VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI

A Mini Project Report

On

“POTATO DISEASE CLASSIFICATION”

Submitted as part of curriculum 2021 scheme with course code 21CSMP67

BACHELOR OF ENGINEERING

In

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Submitted By

RAHUL D R	**4GM21CS075**
SRUJAN K S	**4GM21CS109**
VEDANTH K N	**4GM21CS118**
YATHISH RAO M R	**4GM21CS126**

Project Guide	Submitted To
Nanditha G	**Mr. Kotreshi S N**
Assistant Professor	**Assistant Professor**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)

(Accredited by NBA New Delhi, Valid up to 30.06.2025)

2023-2024

Srishyla Educational Trust (R), Bheemasamudra

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA New Delhi, Valid upto 30.06.2025)

CERTIFICATE

Certified that the Mini Project titled **"POTATO DISEASE CLASSIFICATION"** is a Bonafide work carried out by **RAHUL D R (4GM21CS075), SRUJAN K S (4GM21CS109), VEDANTH K N (4GM21CS118), YATHISH RAO M R (4GM21CS126)** as per curriculum scheme 2021 with course name *Mini Project* and course code *21CSMP67* of Bachelor of Engineering in the Department of Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi, during the year 2023-24. The Mini Project report has been approved as it satisfies the academic requirements with respect to the Mini Project work prescribed for Bachelor of Engineering Degree.

Guide	**Coordinator**	**Head of the Department**
Ms. Nanditha G	**Mr. Kotreshi S N**	**Dr. Veerappa BN**

ACKNOWLEDGEMENT

The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible.

We would like to express our gratitude to our **Principal, Dr. Sanjay Pande MB** for providing us a congenial environment for engineering studies and also for having showed us the way to carry out the Mini Project work.

We consider it a privilege and honour to express our sincere thanks to**, Dr. Veerappa BN** Professor and Head, Department of Computer Science and Engineering for his support and invaluable guidance throughout the tenure of this Mini Project work.

We would like to thank our Guide **Mr. Kotreshi S N**, Assistant Professor, Department of Computer Science and Engineering for his support, guidance, motivation, encouragement for the successful completion of this Mini Project work.

We intend to thank all the teaching and non-teaching staffs of our Department of Computer Science and Engineering for their immense help and co-operation.

Finally, we would like to express our gratitude to our parents and friends who always stood by us.

	Student Name
RAHUL D R	**4GM21CS075**
SRUJAN K S	**4GM21CS109**
VEDANTH K N	**4GM21CS118**
YATHISH RAO M R	**4GM21CS126**

Vision

To build excellent Technocrats in Computer Science and Engineering by continuously striving for excellence in IT industry to meet the challenges of society.

Mission

1. *To train students by adopting effective teaching-learning approach.*
2. *To establish collaborative learning approach with Industry and Professional bodies.*
3. *To develop engineers with Professional-Social ethics and creative Research Culture.*

Program Educational Objectives

1. Graduates able to apply the knowledge of Basics Science and Core Computer science to analyze and solve real world problems.
2. Graduates possess professional skills needed for IT employment and pursue higher education in Computer Science and Engineering.
3. Graduates engage in life-long learning and adapt to changing Environment.
4. Graduates who can succeed as an individual or team leader in multidisciplinary avenues.

Program Specific Outcomes

Graduates of Computer science & engineering are able to:

1. Understand basic principles of programming and core concepts of computer science.
2. Design and analyze software solutions for real world problems using computational models.
3. Develop Applications using C, Object Oriented Concepts, Computer Graphics, Database Management System, Web Programming, Machine Learning and Mobile Application Development to meet current industry and entrepreneurial requirements.

Srishyla Educational Trust ® Bheemasamudra

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

Department of Computer Science & Engineering

Programme outcomes of Computer Science & Engineering

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

In the agricultural sector, accurately identifying diseases in potato leaves is crucial for maintaining crop health and preventing significant losses. Our project, "Potato Disease Classification," aims to assist farmers by providing a reliable tool for identifying three common conditions: Early Blight, Late Blight, and Healthy leaves. Leveraging deep learning, specifically Convolutional Neural Networks (CNN), we utilized TensorFlow to build a robust model. The dataset was pre-processed and augmented to enhance the model's accuracy. The model was trained and validated over 20 epochs, and its performance was evaluated using accuracy and loss metrics. The trained model was then saved in the .keras format for further deployment.

To facilitate real-world application, we developed an API using Django, enabling users to submit images for prediction via a POST request. The server, tested using Postman, accurately returns the class label and prediction confidence. The frontend, created with ReactJS, provides an intuitive interface where users can upload images by dragging and dropping. This integration allows seamless interaction between the user and the model, making disease identification straightforward and accessible. Our project demonstrates significant potential in supporting farmers by providing timely and accurate disease diagnoses, thereby enhancing crop management and productivity.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

In the agricultural sector, potato cultivation plays a vital role in the economy and food supply. However, potato plants are susceptible to various diseases that can significantly impact yield and quality. Among the most common and devastating diseases are Early Blight and Late Blight, which, if not detected and managed promptly, can lead to substantial crop losses. Traditional methods of disease identification are often manual, time-consuming, and prone to human error, making it challenging for farmers to diagnose and address these issues effectively.

