Project Documentation

1. Introduction

- **Project Title:** Pollens Profiling – Automated Classification of Pollen Grains
- **Team Members:**
  - Pallela Bhanu
  - Sailaja Jandhyam
  - Palika Venkata Lakshmi Sriya
  - Nedunuri Swathi

2. Project Overview

- **Purpose:** To automate the process of classifying pollen grain images using deep learning, reducing the manual effort and increasing classification accuracy.
- **Features:**
  - User registration and login
  - Upload pollen grain images
  - Automated classification using CNN model
  - History tracking of classified samples
  - Real-time results with confidence scores

3. Architecture

- **Frontend:**
  - Developed using React.js
  - Responsive UI for image upload and result display
  - Axios for API communication
- **Backend**
  - Built using Node.js with Express.js
  - Handles routing, model invocation, and user management
- **Database:**
  - MongoDB used for storing user data and classification history
  - Mongoose ORM for schema definition and queries

4. Setup Instructions

- **Prerequisites:**
  - Node.js >= v14
  - MongoDB Community Server

- Python 3.8+ (for ML model)

- **Installation:**
  - Download and install **Node.js** and **MongoDB** on your system.
  - Clone the project repository from GitHub to your local machine.
  - Navigate to the backend (server) folder and install the dependencies using npm install.
  - Then, navigate to the frontend (client) folder and run npm install to set up the React environment.
  - Make sure MongoDB is running, and set any required environment variables.
  - Once the setup is done, start both frontend and backend servers.

5. Folder Structure

- **Client:**
  - src/components – UI components
  - src/pages – View pages
  - src/services – Axios API calls
- **Server:**
  - routes/ – Express route handlers
  - controllers/ – Business logic
  - models/ – MongoDB schemas
  - ml_model/ – Python CNN model

6. Running the Application

- Provide commands to start the frontend and backend servers locally.
  - **Frontend:**
    ```
    cd client
    nmp start
    ```
  - **Backend:**
    ```
    cd server
    npm start
    ```

7. API Documentation

Example Endpoint: POST /api/classify

- Request: { image: <base64> }
- Response: { class: "Pine Pollen", confidence: 92.1 }

Other APIs:

- POST /api/login
- POST /api/register
- GET /api/history

8. Authentication

- JWT-based token authentication
- Tokens are stored in browser local storage
- Middleware used to validate user tokens for protected routes

9. User Interface

- Upload image screen
- Dashboard with classification results
- Login and Registration forms
- Responsive design for mobile and desktop

10. Testing

- Manual testing for frontend interactions
- Postman for backend API testing
- Unit tests for core functions (optional)

11. Screenshots or Demo

https://drive.google.com/file/d/1F5VWzo-tuYAcsaVa5hXHp33JKu52gQYA/view?usp=drivesdk

12. Known Issues

- Limited support for very low-resolution images
- UI does not yet handle multiple uploads in batch mode

13. Future Enhancements

- Add support for more pollen species
- Integrate live webcam capture for microscope
- Deploy using Docker + CI/CD
- Add batch image classification
- Incorporate feedback-based retraining of model