Artifical Intelligence And Machine Learning

Project Documentation

PROJECT TITLE:"Pollen Profilling :Automated Classification Of Pollen Grains"

1.Introduction

Team ID:LTVIP2025TMID32756

Team Members:

• Narla Bhuvaneswari[Member 1] – Full Stack Developer (Frontend + Backend Integration)

• Nelli Sravani[Member 2] – AI/ML Developer (Model Training and Prediction)

• Balu Mungamuri[Member 3] – UI/UX Designer (Frontend Components and Layout)

• Yaswanth Muthyala [Member 4] – Database Engineer (MongoDB Management and Integration)

2. Project Overview

Purpose:

This project aims to develop an intelligent system that can automatically classify different types of pollen grains using

machine learning and image processing techniques. This will help botanists, researchers, and environmental scientists quickly

identify pollen species to assist in studies related to biodiversity, allergen monitoring, and climate change.

Features:

Image upload and preprocessing pipeline

CNN-based classification model

RESTFUL API for predictions

Interactive web interface for uploading and visualizing results

Reporting of classification confidence scores

Option to download classification reports

3. Architecture

Frontend:

The frontend is developed using React.js with a component-based architecture. Each section (navbar, upload, result) is modular and reusable. Axios is used to send HTTP requests to the backend. Tailwind CSS ensures responsiveness and clean styling across devices.

Backend:

The backend is powered by Flask (Python) for easy integration with TensorFlow. It handles:
Receiving microscope images from the frontend
Preprocessing images for the model
Running predictions using a pre-trained model
Returning the predicted species and confidence score
Connecting to MongoDB for storing results

Database:

MongoDB is used as a NoSQL database for flexible document storage. Collections include:
predictions – Stores image metadata, prediction result, and timestamp
species_data – Contains taxonomic information about pollen species

4. Setup Instructions

Prerequisites:

To run this project, you need the following installed:

• Node.js (for frontend)

• Python 3.8+ (for backend)

• MongoDB (local or MongoDB Atlas)

• npm (Node package manager)

• pip (Python package manager)

• Installation:

1. Clone the project repository from GitHub.

2. Navigate to the frontend/ folder:

o Install dependencies using npm install.

3. Navigate to the backend/ folder:

o Create and activate a Python virtual environment.

o Install dependencies using pip install -r requirements.txt.

4. Set environment variables (e.g., MONGO_URI, FLASK_APP).

5. Ensure MongoDB is running locally or use an Atlas connection string.

5. Folder Structure

Client:

```
frontend/
 src/
 components/
 Navbar.jsx
 UploadForm.jsx
 ResultDisplay.jsx
 App.jsx
 index.js
 public/
 index.html
 package.json
```

Server:

```
backend/
 app.py
 model/
 model.h5
 utils/
 image_preprocessor.py
 model_loader.py
 routes/
 predict_route.py
  database/
    mongo_connection.py
  uploads/
  .env
```

requirements.txt

6. Running the Application

To run the full application locally:

• Frontend:

• Navigate to the frontend directory.

• Run npm install (once) to install dependencies.

• Run npm start to launch the development server.

Accessible at: http://localhost:3000

• Backend:

• Navigate to the backend directory.

• Activate your Python environment.

• Run python app.py or flask run to start the server.

7. API Documentation

POST /predict

Description: Accepts an uploaded pollen image and returns classification
Method: POST
Request Format: multipart/form-data
Parameters: file (image file to be classified)
Response Example:

```
json
{
  "prediction": "Pinus sylvestris",
  "confidence": 0.95,
  "family": "Pinaceae",
  "allergenic": true
}
```

-

8. Authentication

Authentication is optional in the current implementation. If enabled:

Method: JSON Web Token (JWT)
Flow:
Researcher logs in with institutional credentials
Server generates a token and returns it
Token is stored in localStorage on the frontend
Protected API routes validate the token before granting access

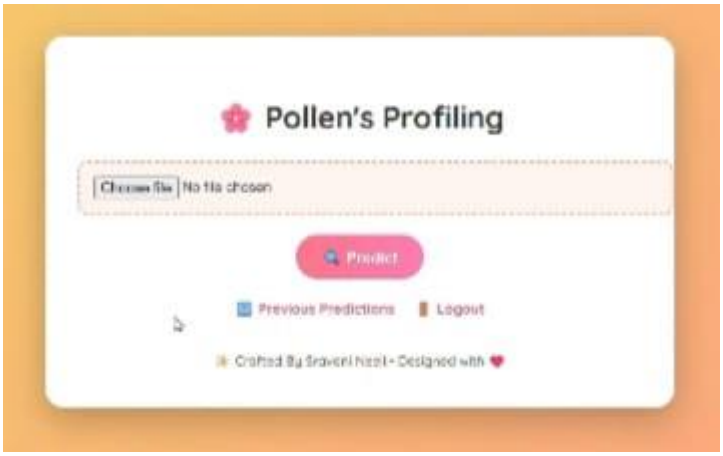
Security: Token expires after a set time to prevent misuse

