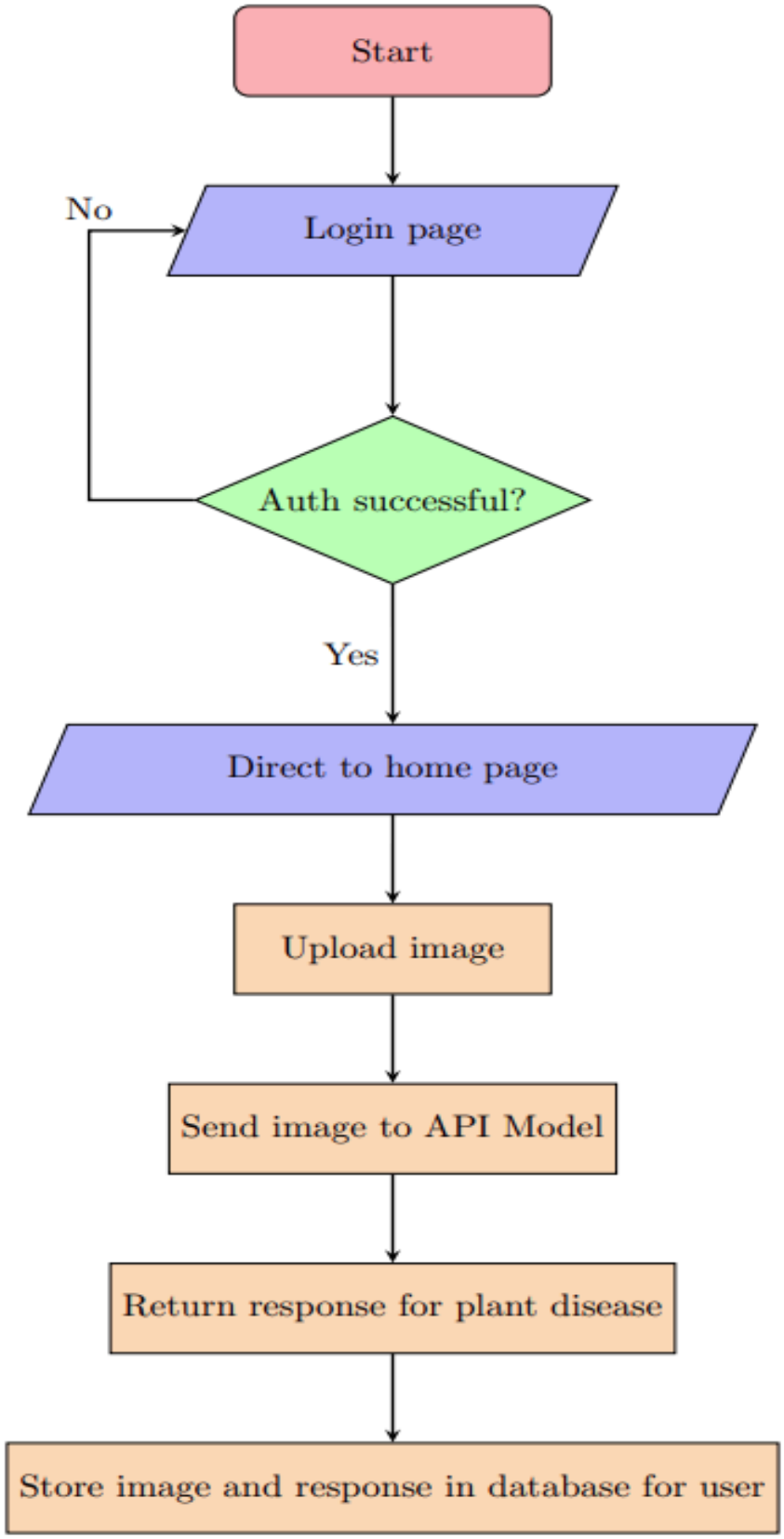
App Features:

- Image recognition: The app would use a pre-trained model to identify different plant species and diseases based on leaf images.
- Disease classification: The model would analyze the image for specific features like discoloration, spots, lesions, and leaf deformations to determine the most likely disease.
- Information Library: The app would provide a comprehensive database of plant diseases, including symptoms, causes, impact, and recommended management practices.
- Treatment recommendations: Based on the diagnosis, the app would suggest specific treatment options, including organic and conventional methods.
- Expert consultation: Users could connect with plant pathologists or other experts for personalized advice and support.

Benefits:

- Early detection and prevention of crop losses.
- Improved decision-making for disease management.
- Reduced reliance on chemical pesticides.
- Empowered farmers and gardeners to take charge of their plant health.
- Increased food security and sustainability.

1.3 Flowchart



2 Chapter 2 (Programs)

TOOLS

In the design (UI/UX) stage:

2.1 Adobe XD:

Adobe XD is a powerful and user-friendly design tool that helps you create stunning user interfaces (UI) and interactive prototypes for websites, mobile apps, and more. It offers a flexible workspace, intuitive tools, and robust prototyping features, making it a popular choice for both designers and developers.

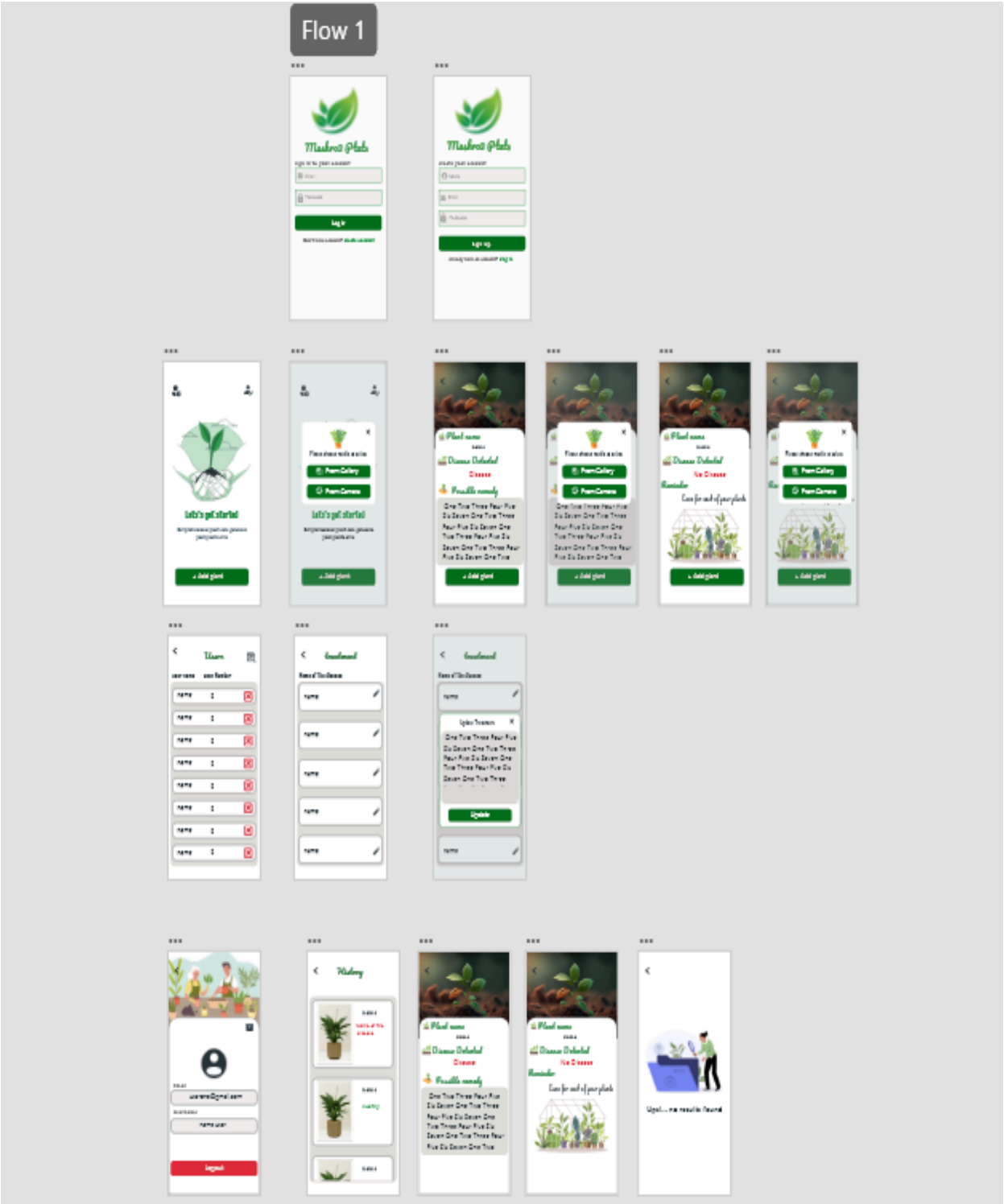
The most important features of Adobe XD are:

- Simple and efficient interface
- Responsive Design Test
- Connect items and test the product
- Accurately locate items (Fixed, absolute)
- Set a value for transition effects

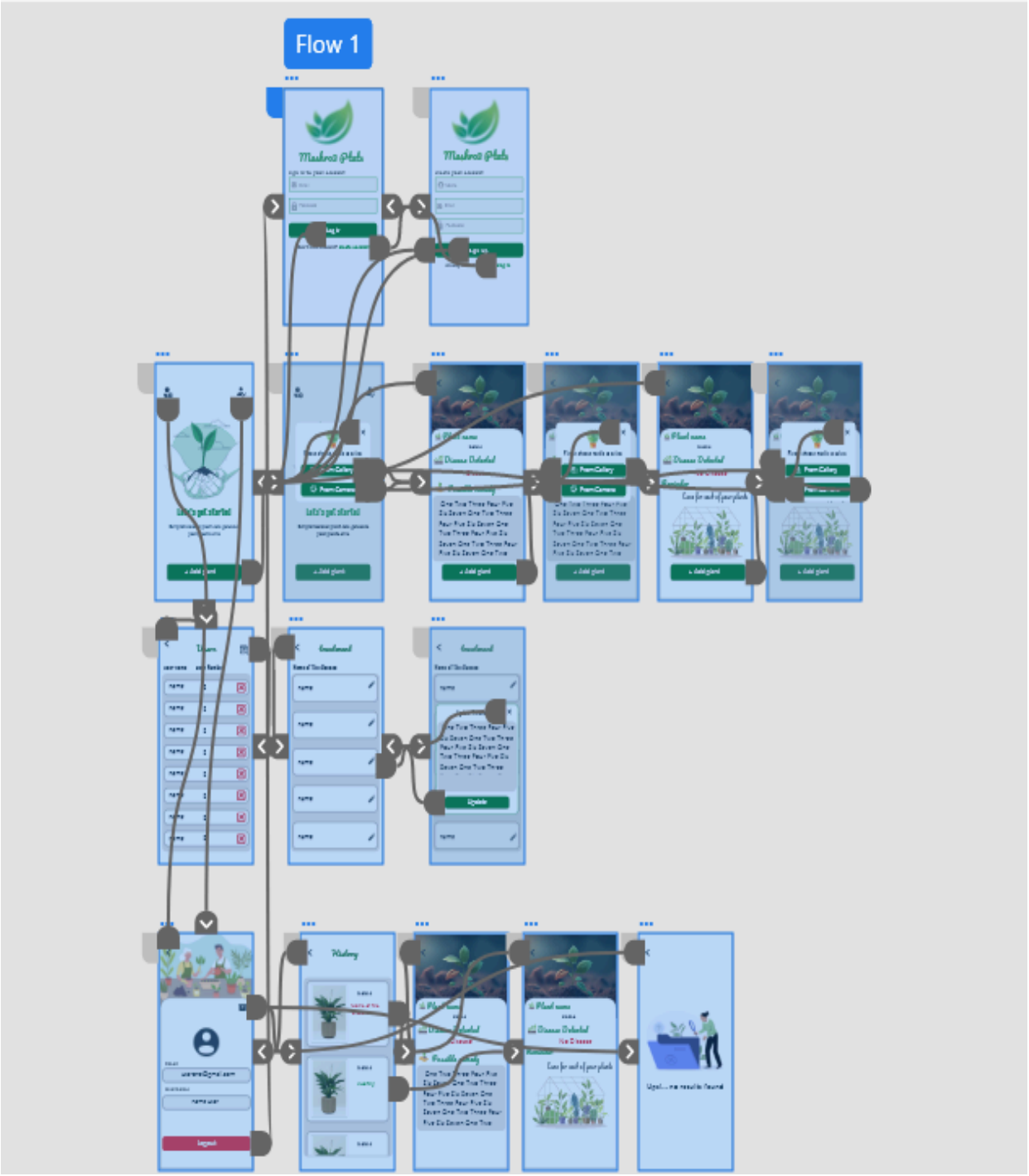
Download link:

https://community.adobe.com/t5/download-install/ct-p/ct-download-and-install?page=1&sort=latest_replies&filter=all&tabid=discussions

The Design screen of our project in Adobe XD:



The Prototype screen of our project in Adobe XD:



In the coding stage:

2.2 Git:

Git is a copy management system that allows you to save files and keep track of their different copies, whether these files belong to one person or several people involved in the files. For example: let's say you have a file containing a line of text, and you save that file in Git and then add two more lines to the file and save it again, Git will let you go back to the file's first state when it only had one line and it will also let you go once Other to last modified, and as long as you save all changes in Git all the copies you made will be available to you whenever you want.

Benefits of Git:

- Roll back to previous versions to avoid any errors during work.
- Keep track of all copies that have been worked freely.
- Git allows team members to simultaneously work on the same files so that everyone on the team has their copy.
- The possibility of creating branches (copies) of the original 15 work files, developing them independently, testing them without affecting the original file, and then merging them with it.

Download link: https://git-scm.com/downloads

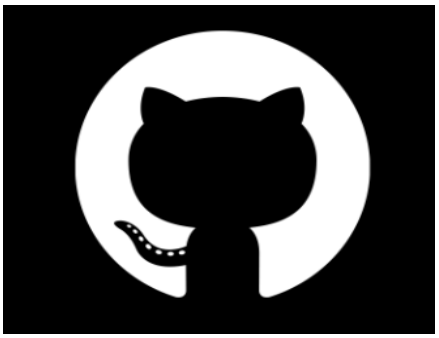
2.3 GitHub:

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere

GitHub essentials like repositories, branches, commits, and Pull Requests. You'll create your own Hello World repository and learn GitHub's Pull Request workflow, a popular way to create and review code.

