**Food Delivery App**

npm create vite@latest

Create a frontend folder and after that cd in that folder and npm I

npm i react-router-dom

npm run dev

**Clearing the project dirctory**

Go into App.jsx and clear the code and write rafce

Delete app.css

Give some basic css in index.css

**Create a folder structure for the project**

In src folder create components, pages folder

Create a Navbar component in that write rafce

In this we learned about useState

**Create subfolders in pages folder**

Create Home, Cart, PlcaeOrder and respective jsx files

**After that we are going to setup the react router**

In main.jsx write code for react router i.e. BrowserRouter from react-router-dom package

After that in App.jsx write Route & Routes code for routing in Home, Cart & PlcaeOrder pages

**Now write code for Home page**

Create a folder named Header in components folder and create a file Header.jsx

Write code for Header.jsx and also paste the header\_img in public folder

While writing the css we wrote keyframes in index.css and then use it in Home.css for fadeIn effect

**Now we are creating ExploreMenu Component**

**After that we will create a useState in Home.jsx for categorires and that useState we will use in ExploreMenu.jsx file so that we can apply onClick functions**

**After that we will create FoodDisplay component but before that we will setup ContextAPI**

For that create a context folder in src& create a file named StoreContext.jsx and write required code, To add support of context we will add necessary code in main.jsx file.

We will mount our food\_list from assets.js file into the contextValue

**We will also create a FoodItem component**

Next using useState we created add food item “button” functionality.

**FOR food item counter, when we have multiple food products it will create multiple state variables i.e. for each product, for solving this we will create a cart - item object in our context and we ll manage the data using manage-cart functionality, so the useState is not into working**

**Now we will create food category filter functionality**

For that we will write required code in FoodDisplay.jsx file

**After that we will create a footer component**

We add that in app.jsx

**After that we create a AppDownload component**

**After that we made the complete home page responsive**

**After that we made our navbar working**

This is done by using the Link of react-router-dom and “to” attribute in it

Also we make scroll-bheavior: smooth

**Now we will create a LoginPopup component**

We will start from App.jsx i.e. create useState overthere and then write onclick function in Navbar.jsx

**After that we will create Cart page**

Link the cart icon on Navbar.jsx to the cart page with the help of Link tag

**After that we created the logic of cart subtotals and stuff using contextAPI**

Also use useNavigate to redirect to PROCEED TO CHECKOUT button to /order

**After that create the PlaceOrder page**

* **In Navbar we saw useState**
* **main.jsx BrowserRouter from react-router-dom and then in App.jsx Routes, Route from react-router-dom and we gave the path attribute**
* **ContextAPI, In main.jsx we have to add required configuration**
* **IMP Note on Home.jsx (Filter functionality)**
* **Login popup functionality**
* **Cart functionality using ContextAPI**

**BACKEND**

**npm init**

Entry point will be **server.js**

**npm i express mongoose jsonwebtoken bcrypt cors dotenv body-parser multer stripe validator nodemon**

Add script(in package.json) "server": "nodemon server.js"

**Create folder structure**

In backend folder create, config, controllers, middleware, models, routes, uploads.

Also create .env file in backend folder

Also add “type”:”module” in pcakage.json

Atlas

krishnap, Setup the Atlas and create a file db.js in config folder and then create a async function then call the function in server.js

**After that we will create a file named foodModel in models folder**

Create the foodModel

**After that we will create APIs using which we can add new food items in our database**

For that in controllers folder create foodController.js

Create addFood function in it and using that create a route, for that create a file foodRoute in routes folder and then write /api/food app.use in server.js

**After that write a post request route in foodRoute.js**

For that we wrote multer configuration and then created a middleware and then wrote the route

**Write another endpoint that will access the uploaded image in uploads folder**

**After that we create a get request route and corresponding controller for listFood**

**Next we will write remove food item i.e. removeFood controller for route**

Here we use the fs from fs

**Creating Admin Panel using the apis that we created (using react (vite))**

**npm create vite@latest**

**Create file structure and cleanup**

**npm i axios react-toastify react-router-dom**

**After that create Navbar and Sidebar components and Add, List, Orders pages in pages folder**

**Write Route and Routes of react-router-dom in App.jsx**

**Now in Sidebar use NavLink for routing in react**

**Create add page**

In the Add page we create image upload functionality with image preview using useState and onChange function with **e.target.files[0]** as callback, and preview using **URL.createObjectURL(image)**

