# DEEP LEARNING MODEL FOR DETECTING DISEASES IN TEA LEAVES

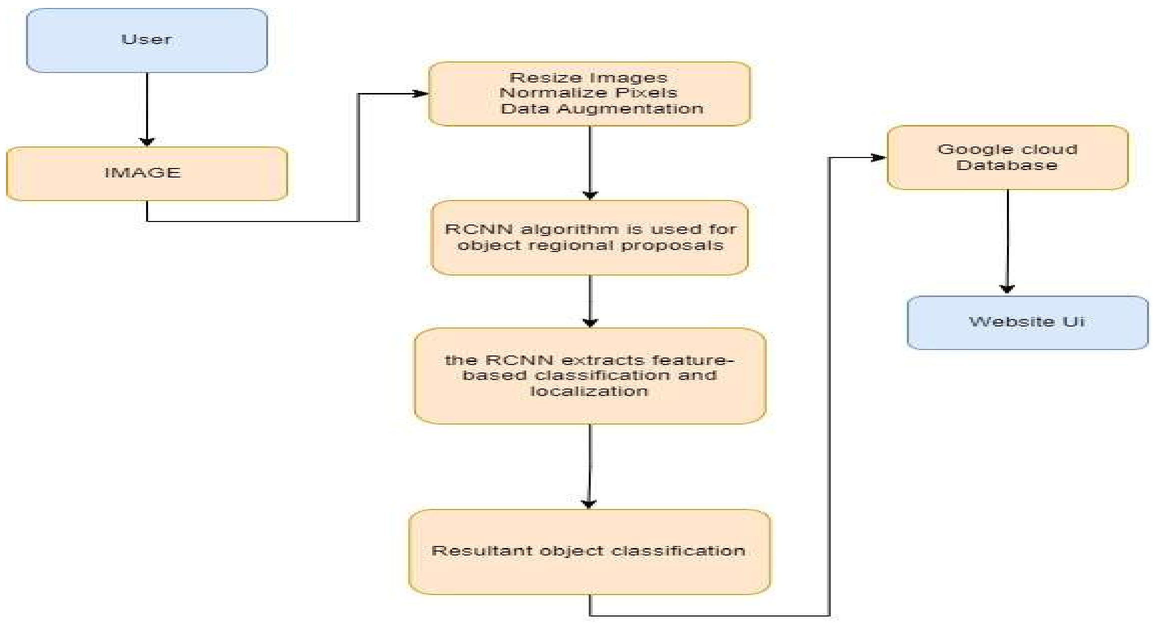
## Project Description:

The objective of this project is to develop a robust deep learning model capable of accurately classifying diseases in tea leaves based on various critical factors. Through a comprehensive analysis of color and other key features, the aim is to detect leaf diseases at an early stage, thus facilitating prompt diagnosis, treatment, and preventive measures to curb the spread of these ailments.

The dataset employed for this project comprises a rich collection of images, encompassing all types of diseases and their various stages, from the initial symptoms to the advanced phases. The overarching goal is to construct a sophisticated deep learning model that can not only identify the presence of diseases but also offer practical solutions to farmers. This assistance is crucial in helping them minimize crop damage and enhance the overall quality of their tea crops, ultimately fostering a healthier and more prosperous tea industry.

By harnessing the power of deep learning, this project strives to empower tea farmers with a cutting-edge tool that can detect diseases in their crops with precision and timeliness. This proactive approach not only safeguards their livelihoods but also contributes to the sustainability and growth of the tea production sector, ensuring a steady supply of high-quality tea leaves to consumers around the world.

## Technical Architecture:


User
IMAGE
Resize Images
Normalize Pixels
Data Augmentation
RCNN algorithm is used for object regional proposals
the RCNN extracts feature-based classification and localization
Resultant object classification
Google cloud Database
Website Ui

## Prerequisites:

To complete this project, you must require following software's, concepts and packages

- **Visual Studio**
- **Python packages:**
  - pip install tensorflow as tf
  - pip install keras
  - from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense
  - from tensorflow.keras.models import Model
  - from tensorflow.keras.optimizers import Adam
  - from tensorflow.keras.preprocessing.image import ImageDataGenerator
  - from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
  - pip install pandas" and click enter.
  - pip install scikit-learn" and click enter.
  - pip install matplotlib" and click enter.
  - pip install scipy" and click enter.
  - pip install pickle-mixin" and click enter.
  - pip install seaborn" and click enter.

## Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **DEEP-LEARNING Concepts**
  - Supervised learning:
  - RCNN
  - Evaluation metrics
- **WEB PAGE Concepts**
  - HTML
  - CSS
  - JAVASCRIPT
  - REACT

## Project Objectives:

By the end of this project, you will:

- Understanding of deep learning techniques and their applications, specifically in the context of disease classification in tea leaves.
- The ability to analyze images effectively, extracting valuable information and features from them for disease detection.
- Build a deep learning model that can accurately classify different diseases in tea leaves, including various stages of infection.
- The knowledge and tools to identify diseases at an early stage, facilitating timely intervention to prevent the spread and minimize crop damage.
- Goal is to contribute to the well-being of farming communities by enabling early disease diagnosis and prevention, leading to increased crop yields and healthier tea production.

## Project Flow:

- User interacts with the UI to input Image.
- Entered image is analyzed by the RCNN model which is integrated.
- Once model analyses the input the disease Classification is showcased on the UI

To accomplish this, we must complete all the activities listed below,

- Data Collection:
  - Gather a diverse dataset of tea leaf images, covering various types of diseases and their different stages of development.

- Data Preprocessing:
  - Clean and prepare the dataset by resizing images, normalizing colors, and labeling them correctly.

- Data Augmentation:
  - Enhance the dataset by applying data augmentation techniques such as rotation, flipping, and brightness adjustments to increase model robustness.

- Model Selection:
  - Choose a deep learning architecture suitable for image classification, such as Convolutional Neural Networks (CNNs).

- Model Training:
  - Train the selected deep learning model on the preprocessed dataset. Fine-tune the

model's hyperparameters for optimal performance.

- Validation and Testing:
    - Evaluate the model's performance using validation data and fine-tune as needed. Test the model on unseen data to assess its real-world accuracy.