GitHub takes things a little further than Git, offering more functionality and resources, as well as an online place to store and collaborate on projects.

A GitHub desktop app is also available, which is designed to simplify essential steps in your GitHub workflow and replace GitHub for Mac and Windows with a unified experience across both platforms.



Download link: https://desktop.github.com/

Some common commands for using Git:

- git init: initializes a new Git repository
- git clone: Creates a local copy of a project that already exists remotely
- git merge: merges development lines together. This command is usually used to combine changes made to two distinct branches
- git pull: Updates the local development line with updates from its remote counterpart.
- git push: Updates the remote repository with any commits made locally to a branch.

2.4 CNN model:

What are CNNs?

CNNs are a type of deep learning algorithm particularly well-suited for tasks involving visual imagery, like image recognition and classification. They excel at finding patterns and extracting features from images, making them powerful tools for various AI applications

How do CNNs work?

Imagine a CNN as an assembly line for analyzing images. Here's a simplified breakdown of the process:

1. Input Layer: The image is first fed into the network as an array of numbers representing the pixel values.
2. Convolutional Layer: This layer applies filters (small matrices of weights) to the image. These filters slide across the image, detecting specific patterns and features. Think of it like scanning the image for edges, textures, or shapes.
3. Pooling Layer: This layer downsamples the output of the convolution, reducing the dimensionality and computational load. It helps summarize the key features extracted by the filters.
4. Activation Layer: This layer introduces non-linearity into the network, allowing it to learn more complex features. It adds a "decision-making" element to the feature extraction process.
5. Fully-Connected Layer: This layer acts like a traditional neural network, connecting all the neurons in the previous layer to all the neurons in the next. It takes the extracted features and uses them to make predictions or classifications.
6. Output Layer: The final layer outputs the network's prediction, such as the image's class (cat, dog, car, etc.) or a probability distribution across different classes.

Benefits of CNNs:

- High Accuracy: CNNs can achieve remarkable accuracy in image recognition and classification tasks, often surpassing human performance.
- Feature Learning: They automatically learn features from data, eliminating the need for manual feature engineering, a tedious and time-consuming process.
- Robustness: CNNs are robust to variations in the input data, such as changes in lighting, pose, or scale.

Applications of CNNs:

- Image Recognition and Classification: Facial recognition, object detection in self-driving cars, medical image analysis, and product recommendations.
- Natural Language Processing: Sentiment analysis, text summarization, and machine translation.
- Signal Processing: Speech recognition, anomaly detection, and time series forecasting.

Examples of Popular CNN Models:

- **AlexNet:** One of the first successful CNNs, winning the ImageNet competition in 2012.
- **VGG16:** A deeper and more complex network than AlexNet, achieving even higher accuracy on ImageNet.
- **ResNet:** Introduced residual connections, which helped alleviate the vanishing gradient problem and train deeper networks effectively.
- **Inception:** Used multiple filters of different sizes in parallel, improving feature extraction and efficiency.

The future of CNNs:

CNNs are a rapidly evolving field with ongoing research and development. They are expected to play an increasingly important role in various AI applications, pushing the boundaries of what's possible in areas like computer vision, robotics, and healthcare.

INTRODUCTION OF AI PHASE:

load model
test model
confusion matrix
deployment model

Dataset:

The New Plant Diseases Dataset is a comprehensive collection of images depicting various plant diseases and their corresponding healthy states. It consists of over 87,000 images of 14 different plants.

2.5 Python:

Python is a high-level, general-purpose programming language known for its simplicity, readability, and versatility. Its intuitive syntax, extensive libraries, and supportive community make it a popular choice for various tasks, from web development and data science to scripting and automation.

Strengths:

- Easy to Learn and Use: Python's syntax is clear and concise, making it beginner-friendly and easy to pick up.
- Highly Readable: Code is self-explanatory, improving code maintainability and collaboration.
- Versatile and Powerful: Supports various tasks like web development, data science, machine learning, scripting, and automation.
- Extensive Libraries and Frameworks: A rich ecosystem of libraries and frameworks for specific functionalities.
- Large and Active Community: Offers vast resources, tutorials, and support for developers of all levels.

- Free and Open-Source: Freely available for use and contributions, promoting open collaboration

Download Link:

https://www.python.org/downloads/

2.6 Jupyter:

Jupyter is an open-source, interactive web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It supports various programming languages, making it a versatile tool for data analysis, machine learning, and scientific research.

Strengths:

- Interactivity: Jupyter Notebooks provide an interactive computing environment where you can execute code in a step-by-step fashion, making it ideal for data exploration and analysis.
- Language Agnostic: Supports multiple programming languages, including Python, R, Julia, and more, allowing users to choose the language that best suits their needs.
- Rich Output: Enables the creation of rich content, including plots, charts, and multimedia elements, directly within the notebook.
- Educational Tool: Widely used in education for teaching and learning programming, data science, and various scientific disciplines.
- Community Support: Jupyter has a large and active community, contributing to a wealth of extensions, plugins, and resources.

2.7 Flask API:

Flask is a lightweight and flexible web framework written in Python. It is popular for developing web applications and APIs due to its simplicity, ease of use, and robust features. Flask's modular design and extensive ecosystem of extensions make it a versatile tool for building various applications.

Strengths:

- Lightweight and Minimalistic: Requires minimal dependencies and resources, ideal for small and scalable applications.
- Easy to Learn and Use: Simple and intuitive API with clear documentation makes it beginner-friendly.
- Modular Design: Allows building applications with modular components for clean and organized code.
- Extensive Extension Ecosystem: Thousands of extensions are available for various functionalities like authentication, databases, and caching.
- Unit Testing: Built-in support for unit testing facilitates robust and maintainable code.
- RESTful API Development: Offers tools and functionalities for building efficient and secure RESTful APIs.
- Rapid Prototyping: Enables quick development and prototyping of web applications.

2.8 MongoDB:

MongoDB is a highly scalable and flexible NoSQL database designed for modern applications. It offers document-oriented data storage, providing powerful querying capabilities and flexible data models. MongoDB's ease of use, high performance, and scalability make it a popular choice for various projects.

Strengths:

- Document-oriented: Stores data in flexible JSON-like documents, facilitating complex data structures.
- High Performance: Offers fast read and write speeds, ideal for demanding applications.
- Scalability: Horizontally scales to accommodate growing data volumes and user demands.
- Rich Querying Capabilities: Supports powerful queries using JSON-style syntax.
- Sharding: Distributes data across multiple servers for increased performance and scalability.
- Open Source: Available under an open-source license, encouraging community collaboration and customization.

Download Link:

https://www.mongodb.com/try/download/community

2.9 VSCode:

VSCode is a free and open-source code editor developed by Microsoft. It is widely used by developers of all levels for its powerful features, extensibility, and user-friendly interface. VSCode offers support for various programming languages and frameworks, making it a versatile tool for any development project.

Strengths:

- User-friendly interface: Easy to learn and navigate, with a customizable layout.
- Lightweight and efficient: Uses few resources and runs smoothly on various systems.
- Extensive extension library: Thousands of extensions are available for various functionalities and languages.
- Integrated terminal: Provides a built-in terminal for command-line tasks.
- Git integration: Supports Git version control directly within the editor.
- Debugging tools: Offers built-in debugging functionalities for different languages.
- Syntax highlighting and code completion: Enhances code readability and writing efficiency.
- Multilingual support: Supports multiple languages for the interface and documentation

Download Link:

https://code.visualstudio.com/download

2.10 Postman:

Postman is a comprehensive API development and testing platform that simplifies the process of designing, testing, and documenting APIs. It offers a user-friendly interface for teams to streamline collaboration throughout the API development lifecycle.

Features:

- API Testing: Allows you to create and execute automated tests for your APIs.
- API Design: Build and prototype APIs quickly with a visual interface.
- Collaboration: Share and collaborate on APIs with team members.
- Monitoring: Keep an eye on API performance and uptime.
- Documentation: Automatically generates API documentation to ensure clarity and understanding.
- Collections: Organize and manage your API requests into collections for efficient testing and development.

How to Use (in Simple Bold Lines):

- Create Requests: Easily create HTTP requests for testing APIs.
- Organize in Collections: Group related requests into collections for better management.
- Execute Tests: Run automated tests to ensure API functionality.
- Visualize Data: View responses in a human-readable format for quick analysis.

2.11 Dart:

Dart is a programming language created by Google and used in web, desktop, and mobile applications. This language was invented by Lars Bak and Kasper Lund and its first version was released in 2011. It is important to know that Dart is a Platform-Cross language, meaning that it works on various platforms, and it is also a language Native, meaning that it deals with hardware directly without intermediate interpreters This gives it a very high speed.

Many consider it an unsuccessful language, but the statistics say the opposite. Dart is an object-oriented programming language that was made by Google as an attempt to replace the JavaScript language, but it did not work.

Dart is an object-oriented programming language, and it implements all concepts of object-oriented programming, including multiple inheritance, under the name mixins.

The Flutter team has full control over the Dart language to fit Flutter. Recently, there has been a big change in the language to fit Flutter.

For example, the word new, which will be recognized by the Java programmer and C#, was made an optional word only to improve the way interfaces are written.

While programming, you use the JIT compiler Dart language: Just in time, thanks to which the Hot Reload feature appeared, which is that when you change the code, only the bytes that have been changed are sent, so the change occurred in your program in less than a second and it remains in its state.

During application deployment, Dart uses the AOT: Ahead of Time compiler, which results in an original application and thus performs great.

2.12 Flutter:

Flutter is simply a smartphone application development package or SDK, specifically used for developing user interfaces or UI and uses the Dart programming language for back-end programming, one of the most important things that was focused on when developing Flutter is to make it a practical way to develop applications quickly and efficiently.

What distinguishes Flutter is that it enables us to build applications for different systems, including Android or IOS for Apple devices, what is even more amazing is that it can also be used as the first language for programming system applications

The new Google "Fuchsia", may displace Android from its place.
We should also know that Flutter is based on Design Material, which was built by Google, and helps design web pages.

Companies or developers who want to reach users in Android and ISO platforms with minimal effort and cost consider Flutter a very suitable solution.

Flutter also helps designers build applications that have a modern and modern design through ready-to-use packages and also with excellent documentation. It also provides so-called widgets, which are UI components in the application, which are available with practical applications that provide appropriate practices for building applications.

Strengths:

- **High Performance:** Flutter utilizes its rendering engine, resulting in significantly smoother and faster performance compared to the other frameworks, especially on high-end devices. This is crucial for our project as it ensures a superior user experience for our target audience.
- **Custom UI/UX:** Flutter offers unparalleled flexibility and control over the user interface. We can create unique and visually stunning designs that

perfectly align with our project's vision and brand. This level of customization is not readily available in other frameworks.

- **Single Codebase:** With Flutter, we can write code once and deploy it to both Android and iOS platforms, significantly reducing development time and effort. This is particularly advantageous for our project, as it allows us to focus on core functionalities and deliver the app to a wider audience.
- **Modern Language:** Dart, the language used in Flutter, is modern, easy to learn, and object-oriented. This makes it accessible to developers familiar with other object-oriented languages, minimizing the learning curve.
- **Hot Reload:** This feature allows us to see changes made to the code reflected in the running app instantly, without restarting the application. This dramatically improves development speed and productivity.
- **Rich Ecosystem:** Flutter boasts a rapidly growing ecosystem of libraries, packages, and tools, providing ready-made solutions for various functionalities. This saves us valuable development time and resources.

2.13 Comparison between Ionic Vs React Native Vs Flutter VsXamar:

Attribute	React Native	Xamarin	Ionic	Flutter
Programming Language	JavaScript + Swift,Objective-C or Java	C# with .net environment	HTML5, CSS, and JavaScript + Typescript	Dart
Performance	Close-to-native ★★★★½	Xamarin Ios/Android: Close-to-native ★★★★ / Xamarin Forms: Moderate ★★★	Moderate ★★	Amazing ★★★★★
GUI	Use Native UI Controllers	Use Native UI Controllers	HTML,CSS	Use Proprietary Widgets and deliver amazing UI
Market and Community Support	Very Strong	Strong	Strong	Not very popular
Use Cases	All apps	Simple apps	Simple apps	All apps
Code Reusability	90% of code is reusable	96% of code is reusable	98% of code is reusable	50-90% (approx.)of code is reusable
Popular Apps	Facebook, Instagram, Airbnb, UberEats	Olo, the World Bank, Storyo.	JustWatch, Pacifica, and Nationwide.	HamiltTon
Pricing	Open-source	Open-source + Paid as well	Open-source + Paid as well	Open-source

2.14 Install Flutter & Dart in Windows:

1- Install IntelliJ IDEA or Android Studio.
2- Go to the following website and download the file:
https://docs.flutter.dev/get-started/install
3- It is recommended that the file be saved to the C:\disk in a special folder.
4- Download the Java packages before starting anything if they are not available.
5- Run the file inside the folder called Flutter_console.bat.
6- After the installation is complete, run cmd and type Flutter in case the package is not recognized we add the package folder in the C disk to the system variables.
7- In the next step, after identifying Flutter, we write in cmd the phrase doctor Flutter, which Tests the availability of all Dart & Flutter requirements.

2.15 Adding a Dart package to the IDE:

_ In this step, we will learn to add a Dart package to Android Studio or IDEA IntliJ:

- We open the desired IDE.
- We choose Configuration, then Plug-ins
- Type Flutter and click Install.
- We restart the IDE program

2.16 Android Studio:

Android Studio is the official integrated development environment (IDE) for Android app development. Built on JetBrains' IntelliJ IDEA platform, it provides a comprehensive set of tools and features specifically designed for Android developers to build modern and high-quality mobile applications.

