AIML Project Documentation

Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

1. Introduction

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

Team Members:

1. A. Jyothendra Sai Ram
2. B. Bindu Bhargavi
3. B. Siddhardha Bheem
4. B. Himanth

2. Project Overview

Purpose:
To develop a system that automatically detects and classifies rotten fruits and vegetables using transfer learning techniques. The goal is to reduce food waste and improve quality control.

Features:

- Image-based detection using deep learning
- Real-time prediction and feedback
- User authentication and prediction history
- Responsive web interface
- Dashboard visualization for analysis

---

3. Architecture

Frontend:
Built using React.js. It includes components for image upload, result display, and dashboard analytics. Axios is used for API requests. Tailwind CSS handles UI styling.

Backend:
Developed using Node.js and Express.js. It handles API routing, image processing, and connects with the ML model and MongoDB.

Database:
MongoDB stores user data, prediction logs, and image metadata. Mongoose is used to define schemas and interact with the database.

---

4. Setup Instructions

Prerequisites:

- Node.js
- MongoDB
- React
- Express
- Mongoose
- Python (for ML model if applicable)

Installation Steps:

1. Clone the repository
2. Navigate to the root directory
3. Run npm install in both client/ and server/ directories
4. Create a .env file with MongoDB URI and server port
5. Start the backend and frontend servers

5. Folder Structure

Client (React Frontend):

- components/ – Reusable UI components
- pages/ – Page-level React components
- services/ – API service files
- assets/ – Static files and images

Server (Node.js Backend):

- routes/ – Defines API endpoints

- controllers/ – Contains business logic
- models/ – Mongoose schemas
- server.js – Entry point

6. Running the Application

- **Frontend:**
  Run npm start in the client/ directory
- **Backend:**
  Run npm start in the server/ directory

7. API Documentation

- POST /api/predict
  - **Description:** Uploads an image and returns prediction
  - **Request Body:** FormData (image file)
  - **Response:** { label: "rotten", confidence: 0.93 }
- POST /api/login
  - **Description:** Authenticates user and returns JWT
- GET /api/history
  - **Description:** Retrieves prediction history for authenticated user

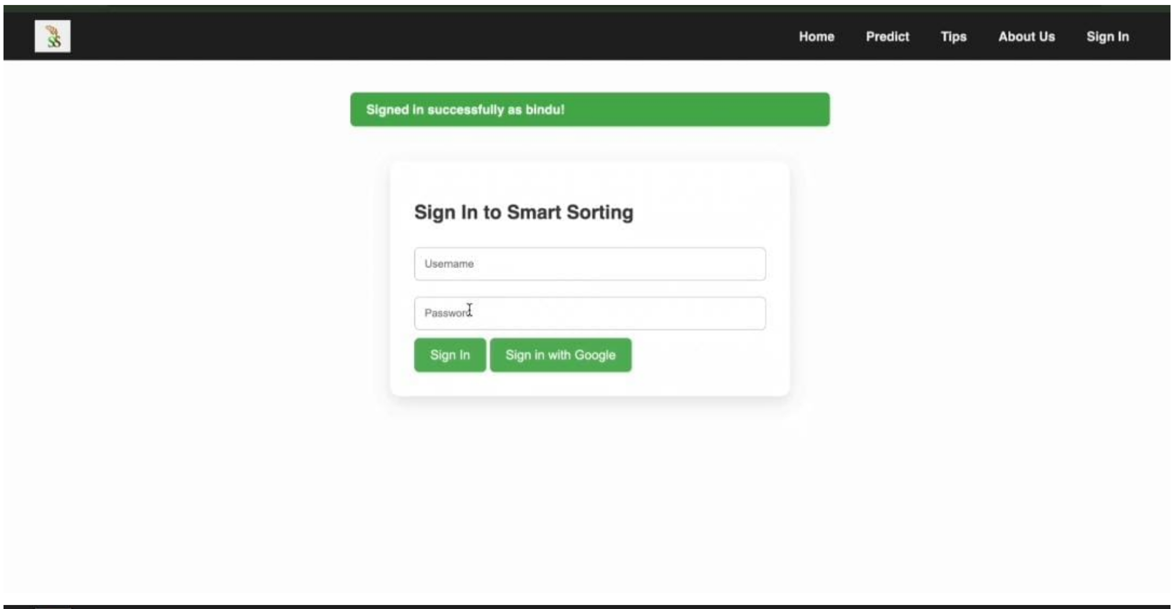
8. Authentication

- Uses **JWT (JSON Web Token)** for secure user authentication
- Tokens are issued during login and required for accessing protected routes
- Passwords are securely stored using hashing (e.g., bcrypt)

9. User Interface

- Clean, responsive design using Tailwind CSS
- Image upload interface
- Real-time result display

