Pollen's Profiling: Automated Classification of Pollen Grains

Team Members:

Name 1: Dama Akhil **(Team Lead)**

Name 2**:** Bhuvinaidu
Name 3: Shillapalli Sravanthi
Name 4: Koleti Raghavi

Phase 1: Brainstorming & Ideation

Objective:

Analyze challenges in manually identifying and classifying pollen grains under microscopy. Explore how transfer learning and image classification can automate the identification process for environmental, botanical, and medical applications.

- **Key Points :**

1. **ProblemStatement:**
   Manual examination of pollen grains is labor-intensive, subjective, and requires significant expertise. Differences in pollen morphology are subtle and often misclassified by non-experts.
2. **ProposedSolution:**
   "Pollen's Profiling" uses deep learning models like VGG16 or MobileNet to classify microscope images of pollen into their respective categories. Transfer learning improves accuracy even on smaller datasets.
3. **Target Users :**
   - Botanists and palynologists
   - Environmental monitoring agencies
   - Allergy research labs
   - Academic and research institutions
   - Agricultural and crop research centers

4. **ExpectedOutcome:**

   An intelligent application that can classify pollen images quickly and accurately, supporting research, allergy forecasts, and automated laboratory processes.

Phase 2: Requirement Analysis

Objective:

Define the software, hardware, and functional requirements for the pollen classification system. Consider image clarity, dataset diversity, and classification challenges.

- **Key Points:**

5. **Technical Requirements:**

   - Languages: Python 3.10+

   - Frameworks: TensorFlow, Keras

   - Tools: Google Colab, Jupyter Notebook, VS Code

   - Hardware: GPU (NVIDIA recommended), 16 GB RAM

6. **Functional Requirements:**

   - Upload pollen microscope image

   - Classify into specific pollen types

   - Show prediction confidence

   - Display image with label

   - Download result/report (Optional)

7. **Constraints & Challenges:**

   - Similar morphology among different species

   - Varying microscope image quality

   - Imbalanced dataset

   - Need for transparency in classification output for research acceptance

Phase 3: Project Design

Objective:

Design a modular system with easy input-output handling and high interpretability for research professionals.

- **Key Points:**

8. **System Architecture:**

   - Input Module → Image Preprocessing

- Classification Module → Transfer Learning
- Output Module → Display prediction & confidence

9. **User Flow:**

   User uploads image → Preprocessing → Classification → Confidence shown → (Optional: Report Export)

10. **UI/UX Considerations:**
    - Minimal interface suitable for lab settings
    - Color-coded prediction for clarity
    - Mobile and web support
    - Clear feedback for low-quality images

Phase 4: Project Planning (Agile)

Objective:

Follow Agile development with iterative testing, collaboration, and refinement.

- **Key Points:**

11. **Sprint Planning:**
    - Sprint 0: Literature review & dataset sourcing
    - Sprint 1: Data cleaning & augmentation
    - Sprint 2: Model training with base CNN
    - Sprint 3: UI setup
    - Sprint 4: Backend integration
    - Sprint 5: Final testing & enhancements
12. **Task Allocation:**
    - ML Engineer: Model architecture, training
    - Data Engineer: Dataset preparation
    - UI Developer: Interface and display
    - Backend Developer: Integration logic
    - QA Engineer: Accuracy, edge case testing
13. **Timeline & Milestones:**
    - Week 1–2: Dataset finalized
    - Week 3–4: Model training completed
    - Week 5: Frontend-backend integration
    - Week 6: Final validations & testing

Phase 5: Implementation

Objective:

Deploy the model into a working application using a clean tech stack.

- **Key Points:**

14. **Technology Stack:**

    - Frontend: HTML, CSS, Streamlit

    - Backend: Flask API

    - Model: Keras with TensorFlow

    - Deployment: Google Colab / Heroku / Docker

15. **Implementation Steps:**

    1. Collect data (e.g., Kaggle, microscopy datasets)

    2. Preprocess & augment images

    3. Load pre-trained model with custom layers

    4. Train and validate model

    5. Save `.h5` model

    6. Build prediction pipeline

    7. Display classification results

16. **Challenges & Fixes:**

    - Overfitting: Mitigated with data augmentation

    - Similar classes: Improved with fine-tuning

    - Performance: Used MobileNet for optimized speed

Phase 6: Functional & Performance Testing

Objective:

Ensure the model works reliably across microscope images, maintains high precision, and serves its intended scientific purpose.

- **Key Points:**

17. **Tests Performed:**

    - Prediction accuracy per class

    - Image batch testing

    - UI performance and clarity

    - Edge testing: blurred or out-of-focus samples

    - Device/resource usage

18. **Results & Fixes:**

    - Accuracy up to ~90–93% achieved

    - UI bugs resolved

    - Alerts added for uncertain predictions

19. **FinalValidation:**

    Model demonstrated strong generalization. Ready for academic demos or lab pilot testing. Supports reproducible research workflows.

20. **Deployment Options:**

    - Google Colab (Demo)

    - Streamlit + Flask + Heroku (Public tool)

    - Docker container (For offline lab use)