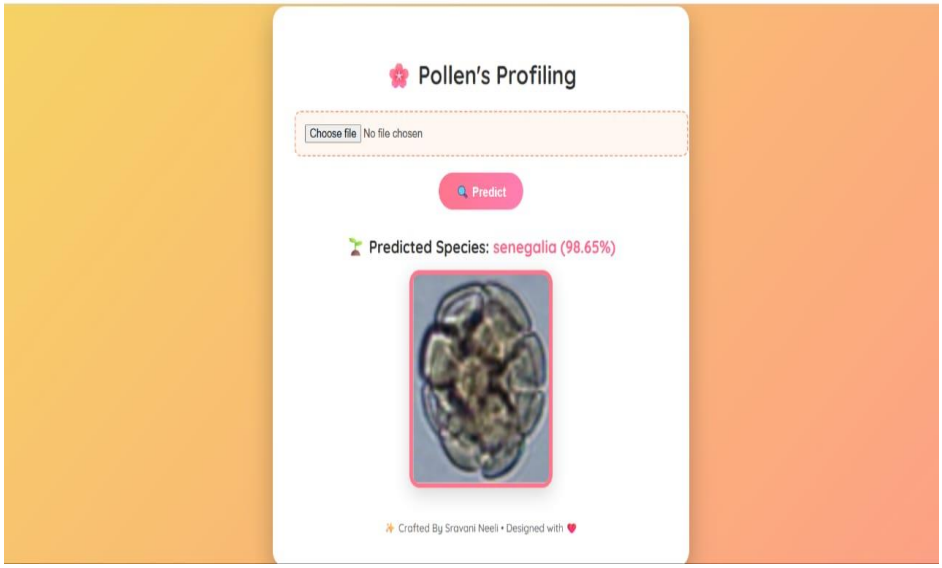
Benefits of Adding Authentication:

Research data integrity
Personalized prediction history
Institutional access control
Secure data sharing capabilities
User-specific research dashboards

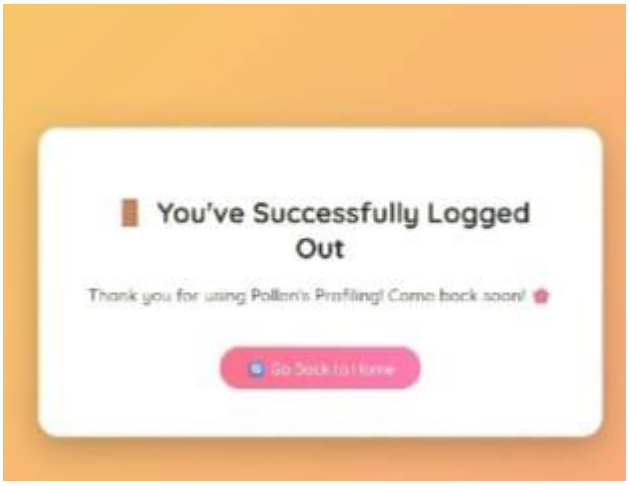
9. User Interface:

HOME PAGE





PREDICTED PAGE



LOG OUT PAGE

10. Testing

Testing Strategy:

To ensure scientific accuracy and reliability of the Pollen Profiling application, rigorous testing was implemented:

Unit Testing:
Python functions for microscope image preprocessing
Model loading and prediction logic verification
Taxonomic data validation tests

Integration Testing:
Full research workflow testing:
Microscope image upload → API call → Model prediction → Scientific result display
Verified MongoDB correctly stores research data with proper metadata

End-to-End Testing:

Manually tested complete researcher workflow across browsers (Chrome, Edge, Firefox)
Verified behavior with various microscope image formats and qualities
Stress tested with high-resolution microscope images

Functional Testing:
Image upload for various microscope formats
Taxonomic classification display
Allergenic potential indicators
Error handling for out-of-focus images
Research data management

Tools Used:

Tool/Library Purpose

Postman Testing API endpoints with microscope images

Vscode Scientific code development
PyTest / unittest Backend testing for research application
Jest Component-level testing for React frontend
MongoDB Compass Research data verification
Browser DevTools Performance analysis for microscope images

11.Demo link:

https://drive.google.com/file/d/14GmZwC33iWF4YB08iHKDeXt0i9Uda1Rc/view?usp=sharing

12. Known Issues

Current limitations of the Pollen Profiling system:

Microscope Image Quality Dependency:
Classification accuracy is highly dependent on image focus, staining, and magnification quality. Poor sample preparation may reduce accuracy.

Rare Species Limitations:
The model may struggle with pollen from endangered or rarely encountered plant species.

Size Limitations:
High-resolution microscope images may require optimization for processing.

Taxonomic Depth:
Current model classifies to genus level; species-level identification requires additional training data.

13. Future Enhancements

Planned scientific improvements:

Digital Microscope Integration:
Direct API connections to laboratory microscope systems for real-time analysis.

Advanced Taxonomic Classification:
Hierarchical classification from family → genus → species with confidence levels at each rank.

Research Collaboration Features:
Institutional user management with shared research databases and team projects.

Model Retraining via Research Feedback:
Peer-reviewed correction system to improve model accuracy through researcher input.

3D Pollen Analysis:
Integration with z-stack microscope images for three-dimensional pollen grain analysis.

Ecological Analytics:
Geographic distribution mapping and seasonal pollen bloom forecasting.

THANK YOU