Project Report

**Pollenʼs Profiling – Automated Classification of Pollen Grains**

**Team ID:** LTVIP2025TMID43572

**Team Member:** varshini durga

1. **Introduction**

Pollenʼs 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.

1. **Ideation Phase**
   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.

* 1. **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 |

* 1. **Brainstorming Ideas**

 Use CNN for visual pattern recognition.  Build lightweight Flask web app.

 Handle class imbalance with weighted loss.  Enable image upload + instant prediction.

1. **Requirement Analysis**
   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 |

* 1. **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, server- side 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 |

1. **Project Design**
   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 |

* 1. **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.

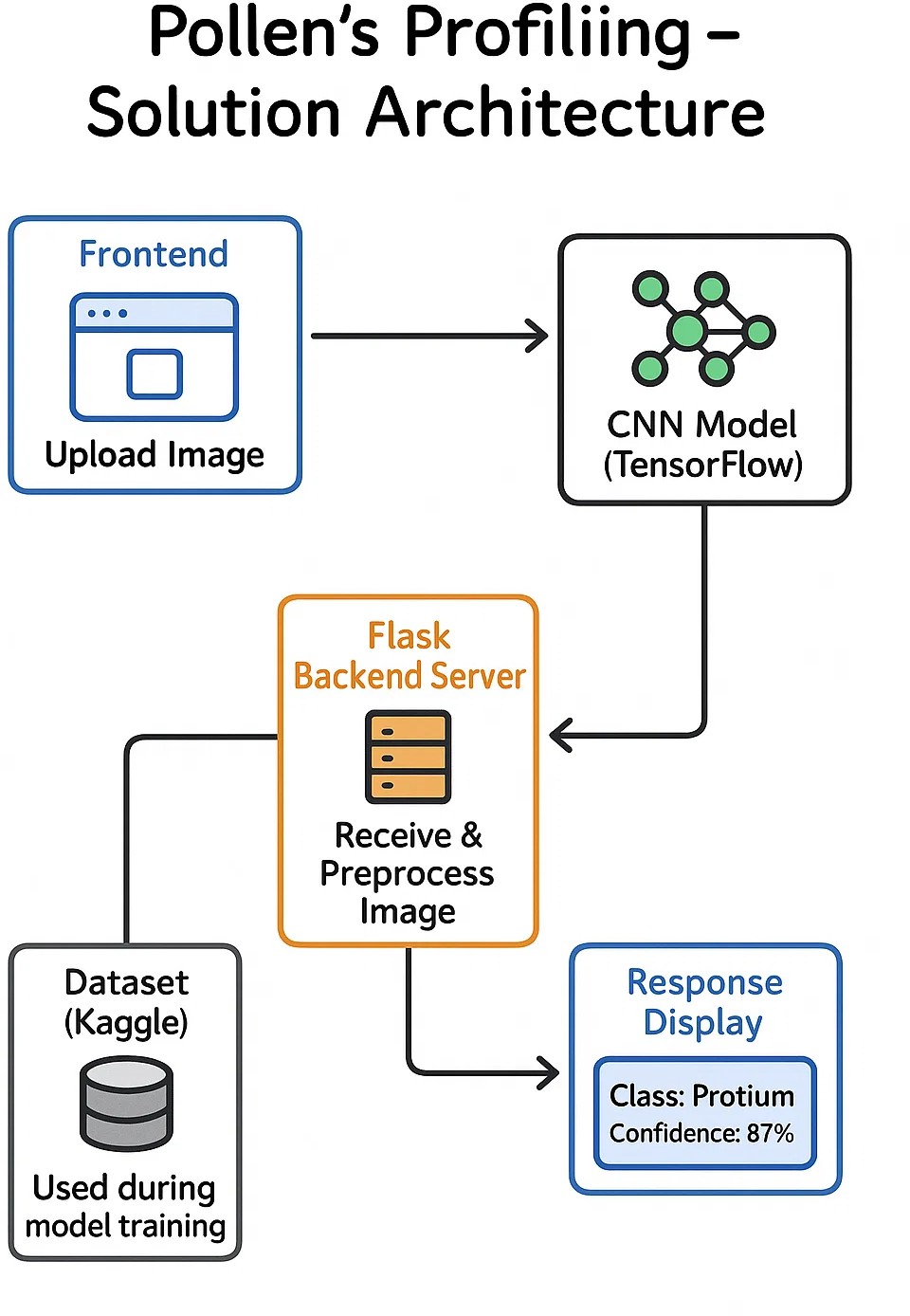
📑 **Proposed Solution Details**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1 | **Problem**  **Statement** | Manual identification of pollen grains is time- consuming, 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 real- time 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, on- prem, 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/big-](https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2025.1507036/full)  [data/articles/10.3389/fdata.2025.1507036/full](https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2025.1507036/full) |

