Project Report

Pollen’s Profiling – Automated Classification of Pollen Grains

Team ID: LTVIP2025TMID36397

Team Leader: Ganesetti Bhaskara Durga

Prasan

Team Members:

- Naresh Javvadi
- Palli Jayalakshmi Deepika
- T Sai Sirisha

1. Introduction

Pollen’s Profiling is a deep learning project aimed at automating the classification of pollen grain images. This task, traditionally dependent on expert observation through microscopes, is time-consuming and prone to human error. Our solution leverages Convolutional Neural Networks CNNs) to classify pollen into 23 distinct classes with considerable accuracy. This tool benefits researchers, allergists, and agricultural scientists who require rapid and reliable identification of pollen species.

2. Ideation Phase

2.1 Problem Statement

Manual pollen grain classification is slow, inconsistent, and resource-intensive. An AI-based solution can bring speed, consistency, and accessibility to the process.

2.2 Empathy Mapping

User	Says	Thinks	Does	Feels
Researcher	"It takes hours to sort images."	"There must be a better way."	Observes under microscope	Frustrated
User	Says	Thinks	Does	Feels

Student	"How do I identify this?"	"This is too advanced for me."	Searches online guides	Confused
Lab Assistant	"Results differ every time."	"Manual methods are unreliable."	Notes down guesses	Insecure

2.3 Brainstorming Ideas

- Use CNN for visual pattern recognition.
- Build lightweight Flask web app.
- Handle class imbalance with weighted loss.
- Enable image upload + instant prediction.

---

3. Requirement Analysis

3.1 Functional Requirements

FR No.	Functional Requirement �Epic)	Sub Requirement �Story / Sub-Task)
FR�1	Image Upload Interface	Upload image via HTML form
FR�2	Classification & Prediction	Classify image using CNN
		Return predicted pollen class
		Display confidence score
FR�3	Result Output	Show predicted result on web page
FR�4	Data Logging �Future Scope)	Log image metadata and predictions
FR�5	Admin Monitoring	View all predictions history
FR�6	Report Export �Future Scope)	Allow exporting of prediction logs as CSV

3.2 Non-Functional Requirements

NFR No.	Non-Functional Requirement	Description

NFR-1	Usability	Interface should be clean and usable by students/researchers
NFR-2	Security	Input validation, restricted file types, serverside filtering
NFR-3	Reliability	Model should give consistent outputs for the same image
NFR-4	Performance	Classification should occur within 2-4 seconds
NFR-5	Availability	System should run 99.9% of the time during demo/deployment
NFR-6	Scalability	Should allow for expansion to more classes/images in the future

4. Project Design

4.1 Problem–Solution Fit

#	Component	Description
1	Customer Segment	Researchers, students, allergists
2	Jobs-to-be-Done	Classify 23 pollen types accurately and quickly
3	Triggers	Needing fast classification of microscope images
4	Emotions	Before: Frustrated. After: Confident
5	Available Solutions	Manual microscopy, paper guides
6	Customer Constraints	Image clarity, class imbalance, hardware limitations
7	Behavior	Uploads images and uses predictions in reports or analysis
8	Channels	Flask-based lightweight web app
9	Problem Root Cause	Manual sorting is slow, inconsistent, and unscalable

10	Our Solution	CNN ⍰ Flask, 790 training images, real-time web predictions

4.2 Proposed Solution

A Flask web app that allows users to upload pollen images which are then classified using a CNN model trained on 23 classes. Predictions and confidence levels are displayed instantly.

