NAAN MUDHALVAN PROJECT

MERN stack powered by MongoDB

GROCERY WEB APP

Team Members,

Kirthiga.R

Mahalakshmi.A

Mahalakshmi.V

Jasmine presilla.J

GROCERY WEB APP

ABSTRACT

The **Grocery Web App** is an online shopping platform designed to provide a seamless and convenient experience for customers seeking a wide range of products. From everyday essentials to specialty items, the app caters to diverse user needs with an intuitive, user- friendly interface that simplifies browsing, product selection, cart management, and secure checkout.

For end- users, the app features a streamlined shopping experience with options to explore product categories, view detailed descriptions, manage their cart, and track orders. Customers can also modify and cancel their bookings, ensuring flexibility and control over their purchases.

The backend of the app supports robust functionality for sellers and administrators. Sellers can efficiently manage inventory, product listings, and orders, while administrators handle customer support, payment processing, and overall app performance monitoring.

Security and privacy are prioritized to protect customer data and ensure secure transactions, fostering trust and reliability among users.

Built with a client- server architecture, the app utilizes a frontend for user interaction and a backend for data management, business logic, and secure integration with payment gateways. APIs enable smooth data flow between these components, while an Entity- Relationship (ER) diagram provides a structural overview, illustrating connections between entities such as users, products, and orders.

The app incorporates role- based access, distinguishing permissions between users and administrators to maintain security and efficiency.

The Grocery Web App aims to deliver a delightful, secure, and personalized online shopping experience for customers while supporting efficient backend operations for sellers and administrators.

1. PROJECT OVERVIEW

Objective:

Create a user- centric, efficient grocery web app that simplifies shopping, enhances user experience, and provides data- driven insights for better decisionmaking.

Scope:

● User account management, including wishlist and order history.

● Product catalog with search, filter, and sort features.

● Real- time inventory tracking and personalized recommendations.

● Seamless payment integration and multiple delivery options.

Goals:

● Enhance customer convenience and engagement.

● Improve the efficiency of the shopping and checkout process.

● Build loyalty through personalized offers and order tracking.

2. FUNCTIONAL REQUIREMENTS

1. User Registration and Login: Allow users to create accounts, log in, and manage profiles.

2. Product Search and Filter: Users can search for products by name, category, and apply filters like price or availability.

3. Shopping Cart and Checkout: Users can add items to a cart, view a summary, and proceed with payment.

4. Payment Gateway Integration: Secure payment options (e.g., credit/debit, PayPal, mobile wallets).

5. Order Tracking: Users can view the status of their orders.

6. Inventory Management: Admins can manage stock, update availability, and add new products.

7. Discounts and Promotions: Admins can apply discounts and offer promotional codes.

8. Notifications: Users receive notifications on order updates, discounts, and more.

9. Customer Support: Chat or contact options for customer queries.

Non- Functional Requirements

1. Scalability: The app should handle growing numbers of users and orders.

2. Performance: Pages should load quickly, and transactions should process in real

time.

3. Security: Secure user data and transactions (e.g., using SSL, encryption).

4. Reliability: High availability with minimal downtime.

5. Usability: Easy- to- navigate interface for both mobile and desktop users.

6. Maintainability: Well- structured codebase for easier updates and debugging.

Hardware Requirements

1. Server Hardware: Sufficient storage, CPU, and memory for the backend server, hosting databases, and API servers.

2. User Devices: Desktop or mobile device for end- users to access the web application.

Software Requirements

1. Frontend: HTML, CSS, JavaScript, frameworks like React or Angular.

2. Backend: Node.js, Django, or similar for handling server- side logic.

3. Database: MySQL, PostgreSQL , or MongoDB for storing user and product data.

4. Cloud Hosting: AWS, Azure, or similar for deployment.

5. Security Software: SSL certificates and firewalls for security measures.

3. ARCHITECTURE DESIGN

1. Frontend (Web & Mobile)

- Built with frameworks like React, Angular, or Vue for web, and React Native or Flutter for mobile.
- Handles user interactions, product browsing, search, cart management, and checkout.

2. Backend (API & Logic Layer)

- REST or GraphQL API built with Node.js, Django, or similar.
- Manages user authentication, product data, order processing, and business logic.
- Communicates with the database and external services.

3. Database (SQL or NoSQL)

- Stores product listings, user data, order details, and inventory status.
- Popular choices include PostgreSQL, MySQL, or MongoDB.

4. Third- Party Integrations

- Payment Gateway for secure transactions (e.g., Stripe or PayPal).
- Geolocation Services for delivery estimation and store locations.
- Notification Services for SMS, email, or push notifications.

5. Admin Dashboard

- Interface for managing products, monitoring orders, and handling inventory.

Interactions:

● The frontend communicates with the backend API for all data and action

requests.

● The backend interacts with the database and third- party services to process

orders and payments.

● The admin dashboard interfaces with the backend for product and order management.

4. UI DESIGN

1.Wireframes / Mockups: Key screens include a clean homepage with product categories, a search bar, and promotions. The product page highlights item details, price, and "Add to Cart" options. The cart screen has a clear item list, total cost, and checkout button for quick navigation.

2. User Experience (UX) Flow: Users begin on the homepage, either searching for items or browsing categories. Adding items to the cart is seamless, with an intuitive cart icon for access. Checkout is quick, guiding users through address input,payment, and order confirmation.

3. Responsive Design: The layout is optimized for both mobile and desktop, with collapsible menus, responsive grids, and touch-friendly buttons on smaller screens, ensuring ease of use on any device.

5. DATABASE DESIGN

1. ER Diagrams:

- Entities: Users, Products, Orders, Cart, Categories, Payments.
- Relationships:
    - Users - Orders: One- to- Many (A user can have multiple orders)
    - Orders - Products: Many- to- Many (An order can include multiple products)
    - Products - Categories: Many- to- One (Each product belongs to one category)
- Key Attributes:
    - oUsers: UserID (PK), Name, Email, Address
    - oProducts: ProductID (PK), Name, Price, Stock
    - oOrders: OrderID (PK), UserID (FK), OrderDate, Status
    - oCart: CartID (PK), UserID (FK)
    - oCategories: CategoryID (PK), Name

2. Data Models:

- Schema for each table defining:
    - Primary and Foreign Keys: Ensuring relationships between tables.
    - Constraints: E.g., Not Null, Unique for Email, Positive for Stock and Price.

3. Data Flow Diagrams (DFDs) (optional for complexity):

● Level 1: User actions like Browse Products, Add to Cart, Place Order.

● Level 2: Movement of data from Order to Payment to confirm purchase and update inventory.

6. API DOCUMENTATION (for Backend)

1. API Overview

This API enables interaction with the backend of the Grocery Web App. It provides endpoints for user management, product catalogs, shopping carts, and order processing. All API requests require authentication via API tokens.

● Base URL: **https://api.groceryapp.com**

● Authentication: Bearer Token required for all endpoints (except for public ones like product listings).

2. Endpoint Documentation

● GET /products

- Description: Fetch a list of available grocery products.
- Parameters:

● category (optional): Filter by product category.

● search (optional): Search by product name.

oResponse:

● 200 OK: Returns list of products.

● Example Response: [{ "id": 1, "name": "Apple", "price": 1.2,"category":"Fruits" }]

● POST /cart

- Description: Add a product to the user's shopping cart.
- Request Body:

● productId: ID of the product.

● quantity: Quantity to add.

- Response:

● 201 Created: Returns cart details with the newly added product.

● Example Response: { "cartId": 123, "items": [{ "productId": 1, "quantity": 2 }] }

● PUT /cart/{cartId}

- Description: Update the quantity of an item in the cart.
- Parameters:

● cartId: Unique identifier of the cart.

- Request Body:

● productId: ID of the product.

● quantity: Updated quantity.

- Response:

● 200 OK: Returns updated cart.

● Example Response: { "cartId": 123, "items": [{ "productId": 1, "quantity": 3 }] }

● POST /checkout

- Description: Process a user's order.
- Request Body:

● cartId: Cart ID to checkout.

● paymentMethod: Chosen payment method.

- Response:

● 200 OK: Order successfully processed.

● Example Response: { "orderId": 456, "status": "Success" }

3. Error Codes and Responses

● 400 Bad Request: The request was invalid or missing required fields.

- Example: { "error": "Missing productId" }

● 401 Unauthorized: Invalid or expired authentication token.

- Example: { "error": "Authentication required" }

● 404 Not Found: The resource (e.g., product, cart) was not found.