1.1 Problem Statement:

Farmers often struggle to accurately identify diseases affecting their potato crops, leading to delayed or inappropriate treatments. This inability to promptly and correctly diagnose leaf diseases results in decreased productivity and increased economic losses. There is a pressing need for an automated, reliable, and user-friendly solution that can assist farmers in identifying potato leaf diseases early and accurately.

1.2 Objectives:

1. **Develop a Deep Learning Model**: Utilize deep learning techniques, specifically Convolutional Neural Networks (CNN), to create a model capable of classifying potato leaves into three categories: Early Blight, Late Blight, and Healthy.

2. **Build an API for Model Deployment**: Implement a Django-based API to allow easy access to the model's prediction capabilities through a POST request.

3. **Create a User-Friendly Frontend**: Design a ReactJS-based frontend application that enables users to upload images of potato leaves and receive disease classification results in real-time.

4. **Ensure High Accuracy and Usability**: Focus on achieving high accuracy in disease classification and ensuring the system is user-friendly and accessible for farmers.

1.3 Proposed Solution:

Our solution involves developing an end-to-end system that integrates deep learning, web development, and user interface design to address the problem of potato leaf disease identification. The project utilizes a dataset of potato leaf images, which is pre-processed and augmented to improve the robustness of the CNN model built using TensorFlow. The trained model, capable of distinguishing between Early Blight, Late Blight, and Healthy leaves, is deployed as a web service via a Django API.

To provide a seamless user experience, we developed a frontend application using ReactJS. This application allows users to upload images of potato leaves through a drag-and-drop interface. The images are then sent to the server for prediction, and the results, including the class label and prediction accuracy, are displayed to the user. By combining advanced machine learning techniques with practical web development, our project aims to deliver a reliable and efficient tool to assist farmers in maintaining the health and productivity of their potato crops.

Chapter 2

LITERATURE SURVEY

The literature survey serves as a comprehensive review of existing research and developments related to potato disease classification using machine learning and deep learning techniques. This chapter explores various methodologies, algorithms, and datasets previously employed, highlighting their strengths and limitations.

2.1 Overview of Potato Disease Classification

Potato disease classification has garnered significant attention due to its impact on agriculture. Early and accurate detection of diseases like Early Blight and Late Blight is crucial for mitigating crop loss. Traditional methods involve manual inspection, which is time-consuming and prone to errors. Hence, the adoption of automated systems using image processing and machine learning techniques has increased.

2.2 Traditional Methods

Initial studies focused on traditional image processing techniques such as color analysis, texture analysis, and morphological features. These methods required manual feature extraction and were often limited by their inability to generalize across different environmental conditions.

2.3 Machine Learning Approaches

Machine learning (ML) algorithms like Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forests have been applied to classify potato diseases. These methods rely on handcrafted features extracted from images, such as color histograms, edge detection, and shape descriptors. Despite their effectiveness, these algorithms often struggle with the complexity and variability of real-world data.

2.4 Deep Learning Advances

The advent of deep learning (DL) revolutionized image classification tasks. Convolutional Neural Networks (CNNs), in particular, have shown remarkable success in automatically learning hierarchical features from raw images. Studies have demonstrated that CNN-based models significantly outperform traditional ML approaches in accuracy and robustness.

2.5 Relevant Studies

- [Ferentinos (2018)]: “Deep Learning Models for Plant Disease Detection and Classification” - focuses on the application of deep learning for plant disease classification.
- [Huang et al. (2020)]: “Transfer Learning for Plant Disease Classification” - discusses the benefits of transfer learning in deep learning models.

2.6 Datasets

Publicly available datasets, such as the PlantVillage dataset, have been extensively used for training and evaluating models. These datasets contain thousands of annotated images of healthy and diseased potato leaves, providing a valuable resource for research.

2.7 Challenges and Future Directions

Despite the progress, challenges remain in developing robust models for potato disease classification. These include handling diverse environmental conditions, varying disease symptoms, and limited annotated data. Future research should focus on improving data augmentation techniques, developing more sophisticated models, and exploring transfer learning to leverage pre-trained networks.

Chapter 3

METHODOLOGY

3.1 Data Collection:

The dataset for the Potato Disease Classification project was sourced from Kaggle. It comprises three folders containing images of potato leaves categorized into Early Blight, Late Blight, and Healthy. Each of the Early Blight and Late Blight folders contains 1000 images, while the Healthy folder contains 152 images. This dataset serves as the foundation for training our Convolutional Neural Network (CNN) model.

3.2 Data Preprocessing:

Preprocessing is a critical step to prepare the dataset for model training. The dataset was partitioned into training, validation, and test sets using a custom function that ensures randomness and reproducibility. This partitioning helps to create balanced datasets for training, validating, and testing the model.

Data augmentation techniques were applied to the training, validation, and test datasets to enhance the model's generalization ability and prevent overfitting. These techniques include resizing, rescaling, random flipping, and random rotation of images, which help the model learn from a variety of image transformations and improve its robustness.

3.3 Model Building:

The core of the project is the Convolutional Neural Network (CNN) model, designed to classify the potato leaf images into the three specified categories. The model architecture includes several convolutional and pooling layers to extract features from the images, followed by dense layers for classification. TensorFlow was used for building this model, utilizing its powerful and flexible tools for defining, training, and evaluating deep learning models.

