# 1. Introduction

Project Title: Pollen's Profiling: Automated Classification of Pollen Grains

Team Members:

1.k.Praveen Kumar

2.K.Sravanthi

3.P.Sadhana

4.N.Mohan sai

# 2. Project Overview

Purpose:

the classification of pollen grains using advanced image processing and machine learning techniques. By leveraging deep learning algorithms and image analysis methods

Features:

* Handcrafted Features
* Automatically Learned Hierarchical Features
* Quantitative Statistical Measures
* Qualitative Descriptions
* Robustness
* Attention Mechanisms

# 3. Architecture

Fundamental Components of a CNN Architecture:

These architectures are designed to efficiently learn features from image data and classify them.

Backend:

API Endpoints, Image Processing & Preprocessing, AI Model Inference

Database:

MongoDB with Mongoose ORM for image metadata and prediction

# 4. Setup Instructions

Prerequisites:

* Node.js
* MongoDB

Installation:

1. Clone the repository
2. Install dependencies using npm3. Set up environment variables in a `.env` file

Environment Variables:

MONGO\_URI=your\_mongo\_connection

JWT\_SECRET=your\_secret

PORT=5000

# 5. Folder Structure

Client:

* Client-side applications
* For data collections
* Entity

Server:

* routes/: API endpoints
* controllers/: Logic for routes
* models/: MongoDB schemas
* server.js: Entry point

# 6. Running the Application

Frontend:

Development/Local Testing

Backend:

Django,API End points,Docker Image

# 7. API Documentation

* API name, Base URL
* Endpoints (The Core of Your API)
* Flask (with extensions)
* Fast API (Highly Recommended for Python API)
* Markdown/README.md

# 8. Authentication

JWT (JSON Web Token)-based authentication.

* Data Anonymization/De-identification
* Secure Deserialization
* Secure CI/CD Pipeline

**9. User Interface**

Note: No screenshots provided as per user's request.

# 10. Testing

Manual testing with Postman and browser.

Basic form and route validations verified.

# 11. Demo Link

View project demo:

https://drive.google.com/file/d/1-C6iiTMkvlpHuerT3PvSnncv9BLQ-Zn3/view?usp=drivesdk

# 12. Known Issues

* Limited Datasets
* mage Quality Variation
* Intra-species Variation
* Overfitting

# 13. Future Enhancements

* Increased Taxonomic Resolution
* Robustness to Image Variation
* Multi-Modal Data Integration
* Batch Processing API

# 14. Sample Code:Flask/Fast API App

# In your FastAPI or Flask endpoint:

from your\_image\_processing\_module import preprocess\_pollen\_image # Assuming you save the above code as a module

@app.post("/predict")

async def predict\_pollen\_grain(file: UploadFile = File(...)):

# ... (file validation and reading bytes as before) ...

try:

image\_bytes = await file.read()

processed\_image = preprocess\_pollen\_image(image\_bytes, target\_size=(224, 224))

# The processed\_image (NumPy array) is now ready to be fed into your ML model

# Example: prediction = your\_model.predict(np.expand\_dims(processed\_image, axis=0))

# np.expand\_dims is often needed to add a batch dimension (e.g., (1, 224, 224, 3))

# ... (rest of your prediction logic) ...

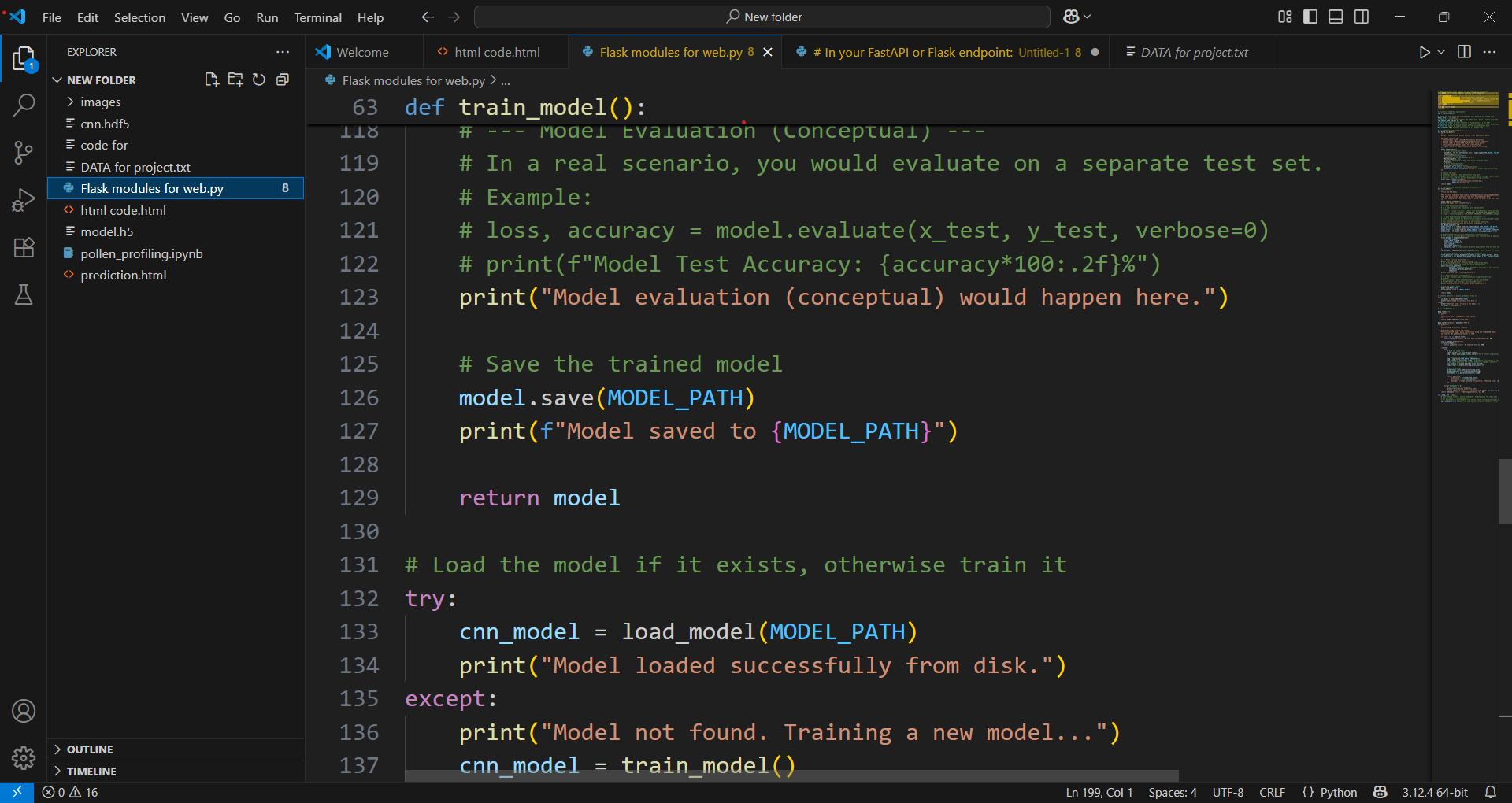
except ValueError as e: # Catch specific errors from preprocess\_pollen\_image

raise HTTPException(status\_code=status.HTTP\_400\_BAD\_REQUEST, detail=str(e))

except RuntimeError as e:

raise HTTPException(status\_code=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR, detail=str(e))

# ... (rest of the endpoint)



HTML CODE:

