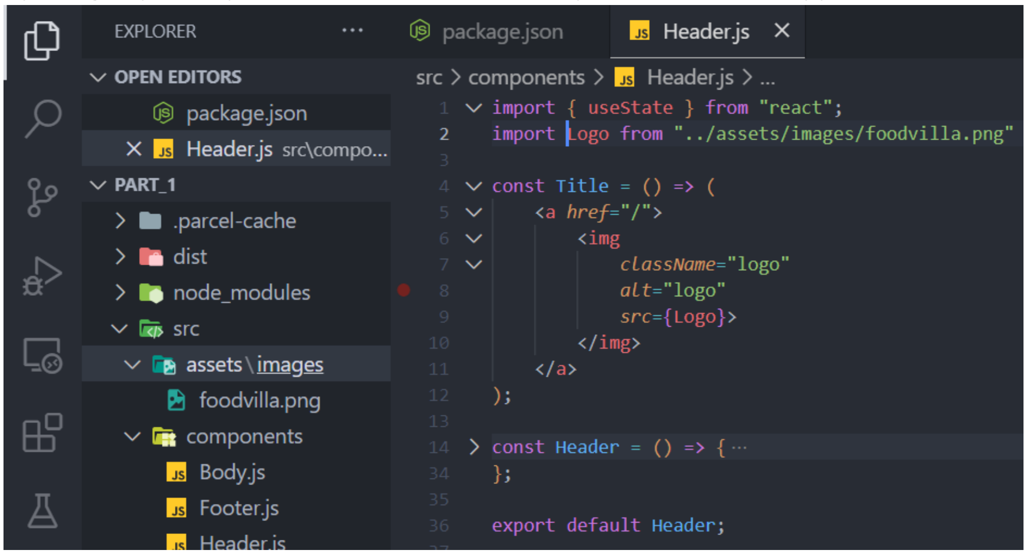
#NOTE :-

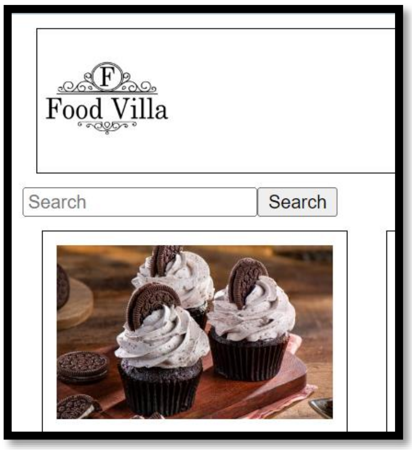
- It is a bad practise to create a component inside another component, because everytime the outer component gets rendered, the inner component gets created.
- It is also a bad practise to declare a state variable inside an if-else block, because then React wouldn't know when that variable should be created. This causes inconsistencies for React.
- Also, never declare a state variable inside a for loop, because it doesn't make sense to initialise a state variable again and again, when we can just change it's value using the set function
- React provides us useState hooks to create local state variable inside our functional component. That's why we should never use useState outside a component.
- We create multiple useEffects according to out requirements.

Using saved images from the local directory :-

We usually create an assets folder inside the src folder and there we create an images folder, where we can store all our images locally. For ex :- I have downloaded a logo named foodvilla.png and placed it inside my images folder. We can call it by using import keyword and use it as a normal piece of JS (i.e. wrapped inside " {} ").



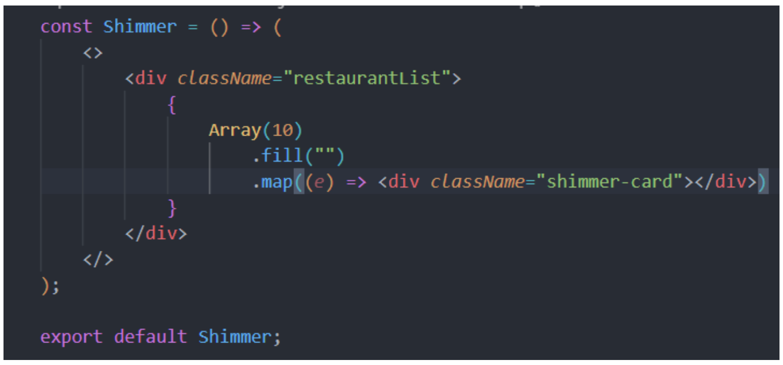
```
import { useState } from "react";
import Logo from "../assets/images/foodvilla.png"

const Title = () => (
  <a href="/">
    <img
      className="logo"
      alt="logo"
      src={Logo}>
    </img>
  </a>
);

const Header = () => {⋯
};

export default Header;
```



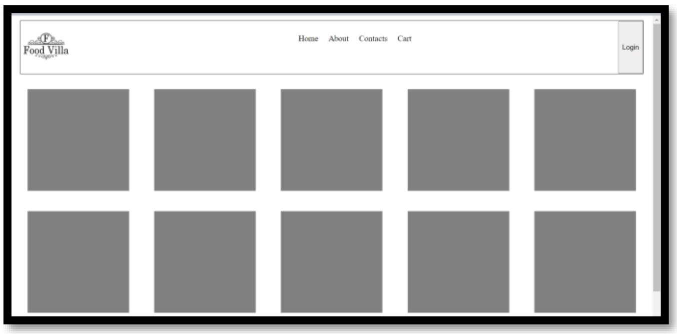
However, apps like Swiggy loads images from CDN (because CDN is faster, it caches our data [images here], it has 100% uptime [read about uptime]. That's why if you right click an image in Swiggy and open it in new tab, you will see a cloudinary CDN URL from which the image is actually fetched.

Implementing Shimmer Effect gracefully :-

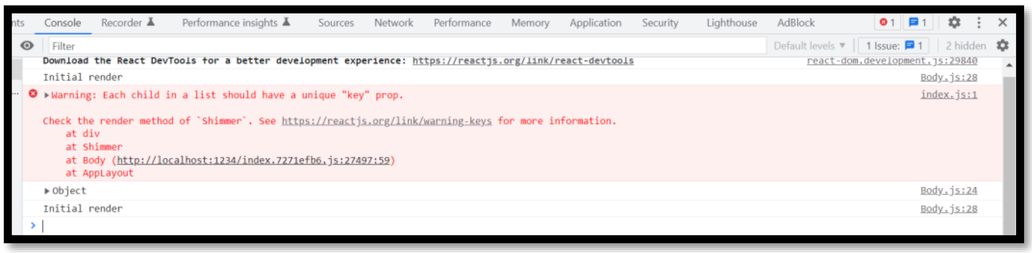
Instead of just showing "Shimmer is loading....", we will create dummy restaurant cards that will act as Shimmer.



```
const Shimmer = () => (
  <>
    <div className="restaurantList">
      {
        Array(10)
          .fill("")
          .map((e) => <div className="shimmer-card"></div>)
      }
    </div>
  </>
);

export default Shimmer;
```



I have also declared some css for the "shimmer-card" class, and the end-result is the left one. Now, you will get an warning of not giving unique keys to each shimmer-card too.



```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools   react-dom.development.js:29840
Initial render                                                                                            Body.js:28
▸Warning: Each child in a list should have a unique "key" prop.                                            index.js:1

Check the render method of `Shimmer`. See https://reactjs.org/link/warning-keys for more information.
    at div
    at Shimmer
    at Body (http://localhost:1234/index.7271efb6.js:27497:59)
    at AppLayout
▸Object                                                                                                   Body.js:24
Initial render                                                                                            Body.js:28
```

To overcome this, we will just use our key as index of each card.

