ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY, KOTTAYAM

PROJECT REPORT ON

LEAFLENS

In partial fulfillment of the requirements for the

Award of the degree of

BVoc SOFTWARE DEVELOPMENT

By

KEERTHI S

III BVoc Software Development

Register No: VB22SWD021

ARCHA P

III BVoc Software Development

Register No: VB22SWD010

Under the guidance of

Ms. NITHYA A B

Department of Computer Applications

2022-2025

ST. TERESA’S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY, KOTTAYAM

PROJECT REPORT ON

LEAFLENS

In partial fulfillment of the requirements for the

Award of the degree of

BVoc SOFTWARE DEVELOPMENT

By

ARCHA P

III BVoc Software Development

Register No: VB22SWD010

Under the guidance of

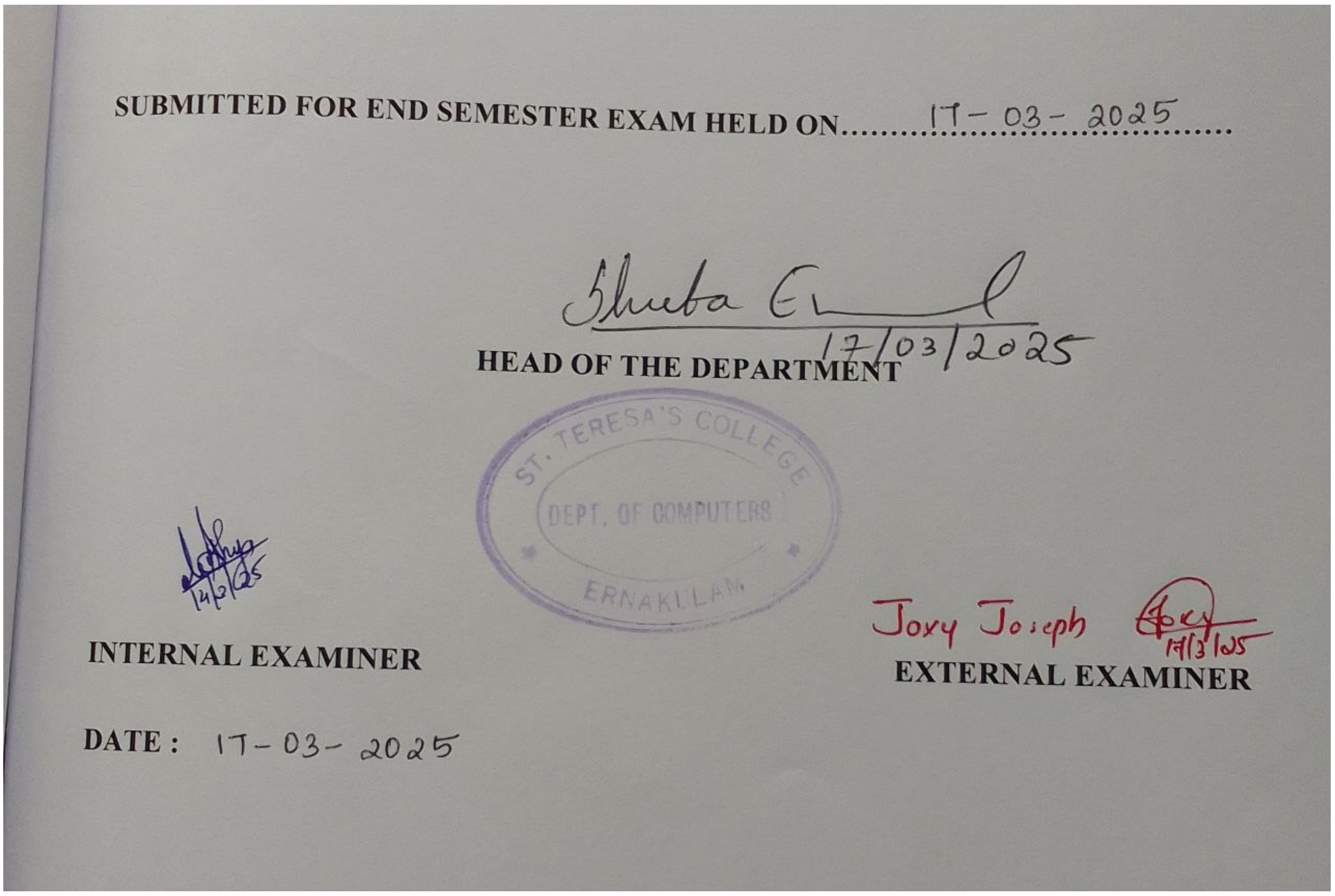
MS. NITHYA A B

Department of Computer Applications

2022-2025

CERTIFICATE

This is to certify that the project report on **LEAFLENS** is a Bonafide record of the work done by **ARCHA P (VB22SWD010)** during the year 2022-2025 and submitted in partial fulfillment of the requirement for the degree of **BVoc Software Development** under Mahatma Gandhi University.



SUBMITTED FOR END SEMESTER EXAM HELD ON 17-03-2025

[illegible] 17/03/2025

HEAD OF THE DEPARTMENT

ST. TERESA'S COLLEGE DEPT. OF COMPUTERS ERNAKULAM

[illegible] 17/3/25

INTERNAL EXAMINER

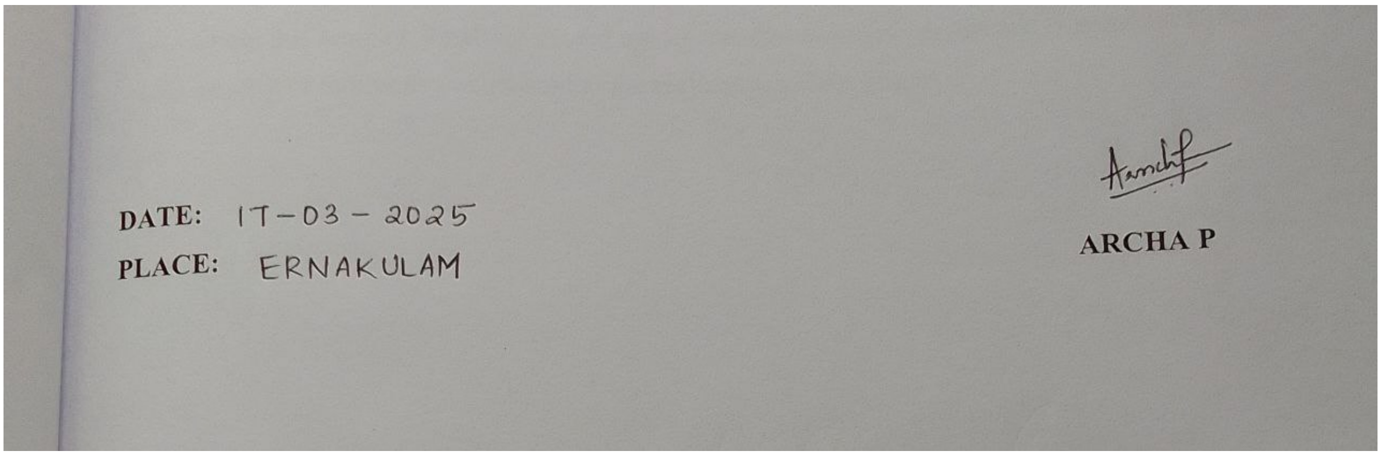
Joxy Joseph [illegible] 17/3/25

EXTERNAL EXAMINER

DATE : 17-03-2025

DECLARATION

I, **ARCHA P** (Register no: **VB22SWD010**), BVoc Software Development final year student of **St. Teresa's College (Autonomous), Ernakulam**, hereby declare that the project submitted named **LEAFLENS** for the **Bachelors of Vocation Degree in Software Development** is my original work. I further declare that the said work has not previously been submitted to any other university or academic body.



DATE: 17-03-2025

PLACE: ERNAKULAM

[illegible]

ARCHA P

ACKNOWLEDGEMENT

First and foremost, I would like to thank God almighty for the successful completion of my project. I express my sincere thanks to Provincial Superior and Manager **Rev. Sr. Nilima CSST, Rev. Sr. Tessa CSST and Principal Dr. Alphonsa Vijaya Joseph of St. Teresa's college (AUTONOMOUS)** for giving me an opportunity to undertake this project. I express my sincere gratitude towards the **Head of the department Ms. SHEEBA EMMANUEL** and, I would like to extend my heartfelt appreciation to **Ms. NITHYA A.B** (Asst. Prof), my project guide for her constant support which helped in the successful completion of my project. I'm grateful to all the faculties of the Department of Computer Applications for their unwavering support and guidance throughout this journey. Finally, I extend my sincere thanks to my parents and friends and all those who directly or indirectly contributed to the realization of this project.

SYNOPSIS

Plant Disease Detector is a web-based machine learning-powered application that aids farmers, gardening experts, and researchers in disease diagnosis of plants. The application diagnoses possible diseases by analyzing pictures of plant leaves uploaded to the system and delivering diagnostic results at high accuracy levels. The system offers the identified disease name as well as a confidence level of the diagnosis for farmers and gardeners to intervene in time in order to preserve their crops. The device utilizes image recognition technology to detect diseases rapidly, accurately, and easily. Additionally, the database of the application is regularly updated with new disease information and better algorithms, keeping it accurate and up-to-date over time. This ongoing development enables the detector to recognize an increasing number of plant diseases on different crops, making it a useful tool for various agricultural applications. The user-friendly interface does not need any special technical expertise, and even a layperson can upload pictures and get diagnostic reports in a matter of seconds. This integrated approach to disease diagnosis and management makes the Plant Disease Detector an important tool for the propagation of sustainable agriculture and food security.

INDEX

1.INTRODUCTION

1.1 ABOUT THE PROJECT

The LeafLens Website is a new solution that aims to make plant disease identification easier and more efficient. Through the use of breakthroughs in machine learning and image processing, the website offers a convenient, user-friendly, and precise tool for the detection and classification of plant diseases from leaf images. This initiative is designed to equip farmers, agricultural experts, and scientists with technologically enhanced knowledge so that interventions could be made at the right time and crops better managed. This online platform does away with traditional manual examination or lab-based testing, which is both costly and not easily accessible. Users need only upload photos of infected plant leaves using an internet browser, and the advanced algorithms in the system interpret the images to detect possible diseases. The output is made available in a concise and readable format, usually consisting of the forecasted disease name, a confidence value that reflects the level of certainty of the diagnosis, and possibly additional information like disease descriptions. This easily accessible information equips users to make informed decisions on crop management, enabling them to take immediate and specific action to reduce crop loss and maximize yields. In addition, the platform's learning ability through continuous training with fresh data maintains its accuracy and enhances its capability to identify more forms of plant disease in a broader variety of plants. This renders it an excellent tool for multiple agricultural environments and supports sustainable and effective farming methods.

2.SYSTEM ANALYSIS

2.1 INTRODUCTION

Plant diseases are a major threat to world agriculture, resulting in huge losses in crop production and quality. Early detection and timely action are essential to control these losses and guarantee food security. Yet, conventional methods of detecting plant diseases, like visual examination by field experts or laboratory analysis, are time-consuming, labour-intensive, and subject to human error. In order to overcome these issues, we are suggesting a Plant Disease Detection Website that is aimed at automating the identification and classification of plant diseases based on sophisticated image processing and machine learning algorithms.

2.2 EXISTING SYSTEM

Today, the identification of plant diseases is highly dependent on human experts, which is time-consuming, tedious and error-prone. Visual examination and laboratory analysis are traditional methods used, but these are limited. There are digital tools available, but they find it difficult to correctly identify diseases in real-time. These digital tools do not have sophisticated technology to process images well, which needs a lot of computational power and cannot deliver high accuracy on different plant species and diseases. To overcome such challenges, we require more solutions that can identify plant diseases rapidly and accurately. This may be achieved through the application of sophisticated computer algorithms, integration with sensors and drones, and creation of specialized databases.

2.3 PROPOSED SYSTEM

The system is supposed to detect and classify plant diseases based on leaf images through enhanced image processing and machine learning methods. The system will include taking high-resolution images of plant leaves and applying pre-processing methods to improve image quality and eliminate noise. Important features in the images will be extracted by employing texture analysis and color histogram analysis. The cameras will then supply the robust machine learning model, like a CNN, with those features. They will be trained to identify whether the disease exists and of which type. It will be well designed to operate efficiently in real-time and immediately provide feedback as well as suggest treatment for the farmers. This computerized process aims to improve accuracy, save labour, and facilitate timely intervention, ultimately serving the agricultural community and enhancing farmers' economic stability.

2.4 SYSTEM SPECIFICATION

Once the analyst has gathered all information that is needed about the software to be built, and has eliminated all completeness, inconsistence, and anomalies from the specification, he begins to systematically arrange the requirements in the form of an SRS document. The software developers use the SRS document to ensure that they developed just what the customer needs. The SRS document assists the maintenance engineers in understanding the new system's functionality.

2.5 OPERATING SYSTEM

Windows 11 is the new major version of Microsoft's Windows NT operating system, announced on October 5, 2021. It replaced Windows 10 (2015) and is free to any Windows 10 devices that are qualified for the new Windows 11 system requirements. Windows 11 comes with a new user interface including a new Start Menu and Taskbar design, enhanced touch support, better security options, and built-in widgets for easy access to information. It also enhanced virtual desktops, gaming performance, and multitasking abilities. But it also encompasses a number of disadvantages like incompatible hardware, limited compatibility with legacy software, and fewer options for personalization.