Now, We are also going to take other data using useState, for that we will have to create a onChangeHandler function and **onChange={onChangeHandler} value={data.name}**

Now, we will write onSubmitHandler which does the API call using axios and use toastify after initializing it in App.jsx and using the toast.success or toast.error method to show alerts

**Now create List.jsx page & make it working**

Create a url variable in App.jsx and pass it to every page

**NOTICE:**

**While storing the data in the databse:**

**We take the data from form and put it in a useState**

**Then we create an object and put that data in it**

**And then finally we will create a API call using axios**

**And then we will asign the value of useState to default.**

**Create user authentication**

**For that create userController.js and userRoute.js and userModel.js**

**After that create register user API in userController**

**After that create login user API in userController**

**THEN IN FRONTEND FOLDER**

**Create a useState variable(data) for Login/Signup popup fo rm, and onChangeHandler in LoginPopup.jsx**

**After that pass the backend url value in ContextValue in StoreContext.jsx**

**npm i axios for onLogin function in frontend folder**

For that we will also create a token useState variable in context i.e. StoreContext.jsx

**After that in Navbar.jsx if user is loggedin then signup button shouldn’t be display, instad of that a profile icon should be display**

**After that create logout functionality**

For that create a logout function in NavBar.jsx and use useNavigate() from react- router-dom for redirection “/” after logout

**Also create a useEffect in StoreContext.jsx so that the user will be logged in even after we reload the page**

**After that fetch food data on frontend from database**

For that we will create fetchFoodList function in StoreContext.jsx and call loadData in useEffect

**NOW we are going to add cart functionality**

In this we want that even after we reload the page or logout the cart data should be stored so when we login again then the cart will be as it is as we left earlier.

**Here we are going to build addToCart, removeFromCart, getCart function in cartController.js and also we will create cartRoute.js in routes folder**

We will write routes in cartRoute.js and then use that in server.js

**Create a file auth.js in middleware folder**

In cartRoute.js file import that and then add it to every single route

The middleware will take the token n convert it into userId n using that userId we can add, remove data form the cart

**Write logic for add remove and get cart functions respectively**

**Now in the frontend folder**

In the storeContext.jsx, add code in addToCart and removeCart function

Also create a new function named loadCartData and mention it in useEffect

**Now, Create place order feature & payment gateway using stripe**

**In the backend folder, create a file orderModel.js in models folder**

Write the schema for order

**After that create a orderController.js in controllers folder, also create the orderRoute in routes folder**

Write the code(endpoint) in server.js

**Write the STRIPE\_SECRET\_KEY in .env file**

For getting that we ll have to create an account in stripe

Then create the logic for placeorder with stripe in orderController.js

**Now in the frontend folder**

In PlaceOrder.jsx, take data from context, then create a useState for form data

Create onChangeHandler function and add it in useEffect for testing(REMOVE the useEffect after use)

After that we write code for axios for payment and placing the order in database

**Now in the backend folder**

In orderController.js we will write code for verifying the payment (verificcation is done using webhooks but it will be time consuming so we will do it using temporary payment verification system)

Create verifyOrder in orderController.js, then create a route in orderRoute.js

**After that in the frontend folder in pages folder, create a Verify folder and in that create Verify.jsx and respective css file**

After that create verify route in App.jsx

Write code in Verify.jsx for verification

**Create user order page**

**In backend folder**

In orderController.js create a userOrders function and also create a route in orderRoute.js write the code in userOrders function

Now link this api to content

**In frontend folder**

In pages folder create a folder named MyOrders folder and in that respective jsx and css files and add route in App.jsx

In myOrders.jsx create fetchOrders function and write code for fetching the orders

After that do the UI part for displaying orders

After that we write useEffect in PlaceOrder.jsx page so if any item is not present in the cart then it should be redirected to /cart

After that add navigate to /myorders in Navbar.jsx

**Display orders in admin panel**

**First create the api in backend folder**

In orderController create a function named listOrders and create a route in orderRoute.js as well

Write code in function

**Now in admin folder**

In Orders.jsx write code to fetch the data using axios

After that do the UI part

**Create order update feature**

**In the backend folder**

In orderController.js file create a function named updateStatus and create a respective route in ordeRoute.js

Write code for it

**In the admin folder**

In Orders.jsx

After that make the track order button working

**NOW WE WILL RUN ALL SERVERS, MAKE SURE TO UPDATE THE frontend\_url in orderController.js according to your frontend port**

Thank You.