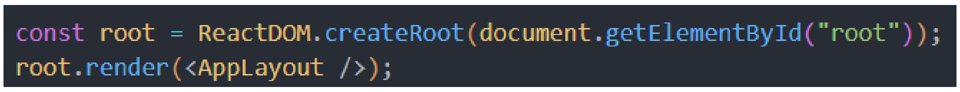
Routing and react-router-dom:-

We will create different pages for different elements like Home, About, Contacts etc. and on clicking those texts on the UI, he should be routed to a different page. For this we have to install a package :- react-router-dom.
Lets 1st create an About Page i.e. an About component.

We can write “rafce” in VSCode to get a boiler plate react component, but always try to avoid it

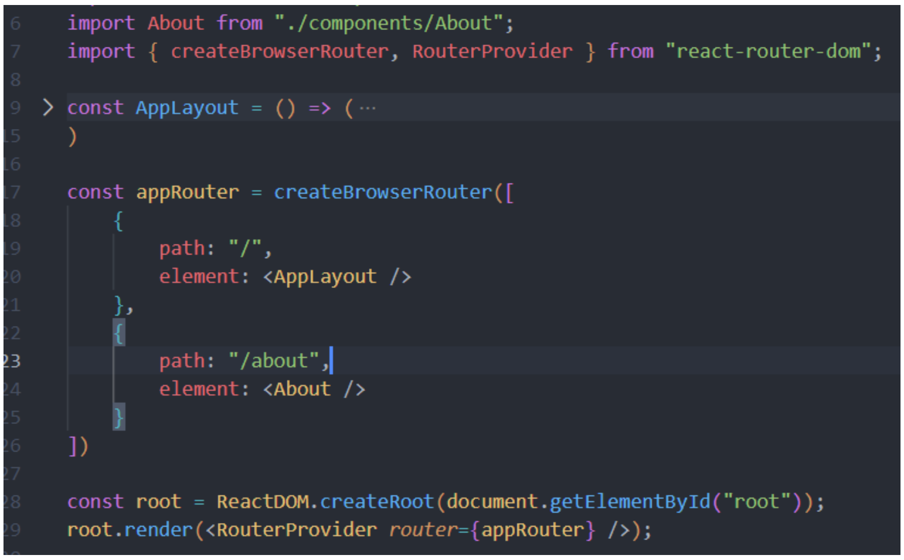
To create a routing, we have to create a configuration for our router. For this we also import **createBrowserRouter** function from the react-router-dom package. Learn more about it from the docs (left side of the homepage in the Routers section :- Link). We have to call this function to create a router (ex :- in our code we created appRouter) and we have to pass in some configuration as parameters to this function to mention which component to render when I load my “/Path”, where Path can be any component (for ex :- In “/About“, “/”, “/Home” etc.).
Now, there are other Routers too provided by react-router-dom, but the one we are using is the most trusted one.

- For basic configuration, we have to pass an array as parameter to the createBrowserRouter(). Each element will have a “path” and an “element” which will tell which element (component) to render when we load a certain path.
- We will always create a router, by calling the createBrowserRouter() function, after the declaration of the component which we will be using in the config parameter for router. This is because, if we make our router even before a component is defined in JS, the component will not be rendered properly
- Earlier we were using



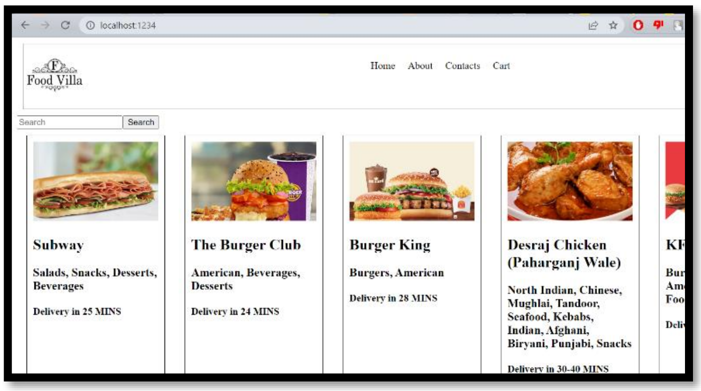
```
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<AppLayout />);
```

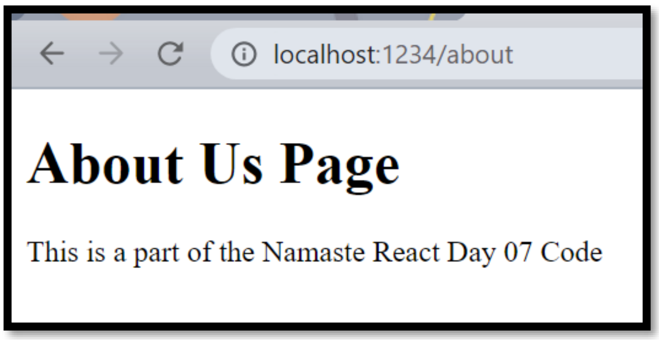
to render the AppLayout component. But now, we will be having 2 components and based on the path we will decide which component to render. So, insteas of rendering AppLayout directly inside the root element, we will be using a component named **RouterProvider** imported from react-router-dom. And this component accept props in the form of “router={appRouter}”.
This will help to load only component specific to that Path.



```
import About from "./components/About";
import { createBrowserRouter, RouterProvider } from "react-router-dom";

> const AppLayout = () => (⋯
)

const appRouter = createBrowserRouter([
    {
        path: "/",
        element: <AppLayout />
    },
    {
        path: "/about",
        element: <About />
    }
])

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<RouterProvider router={appRouter} />);
```

So, when we search for “/” path (1st image), we will get our original homepage, and if we write “/about” after that, we will get our About page (2nd image) :-