The model consists of the following layers:

1. Resizing and rescaling layer to standardize image dimensions and pixel values.
2. Data augmentation layer for random flipping and rotation.
3. Multiple convolutional layers with ReLU activation and max-pooling layers for feature extraction.
4. Flattening layer to convert the 2D feature maps into a 1D feature vector.
5. Dense layers for classification with softmax activation in the final layer to output probabilities for each class.

3.4 Training and Validation:

The model was compiled using the Adam optimizer and Sparse Categorical Crossentropy loss function. The training process involved fitting the model to the training dataset for 20 epochs, with the validation dataset used to monitor performance and prevent overfitting.

The training process included:

- Batch processing with a specified batch size.
- Monitoring training accuracy and loss, as well as validation accuracy and loss, throughout the epochs.
- Using early stopping criteria to prevent overfitting and ensure optimal model performance.

3.5 Model Evaluation:

The model's performance was evaluated using several metrics, primarily accuracy and loss, to assess its effectiveness in classifying the potato leaf images. The results were visualized through accuracy and loss graphs, showing the training and validation performance over the epochs.

The evaluation process included:

- Plotting training and validation accuracy to observe how well the model learns and generalizes.
- Plotting training and validation loss to identify any overfitting or underfitting.
- Using a custom prediction function to test the model on the test dataset and visualize the predictions along with their confidence levels.

Chapter 4

DESIGN AND IMPLEMENTATION

4.1 System Workflow Diagram

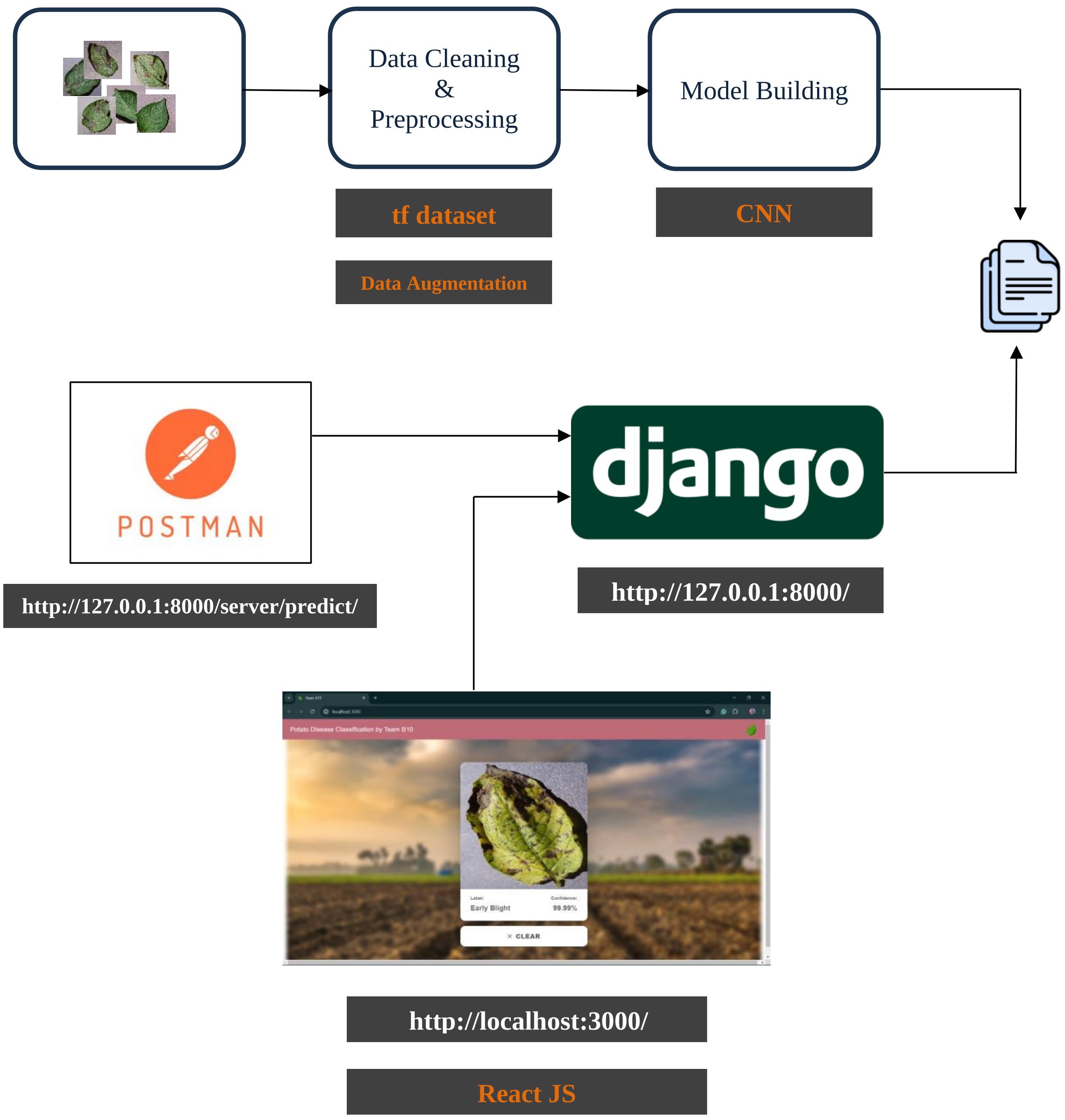


Figure 4.1: System Architecture Desing

4.2 Model Building Using TensorFlow and CNN:

TensorFlow:

- **Open-Source Machine Learning Framework:** TensorFlow is an open-source library developed by Google for numerical computation and machine learning. Supports various platforms and devices, including CPUs, GPUs, and TPUs, making it versatile for different types of machine learning tasks.
- **Comprehensive Ecosystem:** Provides a repository of reusable machine learning models. end-to-end platform for deploying production machine learning pipelines.
- **Ease of Use and Performance:** Simplifies the process of building and training neural networks with a high-level API. Utilizes data flow graphs for efficient computation, ensuring high performance for large-scale machine learning tasks.
- **Community and Support:** Offers comprehensive documentation and tutorials for developers of all skill levels. A large community contributes to the development of TensorFlow, providing a wealth of resources and support.

