Full Stack Development with MERN

Project Documentation format

1. Introduction

• Project Title: Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

• Team Members: List team members and their roles.

Here’s your team list with roles:

**Team Members and Roles**

* **Sonthi Harinadh**– Team Lead
* **Shaik Sameer** – Team Member
* **Shaik Naseema** – Team Member
* **Shaik Ayesha Begum** – Team Member

2. Project Overview

• Purpose: Briefly describe the purpose and goals of the project.

The purpose of this project is to build an intelligent system that quickly and accurately detects and classifies fruits and vegetables as fresh or rotten. This will help in improving food safety, minimizing manual sorting errors, and supporting efficient supply chain management.

• Features: Highlight key features and functionalities.

* **Image Upload:** Allows users to upload images of fruits and vegetables for classification.
* **Smart Classification:** Accurately identifies whether the produce is fresh or rotten using transfer learning.
* **Fast Prediction:** Provides instant classification results.
* **User-Friendly Interface:** Simple and easy-to-navigate web application.
* **Image History (Optional):** Stores uploaded images and results for user reference.
* **Web Deployment:** Accessible through a browser using Flask backend.

3. Architecture

• Frontend: Describe the frontend architecture using React.

Provides a simple user interface for image upload and displays classification results.

• Backend: Outline the backend architecture using Node.js and Express.js.

 Handles API requests, processes image uploads, and returns prediction results.



• Database: Detail the database schema and interactions with MongoDB.

* Stores user details, uploaded images, and classification results for future reference.

4. Setup Instructions

• Prerequisites: List software dependencies (e.g., Node.js, MongoDB).

* **Node.js** (v14 or above) – For backend server
* **MongoDB** – For database storage
* **React.js** – For frontend
* **Flask** (if using Python model separately)
* **Python** (if integrating ML model backend)
* **Basic IDE:** VS Code or any preferred editor

**1. Clone the Repository**

bash

git clone <your-repository-link>

**2. Navigate to Frontend**

bash

**3. Navigate to Backend**

bash

**4. Set Up Environment Variables**

* In the backend folder, create a .env file:

**5. Start MongoDB Server**

bash

**6. Access Application**

Open the frontend in your browser at:

• Installation: Step-by-step guide to clone, install dependencies, and set up the

environment variables.

5. Folder Structure

• Client: Describe the structure of the React frontend.

* **Components:**
  + Home.js – Project introduction and start button
  + Predict.js – Image upload and submission
  + Result.js – Displays prediction result
  + Navbar.js – Navigation bar
  + About.js – Project details and team info
* **API:**  
  Sends images to backend and fetches classification results.

• Server: Explain the organization of the Node.js backend.

* **server.js:** Starts Express server and connects routes.
* **/routes:** Handles API requests (upload, get results).
* **/controllers:** Processes image and prediction logic.
* **/models:** MongoDB schema for storing images and results.
* **/uploads:** Stores uploaded images.

**APIs:**

* POST /upload – Image upload & prediction
* GET /results – Fetch prediction history

6. Running the Application

• Provide commands to start the frontend and backend servers locally.

o Frontend: npm start in the client directory.

cd client

npm start

o Backend: npm start in the server directory.

cd server

npm start

7. API Documentation

• Document all endpoints exposed by the backend.

• Include request methods, parameters, and example responses.

8. Authentication

• Explain how authentication and authorization are handled in the project.

• Include details about tokens, sessions, or any other methods used.

9. User Interface

• Provide screenshots or GIFs showcasing different UI features.

10. Testing

• Describe the testing strategy and tools used.

11. Screenshots or Demo

• Provide screenshots or a link to a demo to showcase the application.

https://drive.google.com/file/d/1E-2SKLkf07DeBCLIdmPswQlzM2LtV32B/view?usp=drivesdk

12. Known Issues

• Document any known bugs or issues that users or developers should be aware of.

13. Future Enhancements

• Outline potential future features or improvements that could be made to the project.

**Future Enhancements**

* **Mobile Application:** Develop an Android/iOS app for easy access.
* **Offline Mode:** Enable offline image processing without internet dependency.
* **Real-Time Camera Integration:** Automate sorting using conveyor belt cameras.
* **Multi-Class Detection:** Detect levels of ripeness or specific fruit/vegetable diseases.
* **User History:** Allow users to track their previous uploads and results.
* **Voice Command Integration:** Enable hands-free image submission and result reading.