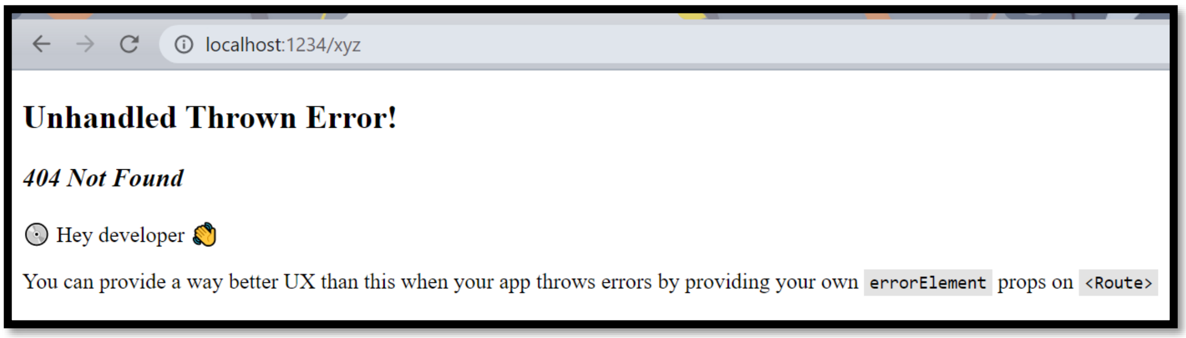

localhost:1234
Food Villa
Home About Contacts Cart
Search
Search
Subway
Salads, Snacks, Desserts, Beverages
Delivery in 25 MINS
The Burger Club
American, Beverages, Desserts
Delivery in 24 MINS
Burger King
Burgers, American
Delivery in 28 MINS
Desraj Chicken (Paharganj Wale)
North Indian, Chinese, Mughlai, Tandoor, Seafood, Kebabs, Indian, Afghani, Biryani, Punjabi, Snacks
Delivery in 30-40 MINS



About Us Page

This is a part of the Namaste React Day 07 Code

- If we load a path, that has been defined in our router, we get an error and ths error page displayed is from the react-router-dom library an the console we get error “No routes matched location /xyz” :-



localhost:1234/xyz

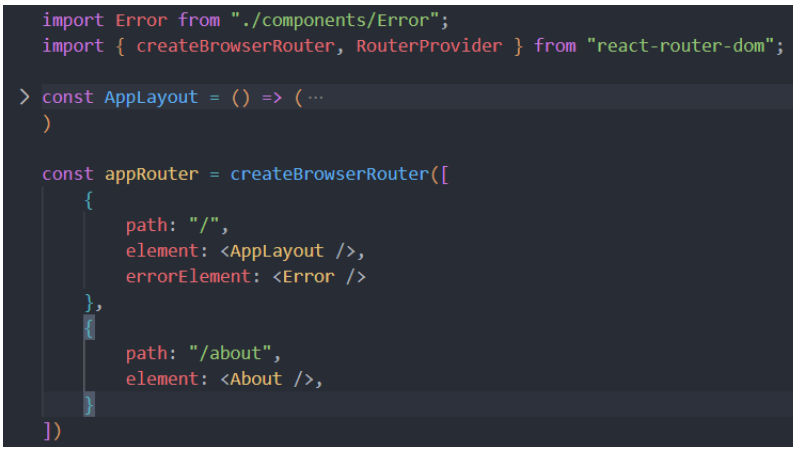
Unhandled Thrown Error!

404 Not Found

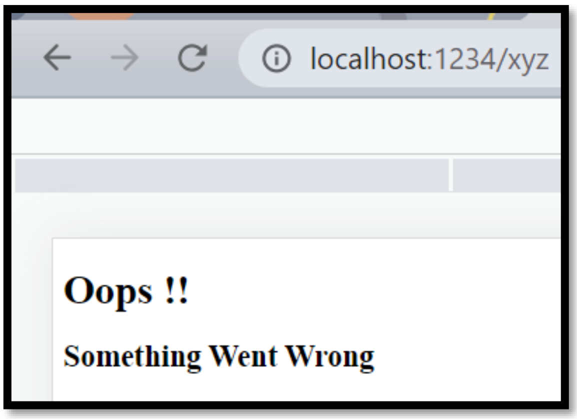
💿 Hey developer 👋

You can provide a way better UX than this when your app throws errors by providing your own `errorElement` props on `<Route>`

Instead of the error page we get from react-router-dom, we can create our own error page (a separate Error component) and we have to pass this Error info/ component to our Router config using the **errorElement** property. Use it in the path object of root directory.



```
import Error from "./components/Error";
import { createBrowserRouter, RouterProvider } from "react-router-dom";

const AppLayout = () => (…
)

const appRouter = createBrowserRouter([
    {
        path: "/",
        element: <AppLayout />,
        errorElement: <Error />
    },
    {
        path: "/about",
        element: <About />,
    }
])
```



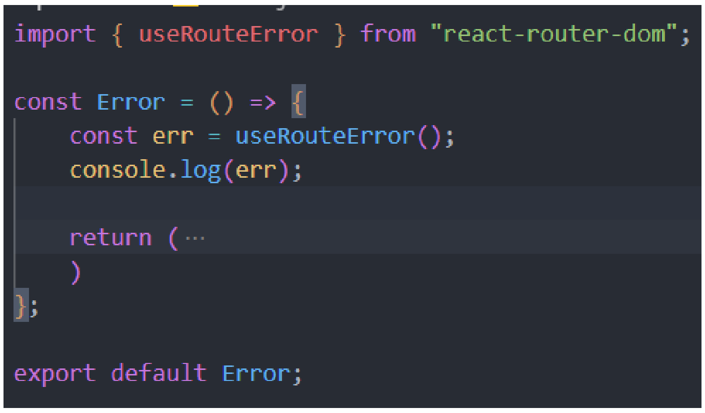
localhost:1234/xyz

Oops !!

Something Went Wrong

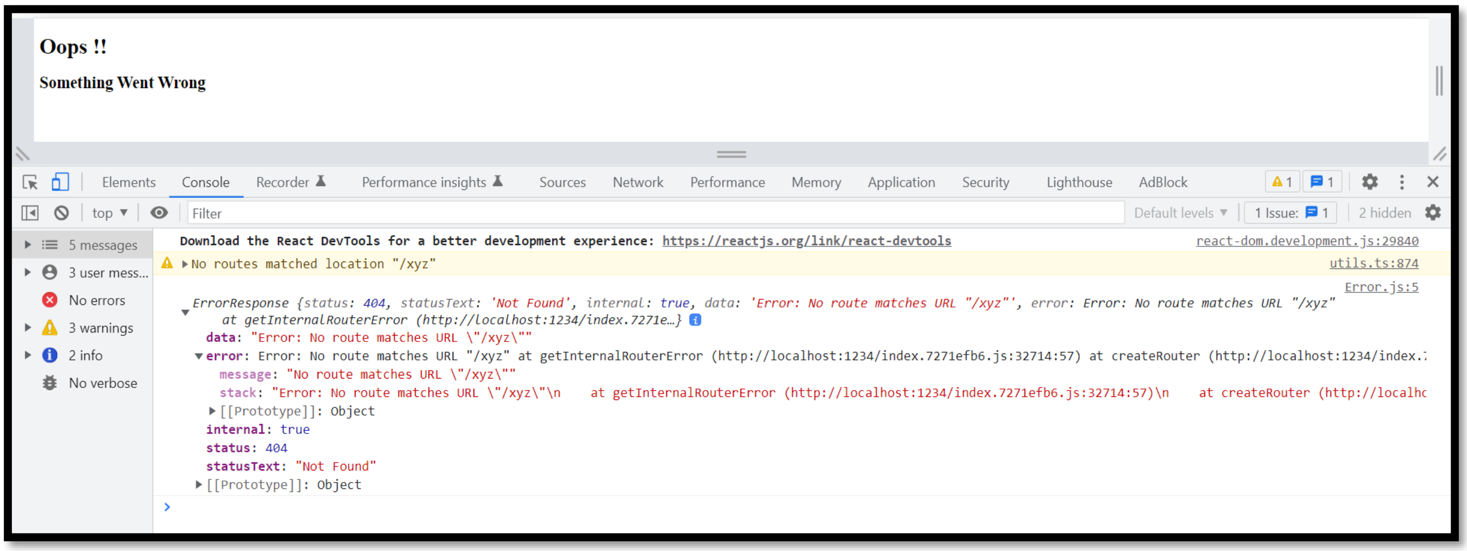
The same error page will be shown if we access any wrong path after the “/”.

