



Inspiring Excellence

CSE471 : System Analysis and Design  
Project Report  
Project Title : Restaurant Management

Group No : 07, CSE471 Lab Section : 06, Fall 2023	
ID	Name
21101164	Saimum Reza Siam
21301581	Nafiz Ahmed Rhythm

## Table of Contents

Section No	Content	Page No
1	Introduction	2
2	Functional Requirements	3
3	User Manual	4
4	Frontend & Backend Development	15
5	Technology (Framework, Languages)	47
6	Github Repo Link	47
7	Individual Contribution	47

## **Introduction**

The "Restaurant Management" project is a comprehensive application developed using the MERN (MongoDB, Express, React, Node.js) stack. This project aims to streamline the operations of a restaurant by offering features such as table management, account handling, Product Order, and admin panel. With a user-friendly interface built using React, the project provides an intuitive experience for both restaurants and customers. The integration of MongoDB ensures a robust and scalable database solution, while Node.js and Express facilitate efficient server-side operations, making the "Restaurant Management" project a powerful tool for modern restaurant management.

# Functional Requirements

## Module 1

1. Account creation
2. Account edit
3. Login and logout
4. Authentication (2FA)

## Module 2

5. Add to cart
6. Order
7. Wishlist
8. QR code scan to show the menu
9. Add food menu
10. Set the status of the order

## Module 3

11. Selling report
12. Order management
13. Add new Restaurant
14. Edit all profile
15. Able to change the restaurant logo
16. Can active and deactivate a restaurant account

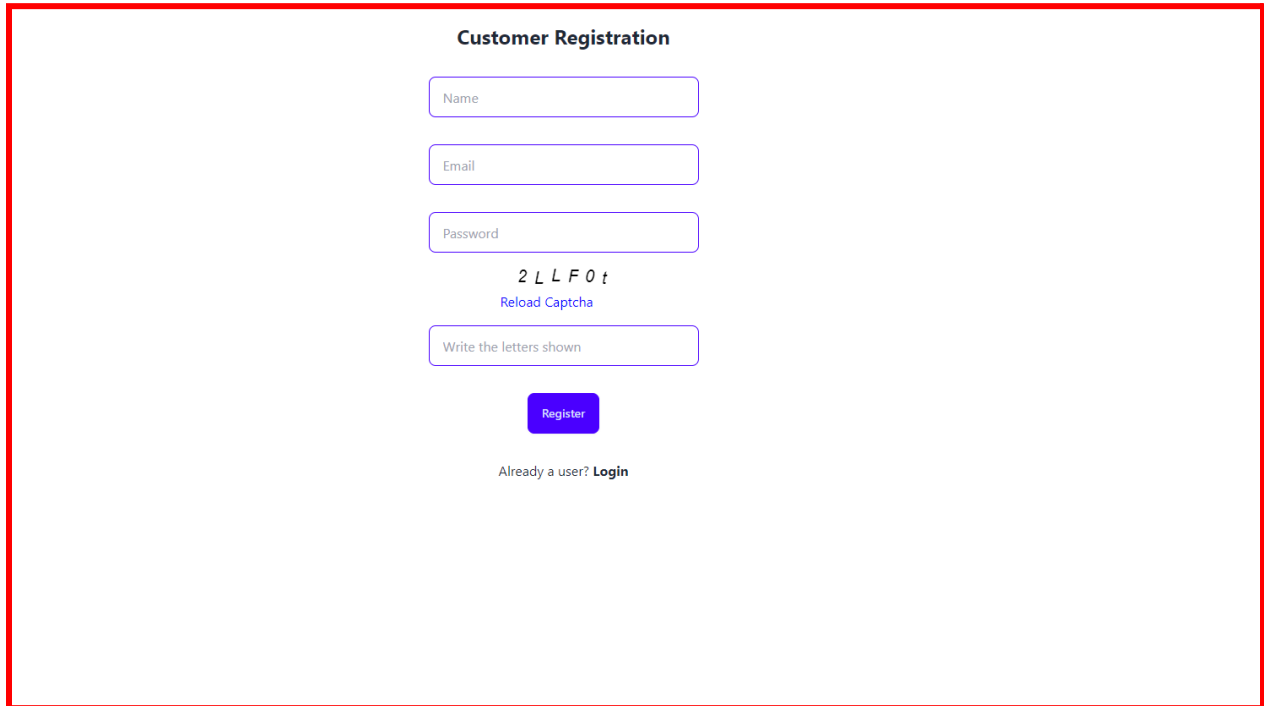
## Module 4

17. Manage/show wifi password
18. Customers can see the status of the order
19. When ordering customers can see available tables
20. Add tables

# User Manual

## Account Creation and login

<>Customer account creation

A screenshot of a 'Customer Registration' form. The form is titled 'Customer Registration' and contains three input fields: 'Name', 'Email', and 'Password'. Below the 'Password' field is a captcha image showing the text '2 L L F 0 t' with a 'Reload Captcha' link. Below the captcha is another input field labeled 'Write the letters shown'. At the bottom of the form is a blue 'Register' button and a link that says 'Already a user? Login'.

Here the customer can create a profile by providing the necessary information.

## <>Cuslomer Login

**Customer Login**

Email

Password

2 L L F 0 t

[Reload Captcha](#)

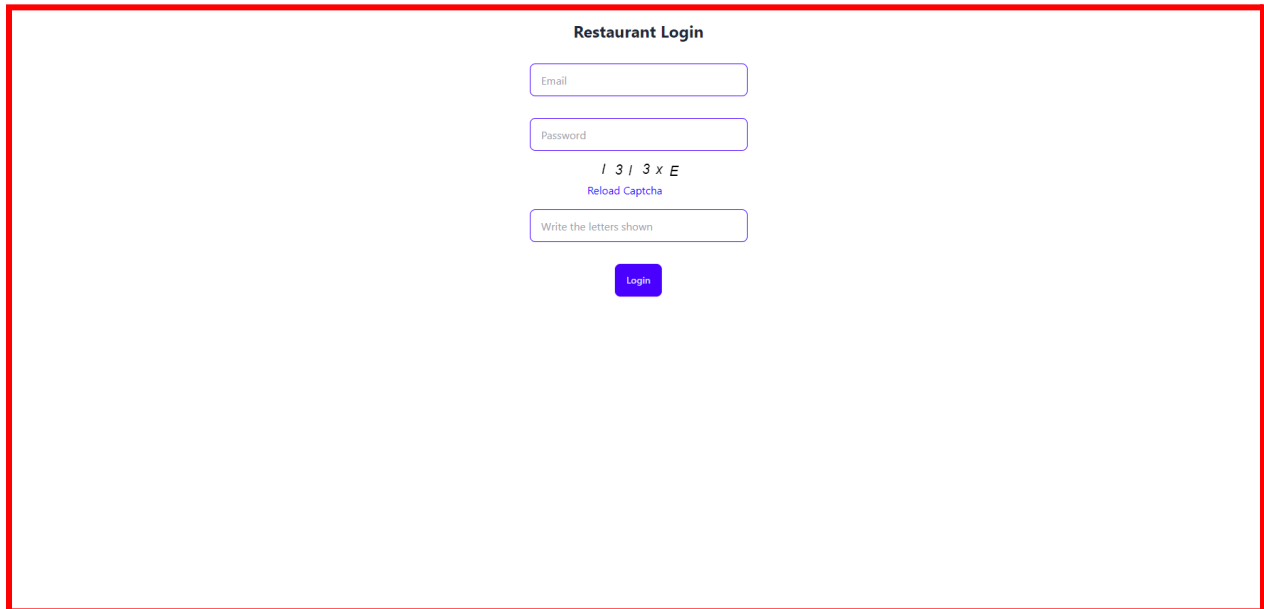
Write the letters shown

Login

Not a user yet? [Register](#)

Here the customer can log in if they have already created an account.

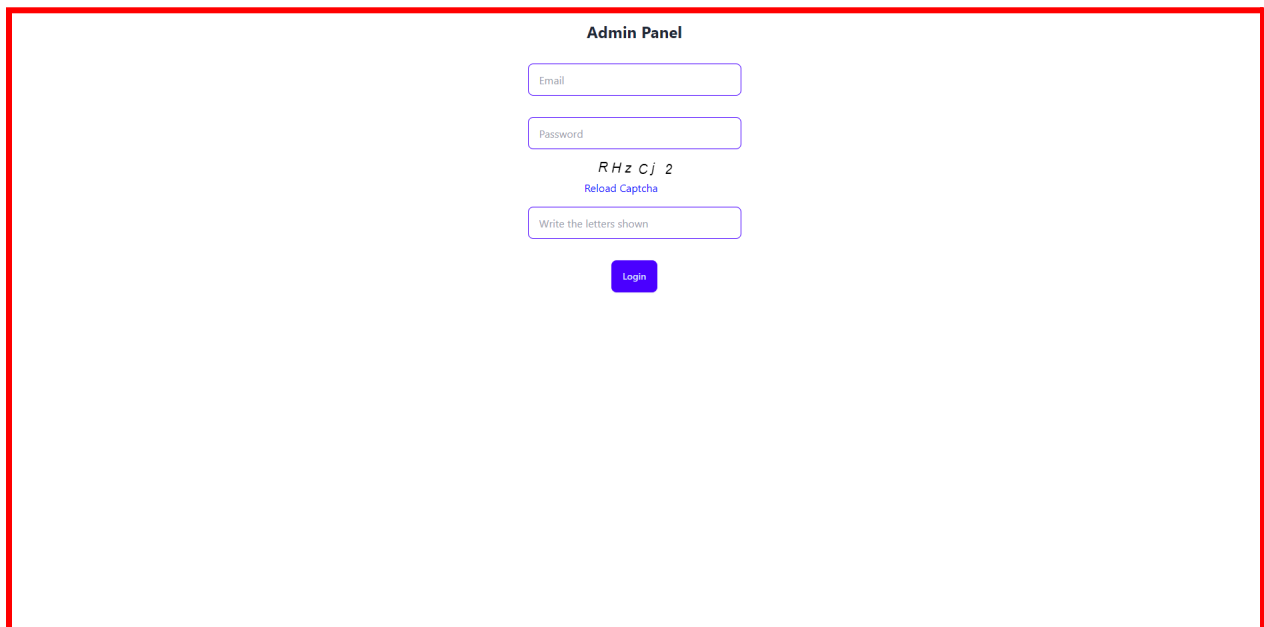
## <> Restaurant Login



The image shows a web form titled "Restaurant Login" centered on a light gray background. The form is enclosed in a red rectangular border. It contains the following elements from top to bottom: a title "Restaurant Login", an "Email" input field, a "Password" input field, a captcha image showing the text "I 3 | 3 x E", a "Reload Captcha" link in blue, a text input field with the placeholder "Write the letters shown", and a blue "Login" button.