- Dashboard with prediction history and statistics

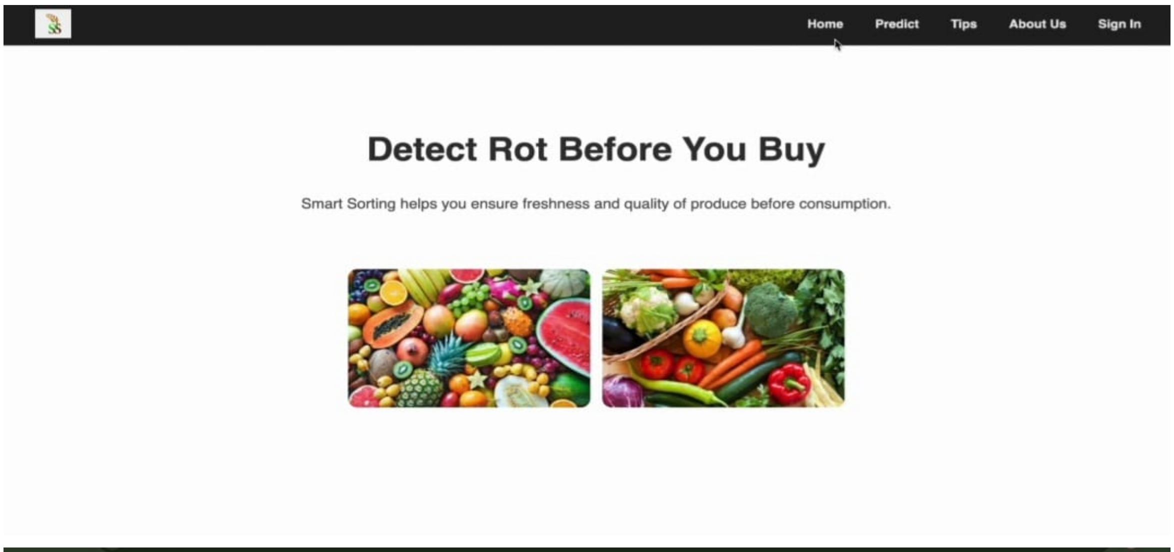


Signed in successfully as bindu!

Sign In to Smart Sorting

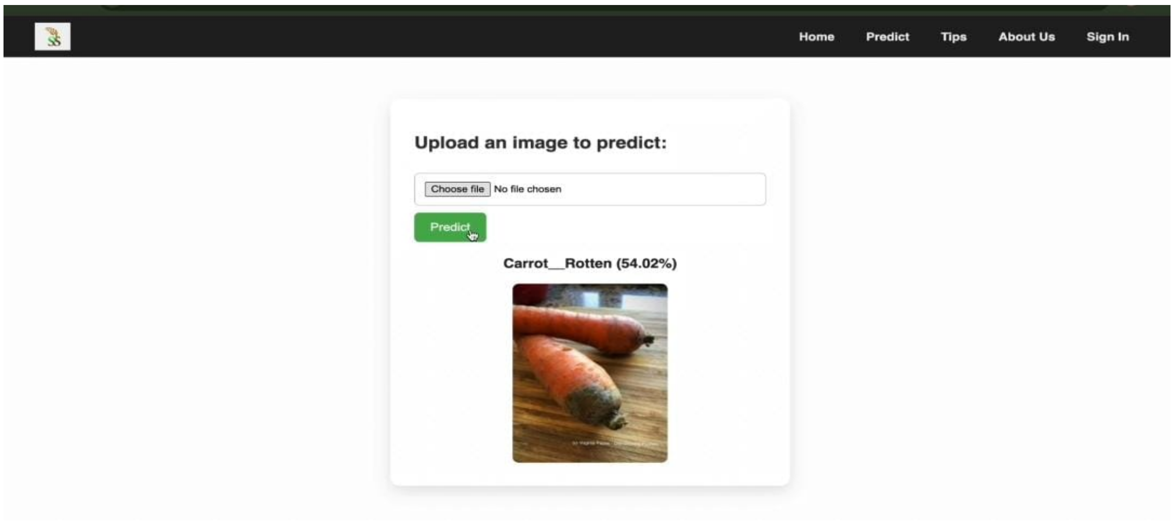
Username

Password



Detect Rot Before You Buy

Smart Sorting helps you ensure freshness and quality of produce before consumption.



Upload an image to predict:

Carrot__Rotten (54.02%)

10. Testing

- **Frontend Testing:**
  Unit tests written with **Jest**
- **Backend Testing:**
  API tests performed using **Mocha**
  Manual API testing done via **Postman**

---

11. Known Issues

- Model may misclassify images under poor lighting conditions
- Large image files can delay predictions
- Currently, no multi-language support

---

12. Future Enhancements

- Improve model accuracy with more diverse training data
- Develop a mobile app version
- Add multi-language interface support
- Integrate with real-time smart sorting hardware