Most Popular Features:

- Layout Editor: Visually design and build app layouts using a drag-and-drop interface.
- Code Editor: Powerful and customizable code editor with syntax highlighting, code completion, and code refactoring tools.
- Emulator and Device Support: Run and test your app on a variety of virtual and physical devices.
- Debugging Tools: Identify and fix bugs with ease using powerful debugging tools and features.
- Build Tools: Build and deploy your app to various channels like the Google Play Store.
- Support for Multiple Programming Languages: Develop your app using Java, Kotlin, C++, and other languages.
- Gradle Support: Build automation tool for managing dependencies and build configurations.
- Version Control Integration: Integrate with version control systems like Git for better collaboration.
- Large and Active Community: Access extensive documentation, tutorials, and support from the community.

Download Link:

https://developer.android.com/studio

2.17 Dio:

Dio is a powerful Dart package for making HTTP requests in Flutter applications, providing a straightforward way to interact with RESTful APIs. It's widely used for handling HTTP requests and responses, offering features for customization, error handling, and asynchronous operations.

Strengths:

- Simplicity: Dio simplifies HTTP requests with a clean and easy-to-use API, suitable for both beginners and experienced developers.
- Asynchronous Support: Dio leverages Dart's asynchronous capabilities, allowing for efficient handling of multiple concurrent requests.
- Customization: Provides flexibility with customizable options for headers, query parameters, request and response interceptors, and more.
- FormData Handling: Excellent support for working with multipart/form-data requests, commonly used for file uploads.
- Interceptors: Interceptors enable you to globally intercept requests or responses, adding functionality like logging, authentication, or error handling.

3 Chapter 3

Codes:

3.1 Ai:

TensorFlow:

TensorFlow, the AI maestro, crafts intelligent models, be it for vision-seeking robots or language-savvy assistants. Its flexible framework effortlessly handles deep learning symphonies and vast datasets, empowering all from novices to seasoned AI conductors to shape the future one masterpiece at a time.

Keras:

Keras, a high-level API on TensorFlow, is the chef's knife for deep learning. Its modular design and concise syntax enable easy creation of diverse architectures. Perfect for quick prototyping, Keras makes deep learning accessible, promoting readability and rapid iteration for both seasoned AI chefs and curious novices.

```
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
import matplotlib.pyplot as plt
tf.compat.v1.set_random_seed(0)
from tensorflow import keras
import numpy as np
np.random.seed(0)
import itertools
import os as os
import pandas as pd
import numpy as np
from tensorflow.keras.utils import image_dataset_from_directory
from tensorflow.keras.layers.experimental.preprocessing import Rescaling
from sklearn.metrics import precision_score, accuracy_score, recall_score, confusion_matrix, ConfusionMatrixDisplay
```

About dataset:

Number of images for each class:

Tomato__Late_blight	1851
Tomato__healthy	1926
Grape__healthy	1692
Orange__Haunglongbing_(Citrus_greening)	2010
Soybean__healthy	2022
Squash__Powdery_mildew	1736
Potato__healthy	1824
Corn_(maize)__Northern_Leaf_Blight	1908
Tomato__Early_blight	1920
Tomato__Septoria_leaf_spot	1745
Corn_(maize)__Cercospora_leaf_spot Gray_leaf_spot	1642
Strawberry__Leaf_scorch	1774
Peach__healthy	1728
Apple__Apple_scab	2016
Tomato__Tomato_Yellow_Leaf_Curl_Virus	1961
Tomato__Bacterial_spot	1702
Apple__Black_rot	1987
Blueberry__healthy	1816
Cherry_(including_sour)__Powdery_mildew	1683
Peach__Bacterial_spot	1838
Apple__Cedar_apple_rust	1760
Tomato__Target_Spot	1827
Pepper,_bell__healthy	1988
Grape__Leaf_blight_(Isariopsis_Leaf_Spot)	1722
Potato__Late_blight	1939
Tomato__Tomato_mosaic_virus	1790
Strawberry__healthy	1824
Apple__healthy	2008
Grape__Black_rot	1888
Potato__Early_blight	1939
Cherry_(including_sour)__healthy	1826
Corn_(maize)__Common_rust_	1907
Grape__Esca_(Black_Measles)	1920
Raspberry__healthy	1781
Tomato__Leaf_Mold	1882
Tomato__Spider_mites Two-spotted_spider_mite	1741
Pepper,_bell__Bacterial_spot	1913
Corn_(maize)__healthy	1859

Some images from the dataset:

Tomato_Early_blight **Peach_Bacterial_spot** **Apple_Black_rot**



Blueberry_healthy **Apple_healthy** **Grape_healthy**



OpenCV:

OpenCV, a widely-used computer vision library, empowers developers with efficient tools for image processing and computer vision tasks, facilitating applications ranging from facial recognition to object detection.

Layers:

1_ Conv2D, the visual detective of deep learning, unveils patterns hidden within images. Like a magnifying lens gliding across a canvas, it meticulously scans for meaningful features, creating a symphony of activations. It's a translation master, expertly transforming raw pixels into a language machines can comprehend. Its

carefully crafted and honed filters act as discerning eyes, extracting the essence of visual information. In the realm of computer vision, Conv2D reigns supreme, empowering machines to perceive the world with clarity and depth

2_ Max Pooling, the artful visual data curator, distills images' essence gracefully and efficiently. Like a seasoned editor trimming a manuscript, it strategically selects only the most striking features, reducing complexity without sacrificing meaning. Acting as a spotlight, it directs attention to the most salient elements, enhancing their impact. It's a master of compression, condensing visual information while preserving its essence, allowing for deeper insights and faster processing. In the grand exhibition of deep learning, Max Pooling2D stands as a vigilant curator, ensuring every detail shines through.

3_ Flattening, uniform in dimensions, breaks down the walls that separate image data. Imagine a tall skyscraper turned into a single busy street. Flatten takes the complex grid of pixels in an image and collapses it into a simplified vector, a single elongated list. This transformation paves the way for deeper analysis, allowing complex neural networks to process the image, extracting higher-level features and relationships. It's like handing the AI a diagram rather than a room-by-room model, allowing for a more comprehensive understanding of the essence of the image. In the grand orchestra of deep learning, Flaten acts as a conductor, uniting disparate elements into a harmonious melody for the neural network to play.

4_ Dense, the indomitable weaver of neural networks, weaves intricate tapestries of knowledge within the mind of a machine. Like threads interlacing through a loom, it forms dense connections between neurons, forging paths for information to flow and patterns to emerge. Each neuron, a vibrant node of understanding, whispers secrets to those it touches, creating a symphony of shared insights. Through this dense symphony, machines grasp the interconnectedness of ideas, enabling them to decode the world's complexities and make reasoned decisions. Dense is the heart of neural networks, where knowledge is woven, insights are forged, and understanding blooms.

5_ Dropout, the artistic saboteur of deep learning, injects a touch of chaos to unleash the true potential of neural networks. Imagine a sculptor chipping away at marble, not to remove, but to refine. Dropout randomly silences neurons during training, forcing the network to rely on its remaining ensemble. This artistic sabotage, as counterintuitive as it may seem, strengthens the overall model. By preventing individual neurons from becoming overly reliant on specific features, Dropout encourages redundancy and fosters a more robust understanding of the data. It's like training a team of artists, not just individual stars, ensuring that the final masterpiece transcends the limitations of any single brushstroke. In the grand canvas of deep

learning, Dropout stands as a bold innovator, adding a touch of serendipity to unveil the hidden beauty within the data.

6_The "optimizer=Adam" in model compilation specifies the use of the Adam optimization algorithm during training. Adam adapts the learning rates for each parameter individually, combining the benefits of both AdaGrad and RMSProp, making it effective for a wide range of deep learning tasks. Its adaptive nature helps converge faster and handle sparse gradients efficiently.

7_ The "Total params" represent the total number of parameters in the neural network model. In this case, there are 76,092,966 parameters. These parameters include both trainable and non-trainable ones. "Trainable params" indicate the number of parameters that will be updated and optimized during the training process. In contrast "Non-trainable params" are fixed and remain constant throughout training, often associated with pre-trained layers or fixed configurations.

```
from tensorflow.keras.optimizers import Adam

model = keras.Sequential()

model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same",input_shape=(256,256,3)))
model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))

model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(1568,activation="relu"))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(38,activation="softmax"))

model.compile(optimizer=Adam(learning_rate=0.0001), loss="sparse_categorical_crossentropy", metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 256, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 256, 256, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 85, 85, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 85, 85, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 28, 28, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 9, 9, 128)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 9, 9, 256)         295168
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 9, 9, 256)         590080
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 9, 9, 512)         3277312
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 9, 9, 512)         6554112
_________________________________________________________________
flatten (Flatten)            (None, 41472)             0
_________________________________________________________________
dense (Dense)                (None, 1568)              65029664
_________________________________________________________________
dropout (Dropout)            (None, 1568)              0
_________________________________________________________________
dense_1 (Dense)              (None, 38)                59622
=================================================================
Total params: 76,092,966
Trainable params: 76,092,966
Non-trainable params: 0
```

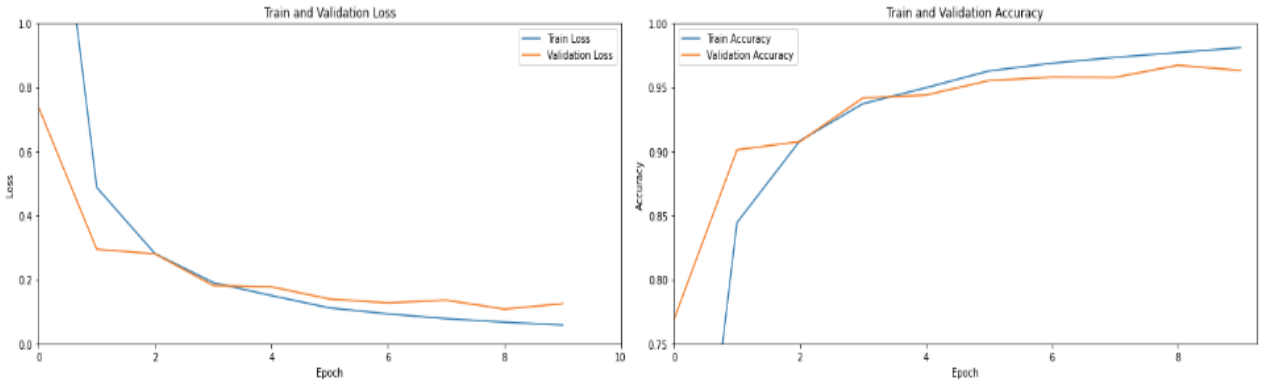
Training data & validation data:

Train: This is the largest dataset used to teach the model. Think of it as your textbook and practice problems. The model analyzes the data, identifies patterns, and adjusts its internal parameters to capture those patterns. The better the training data, the more accurate and generalizable the model's predictions will be.

Valid: This is a smaller dataset used to monitor the model's progress during training. It's like taking periodic progress quizzes. You don't use the valid data to train the model itself, but instead, you measure the model's performance on this unseen data to assess how well it's learning and identify areas for improvement. This helps prevent overfitting (memorizing the training data without understanding the underlying patterns) and ensures the model performs well on real-world data.

Metrics:

```
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
plt.title("Train and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.plot(history.history['loss'],label="Train Loss")
plt.plot(history.history['val_loss'], label="Validation Loss")
plt.xlim(0, 10)
plt.ylim(0.0,1.0)
plt.legend()
print("**********************************************************************")
plt.subplot(1,2,2)
plt.title("Train and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.plot(history.history['accuracy'], label="Train Accuracy")
plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
plt.xlim(0, 9.25)
plt.ylim(0.75,1.0)
plt.legend()
plt.tight_layout()
```



Accuracy:

Accuracy shows how many times the model's predictions hit the bullseye - the number of targets it succeeded in hitting. Simply put, it's the percentage of times the model succeeds in making all of its guesses. Imagine throwing darts: accuracy counts the number of lands that hit the center, not just those near the edge. Although useful, it can be misleading if the classes are unbalanced, so other metrics such as recall or precision are often used alongside it.

- Measures correct predictions as a percentage.
- Higher accuracy means more aiming points and fewer misses.
- A useful result but use it wisely with other tools

```
print("Train Accuracy  : {:.2f} %".format(history.history['accuracy'][-1]*100))
print("Test Accuracy   : {:.2f} %".format(accuracy_score(labels, predictions) * 100))
print("Precision Score : {:.2f} %".format(precision_score(labels, predictions, average='micro') * 100))
print("Recall Score    : {:.2f} %".format(recall_score(labels, predictions, average='micro') * 100))
```

```
Train Accuracy  : 98.11 %
Test Accuracy   : 96.33 %
Precision Score : 96.33 %
Recall Score    : 96.33 %
```

