Project

Project Title:

Pollen's Profiling: Automated Classification of Pollen Grains

Team ID : LTVIP2025TMID34592

Team Name : PollenoVision

Team Members:

- Amulya Dasari
- A Sai Sree
- Addanki Sai Kishore
- Addanki Chandana Krishnaveni

Phase-1: Brainstorming & Ideation

Objective:

To develop a deep learning-based system for automated classification of pollen grains using image processing techniques, aiding in biodiversity research, allergy diagnostics, and agricultural productivity.

Key Points:

1. Problem Statement:

- Manual identification of pollen is time-consuming and prone to error.
- Pollen classification is essential in allergy treatment, environmental science, and crop science.
- There is no accessible and reliable automated system to classify pollen images effectively.

2. Proposed Solution:

- A Convolutional Neural Network (CNN)-based system that classifies pollen grain images based on shape, size, and texture.
- Integration with a Flask-based web application to allow real-time predictions.

3. Target Users:

- Environmental researchers studying plant biodiversity.
- Healthcare professionals diagnosing pollen allergies.
- Agricultural researchers improving crop pollination.

4. Expected Outcome:

- A browser-based classifier that is accurate, fast, and user-friendly.
- Reduced human effort and errors in pollen classification.
- Valuable support for health, ecology, and agriculture sectors.

Phase-2: Requirement Analysis

Objective :

Define the technical and functional requirements of the pollen classification application.

Key Points:

1. Technical Requirements:

- Programming Language: Python
- Libraries: NumPy, TensorFlow, Keras, Flask, OpenCV
- Model file: `cnn.hdf5` / `model.h5`
- Frontend: HTML, CSS, JavaScript

2. Functional Requirements:

- Upload image of pollen grain via web interface.
- Preview the image before submission.
- Display predicted pollen type with confidence.

3. Constraints & Challenges:

- Class imbalance in the dataset.
- Image blur and resolution inconsistencies
- Ensuring mobile responsiveness and fast prediction speeds.

Phase-3: Project Design

Objective:

Define the architecture and user experience flow of the project.

Key Points:

1. User Interface (UI):

- Built with HTML/CSS and JS
- Allows image selection and result viewing

2. Flask Backend:

- Handles file upload
- Loads pre-trained model
- Sends prediction to frontend

3. Machine Learning Model Layer:

- CNN architecture trained using image data
- Softmax activation with multiple pollen class outputs
- Trained and saved as `cnn.hdf5` and `model.h5`

4. Dataset Layer:

- Directory of images categorized by pollen type
- Preprocessing using ImageDataGenerator
- Augmented training and validation set.

Phase-4: Project Planning

Sprint	Task	Priority	Duration	Member(s)	Output
1	Dataset Preprocessing	High	3 hrs	Member 1	Cleaned dataset
1	EDA & Image Previews	High	2 hrs	Member 2	Visual insights
2	CNN Model Training	High	4 hrs	Member 3	Trained model (h5/hdf5)
2	Flask App Integration	Medium	3 hrs	Member 4	Web prediction page
3	UI Testing & Responsiveness	Medium	2 hrs	Members 2, 3	Mobile-friendly UI
3	Final Deployment & Demo	Low	1 hr	Entire Team	Working web app (local)

Phase-5: Project Development

Objective:

Develop and integrate the CNN model and deploy the web application.

Key Points:

- Used TensorFlow/Keras to build CNN with multiple Conv2D and Dropout layers.
- Data was augmented and preprocessed using `ImageDataGenerator`.
- Model saved in both `model.h5` and `cnn.hdf5` format.
- Flask app routes created for upload, prediction, and result display.
- Responsive frontend styled with CSS and interactive JavaScript (`main.js`).

Challenges & Solutions:

- **Image Noise:** Used normalization and resizing.
- **Overfitting:** Used dropout layers and more training data.
- **UI Design:** Tested responsiveness across devices.

Phase-6: Functional & Performance Testing

Test ID	Category	Scenario	Expected Outcome	Status
TC-001	Functional	Clear image	Accurate class prediction	✅Passed
TC-002	Functional	Rare class	Correct prediction	✅Passed
TC-003	Performance	Bulk upload	Prediction in < 3 seconds	⚠ Slightly slow
TC-004	UI Responsiveness	Mobile device rendering	Full responsiveness	✅Passed
TC-005	Deployment	Run on local server	Real-time prediction	Deployed