ShopSmart - Digital Grocery Store Web App

---

1. Introduction

Project Title: ShopSmart

Team Members:

- Aritakula Nava Durga Surya Bhagavan : Full Stack Developer

---

2. Project Overview

Purpose: ShopSmart is a MERN stack-based digital grocery web application that aims to streamline the online grocery shopping experience. It caters to customers, sellers, and administrators, providing them with separate access and roles to perform their respective tasks efficiently.

Features:

- User Registration/Login
- Role-based access (Customer, Seller, Admin)
- Product listing and categorization
- Add to cart, checkout, and order placement
- Admin dashboard for user and order management

---

3. Architecture

Frontend: React.js with Axios for API communication and React Router for navigation.

Backend: Node.js with Express.js; APIs for authentication, product handling, and order processing.

Database: MongoDB (using MongoDB Atlas); connected via Mongoose ODM.

---

4. Setup Instructions

Prerequisites:

- Node.js (v14 or higher)
- npm
- MongoDB Atlas account or local MongoDB setup

Installation:

```
# Clone the repository (if from GitHub)
git clone <your-repo-url>

# Or extract the downloaded folder
cd code

# Backend Setup
cd Backend
npm install

# Frontend Setup
cd ../Frontend
npm install
```

Environment Variables: Create a `.env` file in the `Backend/` folder with the following:

```
MONGO_URI=mongodb+srv://
Suryabhagavan:<your_password>@cluster0.w2yhhtn.mongodb.net/ShopSmart?
retryWrites=true&w=majority&appName=Cluster0
```

---

5. Folder Structure

Frontend:

- `src/`
- `components/`
- `pages/`
- `services/`
- `App.js`

Backend:

- `db/`: MongoDB connection logic
- `models/`: Mongoose schemas
- `routes/`: API routes
- `controllers/`: Business logic
- `index.js`: Entry point

---

6. Running the Application

Backend:

```
cd Backend
node index.js
```

Frontend:

```
cd Frontend
npm start
```

---

7. API Documentation

User Authentication:

- POST `/api/register`
- POST `/api/login`

Product:

- GET `/api/products`
- POST `/api/products` (Admin/Seller)

Order:

- POST `/api/order`
- GET `/api/orders` (Admin/User specific)

---

8. Authentication

- JWT (JSON Web Token) used for session management.
- Tokens stored in browser localStorage.
- Protected routes for admin and seller access.

---

9. User Interface

- Responsive and clean UI using basic CSS and Bootstrap.
- Intuitive product cards, cart interface, and dashboards.

---

10. Testing

- Manual testing of all major user flows
- Postman used for API testing

11. Screenshots or Demo

- Login/Register Page
- Product Listing
- Cart Checkout
- Admin Dashboard

12. Known Issues

- No payment gateway integration yet
- Basic error handling for form validation

13. Future Enhancements

- Add Razorpay/Stripe payment integration
- Improve UI/UX with advanced styling
- Add product search and filtering
- Implement email notifications