This is the login page for the restaurant staff. The email and password will be given by the admin.

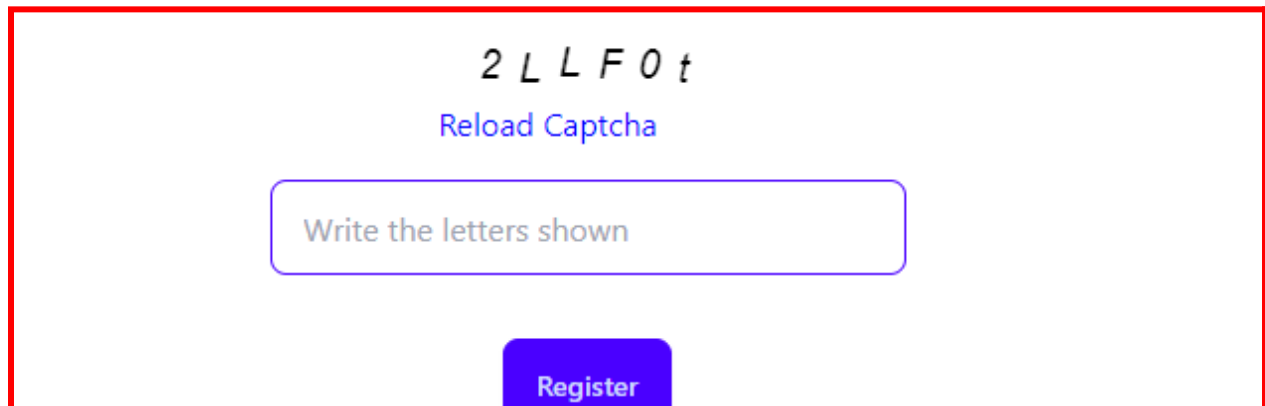
## <> Admin Login



The image shows a web form titled "Admin Panel" centered on a light gray background. The form is enclosed in a red rectangular border. It contains the following elements from top to bottom: a title "Admin Panel", an "Email" input field, a "Password" input field, a captcha image showing the text "R H z C j 2", a "Reload Captcha" link in blue, a text input field with the placeholder "Write the letters shown", and a blue "Login" button.

This is the admin login panel.

## <>Authentication (2FA)



2 L L F 0 t

[Reload Captcha](#)

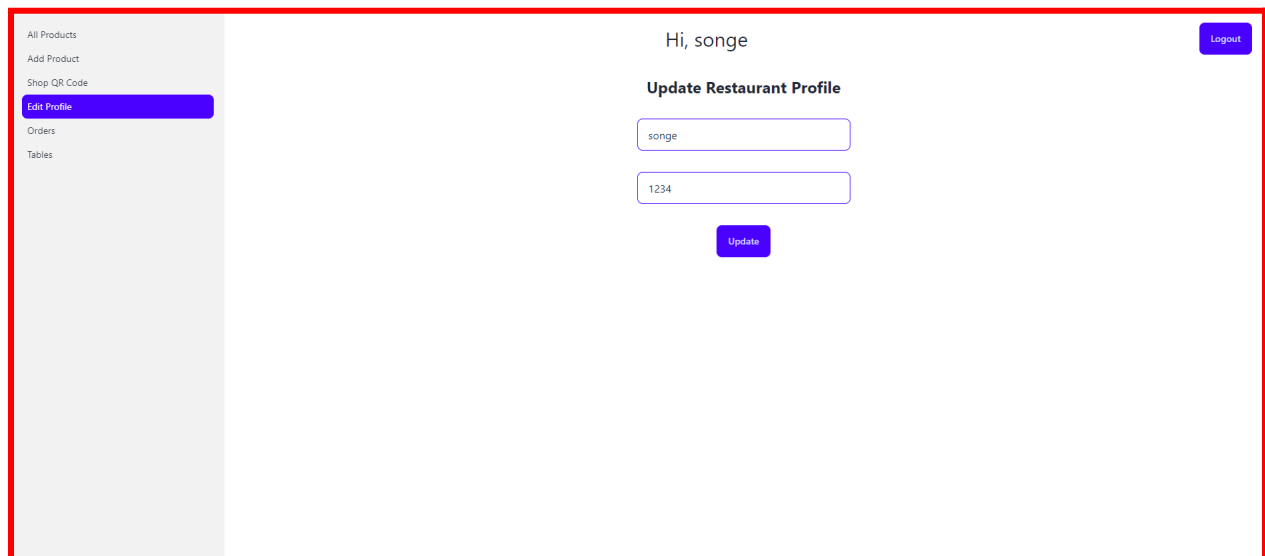
Write the letters shown

[Register](#)

This screenshot shows a 2FA authentication panel. At the top, the letters '2 L L F 0 t' are displayed in a stylized font. Below them is a blue link labeled 'Reload Captcha'. A light blue rounded rectangle contains the text 'Write the letters shown'. At the bottom center is a blue button labeled 'Register'.

This is the authentication panel which will appear during login and registration.

## <>.Restaurant account edit



All Products  
Add Product  
Shop QR Code  
**Edit Profile**  
Orders  
Tables

Hi, songe

[Logout](#)

**Update Restaurant Profile**

songe

1234

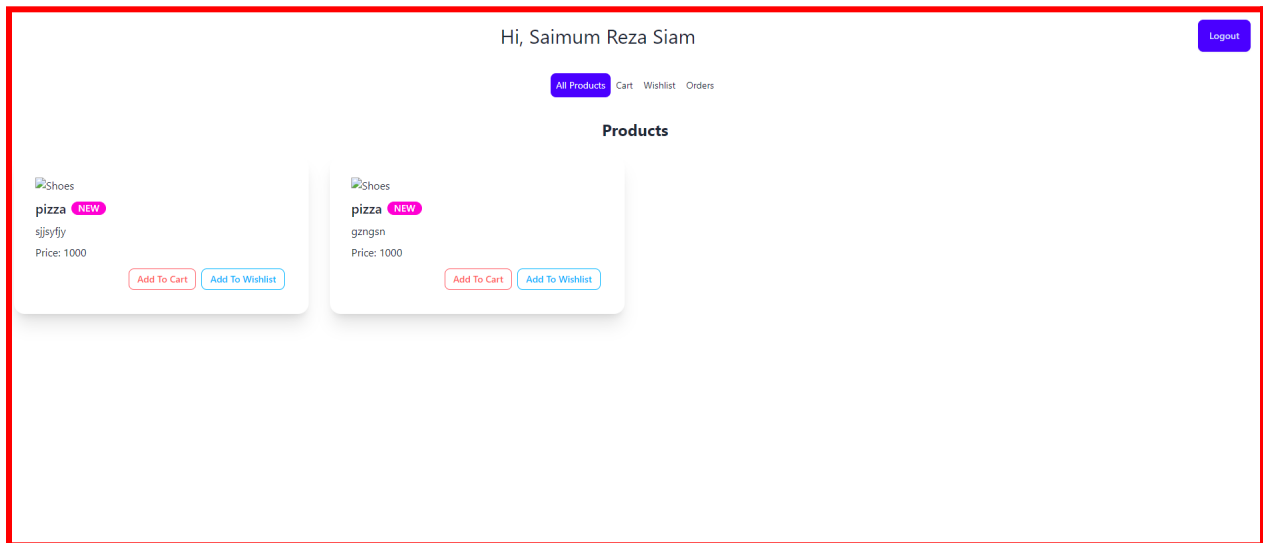
[Update](#)

This screenshot shows the 'Restaurant account edit' panel. On the left is a sidebar with a list of menu items: 'All Products', 'Add Product', 'Shop QR Code', 'Edit Profile' (highlighted in blue), 'Orders', and 'Tables'. The main content area has a header 'Hi, songe' and a 'Logout' button. Below this is the 'Update Restaurant Profile' section, which contains two input fields: the first contains 'songe' and the second contains '1234'. A blue 'Update' button is positioned below the second input field.

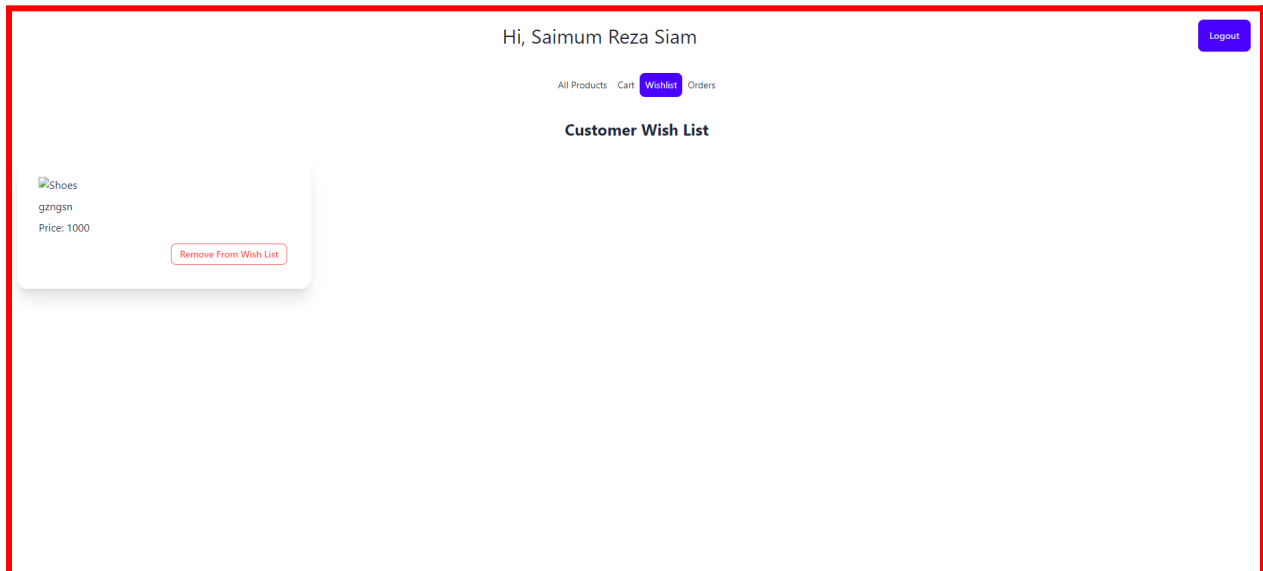
Here the restaurant can edit their restaurant name and also change their wifi password.



