1. Introduction

* Project Title: Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables
* Team Members:
* Z.Prathysha - Frontend Developer
* P.V.Pavani, M.Lohitha- Backend Developer

C.Sangeetha - Database & DevOps

* Shaik.Amreen - ML Engineer

2. Project Overview

* Purpose: To automate the identification of rotten fruits and vegetables using computer vision andtransfer learning, improving sorting efficiency in the supply chain.
* Features:
* Real-time image classification of produce
* Integration with sorting hardware (optional)
* Dashboard for analysis and statistics
* User authentication

3. Architecture

* Frontend: Built using React, follows component-based architecture with state management throughRedux and API interaction using Axios.
* Backend: Developed with Node.js and Express.js, exposes RESTful APIs and handles

authentication and image classification requests.

* Database: MongoDB used for storing user data, logs, and image metadata. Interacted viaMongoose ORM.

4. Setup Instructions

* Prerequisites:
* Node.js (v16+)
* MongoDB (local or cloud) - Python (for ML model training)
* Installation:

1. Clone repository
2. Run `npm install` in /client and /server folders
3. Set up `.env` with MongoDB URI and JWT secret
4. Start ML server (Python Flask if separate)
5. Folder Structure

* Client:
* /src
* /components
* /pages
* /services
* /assets
* Server:
* /routes
* /controllers
* /models
* /middleware

6. Running the Application

* Frontend: `cd client && npm start`
* Backend: `cd server && npm start`

7. API Documentation

* Endpoints:
* POST /api/login - Authenticates user
* GET /api/images - Fetches image metadata
* POST /api/classify - Uploads image and returns classification
* Example Response: { "label": "Rotten", "confidence": 0.92 }

8. Authentication

* Uses JWT tokens stored in HTTP-only cookies.
* Middleware checks for valid token before accessing protected routes.

9. User Interface

* Features a dashboard with image upload, classification results, and statistics.
* Responsive design using Tailwind CSS.

10. Testing

* Tools: Jest for frontend, Mocha & Chai for backend.
* Manual testing of model accuracy with test dataset.

11. Screenshots or Demo

- Screenshots and demo available at: [Include link or folder]

12. Known Issues

* Slight latency in image classification
* Accuracy drops in poor lighting conditions

13. Future Enhancements

* Integration with robotic sorting arms
* Real-time mobile app for field usage
* Multi-language support