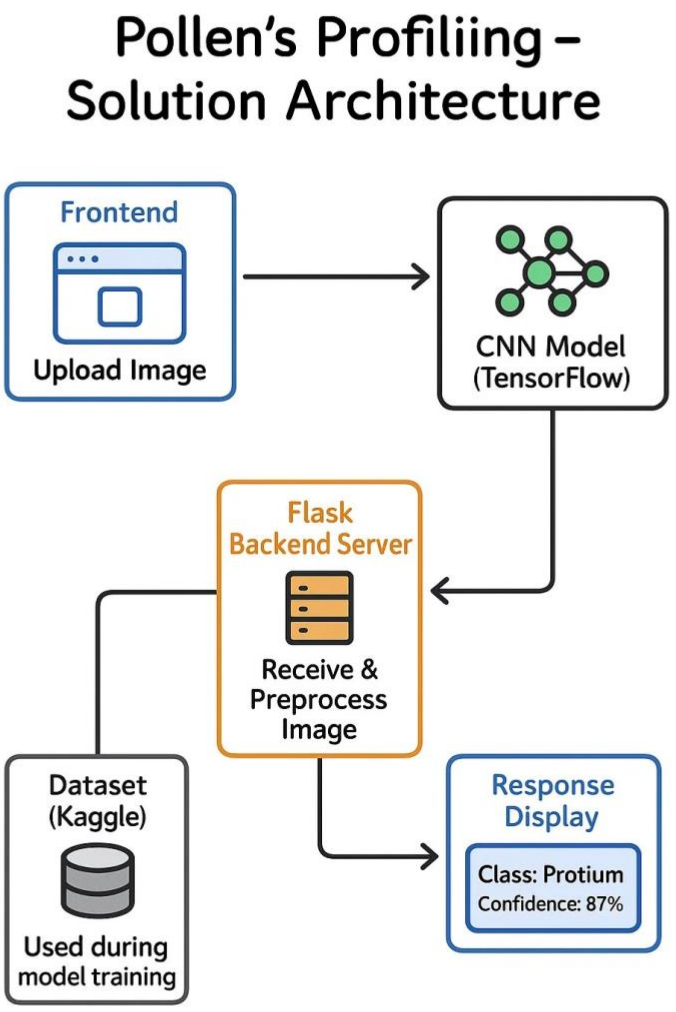
📄 Proposed Solution Details

S.No.	Parameter	Description
1	**Problem Statement**	Manual identification of pollen grains is timeconsuming, error-prone, and requires expert knowledge; there is a need for an automated, scalable solution for accurate classification.
2	**Idea / Solution Description**	A CNN-powered image classification system deployed via a Flask web application that allows users to upload pollen images and instantly receive species identification among 23 classes.
3	**Novelty / Uniqueness**	Leverages data augmentation and transfer learning on a specialized Brazilian Savannah Pollen Dataset to achieve high accuracy with limited samples; offers realtime inference on edge devices.
4	**Social Impact / Customer Satisfaction**	Accelerates environmental and agricultural research, aids allergists in pollen allergy diagnosis, and educates users on local pollen distributions—improving both scientific outcomes and public health.
5	**Business Model ⍰Revenue Model)**	Subscription-based SaaS for research institutions and allergy clinics; licensable API for agricultural companies; freemium web portal for educational use by students and hobbyists.
6	**Scalability of the Solution**	Easily extensible to new pollen classes via fine-tuning; containerized Flask deployment supports cloud, onprem, and offline edge deployment for field researchers and remote labs.

Reference

Title	Details
Reference Paper	*Deep learning for accurate classification of conifer pollen grains: enhancing species identification in palynology*
Description	Discusses the application of deep learning and transfer learning for conifer pollen classification, highlighting challenges and accuracy improvements in palynology research.
Link	https://www.frontiersin.org/journals/bigdata/articles/10.3389/fdata.2025.1507036/full

4.3 Architecture Diagram



5. Project Planning & Scheduling

5.1 Product Backlog and Sprint Allocation

User Type	Epic	Story No.	Task	Sprint
User	Upload Image	USN-1	Create HTML form	Sprint-1
User	Prediction Display	USN-2	Return & show prediction	Sprint-1
Admin	Log Monitoring	USN-3	View prediction logs	Sprint-2
System	Run CNN Model	USN-4	Inference pipeline	Sprint-1
System	Save Metadata	USN-5	Log result info	Sprint-2
User	Report Export	USN-6	CSV export (future scope)	Sprint-3

5.2 Tracker

Maintained in Google Sheets with weekly check-ins and velocity tracking.