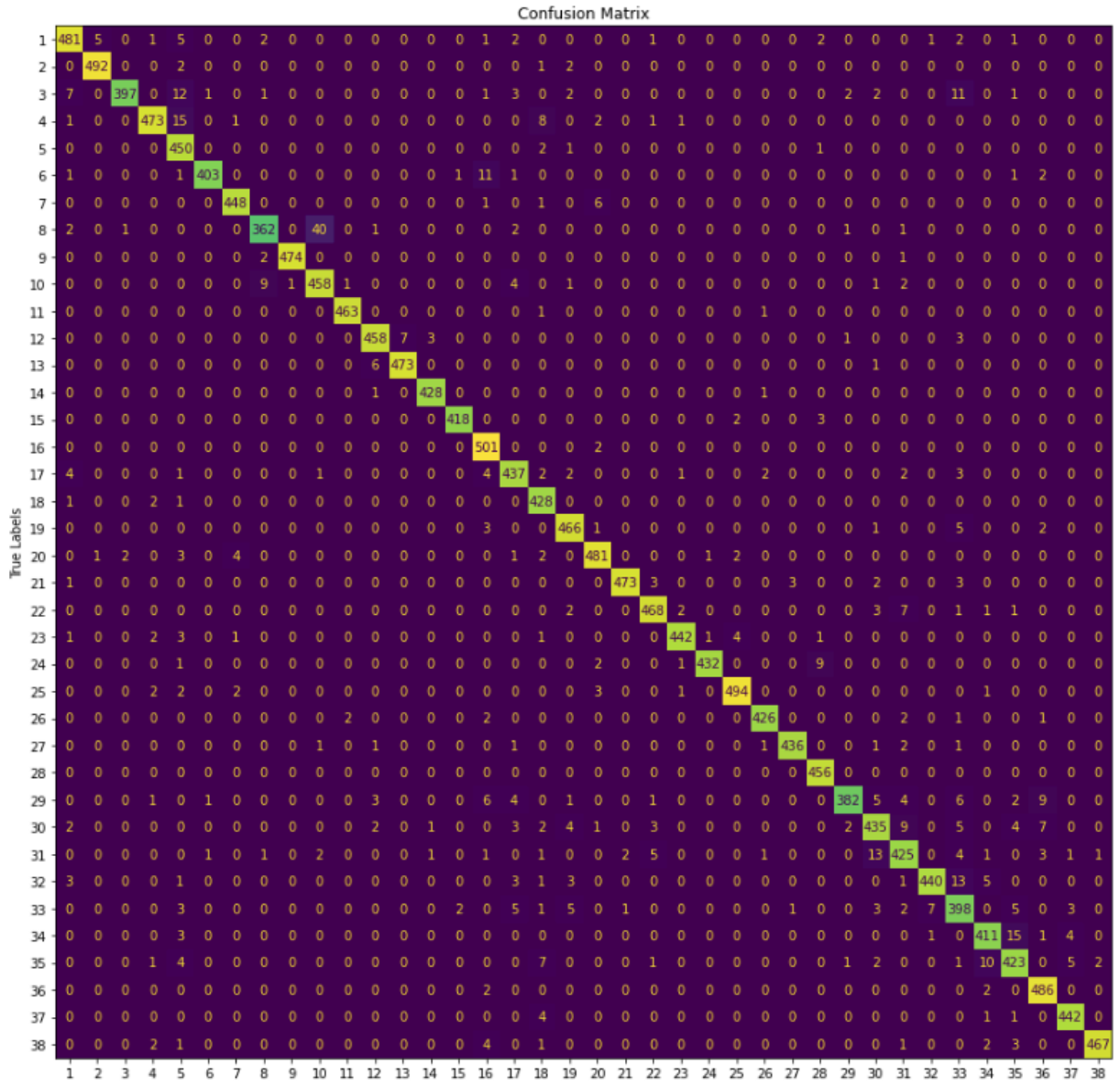
confusion matrix:

A confusion matrix is a table used in machine learning to evaluate the performance of a classification model. It compares the expected and actual classes and provides numbers of true positive, true negative, false positive, and false negative cases. This matrix provides insight into the model's accuracy, precision, recall, and overall performance, which helps in evaluating its effectiveness in dealing with different categories.

Confusion Matrix

```
plt.figure(figsize= (20,5))
cm = confusion_matrix(labels, predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=list(range(1,39)))
fig, ax = plt.subplots(figsize=(15,15))
disp.plot(ax=ax,colorbar= False)
plt.title("Confusion Matrix")
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

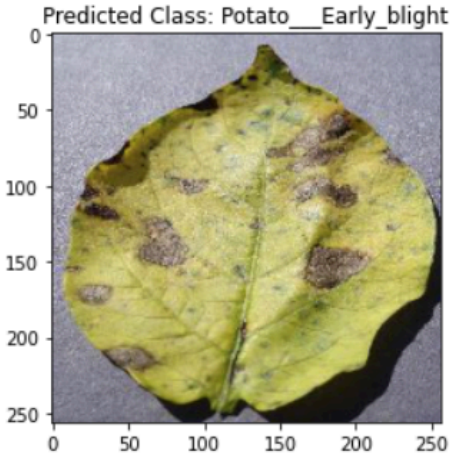
```
<Figure size 1440x360 with 0 Axes>
```



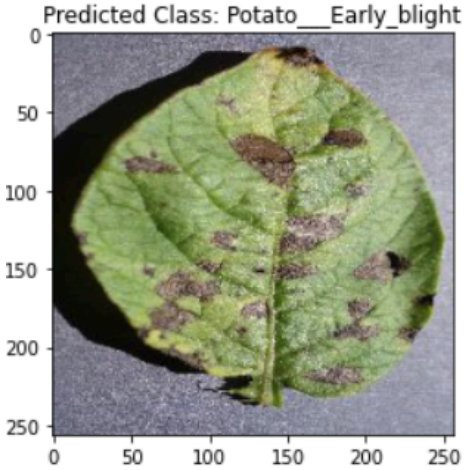
Test:

Testing a trained model involves assessing its performance on a separate dataset (the test set) to ensure generalization and identify issues like overfitting. By comparing predictions to actual outcomes and analyzing metrics such as accuracy, the model's effectiveness in making accurate predictions on new data is validated.

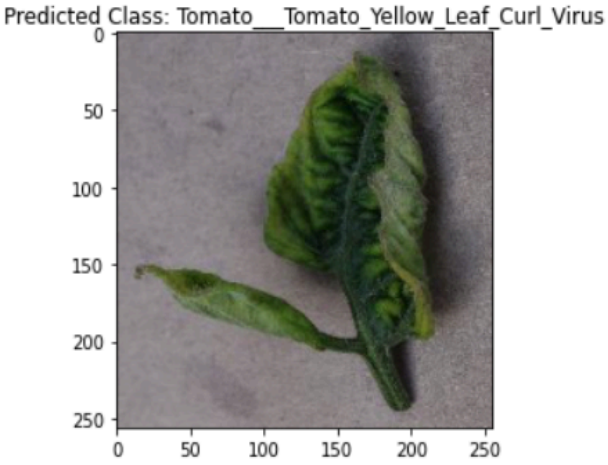
```
image_path = '/kaggle/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight4.JPG'
predicted_class = test_model(model, image_path, class_names)
```



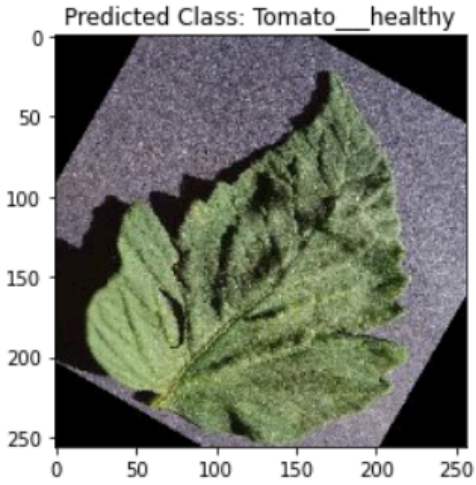

```
image_path = '/kaggle/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight2.JPG'
predicted_class = test_model(model, image_path, class_names)
```



```
image_path = '/kaggle/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus3.JPG'
predicted_class = test_model(model, image_path, class_names)
```




```
image_path = '/kaggle/input/new-plant-diseases-dataset/test/test/TomatoHealthy3.JPG'
predicted_class = test_model(model, image_path, class_names)
```



3.2 API “Flask”:

Flask, a lightweight Python web framework, acts as a seamless bridge between AI and the backend, facilitating swift API development for efficient communication. Its minimalistic design and RESTful capabilities make it a pragmatic choice for seamlessly integrating AI functionalities into the broader application

ecosystem.


File Edit Selection View Go Run Terminal Help
main.py
main.py > classify
from tensorflow import keras,argmax,convert_to_tensor
from flask import Flask, request, jsonify
from PIL import Image
import numpy as np
import io
import base64
# Load the model
model = keras.models.load_model('plant.h5')
class_names=[···
# Define the classification function
def classify_image(image_data):
# Decode the base64 encoded image data
image_data = base64.b64decode(image_data)
# Create a PIL Image from the image data
image = Image.open(io.BytesIO(image_data))
# Preprocess the image
image = image.convert("RGB")
image = image.resize((256, 256))
image = np.array(image) / 255.0
# Make the prediction
predictions = model.predict(convert_to_tensor([image]))
predicted_class_index = argmax(predictions, axis=1).numpy()[0]
predicted_class_label = class_names[predicted_class_index]
return predicted_class_label
# Define the Flask application
app = Flask(__name__)
@app.route('/classify', methods=['POST'])
def classify():
# Get the image data from the request
image_data = request.json['image_data']
# Classify the image
predicted_class_label = classify_image(image_data)
# Return the classification result
return jsonify({'predicted_class_label': predicted_class_label})
if __name__ == '__main__':
app.run(host="0.0.0.0")

3.3 Back end:

Our Packages In the Back End:

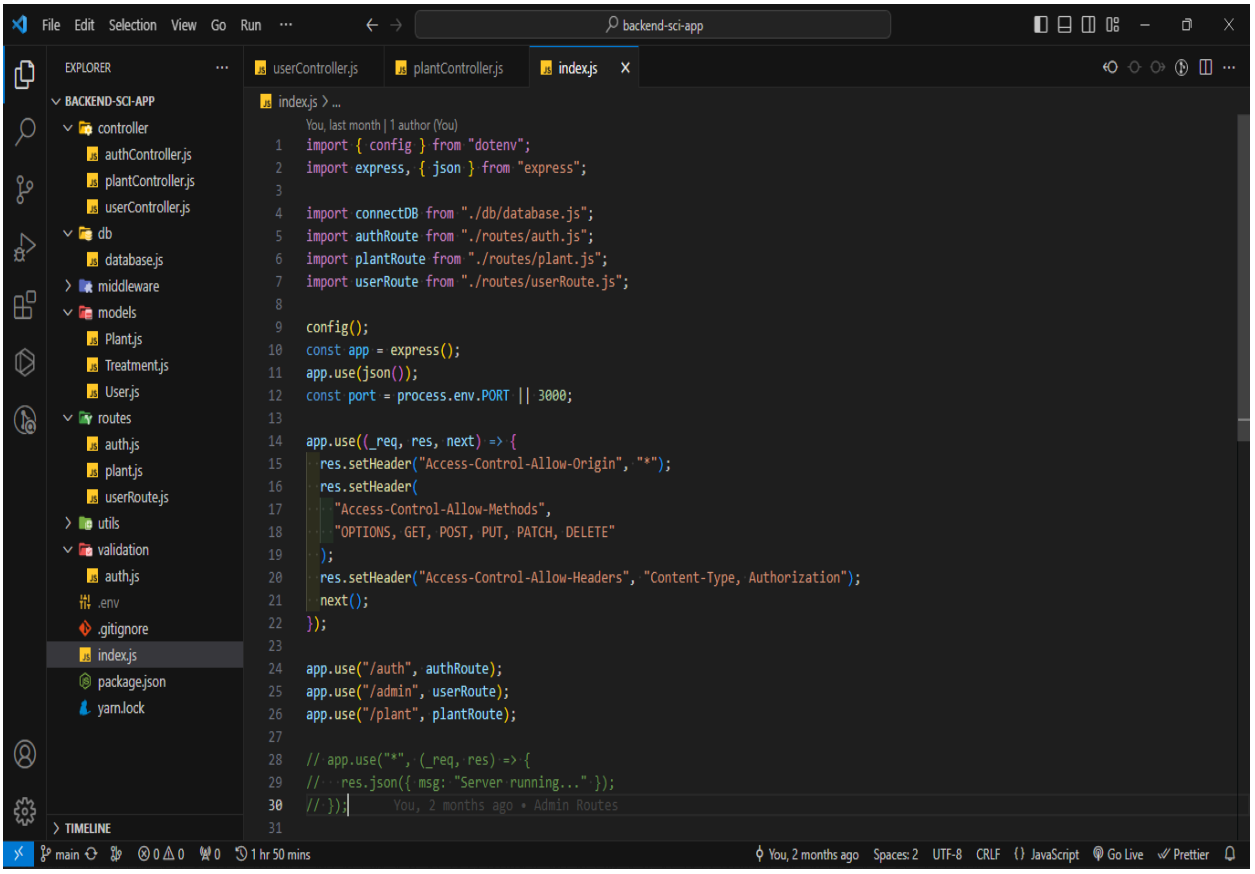
Bcryptjs: A library for hashing passwords using the bcrypt hashing algorithm, enhancing security in web applications.

Express A fast and minimalist web framework for Node.js, facilitating the development of robust and scalable web applications.

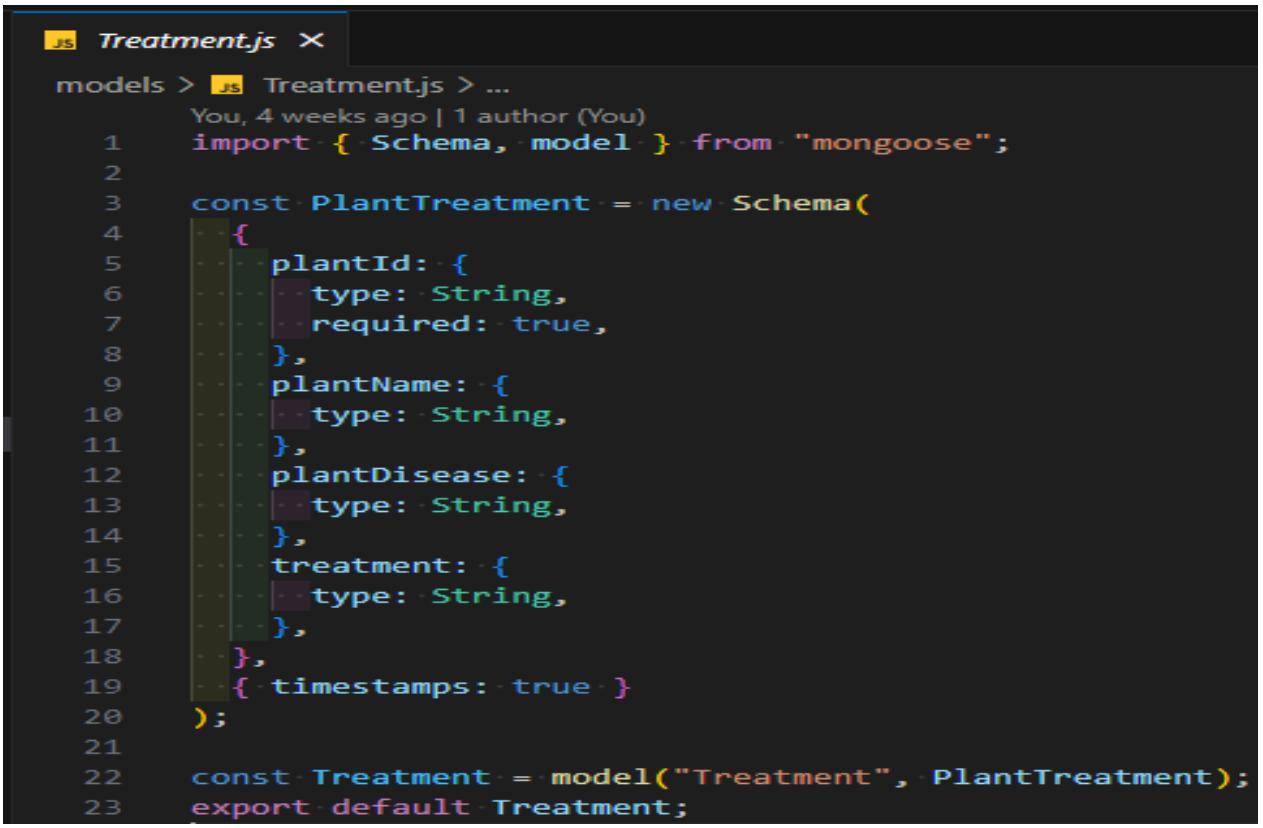
Express Validator: An express middleware that provides validation and sanitation for incoming request data, ensuring data integrity and security.

JSON Web Token: A library for creating and verifying JSON Web Tokens (JWTs), commonly used for secure authentication and data exchange between parties.

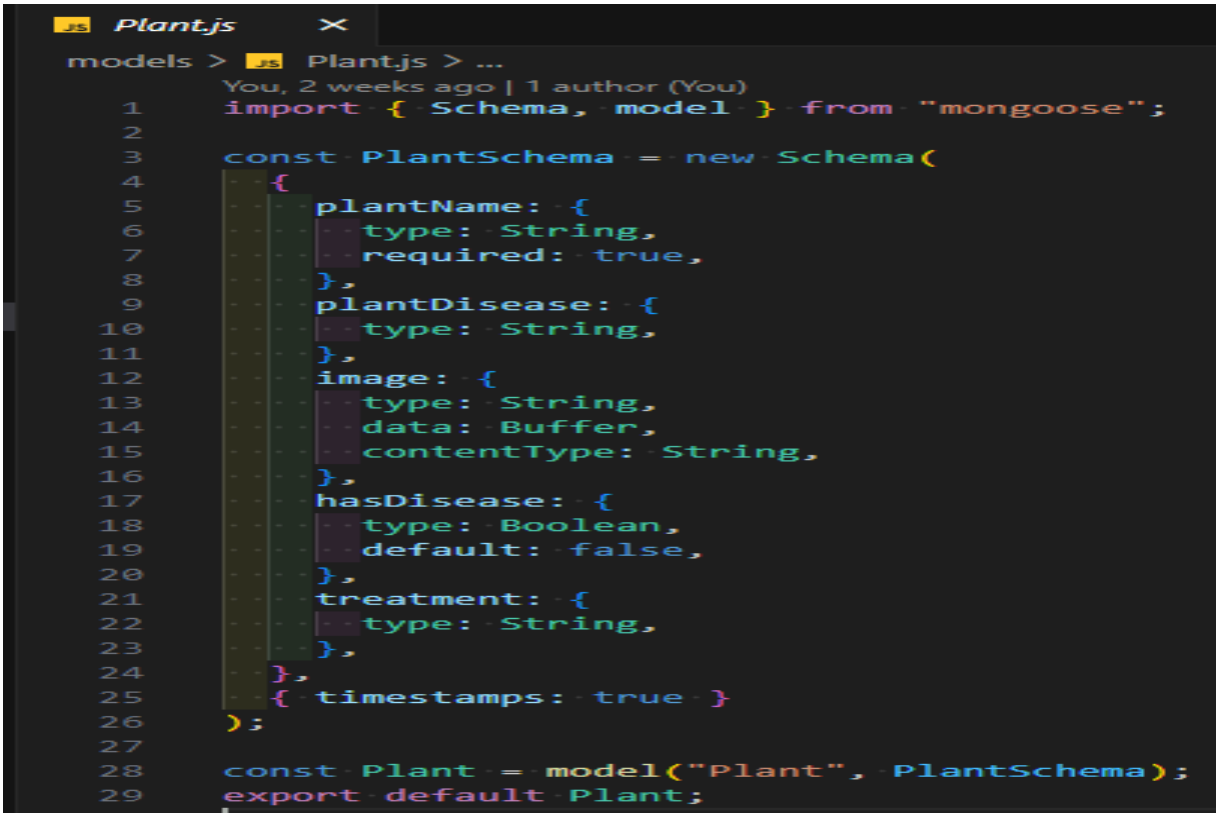
Mongoose: An elegant MongoDB object modeling tool for Node.js, offering a straightforward way to interact with MongoDB databases using JavaScript or TypeScript.



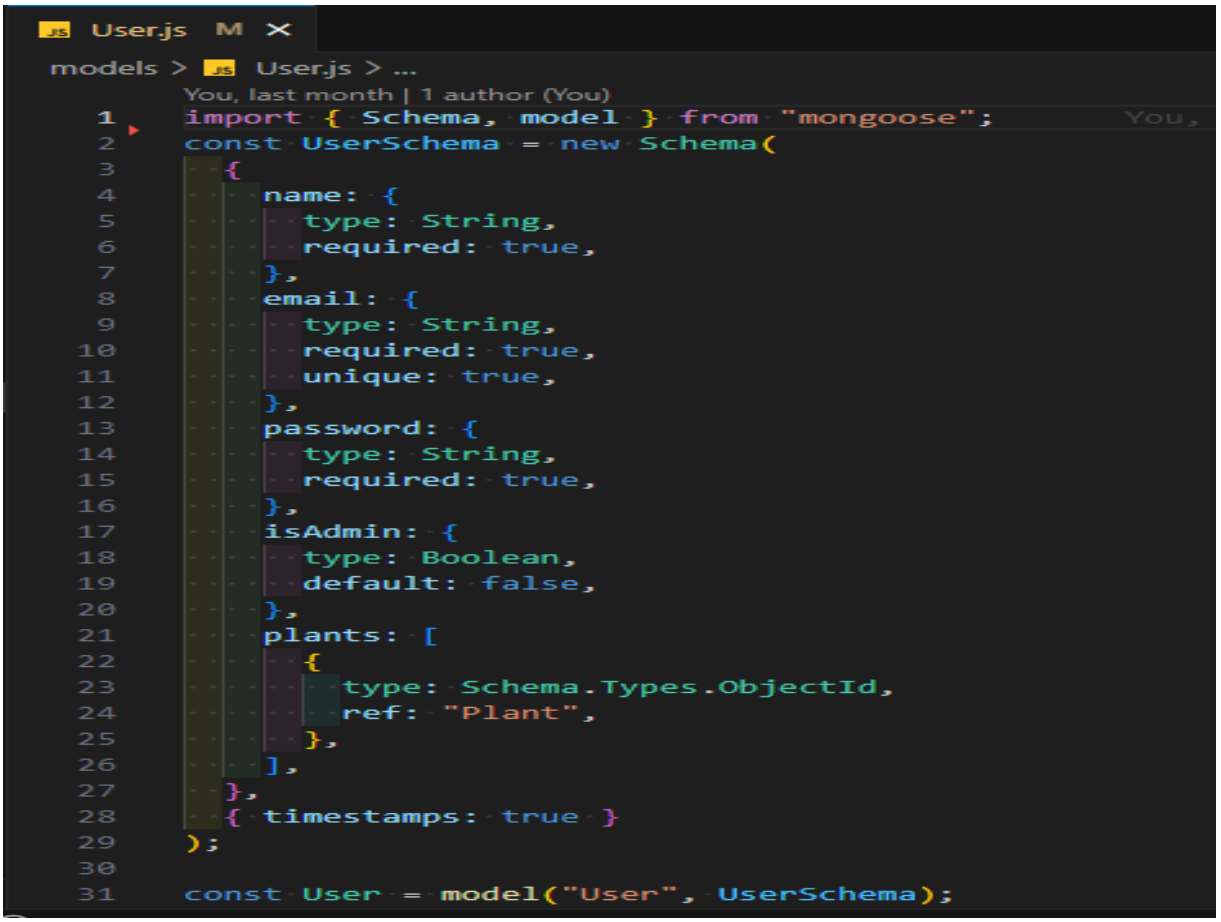
Model treatment:



Model plant:



Model user:



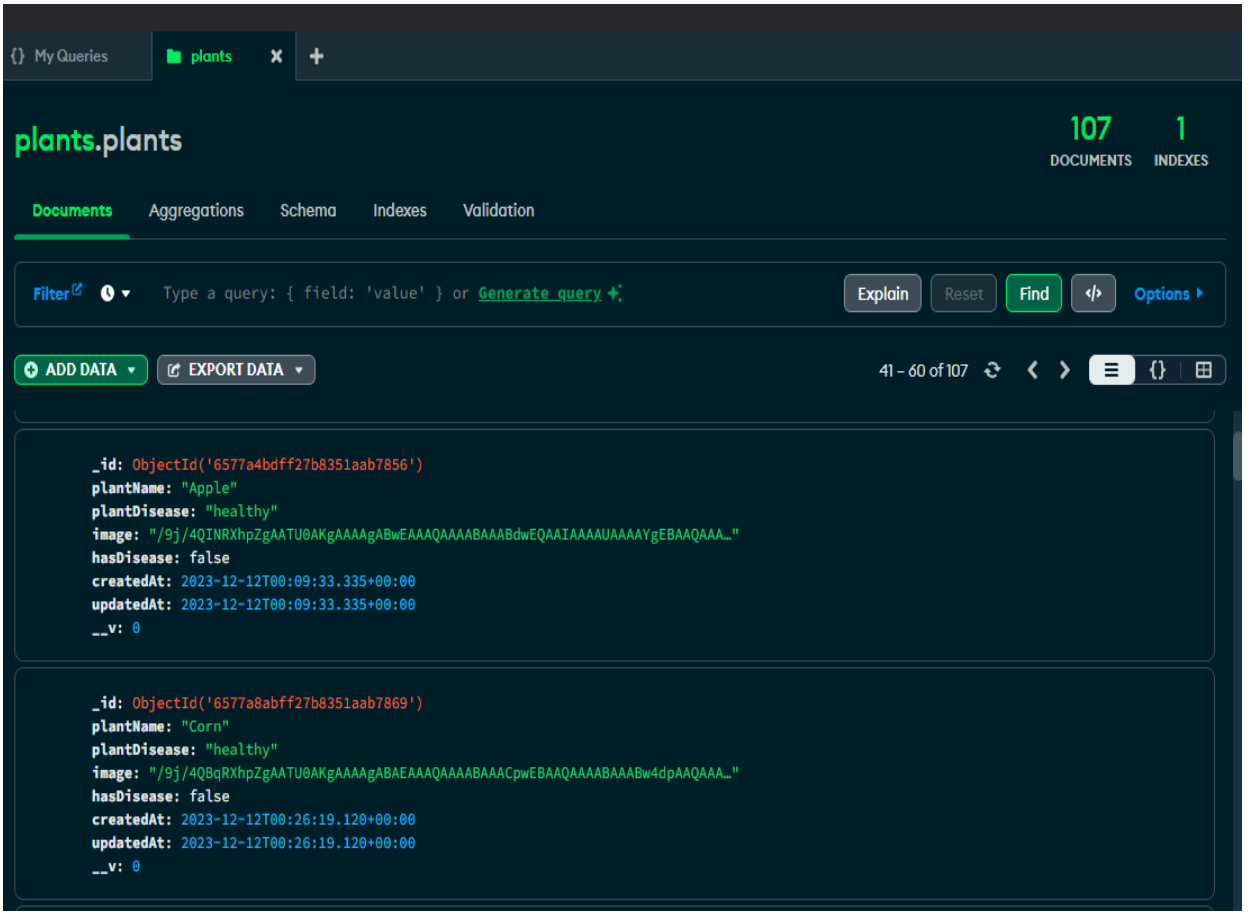
3.4 Database:

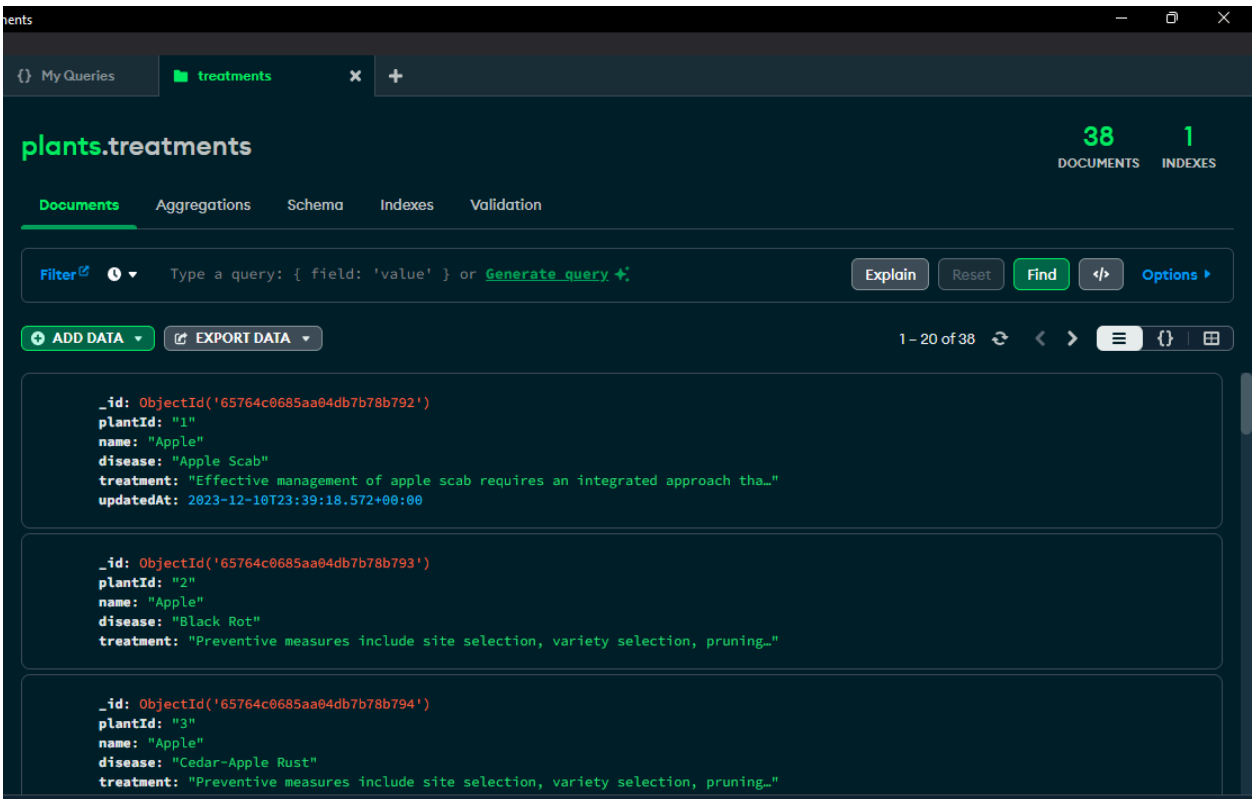
MongoDB:

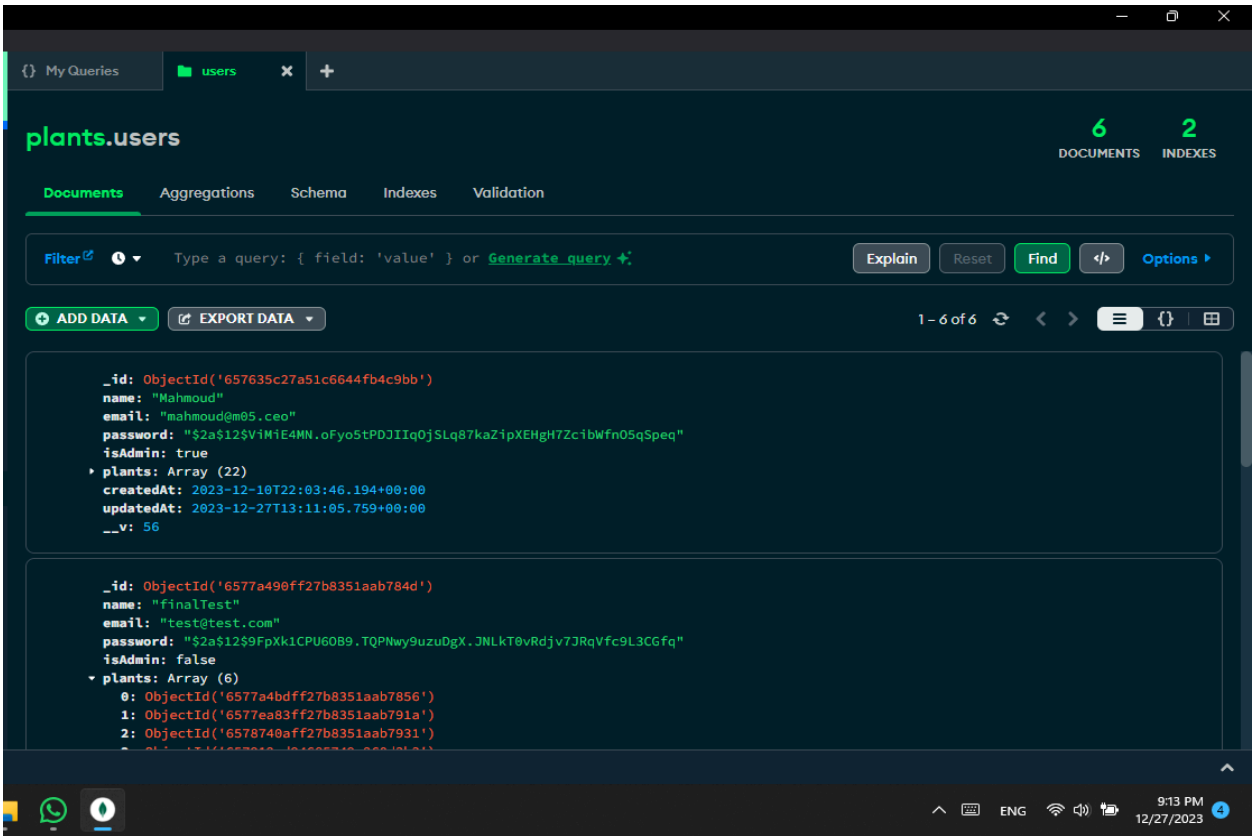