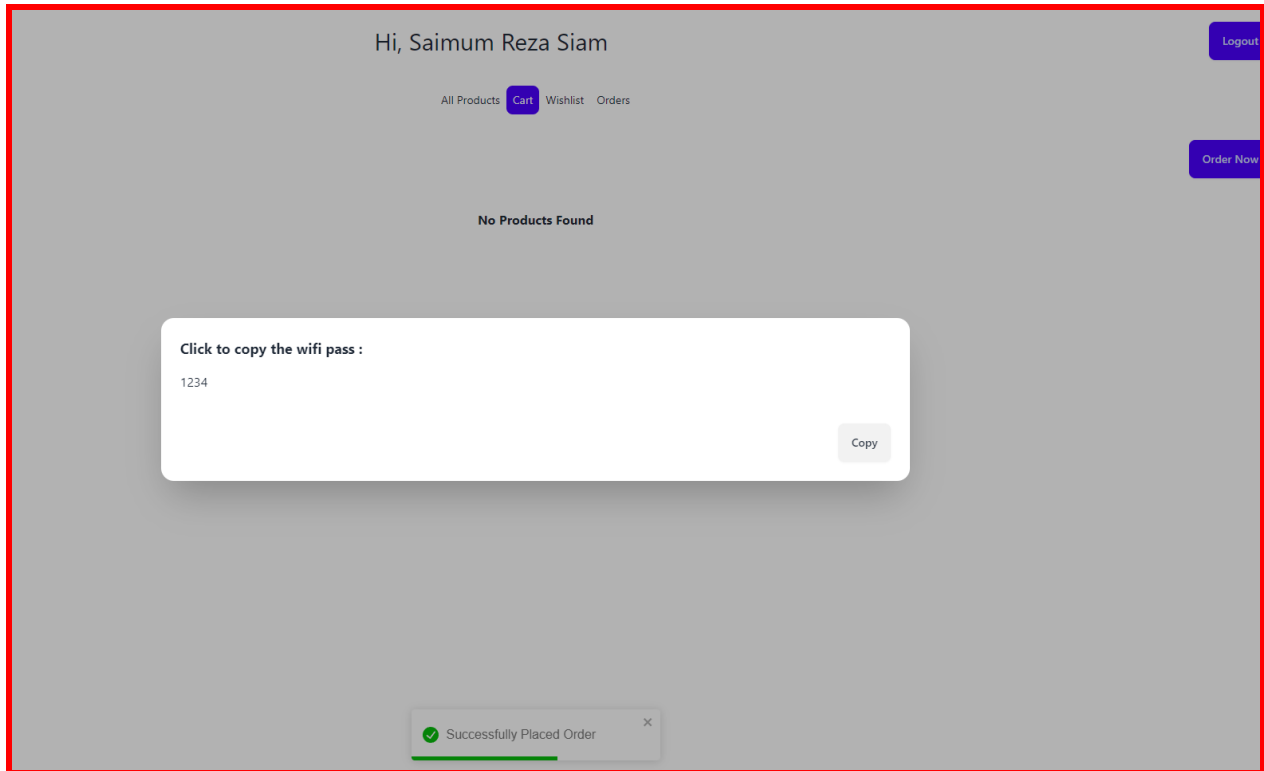
## <>Add to cart and Order and Wishlist



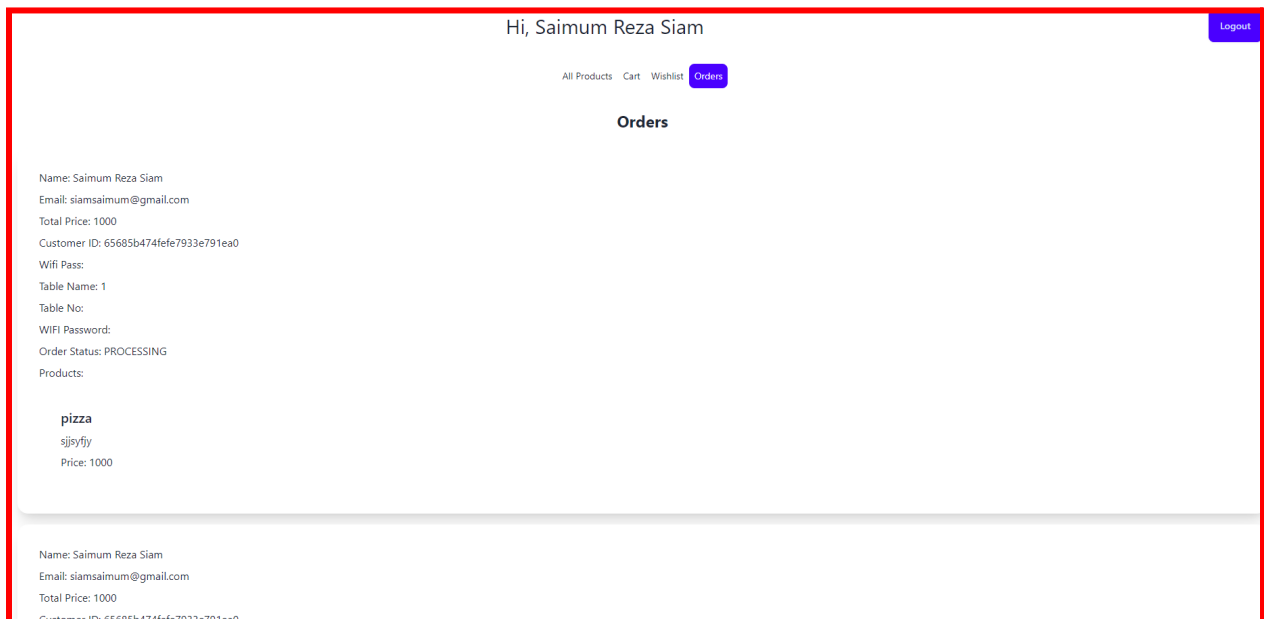
This is the customer dashboard where they can see the available food in that restaurant. They can add food to cart or wishlist them.



These are the cart and wishlist. In cart customer can select the available seat and then press order.



After putting an order, the customer will be given the wifi password. They can copy it and thus use it.



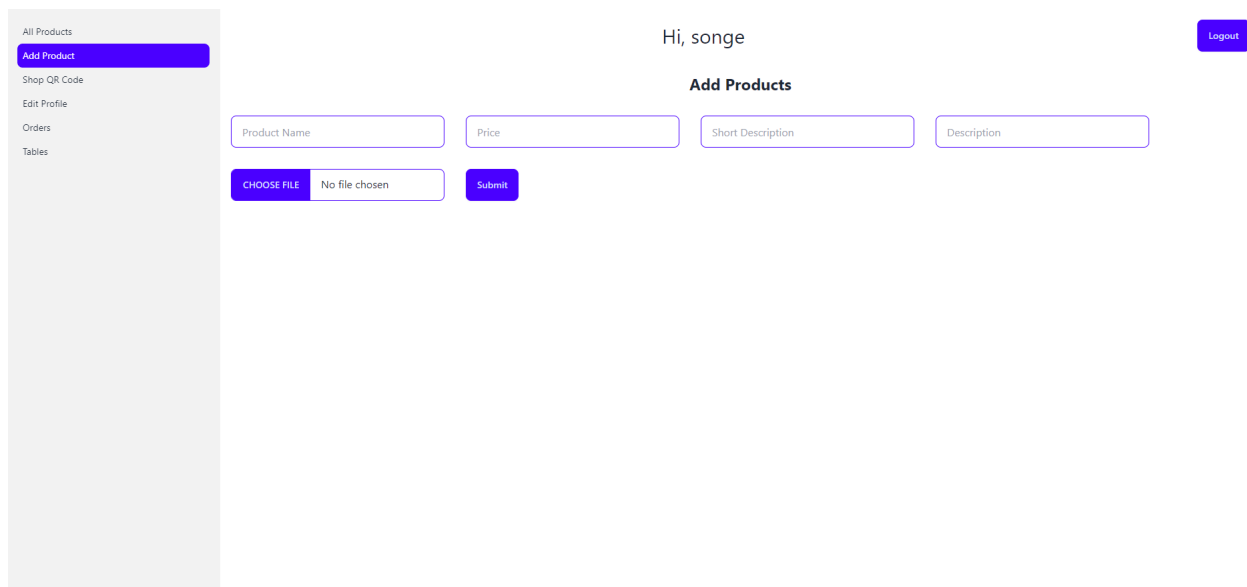
This here is the order log of customer. Here they can see the status of order.

## <>QR code scan



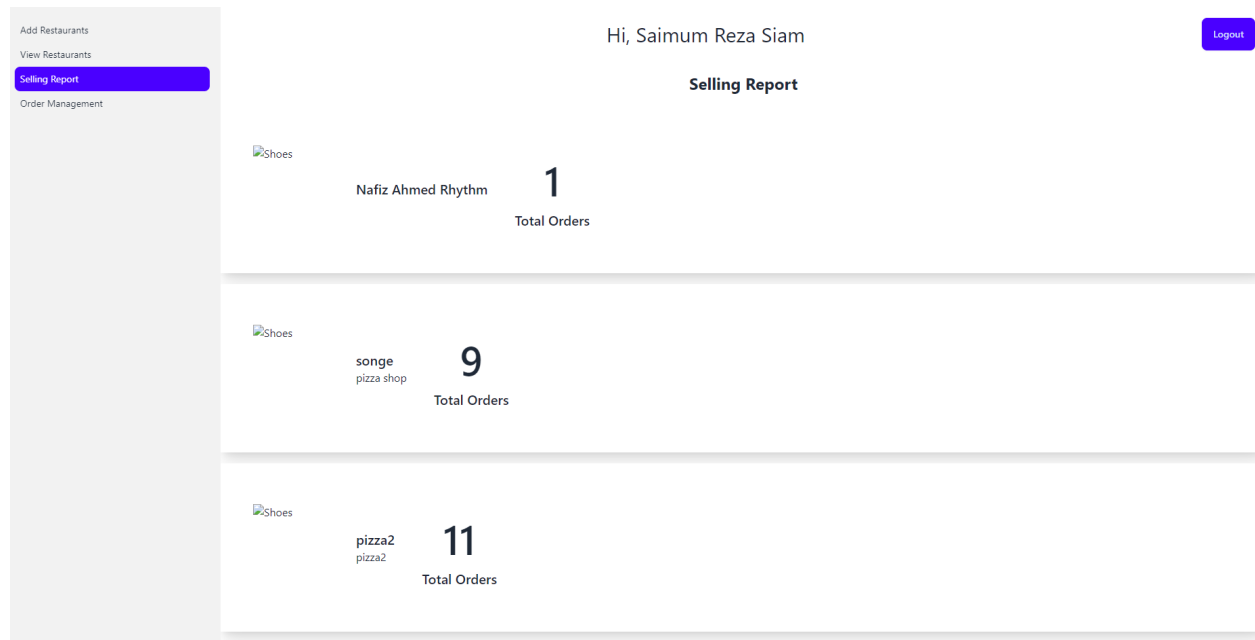
By using this QR code, the customer will be redirected to the menu page.

## <>Add food menu



Here restaurants can add food items to their menu.

## <>Selling report



This here is the selling report where the admin can see how much sell occurred.

## <>Order management

Hi, Saimum Reza Siam

**Orders**

Logout

Add Restaurants  
View Restaurants  
Selling Report  
**Order Management**

Name: Nafiz Ahmed Rhythm  
Email: bitencryptit@gmail.com  
Total Price: 1000  
Customer ID: 6563bc0ff7818e7826edfcc9  
Restaurant ID: cce002d7-174d-4d52-bed4-4f99b0c57f62  
Table Name:  
Table No:  
Order Status:  
Products:  
burger  
new  
Price: 1000  
PLACED ☒ PROCESSING ☐ PREPARING ☐ SERVED ☐ COMPLETE ☐

Name: Nafiz Ahmed Rhythm  
Email: bitencryptit@gmail.com  
Total Price: 1000  
Customer ID: 6563bc0ff7818e7826edfcc9  
Restaurant ID: cce002d7-174d-4d52-bed4-4f99b0c57f62  
Table Name:  
Table No:  
Order Status:  
Products:

The dashboard shows a sidebar with navigation options: 'Add Restaurants', 'View Restaurants', 'Selling Report', and 'Order Management' (highlighted). The main content area displays the user's name 'Hi, Saimum Reza Siam' and a 'Logout' button. Below this, the title 'Orders' is centered. The order details are presented in two identical blocks. Each block contains the following information: Name (Nafiz Ahmed Rhythm), Email (bitencryptit@gmail.com), Total Price (1000), Customer ID (6563bc0ff7818e7826edfcc9), Restaurant ID (cce002d7-174d-4d52-bed4-4f99b0c57f62), Table Name, Table No, Order Status, Products (burger, new), and Price (1000). At the bottom of each block, there is a row of radio buttons for the order status: PLACED (selected), PROCESSING, PREPARING, SERVED, and COMPLETE.

Here the admin can see the order report and the status.

## <>Add new Restaurant

The screenshot shows the 'Add Restaurants' form in the admin panel. The left sidebar contains links: 'Add Restaurants' (highlighted), 'View Restaurants', 'Selling Report', and 'Order Management'. The top right shows the user 'Hi, Saimum Reza Siam' and a 'Logout' button. The form title is 'Add Restaurants'. It includes input fields for 'Restaurant Name', 'Email', 'Password', 'Short Description', and 'Description'. There is a file upload section with a 'CHOOSE FILE' button, the text 'No file chosen', and a 'Submit' button.

The admin can add new restaurants with this panel.

## <>Can active and deactivate a restaurant account

The screenshot shows the 'Restaurants' list in the admin panel. The left sidebar is the same as the previous screenshot. The top right shows the user 'Hi, Saimum Reza Siam' and a 'Logout' button. The form title is 'Restaurants'. The list displays three restaurant entries, each with a profile picture placeholder (labeled 'Shoes'), the restaurant name, and action buttons: 'Deactivate', 'Edit', and 'Delete'. The first entry is 'Nafiz Ahmed Rhythm'. The second entry is 'songe pizza shop' and has an 'Activate' button. The third entry is 'pizza2 pizza2'.

The admin can activate or deactivate a restaurant from this panel. They Can also edit and delete a restaurant from this panel.

## <>Set status of order

All Products

Add Product

Shop QR Code

Edit Profile

**Orders**

Tables

Hi, songe

Logout

**Orders**

Name: raehr

Email: saimumsiam01@gmail.com

Total Price: 1000

Customer ID: 65685bf24fef7933e791eeb

Restaurant ID: 3be8bb90-db80-4649-9fa3-5699627cd73b

Table Name:

Table No:

Order Status: PLACED

Products:

pizza

sjsyfyj

Price: 1000

PLACED

☒

PROCESSING

☐

PREPARING

☐

SERVED

☐

COMPLETE

☐

Name: raehr

Email: saimumsiam01@gmail.com

Total Price: 1000

Customer ID: 65685bf24fef7933e791eeb

Restaurant ID: 3be8bb90-db80-4649-9fa3-5699627cd73b

Table Name:

Table No:

Order Status:

Products:

Here the restaurant can set the status of food. If they tap complete, that table will become available.

## <>Add/remove tables

All Products

Add Product

Shop QR Code

Edit Profile

Orders

**Tables**

Hi, songe

Logout

**Tables**

**Add Tables**

Name

Table No

Submit

1

2

3

Here the restaurant can add or remove tables.

# Frontend & Backend Development

## Login/Register

Frontend:

```
return (
  <div className='flex flex-col items-center justify-center'>

    <h1 className="m-4 text-2xl text-center font-bold">{
      _isAdmin ?
        "Admin Panel" :
        _isCustomer ?
          isLogin ? "Customer Login" : "Customer Registration" :
          "Restaurant Login"
    }</h1>

    {!isLogin ?
      <input
        className="input m-4 input-bordered input-primary w-full max-w-xs"
        type="text"
        placeholder={_isCustomer || _isAdmin ? `Name` : `Restaurant Name`}
        name="name"
        value={formData.name}
        onChange={handleInputChange}
      /> : null
    }

    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Email"
      name="email"
      value={formData.email}
      onChange={handleInputChange}
    />

    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Password"
      name="password"
      value={formData.password}
      onChange={handleInputChange}
    />

    <LoadCanvasTemplate />

    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Write the letters shown"
      name="captcha"
      value={formData.captcha}
      onChange={handleInputChange}
    />

    <button className="btn m-4 btn-primary" onClick={handleSubmit}>{isLogin ? "Login" :
    <button className="m-4" onClick={() => {
      setIsLogin(!isLogin);
    }}>
      {_isCustomer ?
        isLogin ?
```



## Admin Backend:

```
router.post('/login-admin', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password } = req.body;
    var restaurant = await Admin.findOne({ email: email });
    if (restaurant) {
      if (restaurant.password == password) {
        res.status(201).json(restaurant);
        return;
      } else {
        res.status(403).json({ "message": "Wrong Password" });
        return;
      }
    } else {
      res.status(403).json({ "message": "Not signed up" });
      return;
    }
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error logging in' });
    return;
  }
});
```

## Customer Backend:

```
});

// Handling POST request for logging in consumers
router.post('/login-consumer', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password } = req.body;
    var restaurant = await Customer.findOne({ email: email });
    if (restaurant) {
      if (restaurant.password == password) {
        res.status(201).json(restaurant);
        return;
      } else {
        res.status(403).json({ "message": "Wrong Password" });
        return;
      }
    } else {
      res.status(403).json({ "message": "Not signed up" });
      return;
    }
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error logging in' });
    return;
  }
});

// Handling POST request for signing up consumers
router.post('/signup-consumer', async (req, res) => {
  await mongoConnect();
  try {
    const { name, email, password } = req.body;
    var restaurant = await Customer.findOne({ email: email });
    if (restaurant) {
      res.status(403).json({ "message": "Already Registered. Please Login." });
      return;
    } else {
      const idName = uuidv4();
      restaurant = new Customer({
        name: name,
        email: email,
        password: password,
        restaurantId: idName,
      });
      await restaurant.save();
      res.status(201).json(restaurant);
      return;
    }
  } catch (error) {
    console.log(error);
    res.status(403).json({ message: 'Error signing up' });
    return;
  }
});

// Handling POST request for getting consumers profiles
router.post('/profile-consumer', async (req, res) => {
```