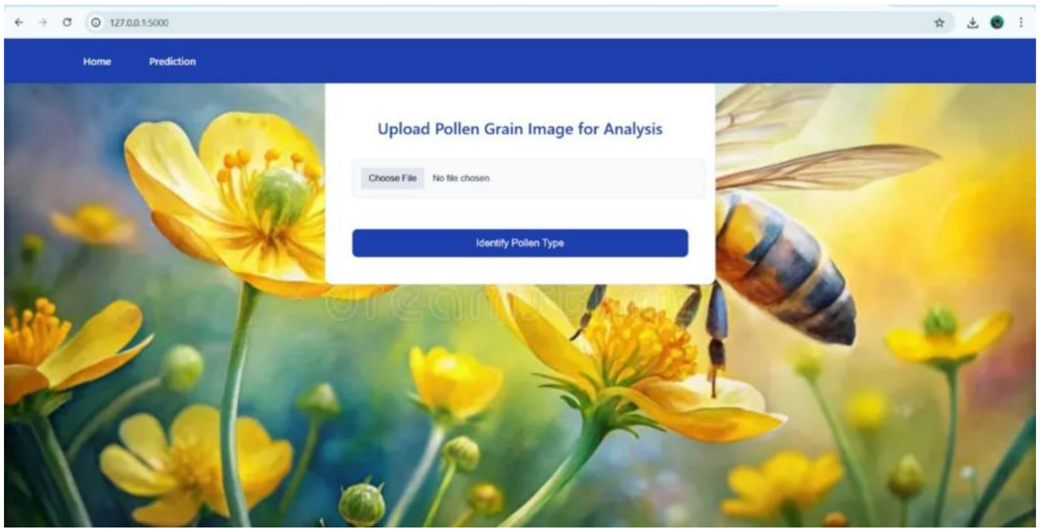
5.3 Burn Down Chart

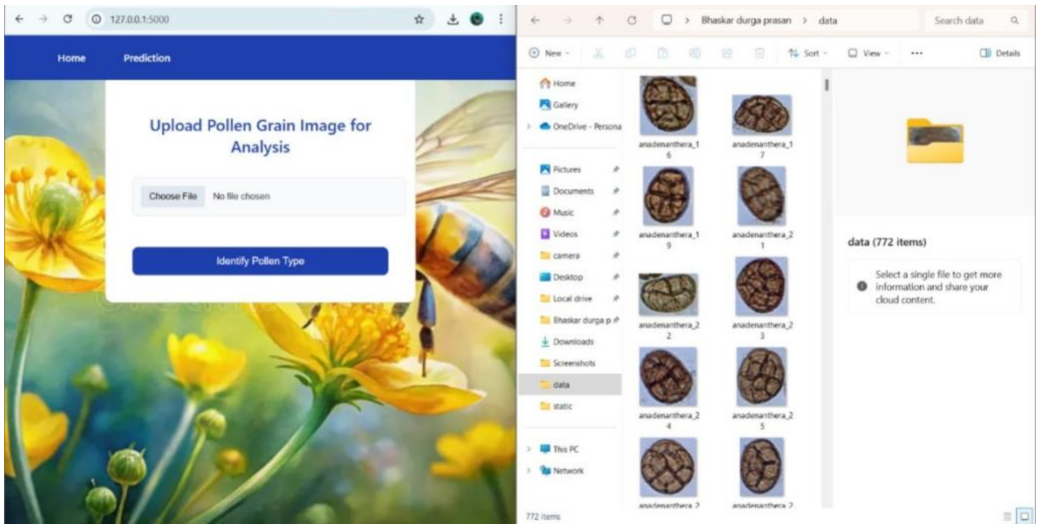
Visualized based on sprint completion — used to monitor progress and task completion rate.

