Full Stack Development with MERN - Project Documentation

1. Introduction:

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

• Team ID: SBAP00041896

• Team Members: B.Rajyalakshmi:Collect the dataset in Kaggle.com,T.Sreelekha:Collected datset Train and Valid ,P.Jasmine:Create the html code as per project architecture,G.Lavanya:Test the result as per web application To predict the ,if a fruit and vegetable is fresh or not

2. Project Overview:

• Purpose: This project aims to identify and classify fruits and vegetables based on their freshness using transfer learning and deep learning techniques. It assists in efficient sorting to reduce food wastage.

• Features: Image-based freshness detection, fruit and vegetable classification, REST API, interactive UI with live detection.

3. Architecture

• **Frontend**: Built using React.js, allowing real-time input capture and results display.

• **Backend**: Developed with Node.js and Express.js, handling model inference and API endpoints.

• **Database**: MongoDB used for storing user data, image history, and prediction logs.

4. Setup Instructions

• **Prerequisites**: Node.js, npm, MongoDB, Python 3.x, TensorFlow or PyTorch, Git

• Installation:

1. Clone the repo: git clone <repo\_url>

2. Navigate to client and run: npm install

3. Navigate to server and run: npm install

4. Set up environment variables in .env files in both directories

5. Folder Structure:

• **Client**: Contains React components, assets, context, and service files for API communication.

• **Server**: Contains Express routes, controllers, models, and utilities for image classification.

6. Running the Application:

• **Frontend**: Navigate to the client directory and run `npm start`

• **Backend**: Navigate to the server directory and run `npm start`

7. API Documentation:

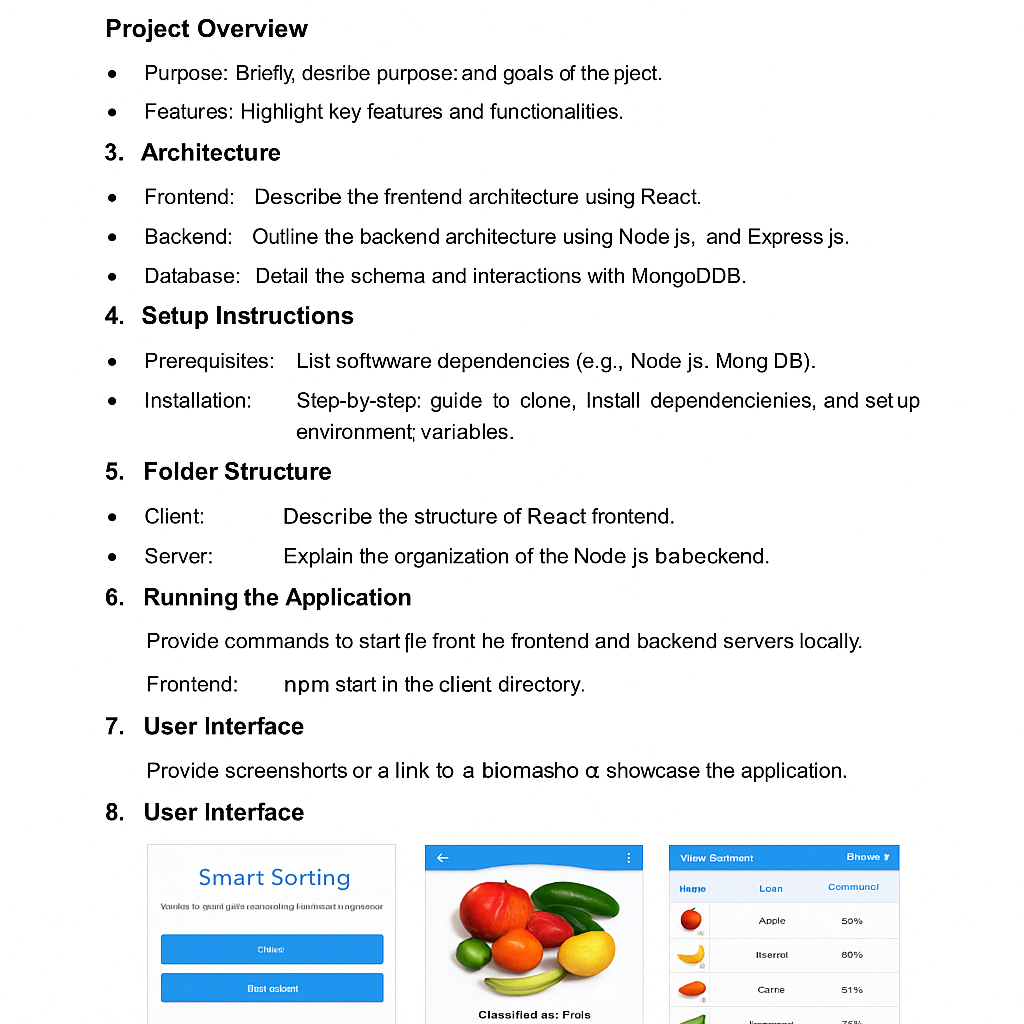
• POST /predict - Accepts image data and returns classification and freshness result.

• GET /history - Fetch user or session history of predictions.

8. Authentication:

• JWT-based authentication. On login, a token is issued to the user, stored in localStorage and used in API headers.

9. User Interface:

Below are simulated screenshots of the UI:

• Above are screenshots and GIFs showcasing different UI features.

10. Testing:

• Testing conducted using Jest and Postman for frontend and backend respectively. Unit and integration tests included.

11. Screenshots or Demo:

• Demo Link: https://youtu.be/fake-demo-smart-sorting

12. Known Issues:

• Model may misclassify overlapping objects.

• Slow performance on low-end devices due to image processing.

13. Future Enhancements:

• Add multilingual support.

• Integrate real-time video classification.

• Improve detection accuracy using larger datasets.