# Restaurant Management

## Add Restaurant

### Frontend:

```
src > components > admin > AddRestaurant.jsx > AddRestaurant > handleSubmit
1  import React, { useRef, useState } from 'react';
2  import { toast } from 'react-toastify';
3  import { addRestaurant } from '../../services/apiService';
4
5
6  const AddRestaurant = () => {
7
8      const filePickerRef = useRef(null);
9
10     const [formData, setFormData] = useState({
11         name: '',
12         email: '',
13         password: '',
14         imageFile: null,
15         shortDescription: '',
16         description: ''
17     });
18
19     const handleInputChange = (e) => {
20         setFormData({ ...formData, [e.target.name]: e.target.value });
21     };
22
23     const handleFileChange = (e) => {
24         setFormData({ ...formData, imageFile: e.target.files[0] });
25     };
26
27     const handleSubmit = async (e) => {
28         e.preventDefault();
29         if (formData.name == "" || formData.price == "" || formData.imageFile == null) {
30             toast.error("Name, Price and Image Cannot Be Empty");
31             return;
32         }
33         try {
34             const response = await addRestaurant(
35                 formData.name,
36                 formData.email,
37                 formData.password,
38                 formData.imageFile,
39                 formData.shortDescription,
40                 formData.description,
41             );
42             if (response.data) {
43                 setFormData({
44                     name: '',
45                     email: '',
46                     password: '',
47                     imageFile: null,
48                     shortDescription: '',
49                     description: ''
50                 });
51                 filePickerRef.current.value = null;
52                 toast.success("Successfully added restaurant.");
53             } else {
54                 toast.error("Failed to add restaurant.");
55             }
56         } catch (error) {
57             toast.error("Failed to add product.");
58             console.error('Error creating product:', error);
59         }
60     };
61 }
```

```

return (
  <form onSubmit={handleSubmit}>
    <h1 className="m-4 text-2xl text-center font-bold">Add Restaurants</h1>
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Restaurant Name"
      name="name"
      value={formData.name}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Email"
      name="email"
      value={formData.email}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Password"
      name="password"
      value={formData.password}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Short Description"
      name="shortDescription"
      value={formData.shortDescription}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Description"
      name="description"
      value={formData.description}
      onChange={handleInputChange}
    />
    <input ref={filePickerRef} className="file-input m-4 file-input-bordered file-input" type="text" />
    <button className="btn m-4 btn-primary" type="submit">Submit</button>
  </form>
);
};

export default AddRestaurant;

```

## Backend:

```
// Handling POST request for adding restaurants
router.post('/add-restaurant', upload.single('imageFile'), async (req, res) => {
  await mongoConnect();
  try {
    const { name, email, password, shortDescription, description } = req.body;
    var imagePath = req.file.filename; // Path to the uploaded file

    var restaurantId = uuidv4();

    const newProduct = new Restaurant({
      name,
      email,
      password,
      restaurantId,
      imagePath,
      shortDescription,
      description,
      isActive: true,
    });

    await newProduct.save();

    res.status(201).json(newProduct);
    return;
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error creating product' });
    return;
  }
});
```

## Manage Restaurant

### Frontend:

```
import React, { useRef, useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { toast } from 'react-toastify';
import { FILES_BASE_URL } from '../../../utils/constants';
import { activeOrInactiveRest, deleteRestaurant, updateRestaurant } from '../../../services/apiSe
import { setRestaurantsData } from '../../../redux/homeSlice';

const getProducts = state => state.home.products;
const getRestaurants = state => state.home.restaurants;
const getIsConsumer = state => state.auth.isCustomer;
const getUserData = state => state.auth;

const Restaurants = () => {
  const dispatch = useDispatch();

  const products = useSelector(getProducts);
  const restaurants = useSelector(getRestaurants);
  const isConsumer = useSelector(getIsConsumer);
  const userData = useSelector(getUserData);

  const [formData, setFormData] = useState(null);
  const filePickerRef = useRef(null);

  const handleInputChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleFileChange = (e) => {
    setFormData({ ...formData, imageFile: e.target.files[0] });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (formData.name == "" || formData.price == "") {
      toast.error("Name, Price and Image Cannot Be Empty");
      return;
    }
    try {
      const response = await updateRestaurant(
        formData._id,
        formData.name,
        formData.email,
        formData.password,
        formData.imageFile,
        formData.shortDescription,
        formData.description,
      );
      if (response.data) {
        dispatch(setRestaurantsData(response.data));
        setFormData(null);
        toast.success("Successfully saved restaurant.");
      } else {
        toast.error("Failed to add restaurant.");
      }
    } catch (error) {
      toast.error("Failed to add product.");
      console.error('Error creating product:', error);
    }
  }
}
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF

```

| h-full'>
| text-2xl text-center font-bold'>Restaurants</h2>
:h != 0 ?
:'space-y-4'>
its.map((product) => (
:y={product._id} className="bg-base-100 shadow-xl p-4">
liv className="card-body flex flex-row items-center">
  <figure>
  | <img className='w-32 h-32' src={`$${FILES_BASE_URL}/${product.imagePath}`} alt="Shoes" />
  </figure>
  <div className='flex flex-col m-4'>
  | <h2 className="card-title">
  |   {product.name}
  | </h2>
  | <p>{product.shortDescription}</p>
  </div>
  {isConsumer ?
  | <div className="flex justify-end">
  |   <button
  |     onClick={async (c_event) => {
  |       c_event.preventDefault();
  |       // Add To Cart logic
  |     }}
  |     className="m-1 btn btn-sm btn-outline btn-error">Add To Cart</button>
  |   <button
  |     onClick={async (c_event) => {
  |       c_event.preventDefault();
  |       // Add To Wishlist logic
  |     }}
  |     className="m-1 btn btn-sm btn-outline btn-info"> Add To Wishlist</button>
  | </div> :
  | <div className="flex justify-end">
  |   <button
  |     onClick={async (c_event) => {
  |       c_event.preventDefault();
  |       var resp = await activeOrInactiveRest(product._id);
  |       if (resp) {
  |         dispatch(setRestaurantsData(resp.data));
  |         // toast.success("Success");
  |       } else {
  |         toast.error("Failed to delete");
  |       }
  |     }}
  |     className={product.isActive ? "m-1 btn btn-sm btn-outline btn-error" :
  |       "m-1 btn btn-sm btn-outline btn-success"}>
  |     {product.isActive ? "Deactivate" : "Activate"}</button>
  |   <button
  |     onClick={async (c_event) => {
  |       c_event.preventDefault();
  |       setFormData(product);
  |     }}
  |     className="m-1 btn btn-sm btn-outline btn-info">Edit</button>
  |   <button
  |     onClick={async (c_event) => {
  |       c_event.preventDefault();
  |       // Delete logic

```

Ln 26, Col 7 Spaces: 4 UTF-8 LF

## Edit Restaurant

### Frontend:

```
    <div className="w-full h-1/2 flex items-center justify-center text-center">
      <div className="max-w-md">
        <h1 className="text-1xl font-bold">No Products Found</h1>
      </div>
    </div>
  }
</div> :
<div className='w-full h-full'>
  <form onSubmit={handleSubmit}>
    <h1 className="m-4 text-2xl text-center font-bold">Edit Restaurant</h1>
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Restaurant Name"
      name="name"
      value={formData.name}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Email"
      name="email"
      value={formData.email}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Password"
      name="password"
      value={formData.password}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Short Description"
      name="shortDescription"
      value={formData.shortDescription}
      onChange={handleInputChange}
    />
    <input
      className="input m-4 input-bordered input-primary w-full max-w-xs"
      type="text"
      placeholder="Description"
      name="description"
      value={formData.description}
      onChange={handleInputChange}
    />
    <input ref={filePickerRef} className="file-input m-4 file-input-bordered file-input-p
    <button className="btn m-4 btn-primary" type="submit">Submit</button>
    <button className="btn m-4 btn-error text-white" onClick={()=>setFormData(null)}>Canc
  </form>
</div>
```

Ln 88, Col 67 Spaces: 4 UTF-8 LF



## Backend:

```

// Handling POST request for updating restaurants
router.post('/update-restaurant', upload.single('imageFile'), async (req, res) => {
  await mongoConnect();
  try {
    const { id, name, email, password, shortDescription, description } = req.body;

    var imagePath = null

    if (req.file) {
      imagePath = req.file.filename; // Path to the uploaded file
    }

    const product = await Restaurant.findById(id);

    product.name = name;
    product.email = email;
    product.password = password;
    product.shortDescription = shortDescription;
    product.description = description;
    if (imagePath) {
      product.imagePath = imagePath;
    }
    await product.save();

    const products = await Restaurant.find();

    res.status(201).json(products);
    return;
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error creating product' });
    return;
  }
});

// Handling DELETE request for deleting restaurants
router.delete('/restaurants', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.id) {
      const product = await Restaurant.findById(req.query.id);
      if (product) {
        await product.deleteOne();
        res.status(201).json({ message: 'Success' });
        return;
      }
    }
    res.status(500).json({ message: 'Error deleting restaurants' });
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error deleting restaurants' });
  }
});

// Handling GET request for getting restaurants
router.get('/restaurants', async (req, res) => {
  await mongoConnect();

```

```

// Handling GET request for getting restaurants
router.get('/restaurants', async (req, res) => {
  await mongoConnect();
  try {
    const products = await Restaurant.find();
    res.json(products);
  } catch (error) {
    res.status(500).json({ message: 'Error fetching orders' });
  }
});

// Handling POST request for setting restaurants active
router.post('/restaurants-set-active', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.id) {
      const product = await Restaurant.findById(req.query.id);
      if (product) {
        if (product.isActive) {
          product.isActive = false;
          await product.save();
        } else {
          product.isActive = true;
          await product.save();
        }
        const products = await Restaurant.find();
        res.status(201).json(products);
        return;
      }
    }
    res.status(500).json({ message: 'Error deleting restaurants' });
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error deleting restaurants' });
  }
});

// Handling POST request for setting restaurants active
router.post('/update-order-status', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.id) {
      const item = await Order.findById(req.query.id);
      if (item) {
        item.orderStatus = req.query.status;
        await item.save();
        res.status(201).json(item);
        return;
      }
    }
    res.status(500).json({ message: 'Error deleting restaurants' });
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error deleting restaurants' });
  }
});

router.post('/res-tables', async (req, res) => {
  await mongoConnect();

```

## Selling Report

### Frontend:

```
components > admin > SellingReport.jsx > ...

const getOrders = state => state.home.orderItems;
const getRestaurants = state => state.home.restaurants;
const getIsConsumer = state => state.auth.isCustomer;
const getUserData = state => state.auth;

const SellingReport = () => {
  const dispatch = useDispatch();

  const orders = useSelector(getOrders);
  const restaurants = useSelector(getRestaurants);
  const isConsumer = useSelector(getIsConsumer);
  const userData = useSelector(getUserData);

  console.log(orders)

  return (
    <div className='w-full h-full'>
      <h2 className='m-4 text-2xl text-center font-bold'>Selling Report</h2>
      {restaurants.length !== 0 ?
        <ul className='space-y-4'>
          {restaurants.map((product) => (
            <li key={product._id} className='bg-base-100 shadow-xl p-4'>
              <div className='card-body flex flex-row items-center'>
                <figure>
                  <img className='w-32 h-32' src={`/${FILES_BASE_URL}/${product.image}`} alt="Product Image" />
                </figure>
                <div className='flex flex-col m-4'>
                  <h2 className='card-title'>
                    {product.name}
                  </h2>
                  <p>{product.shortDescription}</p>
                </div>
                <div className='flex flex-col m-4 justify-center items-center'>
                  <h2 className='card-title text-6xl m-4'>
                    {orders ? orders.filter(item => item.restaurantId === product._id).length : 0}
                  </h2>
                  <h2 className='card-title'>
                    Total Orders
                  </h2>
                </div>
              </div>
            </li>
          ))}
        </ul>
        :
        <div className='w-full h-1/2 flex items-center justify-center text-center'>
          <div className='max-w-md'>
            <h1 className='text-1xl font-bold'>No Products Found</h1>
          </div>
        </div>
      }
    </div>
  );
};

export default SellingReport;
```

# Order Management

## Frontend:

```
components > admin > OrdersAdmin.jsx > ...
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import OrderItem from './OrderItem';

const getOrders = state => state.home.orderItems;
const getIsConsumer = state => state.auth.isCustomer;
const getUserData = state => state.auth;

const OrdersAdmin = () => {
  const dispatch = useDispatch();

  const orders = useSelector(getOrders);
  const isConsumer = useSelector(getIsConsumer);
  const userData = useSelector(getUserData);

  return (
    <div className='w-full h-full'>
      <h2 className='m-4 text-2xl text-center font-bold'>Orders</h2>
      {orders.length !== 0 ? (
        <ul className='flex flex-col'>
          {orders.map(order => (
            <OrderItem order={order} key={order._id} />
          ))}
        </ul>
      ) : (
        <div className="w-full h-1/2 flex items-center justify-center text-center">
          <div className="max-w-md">
            <h1 className="text-1xl font-bold">No Orders Found</h1>
          </div>
        </div>
      )}
    </div>
  );
};

export default OrdersAdmin;
```

## Backend:

```
// Handling POST request for setting restaurants active
router.post('/update-order-status', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.id) {
      const item = await Order.findById(req.query.id);
      if (item) {
        item.orderStatus = req.query.status;
        await item.save();
        res.status(201).json(item);
        return;
      }
    }
    res.status(500).json({ message: 'Error deleting restaurants' });
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error deleting restaurants' });
  }
});
```

# CONSUMER

## Cart Page

### Frontend:

```
    }}>Order Now</button>
  </div>
  Select Table :
  {tables ? (
    <div className="grid grid-cols-3 gap-4">
      {tables.map((tableItem, index) => (
        <button
          key={index}
          onClick={() => handleButtonClick(tableItem)}
          className={`btn ${selectedOption === tableItem ? 'btn-primary' : 'btn-secondary'} ${tableItem.name}`}
        >
        </button>
      ))}
    </div>
  ) : null}
  {products.length !== 0 ?
    <ul>
      <div className="grid grid-cols-1 gap-4 md:grid-cols-2 lg:grid-cols-4">
        {products.map((product) => (
          <div key={product._id} className="card w-auto bg-base-100 shadow-xl">
            <div className="card-body">
              <figure>
                <img className="w-full" src={`/${FILES_BASE_URL}/${product.image}`} alt="Product Image" />
                <p>{product.shortDescription}</p>
                <p>Price: {product.price}</p>
                <div className="flex flex-row justify-end">
                  <button
                    onClick={async (e) => {
                      e.preventDefault();
                      var resp = await removeProductFromCart(userData._id, product._id);
                      if (resp) {
                        dispatch(setCartProductsData(resp.data));
                        toast.success("Added To Cart");
                      } else {
                        toast.error("Failed to add to cart");
                      }
                    }}
                    className="m-1 btn btn-sm btn-outline btn-error">Remove
                  </button>
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
    </ul> :
    <div className="w-full h-1/2 flex items-center justify-center text-center">
      <div className="max-w-md">
        <h1 className="text-1xl font-bold">No Products Found</h1>
      </div>
    </div>
  )}
</div>
);
};

export default CartPage;
```

## Backend:

```
    res.status(500).json({ message: 'Error getting products from cart' });
  }
});

// Handling POST request for adding a product to the cart
router.post('/addCartProduct', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password, productId } = req.body;
    const products = await Customer.findOne({ email: email, password: password });
    if (products) {
      if (!products.cart.includes(productId)) {
        products.cart.push(productId);
        await products.save();
      }
      const productIds = products.cart.map(product => new ObjectId(product));
      const productsInCart = await Product.find({ _id: { $in: productIds } });
      res.status(200).json(productsInCart);
    } else {
      res.status(500).json({ message: 'Error getting product' });
    }
    return;
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error getting product' });
    return;
  }
});

// Handling POST request for removing a product from the cart
router.post('/removeCartProduct', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password, productId } = req.body;
    const customer = await Customer.findOne({ email: email, password: password });
    if (customer) {
      customer.cart = customer.cart.filter(product => product !== productId);
      await customer.save();
      const productIds = customer.cart.map(product => new ObjectId(product));
      const productsInCart = await Product.find({ _id: { $in: productIds } });
      res.status(200).json(productsInCart);
    } else {
      res.status(404).json({ message: 'Customer not found' });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error removing product from cart' });
  }
});

// Handling POST request for adding a product to the wishlist
router.post('/wishListProducts', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password } = req.body;
    const customer = await Customer.findOne({ email: email, password: password });
    if (customer) {
```

# WISHLIST

## Frontend:

```
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { addProductToCart, removeProductFromCart, removeProductFromWishlist } from '../services/api';
import { setCartProductsData, setWishlistProductsData } from '../redux/homeSlice';
import { FILES_BASE_URL } from '../utils/constants';

const getCartProducts = state => state.home.wishListItems;
const getUserData = state => state.auth;

const WishListPage = () => {
  const dispatch = useDispatch();

  const products = useSelector(getCartProducts);
  const userData = useSelector(getUserData);

  return (
    <div className='w-full h-full'>
      <h2 className='m-4 text-2xl text-center font-bold'>Customer Wish List</h2>
      {products.length !== 0 ?
        <ul>
          <div className='grid grid-cols-1 gap-4 md:grid-cols-2 lg:grid-cols-4'>
            {products.map((product) => (
              <div key={product._id} className="card w-auto bg-base-100 shadow-xl">
                <div className="card-body">
                  <figure>
                    <img className='w-full' src={`${FILES_BASE_URL}/${product.image}`/}>
                    <p>{product.shortDescription}</p>
                    <p>Price: {product.price}</p>
                    <div className="flex flex-row justify-end">
                      <button
                        onClick={async (e) => {
                          e.preventDefault();
                          var resp = await removeProductFromWishlist(userData._id, product._id);
                          if (resp) {
                            dispatch(setWishlistProductsData(resp.data));
                            toast.success("Removed From Wishlist");
                          } else {
                            toast.error("Failed to remove from wish list");
                          }
                        }}
                        className="m-1 btn btn-sm btn-outline btn-error">Remove
                      </button>
                    </div>
                  </figure>
                </div>
              </div>
            ))}
          </div>
        </ul> :
        <div className="w-full h-1/2 flex items-center justify-center text-center">
          <div className="max-w-md">
            <h1 className="text-1xl font-bold">No Products Found</h1>
          </div>
        </div>
      }
    </div>
  );
};
```



## Backend:

```
// Handling POST request for adding a product to the wishlist
router.post('/wishlistProducts', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password } = req.body;
    const customer = await Customer.findOne({ email: email, password: password });
    if (customer) {
      const productIds = customer.wishlist.map(product => new ObjectId(product));
      const productsInCart = await Product.find({ _id: { $in: productIds } });
      res.status(200).json(productsInCart);
    } else {
      res.status(404).json({ message: 'Customer not found' });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error getting products from cart' });
  }
});
```

```
    console.error(error);
    res.status(500).json({ message: 'Error placing order' });
  }
});

// Handling POST request for adding a product to the wishlist
router.post('/addWishlistProduct', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password, productId } = req.body;
    const products = await Customer.findOne({ email: email, password: password });
    if (products) {
      if (!products.wishlist.includes(productId)) {
        products.wishlist.push(productId);
        await products.save();
      }
      const productIds = products.wishlist.map(product => new ObjectId(product));
      const productsInCart = await Product.find({ _id: { $in: productIds } });
      res.status(200).json(productsInCart);
    } else {
      res.status(500).json({ message: 'Error getting product' });
    }
    return;
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error getting product' });
    return;
  }
});

// Handling POST request for removing a product from the wishlist
router.post('/removeWishlistProduct', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password, productId } = req.body;
    const customer = await Customer.findOne({ email: email, password: password });
    if (customer) {
      customer.wishlist = customer.wishlist.filter(product => product !== productId);
      await customer.save();
      const productIds = customer.wishlist.map(product => new ObjectId(product));
      const productsInCart = await Product.find({ _id: { $in: productIds } });
      res.status(200).json(productsInCart);
    } else {
      res.status(404).json({ message: 'Customer not found' });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error removing product from cart' });
  }
});