Convolutional Neural Networks (CNNs):

- **Specialized Neural Network Architecture:** CNNs are designed for processing structured grid data like images. Utilizes local connections (convolutions) to capture spatial hierarchies in data.
- **Key Components:** Apply convolution operations to extract features from the input data. Reduce the spatial dimensions of the data, helping to manage computational complexity and prevent overfitting. Perform high-level reasoning by connecting all neurons from the previous layer to each neuron in the current layer.
- **Image Recognition and Beyond:** Widely used for tasks like image and video recognition. Effective in other domains, such as natural language processing and speech recognition, by adapting the architecture to the specific data type.

The model was built using TensorFlow and Keras, featuring a CNN architecture with convolutional and pooling layers to classify potato leaf diseases. It was trained on a dataset with 20 epochs, achieving classification into Early Blight, Late Blight, and Healthy categories.

```
import tensorflow as tf

from tensorflow.keras import models, layers

import matplotlib.pyplot as plt

import numpy as np
```

```
from keras.models import save_model

# Model building
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(3, activation='softmax')
])


# Compiling the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])

# Training and validation
history = model.fit(train_generator, epochs=20, validation_data=validation_generator)

#Saving model in .keras format
save_model(model, "potatoes.keras", save_format="keras")
```

4.3 Backend Development using Django:

Django Framework:

- **High-Level Web Framework:** Django is designed to help developers build robust web applications quickly and efficiently. It includes many built-in features such as authentication, URL routing, and a templating system to streamline development.
- **DRY Principle (Don't Repeat Yourself):** Django encourages reusability and follows the DRY principle to minimize redundancy in code. This approach makes the codebase easier to maintain and extend over time.

- **Built-In Features:** Django provides an ORM for database interactions, allowing developers to work with databases using Python code instead of SQL. Automatically generates an admin interface for managing application data, reducing the need for custom administrative tools. Simplifies form creation and validation, making it easier to handle user input. Includes protections against common security threats like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
- **Modular Structure:** Django's modular architecture allows developers to build applications using reusable components (apps) that can be plugged into different projects. Developers can easily extend functionality by adding or modifying apps without affecting the overall structure of the application.
- **Scalability and Security:** Django is designed to handle high traffic and large-scale applications, making it suitable for growing businesses and projects. Emphasizes security by providing built-in features and best practices for secure web development.

The backend, developed with Django, includes an API endpoint to handle image uploads, process them, and return predictions using the trained model. It efficiently manages POST requests and serves JSON responses with the predicted class and confidence level.

```
# views.py
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from myapp.ml_model import model, CLASS_NAMES
import numpy as np
from PIL import Image
from io import BytesIO
def read_file_as_image(data) -> np.ndarray:
    image = np.array(Image.open(BytesIO(data)))
    return image
@csrf_exempt
def predict(request):
    if request.method == 'POST':
        if 'file' not in request.FILES:
            return JsonResponse({'error': 'No file part'}, status=400)
        file = request.FILES['file']
        image = read_file_as_image(file.read())
        img_batch = np.expand_dims(image, 0)
        predictions = model.predict(img_batch)
        predicted_class = CLASS_NAMES[np.argmax(predictions[0])]
        confidence = np.max(predictions[0])
        return JsonResponse({
            'class': predicted_class,
            'confidence': float(confidence)
        })
    return JsonResponse({'error': 'Invalid request method'}, status=400)
```

# ml_model.py

```
import tensorflow as tf
model_path = "C:/Users/user/OneDrive/Desktop/potato_disease_classification/Training/potatoes.keras"
model = tf.keras.models.load_model(model_path)
CLASS_NAMES = ['Early Blight', 'Late Blight', 'Healthy']
```

# urls.py

```
from django.urls import path
from myapp import views
urlpatterns = [
    path('predict/', views.predict, name='predict'),
    path('test/', views.test_view, name='test_view'),
]
```

4.4 Frontend Development Using React JS:

ReactJS:

- **JavaScript Library for Building User Interfaces:** ReactJS, developed by Facebook, is used for building fast and interactive user interfaces for web applications. Allows developers to build encapsulated components that manage their own state and can be composed to create complex UIs.

- **Virtual DOM:** Utilizes a virtual DOM to optimize updates and rendering, improving performance by reducing direct manipulation of the actual DOM. Ensures efficient updates and rendering of components when the underlying data changes.
- **Declarative Syntax:** Enables developers to describe what the UI should look like based on the current state, simplifying the development process. Makes it easier to predict how the UI will change in response to state updates, improving the maintainability of the codebase.
- **Extensive Ecosystem:** A large ecosystem of tools, libraries, and a strong community supports ReactJS, providing numerous resources for developers. Can be integrated with other libraries and frameworks, such as Redux for state management and Next.js for server-side rendering.