2.6 LANGUAGE OR SOFTWARE PACKAGE

The project is implemented in Python 3.x as the base programming language. It makes use of the Flask web framework for creating the web-based interface and TensorFlow/Keras for deep learning-based plant disease classification. Image preprocessing is handled by NumPy and Pillow, and the storage of the trained model is taken care of by h5py. The database is managed with SQLite, with SQL Alchemy as the ORM for user authentication and class management. Also, Werkzeug is employed in safe password handling and file processing.

2.7 HARDWARE AND SOFTWARE SPECIFICATION

Frontend:

- HTML
- CSS
- JAVA SCRIPT

Backend:

- PYTHON

Operating System:

- Microsoft Windows 10 or above
- Browser: Google Chrome

Software Used:

- Flask
- TensorFlow
- NumPy
- Werkzeug

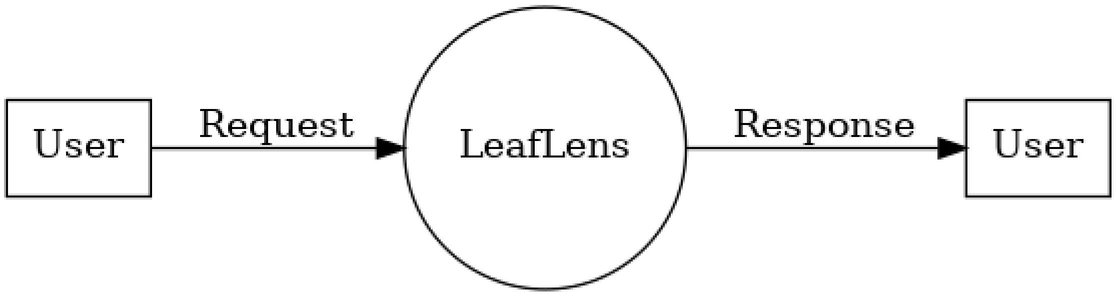
3.SYSTEM DESIGN

3.1INTRODUCTION

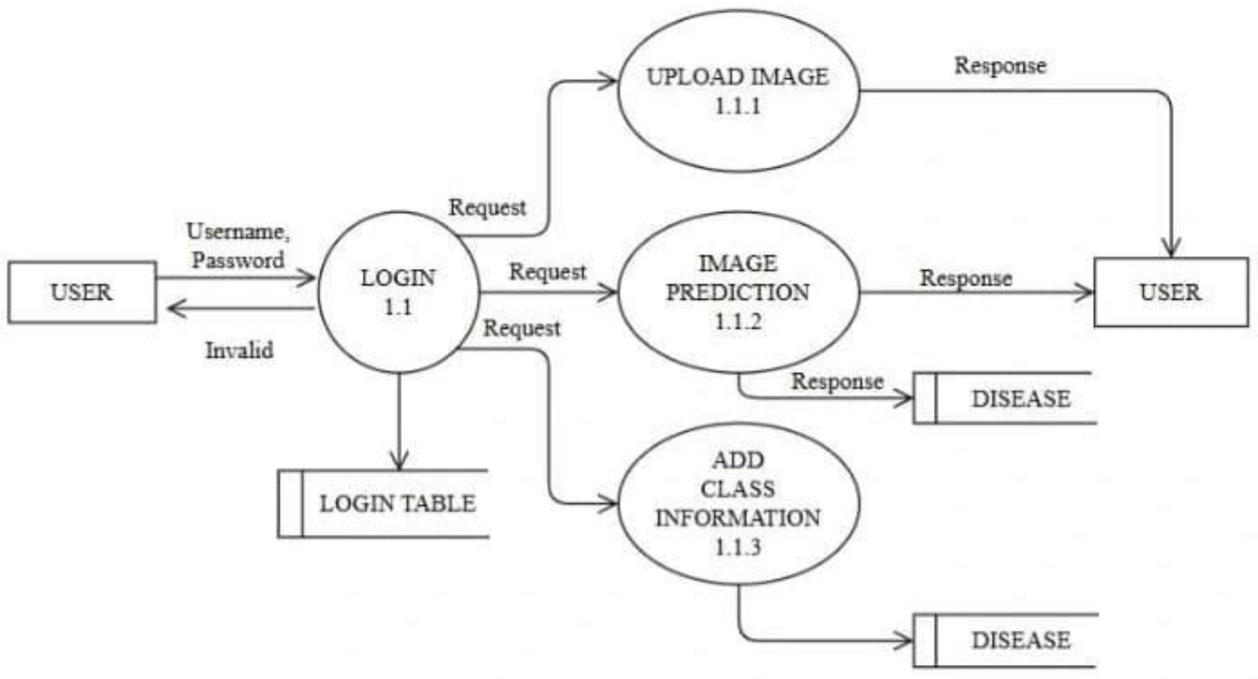
The design of the system for an automated plant disease detection system. The conventional approaches to the identification of plant diseases tend to be based on time-consuming and costly expert examination or laboratory testing. The available digital solutions tend to lack the efficiency and precision required for real-time diagnosis for various plant species. In response to these challenges, our suggested system employs advanced image processing and machine learning methodologies for the automated recognition and diagnosis of plant diseases in leaf images. This will seek to achieve fast, reliable, and cost-effective identification suitable for farmers, agriculture experts, and researchers alike, allowing early intervention and the mitigation of losses due to the destruction of crops.

3.2 DATA FLOW DIAGRAM

LEVEL 0



LEVEL 1.1



3.3 DATABASE DESIGN

Our database schema for the plant disease detection system is designed to store and handle the data required to train and run the model of machine learning efficiently. Our database design follows a relational database paradigm, with the data arranged in interconnected tables. One such important table contains details about the plant species, such as its scientific name, common names, and more. Another important table contains information regarding the various diseases such as disease name, symptoms description, and possible treatments. The main part of the database connects these two tables via a table containing labeled leaf images. Such a database design maintains data integrity, allows for quick data retrieval for model training and prediction, and provides room for future growth with the addition of more plant species and diseases to the system.

TABLE DESIGN

Table 1: LOGIN

Column Name	**Data Type**	**Constraints**	**Description**
User ID	Int	Primary Key	Key of the table.
User Name	Varchar	Not Null	Username of the user.
Password	Varchar	Not Null	Password of the user.

Table 2: DISEASE

Column Name	**Data Type**	**Constraints**	**Description**
Disease Name	Varchar	Not Null	Name of the disease.
Description	Text	Not Null	Description of the disease.

4.SYSTEM DEVELOPMENT

4.1 INTRODUCTION

The Plant Disease Detection Website is a web-based platform that is aimed at automating the identification and classification of plant diseases via machine learning and image processing technologies. The conventional process of plant disease detection, including manual inspection by agricultural specialists or laboratory testing, is time-consuming, costly, and susceptible to errors. This system offers a cost-effective, accurate, and real-time diagnosis solution that enables farmers, agricultural experts, and scientists to identify plant diseases quickly. By simply uploading images of the infected leaves of plants through the website, the users are provided with immediate results, which include the name of the disease, and disease description. The system seeks to improve early detection, minimize crop losses, and promote sustainable agriculture through the strength of artificial intelligence (AI) and deep learning.

4.2 PROCESS DESCRIPTION

This web application allows users to detect plant diseases from uploaded leaf images using a deep learning model. The process includes authentication, image upload, prediction, and data retrieval.

1. **User Authentication**
   - Users register with a username and securely hashed password.
   - Upon login, a session is created for navigation.
2. **Image Upload**
   - Users upload a leaf image via the Dashboard.
   - The image is stored in the static or uploads directory.
3. **Disease Prediction**
   - The image is processed using the **MobileNetV2** model.
   - It is resized (224x224) and normalized before classification.
4. **Results Display**
   - The predicted disease name is retrieved from a dictionary.
   - Additional details are fetched from the **Class Info** table.
   - Results (disease name, description, and image) appear on the Results Page.
5. **Managing Disease Data**
   - Admins or users can add/update disease descriptions.
   - Existing records are updated; new ones are inserted.
6. **Logout & Session Management**
   - Users can log out anytime; sessions expire after 10 minutes of inactivity.

5.SYSTEM TESTING AND IMPLEMENTATION

5.1 INTRODUCTION

Software testing is an integral component of software quality assurance and mark the final check of the specification, design, and coding. System testing assumes logically that every component of the system is correct; the objective will be attained successfully. Implementation enables the users to assume its operation for use and testing. Maintenance alters the current system, addition adds features to the current system, and development replaces the current system.

5.2 SYSYTEM IMPLEMENTATION

Implementation phase is the stage, where the process of transforming a new system design into an operational system takes place. It is the most important stage in obtaining a successful new system. Implementation is the stage if the project, where theoretical design is transformed into a functioning system. At this stage the core workload, the highest up heal and the substantial impact on current practices move to user department. If the implementation stage is if not planned and controllers with care, it can lead to chaos. Implementation stage is a system project. It requires meticulous planning, study of the existing system and its impediments on the implementation, design techniques to implement the changeover procedures, and assessment of change over techniques.

Implementation plan has the following steps:

- Testing the implemented system with the sample data.
- Finding and rectifying the errors.
- Implementing the necessary changes in the system.
- Training and engagement of user staff.
- Software utility installation.

5.3 DEBUGGING

Debugging is the activity of finding, examining, and correcting errors or bugs in software to provide seamless functionality. It encompasses methods such as code review, logging, breakpoints, and debugging tools to monitor problems in logic, syntax, or performance. Proper debugging improves software reliability and performance by identifying and solving problems early in development.

5.3.1 BLACKBOX TESTING

Black box testing is a software testing technique that tests a system's functionality without information about its internal code or design. Testers input and validate output to check if the system is according to specifications. It identifies functionality-related issues, UI issues, performance issues, and security issues with no programming skills and mimicking actual user actions.

For a plant disease classifier app built with Flask, black box testing entails checking for functionality without the use of internal code. Testers provide file uploads, user credentials, and class descriptions to test expected outcomes such as successful registration, login, file upload, and proper disease classification. Functional testing checks that routes are functioning properly, while UI testing verifies page rendering.

5.3.2 WHITEBOX TESTING

White box testing scrutinizes the internal logic, structure of code, and implementation of a system. In contrast to black box testing, it mandates programmers to evaluate code, decisions, loops, and data flow to identify bugs. Techniques used are unit testing, code coverage analysis, and control flow testing, which make the process efficient, secure, and defect-identifying early.

For a white box test of a Flask-based plant disease classifier, unit tests are performed on functions such as `predict_image()` to validate model loading and classification. Developers review database queries to validate proper data handling. Code coverage tests ensure all routes are covered, while debugging tools aid in analyzing authentication, session handling, and prediction accuracy.

5.3.3 SYSTEM SECURITY

System security is the practice and process of protecting computer systems, networks, and software from cyberattacks, unauthorized intrusion, and data breaches. System security involves the use of various security measures like encryption, authentication, firewalls, intrusion detection systems (IDS), and updates to software from time to time to secure confidential information and ensure the integrity, confidentiality, and availability of the system. Strong user authentication, secure coding, data encryption, and ongoing monitoring that can identify and mitigate possible attacks are all part of effective system security. Password hashing, session handling, and access control are some of the security controls used to protect against cyberattacks in web applications. SQL injection, cross-site scripting (XSS), and denial-of-service (DoS) attacks are some of the cyberattacks prevented by these security controls. A properly secured system not only safeguards user information and privacy but also provides trust, reliability, and adherence to security standards and regulations.

6.SCOPE OF FUTURE DEVELOPMENT

6.1 INTRODUCTION

The Plant Disease Detection Website is a major technological leap in agricultural diagnosis. Through the use of machine learning and image processing, the system enables users to detect plant diseases with accuracy and speed. Yet, as technology continues to evolve, there is still much room for growth and improvement. Future work can be directed towards making the system more accurate in disease classification, adding more plant species to be covered, incorporating real-time monitoring functions, and using predictive analytics to enable proactive decisions from farmers. With further refinement of the system, this project can help meaningfully in sustainable agriculture and food security.

6.2 MERITS OF THE SYSTEM

- Automated Disease Detection – The system reduces the need for manual examination, which is time and effort consuming.
- High Accuracy – Deep learning models, like CNN, improve the accuracy of detection.
- User-Friendly Interface – It is available to use via any web browser, and it's simple to utilize for farmers as well as agriculture specialists.
- Cost-Effective – In contrast to laboratory testing, the system offers a cost-effective method for detecting plant diseases.
- Quick Results – Users receive instant feedback, allowing them to take immediate action to prevent crop loss.

6.3 LIMITATIONS OF THE SYSYTEM

- Limited Dataset – The accuracy of the system is based on the diversity and quality of the training dataset. Some of the rare diseases may not be identified.
- Reliance on Image Quality – Bad lighting or unfocused images can result in erroneous predictions.
- No Real-Time Camera Support – At present, the system currently provides only image upload facilities, and there is no real-time detection from a live camera stream.
- Limited Disease Coverage – The system might not cover all plant diseases and species, necessitating periodic updates.
- Internet Dependence – As it is an online-based application, it needs to have a strong internet connection, which might not be feasible at all times in rural regions.

6.4 FUTURE ENHANCEMENT OF THE SYSTEM

- IoT and Drone Integration – Future releases can include IoT-based sensors and drones to obtain real-time photos from fields and upload them automatically for analysis.
- Live Camera Capability – The system can be designed to include real-time disease identification using live camera feeds rather than manual image uploading.
- Mobile App Development – A standalone mobile application can enhance convenience, enabling farmers to access the service offline and synchronize data when internet connectivity is possible.
- Multilingual Support – Including support for multiple languages can increase accessibility to a large variety of users.
- Predictive Analysis – Analyzing historical data and climatic conditions, the system can forecast the possibility of future outbreaks of disease, allowing preventive measures.
- Community and Expert Guidance – A capability where farmers can receive guidance from farming experts or converse with other members can increase learning.
- Treatment Recommendation Integration – More advanced AI models can recommend particular pesticides, fertilizers, or organic treatment based on disease detection.
- Better Deep Learning Models – Boosting the CNN model with bigger datasets and more complex architectures can increase precision and efficiency.

7.CONCLUSION

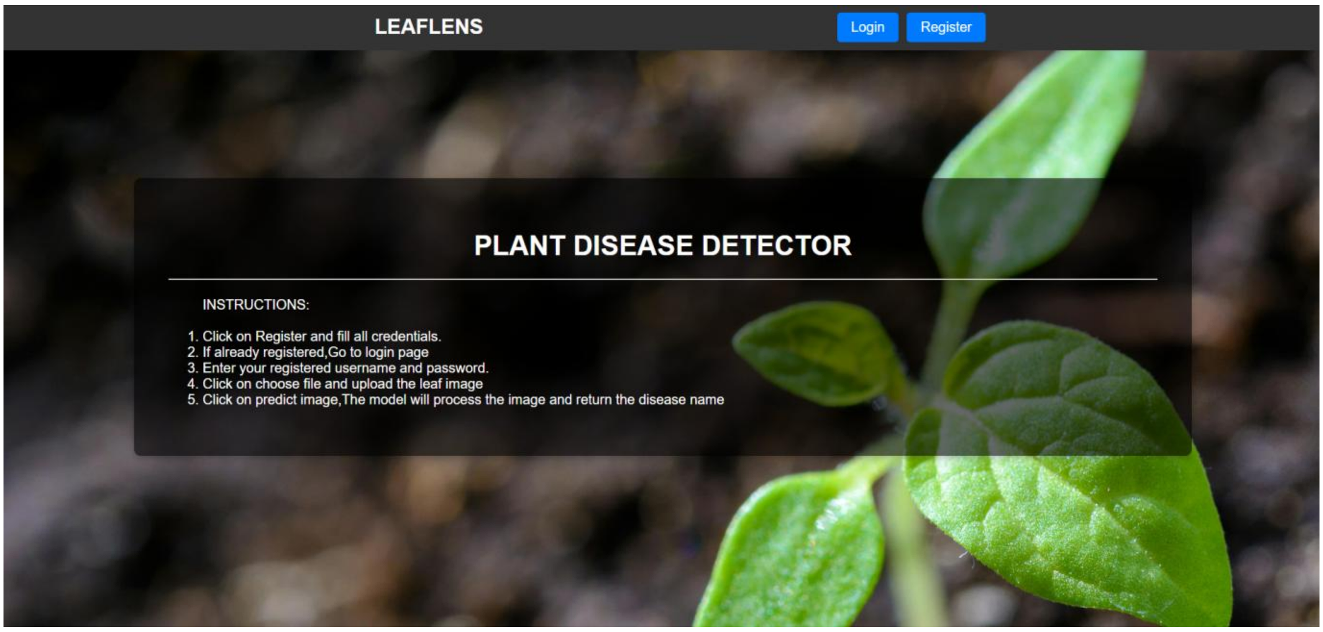
CONCLUSION

The Plant Disease Detection Website offers a contemporary and effective platform for plant disease identification by applying advanced machine learning and image processing methods. By allowing users to easily upload leaf images and obtain disease name and description, the site improves agricultural productivity and minimizes reliance on labor-intensive and time-consuming conventional approaches. This cutting-edge method not only enhances accuracy and accessibility but also enables farmers to act promptly, reducing loss and maximizing production. With ongoing learning from new data, the system adapts to identify an ever-expanding list of plant diseases, and it is a scalable and sustainable smart farming tool. The embedding of technology into farming through this platform is an important step in the direction of a data-centric and effective strategy for plant disease management.

