ProjectTitle:Grocery WebApp

**Team members:**

**Siva jeyanthi.J–210821205104**

**Serin.E–210821205100**

**Gowri.D – 210821205030**

**Gayathri.V – 210821205028**

# Purpose:

Agrocery web app provides convenience, time savings, and personalized experiences for shoppers by allowing them to browse, order, and schedule delivery or pickup of groceries online. It enhances user experience through features like personalized recommendations, shopping lists, and customizable orders. The app helps users save money with price comparisons, discounts, and subscription services. It also promotes healthier choices with dietary filters and nutritional info. For businesses, it improves efficiency with inventory and order management and offers customer insights.

Sustainabilityfeatures,accessibility,andloyaltyprogramsfurtherenhanceitsvalueforbothusersandretailers.

# Features:

Agrocery web app offers a range of features to enhance the shopping experience, including a comprehensive product catalog with search and personalized recommendations, secure payment options, and flexible delivery or pickup choices. Users can customize orders, apply discounts, and set up recurring subscriptions for frequently bought items.Additional features like detailed product information,nutritionfilters,customerreviews,andordertrackinghelpusersmakeinformeddecisions. The app also supports multi-device access, offers customer support, and sends push notifications for deals and order updates.

# Architecture:

Frontend:

The frontend architecture of a grocery web app using React is component-based, where UI elements like product cards, shopping carts, and search bars are built as reusable components. State is managed using React's `useState` or `useReducer` for local state and ContextAPI or Redux for global state (e.g., cart and user data). \*\*React Router\*\* handles page navigation, whileAPI calls (usingAxios or Fetch) fetch dynamic data like product listings and user info.

For styling, \*\*Material-UI\*\* or \*\*Chakra UI\*\* can be used for consistent, accessible components, and \*\*CSS-in-JS\*\* or \*\*CSS Modules\*\* for scoped styling. The app is responsive, optimized with techniques like \*\*code splitting\*\* and \*\*lazy loading\*\* for better performance, especially on mobile. \*\*JWT\*\* handles user authentication, ensuring secure access to protected routes (like checkout). Finally, \*\*unit tests\*\* with \*\*Jest\*\* and \*\*React Testing Library\*\* ensure reliability.

Thisarchitectureisscalable,efficient,andprovidesasmoothuserexperienceacrossdevices.

Backend:

Thebackendarchitectureforagrocerywebappusing\*\*Node.js\*\*and\*\*Express.js\*\*involvessettingupaRESTfulAPItohandleproductlistings,userauthentication, cartmanagement,andorderprocessing.\*\*Express.js\*\*routesAPIrequests,whilea\*\*NoSQLdatabase\*\*(e.g.,MongoDB)storesdatalikeproductsanduserprofiles.

\*\*JWT\*\*handlessecureauthentication,andmiddlewareensuresinputvalidation,errorhandling,androuteprotection.

KeyAPIendpointsincludeproductCRUDoperations,usermanagement,cartactions,andorderprocessing.Paymentintegrationwith\*\*Stripe\*\*or\*\*PayPal\*\*supports secure transactions.The backend is scalable, with options for \*\*Docker\*\* containers and cloud deployment (e.g.,AWS, Heroku).This architecture ensures a secure, efficient, and scalable backend for the web app.

# Database:

Foragrocerywebappusing**MongoDB**,thedatabaseschemaincludesseveralkeycollections:

1. **Product**(products):Storesproductdetailslikename,price,description,stock,andimages.CRUDoperations allow adding, updating, and retrieving products.
2. **User**(users):Storesuserinformation(name,email,password,address)andorderhistory.Supportsuser registration, login, and profile management.
3. **Cart**(carts):Trackseachuser'sshoppingcartwithproductIDs,quantities,andtotalprice.Supports adding/removing items and fetching the current cart.
4. **Order**(orders):Storesorderdetailssuchasuser,items,shippingaddress,andpaymentstatus.Managesorder creation and status updates (e.g., pending, shipped, delivered).
5. **Review**(reviews,optional):Allowsuserstoleaveproductreviewswithratingsandcomments.

# SetupInstructions:

###\*\*GroceryWebAppSetupOverview(Angular+MongoDB)\*\*

The \*\*Grocery WebApp\*\* is a simple e-commerce platform that lets users browse products, add them to their cart, and place orders.Theappuses\*\*Angular\*\*forthefrontend(userinterface)and\*\*MongoDB\*\*forstoringdata(products,useraccounts, orders).

###\*\*1.WhatYouNeedtoGetStarted\*\*

Toruntheapp,makesureyouhavethesetoolsinstalled:

\*\*Node.js\*\*:Requiredtorunthebackendandmanagedependencies.

* \*\*AngularCLI\*\*:AtooltohelpyoucreateandrunAngularprojects.
* \*\*MongoDB\*\*:Adatabasetostoreproductanduserdata.Youcanuse\*\*MongoDBAtlas\*\*(cloud-based)orinstall

\*\*MongoDB\*\*locally.

\*\*Git\*\*:Aversioncontroltoolformanagingthecode. ### \*\*2. How the App Works\*\*

* \*\*Frontend(Angular)\*\*:Thisistheparttheuserinteractswith.Itdisplaysproducts,handlestheshoppingcart,andletsusers place orders. It communicates with the backend to get product data and send order details.

-\*\*MongoDB(Database)\*\*:Storesallthedatafortheapp,suchasproducts,users,andorders.MongoDBisflexibleand easy to work with.

###\*\*3.SettingUptheApp\*\*

1. \*\*ClonetheProject\*\*:Downloadtheapp’scodefromaGitrepository.
2. \*\*InstallDependencies\*\*:
   * Inthe\*\*frontend\*\*folder,run`npminstall`toinstallthenecessarylibraries(Angularcomponents,etc.).
3. \*\*RuntheBackend\*\*:
   * Setupabackendservertohandlerequests(forexample,using\*\*Node.js\*\*and\*\*Express\*\*)andconnectto

\*\*MongoDB\*\*.

1. \*\*RuntheFrontend\*\*:
   * UseAngular’scommand`ngserve`torunthefrontend,whichwillshowtheuserinterfaceinyourwebbrowser. ### \*\*4. Key Features\*\*

* \*\*Frontend(Angular)\*\*:Displaysproducts,shoppingcart,andorderform.Handlesuserinteractionslikeaddingproducts to the cart and placing orders.
* \*\*Database(MongoDB)\*\*:Storesproductdata(e.g.,productnames,prices)anduserdata(e.g.,usernames,orders). ### \*\*Conclusion\*\*

Thisappuses\*\*Angular\*\*forthefrontend,\*\*MongoDB\*\*forthedatabase,andcommunicatesbetweenthetwotoletusers shop online. Once set up, you can browse products, add them to the cart, and place orders.

# Folder Structure:

### Frontend(React)Structure:

* + **/public**:Containstheindex.htmlandstaticassets.
  + **/src**:Allfrontendcode:
    - **/components**:ReusableUIelements(e.g.,ProductCard,Navbar).
    - **/pages**:Apppages(e.g.,Home,Cart,Checkout).
    - **/services**:APIcallstothebackend(e.g.,fetchProducts).
    - **/context**:Globalstatemanagement(e.g.,CartContext).
    - **/styles**:CSSorstyled-componentsforUI styling.

### Backend(Node.js)Structure:

* + **/config**:Configurationfiles(e.g.,databasesetup).
  + **/controllers**:Businesslogic(e.g.,handlingproduct,orderrequests).
  + **/models**:MongoosemodelsforMongoDBschemas(e.g.,Product,Order).
  + **/routes**:APIendpointsforhandlingrequests(e.g.,productRoutes).
  + **/middleware**:Middlewarefunctions(e.g.,authenticationchecks).
  + **/utils**:Helperfunctions(e.g.,JWTtokengeneration).

Thisstructurekeepstheprojectmodularandorganized,separatingconcernsbetweenthefrontend,backend,and database.

# RunningtheApplication:

### RunningtheGroceryWebAppLocally

1. **Backend:**

Navigatetothe**server**directory:

bashCopycode

cdserver

* + Install dependencies: bash

Copycode

npminstall

* + Start the backend: bash

Copycode

npmstart

* + Backendrunsonhttp://localhost:5000(orcustomport).

1. **Frontend:**

Navigate to the **client** directory: bash

Copycode

cdclient

* + Install dependencies: bash

Copycode

npminstall

* + Startthefrontend: bash

Copycode

npmstart

* + - Frontendrunsonhttp://localhost:3000.

1. **Accessthe App:**

Open http://localhost:3000in your browser to view the app. This will run both the **React frontend** and **Node.js backend** locally

# API Documentation:

1. **Products:**
   * **GET/api/products**:Retrieveallproducts.

○

* + - **Response**:Listofproducts(name,price,description).
  + **GET/api/products/**

:RetrieveproductdetailsbyID.

* + - **Response**:Productdetails(name,price,description).
  + **POST/api/products**:Addanewproduct.
    - **Request**:Productdetails(name,price,description).
    - **Response**:Successmessagewithnewproduct.

1. **Orders:**
   * **POST/api/orders**:Placeanorder.
     + **Request**:UserID,items(productID,quantity),totalprice.
     + **Response**:Orderconfirmation.
   * **GET/api/orders/**

:RetrieveordersbyuserID.

* + - **Response**:Listofuserorders.

1. **User Authentication:**
   * **POST/api/auth/register**:Registeranewuser.
     + **Request**:Username,email,password.
     + **Response**:Successmessagewithuserdetails.
   * **POST/api/auth/login**:Loginanexistinguser.
     + **Request**:Email,password.
     + **Response**:JWTtoken.

# Authentication:

## Authentication(Login&Registration):

* + - **JWTTokens**areusedtoauthenticateusers.
    - On**registration**,passwordsarehashedandstoredsecurely.
    - On**login**,validcredentialsreturna**JWTtoken**totheuser.

## Authorization:

* + - Protectedroutes(e.g.,placingorders)requireavalidJWTtoken.
    - The**JWTtoken**isincludedinthe**Authorizationheader**forrequests.
    - **Middleware**verifiesthetokenandgrantsaccesstoprotectedresources.

## TokenExpiry:

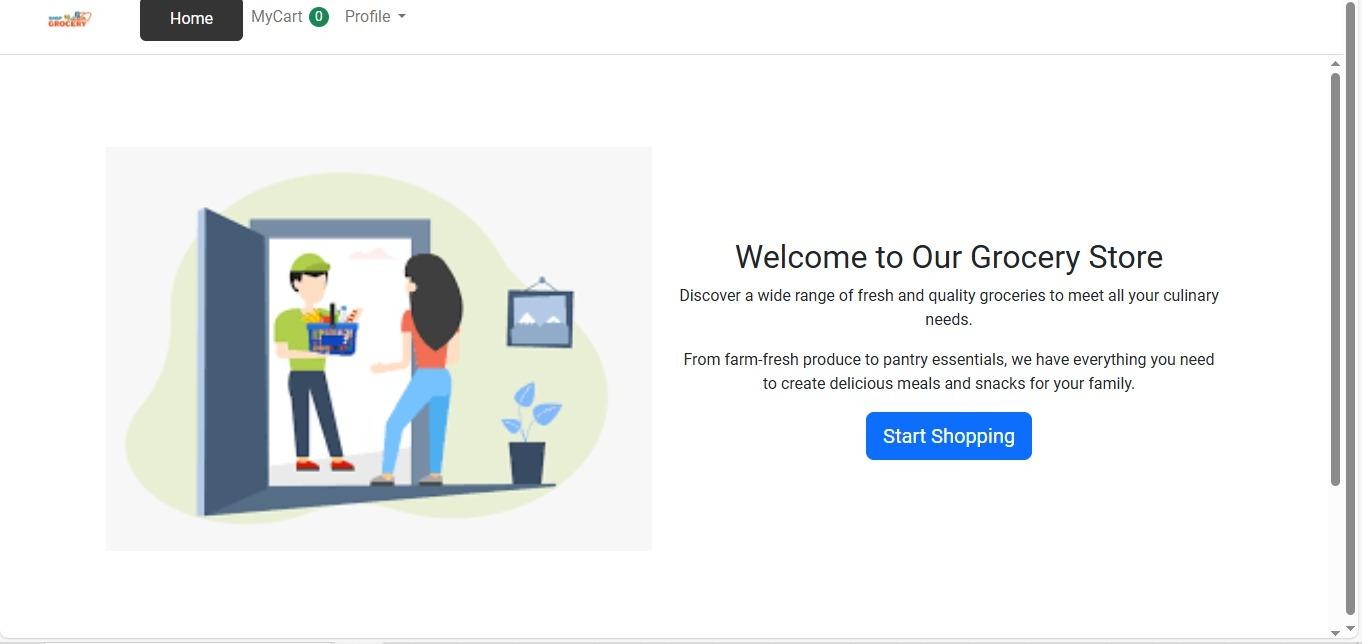
* + - JWTtokenshaveanexpirytime(e.g.,1hour).Usersmustloginagainorusearefreshtoken.

