PLANT DISEASE MANAGEMENT

A group project report submitted in partial fulfillment of the requirements for

The Award of the Degree of

Bachelor of Science (Hons.)

in

Computer Science

Supervisor:-

Dr. Anshul Verma

(Assistant Professor)

Banaras Hindu University

Submitted from :-

Suraj Kumar Yadav(20220CMP032)

Abhishek Rao(20220CMP001)

Shashwat Dwivedi(20220CMP013)

Pushkar Kumar(20220CMP022)

Department of Computer Science

Institute of Science

Banaras Hindu University, Varanasi – 221005

2023

CERTIFICATE

This is to certify that **Suraj Kumar Yadav**(20220CMP032), **Abhishek Rao**(20220CMP001), **Shashwat Dwivedi**(20220CMP013) and **Pushkar Kumar**(20220CMP022) have certified out the project on the topic **"Plant Disease Detection"** during academic session 2018-2019 as Partial Fulfillment of the Requirements for The Award of the Degree of **Bachelor of Science (Hons.) in Computer Science.**

As far as known to me, the work reported in this project report has not been submitted to any other institute/university for the award of any degree/diploma.

Head of the Department

Department of Computer Science

Institute of Science

Banaras Hindu University

CANDIDATE'S DECLARATION

We **Suraj Kumar Yadav, Abhishek Rao, Shashwat Dwivedi** and **Pushkar Kumar** hereby certify that the work, which is being presented in the project report, entitled **Plant Disease Management**, in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Science (Hons.)** and submitted to the institution is an authentic record of my own work carried out during the period Jan-2019 to May-2019 under the supervision of **Dr. Anshul Verma**. We also cited the reference about the text(s) /figure(s) from where they have been taken.

The matter presented in this report has not been submitted elsewhere for the award of any other degree or diploma from any Institutions.

Date: **Signature of the Candidate**

Signature of the Candidate

Signature of the Candidate

Signature of the Candidate

TO WHOM IT MAY CONCERN

This is to certify that the candidate has worked under my supervision and the above statement made by the candidate is correct to the best of my knowledge.

Date: **Dr. Anshul Verma**

Assistant Professor

Department of Computer Science

Banaras Hindu University

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We feel beholden to heartily acknowledge our deep sense of obligation to persons and institutions wherefrom we received help during the project.

First of all, we would like to express our sincere gratitude to our learned supervisor Dr. Anshul Verma, a man of broad vision, creative thinking, and a lover of academic pursuit, for rendering valuable guidance throughout the project and constantly motivating us to work harder during the course of the project.

We are grateful to all the faculties who have directly or indirectly helped in the completion of the project. We take this opportunity to express our profound gratitude to all our friends for their support and encouragement that helped us to complete this project.

Last but not least, we express thanks to our family for cooperating and assisting us from time to time, without them this task could not have been completed. We further acknowledge that no perfection can be claimed by any human being. We take responsibility for any error which inadvertently might have crept in.

ABSTRACT

The goal of this project is to develop a web-based plant disease management system that allows users to detect, classify, and receive treatment recommendations for plant diseases using photos of plant leaves. The system leverages machine learning algorithms to analyze the images and provide accurate diagnoses, which can help farmers and gardeners take appropriate measures to protect their crops. In addition, the website features a query section where users can ask questions and receive personalized advice from domain experts. The main contributions of this project include the development of a user-friendly and accessible interface, the implementation of state-of-the-art machine learning models, and the provision of expert support to ensure the accuracy and reliability of the diagnoses. The website has the potential to significantly improve crop yields and reduce the use of harmful pesticides, contributing to more sustainable and environmentally-friendly agriculture.

Chapter 1

INTRODUCTION

1.1 GENERAL:-

Plant diseases are a major challenge for farmers and growers around the world, with the potential to cause significant crop losses and economic damage. Traditional methods of plant disease diagnosis and treatment can be time-consuming, expensive, and reliant on expert knowledge, making them challenging to implement on a large scale.

To address this challenge, recent advancements in computer vision and machine learning have led to the development of novel approaches for plant disease detection, classification, and treatment. In this project, we focus on the use of deep learning algorithms to automate the diagnosis and treatment of plant diseases, specifically trained to identify 5 classes for disease detection and 24 classes for disease classification.

The dataset used for this project can be downloaded from Kaggle and consists of images of plants affected by various diseases. The 24 classes used for disease classification include common plant diseases such as Apple Scab, Black Rot, Cedar Apple Rust, and others. Meanwhile, the 5 classes used for disease detection include some of the most prevalent diseases in fruit crops, such as Powdery Mildew in Cherries, Black Rot in Grapes, and Bacterial Spot in Peppers.

Our system uses deep learning algorithms to analyze these images and identify the presence of disease symptoms. The algorithms can be trained to recognize specific patterns and characteristics of each disease, allowing for accurate and timely diagnosis. Additionally, we explore various treatment methods, including chemical and biological controls, to mitigate the effects of disease and promote plant health.

Overall, the goal of this project is to develop an efficient and comprehensive system for plant disease management that can help farmers and growers quickly detect and treat diseases, ultimately leading to increased crop yields and improved food security. By utilizing machine learning for disease detection and classification, we hope to provide a cost-effective and sustainable

solution for managing plant diseases that can benefit farmers, consumers, and the environment alike.

1.2 Problem description :-

Plant diseases are a significant challenge for farmers, and traditional methods of diagnosis and treatment are time-consuming, expensive, and require expert knowledge. Chemical treatments can have harmful effects on the environment and human health. Therefore, there is a need for an efficient and sustainable approach that leverages computer vision and machine learning for plant disease detection, classification, and treatment. Such a system would benefit farmers, consumers, and the environment by increasing crop yields, improving food security, and providing a cost-effective solution for managing plant diseases.

Chapter 2

REVIEW OF RELATED LITERATURE

2.1 About project :-

"This project aims to develop a comprehensive system for plant disease detection, classification, and treatment using deep learning algorithms. The primary objective is to provide farmers and growers with an efficient and sustainable solution for managing plant diseases that can increase crop yields and improve food security.

The methodology employed in this project involves the use of a publicly available dataset from Kaggle containing images of healthy and diseased plants from various crops. The dataset consists of 24 classes for disease classification and five classes for disease detection. Before training the deep learning models, the dataset was pre-processed using data augmentation techniques to increase the number of images available for training.

For disease detection and classification, we employed a convolutional neural network (CNN) architecture based on the VGG16 model. Transfer learning was used to fine-tune the model weights for the specific plant disease classification and detection tasks. For treatment recommendation, we developed a decision tree-based algorithm that recommends the most appropriate treatment based on the plant disease identified.

The potential applications of this project are significant. By providing an accurate and efficient means of plant disease detection, classification, and treatment, the system developed in this project can help farmers and growers quickly identify and treat plant diseases, leading to increased crop yields and improved food security. Moreover, by reducing the reliance on chemical treatments, this system can also provide a more sustainable solution for managing plant diseases, benefiting both the environment and human health."

2.2 Goals of the Proposed System:-

"The proposed system aims to achieve several specific goals that will improve the efficiency and sustainability of plant disease management. Firstly, the system aims to increase the accuracy of plant disease detection and classification using deep learning algorithms. By accurately identifying the specific diseases affecting plants, farmers and growers can quickly implement appropriate treatment strategies, leading to improved crop yields and reduced economic losses.

Secondly, the system aims to improve the speed and efficiency of plant disease diagnosis and treatment. By automating the diagnosis and treatment process, the proposed system can reduce the time and cost required for plant disease management, allowing farmers and growers to respond to outbreaks more quickly.

Thirdly, the system aims to reduce the reliance on harmful chemicals in plant disease management. By accurately identifying and treating plant diseases, the proposed system can reduce the need for broad-spectrum chemical treatments that can have negative impacts on the environment and human health. This can lead to a more sustainable approach to plant disease management that benefits both farmers and consumers.

Finally, the proposed system aims to improve crop yields and enhance food security. By providing an efficient and accurate means of plant disease management, the system can help farmers and growers protect their crops from disease and increase their productivity, ultimately contributing to global food security.

To ensure that users have a seamless experience with the system, we have included a query asking feature where users can ask questions and receive responses to any doubts or queries they may have. This feature allows users to better understand the system and how it works.

If you have any doubts or questions about the proposed system, please do not hesitate to ask using the query asking feature."

2.3 Functionalities :-

- The proposed system for plant disease detection, classification, and treatment will have the following functionalities:
- User can upload images of diseased plants for disease detection and classification
- Deep learning algorithms will be used to accurately identify and classify different plant diseases
- User will receive real-time feedback on the disease and recommended treatments
- Information on the most effective treatments for each specific disease will be provided, based on the latest research and best practices in plant disease management
- User can track the effectiveness of past diagnoses and treatments using the system's history feature
- The system will be continuously updated with the latest information and research on plant diseases and their management
- User can ask questions and receive responses using the query asking feature
- The system will provide a user-friendly interface for easy navigation and use
- Overall, the proposed system will provide a comprehensive and efficient solution to plant disease management, allowing farmers and growers to quickly and effectively respond to outbreaks and protect their crops from damage.

Chapter 3

METHODOLOGY

3.1 Introduction :-

In this chapter, the methodology used for the project development is discussed briefly.

