**Automated Classification Of Pollen Grains**

Team ID : LTVIP2025TMID35957

Team Member : Galla pujitha

**Phase 1: Brainstorming & Ideation**

**Objective:**

The objective of this phase is to clearly identify the problem statement and define the project's purpose and impact. It aims to understand how the project will address a real-world issue and benefit its target users.

**Key Points:**

Problem Statement:  
Manual identification and classification of pollen grains under microscopes is time-consuming and error-prone. This affects researchers, environmentalists, and agriculturists relying on accurate pollen identification.  
  
Proposed Solution:  
A Deep Learning-based automated image classifier for pollen grains that can predict the class/species based on uploaded images through a web interface.  
  
Target Users:  
- Botanists  
- Environmental Scientists  
- Researchers  
- Agricultural Experts  
  
Expected Outcome:  
An efficient and accurate pollen classifier that automates identification, reduces manual errors, and improves research productivity.

**Phase 2: Requirement Analysis**

**Objective:**

To identify technical and functional requirements necessary to develop the pollen grain classification system.

**Key Points:**

Technical Requirements:  
- Language: Python  
- Libraries: TensorFlow, Keras, OpenCV, Flask, Pandas, NumPy, Scikit-learn  
- Tools: Jupyter Notebook, VS Code  
  
Functional Requirements:  
- Upload pollen grain images  
- Predict species/class  
- Show prediction result on the web interface  
  
Constraints & Challenges:  
- Dataset imbalance  
- Handling incorrect file uploads  
- Preventing model overfitting

**Phase 3: Project Design**

**Objective:**

To design the system architecture, user flow, and interface that support the functionality of the pollen classification system.

**Key Points:**

System Architecture:  
User → Web Interface → Flask Backend → Trained CNN Model → Output Prediction  
  
User Flow:  
1. Open the web app  
2. Upload pollen image  
3. Click Predict  
4. View result  
  
UI/UX Considerations:  
- Clean interface with simple buttons  
- Display predictions clearly  
- Error handling for invalid files

**Phase 4: Project Planning (Agile Methodology)**

**Objective:**

To create an effective development strategy by dividing work using agile methodologies.

**Key Points:**

Sprint Planning:  
- Sprint 1: Data collection & preprocessing  
- Sprint 2: Model training  
- Sprint 3: Flask app development  
- Sprint 4: Testing & Deployment  
  
Task Allocation:  
- Jagadish: Data handling, model creation  
- Manikanta: Flask backend integration  
- Vijay: Frontend development  
- Pavan: Testing  
- Lakshmanarao: Documentation  
  
Timeline:  
- Week 1: Dataset preparation  
- Week 2: Model training  
- Week 3: App development  
- Week 4: Testing & Deployment

**Phase 5: Project Development**

**Objective:**

To develop and integrate all modules of the project including data processing, model creation, and deployment.

**Key Points:**

Technology Stack:  
- Python, TensorFlow, Keras, Flask, HTML, CSS  
  
Development Process:  
- Dataset collected from Kaggle  
- Preprocessing images (resize, normalize)  
- Model built using CNN  
- Flask backend created for predictions  
- Frontend HTML for user interaction  
  
Challenges:  
- Dataset imbalance  
- Upload directory missing errors  
- Resolved image format and preprocessing errors

**Phase 6: Functional & Performance Testing**

**Objective:**

To ensure the project functions as expected and performs optimally with real-world data.

**Key Points:**

Test Cases:  
- Valid image upload → Correct prediction  
- Invalid file upload → Shows error  
- No file uploaded → Shows warning  
  
Bug Fixes:  
- Upload folder errors resolved  
- Path inconsistencies corrected  
  
Validation:  
Model achieved ~85% accuracy. The web application runs smoothly and returns correct predictions.

**Final Submission**

Checklist:  
✅ Project Report (This document)  
⬜ Demo Video (Link: \_\_\_\_\_\_\_\_\_\_\_)  
⬜ GitHub Repository (Link: \_\_\_\_\_\_\_\_\_\_\_)  
⬜ Presentation Slides  
  
Source Code and Dataset details are appended below.  
  
Dataset Source: Kaggle