'MongoDB' is a NoSQL database renowned for its flexibility and scalability. It employs a document-oriented data model, storing information in BSON format, a binary representation of JSON-like documents. MongoDB's dynamic schema enables easy adaptation to evolving data structures. The database's horizontal scaling capabilities support the distribution of data across multiple servers, enhancing performance and accommodating growing workloads. Its rich query language and robust indexing contribute to efficient data retrieval. Widely embraced in modern web development, MongoDB facilitates agile handling of diverse and unstructured data, making it a popular choice for applications requiring real-time analytics and dynamic data handling.

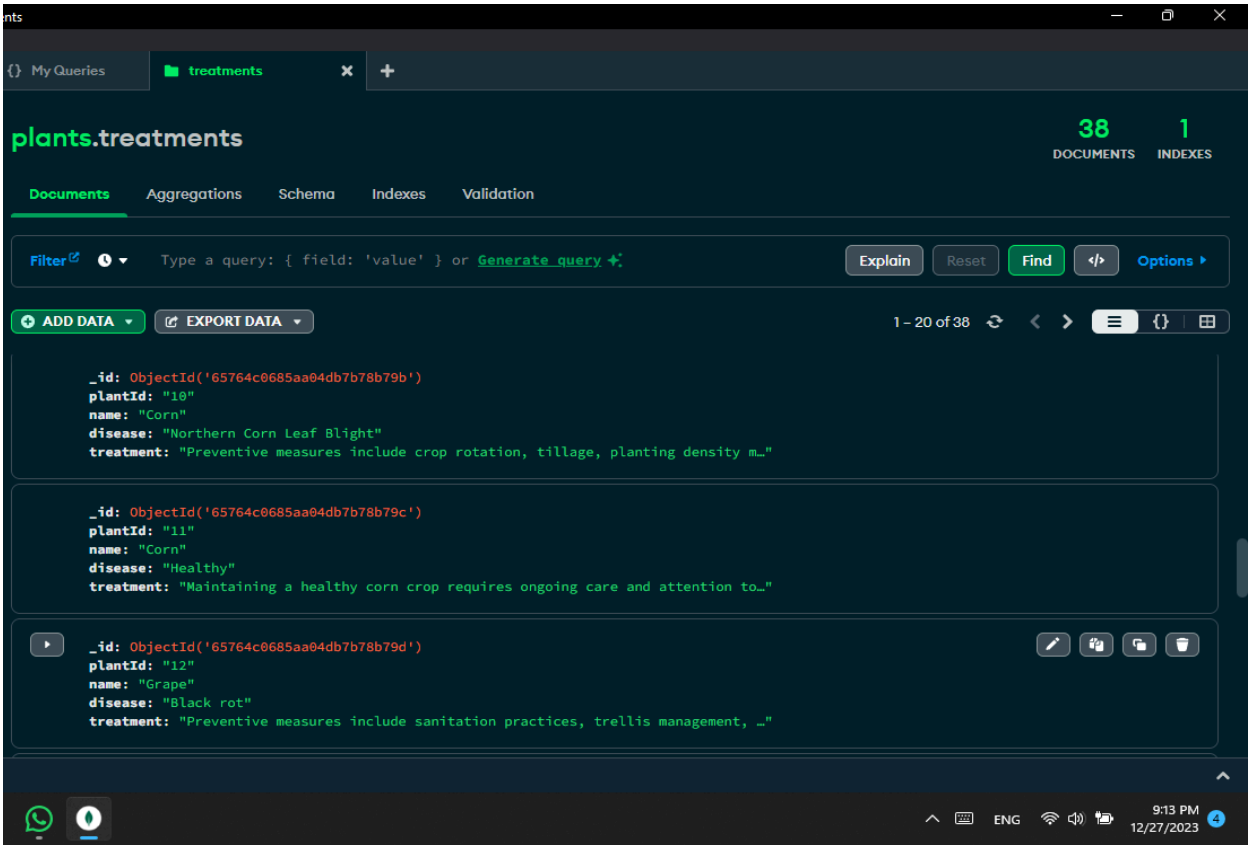
Why to use "MongoDB" as a database:

I choose MongoDB over other database systems for its unmatched flexibility and scalability. MongoDB's document-oriented approach allows me to store and manage data in a way that mirrors the structure of my application, providing a seamless and intuitive experience. The dynamic schema accommodates changes in data requirements, offering adaptability in dynamic development environments. Its support for horizontal scaling ensures efficient handling of increasing workloads and enhances performance by distributing data across multiple servers. The ease of use in querying and indexing, coupled with a robust ecosystem, simplifies complex data retrieval tasks. Overall, MongoDB empowers me to work with diverse and unstructured data in a highly efficient and agile manner, making it the ideal choice for my projects requiring real-time analytics and dynamic data management.

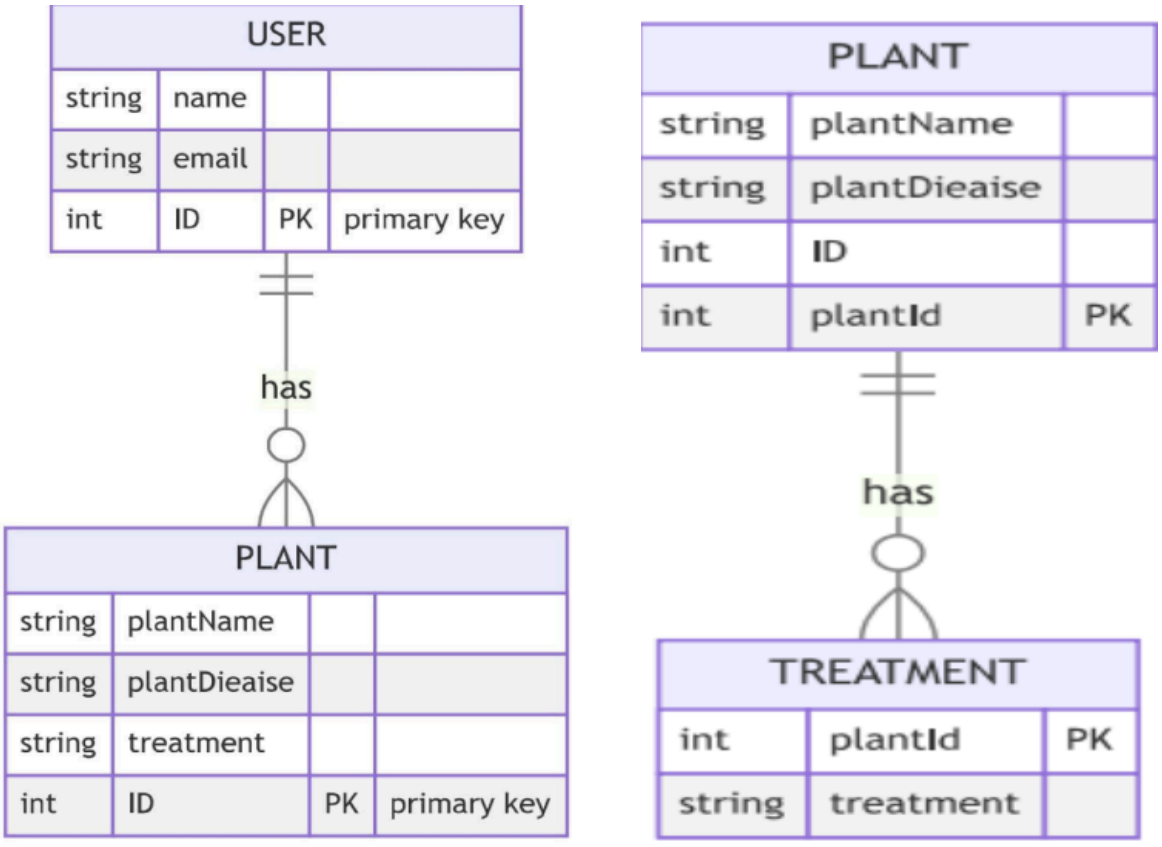



My Queries
treatments
plants.treatments
38
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 20 of 38
_id: ObjectId('65764c0685aa04db7b78b792')
plantId: "1"
name: "Apple"
disease: "Apple Scab"
treatment: "Effective management of apple scab requires an integrated approach tha…"
updatedAt: 2023-12-10T23:39:18.572+00:00
_id: ObjectId('65764c0685aa04db7b78b793')
plantId: "2"
name: "Apple"
disease: "Black Rot"
treatment: "Preventive measures include site selection, variety selection, pruning…"
_id: ObjectId('65764c0685aa04db7b78b794')
plantId: "3"
name: "Apple"
disease: "Cedar-Apple Rust"
treatment: "Preventive measures include site selection, variety selection, pruning…"


My Queries
users
plants.users
6
DOCUMENTS
2
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 6 of 6
_id: ObjectId('657635c27a51c6644fb4c9bb')
name: "Mahmoud"
email: "mahmoud@m05.ceo"
password: "$2a$12$ViMiE4MN.oFyo5tPDJIIqOjSLq87kaZipXEHgH7ZcibWfnO5qSpeq"
isAdmin: true
plants: Array (22)
createdAt: 2023-12-10T22:03:46.194+00:00
updatedAt: 2023-12-27T13:11:05.759+00:00
__v: 56
_id: ObjectId('6577a490ff27b8351aab784d')
name: "finalTest"
email: "test@test.com"
password: "$2a$12$9FpXk1CPU6OB9.TQPNwy9uzuDgX.JNLkT0vRdjv7JRqVfc9L3CGfq"
isAdmin: false
plants: Array (6)
0: ObjectId('6577a4bdff27b8351aab7856')
1: ObjectId('6577ea83ff27b8351aab791a')
2: ObjectId('6578740aff27b8351aab7931')
ENG
9:13 PM
12/27/2023

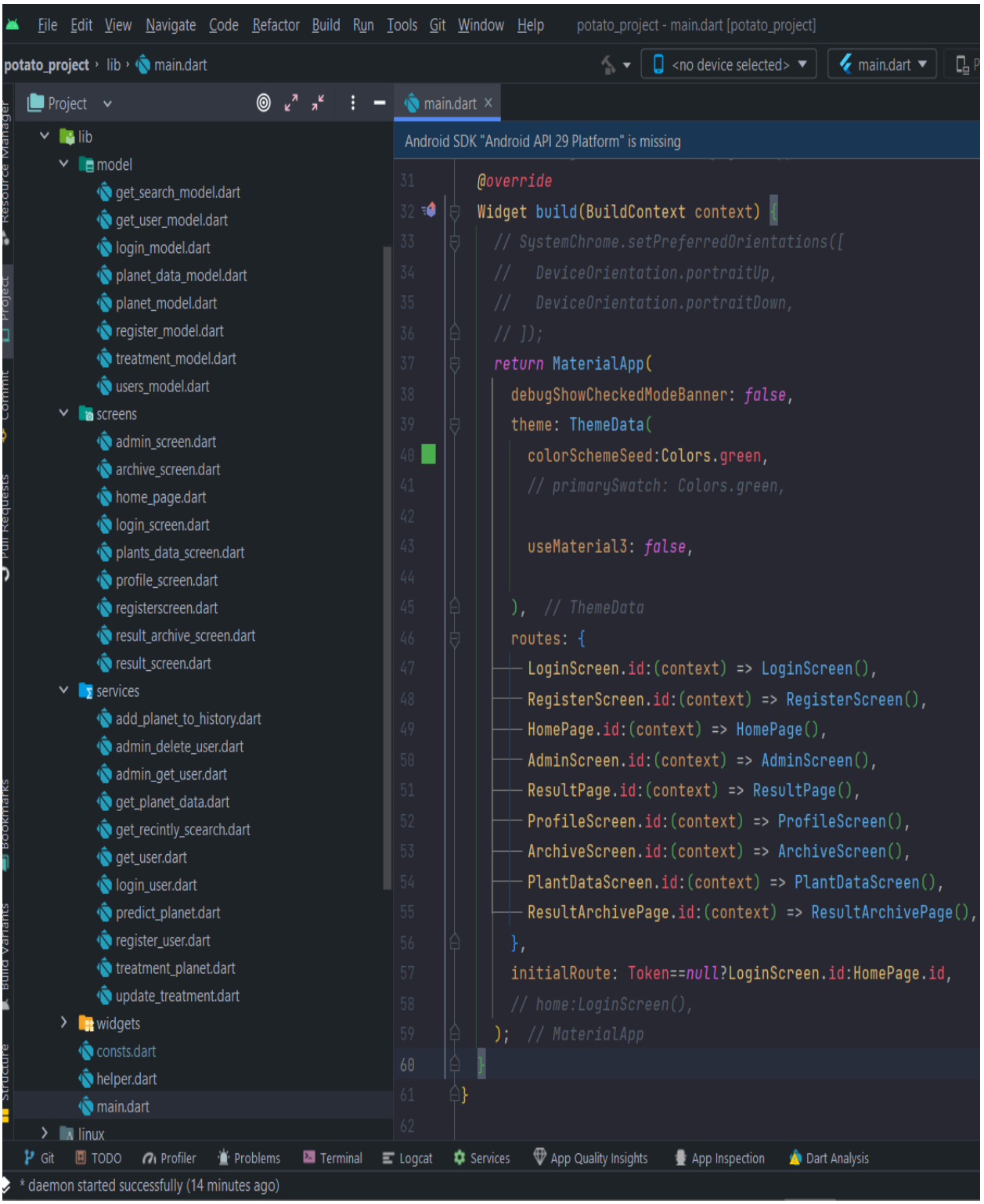


Database Flowchart



3.5 flutter app:

The main page: is the code editor for the Potato Project, The code editor shows the files and folders in the project



Handle API with (Dio):

