**🛒 Grocery Webapp-**

Welcome to our grocery web application! This project is designed to deliver a smooth and convenient online shopping journey for users. Whether you’re a tech-savvy user, a homemaker managing daily needs, or someone who loves online shopping, our app caters to all.

Our grocery web app provides an easy-to-navigate interface and a user-centric design, allowing customers to:

Explore various product categories

View detailed product descriptions

Add items to their shopping cart

Proceed through a secure and efficient checkout process

We focus on delivering a seamless experience that prioritizes user satisfaction.

The backend of our application is packed with powerful features to support both sellers and admins. Sellers can effortlessly manage product listings, stock details, and incoming orders. Meanwhile, admins have tools to handle customer feedback, monitor transactions, and maintain overall platform stability and performance.

Security is a top priority. We ensure that all user information is protected, and personal data is handled with confidentiality. Our aim is to create a secure, trustworthy environment for digital shopping.

We are glad to have you join us in this project and are excited about what we’ll achieve together. Get ready to build a user-friendly, efficient grocery web app and enhance your development skills along the way.

**🚀 Features:**

**👤 User Side**

* User registration and login
* Browse groceries with images, prices, and descriptions
* Add/remove items from the shopping cart
* Place orders
* Responsive and user-friendly UI

**🔐 Admin/Seller Side**

* Admin login panel
* Add/Edit/Delete products
* View and manage orders
* Product stock tracking

**Architecture**

Technical Architecture:

**(First Here Diagram Comes)**

The technical design of the grocery web application follows a client-server architecture, where the frontend acts as the client and the backend functions as the server.

The frontend is in charge of user interface, experience, and interaction — it displays content and gathers user input.

The backend manages data storage, business rules, and external integrations like payment gateways and databases.

The system is broken down into different services such as:

Shopping Cart Service

Product Catalog Service

User Service

Order/Shipping Service

Payment Gateway Service

Cart & Product Database

Catalog and User Database

These services interact through APIs, allowing smooth and secure communication between frontend and backend components. The use of APIs ensures efficient data sharing and system performance across all layers.

This modular structure improves maintainability, allows scalability, and ensures that each component can be updated independently without disrupting the overall functionality.

ER Diagram:

**(First Here Diagram Comes)**

The Entity-Relationship (ER) diagram for the grocery web application provides a clear visual of how different components within the system—such as users, products, categories, orders, and reviews—are interconnected.

It showcases the relationships among various database entities, helping to illustrate how data flows throughout the application. This representation is crucial for understanding the overall database schema and the way different modules interact within the app.

By analyzing the ER diagram, developers can easily comprehend the database structure and ensure smooth implementation of backend functionality.

**Project Structure-**

**(Here the screenshot of files created)**

This project is built using Angular and follows a modular folder-based structure. Below is a short description of the main folders and files involved:

src/app/components: Includes reusable components designed for the user-facing part of the application. These may include components for login, registration, product listings, cart, order history, feedback, product details, and more.

src/app/modules: Organizes the app into logical feature modules. For example, the admin module is structured with specific components like add-category, add-product, admin-dashboard, feedback, orders, and others for better separation of concerns.

src/app/app-routing.module.ts: Manages application routing. It maps routes to their respective components, defining which view loads based on the navigation path.

src/app/app.component.ts / .html / .css: Represents the root component of the app. It sets up the shell for the entire application layout and serves as the starting point for rendering.

**Pre-requisites -**

To build a full-stack Grocery web app using Angular, Node.js, and MongoDB, the following tools and technologies must be installed and configured. These are the essential prerequisites:

1. Node.js and npm (Node Package Manager)

Node.js enables JavaScript to run on the server side, and npm helps manage packages and dependencies.

Download Node.js: https://nodejs.org/en/download/

Installation Instructions: https://nodejs.org/en/download/package-manager/

2. MongoDB

MongoDB is a NoSQL database used to store grocery product details, user accounts, and orders. It can be installed locally or used via cloud services.

Download MongoDB: https://www.mongodb.com/try/download/community

Installation Instructions: https://docs.mongodb.com/manual/installation/

3. Express.js

Express is a lightweight web framework for Node.js used for creating RESTful APIs and managing routes.

Install via Terminal:

npm install express

4. Angular & Angular CLI

Angular is used to create the front-end user interface of your web app.