- Now, the error message we are showing is very vague. To make it more detailed, we use a hook given to us by react-router-dom named **useRouteError**. We can identify hooks when there’s a prefix “use”. We can console log the return value we get from the function on accessing a wrong path. (Left image is the code in Error.js)



```
import { useRouteError } from "react-router-dom";

const Error = () => {
    const err = useRouteError();
    console.log(err);

    return (…
    )
};

export default Error;
```

And below we can see that console.log(err) gives us an object which automatically determines the type of error (for ex :- here, it is Error 404 -> Not Found)

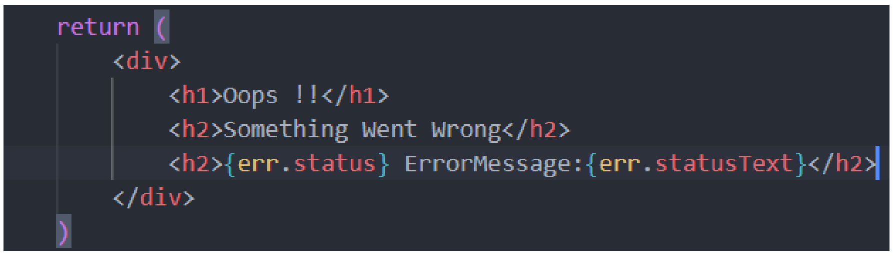


Oops !!

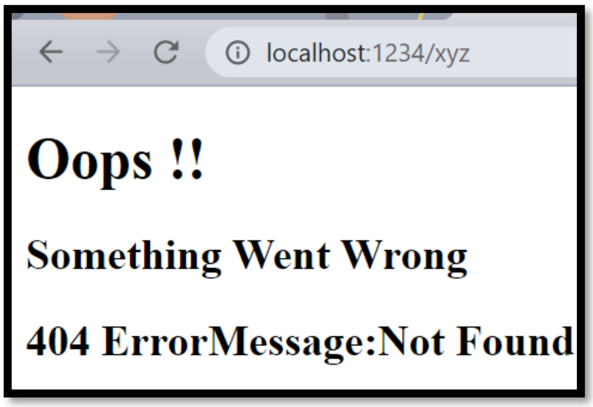
Something Went Wrong

```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools    react-dom.development.js:29840
⚠ ▶No routes matched location "/xyz"                                                                          utils.ts:874
                                                                                                              Error.js:5
▼ErrorResponse {status: 404, statusText: 'Not Found', internal: true, data: 'Error: No route matches URL "/xyz"', error: Error: No route matches URL "/xyz"
      at getInternalRouterError (http://localhost:1234/index.7271e…}
   data: "Error: No route matches URL \"/xyz\""
  ▼error: Error: No route matches URL "/xyz" at getInternalRouterError (http://localhost:1234/index.7271efb6.js:32714:57) at createRouter (http://localhost:1234/index.
     message: "No route matches URL \"/xyz\""
     stack: "Error: No route matches URL \"/xyz\"\n    at getInternalRouterError (http://localhost:1234/index.7271efb6.js:32714:57)\n    at createRouter (http://localh
    ▶[[Prototype]]: Object
   internal: true
   status: 404
   statusText: "Not Found"
  ▶[[Prototype]]: Object
>
```

Now, we can use this object, by destructuring it and showing the type of error in our UI too :-



```
return (
    <div>
        <h1>Oops !!</h1>
        <h2>Something Went Wrong</h2>
        <h2>{err.status} ErrorMessage:{err.statusText}</h2>
    </div>
)
```



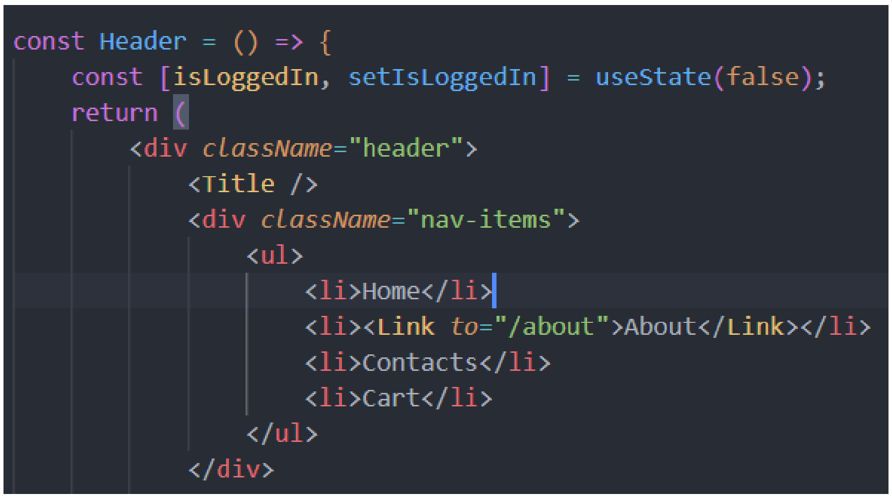
localhost:1234/xyz

Oops !!

