Full Stack Development with MERN – Project Documentation

---

1. Introduction

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

Team Members:

- ✓ **J. Amrutha Lakshmi** (Team Leader)
- ✓ **A. Venkata Vinay**(Team Member)
- ✓ **B. Chandhan Kumar**(Team Member)
- ✓ **D. Venkata Nivas Reddy**(Team Member)

---

2. Project Overview

Purpose: To develop a full-stack application that classifies fruits and vegetables as healthy or rotten using transfer learning and enables end users to test predictions via a responsive web interface.

Features:

- Image upload and classification (healthy/rotten)
- Pre-trained model (VGG16) for predictions
- Authentication system for users
- Data stored and retrieved from MongoDB
- React-based UI with responsive design

---

3. Architecture

Frontend:

- Built using React.js
- UI for image upload and prediction results
- Axios for API communication

Backend:

- Node.js with Express.js
- Routes for image handling and user authentication
- Model prediction handled on the server side

Database:

- MongoDB
- Stores user profiles, login credentials, and logs of predictions

4. Setup Instructions

Prerequisites:

- Node.js and npm
- MongoDB (local or cloud like MongoDB Atlas)

Installation:

```
# Clone the
repository git clone
<repo-url> cd
project-folder

# Install dependencies for
frontend cd client
npm install

# Install dependencies for backend
cd ../server
npm install

# Create a `.env` file in server with the following:
MONGO_URI=<your_mongodb_connection>
JWT_SECRET=<your_secret_key>
```

---

5. Folder Structure

Client:

```
client/
├── public/
├── src/
│   ├── components/
│   ├── pages/
│   └── App.js
```

Server:

```
server/
├── controllers/
├── routes/
```

```
├── models/
├── app.js
```

---

6. Running the Application

Frontend:

```
cd client
npm start
```

Backend:

```
cd server
npm start
```

---

7. API Documentation

POST /api/predict

- Request: Multipart image file
- Response: Prediction result

POST /api/login

- Body: { email, password }
- Response: Auth token

POST /api/register

- Body: { name, email, password }
- Response: User created

---

8. Authentication

- JSON Web Tokens (JWT) used for session management
- Token issued on login and validated on protected routes
- Stored in localStorage on frontend

---

9. User Interface

- Image upload component
- Prediction result display
- Authentication pages (Login/Register)

Screenshots will be attached in the final document.

---

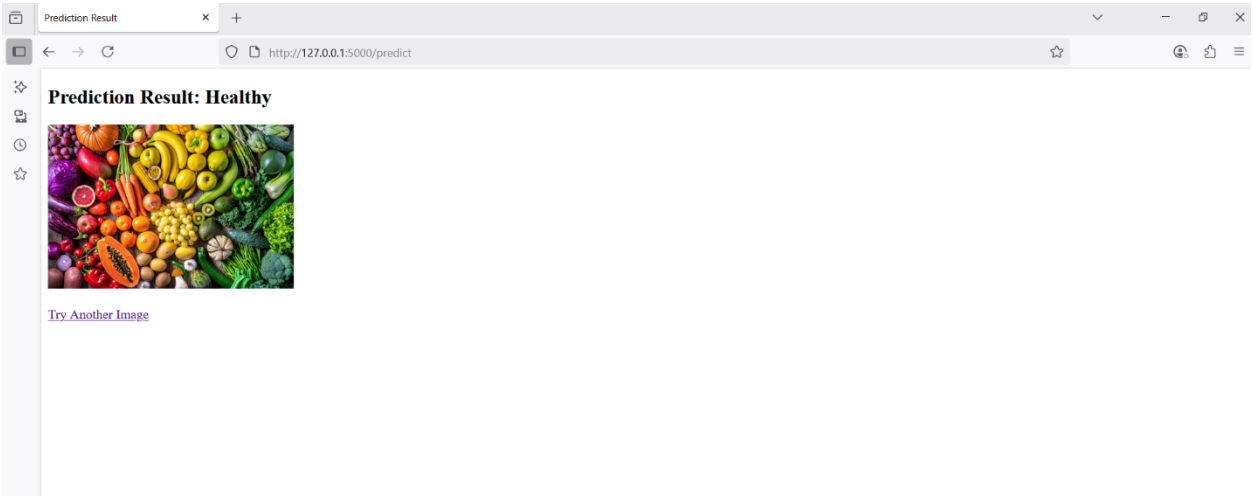
10. Testing

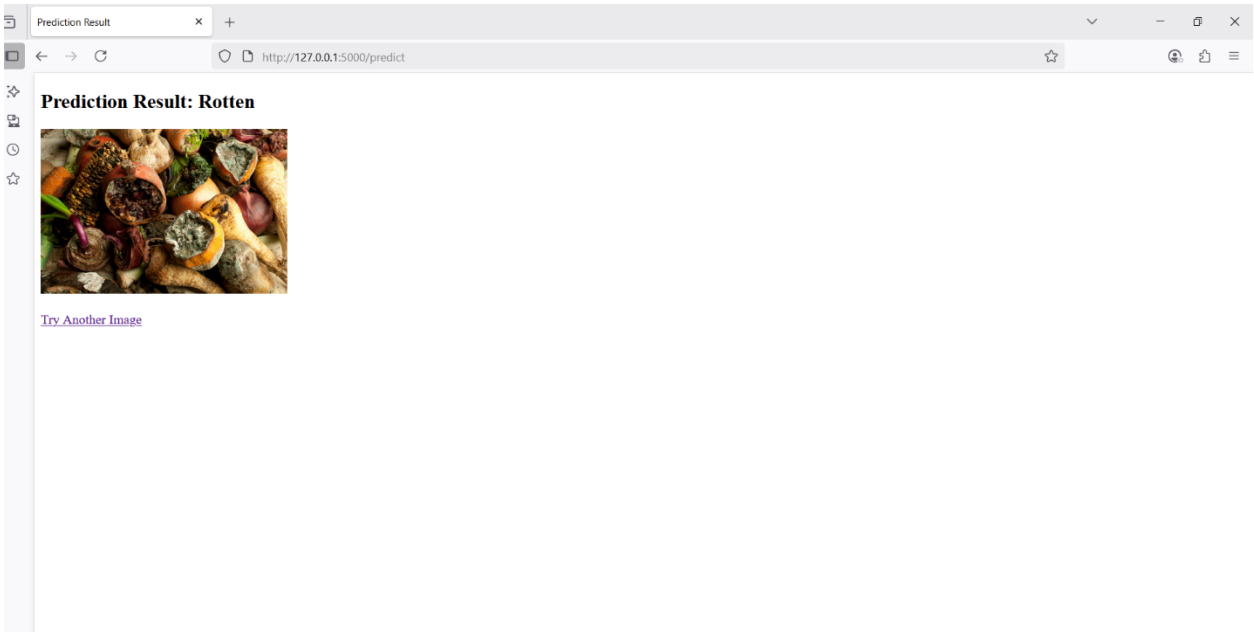
- Manual testing of UI & API endpoints
- Postman used for backend testing
- Frontend tested using React Developer Tools

---

11. Screenshots or Demo

- Flask Model Integration
- Upload to Predict Output
- Auth Pages (Login/Register)





Demo Link: https://github.com/jadaamruthalakshmi/Smart-Sorting-Project/blob/main/Project%20demonstration%20video/Demo%20video.mp4

---

12. Known Issues

- No drag-and-drop upload yet
- No dark mode support in UI
- Deployment pending for cloud

---

13. Future Enhancements

- Implement drag & drop UI for image upload
- Add admin dashboard for managing users
- Deploy frontend & backend on Render/Netlify/Heroku
- Improve UI/UX with animations and real-time progress