8.APPENDIX

8.1 INPUT AND OUTPUT SCREEN

HOME PAGE

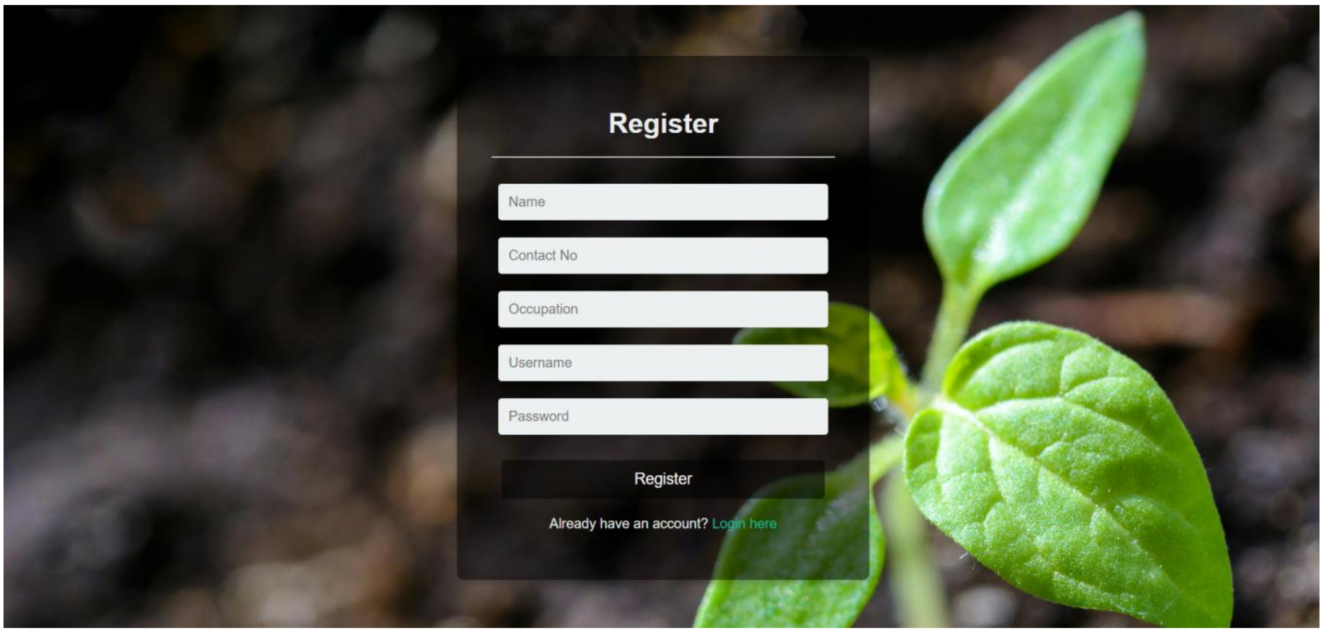


PLANT DISEASE DETECTOR

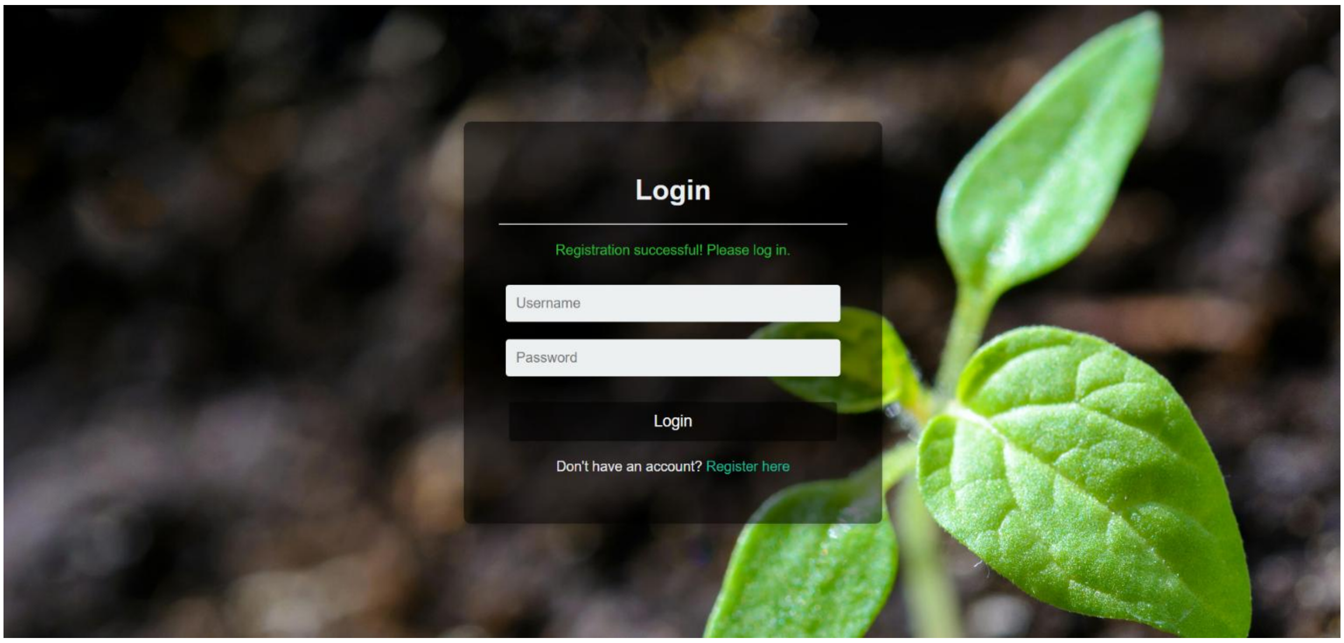
INSTRUCTIONS:

1. Click on Register and fill all credentials.
2. If already registered,Go to login page
3. Enter your registered username and password.
4. Click on choose file and upload the leaf image
5. Click on predict image,The model will process the image and return the disease name