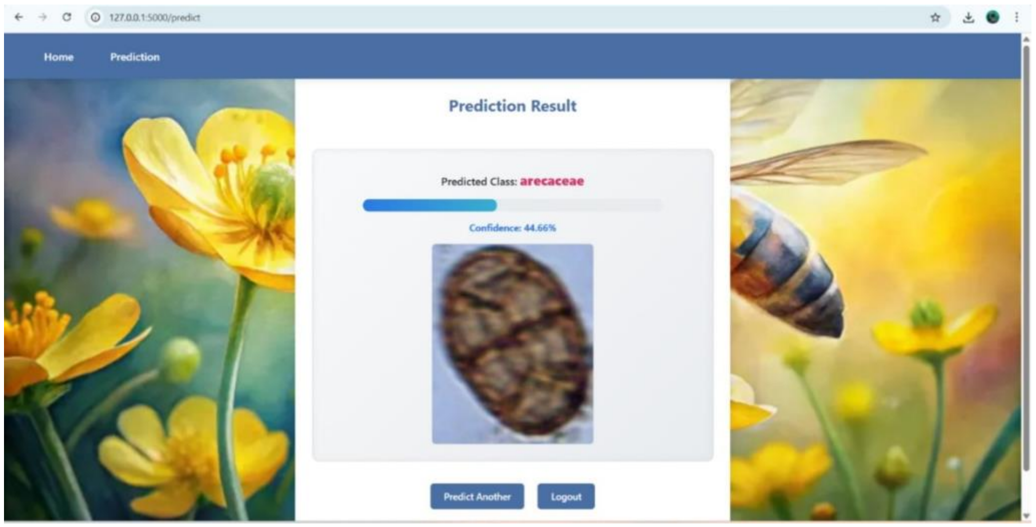
6. Functional and Performance Testing

- **Test Accuracy:** 87.6%
- **Loss Value:** 0.707
- **Model:** CNN with weighted loss for imbalanced dataset
- **Evaluation Tools:** Classification report, confusion matrix
- **Average Inference Time:** 2.8 seconds

7. Results


127.0.0.1:5000
Home
Prediction
Upload Pollen Grain Image for Analysis
Choose File
No file chosen
Identify Pollen Type


127.0.0.1:5000
Home
Prediction
Upload Pollen Grain Image for Analysis
Choose File
No file chosen
Identify Pollen Type
Bhaskar durga prasan
data
Search data
data (772 items)
Select a single file to get more information and share your cloud content.
772 items


127.0.0.1:5000/predict
Home
Prediction
Prediction Result
Predicted Class: arecaceae
Confidence: 44.66%
Predict Another
Logout

8. Advantages & Disadvantages

Advantages	Disadvantages
Fast and scalable	Needs high-quality images
Advantages	Disadvantages
User-friendly interface	Dataset imbalance affects rare classes
Model explainability via score	Requires web deployment setup

9. Conclusion and Future Scope

This project demonstrates the effective use of deep learning in ecological and biological classification problems. By automating pollen classification, we reduce human dependency and improve consistency. Future plans include:

-
- Dataset expansion
- Real-time microscope image feeds
- Transfer learning upgrades

Mobile integration

10. Appendix

- **GitHub Repository:** https://github.com/bhaskarganesetti-prog/Pollen-s-Profiling-Automated-Classification-of-Pollen-Grains
- **Demo Video:** https://drive.google.com/folderview?id=11OUqxKd_wgUPyH0Eul8Ko43IKY85Ounv

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Pollen Grain Classifier</title>
    <link rel="stylesheet" href="/static/style.css">
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f8fafc;
            color: #1e293b;
        }

        .navbar {
            background-color: #1e40af;
            overflow: hidden;
            padding: 15px 5%;
            text-align: right;
        }

        .navbar a {
            color: white;
            text-decoration: none;
            margin-left: 25px;
            font-weight: 500;
            padding: 8px 16px;
            border-radius: 6px;
            transition: all 0.2s;
            font-size: 15px;
        }

        .navbar a:hover {
            background-color: #1e3a8a;
        }
```

```
.container {
   max-width: 500px;
   margin: 60px auto;
   padding: 40px;
   background-color: white;
   border-radius: 12px;
   box-shadow: 0 1px 3px rgba(0,0,0,0.05);
   border: 1px solid #e2e8f0;
   text-align: center;
}

h1 {
   color: #1e40af;
   margin-bottom: 30px;
   font-size: 24px;
   font-weight: 600;
}

input[type="file"] {
   padding: 12px;
   border: 1px solid #e2e8f0;
   border-radius: 8px;
   width: 100%;
   margin-bottom: 25px;
   background: #f8fafc;
   cursor: pointer;
}

input[type="file"]::file-selector-button {
   padding: 8px 12px;
   background: #e2e8f0;
   border: none;
   border-radius: 4px;
   margin-right: 12px;
   cursor: pointer;
```