The frontend, built with ReactJS, allows users to upload potato leaf images and receive real-time predictions. It features a drag-and-drop interface, image preview, and displays prediction results by interacting with the Django API.

// home.js

```
import React, { useState, useEffect } from "react";

import { Button, AppBar, Toolbar, Typography, Card, CardMedia, CardContent, CircularProgress } from
"@material-ui/core";

import { DropzoneArea } from "material-ui-dropzone";

import axios from "axios";

const ImageUpload = () => {

  const [selectedFile, setSelectedFile] = useState();

  const [preview, setPreview] = useState();

  const [data, setData] = useState();

  const [isLoading, setIsLoading] = useState(false);

  const sendFile = async () => {

    let formData = new FormData();

    formData.append("file", selectedFile);

    let res = await axios.post(process.env.REACT_APP_API_URL, formData);

    if (res.status === 200) {

      setData(res.data);

    }

    setIsLoading(false);
```

```
};
useEffect(() => {
  if (!selectedFile) return;
  const objectUrl = URL.createObjectURL(selectedFile);
  setPreview(objectUrl);
  setIsLoading(true);
  sendFile();
}, [selectedFile]);
const onSelectFile = (files) => {
  setSelectedFile(files[0]);
  setData(undefined);
};
return (
  <div>
    <AppBar position="static">
      <Toolbar>
        <Typography variant="h6">Potato Disease Classification</Typography>
      </Toolbar>
    </AppBar>
    <DropzoneArea onChange={onSelectFile} />
    {preview && (
      <Card>
        <CardMedia image={preview} component="img" />
        {isLoading ? (
          <CircularProgress />
        ) : data ? (
          <CardContent>
            <Typography>Label: {data.class}</Typography>
            <Typography>Confidence: {(parseFloat(data.confidence) * 100).toFixed(2)}%</Typography>
          </CardContent>
        ) : null}
        </Card>
      )}
    </div>
  );
};
export default ImageUpload;
```

Chapter 5

RESULTS

5.1 Model Performance:

- **Accuracy and Loss Graphs:** Include snapshots of the training and validation accuracy and loss graphs. These graphs illustrate how the model's accuracy improved and how the loss decreased over the 20 epochs, indicating the model's learning process.