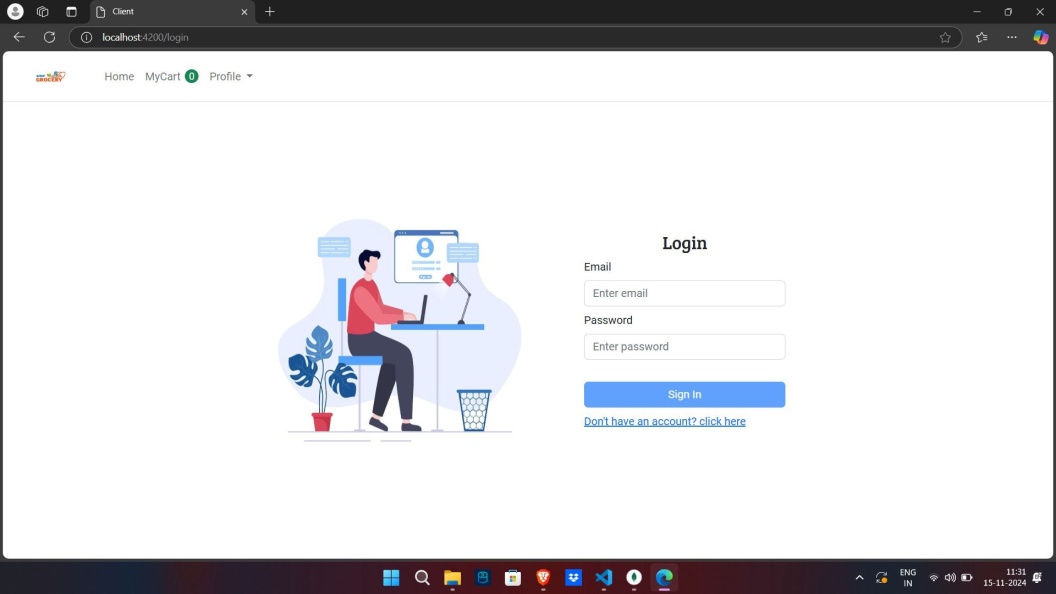
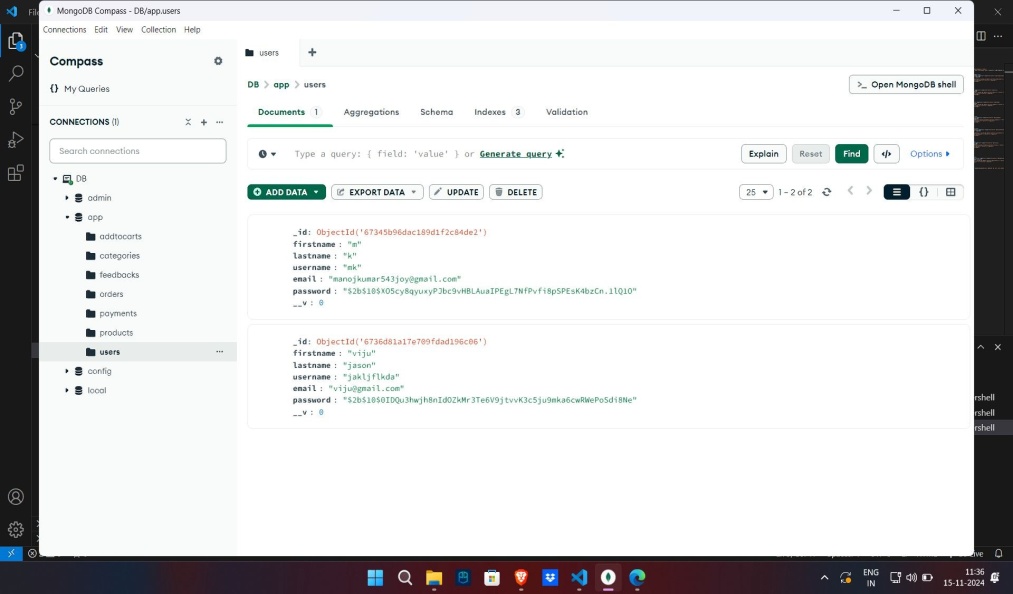
## Security:

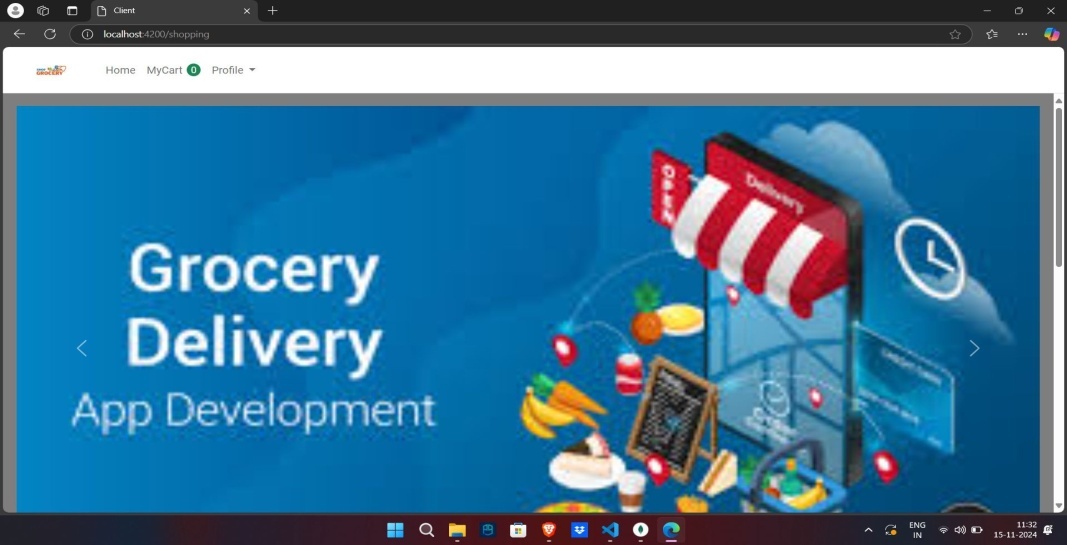
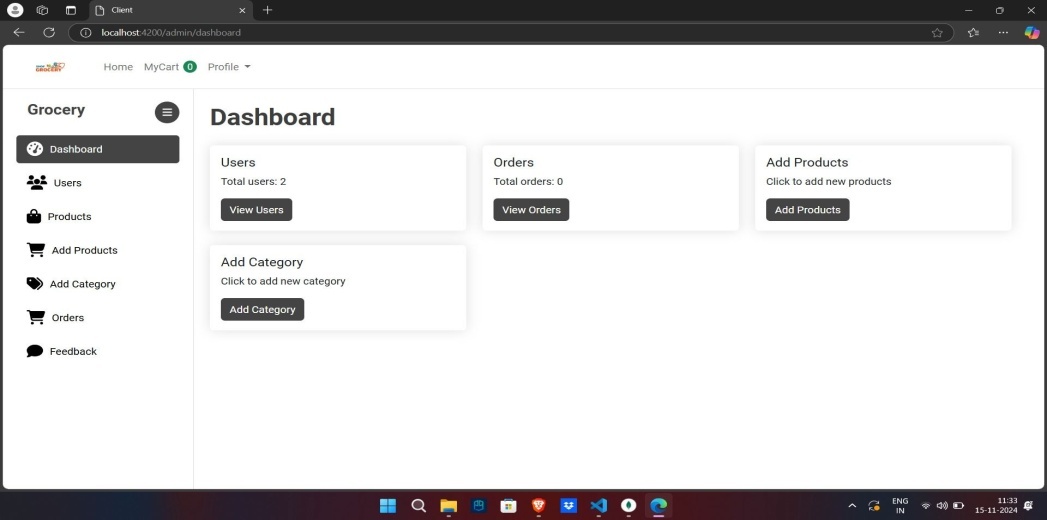
* + - Passwordsarehashedwith**bcrypt**.
    - Communicationisover**HTTPS**forsecurity.
    - **JWTsecret**ensuressecuresigningandverificationoftokens.