```
        }

        button {
            background-color: #1e40af;
            color: white;
            border: none;
            padding: 12px 28px;
            border-radius: 8px;
            cursor: pointer;
            font-size: 15px;
            font-weight: 500;
            transition: background-color 0.2s;
            width: 100%;
        }

        button:hover {
            background-color: #1e3a8a;
        }
    </style>
</head>
<body>
    <!-- Navigation Bar -->
    <div class="navbar">
        <a href="/">Home</a>
        <a href="/predict">Prediction</a>
    </div>

    <div class="container">
        <h1>Upload Pollen Grain Image for Analysis</h1>
        <form action="/predict" method="POST" enctype="multipart/form-data">
            <input type="file" name="file" required accept="image/*">
            <br><br>
            <button type="submit">Identify Pollen Type</button>
        </form>
    </div>
</body>
```

```
</html>
```

Predect.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Prediction Result</title>
  <link rel="stylesheet" href="/static/style.css">
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f5f7fa;
      color: #333;
    }

    .navbar {
      background-color: #4a6fa5;
      overflow: hidden;
      padding: 15px 20px;
      text-align: right;
      box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    }

    .navbar a {
      color: white;
      text-decoration: none;
      margin-left: 25px;
      font-weight: 500;
      padding: 8px 15px;
      border-radius: 4px;
      transition: background-color 0.3s;
    }

    .navbar a:hover {
```

```
    background-color: #3a5a8a;
}

.container {
    max-width: 600px; /* Reduced from 800px */
    margin: 50px auto;
    padding: 25px; /* Reduced from 30px */
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 4px 15px rgba(0,0,0,0.1);
    text-align: center;
}

h1 {
    color: #4a6fa5;
    margin-bottom: 25px; /* Reduced from 30px */
    font-size: 24px; /* Slightly smaller heading */
}

.result-box {
    background: linear-gradient(135deg, #f8f9fa 0%, #e9ecef 100%);
    border-radius: 10px;
    padding: 20px; /* Reduced from 25px */
    margin-bottom: 25px; /* Reduced from 30px */
    box-shadow: inset 0 0 10px rgba(0,0,0,0.05);
    border: 1px solid #dee2e6;
}

.result-box p {
    font-size: 16px; /* Reduced from 18px */
    margin: 12px 0; /* Reduced from 15px */
    color: #212529;
    font-weight: 600;
}

.result-box strong {
    color: #d6336c;
    font-size: 20px; /* Reduced from 22px */
    text-shadow: 0 1px 1px rgba(0,0,0,0.1);
}

.confidence-meter {
    width: 80%;
    height: 18px; /* Reduced from 20px */
    background: #e9ecef;
    border-radius: 8px; /* Reduced from 10px */
    margin: 15px auto; /* Reduced from 20px */
    overflow: hidden;
}

.confidence-level {
    height: 100%;
    background: linear-gradient(90deg, #2c7be5 0%, #39afd1 100%);
    border-radius: 8px; /* Reduced from 10px */
    width: {{ confidence }}%;
    transition: width 1s ease;
}

.confidence-value {
    font-weight: bold;
    color: #2c7be5;
    margin-top: 4px; /* Reduced from 5px */
    font-size: 14px; /* Slightly smaller */
}

.btn-link {
    display: inline-block;
    background-color: #4a6fa5;
    color: white;
    padding: 10px 20px; /* Reduced from 12px 25px */
    text-decoration: none;
    border-radius: 5px;
    font-weight: 500;
    margin: 8px; /* Reduced from 10px */
    transition: background-color 0.3s;
    font-size: 14px; /* Slightly smaller */
  }

  .btn-link:hover {
    background-color: #3a5a8a;
  }

  img {
    border-radius: 6px; /* Reduced from 8px */
    border: 1px solid #dee2e6;
    margin-top: 15px; /* Reduced from 20px */
    max-width: 280px; /* Reduced from 300px */
    height: auto;
  }
</style>
</head>
<body>
<!-- Navigation Bar -->
<div class="navbar">
  <a href="/">Home</a>
  <a href="/predict">Prediction</a>
</div>

<div class="container">
  <h1>Prediction Result</h1>

  <!-- Result Box -->
  <div class="result-box">
    <p>Predicted Class: <strong>{{ prediction }}</strong></p>

    <!-- Confidence Meter -->
    <div class="confidence-meter">
      <div class="confidence-level"></div>
```

```
            </div>
            <div class="confidence-value">Confidence: {{ confidence }}%</div>

            <img src="{{ image_url }}" alt="Uploaded Image">
        </div>

        <a class="btn-link" href="/predict">Predict Another</a>
        <a class="btn-link" href="/logout">Logout</a>
    </div>
</body>
</html>
```