- Early Disease Detection:
    - Implement algorithms for early disease detection based on the model's predictions. Develop a system that can identify diseases in tea leaves at their initial stages.

- Solution Provision:
    - Create a component that provides actionable recommendations to farmers based on disease diagnosis, suggesting appropriate treatments and preventive measures.

- User Interface:
    - Develop a user-friendly interface for farmers to interact with the system, making it accessible and easy to use in the field.

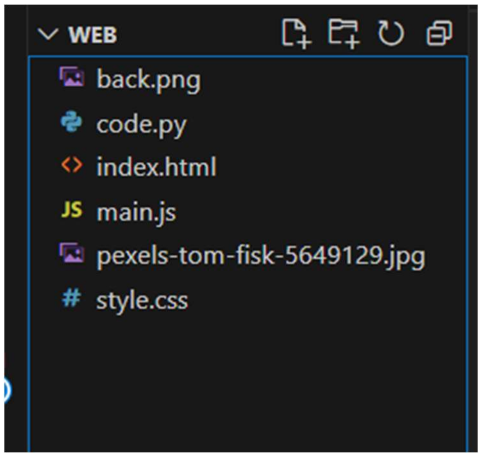
- Deployment:
    - Deploy the system in tea farms, either as a mobile application or a web service, ensuring it reaches the target users.

- Farmer Training:
    - Conduct training sessions for tea farmers to educate them on how to use the system effectively for disease detection and solution implementation.

- Monitoring and Feedback:
    - Continuously monitor the system's performance and gather feedback from users to make necessary improvements and updates

## Project Structure:

Project folder contains files as shown below



WEB
back.png
code.py
index.html
main.js
pexels-tom-fisk-5649129.jpg
style.css

- We are building a web page using HTML, CSS, and JavaScript and integrating with python script app.py for scripting.
- Model.pkl is our saved model for processing the image.
- Data folder contains dataset and  folder contains html, CSS and JavaScript files and python file

## Milestone 1: Problem Understanding

### Activity 1: Specify the business problem

The problem statement for Deep Learning Model for Detecting Diseases in Tea Leaves is to creative classification model that will accurately what type of disease it has and what precaution, prevention and cure method that will help in early diagnosis and stopping it form further spreading.

### Activity 2: Business requirements

Here are some potential business requirements for Share price predictor.

a. <u>Accurate forecasting</u>: The classification model must be able to accurately forecast disease for a given information of the leaf The accuracy of the forecastingcrucial for diagnosis

b. <u>Real-time data acquisition</u>: The predictor must be able to acquire real-time data information and other relevant factors that will help in identification for leaf disease. The data acquisition must be seamless and efficient to ensure that the predictor always get the most accurate result.

c. User-friendly interface: User have a user-friendly interface that is easy to navigate and understand. The interface shows the results of the user in a clear and concise manner to enable farmer to take further action

d. Report generation: Generating the outlining the predicted disease in leaf in a particular region, along with an analysis of the factors that have contributed to their performance. The output not only show what kind of disease but also the prevention and precaution and cure method

**Activity 3: Literature Survey**

In order to ensure the prevention and management of tea leaf diseases, it is essential to have a reliable and accurate detection and identification system. Currently, the detection of tea leaf diseases is conducted manually, which can lead to delays in the detection of the disease and can have a negative impact on the quality and productivity of the yield.

[1] Soeb, M.J.A., Jubayer, M.F., Tarin, T.A., Al Mamun, M.R., Ruhad, F.M., Parven, A., Mubarak, N.M., Karri, S.L. and Meftaul, I.M., 2023. Tea leaf disease detection and identification based on YOLOv7 (YOLO-T). Scientific reports, 13(1), p.6078.
This study seeks to address this issue by introducing an artificial intelligence-based solution to the challenge of detecting tea leaf disease. The study trains the fastest single-stage object detection model (YOLV7) on a dataset of diseased tea leaves collected from four major tea gardens located in Bangladesh. The dataset consists of 4000 digital images depicting five different types of leaf disease, which are manually annotated and presented to the user. Data augmentation approaches are also included to address the issue of inadequate sample sizes. Results of the study include 97.3% detection accuracy, 96.7% precision, 96.4% recall, 98.2% mAP value, and 0.

[2] Lanjewar, M.G. and Panchbhai, K.G., 2023. Convolutional neural network-based tea leaf disease prediction system on smartphone using PaaS cloud. *Neural Computing and Applications*, *35*(3), pp.2755-2771.
This research studies the development and implementation of a disease prediction system in real time using a Convolutional Neural Network (CNN) on a PaaS cloud platform. CNN is used to forecast tea leaf disease, achieving a 100% accuracy rate for training and validation, as well as for test datasets. Additionally, the same dataset is applied to the Deep Convolutional Networks (DCNNs) like ResNet50 and Xception, as well as NASNetMobile, for tea leaf

disease prediction. All models were evaluated using the confusion matrix, as well as the K-fold Cross-validation method. Comparative results demonstrate that CNN outperforms DCNNs and literature-based models in terms of its remarkable accuracy. Upon successful deployment, the model hyperlink is available on the PaaS cloud. The smartphone can be used to capture the tea leaf image and upload it to the cloud, allowing the cloud system to automatically predict the disease and display it on the mobile display.

[3] Bhagat, M. and Kumar, D., 2023. Performance enhancement of kernelized SVM with deep learning features for tea leaf disease prediction. *Multimedia Tools and Applications*, pp.1-18. Because there are so few photos of tea leaves, categorization is quite challenging. The overfitting of the model happens very often. In order to deal with this, an image augmentation technique was use in this work that resulted in a dataset that was roughly fourteen times larger. However, this quantity of datasets is still insufficient for classification using deep learning. Thus, machine learning-based classification was done here, while deep learning was used for feature extraction. In order to achieve superior classification results, we have suggested a hybrid technique in this work that combines a machine learning-based classifier with deep learning-based characteristics of the augmented dataset. The suggested method uses the tea leaf dataset's color features to extract them using VGG-16. This feature is used to build the model, and various machine learning-based classifiers are used for the classification task, including kernelized SVM, Random Forest, XGB, and KNN. The suggested model used VGG-16 features and SVM and linear kernel-based SVM to get the best classification accuracy. The suggested model has a 96.67% accuracy rate.

[4] Mukhopadhyay, S., Paul, M., Pal, R. and De, D., 2021. Tea leaf disease detection using multi-objective image segmentation. *Multimedia Tools and Applications*, *80*, pp.753-771. This suggested method uses the tea leaf dataset's color features to extract them using VGG-16. The VGG16 model, which has a deep architecture with 16 weight layers, uses max-pooling layers and a convolutional neural network (CNN) structure with 3x3 convolutional filters to extract features from input data. This allows the model to learn complex and abstract features. This feature is used to build the model, and various machine learning-based classifiers are used for the classification task, including kernelized SVM, Random Forest, XGB, and KNN. The suggested model used VGG-16 features and SVM and linear kernel-based SVM to get the best classification accuracy. This model has a 96.67% accuracy rate.

**Activity 4: Social or Business Impact.**

Social Impact:

- Improved Food Security: By helping tea farmers detect diseases early and providing solutions, the project contributes to food security by safeguarding tea crop yields and ensuring a consistent supply of tea products.

- Healthier Tea Production: The quality of tea leaves is crucial for producing high-quality tea products. Disease control results in healthier tea plants, positively impacting the overall quality of the tea available to consumers.

- Sustainable Agriculture: By reducing the need for extensive chemical treatments and minimizing crop damage, the project promotes more sustainable and environmentally friendly agricultural practices.

Business Impact:

- Increased Crop Yields: Disease management through early detection and solutions leads to increased crop yields, benefiting both small and large tea farms. Higher yields translate to higher revenues.
- Quality Assurance: Consistently high-quality tea leaves enhance the reputation of tea brands, attracting consumers willing to pay a premium for quality tea products.
- Supply Chain Resilience: The project promotes a more resilient and reliable supply chain, reducing supply chain disruptions caused by disease outbreaks.

## Milestone 2: Data Collection

Deep learning model depends heavily on data, it is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

**Activity 1: Download the dataset**

In this project, we have used tea leaf disease classification. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.
Link: https://www.kaggle.com/datasets/shashwatwork/identifying-disease-in-tea-leafs

As the dataset is downloaded, let us read and understand the data properly

**Activity 2: Importing the libraries**

Import the libraries:

import tensorflow as tf

```
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
```

**Activity 3: Read the Dataset**

Our dataset format be in jpg format

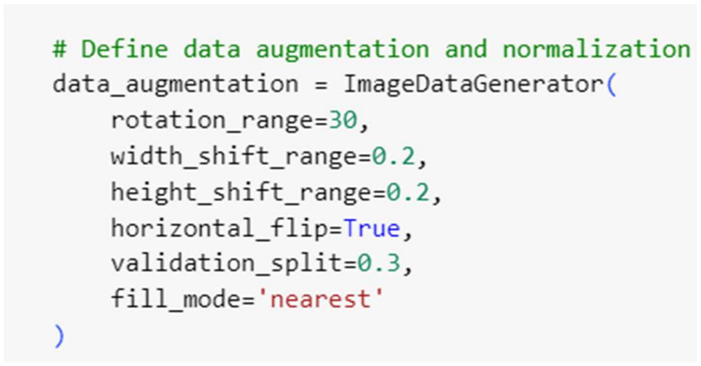
## Milestone 3: Data Pre-processing

### Activity 1: Data Annotation:

Annotate the dataset by labeling each image with the corresponding disease class (e.g., "Healthy," "Rust," "Blister Blight," etc.). Accurate labeling is essential for supervised learning.

### Activity 2: Image Resizing:

Resize all images to a uniform size, typically the input size expected by the deep learning model. This ensures consistency and reduces computational overhead.
Data Augmentation:
Apply data augmentation techniques to increase the dataset's size and diversity. Techniques like rotation, flipping, cropping, and color jittering can help the model generalize better.



```
# Define data augmentation and normalization
data_augmentation = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    validation_split=0.3,
    fill_mode='nearest'
)
```

### Activity 3: Normalization:

Normalize the pixel values of the images to a common scale (e.g., [0, 1] or [-1, 1]). Normalization improves model convergence and stability.

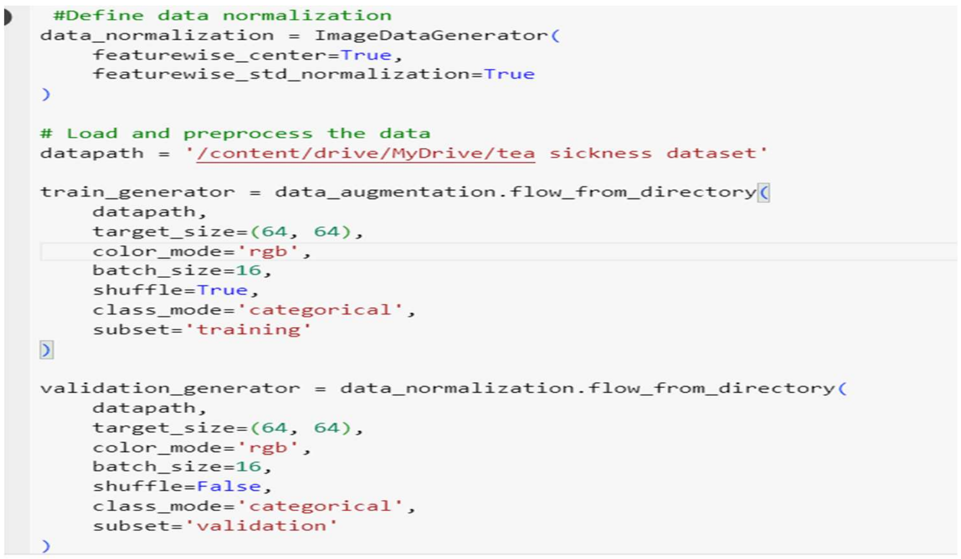
### Activity 4:  Handling Class Imbalance:

Address any class imbalance issues by oversampling or under sampling, or by using techniques like class-weighting during training.

### Activity 5: Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the dataset

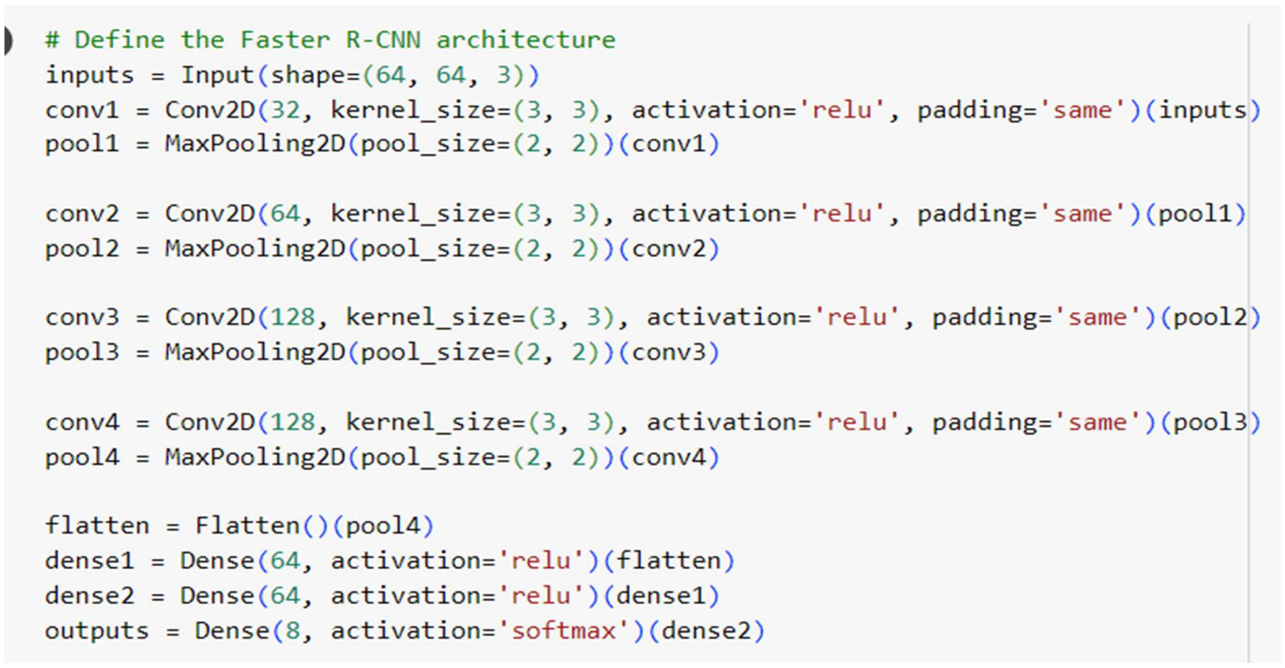
Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function fromsklearn. As paranmeters, we are passing x, y, test_size, random_state.



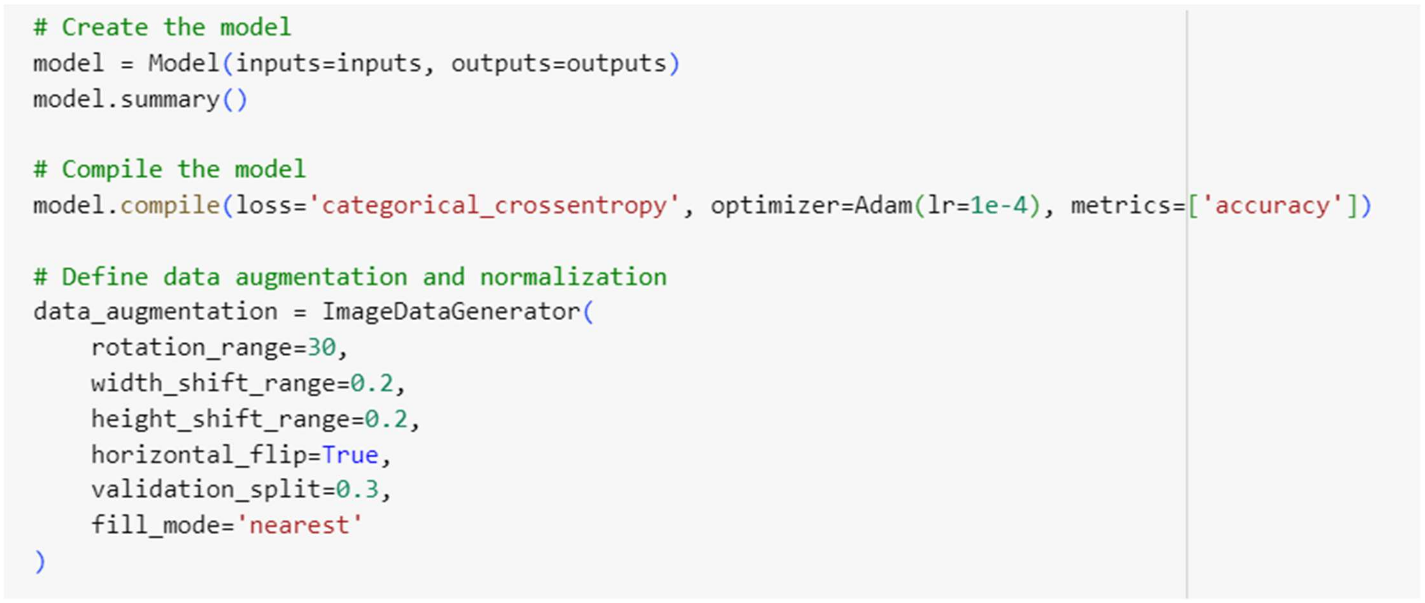
```
#Define data normalization
data_normalization = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True
)

# Load and preprocess the data
datapath = '/content/drive/MyDrive/tea sickness dataset'

train_generator = data_augmentation.flow_from_directory(
    datapath,
    target_size=(64, 64),
    color_mode='rgb',
    batch_size=16,
    shuffle=True,
    class_mode='categorical',
    subset='training'
)

validation_generator = data_normalization.flow_from_directory(
    datapath,
    target_size=(64, 64),
    color_mode='rgb',
    batch_size=16,
    shuffle=False,
    class_mode='categorical',
    subset='validation'
)
```

## Milestone 4: Model Building

Now our data is cleaned and it’s time to build the model. We will train our data on RCNN algorithms. Forthis project we creating our own RCNN architecture. The model is saved based on its performance.



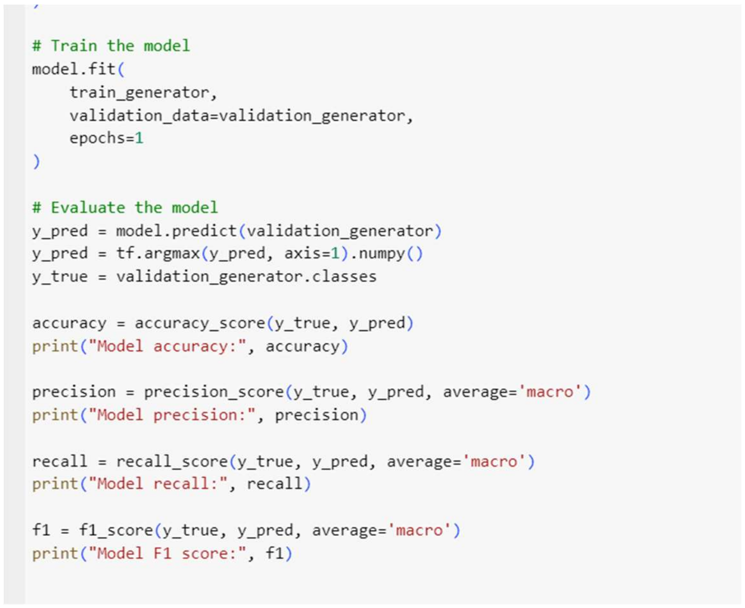
```
# Define the Faster R-CNN architecture
inputs = Input(shape=(64, 64, 3))
conv1 = Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same')(inputs)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

conv2 = Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same')(pool1)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

conv3 = Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same')(pool2)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

conv4 = Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same')(pool3)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

flatten = Flatten()(pool4)
dense1 = Dense(64, activation='relu')(flatten)
dense2 = Dense(64, activation='relu')(dense1)
outputs = Dense(8, activation='softmax')(dense2)
```



```
# Create the model
model = Model(inputs=inputs, outputs=outputs)
model.summary()

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=1e-4), metrics=['accuracy'])

# Define data augmentation and normalization
data_augmentation = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    validation_split=0.3,
    fill_mode='nearest'
)
```

**Activity 1: RCNN (Region-based Convolutional Neural Network)**

RCNN algorithm is initialized, and training data is passed to the model with. fit () function. Test data is predicted with. predict () function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.



```
# Train the model
model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=1
)

# Evaluate the model
y_pred = model.predict(validation_generator)
y_pred = tf.argmax(y_pred, axis=1).numpy()
y_true = validation_generator.classes

accuracy = accuracy_score(y_true, y_pred)
print("Model accuracy:", accuracy)

precision = precision_score(y_true, y_pred, average='macro')
print("Model precision:", precision)

recall = recall_score(y_true, y_pred, average='macro')
print("Model recall:", recall)

f1 = f1_score(y_true, y_pred, average='macro')
print("Model F1 score:", f1)
```

### Activity 2: Classifying the type of disease on tea leaf

Our model is performing well. So, we are saving the model by pickle.dump().

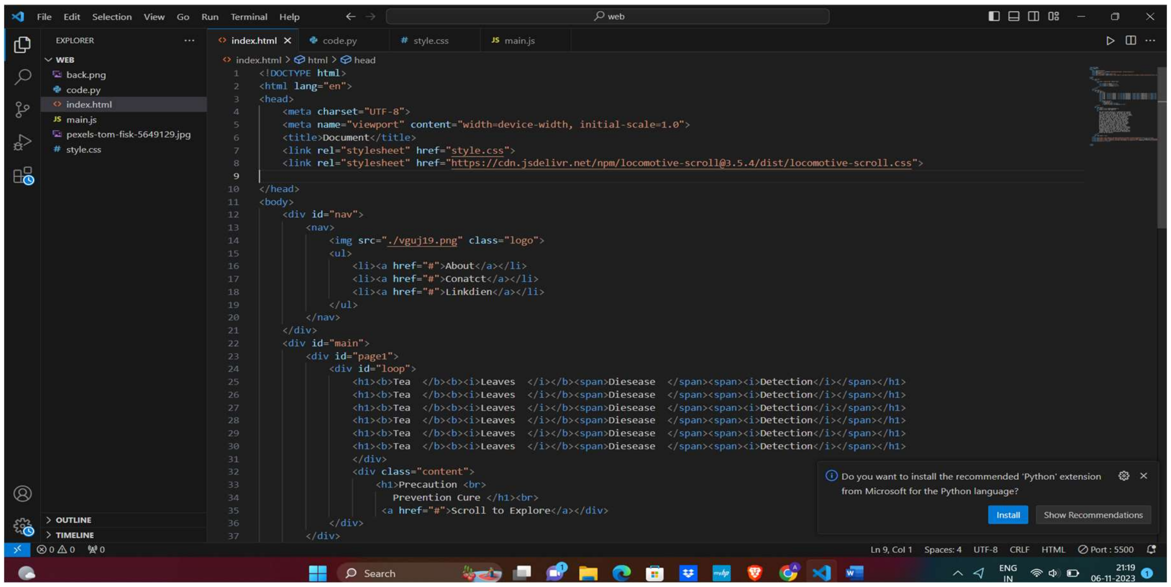
```python
# Specify the filename for the pickle file
pickle_file_name = 'model.pickle'

# Open the file in binary write mode (wb) to create the pickle file
with open(pickle_file_name, 'wb') as file:
    # Use the pickle.dump() function to serialize and write the model to the file
    pickle.dump(model, file)

# The model has been pickled and saved to 'model.pickle' in the current directory
```

## Milestone 5: Application Building

We will develop a web application that integrates with the previously created disease detection model. The application will feature a user interface where users can input an image. The uploaded image will be processed by the model, and the classified disease will be displayed on the user interface. This section comprises the following tasks:

### Activity 1: Building HTML Pages

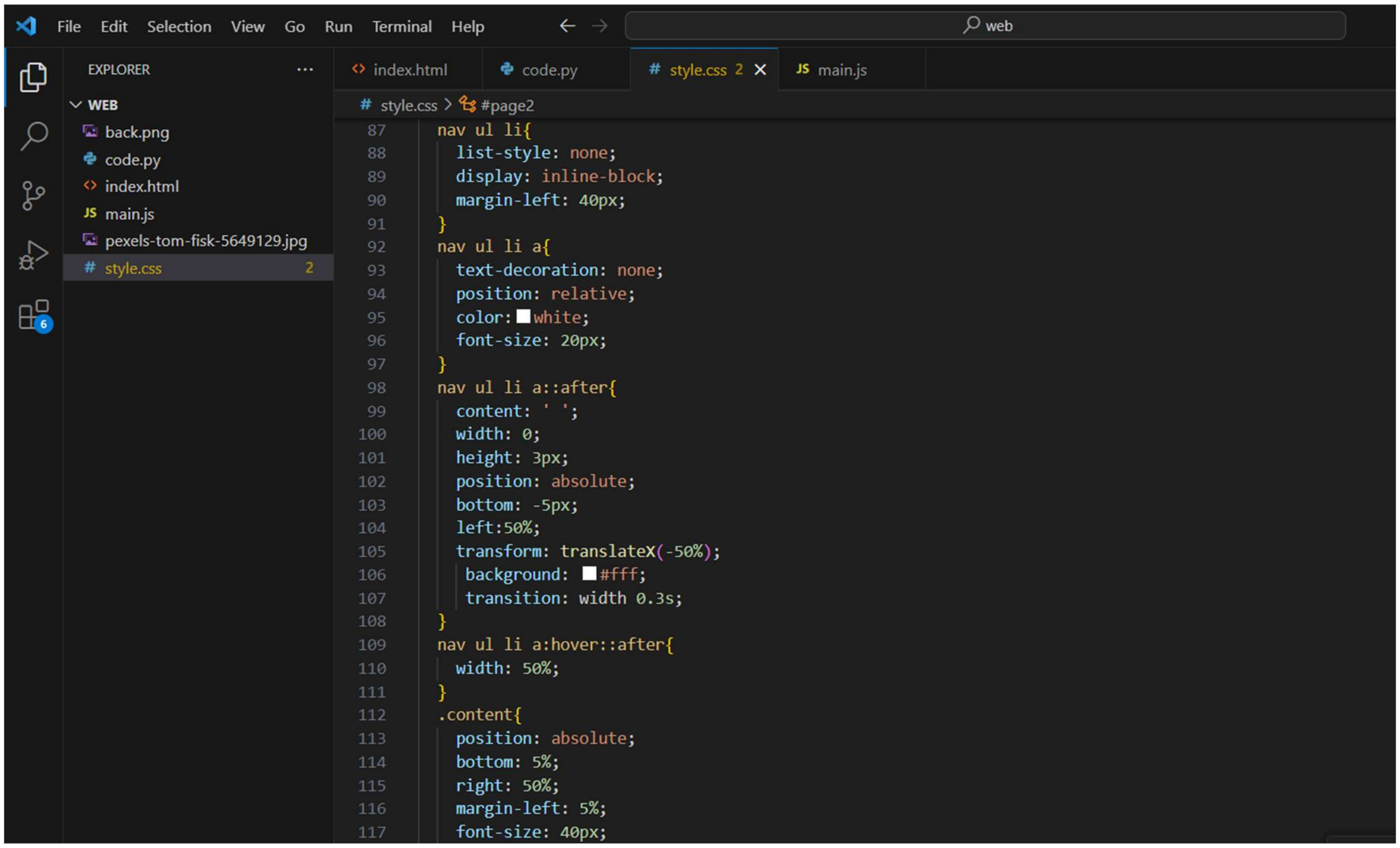
For this project, you need to create three HTML files and save them in a folder named "Templates." The HTML files are as follows:
Page1.html: The home page where users can upload an image.
Page2.html: A page that indicates image processing is in progress.
Page3.html: The result page showing the detected disease



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/locomotive-scroll@3.5.4/dist/locomotive-scroll.css">

</head>
<body>
    <div id="nav">
        <nav>
            <img src="./vguj19.png" class="logo">
            <ul>
                <li><a href="#">About</a></li>
                <li><a href="#">Conatct</a></li>
                <li><a href="#">Linkdien</a></li>
            </ul>
        </nav>
    </div>
    <div id="main">
        <div id="page1">
            <div id="loop">
                <h1><b>Tea  </b><b><i>Leaves  </i></b><span>Diesease  </span><span><i>Detection</i></span></h1>
                <h1><b>Tea  </b><b><i>Leaves  </i></b><span>Diesease  </span><span><i>Detection</i></span></h1>
                <h1><b>Tea  </b><b><i>Leaves  </i></b><span>Diesease  </span><span><i>Detection</i></span></h1>
                <h1><b>Tea  </b><b><i>Leaves  </i></b><span>Diesease  </span><span><i>Detection</i></span></h1>
                <h1><b>Tea  </b><b><i>Leaves  </i></b><span>Diesease  </span><span><i>Detection</i></span></h1>
                <h1><b>Tea  </b><b><i>Leaves  </i></b><span>Diesease  </span><span><i>Detection</i></span></h1>
                </div>
                <div class="content">
                    <h1>Precaution <br>
                      Prevention Cure </h1><br>
                     <a href="#">Scroll to Explore</a></div>
            </div>
        </div>
```

Do you want to install the recommended 'Python' extension from Microsoft for the Python language?

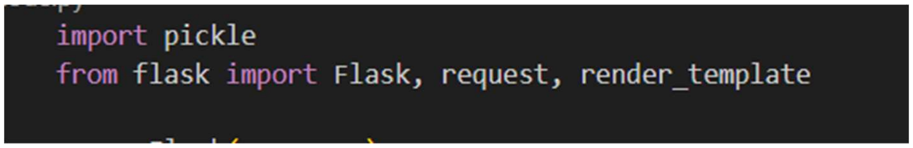
## CSS file for the pages:



```css
nav ul li{
  list-style: none;
  display: inline-block;
  margin-left: 40px;
}
nav ul li a{
  text-decoration: none;
  position: relative;
  color: white;
  font-size: 20px;
}
nav ul li a::after{
  content: ' ';
  width: 0;
  height: 3px;
  position: absolute;
  bottom: -5px;
  left:50%;
  transform: translateX(-50%);
   background: #fff;
   transition: width 0.3s;
}
nav ul li a:hover::after{
  width: 50%;
}
.content{
  position: absolute;
  bottom: 5%;
  right: 50%;
  margin-left: 5%;
  font-size: 40px;
```

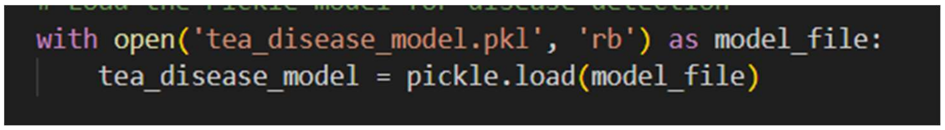
## Activity 2: Build Python code:

Import the libraries



```python
import pickle
from flask import Flask, request, render_template
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is ourWSGI application. Flask constructor takes the name of the current module (_name_) as argument.