- Example: { "error": "Product not found" }

● 500 Internal Server Error: Unexpected server issue.

- Example: { "error": "Something went wrong. Please try again later." }

7. SECURITY MEASURES

● Authentication & Authorization: Users are authenticated using JWT (JSON Web Tokens) for secure login. Role- based access control (RBAC) ensures different levels of authorization for admins, customers, and vendors.

● Input Validation & Sanitization: Input fields are validated and sanitized to prevent SQL injection, XSS, and CSRF attacks. Prepared statements and secure coding practices are used to ensure safe interactions with the database.

8. TESTING

● Unit Testing: Ensure individual components (like cart functionality, product filtering) work as expected.

● Integration Testing: Verify data flows between modules, such as the checkout process and payment gateway.

● UI/UX Testing: Ensure user interface is intuitive, responsive, and meets design standards.

● End- to- End Testing: Simulate user interactions (browsing, adding to cart, checkout) to ensure the entire system functions as expected.

Test Cases

1. Product Search

- Input: Product name (e.g., "apple")
- Output: List of products matching the search query
- Criteria: Success if results match the query, failure if no results or incorrect results appear.

2. Add to Cart

- Input: Product selection
- Output: Product added to cart
- Criteria: Success if cart updates with the correct product and quantity.

3. Checkout Process

- Input: Payment details
- Output: Order confirmation and payment receipt
- Criteria: Success if payment is processed and order is confirmed.
- Bug Tracking

● Tools: Use tools like Jira, Trello, or GitHub Issues for bug tracking.

● Workflow:

1. Report bugs with detailed steps to reproduce.

2. Assign priority and severity.

3. Track progress and mark bugs as resolved after fixes are deployed.

9. DEPLOYMENT AND SETUP GUIDE

1. Installation Instructions

● Clone the repository:

git clone <repository_ url>

● Navigate to the project directory:

cd grocery- webapp

● Install dependencies:

npm install (for Node.js apps) or pip install - r requirements.txt (for Python apps).

● Set up the database:

Run migration commands:

npm run migrate or python manage.py migrate

● Configure environment variables as described below.

2. Deployment Process

● Production Setup:

Set up a production- ready environment with necessary services

● CI/CD Pipeline:

Use GitHub Actions, Jenkins, or GitLab CI for automatic testing, building, and

deployment.

10. USER GUIDE

User Onboarding

1. Sign Up/Login: Create an account using your email or sign in with your social media accounts.

2. Set Preferences: Select your preferred shopping categories (e.g., fruits, vegetables, dairy) and delivery preferences (home delivery or store pickup).

3. Browse the Store: Explore a wide range of grocery items through categories or search bar.

4. Add Items to Cart: Select items and add them to your shopping cart.

5. Checkout & Payment: Review your cart, select payment methods, and complete your purchase.

Features and Functions

1. Product Search: Use the search bar to find specific products quickly.

2. Filters: Narrow down results by price, category, and dietary preferences (vegan, gluten- free).

3. Order Tracking: View your order status, estimated delivery time, and realtime tracking.

4. Reorder: Quickly reorder your past purchases for convenience.

5. Discounts & Offers: Check for current discounts or promotional offers on the homepage.

6. Multiple Payment Options: Pay using credit/debit cards, e-wallets, or cash on delivery.

Screenshots / Visual Aids

1. Homepage: A visual of the homepage showcasing categories and featured deals.

2. Shopping Cart: A screenshot of the cart, highlighting the checkout button.

3. Order Tracking: Visual of the order status page with tracking details.

11. MAINTENANCE AND FUTURE ENHANCEMENTS

Maintenance Plan: Regular updates will be scheduled monthly to ensure optimal performance, security patches, and bug fixes. Downtime will be minimized and communicated in advance.

Future Scope/Roadmap: Plans include adding AI- powered personalized recommendations, expanding payment options, integrating loyalty programs, and enhancing the mobile app experience.

12. GLOSSARY / APPENDIX

- API (Application Programming Interface): A set of protocols that allow different software applications to communicate.
- CRUD (Create, Read, Update, Delete) : Basic operations performed on data in a database.
- UI (User Interface): The visual elements of a web or app interface that users interact with.
- UX (User Experience): The overall experience of a user when interacting with a product.
- Database: A structured collection of data stored electronically.