```dart
import 'package:dio/dio.dart';
import 'package:potato_project/consts.dart';
import 'package:potato_project/model/planet_model.dart';

class PredictPlanet{
  final dio = Dio();
  Future<PlanetModel> predict({required String im64})async{

    final response = await dio.post(
        '$KApi/classify', data:
    {
      "image_data": im64,
    });


    Map<String, dynamic> json = response.data;

    PlanetModel prediction = PlanetModel.fromJson(json);

    return prediction;



  }
}
```

Login Process:

```
CustomButon(
    txt: 'Login',
    onTap: () async {
      FocusManager.instance.primaryFocus?.unfocus();
      if (formKey.currentState!.validate()) {
        isLoading = true;
        setState(() {});
        try {
          // await Login().LoginUser(email: email,password: password);
          var userData = await Login()
              .LoginUser(email: email, password: password);
          bool isAdmin = userData.data!.isAdmin ?? false;
          String token = userData.data!.token ?? " ";
          String id = userData.data!.userId ?? " ";
          SharedPreferences prefs =
              await SharedPreferences.getInstance();
          print(isAdmin);
          prefs.setBool(KisAdmin, isAdmin);
          prefs.setString(KToken, token);
          prefs.setString(KId, id);
          Navigator.pushNamedAndRemoveUntil(
            context,
            HomePage.id,
            (route) => false,
          );
        } on DioException catch (e) {
          loginError errorData =
              loginError.fromJson(e.response!.data);
          String? errorMessage = errorData.data!.msg;
          MessageToUser(
            Message: errorMessage!,
            state: MessageState.ERROR,
            context: context,
          );
        } catch (e) {
          MessageToUser(
            Message: e.toString(),
            state: MessageState.ERROR,
            context: context,
          );
        }
        isLoading = false;
        setState(() {});
      }
    }),  // CustomButon
```

Get an image and classify it Process:

```
Future getImage(ImageSource media) async {
  var img = await picker.pickImage(
      source: media, imageQuality: 50, maxHeight: 190.0, maxWidth: 190.0);
  image = img;
  setState(() {});
  if (image != null) {
    final bytes = io.File(image!.path).readAsBytesSync();
    String img64 = base64Encode(bytes);
    result = null;
    isLoading = true;
    setState(() {});
    try {
      result = await PredictPlanet().predict(im64: img64);
      SharedPreferences prefs = await SharedPreferences.getInstance();
      id = await prefs.getString(KId);
      treatment = await GetTreatment().Treatment(
          userId: id!, plantId: result!.predictedClassLabel.id.toString());
      await AddPlanetData().AddPlanet(
          id: id,
          planetName: result!.predictedClassLabel.planetName,
          planetDisease: result!.predictedClassLabel.planetDisease,
          hasDisease: result!.predictedClassLabel.planetDisease == "healthy"
              ? false
              : true,
          plantId: result!.predictedClassLabel.id.toString(),
          image: img64);
      await Navigator.pushNamed(context, ResultPage.id,
          arguments: [treatment, result]);
      isLoading = false;
      setState(() {});
    } on DioException catch (e) {
      isLoading = false;
      setState(() {});
      MessageToUser(
          Message: e.toString(), context: context, state: MessageState.ERROR);

      print(e.toString());
    }
    setState(() {});
  } else {
    MessageToUser(Message: "Can't pick image!!", context: context);
  }
}
```

Get User Data (Admin) Process:

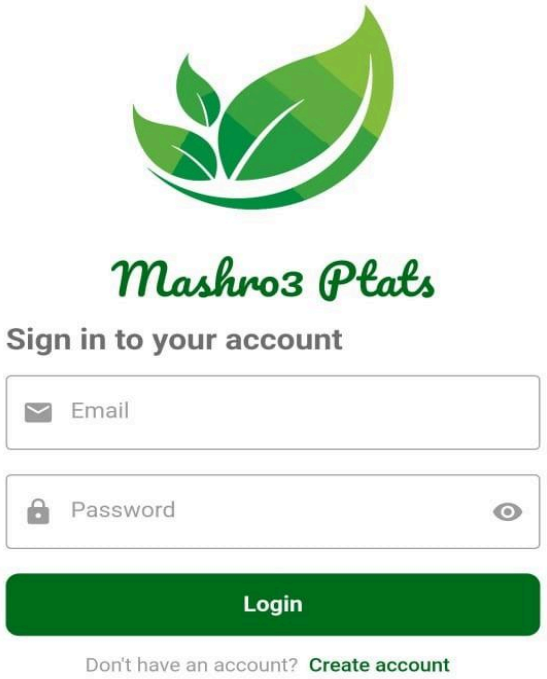
```
Visibility(
  visible: isAdmin ?? false,
  child: IconButton(
      onPressed: () async {
        isLoading = true;
        setState(() {});
        try {
          var Users =
              await GetUserData().UsersList(id: id, token: token);

          Navigator.pushNamed(context, AdminScreen.id,
              arguments: Users);
        } on DioException catch (e) {
          var errorMessage = e.response!.data['msg'];