* 1. **Architecture Diagram**

****

1. **Project Planning & Scheduling**
   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 |

* 1. **Tracker**

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

* 1. **Burn Down Chart**

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

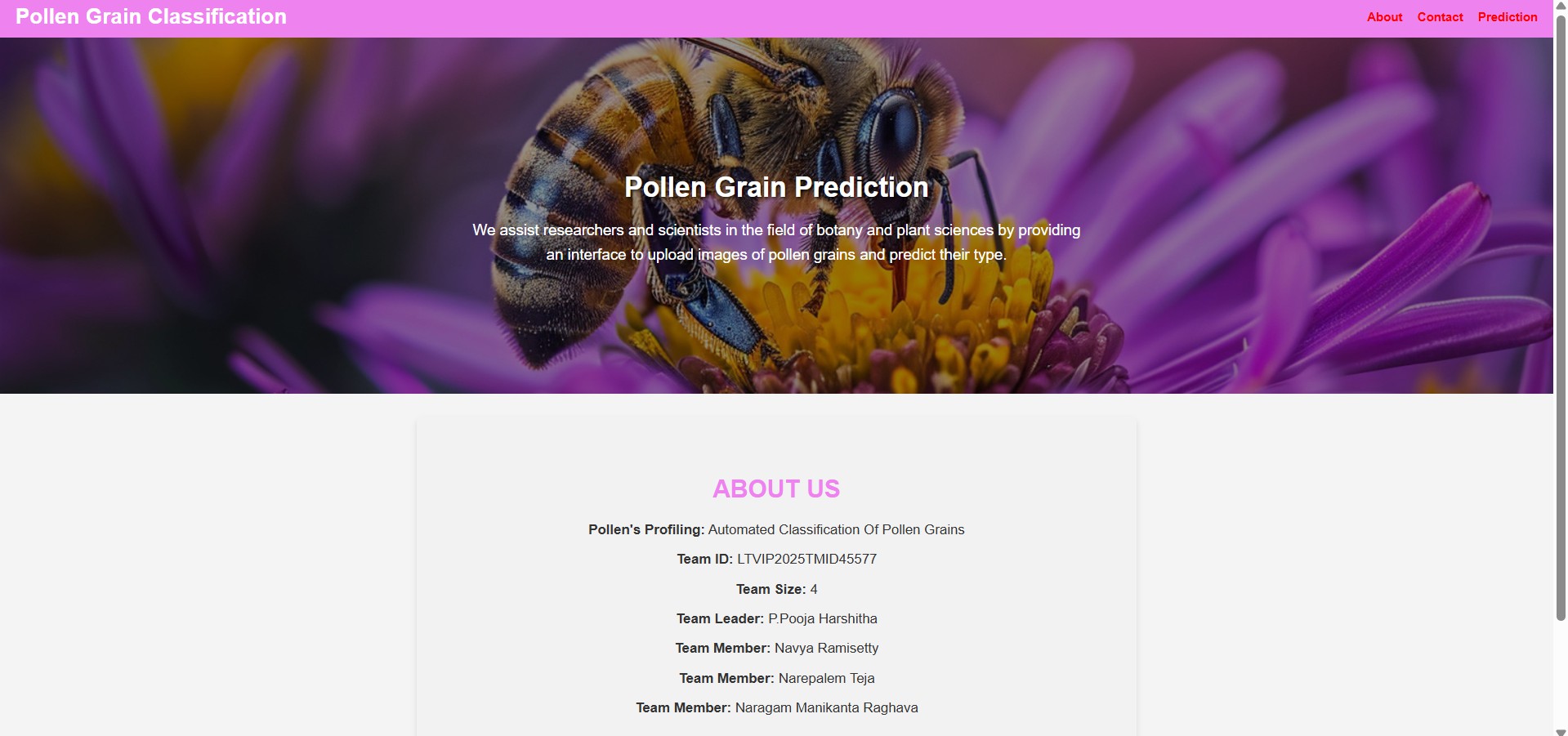
1. **Functional and Performance Testing**

