**Plant Disease Detection AI**

**Submitted for**

**Statistical Machine Learning CSET211**

Submitted by:

**(E23CSEU1080) Sumit Parashar**

Submitted to

**Mr. Amit Soni Arya**

**July-Dec 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sr.No | Content | Page No |
| 1. | **Introduction** |  |
| 2. | **Objectives** |  |
| 3. | **Technology Stack** |  |
| 4. | **Methodology** |  |
| 5. | **Implementation** |  |
| 6. | **Results** |  |
| 7. | **Challenges and Limitations** |  |
| 8. | **Future Work** |  |
| 9. | **Conclusion** |  |
| 10. | **References** |  |

**1. Introduction**

Agriculture is vital for food security and economic stability. However, plant diseases can severely impact crop yields. This project aims to develop an AI-powered system that detects plant diseases from images, providing farmers and agricultural experts with timely information to manage crop health effectively.

**2. Objectives**

1. To create a user-friendly interface for uploading plant images.
2. To develop a convolutional neural network (CNN) model that accurately classifies plant diseases.
3. To deliver results with actionable recommendations based on detected diseases.

**3. Technology Stack**

* **Frontend**: Streamlit for creating the user interface.
* **Backend**: TensorFlow for model training and predictions.
* **Model**: CNN model is used.
* **Deployment**: Streamlit for web-based access.

**4. Methodology**

**4.1 Data Collection**

Data was sourced from public datasets available on Kaggle.

**4.2 Model Development**

1. **Data Preprocessing**: Images were resized to 128x128 pixels and normalized for training.
2. **Model Training**: A CNN was built and trained on the dataset, achieving an accuracy of over 97%.
3. **Model Saving**: The trained model was saved in the Keras format for easy loading during predictions.

**4.3 Application Development**

* **User Interface**: Built using Streamlit, allowing users to upload images and receive disease predictions.
* **Image Prediction Function**: A function was created to handle image uploads, preprocess the images, and predict using the trained model.

**4.4 Deployment**

The application was deployed using Streamlit, allowing users to access the disease detection system via a web browser.

**5. Implementation**

The following code snippet outlines the key components of the application:

import streamlit as st

import tensorflow as tf

import numpy as np

# Background styling

page\_bg\_img = """

<style>

[data-testid="stAppViewContainer"]{

background-size: cover;

}

</style>

"""

st.markdown(page\_bg\_img, unsafe\_allow\_html=True)

# Function for model prediction

def model\_prediction(test\_image):

model = tf.keras.models.load\_model("trained\_plant\_disease\_model.keras")

image = tf.keras.preprocessing.image.load\_img(test\_image, target\_size=(128, 128))

input\_arr = tf.keras.preprocessing.image.img\_to\_array(image)

input\_arr = np.array([input\_arr])

predictions = model.predict(input\_arr)

return np.argmax(predictions)

# Sidebar and main page layout

st.sidebar.title("Dashboard")

app\_mode = st.sidebar.selectbox("Select Page", ["Home", "Disease Detection"])

if app\_mode == "Home":

st.title("Plant Disease Detection")

st.write("Upload an image of a plant to detect diseases.")

elif app\_mode == "Disease Detection":

test\_image = st.file\_uploader("Upload Image", type=["jpg", "png"])

if st.button("Predict"):

result\_index = model\_prediction(test\_image)

class\_names = [...] # List of class names

st.success(f"Prediction: {class\_names[result\_index]}")