          MessageToUser(
              Message: errorMessage,
              context: context,
              state: MessageState.ERROR);
        } catch (e) {
          MessageToUser(
              Message: e.toString(),
              context: context,
              state: MessageState.ERROR);
        }
        isLoading = false;
        setState(() {});
      },
      icon: Image.asset("assets/images/icon_admin.png")),  // IconButton
),  // Visibility
```

Treatment Update (Admin) Process:

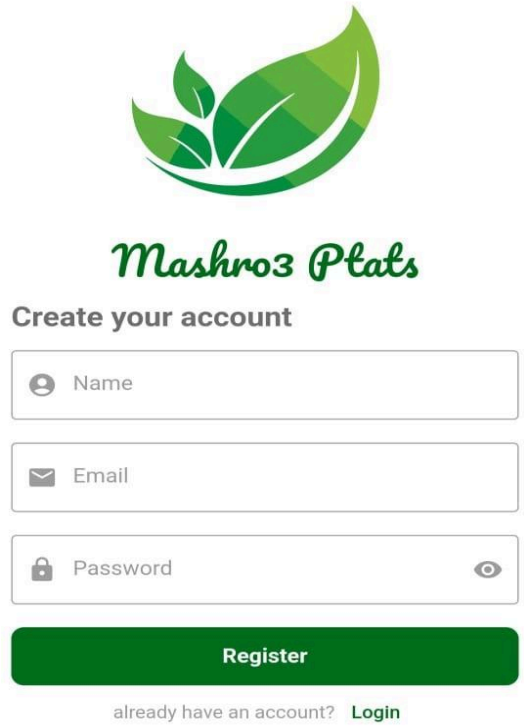
```
onPressed: () async {
  isLoadingAlert = true;
  setState(() {});
  try {
    await UpdateTreatment().Update(
        token: token,
        id: id,
        plantId: users.data[index].id.toString(),
        treatment: treatmentControler.text);
    isLoadingAlert = false;
    setState(() {});
    Navigator.of(context).pop();
  } catch (e) {
    isLoadingAlert = false;
    setState(() {});
    MessageToUser(
        Message: e.toString(),
        context: context,
        state: MessageState.ERROR);
```

3.6 Output:

- login screen, prompting users to enter their email and password.

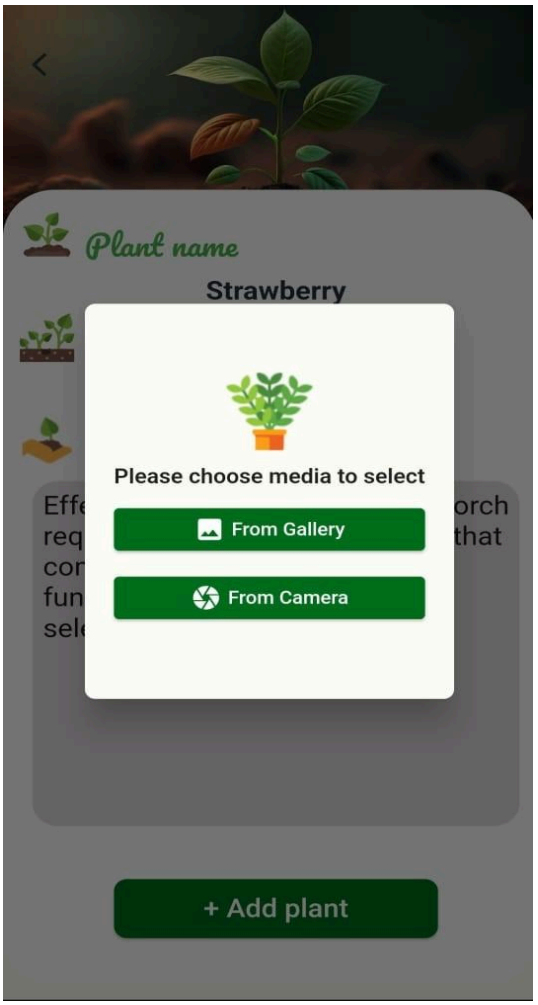


- sign-up page, prompting users to create a new account.



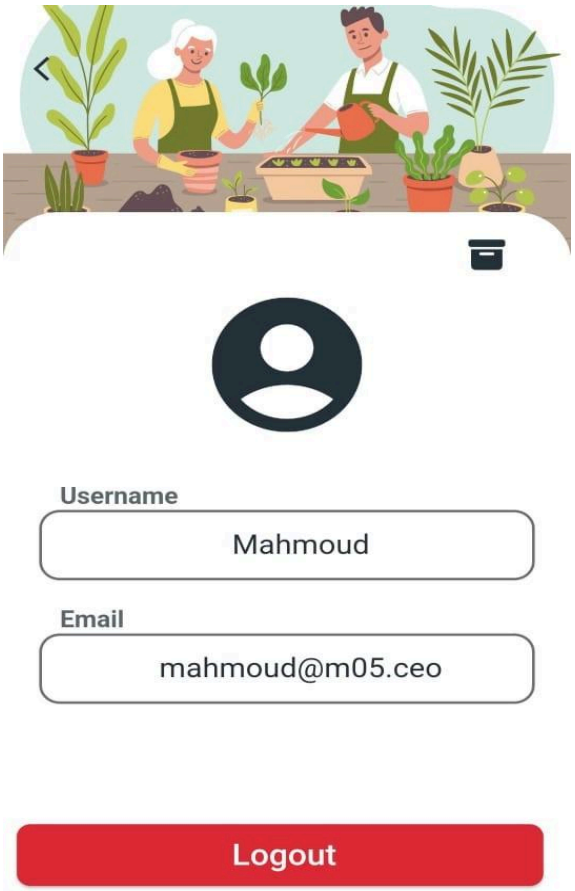
- The app's welcome screen invites users to get started, with the first image of a plant

- Add a photo of the plant with your camera or from the gallery.

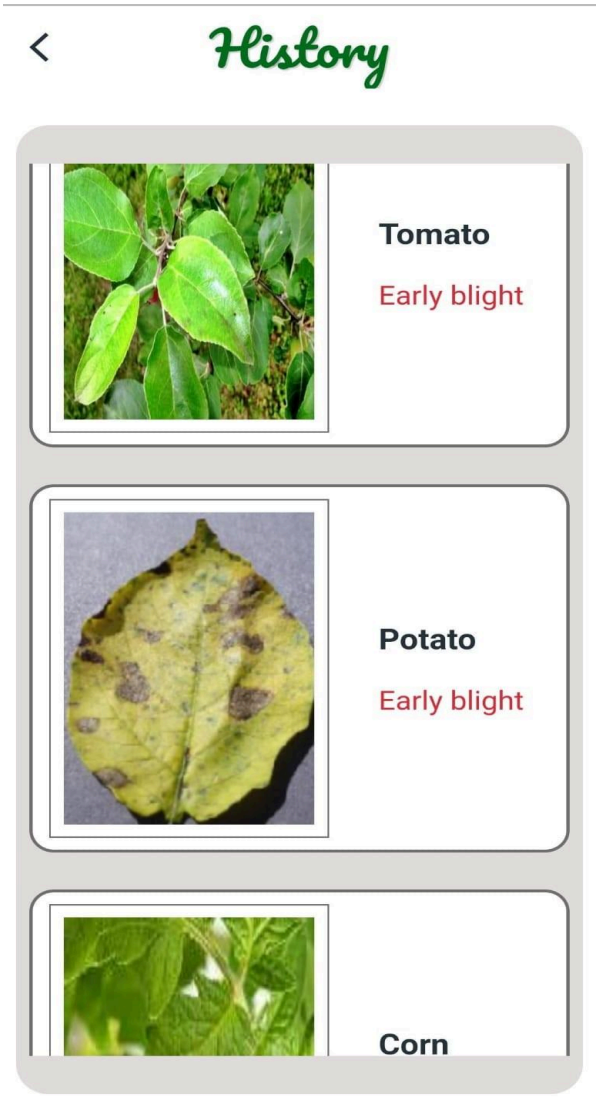


- The screen shows a strawberry plant that has been added to the application and analysis.

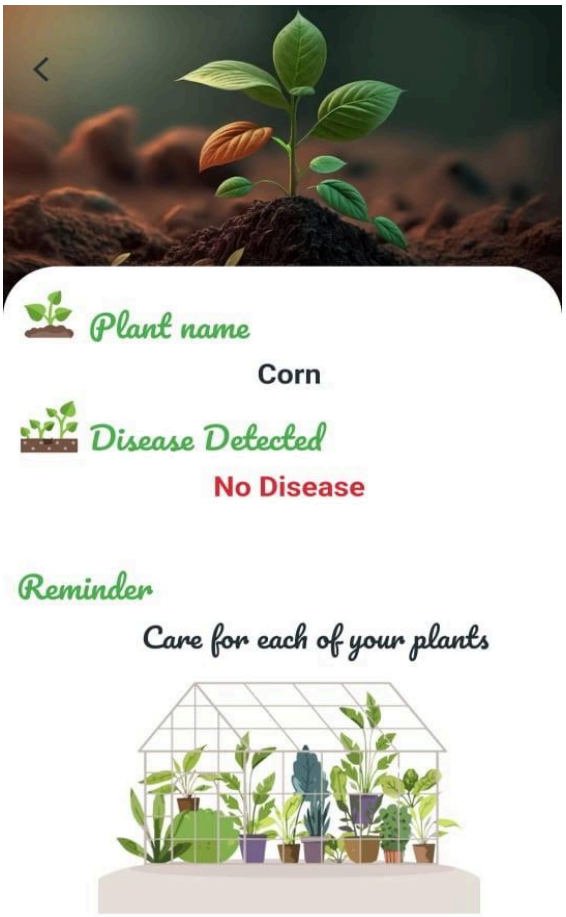
- To log out of the used account.



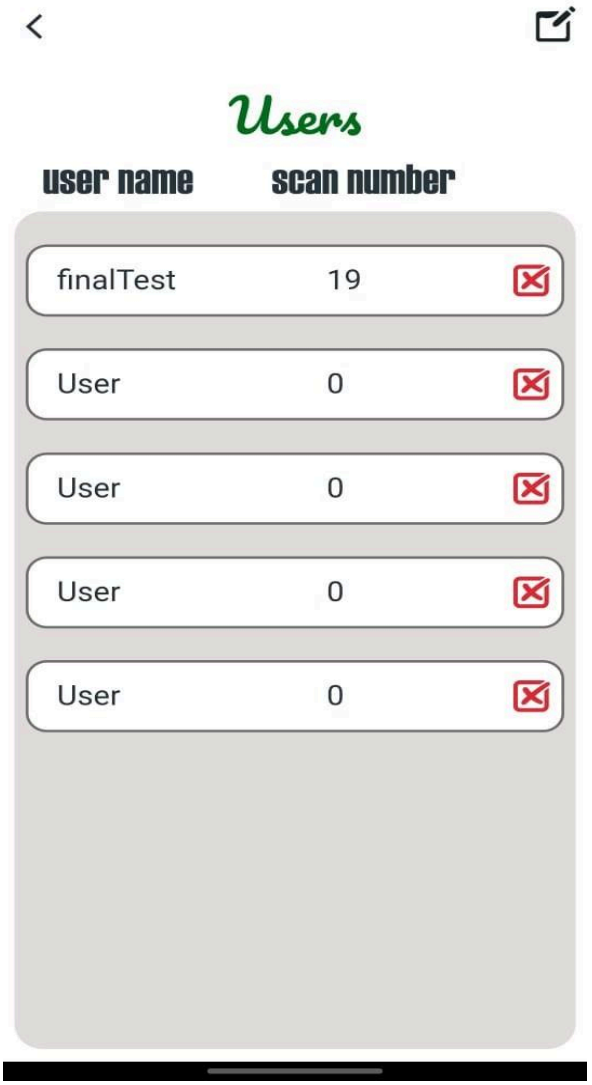
- Record all pictures of the previous plants and operations that he examined for the user.



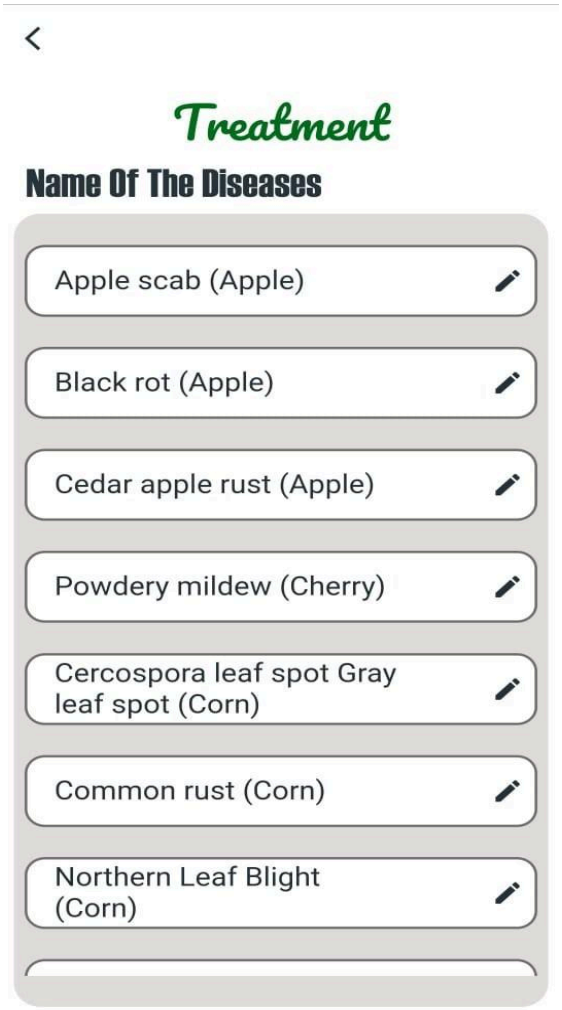
- The screen shows a corn plant after analysis.



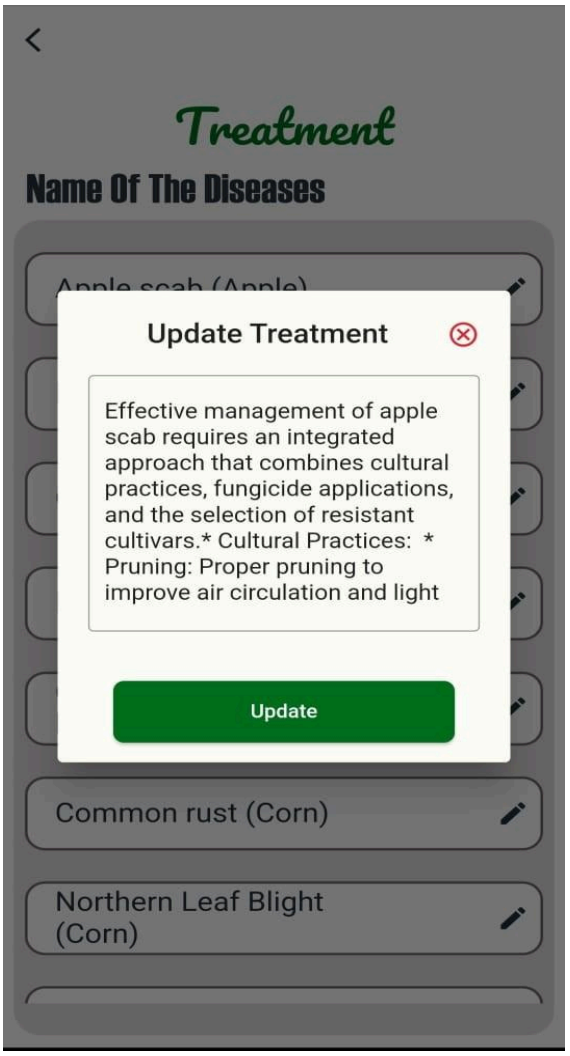
- Register for all user accounts.



- List of plants and their diseases.



- Modifying the treatment of diseases is subject to the admin’s ability.



4 CONCLUSION

This groundbreaking application is poised to transform plant disease identification and treatment, employing advanced AI algorithms for swift and accurate diagnoses. Users stand to benefit significantly from personalized treatment solutions based on real-time data analysis, optimizing plant disease management strategies. The application's contribution extends to the realm of agriculture, enabling farmers and professionals to proactively address disease threats and enhance overall crop health. With a user-friendly interface, individuals of varying technical expertise can effortlessly navigate the application, ensuring quick access to vital information. Beyond individual users, the application's overarching goal is to contribute to global food security by empowering users with effective tools for sustainable plant disease management, ultimately playing a pivotal role in sustaining agricultural productivity.

5 REFERENCES

1. Comparison method for classifying bare PCB defects. Available at: CNN-based reference comparison method for classifying bare PCB defects - Wei - 2018 - The Journal of Engineering - Wiley Online Library

2. Improving Accuracy and Efficiency through AutoML and Model Scaling. Available at: https://blog.research.google/2019/05/efficientnet-improving-accuracy-and.html?m=1

3. TensorFlow Tutorials - Convolutional Neural Network (CNN). Available at: https://www.tensorflow.org/tutorials/images/cnn

4. Hello World, GitHub Guides: https://docs.github.com/en/get-started/quickstart/hello-world

5. Fei-Fei Li and Andrej Karpathy, "Convolutional Neural Networks for Visual Recognition"، 2021, Stanford University.

6. Ian Goodfellow, Yoshua Bengio, and Aaron Courville, "Deep Learning"، 2016, MIT Press.

7. Node.js Official Documentation. Available at: https://nodejs.org/en/docs/

8. Node.js quickstart. Available at: https://developers.google.com/sheets/api/quickstart/nodejs

9. Mario Casciaro, "Node.js Design Patterns", 2014, Packt Publishing.

10. Kyle Simpson, "You Don't Know JS: Async & Performance", 2015, O'Reilly Media.

11. MongoDB Official Documentation. Available at: https://docs.mongodb.com/

12. MongoDB Relationships (Embedded & Reference) – Database References. Available at: _https://data-flair.training/blogs/mongodb-relationships/

13. Kyle Banker, "MongoDB in Action"، 2011, Manning Publications.

14. Alok Kumar, "Learning MongoDB: Schema, Queries, and Application Design"، 2013, Packt Publishing.

15. Flask Official Documentation. Available at: https://flask.palletsprojects.com/

16. RESTful API with Flask by Miguel Grinberg. Available at: https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

17. Jose Salvatierra, "REST APIs with Flask and Python"، 2019, Independently published

18. Flutter Documentation. Available at: https://flutter.dev/docs

19. Flutter SDK. Available at: Flutter SDK reference | Harness Developer Hub

20. Dart Language Tour. Available at:https://dart.dev/guides/language/language-tour

21. Eric Windmill, "Flutter in Action"، 2019, Manning Publications

22. Packt Publishing, "Google Flutter Mobile Development Quick Start Guide"، 2019, Packt Publishing

23. Ivo Balbaert, "Learning Dart - Second Edition"، 2014, Packt Publishing

24. Adobe XD Official Website and Documentation. Available at: https://www.adobe.com/products/xd.html

25. Cindy Snyder Dionisio, "Mastering Adobe XD: Design, Prototype, and Share Engaging User Experiences"، 2018, Apress