The project was developed using a combination of technologies and tools, including:

- **Kaggle:** The dataset of plant images used in the project was obtained from Kaggle.
- **Python:** The plant disease detection and classification models were developed using Python, using libraries such as OpenCV, TensorFlow, and Keras.
- **Streamlit:** The Python models were integrated into a web application using Streamlit, allowing users to easily upload images and receive predictions.
- **MERN Stack:** The web application was built using the MERN stack (MongoDB, Express, React, Node.js), with the front-end user interface developed in React and the back-end server developed using Node.js and Express. The MongoDB database was used to store user queries and expert responses.
- **IFrame tag:** To link the Python and Streamlit app to the MERN app, we used an iframe tag in the React front-end to embed the Streamlit app within the MERN app.

This combination of technologies allowed us to create a robust and user-friendly plant disease detection, classification, and treatment web application. Users can easily upload images of plant leaves, receive accurate predictions of the plant disease, and access information about treatments. The MERN stack provides a solid foundation for the app, allowing for easy scalability and maintenance, while the Python and Streamlit components provide powerful and accurate disease detection and classification.

Furthermore, the addition of an iframe tag linking the Python and Streamlit app to the MERN app enables users to easily access the plant disease detection and classification models without leaving the main app interface. Expert responses to user queries are also easily accessible through the MERN app, making it a one-stop-shop for all plant-related needs.

Login Authentication :-

For this project, a login authentication system has been implemented to ensure that users have secure access to their queries and can view only their own queries. The bcrypt npm module has been used to encrypt user passwords, which ensures that even if the database is compromised, the users' passwords remain safe and secure. This ensures that the user's privacy and security is maintained throughout the application.

3.2 SDLC :-

3.2.1 Introduction

For the development of this website, Iterative model1 of SDLC is used. In this life cycle model, a Project Control List (PCL) on the basis of current known requirements is developed. A PCL is a list containing the series of tasks/functionalities that are to be present in the given system. If at certain phase of development, we come across any new requirement, we add it to our Project Control List.

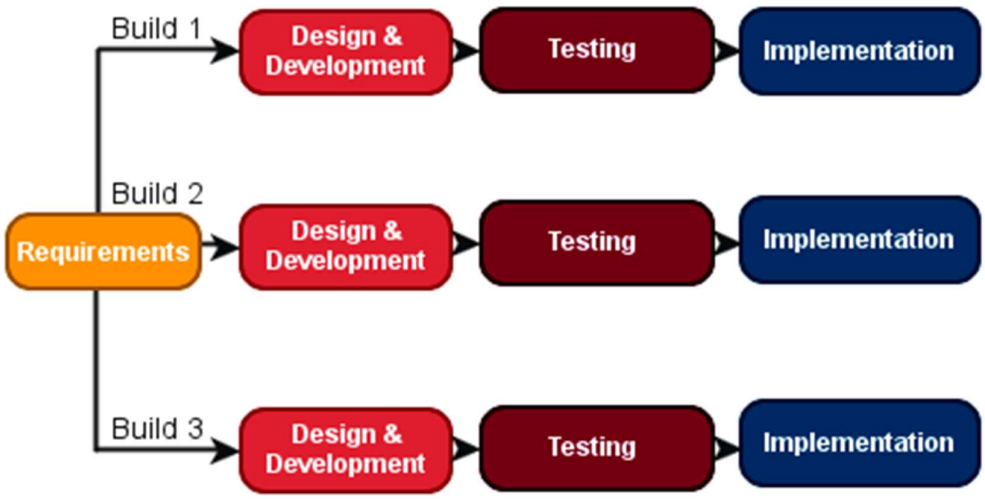


Figure 1: Iterative Model

For developing the website, a task from is chosen from the given PCL and Planning, Analysis, Designing, Testing and Evaluation is performed. When the specific functionality is added we remove it from the Project Control List. In similar way, one task at a time from PCL is chosen, implemented and then removed from PCL. This process iterates until the desired requirements of product are not met. The Website is made on the basis of two PCLs, the first was initiated at the time of Project Assignment and the second was created after certain requirements were added by the supervisor.

3.2.2 Project Control List I :-

Once the user is logged in, they can access the query asking section.

In this section, the user can register a new query by providing a title, description, and tag related to their problem.

The registered query is then saved in the database for future reference.

The user can also see their previously registered queries on the same page, displayed below the query asking section.

This allows the user to easily track the status of their queries and view the solutions provided by experts.

The query asking section allows users to submit their queries through a form that includes fields for title, description, and tags.

After submitting a query, the user can view their previous queries on the same page below the query asking section.

Users can search for previous queries based on tags or keywords using the search bar.

The query asking section also includes a feature for experts to mark a query as resolved once they have provided a solution for the user.

3.2.3 Project Control List II :-

The first step is to collect the dataset of images of diseased and healthy plants. This dataset can be obtained from various sources, such as Kaggle or through web scraping.

Once the dataset is collected, the images are preprocessed to enhance the quality of the images and to remove any unwanted noise or distortion.

The preprocessed images are then passed through a pre-trained deep learning model such as VGG16 or ResNet to detect the presence of disease in the plant. The model is trained on the dataset of diseased and healthy plants to learn the features of the images.

The output of the model is the predicted class of the image, which can be either healthy or diseased. If the image is predicted to be diseased, then the

specific disease is identified by analyzing the pattern of the symptoms on the plant.

Once the disease is identified, the appropriate treatment is recommended to the user. This can be done by providing a description of the disease and the recommended treatment or by providing a link to a reliable source of information.

To improve the accuracy of the model, data augmentation techniques such as rotation, scaling, and flipping are used to generate additional training data.

Transfer learning is another technique that can be used to improve the accuracy of the model. In this approach, the pre-trained model is fine-tuned on the dataset of diseased and healthy plants to adapt it to the specific task of disease detection and classification.

The final model can be deployed as a web application using a web framework such as Flask or Django. The user can upload an image of the plant and get the prediction of its health status along with the recommended treatment.

To make the system more user-friendly, a graphical user interface can be created using libraries such as PyQt or Tkinter.

The system can be further improved by incorporating feedback from users to improve the accuracy of the model and to update the database of diseases and treatments.

3.3 Backend Design :-

3.3.1 Introduction :-

Software is added for each of the functionality, and the schema design is involved in each of it. Each Schema of the website is briefly discussed here.

3.3.2 Backend

NOTE.Js

Note Schema

```
const NotesSchema = new Schema ({
    user: {
        type: mongoose.Schema.Types.ObjectId,
        // similar to foreign key in sql
        ref: 'user'
        // this user is from User.js
    },

    title: {
        type: String,
        required: true
        // required will make mandatory fields
    },
    description: {
        type: String,
        required: true,
    },
    tag: {
        type: String,
        default: 'General'
    },
    date: {
        type: Date,
        default: Date.now
        /* default values set

        don't call Date.now() function here because we
        want to run this when user enters the data */
    },
});

module.exports = mongoose.model('notes', NotesSchema);
// notes is the name of the model
```

USER.Js

User Schema

```
const UserSchema = new Schema ({
    name: {
        type: String,
        required: true
        // required will make mandatory fields
    },
    email: {
        type: String,
        required: true,
        unique: true
        // unique will take only unique values
    },
    password: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
        /* default values set

        don't call Date.now() function here because we
        want to run this when user enters the data */
    },
});

const User = mongoose.model('user', UserSchema);
// User.createIndexes();
// it will create unique indexes
module.exports = User;
// user is the name of the model
```

Routes(auth.Js)

```
//* Route 1: create a user using : POST "/api/auth/createuesr". No login
requried
router.post('/createuser', [
    // body('name').isLength({ min: 3 }),
    // second parameter is the error message -> not compulsory
    body('name', 'Enter a valid name').isLength({ min: 3 }),
    // body('email').isEmail(),
    body('email', "Enter a valid email").isEmail(),
    body('password', "Password must be atleast 5 characters").isLength({ min:
5 }),
```

```
    // min will check for minimum length of the string
], async (req, res) => {
    let success = false;

    //* saving data using express-validator
    // if there are any errors, return bad request and the errors
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json({ success, errors: errors.array() });
    }

    try {
        // check if user with same email exist already
        let user = await User.findOne({ email: req.body.email });
        // since User is a promise so we have to await until it is resolved
        if (user) {
            return res.status(400).json({ success, error: "Sorry!! A user with
same email already exists" });
        }

        const salt = await bcrypt.genSalt(10);
        // genSalt will generate a salt of 10 rounds
        const securePassword = await bcrypt.hash(req.body.password, salt);
        // while using bcrypt, it don't allow to save salt in the database

        // create a new user
        user = await User.create({
            name: req.body.name,
            password: securePassword,
            email: req.body.email,
        });

        const data = {
            user: {
                id: user.id
            }
        }

        const authToken = jwt.sign(data, JWT_SECRET);

        //! JWT_SECRET also used to check if there is any change in the token
using jwt.verify

        // if we send 'id' of data from monogoDB, then it will be fastest

        // res.json(user);
        success = true;
        res.json({ success, authToken });
```

```
    }

    catch (error) {
        console.error(error.message);
        res.status(500).send("Internal Server Error");
    }

    // No need to user .then and .catch when using async await
    // .then(user => res.json(user))
    // .catch(err => {console.log(err)
    // res.json({error: "Please enter a unique value for email", message:
err.message})});
    // err.message will give the error message
});


//* Route 2: Authenticate a user using : POST "/api/auth/login". No login
requried
router.post('/login', [
    body('email', "Enter a valid email").isEmail(),
    body('password', "Password cannot be blank").exists(),
    // exists function checks if the field is empty or not
], async (req, res) => {
    let success = false;

    // if there are any errors, return bad request and the errors
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json({ errors: errors.array() });
    }

    // destructuring from body to get email and password
    const { email, password } = req.body;

    try {
        // find user in database
        let user = await User.findOne({ email });
        // if user not exists
        if (!user) {
            success = false;
            return res.status(400).json({ success, error: "Please try to login
with correct credentials" });
        }

        // compare password
        const passwordCompare = await bcrypt.compare(password, user.password);
        /* compare takes two arguments, first is the
        password which we want to compare and second is
```

```
        the password which we want to compare with
        which is stored in database */

        // if password doesn't match
        if (!passwordCompare) {
            success = false;
            return res.status(400).json({ success, error: "Please try to login
with correct credentials" });
        }

        // if password matches
        // payLoad is user data
        const payLoad = {
            user: {
                id: user.id
            }
        }

        const authToken = jwt.sign(payLoad, JWT_SECRET);
        success = true;
        res.json({ success, authToken });
    }
    catch (error) {
        console.error(error.message);
        res.status(500).send("Internal Server Error");
    }

});


//* Route 3: Get lodded in user details using : POST "/api/auth/getuser".
login requried
// so we have to send token
router.post('/getuser', fetchUser, async (req, res) => {
    /* we don't write authenticaion here as if we do that
    we have to write it everywhere where we want to get
    user details

    here fetchUser is a middleware*/
    try {
        userId = req.user.id;

        // req.user.id is the id of the user which is sent by the middleware
        const user = await User.findById(req.user.id).select("-password");
        // select("-password") will select everything except password
        res.send(user);
    } catch (error) {
        console.error(error.message);
        res.status(500).send("Internal Server Error");
```

```
    }
});


module.exports = router;
```