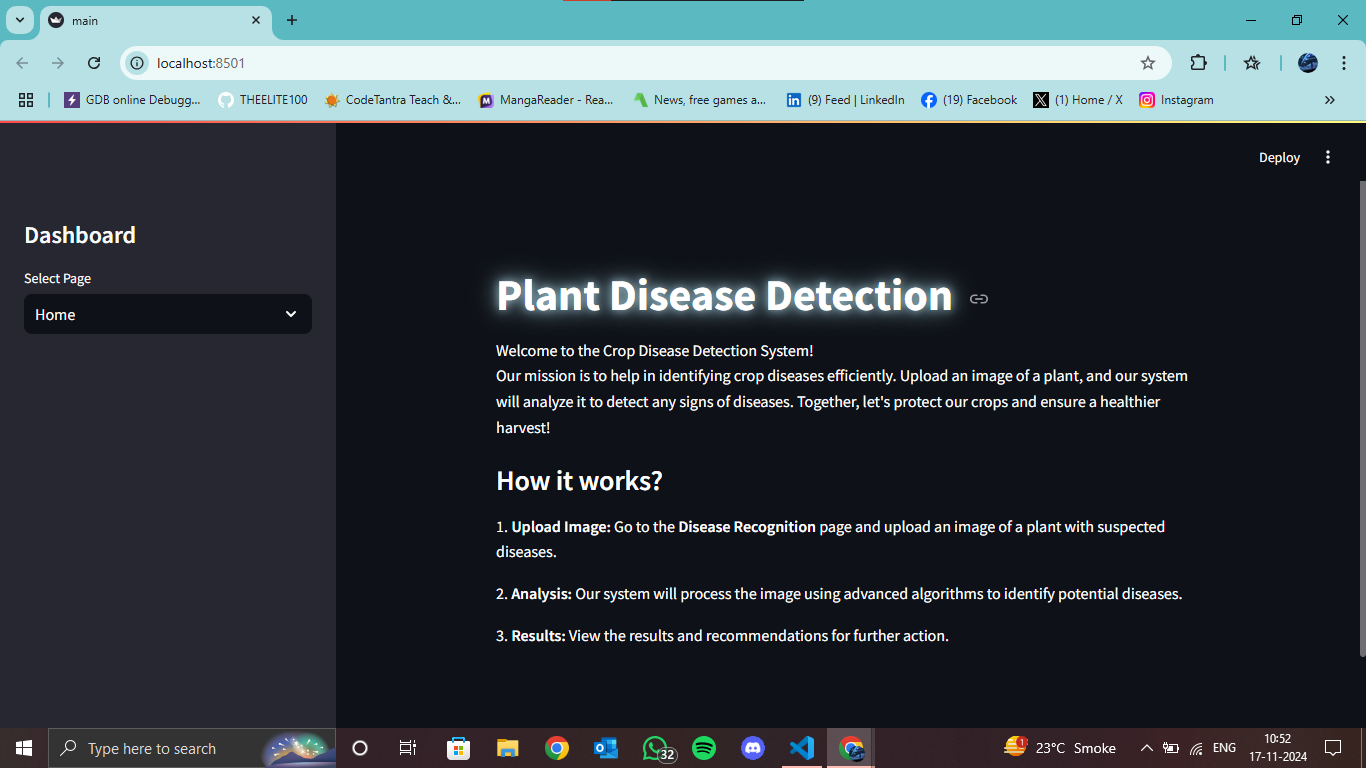
**6. Hardware/Software Required:**

**The code can be run in any Python IDE(VS Code is most suitable) and you need Streamlit, Tensorflow and numpy installed.**

**6. Results**

The application effectively detects a variety of plant diseases. Users can upload images, and the system provides instant feedback on the health of the plant along with the predicted disease name.

HomePage:



**DashBoard:**

**A screenshot of a computer

Description automatically generated**

**OutPut:**

**A screenshot of a computer

Description automatically generated**

**7. Challenges and Limitations**

* **Model Accuracy**: While the model performs well on the training dataset, real-world accuracy may vary.
* **Dataset Diversity**: Limited representation of certain diseases could impact prediction reliability.
* **User Experience**: Further enhancements are needed to make the interface more intuitive.

**8. Future Work**

* **Model Improvement**: Continuously refine the model with more diverse data and advanced techniques.
* **Mobile Application**: Develop a mobile version for easier access by farmers in the field.
* **Recommendation System**: Implement a system that suggests treatments based on detected diseases.

**9. Conclusion**

The Plant Disease Detection AI project successfully demonstrates the potential of machine learning in agriculture. By providing quick and reliable disease diagnosis, the system can aid in improving crop health management and food security.

**10. References**

* Kaggle Dataset: [New Plant Diseases Dataset](https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset)
* TensorFlow Documentation
* Streamlit Documentation

11. **GitHub Link of Your Complete Project**

**Link: https://github.com/THEELITE100/Plant-Disease-Detection**