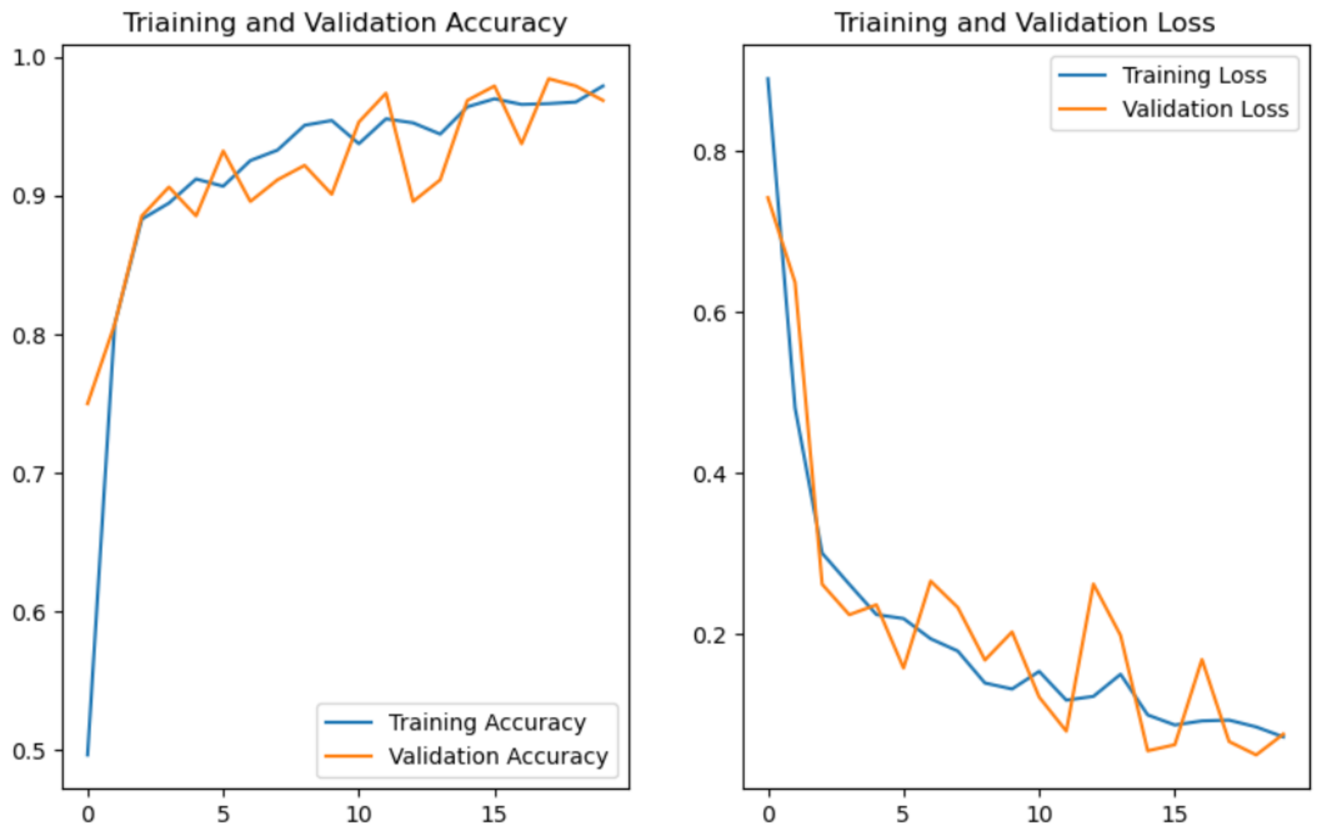


Figure 5.1: Accuracy and Loss Graphs

- **Prediction Examples:** Add the prediction photos, showcasing the actual labels and the predicted labels along with confidence scores. This will visually demonstrate the model's accuracy and reliability in classifying potato leaf diseases.

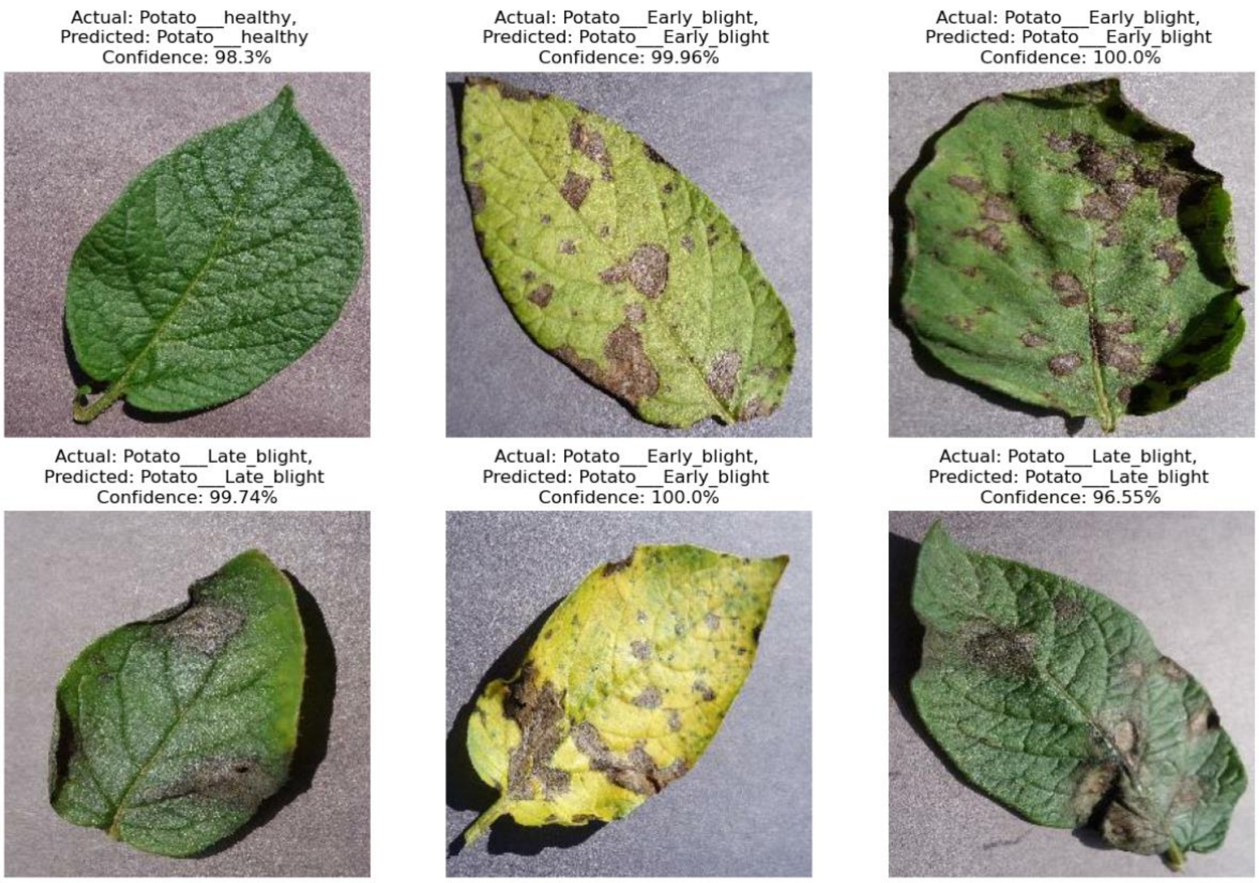
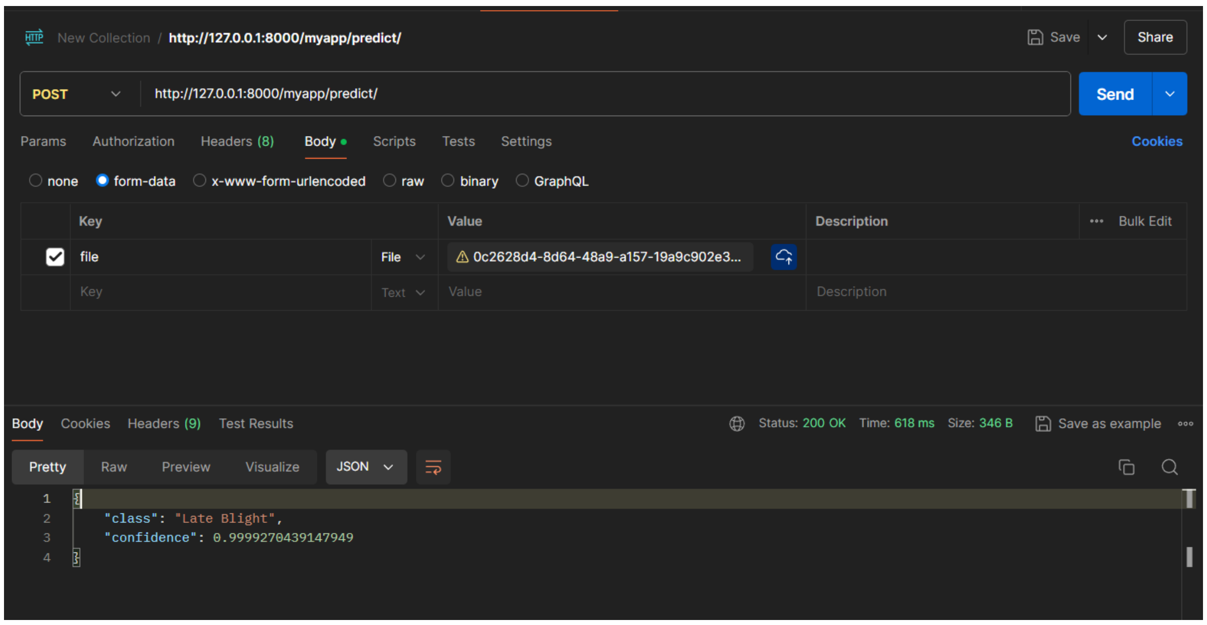