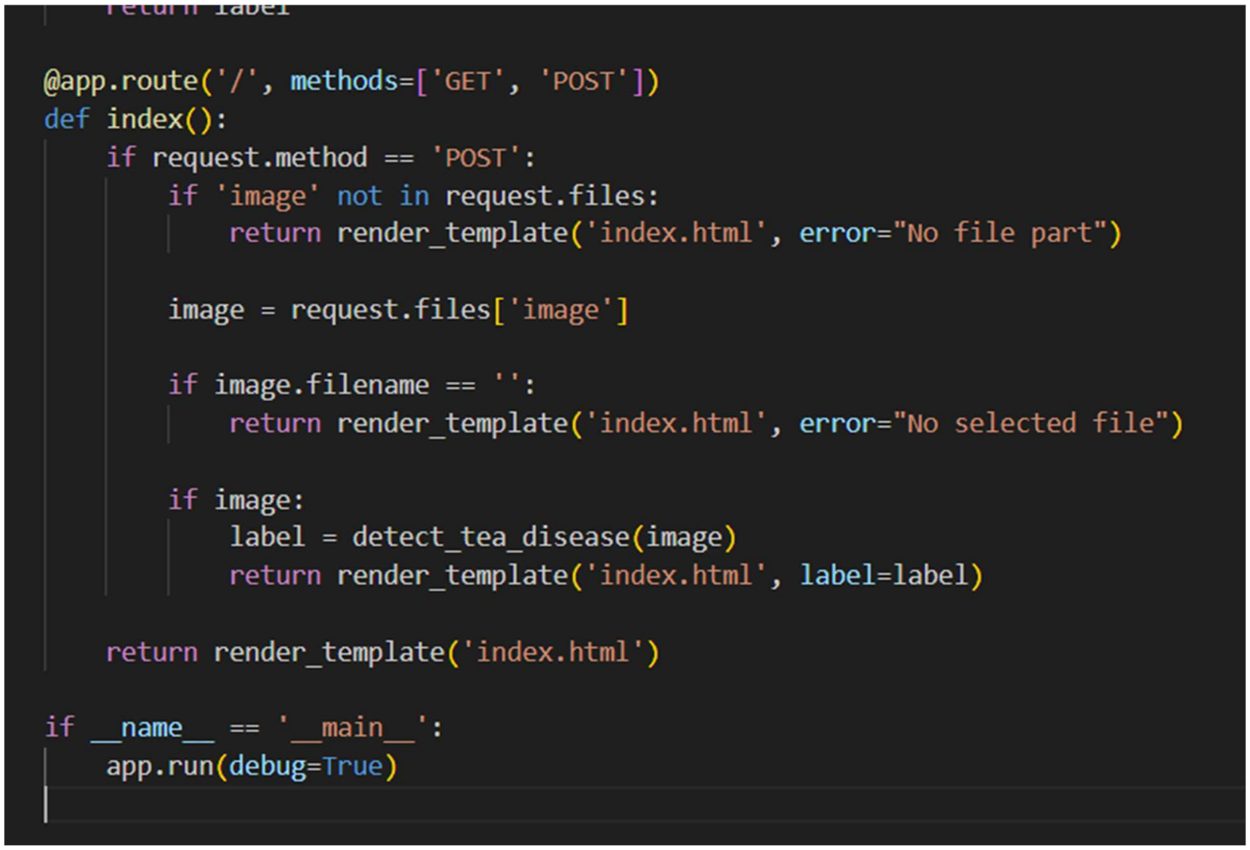


```python
with open('tea_disease_model.pkl', 'rb') as model_file:
    tea_disease_model = pickle.load(model_file)
```

Render HTML page Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the webserver is opened in browser, the html page will be rendered. Whenever you enter the values from the htmlpage the values can be retrieved using POST Method.