Something Went Wrong

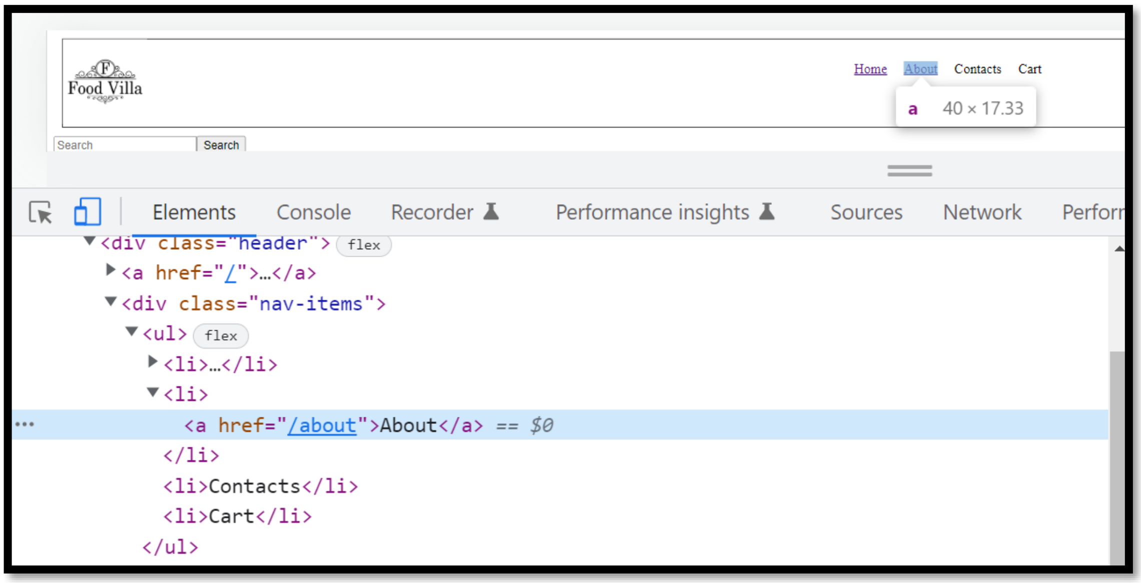
404 ErrorMessage:Not Found

- Now, we have built the About Page and we want to redirect the user to that page when he clicks on the About text. The 1st idea is to use an anchor tag, the problem with that is, it tries to refresh the whole page (see that by clicking the logo, since we used an anchor tag there). But here we are building a **Single Page Application (SPA)**. This means when we are going to another page, our app should not make a network call to that page to fetch the required data. For this, react-router-dom gives us a tag named **Link.** It works in the same way as anchor tag, just in place of "a" and "href", there is "Link" and "to".



```
const Header = () => {
    const [isLoggedIn, setIsLoggedIn] = useState(false);
    return (
        <div className="header">
            <Title />
            <div className="nav-items">
                <ul>
                    <li>Home</li>
                    <li><Link to="/about">About</Link></li>
                    <li>Contacts</li>
                    <li>Cart</li>
                </ul>
            </div>
```

Now, if you will inspect the "About" text, you will that it is actually an anchor tag. So, behind the scenes the Link tag uses the anchor tag only. React keeps track of our links through Link, but browser understands only anchor tag, so at the end in browser, Link gets converted to "a"



```
▼<div class="header"> flex
  ▶<a href="/">…</a>
  ▼<div class="nav-items">
    ▼<ul> flex
      ▶<li>…</li>
      ▼<li>
          <a href="/about">About</a> == $0
        </li>
        <li>Contacts</li>
        <li>Cart</li>
      </ul>
```

Types of Routing :-

There are 2 types of Routing :-

i. **Server Side Routing** :- The way in which all our pages come from the server i.e. we have make a network call to load a new page.

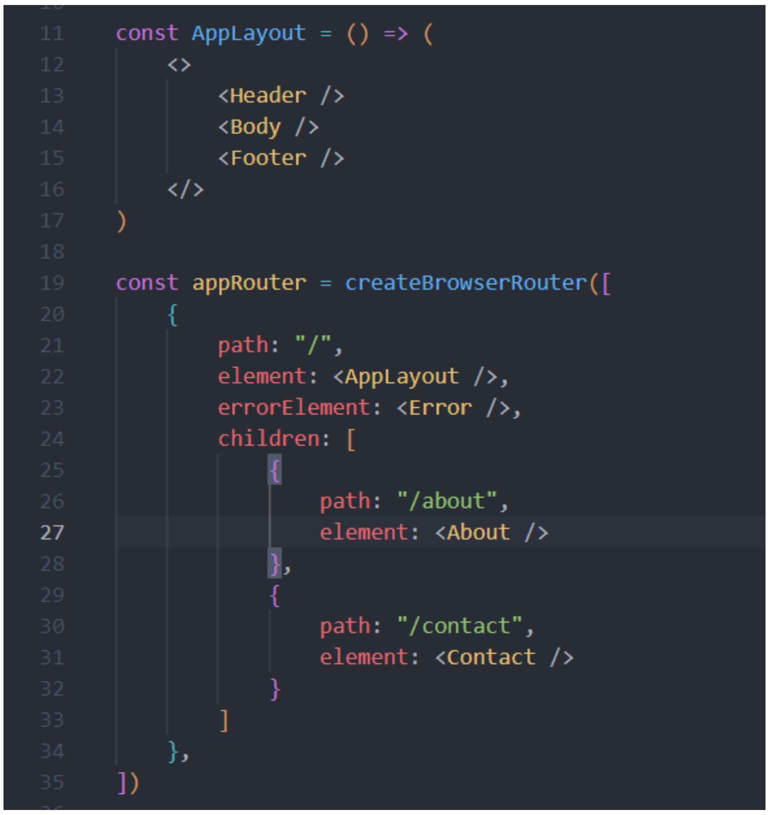
ii. **Client Side Routing** :- The way in which the client can access any page without making any network call to our server because all our components will be already there in the code. This is how React works.

Problem with our About Page

When our About Page is loaded, we lose our header component too. So, we need our About component between the Header and the Footer. Same is for all the Home, Contacts components . For this, we have to change our Routing config. Earlier, we said that if path is “/about” we will load the About page, but what if we have to change just the Body component and not the entire AppLayout. So, I have to make my About page a children of AppLayout.

Solving the Problem through Nested Routing and Outlet component

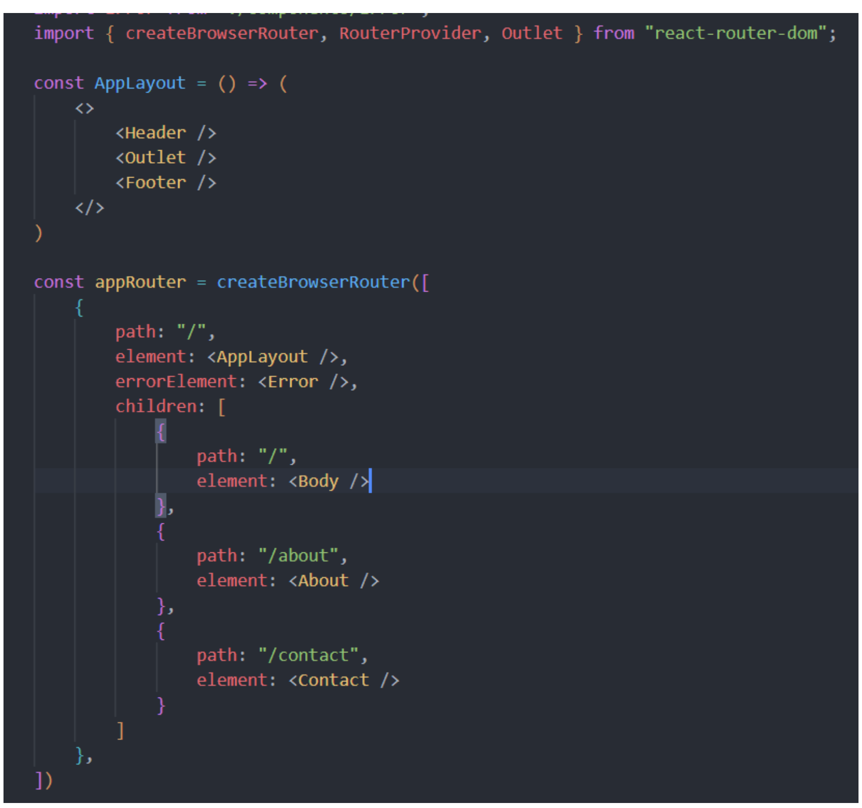
We can create another component too for Contacts. Now, just like we specified the rendering of About Page in appRouter using an object, we have to remove that object from there and keep it inside a “children” array and this “children” array will be a property of the home path. We will also put the object having the path for Contact Page in it.