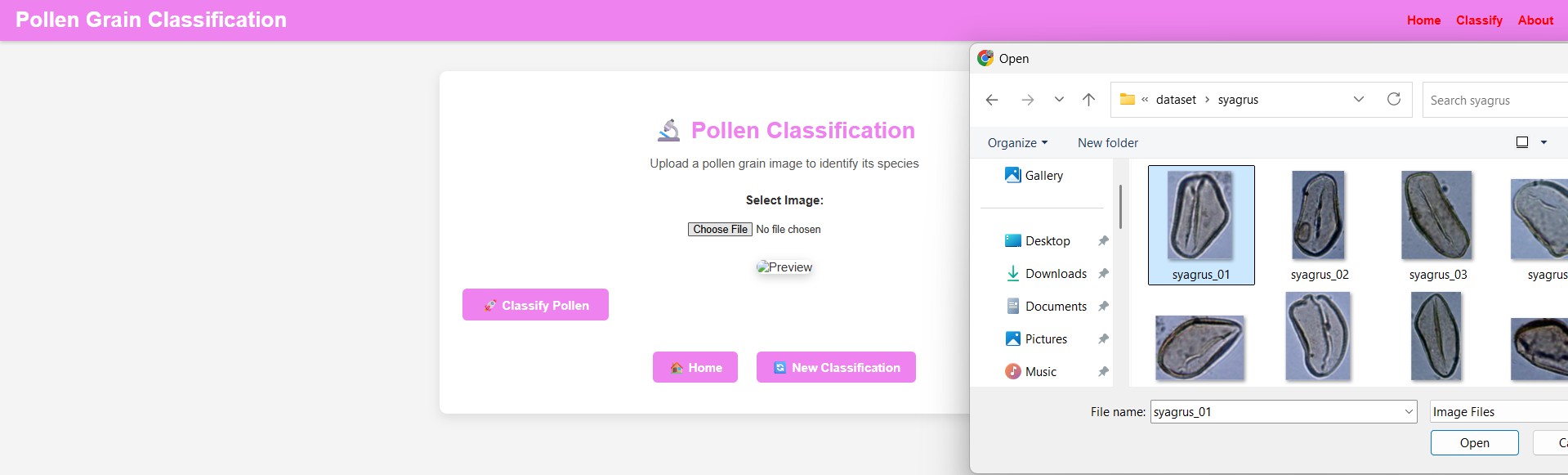
 **Test Accuracy:** 87.6%

 **Loss Value:** 0.707

 **Model:** CNN with weighted loss for imbalanced dataset  **Evaluation Tools:** Classification report, confusion matrix  **Average Inference Time:** 2.8 seconds

1. **Results**



****

1. **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 |

1. **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:

 Mobile integration  Dataset expansion

 Real-time microscope image feeds  Transfer learning upgrades

1. **Appendix**

# GitHub Repository:

[https://github.com/NMRL24/pollens-profiling-automated-classification-of-](https://github.com/NMRL24/pollens-profiling-automated-classification-of-pollen-grains) [pollen-grains](https://github.com/NMRL24/pollens-profiling-automated-classification-of-pollen-grains)

# Demo Video:

[https://drive.google.com/file/d/1kN26Ol\_PGkHjNmTMIRHIAcHumlcSNfup/view?](https://drive.google.com/file/d/1kN26Ol_PGkHjNmTMIRHIAcHumlcSNfup/view?usp=sharing) [usp=sharing](https://drive.google.com/file/d/1kN26Ol_PGkHjNmTMIRHIAcHumlcSNfup/view?usp=sharing)

index.html

!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Pollen Grain Classification</title>

<style> body {

font-family: Arial, sans-serif; margin: 0;

padding: 0;

background-color: #f4f4f4; color: #333;

}

header {

background-color: #EE82EE; padding: 10px 20px;

color: white; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap;

box-shadow: 0 2px 5px rgba(0,0,0,0.2);

}

header h1 

font-size: 28px; font-weight: bold; margin: 0;

}

nav a { color: red;

margin-left: 15px;

text-decoration: none; font-weight: bold; position: relative;

transition: color 0.3s ease;

}

nav a::after { content: '';

position: absolute;

width: 0%; height: 2px;

bottom: -2px; left: 0;

background-color: red; transition: width 0.3s ease;

}

nav a:hover { color: white;

}

nav a:hover::after { width: 100%;

}

.hero {

background-image: linear-gradient(rgba(0,0,0,0.4), rgba(0,0,0,0.4)), background-size: cover;

url("{{ ur

background-position: center; color: white;

text-align: center; padding: 145px 100px;

}

.hero h2 

font-size: 36px; font-weight: bold;

margin-bottom: 20px;

text-shadow: 2px 2px 4px rgba(0,0,0,0.7);

}

.hero p {

font-size: 20px; margin-top: 20px; line-height: 1.6;

max-width: 800px; margin-left: auto; margin-right: auto;

}

.about {

text-align: center; padding: 50px 20px;

background-color: #f2f2f2; color: #333;

margin: 30px auto; max-width: 900px; border-radius: 8px;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

}

.about h2 

font-size: 32px;

margin-bottom: 20px; color: #EE82EE;

}

.about p {

max-width: 800px; margin: 0 auto 10px auto; font-size: 18px;

line-height: 1.6; text-align: center;

}

footer {

background-color: #333; color: white;

text-align: center; padding: 20px; margin-top: 50px;

font-size: 14px;

}

@media (max-width: 768px) {

.hero h2 

font-size: 28px;

}

.hero p {

font-size: 16px;

}

nav a {

display: block; margin: 10px 0;

}

header {

flex-direction: column; align-items: flex-start;

}

}

</style>

</head>

<body>

<header>

<h1Pollen Grain Classification</h1

<nav>

<a href="/"About</a>

<a href="#"Contact</a>

<a href="/predict"Prediction</a>

</nav>

</header>

<main>

<section class="hero">

<h2Pollen Grain Prediction</h2

<p>

We assist researchers and scientists in the field of botany and plant science

</p>

</section>

<section class="about" id="about">

<h2ABOUT US/h2

<p><strong>Pollen's Profiling:</strong> Automated Classification Of Pollen G

<p><strong>Team ID/strong> LTVIP2025TMID45577/p>

<p><strong>Team Size:</strong> 4/p>

<p><strong>Team Leader:</strong> P.Pooja Harshitha</p>

<p><strong>Team Member:</strong> Navya Ramisetty</p>

<p><strong>Team Member:</strong> Narepalem Teja</p>

<p><strong>Team Member:</strong> Naragam Manikanta Raghava</p>

</section>

</main>

<footer>

<p>&copy; 2025 Pollen Grain Classification. All rights reserved.</p>

</footer>

</body>

</html>

predict.html

!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Pollen Classification - Predict</title>

<style> body {

font-family: Arial, sans-serif; margin: 0;

padding: 0;

background-color: #f4f4f4; color: #333;

}

header {

background-color: #EE82EE; padding: 10px 20px;

color: white; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap;

box-shadow: 0 2px 5px rgba(0,0,0,0.2);

}

header h1 

font-size: 28px; font-weight: bold; margin: 0;

}

nav a { color: red;

margin-left: 15px;

text-decoration: none; font-weight: bold; position: relative;

transition: color 0.3s ease;

}

nav a::after { content: '';

position: absolute;

width: 0%; height: 2px;

bottom: -2px; left: 0;

background-color: red; transition: width 0.3s ease;

}

nav a:hover { color: white;

}

nav a:hover::after { width: 100%;

}

.container {

max-width: 900px; margin: 40px auto; padding: 0 20px;

}

.main-card {

background-color: white; border-radius: 10px; padding: 30px;

box-shadow: 0 5px 15px rgba(0,0,0,0.1);

}

.card-header { text-align: center;

margin-bottom: 30px;

}

.card-header h2  font-size: 30px;

color: #EE82EE; margin-bottom: 10px;

}

.card-header p { font-size: 16px; color: #555;

}

.file-upload-wrapper { margin-bottom: 20px; text-align: center;

}

input[type="file"] { padding: 10px; margin-top: 10px;

}

.file-name {

font-style: italic; font-size: 14px; color: #555; margin-top: 5px;

}

.image-preview img { max-width: 100%; max-height: 300px; margin-top: 15px; border-radius: 10px;

box-shadow: 0 4px 12px rgba(0,0,0,0.2);

}

.submit-button {

background-color: #EE82EE;

color: white; border: none; padding: 10px 25px; font-size: 16px;

font-weight: bold; border-radius: 6px; cursor: pointer;

transition: background 0.3s ease;

}

.submit-button:hover {

background-color: #d76add;

}

.loading {

text-align: center; margin-top: 15px;

}

.result-section { margin-top: 30px;

background-color: #f2f2f2; padding: 20px;

border-radius: 10px; text-align: center;

}

.species {

font-size: 24px; font-weight: bold; color: #4CAF50;

}

.confidence {

font-size: 16px; margin-top: 10px;

}

.nav-links {

text-align: center; margin-top: 30px;

}

.nav-links a {

display: inline-block; margin: 10px; padding: 10px 20px;

background-color: #EE82EE; color: white;

text-decoration: none; border-radius: 6px;

font-weight: bold;

transition: background 0.3s ease;

}

.nav-links a:hover {

background-color: #d76add;