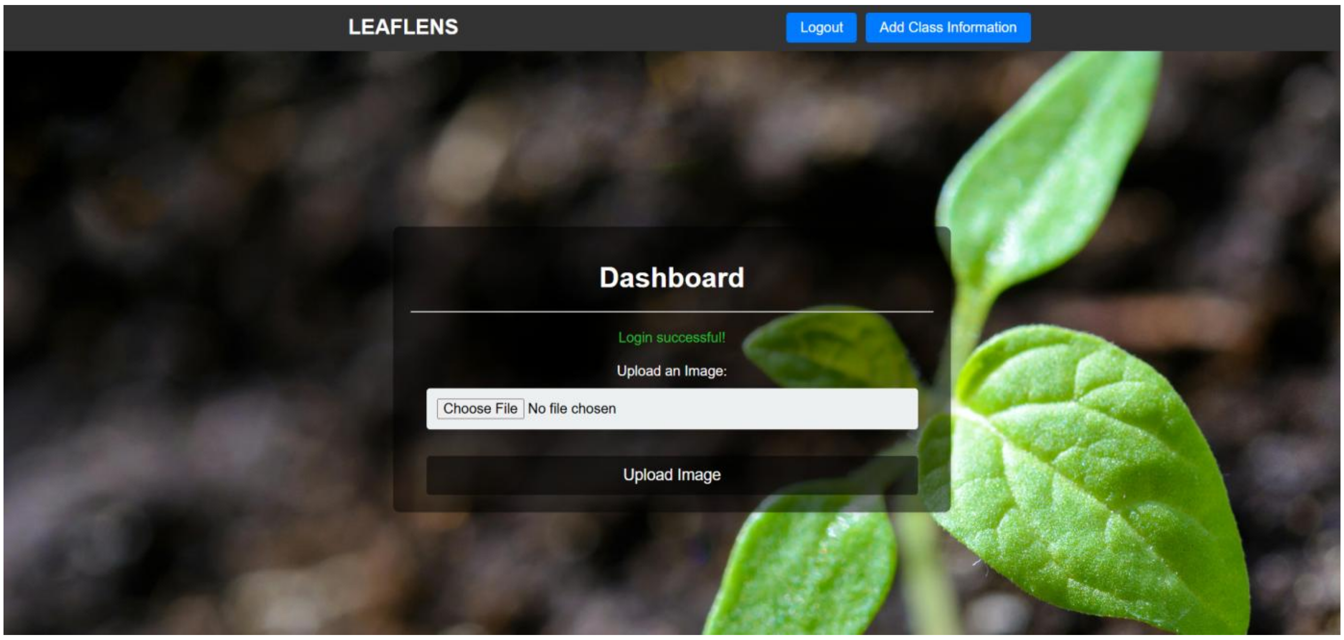
REGISTRATION PAGE

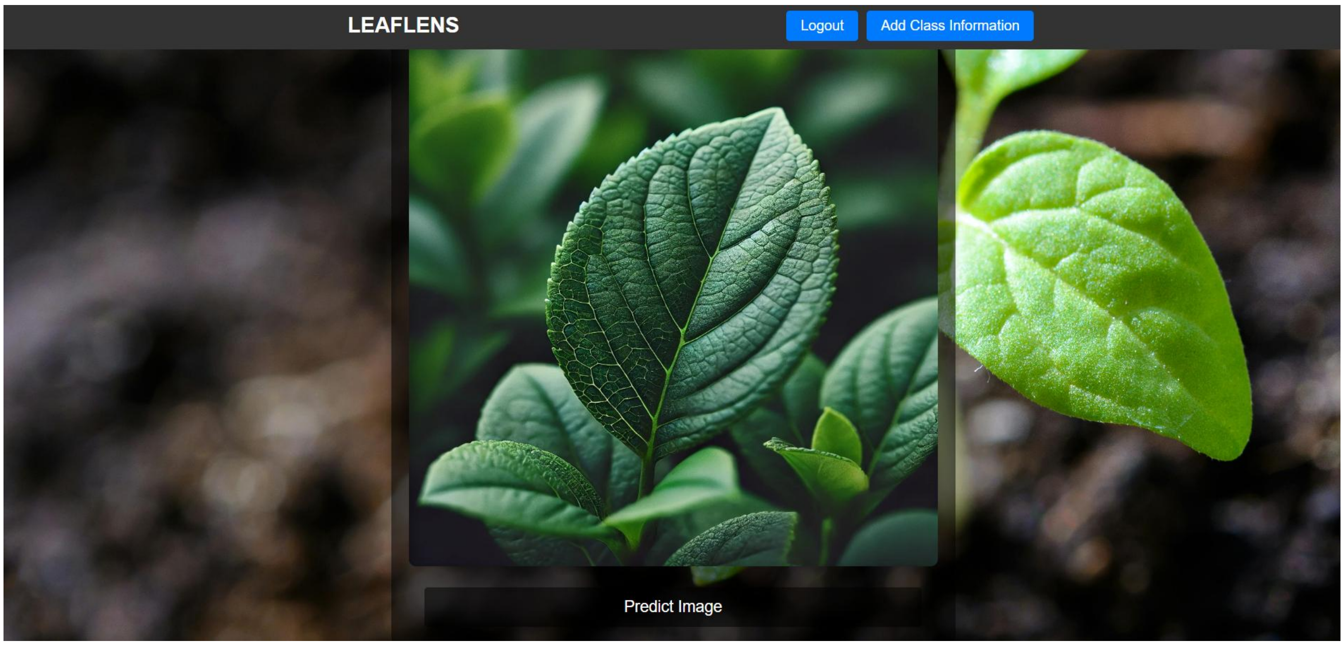


LOGIN PAGE

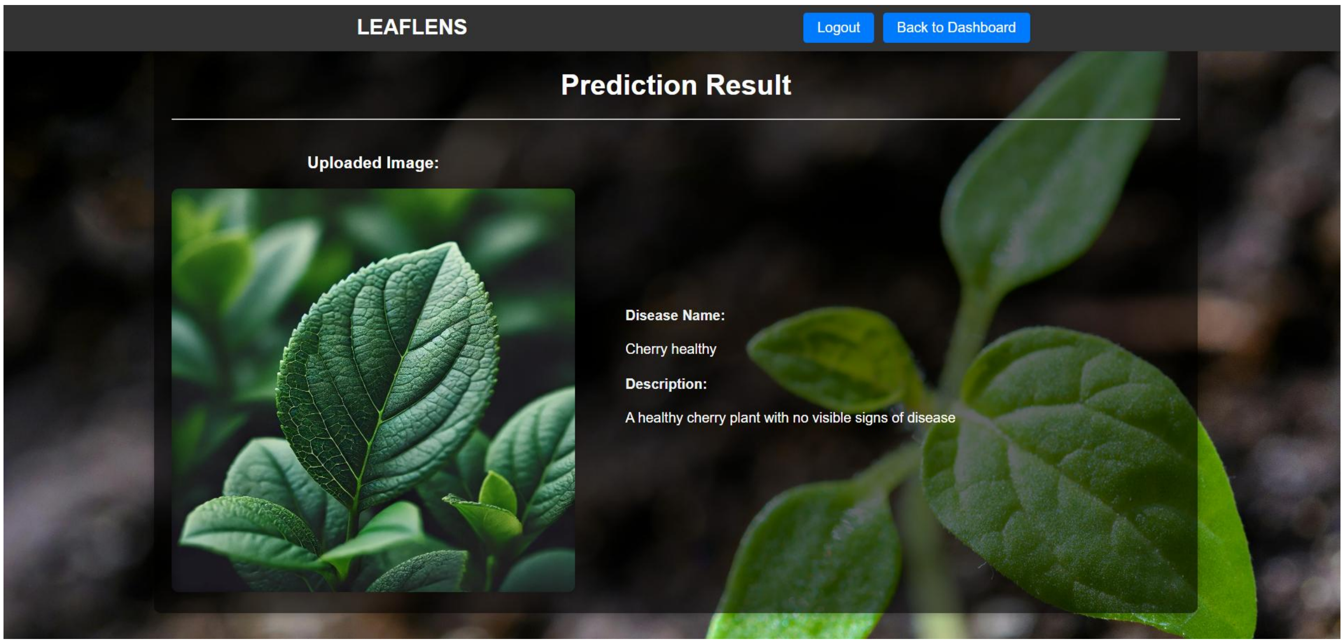

Login
Registration successful! Please log in.
Username
Password
Login
Don't have an account? Register here

UPLOAD PAGE





RESULT PAGE



Prediction Result

Uploaded Image:

Disease Name:

Cherry healthy

Description:

A healthy cherry plant with no visible signs of disease

8.2 SAMPLE CODE

```
# import the necessary packages

import os

import h5py

import numpy as np

import tensorflow as tf

from datetime import timedelta

from flask_sqlalchemy import SQLAlchemy

from werkzeug.utils import secure_filename

from tensorflow.keras.preprocessing.image import load_img, img_to_array # type: ignore

from werkzeug.security import generate_password_hash, check_password_hash

from flask import Flask, render_template, request, redirect, url_for, session, flash

# Global variables

folder_dict = {'0': 'Apple Apple scab', '1': 'Apple Black rot', '2': 'Apple Cedar apple rust', '3': 'Apple
healthy', '4': 'Blueberry healthy', '5': 'Cherry Powdery mildew', '6': 'Cherry healthy', '7': 'Corn
Cercospora leaf spot Gray leaf spot', '8': 'Corn Common rust', '9': 'Corn Northern Leaf Blight', '10':
'Corn healthy', '11': 'Grape Black rot', '12': 'Grape Esca', '13': 'Grape Leaf blight', '14': 'Grape
healthy', '15': 'Orange Haunglongbing', '16': 'Peach Bacterial_spot', '17': 'Peach healthy', '18':
'Pepper,_bell Bacterial spot', '19': 'Pepper,bell healthy', '20': 'Potato Early blight', '21': 'Potato Late
blight', '22': 'Potato healthy', '23': 'Raspberry healthy', '24': 'Soybean healthy', '25': 'Squash
Powdery mildew', '26': 'Strawberry Leaf scorch', '27': 'Strawberry healthy', '28': 'Tomato
Bacterial_spot', '29': 'Tomato Early_blight', '30': 'Tomato Late blight', '31': 'Tomato Leaf Mold',
'32': 'Tomato Septoria leaf spot', '33': 'Tomato Spider mites Two-spotted spider mite', '34': 'Tomato
Target Spot', '35': 'Tomato Tomato Yellow Leaf Curl Virus', '36': 'Tomato Tomato mosaic virus',
'37': 'Tomato healthy'}

# functions used in the app

def predict_image(filepath):

model = tf.keras.models.load_model('static/models/plant_disease_mobilenetv2.h5')

# Load and preprocess the image

image = load_img(filepath, target_size=(224, 224))

image_array = img_to_array(image) / 255.0

image_array = np.expand_dims(image_array, axis=0)

# Make a prediction

prediction = model.predict(image_array)

class_index = np.argmax(prediction)

return folder_dict[str(class_index + 1)]

# Initialize the Flask app

app = Flask(__name__)

app.secret_key = '2d6899c71c83a9c805ebb6d3aa5ceb0c'

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'

app.config['SQLALCHEMY_BINDS'] = {'classinfo': 'sqlite:///classinfo.db'}

app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=10)

db = SQLAlchemy(app)

# Configure the upload folder
```

```
UPLOAD_FOLDER = os.path.abspath('static/uploads')

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Classes

# User model

class User(db.Model):

id = db.Column(db.Integer, primary_key=True)

username = db.Column(db.String(80), nullable=False, unique=True)

password = db.Column(db.String(200), nullable=False)

# ClassInfo model for storing class names and disease info

class ClassInfo(db.Model):

__bind_key__ = 'classinfo'

id = db.Column(db.Integer, primary_key=True)

class_name = db.Column(db.String(100), nullable=False, unique=True)

description = db.Column(db.Text, nullable=False)

# Create the database tables

with app.app_context():

db.create_all()

# routes

# Home route

@app.route('/')
```

```
def home():

if 'user_id' in session:

return redirect(url_for('dashboard'))

return render_template('index.html')
```

REGISTRATION

```
def register():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

existing_user = User.query.filter_by(username=username).first()

if existing_user:

flash('Username already exists. Please choose a different one.')

return redirect(url_for('register'))

hashed_password = generate_password_hash(password, method='pbkdf2:sha256')

new_user = User(username=username, password=hashed_password)

db.session.add(new_user)

db.session.commit()

session.clear()

flash('Registration successful! Please log in.')

return redirect(url_for('home'))

return render_template('register.html')
```

LOGIN

```
# Login route

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form.get('username')

password = request.form.get('password')

# Find the user in the database

user = User.query.filter_by(username=username).first()

if user and check_password_hash(user.password, password):

session['user_id'] = user.id

session.permanent = True

flash('Login successful!')

return redirect(url_for('dashboard'))

else:

flash('Invalid credentials. Please try again.')

return redirect(url_for('login'))

return render_template('login.html')

# Register route

@app.route('/register', methods=['GET', 'POST'])
```

```
# Dashboard route

@app.route('/dashboard')

def dashboard():

if 'user_id' not in session:

flash('Please log in to access this page.')

return redirect(url_for('home'))

return render_template('dashboard.html')

# Route to add or update class information

@app.route('/add_class_info', methods=['GET', 'POST'])

def add_class_info():

if request.method == 'POST':

class_name = request.form['class_name']

description = request.form['description']

# Check if class name already exists

existing_class = ClassInfo.query.filter_by(class_name=class_name).first()

if existing_class:

# Update the existing class description

existing_class.description = description

db.session.commit()

flash(f'Class "{class_name}" updated successfully.')

else:
```

PREDICT DISEASE

```
@app.route('/dashboard/predict', methods=['POST'])
def predict():
    if 'user_id' not in session:
        flash('Please log in to access this page.')
        return redirect(url_for('home'))
    image_path = request.form['image_path'].replace('/static/', 'static/')
    # Make a prediction
    class_name = predict_image(image_path)
    # Fetch class info from the ClassInfo table
    class_info = ClassInfo.query.filter_by(class_name=class_name).first()
    if class_info:
        description = class_info.description
    else:
        description = "No information available for this class."
    image_path = '/' + image_path
    print(f"imagepath = {image_path}")
    # Redirect to the results page and pass the prediction and description
    return render_template('results.html', prediction=class_name, description=description,
image_path=image_path)
```

ADD NEW CLASS INFO

```
new_class = ClassInfo(class_name=class_name, description=description)

db.session.add(new_class)

db.session.commit()

flash(f'Class "{class_name}" added successfully.')

return redirect(url_for('dashboard'))

return render_template('addclassinfo.html', folder_dict=folder_dict)

# Upload route

@app.route('/dashboard/upload', methods=['POST'])

def upload():

if 'user_id' not in session:

flash('Please log in to access this page.')

return redirect(url_for('home'))

if 'file' not in request.files:

flash('No file part in the request.')

return redirect(url_for('dashboard'))

file = request.files['file']

if file.filename == '':

flash('No selected file.')return redirect(url_for('dashboard'))

if file:
```

```
filename = secure_filename(file.filename)

filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)

file.save(filepath)

flash('File uploaded successfully.')

image_url = url_for('static', filename=f'uploads/{filename}')

return render_template('dashboard.html', image_url=image_url)

return redirect(url_for('dashboard'))
```

9.BIBLIOGRAPHY

REFERENCE

1. **Plant Pathology-** 5th Edition- Agrios, G. N- 2005.
2. **Introductory Plant Pathology -**Chaube H. S, & Singh, S. K- 2001.
3. **Learning Python** -5th Edition- Lutz, M. (2013)
4. **Deep Learning**- Goodfellow, I., Bengio, Y., & Courville, A. -2016.
5. **Kaggle -** Plant Disease Detection Dataset- https://www.kaggle.com/datasets/vipoooool/leaf-disease-detection
6. **PlantVillage -** Plant Disease Identification- https://plantvillage.psu.edu/
7. **CNN-** https://en.wikipedia.org/wiki/Convolutional_neural_network