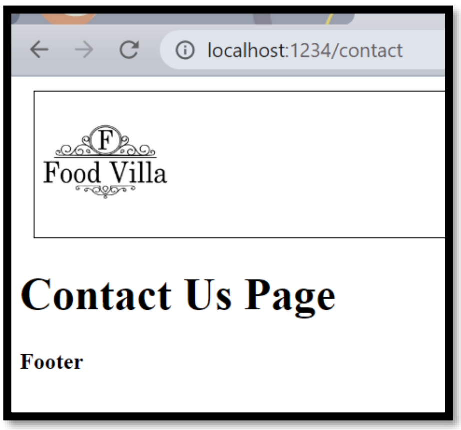
```
11  const AppLayout = () => (
12      <>
13          <Header />
14          <Body />
15          <Footer />
16      </>
17  )
18
19  const appRouter = createBrowserRouter([
20      {
21          path: "/",
22          element: <AppLayout />,
23          errorElement: <Error />,
24          children: [
25              {
26                  path: "/about",
27                  element: <About />
28              },
29              {
30                  path: "/contact",
31                  element: <Contact />
32              }
33          ]
34      },
35  ])
```

However, if you go to the browser you will see that only the Header,Body and the Footer components are getting loaded, not the About, Contact components for paths :- “/”, “/about”, “contact”.
Now, you might think of writing the About and Contact component between line 13 and line 15 too, but then it will load all Body, About and Contacts component at the same time for all the 3 paths.
So, we can say that between line 13 and line 15 is the Outlet where we want to fill in different pages.

So, react-router-dom gives us a component named **Outlet**, which we can import and specify between line 13 and line 15, instead of <Body /> and our Outlet will be filled by the children config.
So now, alongside About and Contact component, we have to make an object specifying when our Body component will be loaded too.



```
import { createBrowserRouter, RouterProvider, Outlet } from "react-router-dom";

const AppLayout = () => (
    <>
        <Header />
        <Outlet />
        <Footer />
    </>
)

const appRouter = createBrowserRouter([
    {
        path: "/",
        element: <AppLayout />,
        errorElement: <Error />,
        children: [
            {
                path: "/",
                element: <Body />
            },
            {
                path: "/about",
                element: <About />
            },
            {
                path: "/contact",
                element: <Contact />
            }
        ]
    },
])
```



localhost:1234/contact

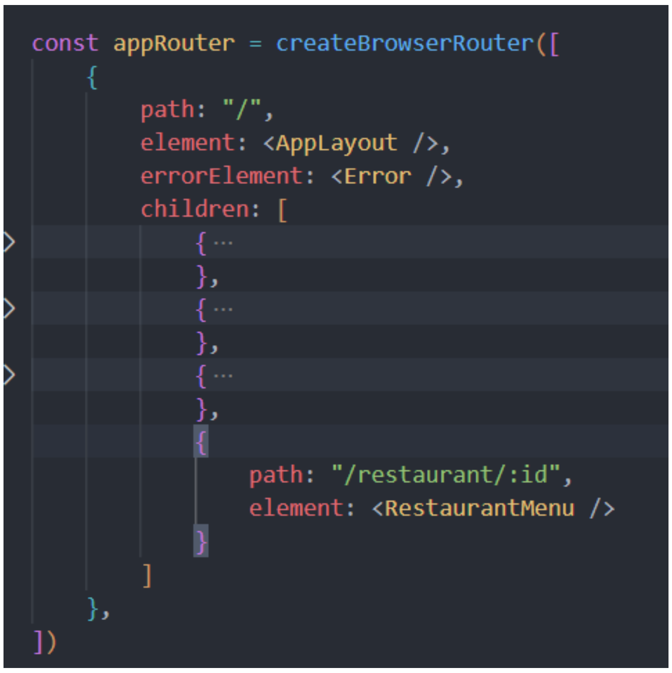
Food Villa

Contact Us Page

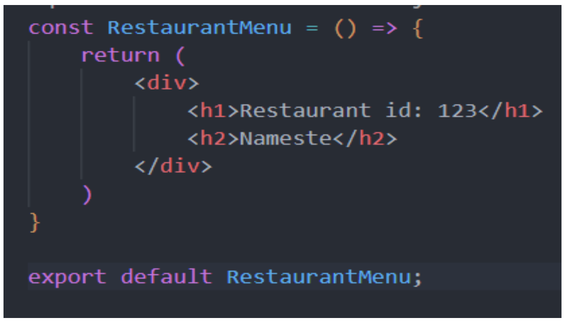
Footer

Dynamic Routing :-

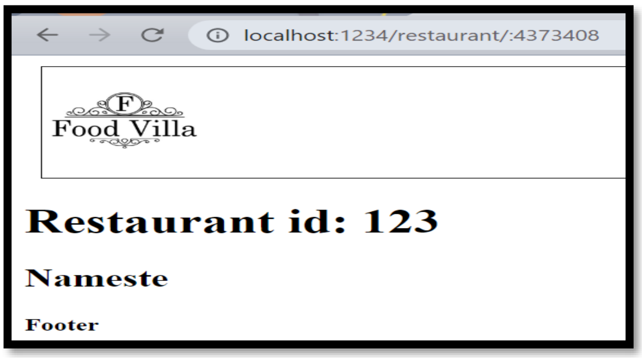
We want create a feature such that when we click on a restaurant-card, we will directed to another page (let's say RestaurantMenu Page). For this we need dynamic routing and not static.



```
const appRouter = createBrowserRouter([
    {
        path: "/",
        element: <AppLayout />,
        errorElement: <Error />,
        children: [
            {…
            },
            {…
            },
            {…
            },
            {
                path: "/restaurant/:id",
                element: <RestaurantMenu />
            }
        ]
    },
])
```



```
const RestaurantMenu = () => {
    return (
        <div>
            <h1>Restaurant id: 123</h1>
            <h2>Nameste</h2>
        </div>
    )
}

export default RestaurantMenu;
```