}

</style>

</head>

<body>

<header>

<h1Pollen Grain Classification</h1

<nav>

<a href="/"Home</a>

<a href="/predict"Classify</a>

<a href="/#about"About</a>

</nav>

</header>

<div class="container">

<div class="main-card">

<div class="card-header">

<h2🔬 Pollen Classification</h2

<p>Upload a pollen grain image to identify its species</p>

</div>

<form method="POST" enctype="multipart/form-data" id="uploadForm">

<div class="file-upload-wrapper">

<label for="fileInput"><strong>Select Image:</strong></label><br />

<input type="file" name="image" accept="image/\*" required id="fileInput"

<div class="file-name" id="fileName"></div>

<div class="image-preview" id="imagePreview">

<img id="previewImg" src="" alt="Preview">

</div>

</div>

<button type="submit" class="submit-button" id="submitBtn" disabled>🚀

<div class="loading" id="loading" style="display: none;">

<p>Analyzing image...</p>

</div>

</form>

% if prediction and image\_path %

<div class="result-section" id="resultSection">

<div class="species">🌿 {{ prediction }}</div>

% if confidence %

<div class="confidence"Confidence: {{ '%.2f' % confidence %/div>

% else %

<div class="confidence"Confidence: Not Available</div>

% endif %

<div style="margin-top: 15px;">

<img src="{{ url\_for('static', filename='uploads/' + image\_path) }}" alt="Up

</div>

</div>

% endif %

<div class="nav-links">

<a href="/">🏠 Home</a>

<a href="/predict">🔄 New Classification</a>

</div>

</div>

</div>

<script>

const fileInput = document.getElementById('fileInput');

const fileName document.getElementById('fileName');

=

=

const imagePreview const previewImg

=

=

document.getElementById('imagePreview'); document.getElementById('previewImg');

const submitBtn document.getElementById('submitBtn');

const loading document.getElementById('loading');

=

fileInput.addEventListener('change', function (e) {

=

const file if (file) {

e.target.files[0];

fileName.textContent = `Selected: ${file.name}`;

fileName.style.display 'block';

=

const reader reader.onload

=

=

new FileReader(); function (e) {

previewImg.src e.target.result;

=

=

imagePreview.style.display

};

reader.readAsDataURL(file);

=

'block';

submitBtn.disabled

} else {

false;

fileName.style.display 'none';

=

imagePreview.style.display 'none';

=

=

}

});

submitBtn.disabled

true;

document.getElementById('uploadForm').addEventListener('submit', function ( if (!fileInput.files.length) {

e.preventDefault();

alert('Please select an image.');

} else {

submitBtn.style.display 'none'; loading.style.display 'block';

}

});

</script>

</body>

</html>

=

=

logout.html

!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Logout</title>

<style> body {

margin: 0;

font-family: Arial, sans-serif; background-color: #f4f4f4; text-align: center;

color: #333;

}

header {

background-color: #EE82EE; padding: 10px 20px;

color: white; display: flex;

justify-content: space-between; align-items: center;

box-shadow: 0 2px 5px rgba(0,0,0,0.2);

}

nav a { color: red;

margin-left: 15px;

text-decoration: none; font-weight: bold; position: relative;

}

nav a::after { content: '';

position: absolute; width: 0%;

height: 2px; bottom: -2px; left: 0;

background-color: red; transition: width 0.3s ease;

}

nav a:hover { color: white;

}

nav a:hover::after { width: 100%;

}

.logout-section {

background-image: linear-gradient(rgba(0,0,0,0.4), rgba(0,0,0,0.4)),

url("{{ url\_for('static', filename='xbee.jpg') }}");

background-size: cover;

background-position: center; color: white;

padding: 120px 20px; min-height: 500px;

display: flex;

flex-direction: column; justify-content: center; align-items: center;

}

.logout-section h2  font-size: 36px;

margin-bottom: 20px;

text-shadow: 2px 2px 4px rgba(0,0,0,0.7);

}

.logout-section a {

background-color: #007C80; color: white;

padding: 10px 25px; font-size: 16px; border-radius: 6px; text-decoration: none; font-weight: bold;

box-shadow: 0 0 8px rgba(0,0,0,0.3);

}

.logout-section a:hover {

background-color: #005f61;

}

footer {

background-color: #333; color: white;

text-align: center; padding: 20px; font-size: 14px;

}

</style>

</head>

<body>

<header>

<h1Pollen Grain Classification</h1

<nav>

<a href="/"About</a>

<a href="#"Contact</a>

<a href="/predict"Prediction</a>

</nav>

</header>

<div class="logout-section">

<h2You have successfully logged out.</h2

<a href="/"Back to Home</a>

</div>

<footer>

<p>&copy; 2025 Pollen Grain Classification. All rights reserved.</p>

</footer>

</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)