Actual: Potato__healthy,
Predicted: Potato__healthy
Confidence: 98.3%
Actual: Potato__Early_blight,
Predicted: Potato__Early_blight
Confidence: 99.96%
Actual: Potato__Early_blight,
Predicted: Potato__Early_blight
Confidence: 100.0%
Actual: Potato__Late_blight,
Predicted: Potato__Late_blight
Confidence: 99.74%
Actual: Potato__Early_blight,
Predicted: Potato__Early_blight
Confidence: 100.0%
Actual: Potato__Late_blight,
Predicted: Potato__Late_blight
Confidence: 96.55%

Figure 5.2: Prediction Examples

5.2 API Functionality:

Postman Test: Include a snapshot of the Postman application showing a successful POST request to the Django API. This demonstrates that the API can correctly predict the class and confidence of the uploaded image.


New Collection / http://127.0.0.1:8000/myapp/predict/

POST http://127.0.0.1:8000/myapp/predict/

Body: form-data

Key		Value	Description
file	File	0c2628d4-8d64-48a9-a157-19a9c902e3...	

Status: 200 OK Time: 618 ms Size: 346 B

```
{
    "class": "Late Blight",
    "confidence": 0.9999270439147949
}
```

Figure 4.3: Postman Test

5.3 Frontend Functionality:

- **Before Prediction**: Include a snapshot of the frontend interface before the user uploads an image. This shows the user-friendly design and the drag-and-drop functionality for uploading images.

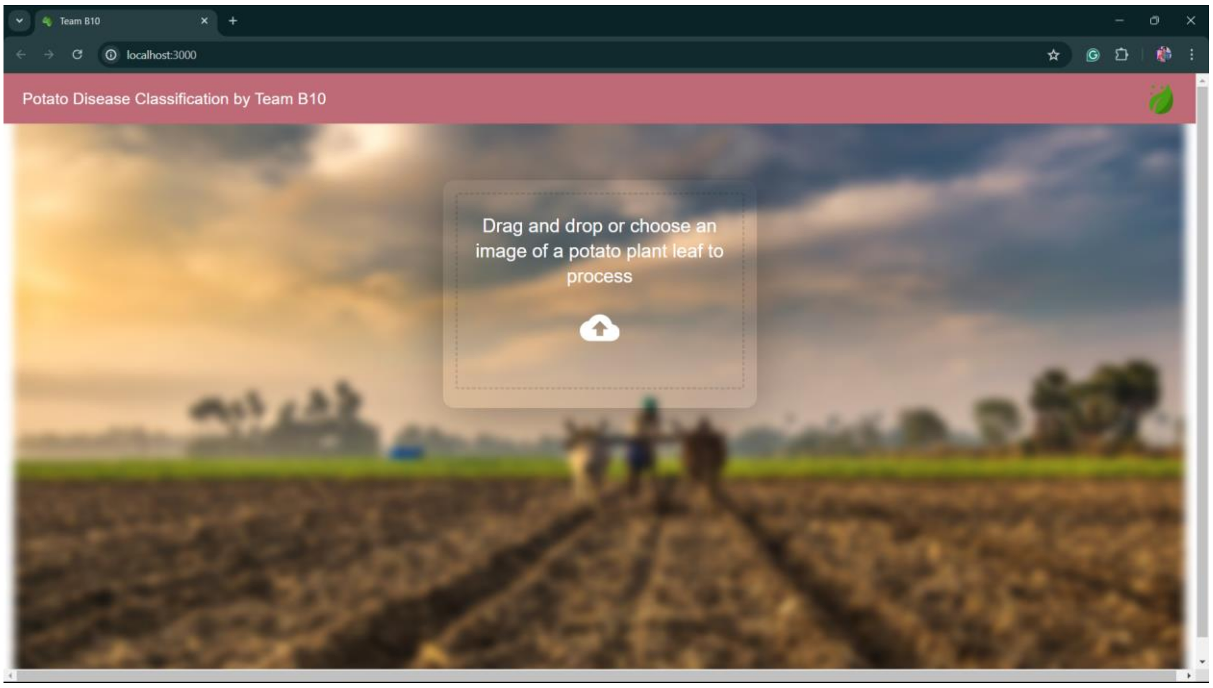

Team B10
localhost:3000
Potato Disease Classification by Team B10
Drag and drop or choose an image of a potato plant leaf to process

Figure 5.4: Before Prediction

- **After Prediction:** Include a snapshot of the frontend interface after an image has been uploaded and processed. This shows the predicted class and confidence, demonstrating the end-to-end functionality of the application.