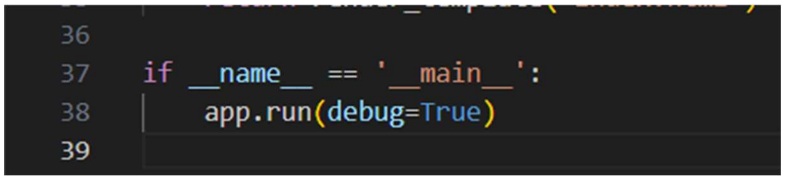
Retrieves the value from UI:



```
        return label

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        if 'image' not in request.files:
            return render_template('index.html', error="No file part")

        image = request.files['image']

        if image.filename == '':
            return render_template('index.html', error="No selected file")

        if image:
            label = detect_tea_disease(image)
            return render_template('index.html', label=label)

    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

In this section of the application, we direct our app to the predict() function. Within the predict() function, we collect all the data submitted from the HTML page through a POST request and store it in an array. This array serves as input to the model.predict() function. The model.predict() function processes the data and generates a prediction. Finally, this prediction is presented in the specified text element as previously defined in the submit.html page.

Main Function:

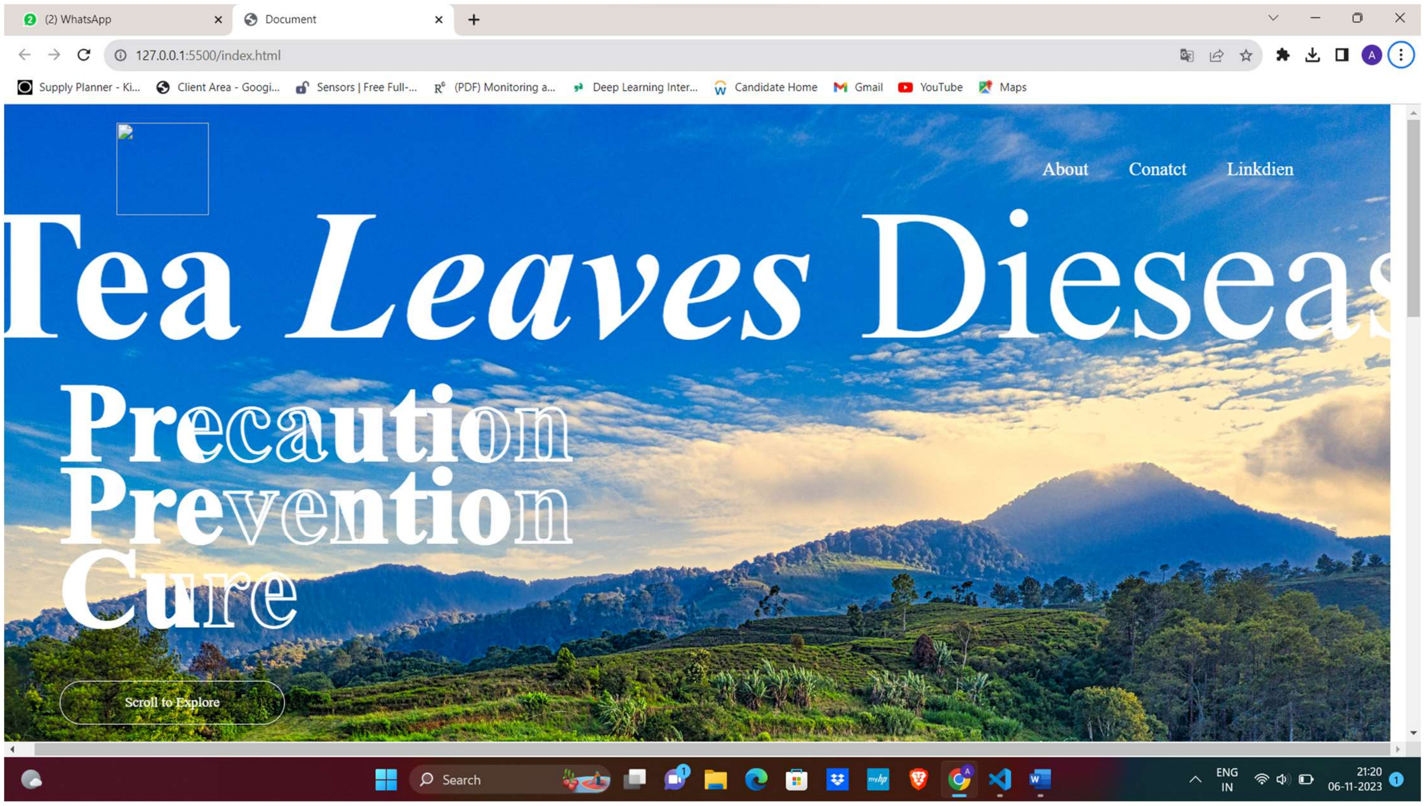


```
if __name__ == '__main__':
    app.run(debug=True)
```

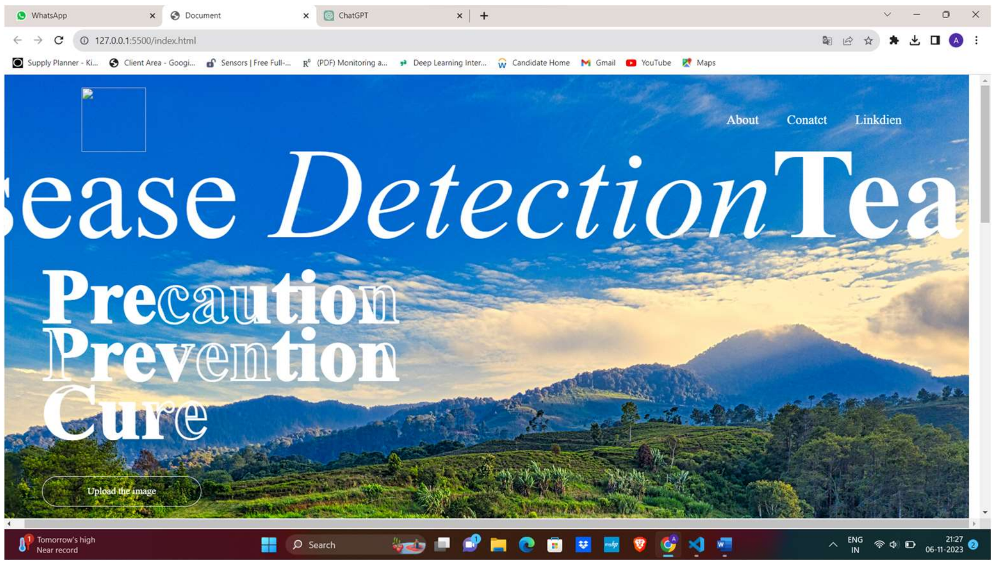
**Activity 3: Run the application**

Open Visual studio code and import all the project folders.

When you run the code.py file and click on the server url in terminal, you will be redirected to home page. The home page will look like:



If you click on About option, you will be redirected to about.html page and it looks like:



Then click on upload the image from bottom left corner, you will be redirected to desktop to upload the image and click on predict button, the type of disease will be given on output.

## Conclusion:

the deployment of the RCNN (Region-based Convolutional Neural Network) algorithm for tea leaf disease detection within a user-friendly website represents a substantial advancement in the realm of agriculture and disease management. The RCNN algorithm's capacity to accurately identify a range of tea leaf diseases is a testament to the power of deep learning and computer vision, offering swift and precise diagnoses. This not only aids in the early detection and management of diseases but also contributes to the overall health and productivity of tea crops. The web-based interface's intuitive design ensures accessibility to a broad spectrum of users, from farmers to researchers, while the potential for continuous model improvement based on user feedback promises ongoing enhancements in disease detection accuracy. Furthermore, this project serves as a model for similar systems in the realm of crop disease detection and underscores the potential for technology to play a pivotal role in advancing agriculture and global food security.