localhost:1234/restaurant/:4373408

Food Villa

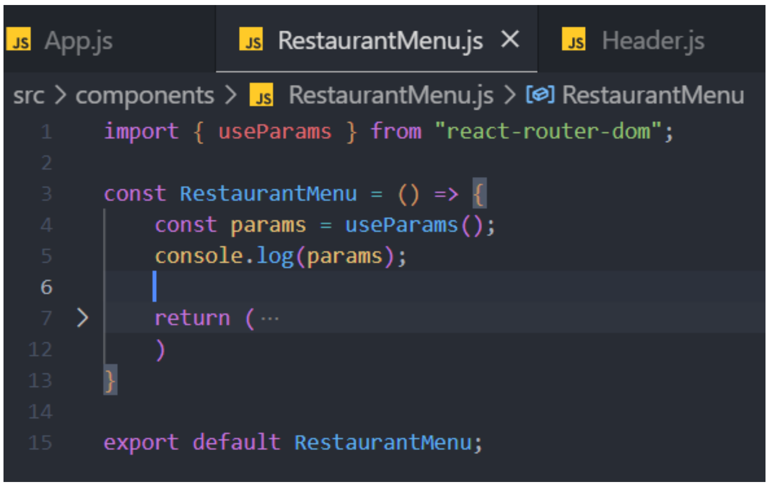
Restaurant id: 123

Nameste

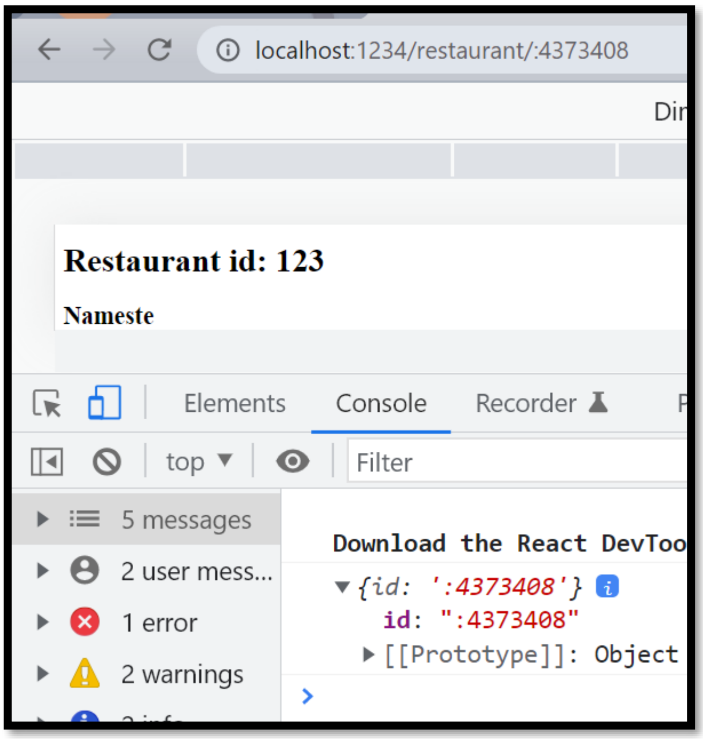
Footer

useParams Hook:-

Now, to consume the "id" in out path variable, react-router-dom provides us useParams Hook. This function returns the parameter in out path. So, if we try to print the return value, then :-
And we can see that we can use our "id" by destructuring the id object.



```
import { useParams } from "react-router-dom";

const RestaurantMenu = () => {
    const params = useParams();
    console.log(params);

    return (…
    )
}

export default RestaurantMenu;
```



Restaurant id: 123

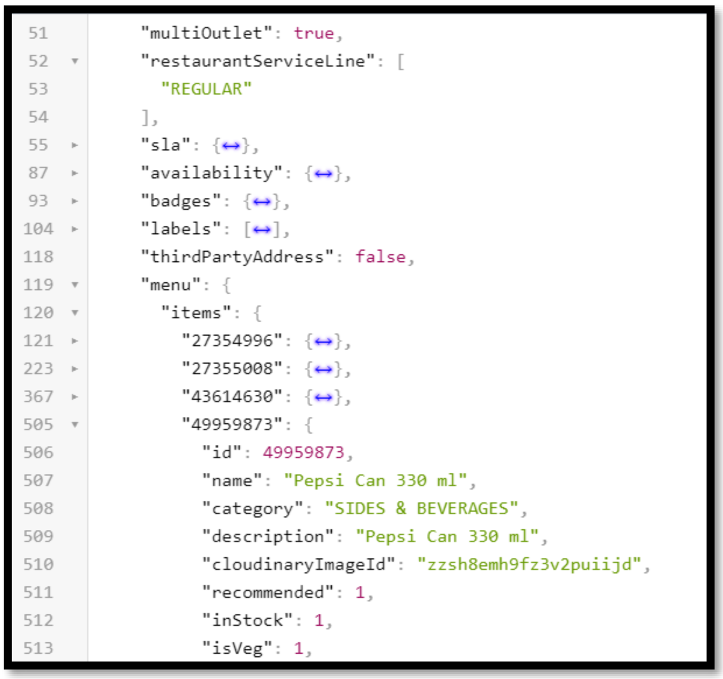
Nameste

```
Download the React DevToo
▼{id: ':4373408'}
   id: ":4373408"
  ▶[[Prototype]]: Object
```

Using Swiggy API to make RestaurantMenu page:-

- Now, instead of making the RestaurantMenu page data by ourselves, we will try to use the API of any one restaurant of Swiggy (see from 2:00:19 till the API is fetched to see how we get the API). Then, we know that fetching the API should be done inside an async function which should be called inside the callback function of useEffect Hook. So, we will import useEffect and code the above things.
- Also, since now we are rendering data from API, we know that the data will be fetched only after the initial rendering. But during that initial render, the component will not get access to our API data and the data will be shown as undefined. To overcome that, we will do Optional rendering :- render the Shimmer if API data is not fetched, else render the RestaurantMenu component.

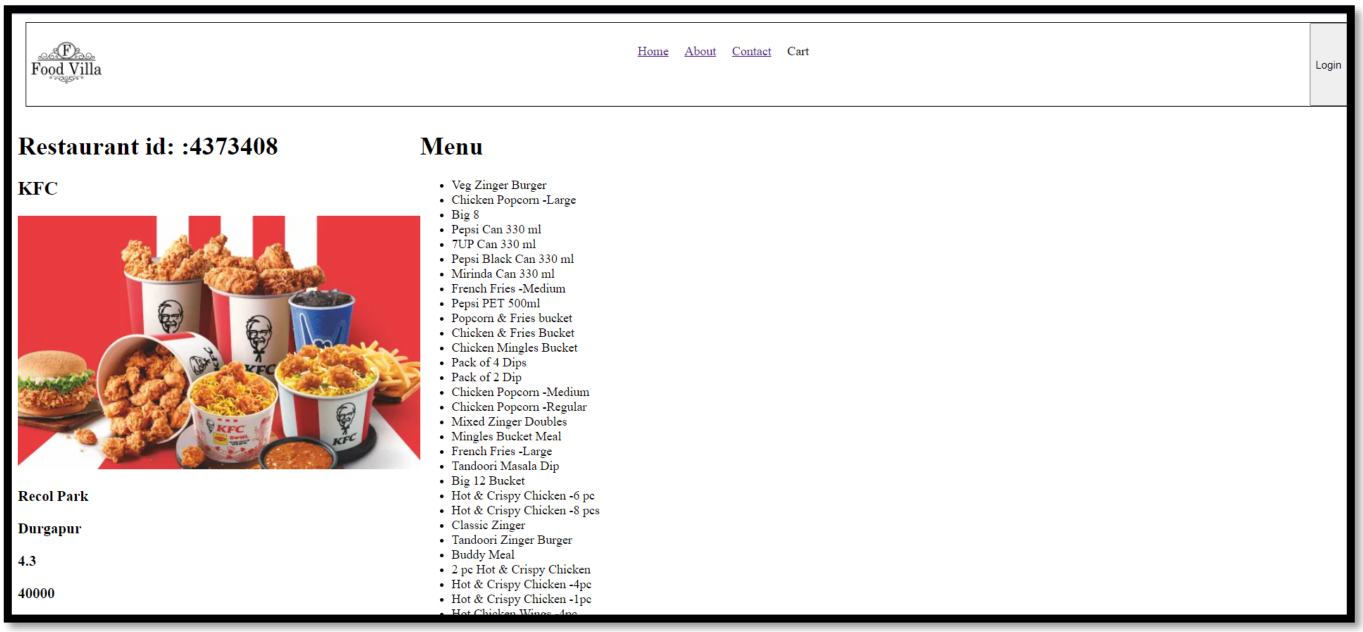
- Also, to use the API data, we need a local state variable (here we use restaurant) and we populate it with the API data in the async function.
- You will see that in the API data, there is a key named "items" inside "menu" which is an Object of Objects



```
      "multiOutlet": true,
      "restaurantServiceLine": [
        "REGULAR"
      ],
      "sla": {↔},
      "availability": {↔},
      "badges": {↔},
      "labels": [↔],
      "thirdPartyAddress": false,
      "menu": {
        "items": {
          "27354996": {↔},
          "27355008": {↔},
          "43614630": {↔},
          "49959873": {
            "id": 49959873,
            "name": "Pepsi Can 330 ml",
            "category": "SIDES & BEVERAGES",
            "description": "Pepsi Can 330 ml",
            "cloudinaryImageId": "zzsh8emh9fz3v2puiijd",
            "recommended": 1,
            "inStock": 1,
            "isVeg": 1,
```

We would also like to iterate over all the items in the "items" Object. So, to do that, we will use Object.values() to convert it into an array and then map over each object one by one. You can check the data type of "items" in the console before and after passing through Object.values(), by writing a JS code of console.log(restaurant?.data?.items), in the rendered component .

- Also add a display of flex to display the 2 divs in RestaurantMenu side by side (see the code, you will understand).



Restaurant id: :4373408

KFC

Recol Park

Durgapur

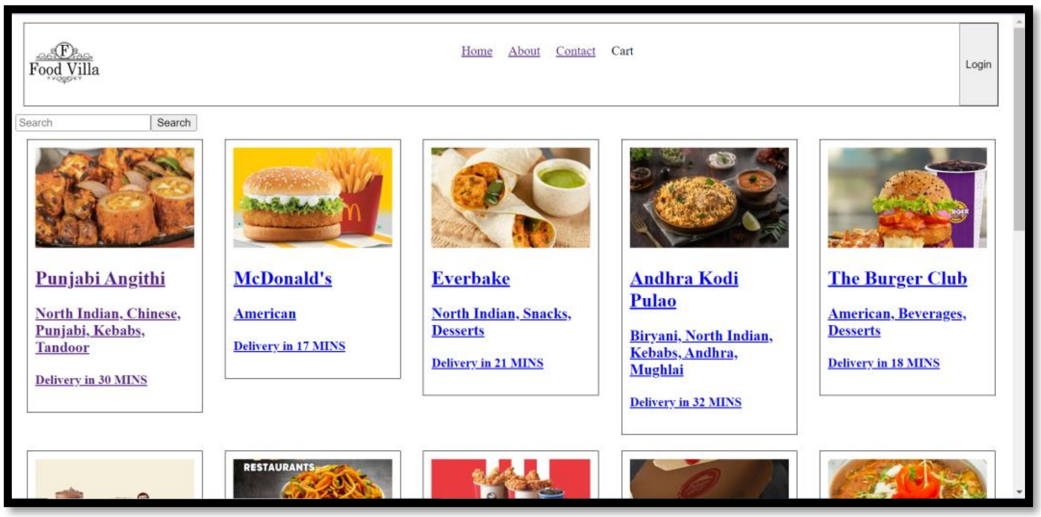
4.3

40000

Menu

- Veg Zinger Burger
- Chicken Popcorn -Large
- Big 8
- Pepsi Can 330 ml
- 7UP Can 330 ml
- Pepsi Black Can 330 ml
- Mirinda Can 330 ml
- French Fries -Medium
- Pepsi PET 500ml
- Popcorn & Fries bucket
- Chicken & Fries Bucket
- Chicken Mingles Bucket
- Pack of 4 Dips
- Pack of 2 Dip
- Chicken Popcorn -Medium
- Chicken Popcorn -Regular
- Mixed Zinger Doubles
- Mingles Bucket Meal
- French Fries -Large
- Tandoori Masala Dip
- Big 12 Bucket
- Hot & Crispy Chicken -6 pc
- Hot & Crispy Chicken -8 pcs
- Classic Zinger
- Tandoori Zinger Burger
- Buddy Meal
- 2 pc Hot & Crispy Chicken
- Hot & Crispy Chicken -4pc
- Hot & Crispy Chicken -1pc

- Now, we have taken the API for one random restaurant from Swiggy. How about taking using each restaurant's data from the restaurants we are showing in our home page. To do, we will call the API dynamically i.e. using that specific restaurant's ID in the RestaurantMenu.js and we will also provide Link tag to each Restaurant Card in Body.js.



Punjabi Angithi
North Indian, Chinese, Punjabi, Kebabs, Tandoor
Delivery in 30 MINS

McDonald's
American
Delivery in 17 MINS

Everbake
North Indian, Snacks, Desserts
Delivery in 21 MINS

Andhra Kodi Pulao
Biryani, North Indian, Kebabs, Andhra, Mughlai
Delivery in 32 MINS

The Burger Club
American, Beverages, Desserts
Delivery in 18 MINS

#NOTES :-

- Learn more about react-router-dom from it’s officalDocs
- We can create forms in React much easier using a library called Formik
- We might encounter Regex expressions while using Formik, to understand what any Regex means, use this

When time, create a Login Page with Formik