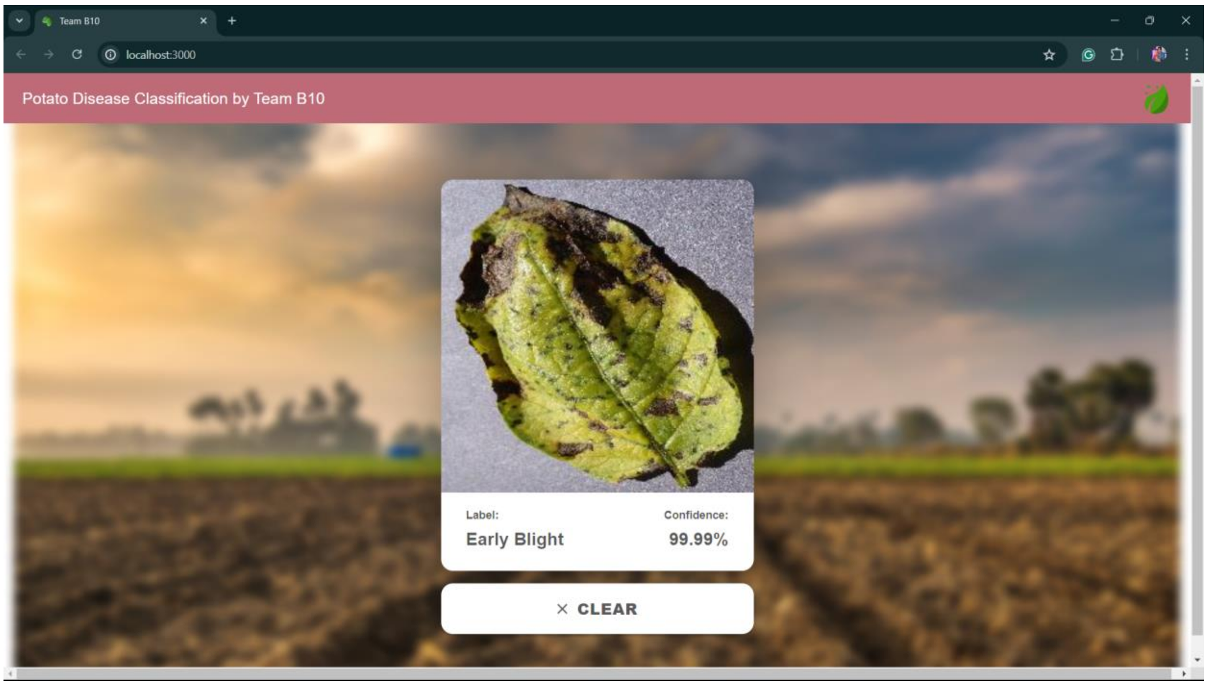


Figure 5.5: After Prediction

Chapter 6

CONCLUSION

In this project, we successfully developed a deep learning model to classify potato leaf diseases using convolutional neural networks (CNNs). The model was trained on a dataset of labelled images and achieved high accuracy in predicting three classes: Early Blight, Late Blight, and Healthy. The implementation of the Django backend and ReactJS frontend created a seamless user experience, allowing users to upload images and receive real-time predictions with confidence scores.

The results, including accuracy and loss graphs, prediction examples, and API tests, demonstrated the model's effectiveness and practical applicability in real-world scenarios. This system can assist farmers and agricultural professionals in early disease detection, potentially reducing crop loss and improving overall yield. Future work can focus on expanding the dataset, improving model accuracy, and enhancing the user interface to provide a more robust and user-friendly tool for the agricultural community.

REFERENCES

1 **TensorFlow** - TensorFlow is an open-source platform for machine learning. (https://www.tensorflow.org/)

2 **Django** - Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. (https://www.djangoproject.com/)

3 **ReactJS** - A JavaScript library for building user interfaces. (https://reactjs.org/)

4 **Material-UI** - React components for faster and easier web development. (https://material-ui.com/)

5 **Potato Leaf Disease Dataset** - The dataset used for training and testing the model, obtained from Kaggle.

6 **ImageDataGenerator** - Keras API for generating batches of tensor image data with real-time data augmentation. (https://keras.io/api/preprocessing/image/)

7 **Postman** - A platform for API development. (https://www.postman.com/)

8 **Numpy** - A fundamental package for scientific computing with Python. (https://numpy.org/)

9 **Pillow** - The Python Imaging Library adds image processing capabilities to your Python interpreter. (https://python-pillow.org/)

10 **Python** - An interpreted, high-level and general-purpose programming language. (https://www.python.org/)