Install Angular CLI Globally:

npm install -g @angular/cli

Verify Installation:

ng version

Create a New Angular Project:

ng new client

Navigate to Project Directory:

cd client

Start the Development Server:

ng serve

or

npm start

View in Browser:

Visit http://localhost:4200 to see your app running.

Once setup is complete, you can start building your app in the src directory.

For more: https://angular.io

5. Basic Web Development Knowledge

Make sure you're comfortable with:

HTML – for structuring web pages

CSS – for styling and layout

JavaScript – for interactivity and dynamic behavior on the client side

6. Database Connectivity (Node.js + MongoDB)

Use Mongoose (an ODM for MongoDB) to connect and perform CRUD operations from Node.js.

Learn more here:

https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb/

7. Front-End Development with Angular

Angular helps you create a dynamic and responsive interface for:

Product listings

Shopping cart

Checkout and orders

Admin dashboards for inventory and analytics

8. Version Control with Git

Track code changes, collaborate with others, and manage your development workflow using Git.

Git Download: https://git-scm.com/downloads

Use GitHub or Bitbucket to host your repository.

9. Development Environment / Code Editor

Use any modern code editor or IDE. Recommended options:

Visual Studio Code: https://code.visualstudio.com/download

Sublime Text: https://www.sublimetext.com/download

WebStorm: https://www.jetbrains.com/webstorm/download

Once all the prerequisites are installed and configured, you’re ready to start building your full-stack Grocery web application by integrating the back end (Node.js + Express + MongoDB) with the front end (Angular).

**Git Repository Cloning-**

To run the existing grocery-webapp project downloaded from GitHub, follow the step-by-step instructions below:

1. Clone the Repository

Open your terminal or command prompt.

Navigate to the directory where you want to store the grocery web app.

Run the following command to clone the project repository:

git clone https://github.com/Bharath136/grocery-webapp

2. Install Dependencies

Navigate into the cloned project directory:

cd grocery-webapp

Install all the necessary dependencies listed in the package.json file:

npm install

3. Start the Development Server

To start the server, run either of the following commands:

npm run dev

or

npm run start

By default, the app will run at:

http://localhost:5100

You can modify the default port by updating the .env file in the project root.

4. Access the Application

Open your web browser and go to:

http://localhost:5100

You should now see the homepage of the grocery web app, indicating that the setup was successful.

Video Tutorial

For a step-by-step video walkthrough of the setup process, watch this tutorial:

https://drive.google.com/file/d/1KTGK0XZj0XWOiDeNKJVRKQHXLyVWZYLM/view?usp=sharing

Project Repository

You can explore the full project code here:

https://github.com/Bharath136/grocery-webapp

**Role Based Access**

Roles of Admin and User can be defined for an online grocery web application.

(diagram)

**Admin Role**

The admin role grants complete authority and system-wide administrative access within the application.

Key Responsibilities and Permissions:

Shop and Product Management:

Admins can add, modify, and remove store-related data, including product listings.

Booking Management:

Admins can oversee all bookings made by users or agents. This includes actions like viewing, editing, approving, cancelling, or otherwise managing bookings.

User Management:

Admins have the ability to create, update, and delete user accounts. They can also assign or modify user roles and permissions.

Report Generation:

Admins are allowed to access and generate reports and analytics based on bookings, product details, order counts, and overall sales performance.

User Role

**User Role**

Users are the customers of the grocery web application. They can browse through products, search for items of interest, and place bookings for selected products.

Responsibilities and Permissions:

Search Products:

Users can explore the product catalog and search for items based on categories or preferences.

View Product Details:

Users are allowed to view product descriptions, pricing, availability, and complete the purchase process.

Manage Bookings:

Users can book products, view their order history, and have the option to update booking information, change shipping details, or even cancel orders.

Manage Cart:

Users can access their cart, view its contents, and make changes before proceeding to checkout.

**Project Flow**

* Frontend Development
* Backend Development
* Integration

**Frontend Development**

Frontend development involves building the user interface (UI) and implementing the visual elements of the online shopping web application. It focuses on creating an intuitive and engaging user experience that allows users to interact with the application seamlessly.

To Setup the frontend development and to connect node.js with MongoDB Database we have gone through this video link: -

<https://drive.google.com/file/d/1b5bMvnqmASXLnSZ74B2t3EzNjuWHj63g/view?usp=drive_link>

**User Interface (UI) Design**

Develop a visually appealing and consistent interface by applying modern UI design principles.

Use tools like Adobe XD, Sketch, Figma, or InVision to create wireframes, layouts, and interactive mockups.

Focus on typography, color schemes, spacing, and visual hierarchy to enhance usability.

Apply responsive design practices to ensure the interface works well across all screen sizes and devices.

**Product Catalog**

Build and design a product listing page showcasing product images, names, descriptions, prices, and other essential details.

Add a search feature to help users quickly locate specific products.

Incorporate filtering and sorting functionalities to improve the overall product browsing experience.

**Shopping Cart and Checkout Process**

Develop a shopping cart module that enables users to add items, view their cart, update product quantities, and remove products as needed.

Implement a step-by-step checkout workflow, covering stages like entering shipping details, choosing a payment method, and reviewing the final order before placing it.

**User Authentication and Account Management**

Develop a secure registration and login system to manage user access.

Create user profile pages that allow individuals to view and update their personal details, saved addresses, payment options, and past order history.

Implement authentication guards to control access to restricted pages and features, ensuring only authorized users can access them.

**Payment Integration**

Connect the application with a reliable payment gateway such as Stripe, PayPal, or Braintree for processing payments.

Design a secure and user-friendly payment process that allows customers to enter their payment details and complete purchases smoothly.

Manage both successful and failed transactions, and display clear, helpful feedback messages to guide users accordingly.

**Backend Development**

Backend development involves building the server-side components and logic of the online shopping web application. It focuses on handling the business logic, processing requests from the front end, and interacting with the database. The following activities are part of the backend development process:

Setup Backend-

1. Set Up Project Structure

Create a new backend folder and initialize it using npm init.

Install essential packages such as Express, Mongoose, etc.

2. Database Configuration

Configure a MongoDB database either locally or using MongoDB Atlas.

Design collections for key entities like users, products, orders, etc.

3. Create Express.js Server

Build an Express.js server to handle API requests.

Use middleware like body-parser and cors for request processing.

4. Define API Routes

Separate API logic into different route files (e.g., /users, /products, etc.).

Define endpoints for CRUD operations and connect them with controllers.

5. Implement Data Models

Create Mongoose schemas and models for users, products, bookings, etc.

Perform CRUD operations through models and apply schema validations.

API Design and Development-

Design RESTful APIs for major modules like user, product, cart, booking, etc.

Ensure APIs follow standard HTTP status codes and response formats.

Secure APIs with proper validations and authentication checks.

User Management and Authentication-

Implement user registration and login functionality.

Use JWT or session-based authentication strategies.

Encrypt user passwords and secure all protected routes.

Product Catalog and Inventory Management-

Design APIs to add, update, fetch, and delete products.

Handle inventory tracking and availability updates dynamically.

Shopping Cart and Order Management-

Enable users to manage their cart: add/remove items, update quantities.

Allow order placement, cancellation, and status tracking.

Payment Gateway Integration-

Use services like Razorpay, Stripe, or others for secure payments.

Integrate payment verification and order confirmation logic.

Shipping and Logistics Integration-

Integrate logistics APIs for shipping, delivery tracking, and notifications.

Store shipping details with each order and update as needed.

Database Integration-

Choose and connect to a suitable database like MongoDB.

Use models and queries to manage application data efficiently.

External Service Integration-

Integrate external tools like email services, SMS alerts, and analytics.

Example: Send confirmation emails after order placement using Nodemailer.

Security and Data Protection-

Use best practices like input validation, HTTPS, and token verification.

Protect user data using encryption and role-based access control.

Error Handling and Logging-

Implement centralized error handling to catch unexpected issues.

Use middleware in Express to handle and return meaningful error messages.

Integrate logging tools like Winston or Morgan to log API requests, errors, and system activities for easier debugging and maintenance.

**Integration**

Integration is the process of combining and connecting the frontend and backend components of the online flower shop web application to create a unified and fully functional system. It involves establishing communication channels, exchanging data, and ensuring seamless interaction between the frontend UI and backend APIs. The following activities are part of the integration process: