
# A basic Food Delivery Website using MERN stack

## A PROJECT REPORT

***Submitted by:***

Riya Saxena – 2115000863

Vaishnavi Sharma - 2115001086

Anand Pratap Singh - 2115000143

***Submitted to:***

Mr. Sanjay Madaan

Technical Trainer (CEA Dept.)

***in partial fulfilment for the award of the degree of***

## Bachelor of Technology

**IN**

Computer Science and Engineering

3rd Year Session (2023-2024)

**GLA University, Mathura**

## BONAFIDE CERTIFICATE

Certified that this project report **"Food Delivery Website using MERN stack"** is the bonafide work of "**Riya Saxena, Vaishnavi Sharma,** & **Anand Pratap Singh"** who carried out the project work under my supervision.

| | |
|---|---|
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| **Dr. Sandeep Rathor** | **Mr. Sanjay Madaan** |
| 3rd Year HOD **(**CEA Dept.) | Technical Trainer (CEA Dept.) |
| **HOD SIGNATURE** | **SUPERVISOR SIGNATURE** |

# CONTENTS

# ABSTRACT

"Introducing our MERN-powered food delivery app, where simplicity meets satisfaction. With a user-centric design, customers can easily create their account by sign-up with their some details and can login to browse menus and can place orders with a few taps, Seamless integration allows users to effortlessly review their order history, organized by date, ensuring a smooth and enjoyable experience every time they satisfy their cravings. Embracing convenience without compromising on quality, our app redefines the art of dining in."

# INTRODUCTION

## 1.1 Project Objective:

The main objective of our project is to develop a user-friendly food delivery website that streamlines the process of ordering food online. By integrating the MERN stack, we aim to provide a seamless experience for users to browse menus, place orders, and track their order history. Additionally, the platform will include a checkout option to clear the cart once the order is delivered.

## 1.2 Background:

In response to the evolving demands of modern life, our team developed a MERN-based food delivery app to streamline the dining experience. Recognizing the challenges of traditional dining, such as limited options and long wait times, we sought to provide a solution that seamlessly connects hungry customers with a diverse array of culinary delights. Through our platform, users can easily browse menus, place orders, and track order history, while restaurant partners benefit from efficient order management. More than just a transactional tool, our app fosters community, supports local businesses, and enhances the overall dining experience, ushering in a new era of convenience and culinary exploration.

## 1.3 Scope of the project:

Our project will encompass the creation of a robust food delivery platform with user authentication (login and sign up), a cart portal powered by use Context and use Reducer for efficient state management, and an order history section. The checkout option will be implemented to clear the cart once the user had added all its required food items.

## 1.4 Significance of the project:

This food delivery project is a game-changer because it takes away all the hassle of figuring out what to eat and where to get it. You can just pick up your phone, browse through tons of delicious options from nearby restaurants, and have your favourite meal delivered right to your doorstep. It's not just about convenience though; it's also about supporting local eateries and giving them a boost. Plus, in today's world, where staying safe is more important than ever, having the option to get food delivered means you can enjoy tasty meals without putting yourself at risk. So, it's not just about making life easier; it's about making it tastier, more supportive of local businesses, and safer for everyone.

# SYSTEM ARCHITECTURE

## 2.1 User Interface:

**Frontend Interface:** This is the user-facing part of the system where attendees interact with the platform. It includes features for signup, login, browsing menu and adding food items to cart according to their need also user can check their order history too.

**Backend Server:** The backend server manages data storage, processing, and retrieval. It handles user authentication, database operations, and communication with other system components.

**Database**: The live server database stores all relevant data, including user's email, Contact number, name, location information, and access permissions. It is designed for scalability, reliability, and efficient querying.

**Security Measures**: Implement security measures such as password encryption with JSON Web Token (JWT).

## 2.2 Technologies Used:

**Frontend Technologies:**

- HTML/CSS/JavaScript/React/Bootstrap

**Backend Technologies:**

- **Programming Languages:** Language like JavaScript is commonly used for developing the backend logics.

- **Web Frameworks:** Express.js (Node.js) provide a structured approach for backend development.

- **Database Management Systems (DBMS):** MongoDB is used for storing all food details and also user information, and other relevant data.

**Authentication and Authorization:**

- **Authentication:** JSON web token is used to provide authentication and authorization mechanisms.

**Version Control and Collaboration:**

- **Version Control Systems**: Git and platforms like GitHub for managing source code, tracking changes, and facilitating collaboration among developers.

## 2.3 Features:

**User Sign-up and Login:** Users can create accounts using their email id, name, location and can setup a password after this they can login using their valid email Id and password.

**Food Category:** User can register can search their food items according to category like starters, pizza etc.

**Add to Cart**: Users can add there required food items to there cart according to there need like how much they want, quantity etc.

**Check-Out and Order History:** After successfully adding there required items to cart user can simply checkout that will clear the whole cart at once and after this all of the user ordered history will be shown to My Order page respective to date on which they have ordered.

# TESTING & CHALLENGES FACED

## 3.1 Testing

• **Testing Phases:** Employ comprehensive testing, including unit, integration, system, user acceptance testing, to verify functionality and performance.

• **Functional Testing:** Ensure all features like food details, browsing, login and Sign-up of users working as intended Description/feature Selection – extracts the description of locations is suitable for further process.

• **Compatibility Testing:** Verify consistent performance across different devices, browsers, and operating systems.

## 3.2 Challenges:

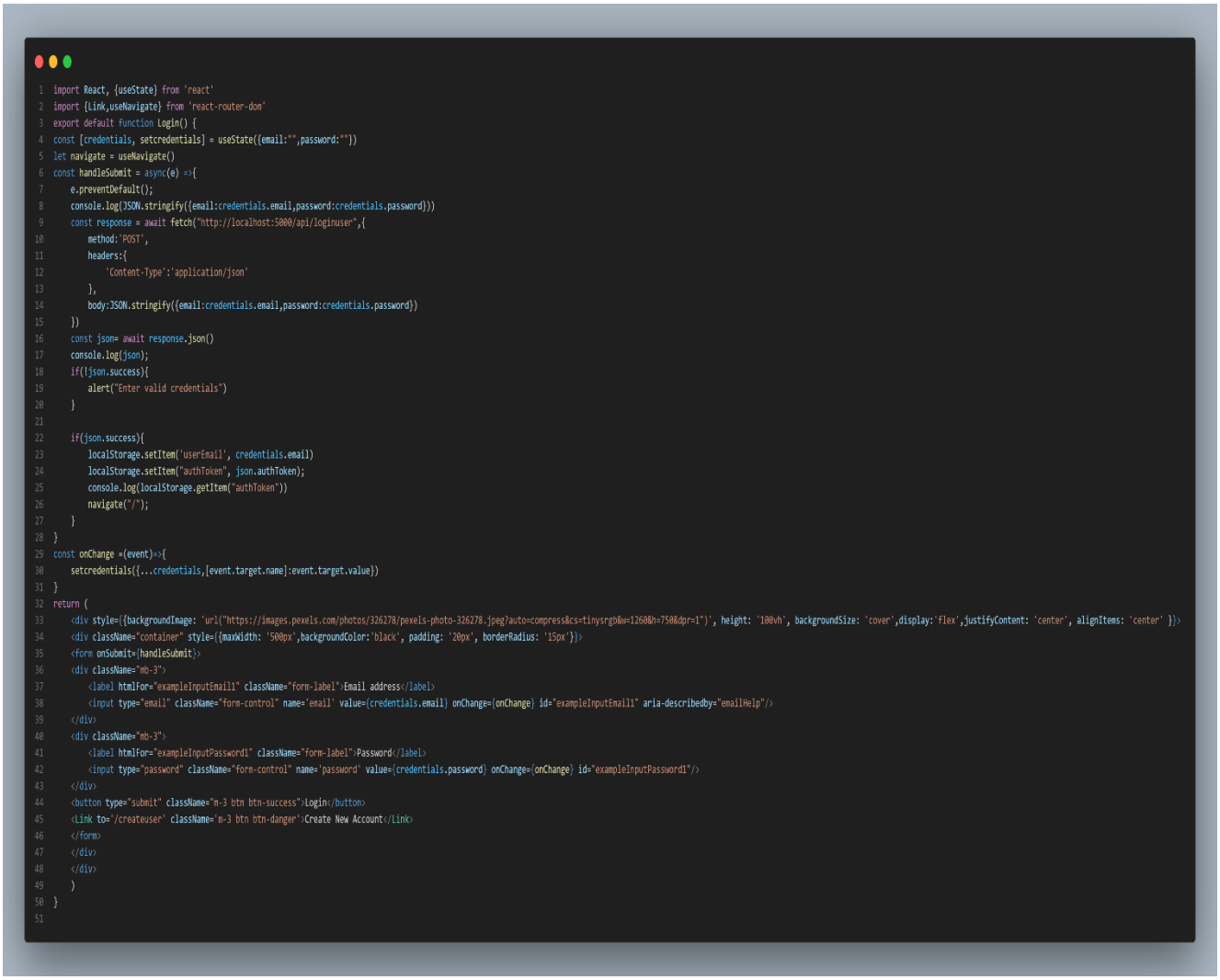
Implementing JWT for authentication presents challenges in securely generating, storing, and validating tokens across various endpoints, while ensuring expiration and renewal mechanisms. Simultaneously, creating a cart portal using useReducer demands meticulous structuring of state management logic to handle user interactions like adding, removing, and updating items in the cart, all while synchronizing with backend data and maintaining consistency between client-side and server-side operations. Balancing security, performance, and reactivity in both JWT implementation and cart portal development requires careful consideration and expertise.

# CONCLUSION

In conclusion, our mini project aims to redefine the food delivery experience by seamlessly integrating JWT authentication for user security, a user-friendly login/signup process, and a streamlined cart portal using useReducer for efficient order management. With a focus on simplicity and convenience, users can effortlessly log in or sign up, browse and select their favourite dishes, and manage their order history with ease. By addressing these challenges and incorporating user-centric features, we strive to enhance the overall satisfaction of customers and restaurant partners alike, while supporting local businesses and fostering a more enjoyable dining experience for all.

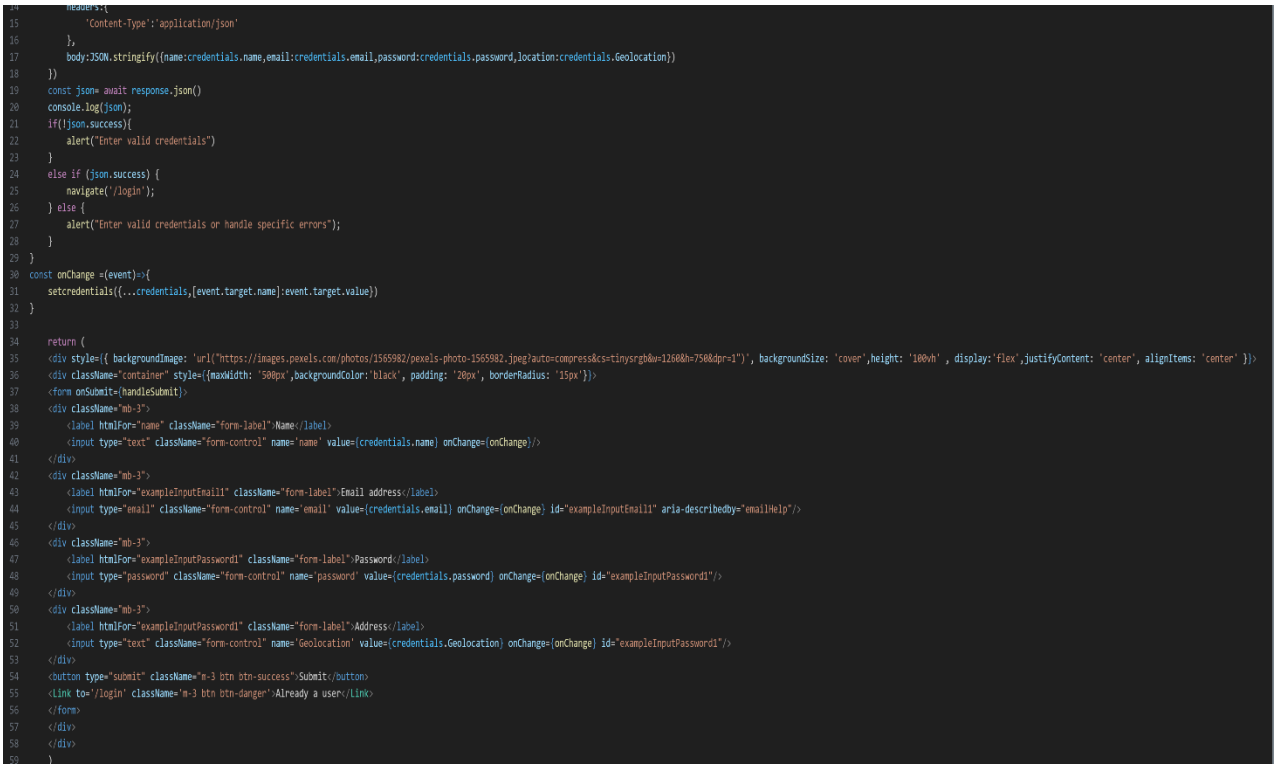
# CODE SNIPPETS

**Login page:**



```jsx
import React, {useState} from 'react'
import {Link,useNavigate} from 'react-router-dom'
export default function Login() {
const [credentials, setcredentials] = useState({email:"",password:""})
let navigate = useNavigate()
const handleSubmit = async(e) =>{
    e.preventDefault();
    console.log(JSON.stringify({email:credentials.email,password:credentials.password}))
    const response = await fetch("http://localhost:5000/api/loginuser",{
        method:'POST',
        headers:{
            'Content-Type':'application/json'
        },
        body:JSON.stringify({email:credentials.email,password:credentials.password})
    })
    const json= await response.json()
    console.log(json);
    if(!json.success){
        alert("Enter valid credentials")
    }

    if(json.success){
        localStorage.setItem('userEmail', credentials.email)
        localStorage.setItem("authToken", json.authToken);
        console.log(localStorage.getItem("authToken"))
        navigate("/");
    }
}
const onChange =(event)=>{
    setcredentials({...credentials,[event.target.name]:event.target.value})
}
return (
    <div style={{backgroundImage: 'url("https://images.pexels.com/photos/326278/pexels-photo-326278.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1")', height: '100vh', backgroundSize: 'cover',display:'flex',justifyContent: 'center', alignItems: 'center' }}>
    <div className="container" style={{maxWidth: '500px',backgroundColor:'black', padding: '20px', borderRadius: '15px'}}>
    <form onSubmit={handleSubmit}>
    <div className="mb-3">
        <label htmlFor="exampleInputEmail1" className="form-label">Email address</label>
        <input type="email" className="form-control" name='email' value={credentials.email} onChange={onChange} id="exampleInputEmail1" aria-describedby="emailHelp"/>
    </div>
    <div className="mb-3">
        <label htmlFor="exampleInputPassword1" className="form-label">Password</label>
        <input type="password" className="form-control" name='password' value={credentials.password} onChange={onChange} id="exampleInputPassword1"/>
    </div>
    <button type="submit" className="m-3 btn btn-success">Login</button>
    <Link to='/createuser' className='m-3 btn btn-danger'>Create New Account</Link>
    </form>
    </div>
    </div>
    )
}

```

## Sign-Up page:



```
        headers:{
            'Content-Type':'application/json'
        },
        body:JSON.stringify({name:credentials.name,email:credentials.email,password:credentials.password,location:credentials.Geolocation})
    })
    const json= await response.json()
    console.log(json);
    if(!json.success){
        alert("Enter valid credentials")
    }
    else if (json.success) {
        navigate('/login');
    } else {
        alert("Enter valid credentials or handle specific errors");
    }
}
const onChange =(event)=>{
    setcredentials({...credentials,[event.target.name]:event.target.value})
}

    return (
    <div style={{ backgroundImage: 'url("https://images.pexels.com/photos/1565982/pexels-photo-1565982.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1")', backgroundSize: 'cover',height: '100vh' , display:'flex',justifyContent: 'center', alignItems: 'center' }}>
    <div className="container" style={{maxWidth: '500px',backgroundColor:'black', padding: '20px', borderRadius: '15px'}}>
    <form onSubmit={handleSubmit}>
    <div className="mb-3">
        <label htmlFor="name" className="form-label">Name</label>
        <input type="text" className="form-control" name='name' value={credentials.name} onChange={onChange}/>
    </div>
    <div className="mb-3">
        <label htmlFor="exampleInputEmail1" className="form-label">Email address</label>
        <input type="email" className="form-control" name='email' value={credentials.email} onChange={onChange} id="exampleInputEmail1" aria-describedby="emailHelp"/>
    </div>
    <div className="mb-3">
        <label htmlFor="exampleInputPassword1" className="form-label">Password</label>
        <input type="password" className="form-control" name='password' value={credentials.password} onChange={onChange} id="exampleInputPassword1"/>
    </div>
    <div className="mb-3">
        <label htmlFor="exampleInputPassword1" className="form-label">Address</label>
        <input type="text" className="form-control" name='Geolocation' value={credentials.Geolocation} onChange={onChange} id="exampleInputPassword1"/>
    </div>
    <button type="submit" className="m-3 btn btn-success">Submit</button>
    <Link to='/login' className='m-3 btn btn-danger'>Already a user</Link>
    </form>
    </div>
    </div>
    )
```

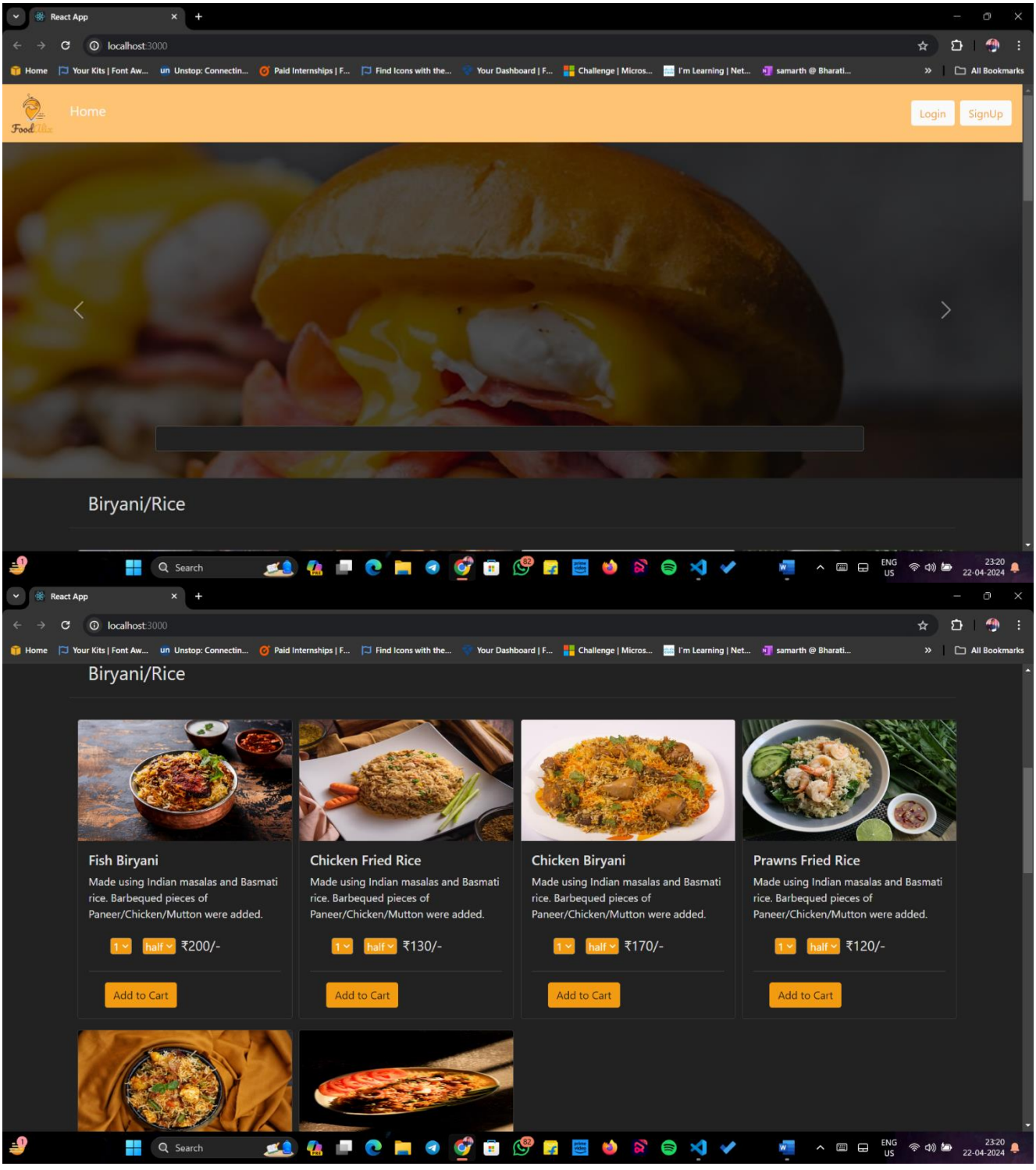
## My Orders:

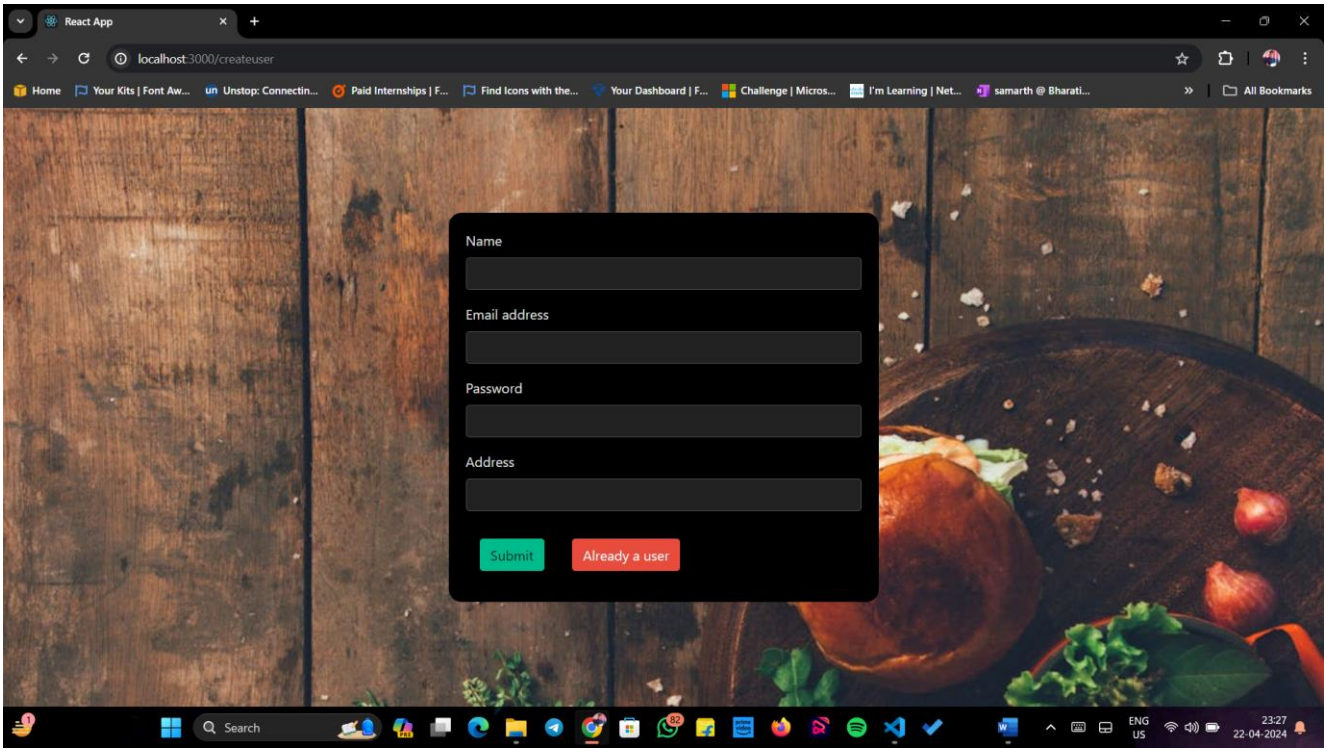
```
useEffect(() => {
    fetchMyOrder()
}, [])

return (
    <div>
        <div>
            <Navbar />
        </div>

        <div className='container'>
            <div className='row'>

                {orderData != {} ? Array(orderData).map(data => {
                    return (
                        data.orderData ?
                            data.orderData.order_data.slice(0).reverse().map((item) => {
                                return (
                                    item.map((arrayData) => {
                                        return (
                                            <div  >
                                                {arrayData.Order_date ? <div className='m-auto mt-5'>

                                                    {data = arrayData.Order_date}
                                                    <hr />
                                                </div> :

                                                    <div className='col-12 col-md-6 col-lg-3' >
                                                        <div className="card mt-3" style={{ width: "16rem", maxHeight: "360px" }}>
                                                            {/* <img src={arrayData.img} className="card-img-top" alt="..." style={{ height: "120px", objectFit: "
                                                            <div className="card-body">
                                                                <h5 className="card-title">{arrayData.name}</h5>
                                                                <div className='container w-100 p-0' style={{ height: "38px" }}>
                                                                    <span className='m-1'>{arrayData.qty}</span>
                                                                    <span className='m-1'>{arrayData.size}</span>
                                                                    <span className='m-1'>{data}</span>
                                                                    <div className=' d-inline ms-2 h-100 w-20 fs-5' >
                                                                        ₹{arrayData.price}/-
                                                                    </div>
                                                                </div>
                                                            </div>
                                                        </div>

                                                    </div>
```

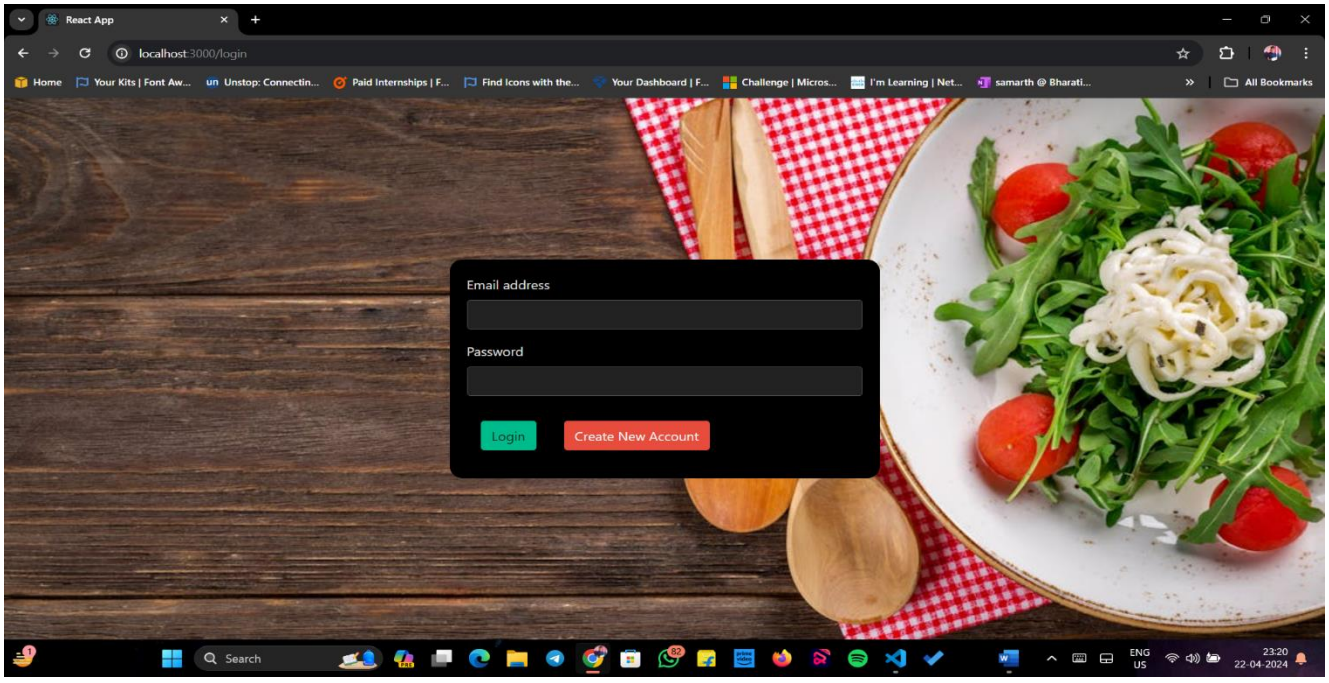
## Cart page:

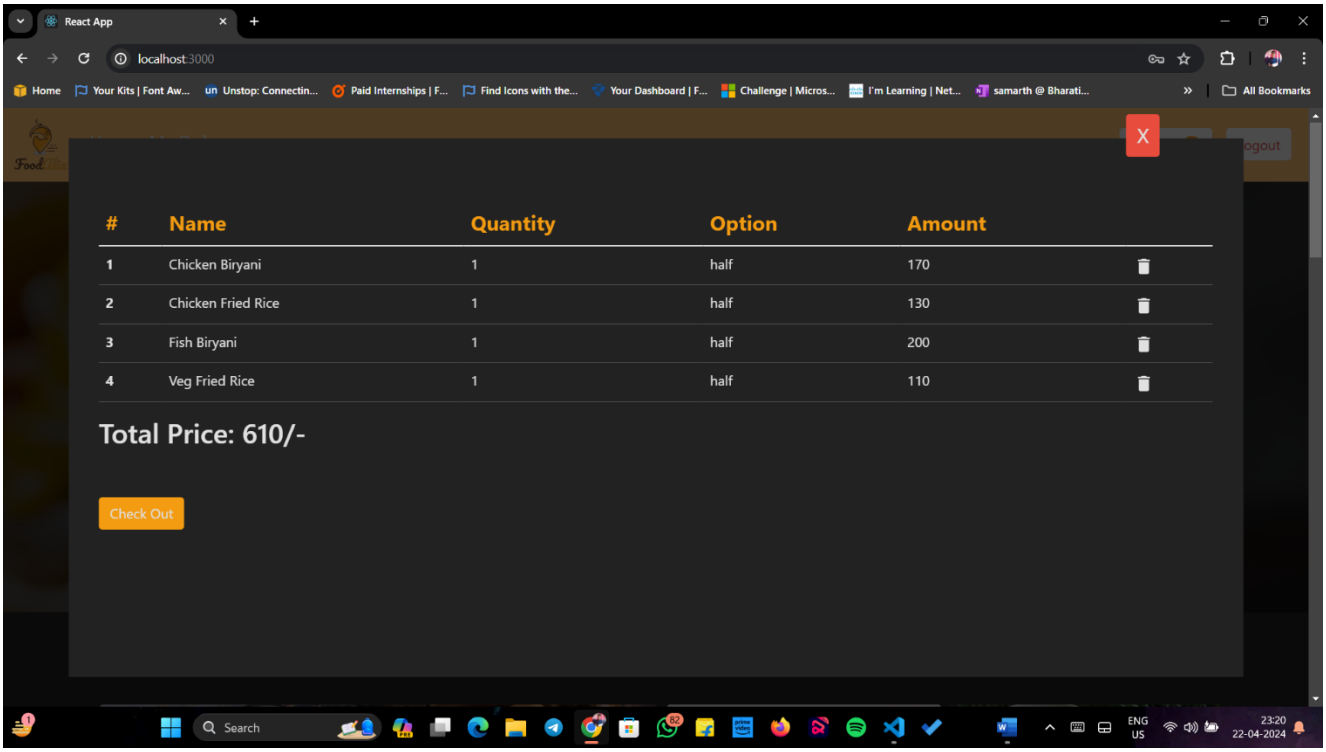
```jsx
import React from 'react'
import Delete from '@material-ui/icons/Delete'
import { useCart, useDispatchCart } from '../Components/ContextReducer';

export default function Cart() {
    let data = useCart();
    let dispatch = useDispatchCart();

    if (!data || data.length === 0) {
        return (
            <div>
                <div className='m-5 w-100 text-center fs-3'>The Cart is Empty!</div>
            </div>
        );
    }

    const handleCheckOut = async () => {
        let userEmail = localStorage.getItem("userEmail");
        let response = await fetch("http://localhost:5000/api/orderData", {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            order_data: data,
            email: userEmail,
            order_date: new Date().toDateString()
        })
        });
        console.log("JSON RESPONSE:::::", response.status)
        if (response.status === 200) {
            dispatch({ type: "DROP" })
        }
    }

    let totalPrice = data.reduce((total, food) => total + food.price, 0)
    return (
        <div>
            <div className='container m-auto mt-5 table-responsive  table-responsive-sm table-responsive-md' >
                <table className='table table-hover '>
                    <thead className=' text-warning fs-4'>
                        <tr>
                            <th scope='col' >#</th>
                            <th scope='col' >Name</th>
                            <th scope='col' >Quantity</th>
                            <th scope='col' >Option</th>
                            <th scope='col' >Amount</th>
                            <th scope='col' ></th>
                        </tr>
                    </thead>
                    <tbody>
                        {data.map((food, index) => (
                            <tr key={index}>
                                <th scope='row' >{index + 1}</th>
                                <td >{food.name}</td>
                                <td>{food.qty}</td>
                                <td>{food.size}</td>
                                <td>{food.price}</td>
                                <td ><button type="button" className="btn p-0"><Delete onClick={() => { dispatch({ type: "REMOVE", index: index }) }} /></button> </td>
                            </tr>
                        ))}
                    </tbody>
                </table>
                <div><h1 className='fs-2'>Total Price: {totalPrice}/-</h1></div>
                <div>
                    <button className='btn mt-5 ' style={{"background":"#F39C12"}} onClick={handleCheckOut}> Check Out </button>
                </div>
            </div>
        </div>
    )
}

```

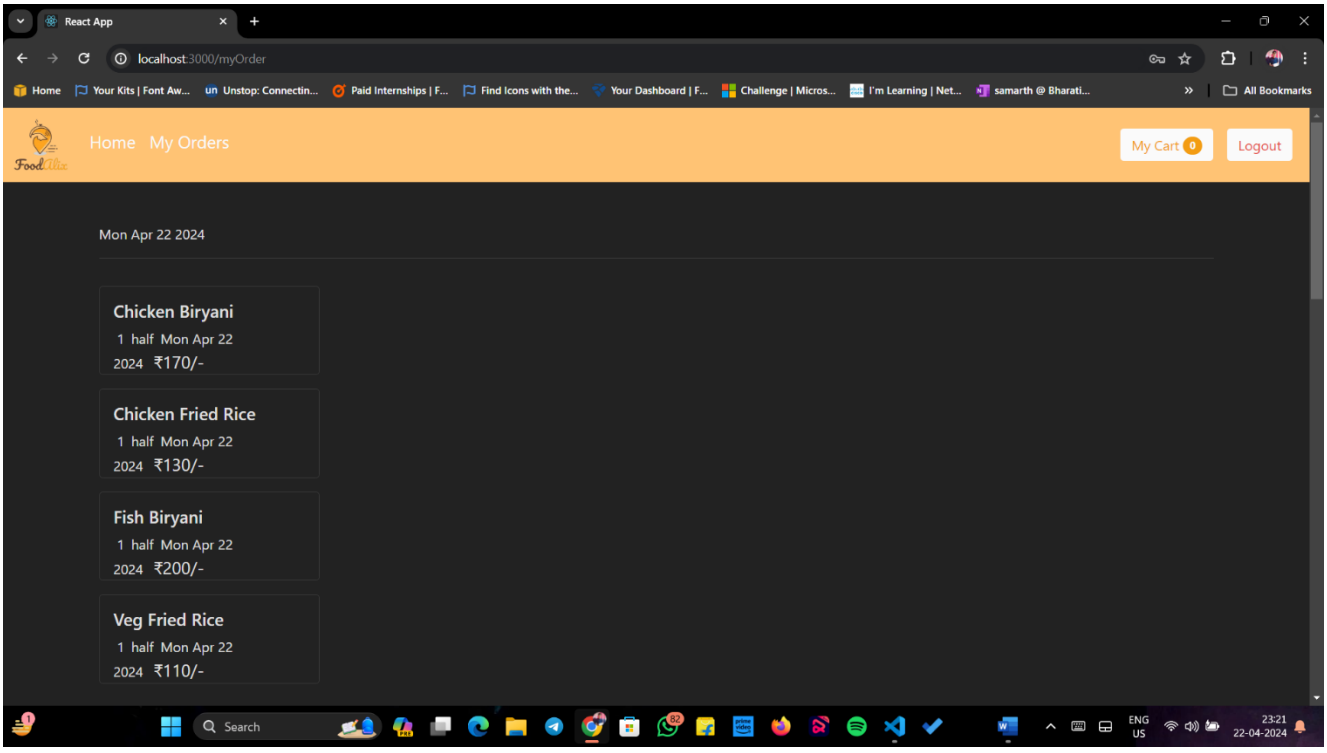
# Project Snippets:




React App
localhost:3000/createuser
Name
Email address
Password
Address
Submit
Already a user


React App
localhost:3000/login
Email address
Password
Login
Create New Account


React App
localhost:3000
Home
Your Kits | Font Aw...
Unstop: Connectin...
Paid Internships | F...
Find Icons with the...
Your Dashboard | F...
Challenge | Micros...
I'm Learning | Net...
samarth @ Bharati...
All Bookmarks
X
# Name Quantity Option Amount
1 Chicken Biryani 1 half 170
2 Chicken Fried Rice 1 half 130
3 Fish Biryani 1 half 200
4 Veg Fried Rice 1 half 110
Total Price: 610/-
Check Out
Search
ENG US
23:20 22-04-2024


React App
localhost:3000/myOrder
Home
Your Kits | Font Aw...
Unstop: Connectin...
Paid Internships | F...
Find Icons with the...
Your Dashboard | F...
Challenge | Micros...
I'm Learning | Net...
samarth @ Bharati...
All Bookmarks
Home
My Orders
My Cart 0
Logout
Mon Apr 22 2024
Chicken Biryani
1 half Mon Apr 22 2024 ₹170/-
Chicken Fried Rice
1 half Mon Apr 22 2024 ₹130/-
Fish Biryani
1 half Mon Apr 22 2024 ₹200/-
Veg Fried Rice
1 half Mon Apr 22 2024 ₹110/-
Search
ENG US
23:21 22-04-2024

# REFERENCES


- Google is used for reading node.js, express.js etc. documentation in Medium.
- Took help from React Documentation for using React JS in our project.
- Took references from YouTube videos for implementing JWT token, using useReducer for cart portal and many more.
- https://youtu.be/S20PCL9e_ks?si=BlQtkdS_tTzqOyqc
- https://youtu.be/VdXGIEYZuCw?si=47uJWbsNAt1j-3ef
- https://youtube.com/playlist?list=PLI0saxAvhd_OdRWyprSe3Mln37H0u4DAp&si=U3MmgU5G-IspCzUk
- https://getbootstrap.com/docs/5.0/getting-started/introduction/
- https://react.dev/learn
- https://www.mongodb.com/docs/