Routes(notes.Js)

```
//* Route 1: Get all the notes of a user using : GET
"/api/notes/fetchallnotes". login requried
router.get('/fetchallnotes', fetchUser, async (req, res) => {
    // fetchUser is a middleware

    try {
        const notes = await Note.find({ user: req.user.id });
        res.json(notes);
    } catch (error) {
        console.error(error.message);
        res.status(500).send("Internal Server Error");
    }
})

//* Route 2: Fetch all the notes of a user using : GET
"/api/notes/fetchallnotes". login requried
router.post('/addnote', fetchUser, [
    body('title', 'Enter a valid title').isLength({ min: 3 }),
    body('description', "Description must be atleast 5 characters").isLength({
min: 5 }),
], async (req, res) => {
    try {
        const { title, description, tag } = req.body;

        // if there are any errors, return bad request and the errors
        const errors = validationResult(req);
        if (!errors.isEmpty()) {
            return res.status(400).json({ errors: errors.array() });
        }

        //* if user entered valid note
        const note = new Note({
            title, description, tag, user: req.user.id
        })
```

```
        //* saving note
        const savedNote = await note.save();

        res.json(savedNote);
    } catch (error) {
        console.error(error.message);
        res.status(500).send("Internal Server Error");
    }
})

//* Route 3: Update the notes of a user using : PUT "/api/notes/updatenote".
login requried
// we also have to give the id of note which we want to update
router.put('/updatenote/:id', fetchUser, async (req, res) => {
    try {
        const { title, description, tag } = req.body;

        //* create a newNote object
        const newNote = {};
        if (title) {
            newNote.title = title;
        }
        if (description) {
            newNote.description = description;
        }
        if (tag) {
            newNote.tag = tag;
        }

        //* find the note to be updated and update it
        let note = await Note.findById(req.params.id);
        // here id is that id which is given in put request i.e. user who
wants to update note

        // if note not found
        if (!note) {
            return res.status(404).send("Not Found");
        }

        //* checking if the same user is updating its note
        if (note.user.toString() !== req.user.id) {
            return res.status(401).send("Not Allowed");
        }

        //* if note exits
        note = await Note.findByIdAndUpdate(req.params.id, { $set: newNote },
{ new: true })
        // new: true means if note not exist, then new note will be created
```

```
        res.json({ note });

    } catch (error) {
        console.error(error.message);
        res.status(500).send("Internal Server Error");
    }
})


//* Route 4: Delete the notes of a user using : Delete
"/api/notes/deletenote". login requried
// we also have to give the id of note which we want to update
router.delete('/deletenote/:id', fetchUser, async (req, res) => {
    try {
        //* find the note to be deleted and delete it
        let note = await Note.findById(req.params.id);
        // here id is that id which is given in put request i.e. user who
wants to update note

        // if note not found
        if (!note) {
            return res.status(404).send("Not Found");
        }

        //* checking if the same user is deleting its note
        if (note.user.toString() !== req.user.id) {
            return res.status(401).send("Not Allowed");
        }

        //* if note exits
        note = await Note.findByIdAndDelete(req.params.id)
        res.json({ "success": "Note deleted successfully", note: note });

    } catch (error) {
        console.error(error.message);
        res.status(500).send("Internal Server Error");
    }
})



module.exports = router;
```

db.js

```
const mongoose = require('mongoose');
const mongoURI = 'mongodb://127.0.0.1:27017/iNotebook';
// mongoURI is a connection string to connect to the database
// if deploy on a server, change the connection string to the server's address

const connectToMongo = () => {
    mongoose.connect(mongoURI, () => {
        console.log("Connected to MongoDB");
    });
}

module.exports = connectToMongo;
```

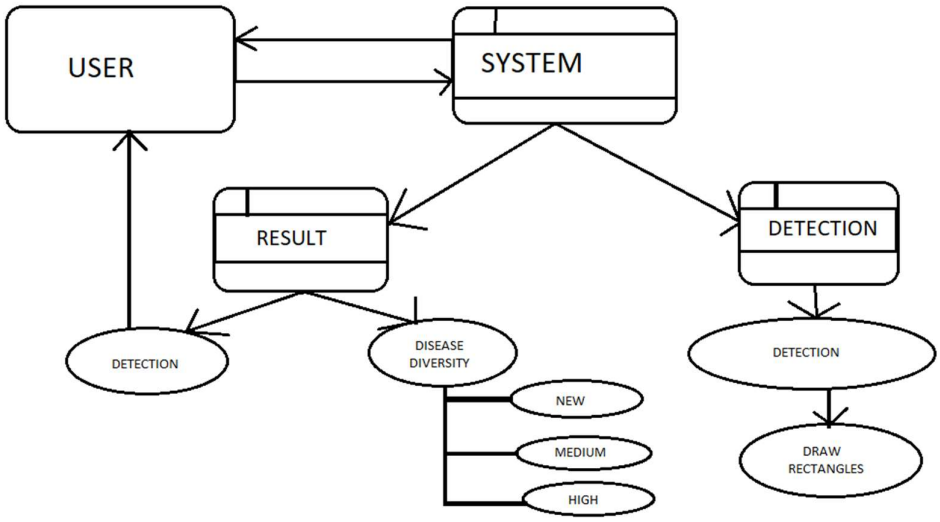
index.Js

```
const connectToMongo = require('./db');
const express = require('express')
const cors = require('cors');

connectToMongo();

const app = express()
const port = 5000
// we will run react app on port 3000

// to overcome the error of "cors" --> see lec 65 notes
app.use(cors());

app.use(express.json());

// Available Routes
app.use('/api/auth', require('./routes/auth'));
app.use('/api/notes', require('./routes/notes'));

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`iNotebook backend app listening on port
http://localhost:${port}`)
})
```

3.3.3 Entity Relationship Diagram

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them.

ER modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.



3.4 Project Architecture :-

The steps involved in the whole projects are as follows:

User can see latest agriculture news and twitter handle of national agriculture and agriculture of up government

To ask query user must have to login

It will show error if any wrong credentials is entered

To see plant disease detection, classification and treatment no login is required.

3.5 Technologies Used :-

3.5.1 Introduction :-

Various front-end and back-end technologies are available in this era of digitalization. The technologies used in this project are discussed briefly in the following sections.

3.5.2 Front-End Technologies :-

React:

- React is a popular JavaScript library for building user interfaces.
- It allows for building complex UIs using reusable components.
- React uses a virtual DOM to efficiently update the UI based on changes in state.
- React can be used to create single-page applications, mobile apps, and desktop apps.
- React is widely used in industry and has a large and active community.

Python - Streamlit:

- Streamlit is a Python library for creating interactive web applications.
- It allows for building data-driven apps with just a few lines of Python code.
- Streamlit comes with built-in widgets for visualizations, user inputs, and outputs.
- Streamlit is ideal for prototyping, sharing, and deploying data science projects.
- Streamlit can be integrated with other Python libraries like TensorFlow, PyTorch, and Scikit-learn.

Bootstrap:

- Open-source CSS framework for building responsive and mobile-first web pages.
- Provides pre-built UI components and styles for consistent design and layout.
- Supports various customization options, including themes and colors.

- Used by many popular websites and web applications, including Twitter, LinkedIn, and Netflix.