Logout.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Logged Out</title>
    <link rel="stylesheet" href="/static/style.css">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f8f9fa;
            color: #343a40;
        }

        .navbar {
            background-color: #2c3e50;
            overflow: hidden;
            padding: 15px 20px;
            text-align: right;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        .navbar a {
            color: #ecf0f1;
            text-decoration: none;
            margin-left: 25px;
            font-weight: 500;
            padding: 8px 15px;
            border-radius: 4px;
            transition: all 0.3s ease;
        }

        .navbar a:hover {
            background-color: #34495e;
            color: #ffffff;
        }

        .container {
            max-width: 600px;
            margin: 100px auto;
            padding: 40px;
            background-color: white;
            border-radius: 8px;
            box-shadow: 0 4px 15px rgba(0,0,0,0.1);
            text-align: center;
        }

        h1 {
            color: #2c3e50;
            margin-bottom: 30px;
            font-size: 28px;
        }

        .btn-link {
            display: inline-block;
            background-color: #3498db;
            color: white;
            padding: 12px 25px;
            text-decoration: none;
            border-radius: 5px;
```

```
            font-weight: 500;
            transition: background-color 0.3s;
            margin-top: 20px;
        }

        .btn-link:hover {
            background-color: #2980b9;
        }
    </style>
</head>
<body>
    <!-- Navigation Bar -->
    <div class="navbar">
        <a href="/">Home</a>
        <a href="/predict">Prediction</a>
    </div>

    <div class="container">
        <h1>You have been logged out!</h1>
        <a class="btn-link" href="/">Go to Home</a>
    </div>
</body>
</html>
```

app.py

```
import os
import numpy as np
from flask import Flask, request, render_template
from keras.preprocessing import image
from keras.models import load_model
# Initialize Flask app
app = Flask(__name__)
# Load your trained Keras model
model = load_model("model.h5", compile=False)
```

```
# List of class labels (ensure it matches your model's output order)
labels = [
 'anadenanthera', 'arecaceae', 'arrabidaea', 'cecropia', 'chromolaena',
 'combretum', 'croton', 'dipteryx', 'eucalipto', 'faramea', 'hyptis',
 'mabea', 'matayba', 'mimosa', 'myrcia', 'protium', 'qualea', 'schinus',
 'senegalia', 'serjania', 'syagrus', 'tridax', 'urochloa'
]
# Home route
@app.route('/')
@app.route('/index.html')
def index():
 return render_template('index.html')
# Prediction route
@app.route('/predict', methods=['GET', 'POST'])
def predict():
 if request.method == 'POST':
 file = request.files['image']
 if not file:
 return render_template('predict.html', prediction=None, image_path=None
 # Save uploaded file to static/uploads directory
 basepath = os.path.dirname(__file__)
 upload_folder = os.path.join(basepath, 'static', 'uploads')
 os.makedirs(upload_folder, exist_ok=True)
 upload_path = os.path.join(upload_folder, file.filename)
 file.save(upload_path)
 # Preprocess the image
 img = image.load_img(upload_path, target_size=(128, 128))
 x = image.img_to_array(img)
 x = np.expand_dims(x, axis=0)
 # Predict
 probs = model.predict(x)[0]
pred = np.argmax(probs)
 predicted_label = labels[pred]
 confidence = float(probs[pred] * 100)
 return render_template('predict.html',
 prediction=predicted_label,
 image_path=file.filename,
 confidence=confidence)
 return render_template('predict.html', prediction=None, image_path=None, con
# Logout route
@app.route('/logout.html')
def logout():
 return render_template('logout.html')
# Run the app
if __name__ == "__main__":
 app.run(debug=True)
```