Thissystemensuressecureuseraccessandprotectssensitivedataacrosstheapp.

# User Interface:



# Testing:

1. **FrontendTesting(React):**

**Tools**:**Jest**,**ReactTestingLibrary**

* + **Focus**:
    - UnittestsforReactcomponents(e.g.,Cart,ProductList).
    - MockingAPIcallsforisolatedtesting.
    - Componentinteractionandrendering.

1. **BackendTesting(Node.js/Express):**
   * **Tools**:**Jest**,**Supertest**
   * **Focus**:
     + Unittestsforbackendlogicand functions.
     + APIendpointtesting(e.g.,GET/POSTrequests).
     + Mockingdatabaseinteractions.
2. **End-to-End(E2E)Testing:**
   * **Tool**:**Cypress**
   * **Focus**:
     + Fulluserflowtesting(e.g.,login,shopping,checkout).
     + UI/UX,responsiveness,anderrorhandling.
3. **ManualTesting:**
   * **Focus**:
     + Exploratorytesting,cross-browser,andmobiletesting.
4. **CI/CDIntegration:**
   * **Tools**:**GitHub Actions**,**Jenkins**
   * **Focus**:Automatedtestsoneachcodechangetoensurecodequality.

Thisapproachensuresthoroughtestingacrossalllayersoftheapp—frontend,backend,anduserinteractions.

# KnownIssues:

UserInterfaceisnotgood.

FutureEnhancements:

1. **UserAccountManagement**:Profilecustomization,orderhistory,andreorderingoptions.
2. **AdvancedSearch&Filters**:Improvedsearchwithauto-suggestions,filters,andvoice search.
3. **RecommendationSystem**:Personalizedproductsuggestionsbasedonuserbehavior.
4. **Real-timeInventoryUpdates**:Stockalertsandreal-timeproductavailability.
5. **Subscription&RecurringOrders**:Grocerysubscriptionsandscheduledorders.
6. **Multi-language&CurrencySupport**:Localizationandcurrencyoptionsforglobalusers.
7. **MobileApp**:DevelopnativeiOS/Androidappswithofflinecapabilities.
8. **PaymentGatewayIntegration**:Addmorepaymentoptions(e.g.,GooglePay,PayPal, Crypto).
9. **CustomerReviews**:Productratingsandreviewfiltering.
10. **Admin Dashboard**:Advanced analytics, order management, and insights. **11.SustainabilityFeatures**:Eco-friendlyproductsandcarbonfootprinttracking. **12.Delivery Enhancements**: Same-day delivery and real-time tracking.