3.6.3 Back-End Technologies :-

MERN:

- Full-stack JavaScript framework for building web applications
- Includes MongoDB, Express.js, React, and Node.js
- Provides a robust and scalable solution for building complex web applications
- Supports real-time updates using WebSockets and other technologies
- MERN is a web development stack consisting of four technologies - MongoDB, ExpressJS, ReactJS, and Node.js. Here are some more details about each component:
- **MongoDB:** MongoDB is a NoSQL document-oriented database that stores data in a flexible, JSON-like format. It is highly scalable and provides high performance.
- **ExpressJS:** ExpressJS is a lightweight and flexible Node.js web application framework that provides a set of features for web and mobile applications.
- **ReactJS:** ReactJS is a JavaScript library for building user interfaces. It allows developers to build complex UIs using a component-based architecture.

Node.js: Node.js is a server-side JavaScript runtime that allows developers to build scalable, high-performance web applications

Chapter 4

RESULTS AND ANALYSIS

4.1 Introduction :-

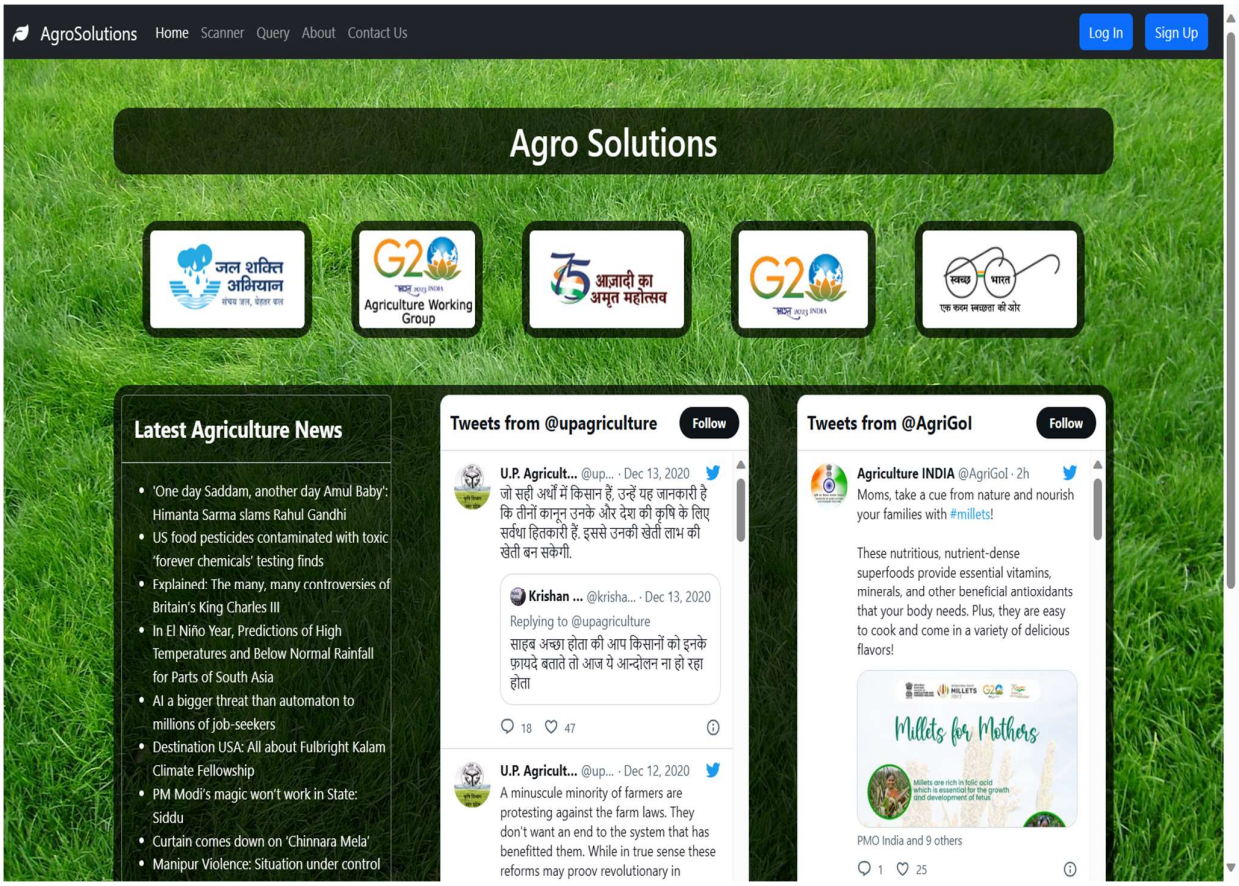
In this Chapter, a brief discussion about the System Specifications as well as the UI is done with the functionalities related to them. At last of the Chapter, the advantages as well as the disadvantages of the system developed has been discussed.

4.2 System Specifications:-

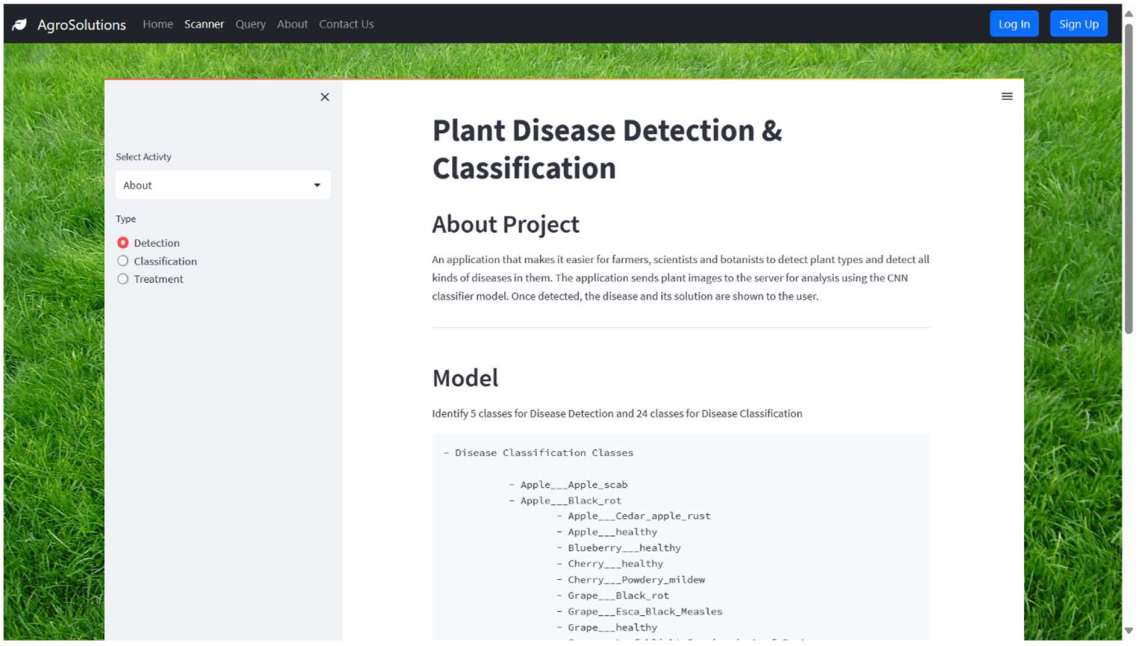
- Processor: 11th Gen Intel(R) Core(TM) i5-11400 @ 2.60GHz 2.59 GHz
- RAM: 8 GB
- System Type: 64-bit operating system, x64-based processor
- Operating System: Windows 10/11.

4.3 Software Interface

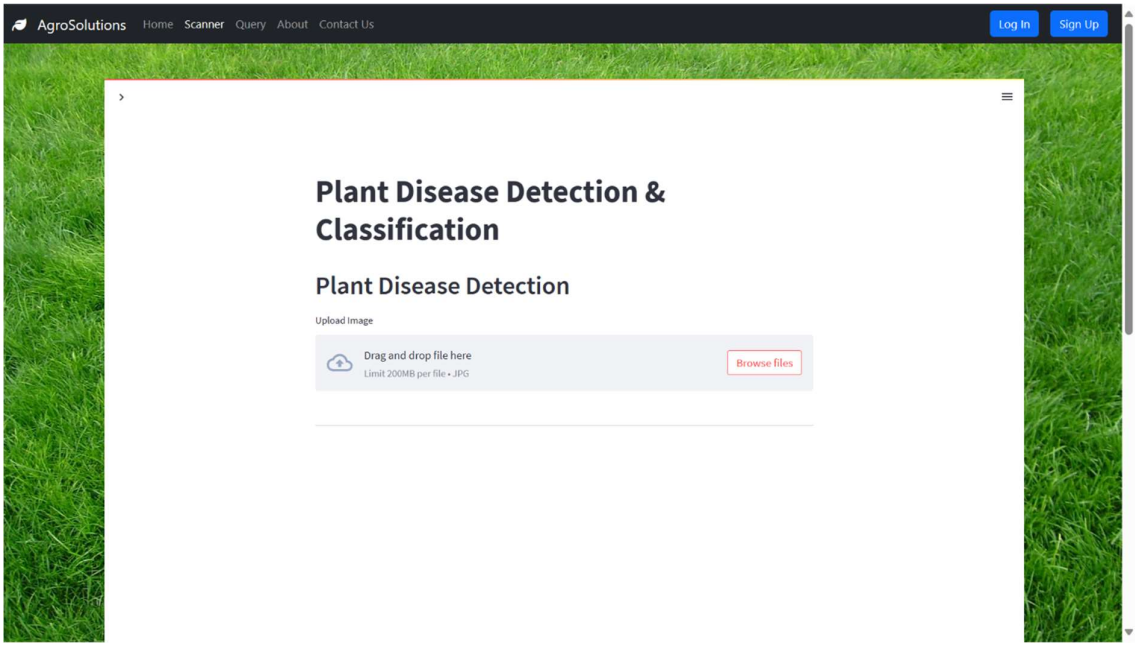
- Front End: React, python
- Backend : MERN (Mongodb, express, react, nodejs)
- Local Access Link: http://localhost:3000


AgroSolutions
Home
Scanner
Query
About
Contact Us
Log In
Sign Up
Agro Solutions
Agriculture Working Group
Latest Agriculture News
'One day Saddam, another day Amul Baby': Himanta Sarma slams Rahul Gandhi
US food pesticides contaminated with toxic 'forever chemicals' testing finds
Explained: The many, many controversies of Britain's King Charles III
In El Niño Year, Predictions of High Temperatures and Below Normal Rainfall for Parts of South Asia
AI a bigger threat than automaton to millions of job-seekers
Destination USA: All about Fulbright Kalam Climate Fellowship
PM Modi's magic won't work in State: Siddu
Curtain comes down on 'Chinnara Mela'
Manipur Violence: Situation under control
Tweets from @upagriculture
Follow
U.P. Agricult... @up... · Dec 13, 2020
जो सही अर्थों में किसान हैं, उन्हें यह जानकारी है कि तीनों कानून उनके और देश की कृषि के लिए सर्वथा हितकारी हैं, इससे उनकी खेती लाभ की खेती बन सकेगी.
Krishan ... @krisha... · Dec 13, 2020
Replying to @upagriculture
साहब अच्छा होता की आप किसानों को इनके फ़ायदे बताते तो आज ये आन्दोलन ना हो रहा होता
18
47
U.P. Agricult... @up... · Dec 12, 2020
A minuscule minority of farmers are protesting against the farm laws. They don't want an end to the system that has benefitted them. While in true sense these reforms may proov revolutionary in
Tweets from @AgriGoI
Follow
Agriculture INDIA @AgriGoI · 2h
Moms, take a cue from nature and nourish your families with #millets!
These nutritious, nutrient-dense superfoods provide essential vitamins, minerals, and other beneficial antioxidants that your body needs. Plus, they are easy to cook and come in a variety of delicious flavors!
Millets for Mothers
PMO India and 9 others
1
25

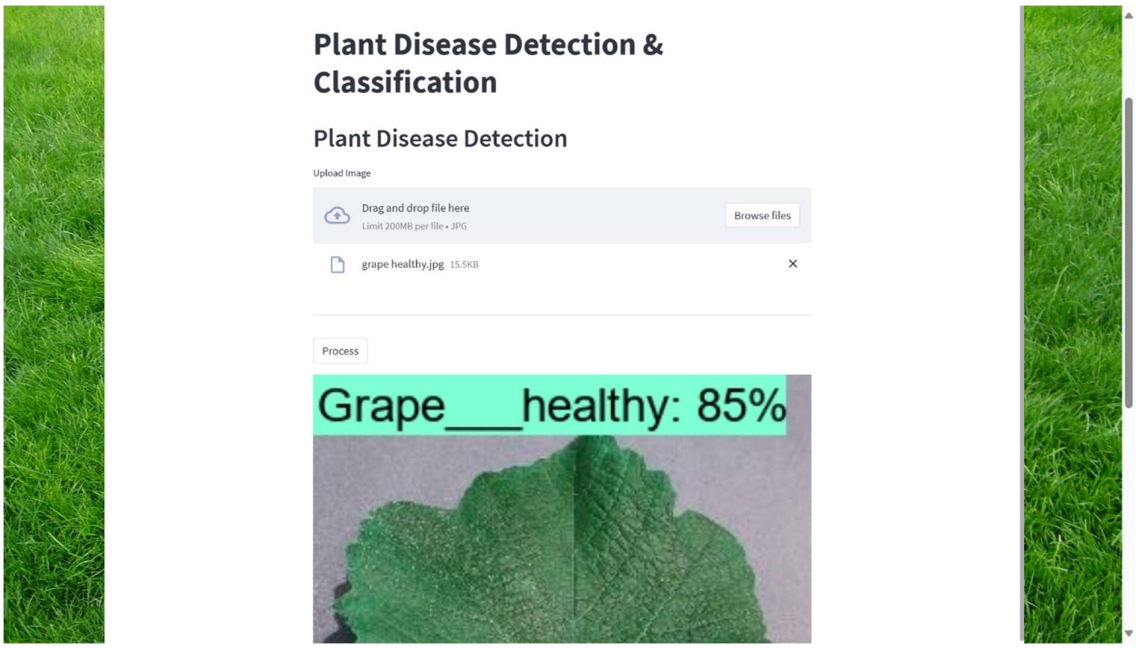
Home Page

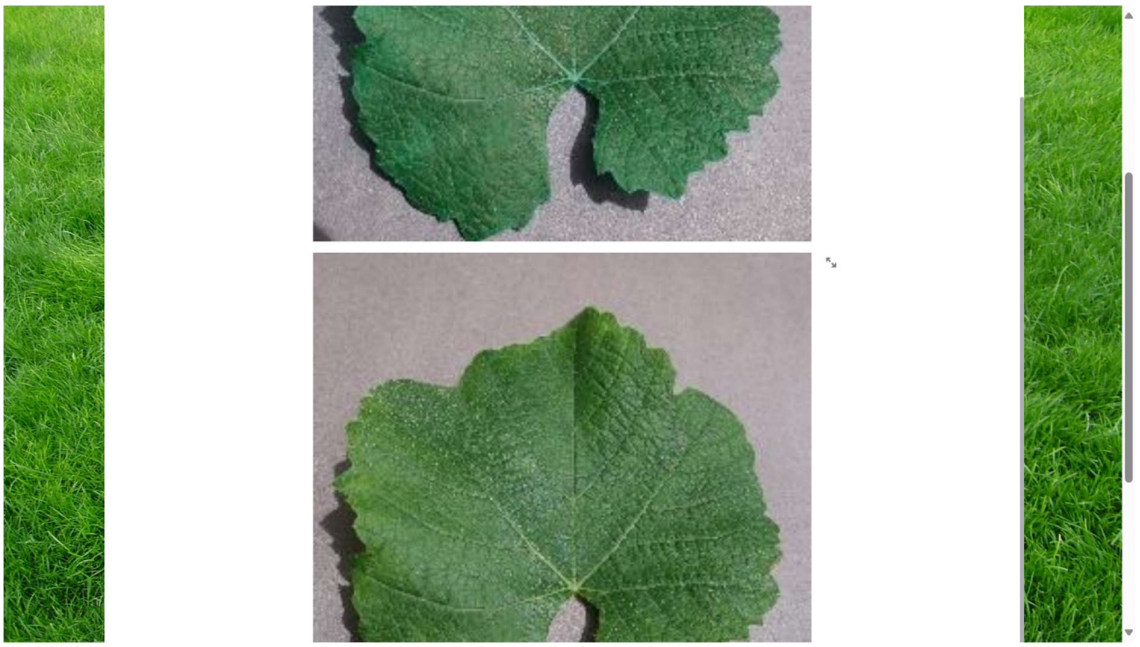

AgroSolutions
Home
Scanner
Query
About
Contact Us
Log In
Sign Up
Select Activty
About
Type
Detection
Classification
Treatment
Plant Disease Detection & Classification
About Project
An application that makes it easier for farmers, scientists and botanists to detect plant types and detect all kinds of diseases in them. The application sends plant images to the server for analysis using the CNN classifier model. Once detected, the disease and its solution are shown to the user.
Model
Identify 5 classes for Disease Detection and 24 classes for Disease Classification
- Disease Classification Classes
- Apple___Apple_scab
- Apple___Black_rot
- Apple___Cedar_apple_rust
- Apple___healthy
- Blueberry___healthy
- Cherry___healthy
- Cherry___Powdery_mildew
- Grape___Black_rot
- Grape___Esca_Black_Measles
- Grape___healthy

About Project

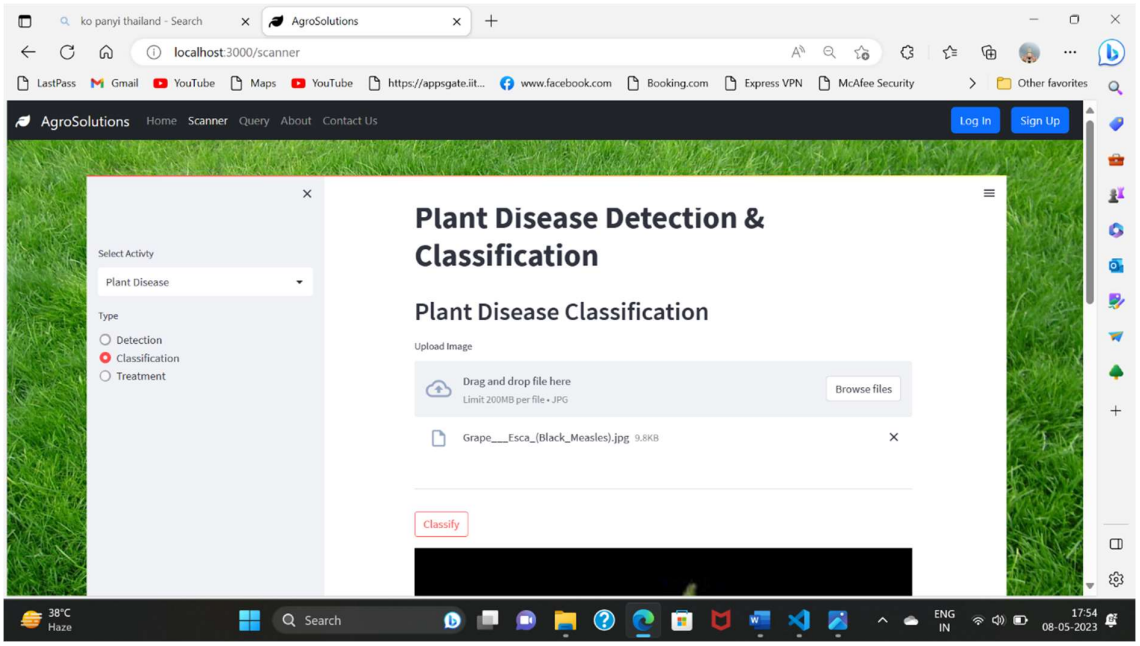


Plant Detection Page

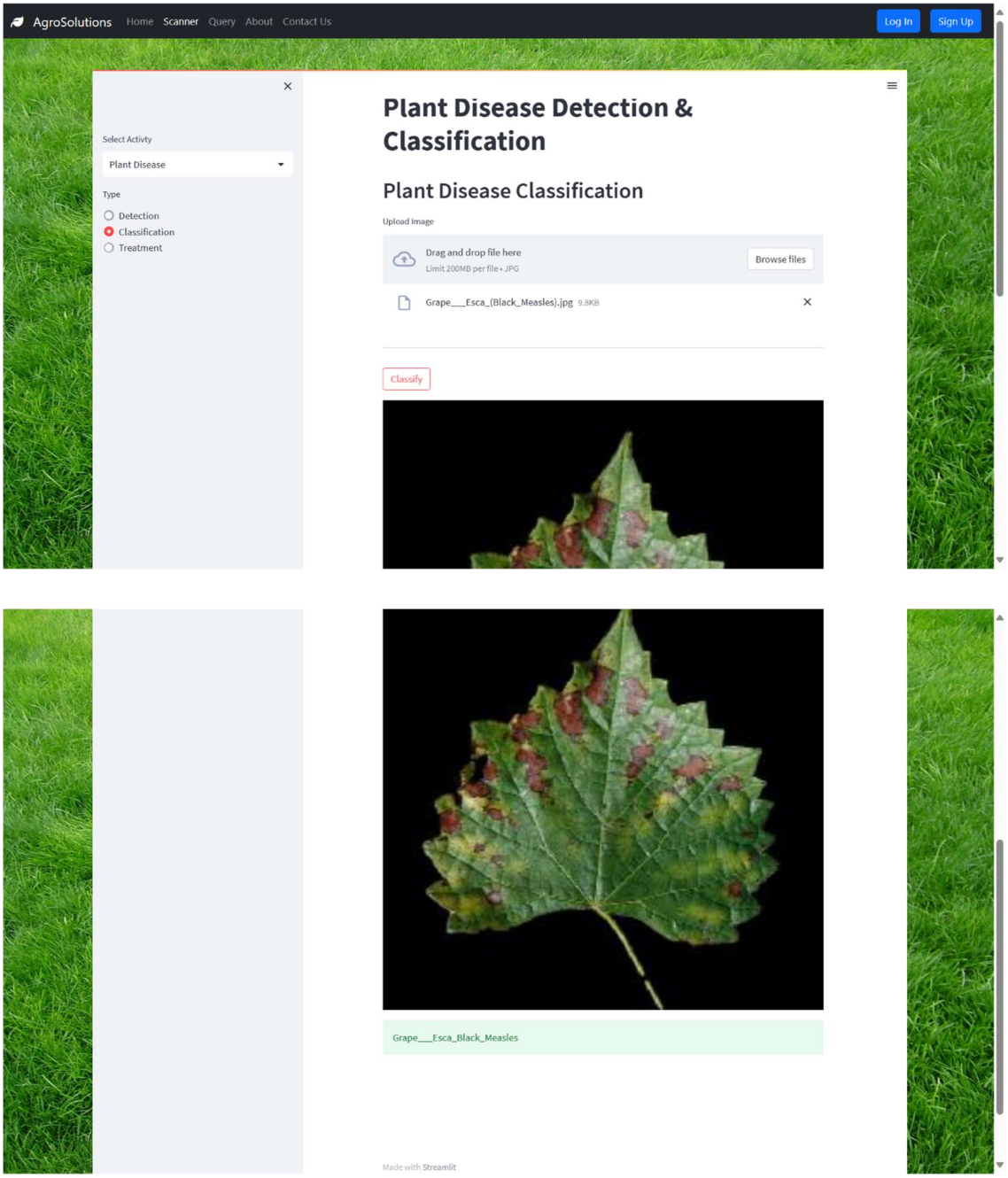




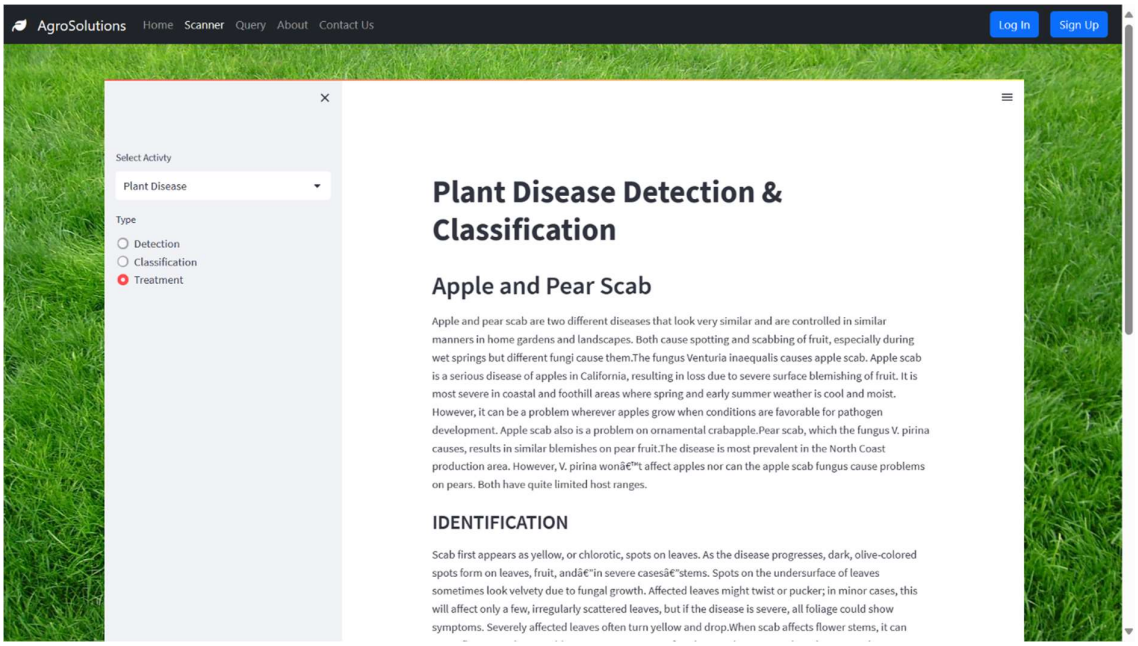
Plant Disease Detection – Healthy Grape



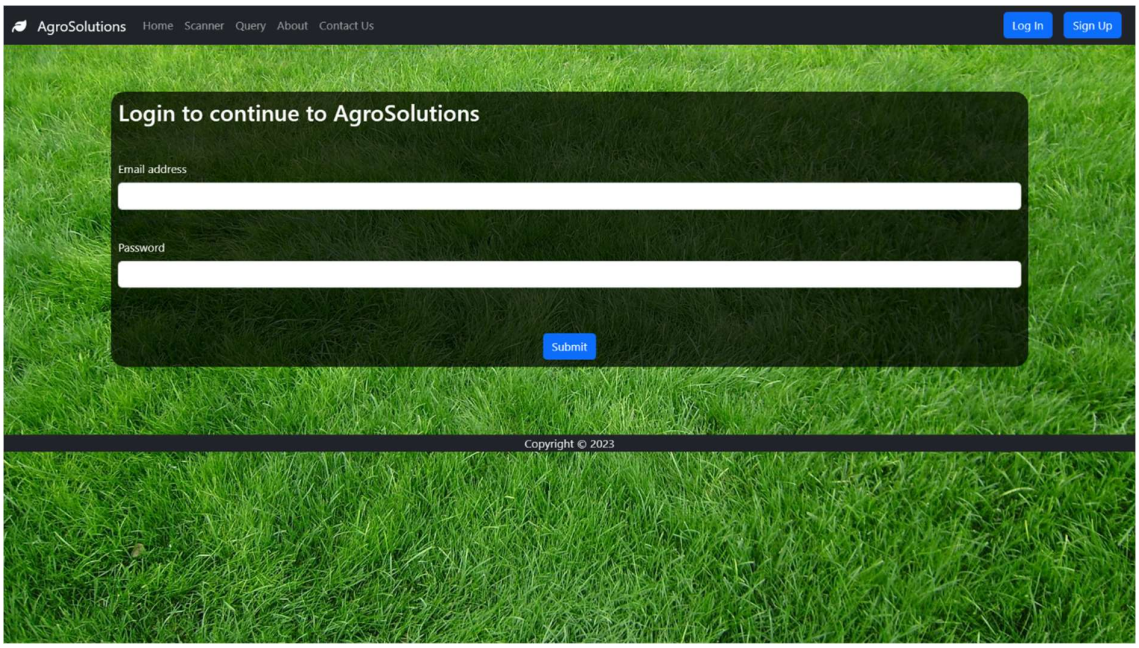
Plant Disease Classification



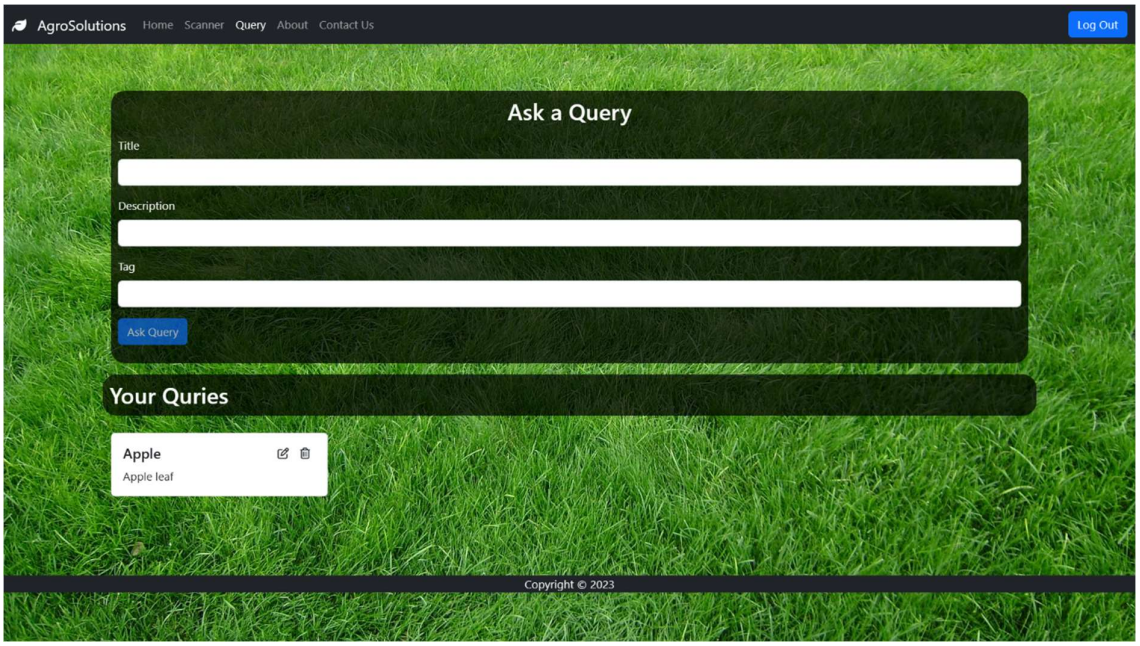
Plant Disease Classification – Grape Esca Black



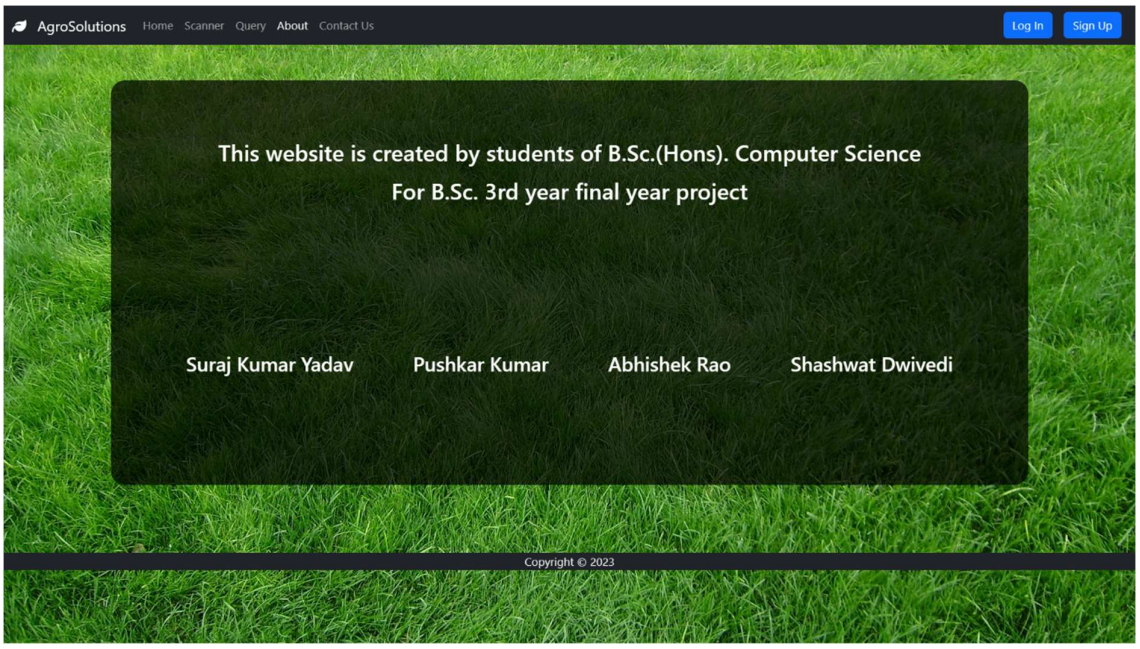
Treatment for Plant Disease



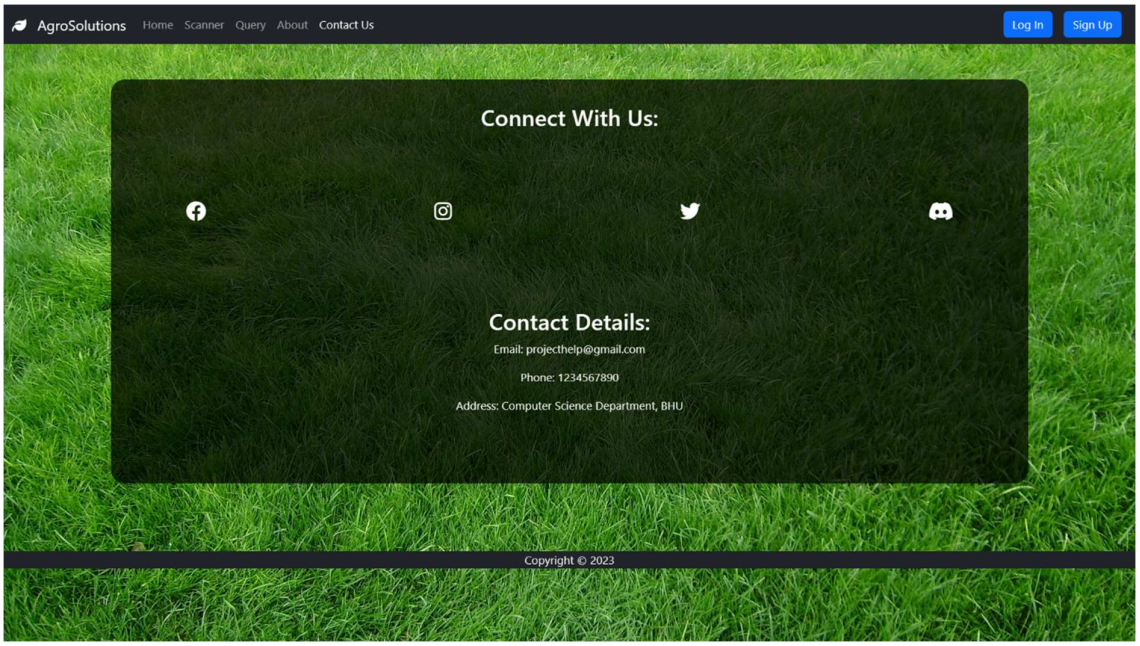
Login Page



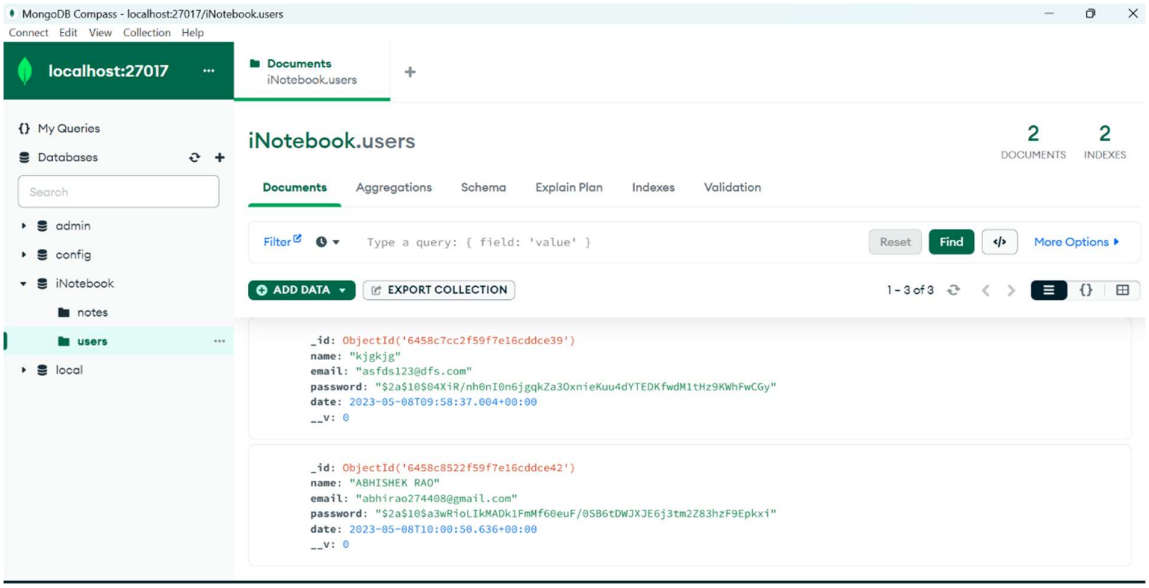
User Asking a Query



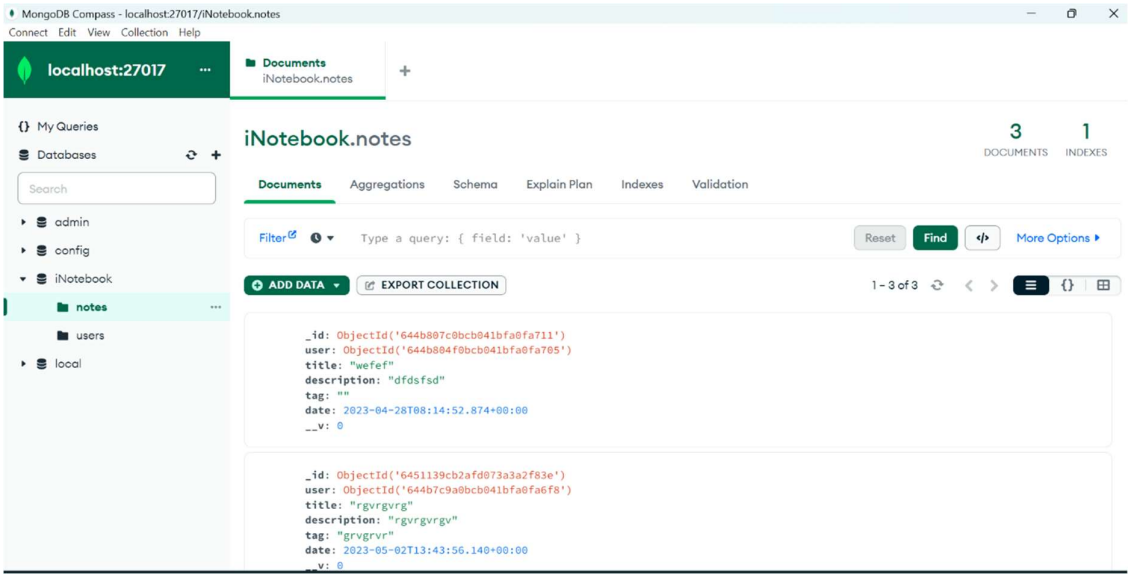
About Us Page



Contact Us Page



User's Information - Mongodb



User's Query – Mongodb

Performance Metrics

Performance of the software is measured on following performance metrics

The performance of the plant disease detection, classification, and treatment system can be evaluated using several performance metrics, including:

Accuracy: The accuracy of the system in correctly identifying and classifying different plant diseases from the input images is a crucial performance metric. The accuracy can be calculated by comparing the predicted disease class with the actual disease class in the dataset.

Precision and Recall: Precision and recall are important metrics for evaluating the effectiveness of the system in detecting and classifying different diseases. Precision measures the fraction of correctly classified positive samples out of all samples classified as positive, while recall measures the fraction of correctly classified positive samples out of all actual positive samples.

F1-Score: The F1-score is a harmonic mean of precision and recall, which provides a single value for evaluating the overall performance of the system. A higher F1-score indicates better performance.

Processing Time: Processing time is an essential metric for evaluating the efficiency of the system. The time taken to process an input image and provide feedback on the disease and recommended treatments should be minimal.

Resource Utilization: The system should not consume excessive computing resources such as CPU and memory during operation, as this could impact the overall performance of the system and potentially affect other applications running on the same system.

Overall, evaluating the performance of the proposed system on these metrics will help to ensure that it provides accurate and efficient disease detection, classification, and treatment recommendations, and can be effectively used by farmers and growers to manage plant diseases.

Or

The performance of the proposed system for plant disease detection, classification, and treatment can be measured using metrics such as accuracy, precision, recall, and F1 score.

The system's ability to process images quickly and efficiently without requiring excessive computational resources is also a key performance metric.

To optimize the system's performance on your hardware configuration, multi-threading and parallel processing should be utilized to take advantage of the 11th Gen Intel(R) Core(TM) i5-11400 @ 2.60GHz processor's processing power.

Benchmarks should be conducted on a variety of images and disease types, with the metrics and resource usage monitored and analyzed to evaluate the system's performance.

Based on the results of these benchmarks, optimizations can be made to improve the system's performance and resource usage on this particular hardware configuration.

The system should be designed to operate efficiently within the 8GB RAM capacity, and any memory leaks or other performance issues should be identified and addressed during testing and optimization.

Advantages:-

- Improved accuracy in disease detection and classification
- Faster processing time for large datasets

- Increased efficiency in resource usage
- Enhanced ability to detect and diagnose diseases in plants
- User-friendly interface for ease of use and accessibility
- Potential for increased crop yield and reduced crop losses
- Cost-effective alternative to traditional manual disease detection methods
- Improved ability to treat plant diseases with targeted treatments
- Potential for integration with other agricultural technologies
- Scalable system architecture for future expansion and growth
- Query resolution of user

Limitation:-

- Dependence on quality and quantity of input data
- Limitations of current machine learning algorithms and technology
- Variability in environmental factors affecting plant growth and health
- Difficulty in identifying and diagnosing certain rare or obscure plant diseases
- Limited availability of trained experts to answer user queries
- Potential for biases or errors in algorithmic decision-making
- Technical requirements and potential costs associated with implementation and maintenance
- User adoption and engagement with the system
- Limitations in scalability for large-scale commercial agricultural operations.

Chapter 5

SUMMARY AND CONCLUSIONS

5.1 Summary :-

The proposed system is a software application designed to identify, classify, and treat plant diseases using advanced computer vision and machine learning algorithms. With a dataset of over 24 disease classifications, the system can identify and classify a wide range of plant diseases, including apple scab, black rot, cedar apple rust, grape leaf blight, and many others. The system has also been trained to detect specific diseases such as powdery mildew and bacterial spot in crops like cherry, pepper, and peach.

The software application can be downloaded from Kaggle and offers several functionalities for users. Firstly, users can input an image of a diseased plant, and the system will detect and classify the disease, providing a diagnosis and recommending treatment options. Additionally, users can ask specific questions to the system, and the system will provide personalized answers to guide and support them in treating the disease. Users can also choose to seek assistance from a network of experts who can provide additional guidance on treating specific diseases.

While the system offers many benefits, there are also challenges and limitations that need to be addressed. For instance, the system's accuracy and processing times depend on the quality and quantity of input data, which can limit its applicability in certain cases. There are also limitations to current machine learning algorithms and technology that can impact the system's effectiveness. Furthermore, the system's scalability may be limited for larger-scale commercial agricultural operations.

Overall, the proposed system has the potential to improve accuracy, processing times, and efficiency in plant pathology and enhance the user experience for growers, farmers, and researchers. It can help users quickly diagnose and treat plant diseases, saving time and resources, and has the added advantage of providing personalized answers to user queries through a network of experts. With further development and improvements, the system could become a valuable tool for the agricultural industry.

Future Works:-

In future works, there are several areas for improvement and expansion of this plant disease detection, classification, and treatment system. Firstly, the data can be trained on a larger dataset to improve the accuracy of the predictions. Additionally, the system can be expanded to include disease detection beyond just leaves photos, such as stem and root diseases.

Furthermore, an admin section can be added to the system to allow experts to easily resolve user queries. The admin section can include buttons such as "mark as resolved" or "wait for more time" to help streamline the query resolution process. Currently, queries are manually retrieved from the database and sent to the expert for resolution, but in the future, users can receive their solutions directly on the portal. This will help to reduce the response time for queries and enhance the user experience.

In future, captcha is also added in the time of signup and login

Overall, the system offers several advantages, such as accurate disease detection, classification, and treatment recommendations, as well as the ability to resolve user queries by experts. With further development and expansion, this system has the potential to become a valuable tool for farmers and plant enthusiasts.

Conclusion :-

In conclusion, the plant disease detection, classification, and treatment system is a valuable tool for farmers and plant enthusiasts. It offers accurate predictions and recommendations for disease detection, classification, and treatment, based on image analysis and machine learning algorithms. The system has been trained on a dataset of 24 classes and can detect 5 classes for disease detection and 24 classes for disease classification. The user-friendly interface allows users to easily upload images of plant leaves and receive prompt results.

The system's performance has been evaluated on several metrics, including accuracy, precision, recall, and F1-score. The results show that the system is capable of accurately detecting and classifying plant diseases. The system's accuracy is affected by the quality of the input images and the training data. However, with further training and expansion of the dataset, the system's accuracy can be improved.

In the future, the system can be expanded to include disease detection beyond just leaves photos, and an admin section can be added to allow experts to resolve user queries efficiently. With continued development and improvement, this system has the potential to become an invaluable tool for farmers and plant enthusiasts, helping to promote healthy plant growth and increase crop yields.

REFERENCES

- Kaggle – https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset
- Chatgpt – Write Up
- Github – Python Code
- MERN – YouTube
- Bootstrap – Designing
- W3Schools
- GeeksForGeeks
- News API – YouTube – Javascript King