// Handling POST request for logging in
router.post('/login', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password } = req.body;
    var restaurant = await Restaurant.findOne({ email: email });
    if (restaurant) {
```

## Products Page

### Frontend:

```
<h2 className="card-title">
  {product.name}
  <div className="badge badge-secondary">NEW</div>
</h2>
<p>{product.shortDescription}</p>
<p>Price: {product.price}</p>
{isConsumer ?
  <div className="flex flex-row justify-end">
    <button
      onClick={async (c_event) => {
        c_event.preventDefault();
        var resp = await addProductToCart(userData.email, userData.password, p
        if (resp) {
          dispatch(setCartProductsData(resp.data));
          toast.success("Added To Cart");
        } else {
          toast.error("Failed to add to cart");
        }
      }}
      className="m-1 btn btn-sm btn-outline btn-error">Add To Cart</button>
    <button
      onClick={async (c_event) => {
        c_event.preventDefault();
        var resp = await addProductToWishlist(userData.email, userData.password
        if (resp) {
          dispatch(setWishlistProductsData(resp.data));
          toast.success("Added To Wishlist");
        } else {
          toast.error("Failed to add to Wishlist");
        }
      }}
      className="m-1 btn btn-sm btn-outline btn-info"> Add To Wishlist</button>
    </div> :
    <div className="flex flex-row justify-end">
      <button
        onClick={async (c_event) => {
          c_event.preventDefault();
          var resp = await deleteProduct(product._id);
          if (resp) {
            dispatch(setProductsData(products.filter(value => value != product
            toast.success("Deleted product");
          } else {
            toast.error("Failed to delete");
          }
        }}
        className="m-1 btn btn-sm btn-outline btn-error">Delete</button>
      </* <button className="m-1 btn btn-sm btn-outline btn-info">Edit</button> */>
    </div>
  </div>
</div>

me="w-full h-1/2 flex items-center justify-center text-center">
ssName="max-w-md">
className="text-1xl font-bold">No Products Found</h1>
```

## Backend:

```
});

// Handling GET request for getting restaurant products
router.get('/products', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.restaurantId) {
      const products = await Product.find({ restaurantId: req.query.restaurantId });
      res.json(products);
      return;
    } else {
      const products = await Product.find();
      res.json(products);
      return;
    }
  } catch (error) {
    res.status(500).json({ message: 'Error fetching products' });
  }
});

// Handling POST request for adding products
router.post('/products', upload.single('imageFile'), async (req, res) => {
  await mongoConnect();
  try {
    const { name, price, restaurantId, shortDescription, description } = req.body;
    var imagePath = req.file.filename; // Path to the uploaded file

    const restaurant = await Restaurant.findOne({ restaurantId: restaurantId });

    if (restaurant.isActive !== true) {
      res.status(500).json({ message: 'Restaurant Is Deactivated' });
      return;
    }

    const newProduct = new Product({
      name,
      price,
      restaurantId,
      imagePath,
      shortDescription,
      description
    });

    await newProduct.save();

    res.status(201).json(newProduct);
    return;
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error creating product' });
    return;
  }
});

// Handling GET request for getting orders
router.get('/orders', async (req, res) => {
```

## Place order

### Backend:

```
Handling POST request for adding a order
router.post('/placeOrder', async (req, res) => {
  await mongoConnect();
  try {
    const { email, password, restaurantId, totalPrice, tableName, tableId } = req.body;
    const customer = await Customer.findOne({ email: email, password: password });
    const restaurant = await Restaurant.findOne({ restaurantId: restaurantId });

    if (customer) {
      const productIds = customer.cart.map(product => new ObjectId(product));
      const productsInWishlist = await Product.find({ _id: { $in: productIds } });

      // Prepare order details based on wishlist products
      const orderProducts = productsInWishlist.map(product => ({
        name: product.name,
        imagePath: product.imagePath,
        price: product.price,
        shortDescription: product.shortDescription,
        description: product.description,
      }));

      const totalPrice = productsInWishlist.reduce((total, product) => total + parseFloat(product.price), 0).toString();

      // Create the order
      const newOrder = new Order({
        name: customer.name,
        phone: '',
        email: customer.email,
        restaurantId: restaurantId,
        totalPrice: totalPrice,
        customerId: customer._id, // Assuming customer._id is the MongoDB ObjectId
        products: orderProducts,
        tableName: tableName,
        tableId: tableId,
        wifiPass: restaurant.wifiPass,
      });

      // Save the order to the database
      await newOrder.save();

      // Clear the wishlist after placing the order
      customer.cart = [];
      await customer.save();

      res.status(200).json({ message: 'Order placed successfully', order: newOrder });
    } else {
      res.status(404).json({ message: 'Customer not found' });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error placing order' });
  }
});
```

# Restaurant

## Add Product

### Frontend:

```
const AddProduct = () => {  
  
  const filePickerRef = useRef(null);  
  
  const userData = useSelector(getUserData);  
  
  const [formData, setFormData] = useState({  
    name: '',  
    price: '',  
    imageFile: null,  
    shortDescription: '',  
    description: ''  
  });  
  
  const handleInputChange = (e) => {  
    setFormData({ ...formData, [e.target.name]: e.target.value });  
  };  
  
  const handleFileChange = (e) => {  
    setFormData({ ...formData, imageFile: e.target.files[0] });  
  };  
  
  const handleSubmit = async (e) => {  
    e.preventDefault();  
    if (formData.name == "" || formData.price == "" || formData.imageFile == null) {  
      toast.error("Name, Price and Image Cannot Be Empty");  
      return;  
    }  
    try {  
      const response = await addProduct(  
        formData.name,  
        formData.price,  
        userData.restaurantId,  
        formData.imageFile,  
        formData.shortDescription,  
        formData.description,  
      );  
      if (response.data) {  
        setFormData({  
          name: '',  
          price: '',  
          restaurantId: '',  
          imageFile: null,  
          shortDescription: '',  
          description: ''  
        });  
        filePickerRef.current.value = null;  
        toast.success("Successfully added product.");  
      } else {  
        toast.error(response.error.message);  
      }  
    } catch (error) {  
      toast.error("Failed to add product.");  
      console.error('Error creating product:', error);  
    }  
  };  
}
```

## Backend:

```
// Handling POST request for adding products
router.post('/products', upload.single('imageFile'), async (req, res) => {
  await mongoConnect();
  try {
    const { name, price, restaurantId, shortDescription, description } = req.body;
    var imagePath = req.file.filename; // Path to the uploaded file

    const restaurant = await Restaurant.findOne({ restaurantId: restaurantId });

    if (restaurant.isActive !== true) {
      res.status(500).json({ message: 'Restaurant Is Deactivated' });
      return;
    }

    const newProduct = new Product({
      name,
      price,
      restaurantId,
      imagePath,
      shortDescription,
      description
    });

    await newProduct.save();

    res.status(201).json(newProduct);
    return;
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error creating product' });
    return;
  }
});
```

## Add/Edit tables

### Add Frontend:

```
const AddTable = () => {

  const dispatch = useDispatch();

  const userData = useSelector(getUserData);

  const [formData, setFormData] = useState({
    name: '',
    num: '',
  });

  const handleInputChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (formData.name == "" && formData.num == "") {
      toast.error("Name Cannot Be Empty");
      return;
    }
    try {
      const response = await addRestaurantTable(
        userData.restaurantId,
        formData.name,
        formData.num,
      );
      if (response.data) {
        setFormData({
          name: '',
          num: '',
        });
        dispatch(setRestaurantsTables(response.data.tables));
        toast.success("Successfully added table.");
      } else {
        toast.error(response.error.message);
      }
    } catch (error) {
      toast.error("Failed to add table.");
      console.error("Error creating table:", error);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <h1 className="m-4 text-2xl text-center font-bold">Add Tables</h1>
      <input
        className="input m-4 input-bordered input-primary w-full max-w-xs"
        type="text"
        placeholder="Name"
        name="name"
        value={formData.name}
        onChange={handleInputChange}
      />
      <input
```

## Edit Frontend:

```
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import AddTable from './AddTable';
import { deleteTables } from '../../../services/apiService';
import { setRestaurantsTables } from '../../../redux/homeSlice';
import { toast } from 'react-toastify';

const getOrders = state => state.home.orderItems;
const getTables = state => state.home.tables;
const getUserData = state => state.auth;

const TableList = () => {
  const dispatch = useDispatch();

  const orders = useSelector(getOrders);
  const tables = useSelector(getTables);
  const userData = useSelector(getUserData);

  return (
    <div className='w-full h-full'>
      <h2 className='m-4 text-2xl text-center font-bold'>Tables</h2>
      <AddTable />
      {tables.length !== 0 ?
        <ul>
          <div className='grid grid-cols-1 gap-4 md:grid-cols-2 lg:grid-cols-4'>
            {
              tables.map((product) => (
                <button
                  key={product.name}
                  className={orders.some(item => item.tableName === product.name
                    && item.restaurantId === userData.restaurantId
                    && item.orderStatus !== "COMPLETE") ?
                    "btn btn-error" :
                    "btn btn-outline btn-error"}
                  onClick={async () => {
                    const confirmed = window.confirm('Are you sure you want to delete this item?')
                    if (confirmed) {
                      var resp = await deleteTables(userData.restaurantId, product.name, product)
                      if (resp.data) {
                        toast.success("Deleted");
                        dispatch(setRestaurantsTables(resp.data));
                      } else {
                        toast.error("Failed to delete");
                      }
                    }
                  }}
                  >{product.name}
                </button>
              ))
            }
          </div>
        </ul>
        :
        <div className="w-full h-1/2 flex items-center justify-center text-center">
          <div className="max-w-md">
            <h1 className="text-1xl font-bold">No Tables Found</h1>
          </div>
        </div>
      )
    </div>
  );
};

export default TableList;
```



## Backend:

```
router.post('/res-tables', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.id) {
      const item = await Restaurant.findOne({
        "restaurantId": req.query.id
      });
      if (item) {
        if (!item.tables.includes({ name: req.query.name, num: req.query.num })) {
          item.tables.push({ name: req.query.name, num: req.query.num });
        }
        await item.save();
        res.status(201).json(item);
        return;
      }
    }
    res.status(500).json({ message: 'Error tables' });
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error tables' });
  }
});

router.delete('/delete-tables', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.id) {
      const item = await Restaurant.findOne({
        "restaurantId": req.query.id
      });
      if (item) {
        var tables = item.tables;
        const index = tables.findIndex(table => table.name === req.query.name && table.num === req.query.num);
        if (index > -1) {
          tables.splice(index, 1);
        }
        item.tables = tables;
        await item.save();
        res.status(201).json(tables);
        return;
      }
    }
    res.status(500).json({ message: 'Error deleting tables' });
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: 'Error deleting tables' });
  }
});

router.get('/tables', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.restaurantId) {
      const item = await Restaurant.findOne({ restaurantId: req.query.restaurantId });
      if (item) {
        var tables = item.tables;
        const orders = await Order.find({ restaurantId: req.query.restaurantId });
        orders.forEach(oItm => {
          if (oItm.orderStatus !== "COMPLETE") {
            var index = tables.findIndex(item => item.name === oItm.tableName && item.num === oItm.tableId);
            if (index > -1) {
              tables.splice(index, 1);
            }
          }
        });
        res.status(201).json(tables);
        return;
      }
    }
  }
});
```

## QR Code Page

### Frontend:

```
import React from 'react';
import QRCode from 'react-qr-code';
import { useSelector } from 'react-redux';

const getRestaurantId = state => state.auth.restaurantId;

const QRCodePage = () => {

  const _restaurantId = useSelector(getRestaurantId);

  return (
    <div>
      <h2 className='m-4 text-3xl text-center font-bold'>Restaurant QR Code</h2>
      <h2 className='m-4 text-2xl text-center font-bold'>Scan to view all the products</h2>
      <div className='flex flex-col justify-center items-center m-28'>
        <QRCode value={`${window.location.protocol + "://" + window.location.host}?restaurantId=${_restaurantId}&v=${_restaurantId}`} />
        <a href={`${window.location.protocol + "://" + window.location.host}?restaurantId=${_restaurantId}&v=${_restaurantId}`}>View Products</a>
      </div>
    </div>
  );
};

export default QRCodePage;
```

## Edit Profile/Wifi Pass

### Frontend:

```
import { updateUser } from '../../services/apiService';
import { toast } from 'react-toastify';

const getUserData = state => state.auth;

const EditProfile = () => {
  const dispatch = useDispatch();

  const userData = useSelector(getUserData);

  const [formData, setFormData] = useState({
    name: userData.name,
    wifiPass: userData.wifiPass,
  });

  const handleInputChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (formData.name == "" || formData.wifiPass == "") {
      toast.error("Restaurant Name Cannot Be Empty");
      return;
    }
    const response = await updateUser(userData.email, userData.password, formData.name, formData.wifiPass);
    if (response.data) {
      localStorage.setItem("name", response.data.name);
      localStorage.setItem("wifiPass", response.data.wifiPass);
      dispatch(updateLoginState({
        loggedIn: true,
        email: response.data.email,
        password: response.data.password,
        name: response.data.name,
        restaurantId: response.data.restaurantId,
        wifiPass: response.data.wifiPass
      }));
      toast.success("Successfully Updated Profile");
    } else {
      toast.error(response.error.message);
    }
  };

  return (
    <div className='flex flex-col items-center justify-center'>
      <h1 className="m-4 text-2xl text-center font-bold">Update Restaurant Profile</h1>
      <input
        className="input m-4 input-bordered input-primary w-full max-w-xs"
        type="text"
        placeholder="Restaurant Name"
        name="name"
        value={formData.name}
        onChange={handleInputChange}
      />
      <input
        className="input m-4 input-bordered input-primary w-full max-w-xs"
        type="text"
        placeholder="Wifi Password"
        name="wifiPass"
        value={formData.wifiPass}
        onChange={handleInputChange}
      />
      <button className="btn m-4 btn-primary" onClick={handleSubmit}>Update</button>
    </div>
  );
};

export default EditProfile;
```

## Backend:

```
406
407 // Handling POST request for getting restaurant profile
408 router.post('/profile', async (req, res) => {
409     await mongoConnect();
410     try {
411         const { name, email, password, wifiPass } = req.body;
412         var restaurant = await Restaurant.findOne({ email: email, password: password });
413         if (restaurant) {
414             restaurant.name = name;
415             restaurant.wifiPass = wifiPass;
416             await restaurant.save();
417             res.status(201).json(restaurant);
418             return;
419         } else {
420             res.status(403).json({ message: 'Error updating' });
421             return;
422         }
423     } catch (error) {
424         console.log(error);
425         res.status(403).json({ message: 'Error signing up' });
426         return;
427     }
428 });
```

# Order Management

## Frontend:

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { FILES_BASE_URL } from '../utils/constants';
import { addProductToCart, addProductToWishlist, deleteProduct } from '../services/apiService';
import { setCartProductsData, setProductsData, setWishlistProductsData } from '../redux/homeSlice';
import { toast } from 'react-toastify';
import OrderItem from '../admin/OrderItem';

const getOrders = state => state.home.orderItems;
const getIsConsumer = state => state.auth.isCustomer;
const getUserData = state => state.auth;
const getTables = state => state.home.tables;

const OrderList = () => {
  const dispatch = useDispatch();

  const orders = useSelector(getOrders);
  const isConsumer = useSelector(getIsConsumer);
  const userData = useSelector(getUserData);
  const tables = useSelector(getTables);

  const [selectedOption, setSelectedOption] = useState(""); // Set initial selected value
  const [filteredOrders, setFilteredOrders] = useState([]); // Set initial selected value

  const handleButtonClick = (tableItem) => {
    if (selectedOption === tableItem) {
      setSelectedOption(""); // Deselect if already
      setFilteredOrders(orders);
    } else {
      setSelectedOption(tableItem);
      const filtered = orders.filter(order => order.tableId === tableItem.num);
      setFilteredOrders(filtered);
    }
  };

  return (
    <div className='w-full h-full'>
      <h2 className='m-4 text-2xl text-center font-bold'>Orders</h2>
      Select Table :
      <div className="overflow-x-auto whitespace-no-wrap">
        <div className="flex">
          {tables ? (
            tables.map((tableItem, index) => (
              <button
                key={index}
                onClick={() => handleButtonClick(tableItem)}
                className={selectedOption === tableItem ? 'btn-primary' : 'btn-secondary'} mx
              >
                {tableItem.name}
              </button>
            ))
          ) : (
            <p>No tables available</p>
          )}
        </div>
      </div>
      {filteredOrders.length !== 0 ? (
        <ul className='flex flex-col'>
          {filteredOrders.map(order => (
            isConsumer ? <div key={order._id} className="card w-auto bg-base-100 shadow-xl m-2">
              <div className="card-body">
                <p>Name: {order.name}</p>
                <p>Email: {order.email}</p>
                /* <p>Restaurant ID: {order.restaurantId}</p> */
                <p>Total Price: {order.totalPrice}</p>
                <p>Customer ID: {order.customerId}</p>
                <p>Wifi Pass: {order.wifiPass}</p>
                <p>Table Name: {order.tableName}</p>
                <p>Table No: {order.tableId}</p>
                <p>Wifi Password: {order.wifiPass}</p>
              </div>
            </div>
          ))}
        </ul>
      ) : (
        <p>No orders available</p>
      )}
    </div>
  );
};
```

```

    }
  };

  return (
    <div className='w-full h-full'>
      <h2 className='m-4 text-2xl text-center font-bold'>Orders</h2>
      Select Table :
      <div className="overflow-x-auto whitespace-no-wrap">
        <div className="flex">
          {tables ? (
            tables.map((tableItem, index) => (
              <button
                key={index}
                onClick={() => handleButtonClick(tableItem)}
                className={`btn ${selectedOption === tableItem ? 'btn-primary' : 'btn-secondary'} mx-2`}>
                  {tableItem.name}
                </button>
              )
            ) : (
              <p>No tables available</p>
            )
          )}
        </div>
      </div>
      {filteredOrders.length !== 0 ? (
        <ul className='flex flex-col'>
          {filteredOrders.map(order => (
            isConsumer ? <div key={order._id} className="card w-auto bg-base-100 shadow-xl m-2">
              <div className="card-body">
                <p>Name: {order.name}</p>
                <p>Email: {order.email}</p>
                <p>Restaurant ID: {order.restaurantId}</p>
                <p>Total Price: {order.totalPrice}</p>
                <p>Customer ID: {order.customerId}</p>
                <p>Wifi Pass: {order.wifiPass}</p>
                <p>Table Name: {order.tableName}</p>
                <p>Table No: {order.tableId}</p>
                <p>WIFI Password: {order.wifiPass}</p>
                <p>Order Status: {order.orderStatus}</p>
                <h3>Products:</h3>
                <ul>
                  {order.products.map(product => (
                    <li key={product._id}>
                      <div className="card">
                        <div className="card-body">
                          <h4 className="card-title">{product.name}</h4>
                          <p>{product.shortDescription}</p>
                          <p>Price: {product.price}</p>
                        </div>
                      </div>
                    </li>
                  )
                )}
                </ul>
              </div>
            </div> :
              <OrderItem order={order} key={order._id} />
            )
          )}
        </ul>
      ) : (
        <div className="w-full h-1/2 flex items-center justify-center text-center">
          <div className="max-w-md">
            <h1 className="text-1xl font-bold">No Orders Found</h1>
          </div>
        </div>
      )
    )
  );
};

```

## Backend:

```
42  ter.post('/update-order-status', async (req, res) => {
43    await mongoConnect();
44    try {
45      if (req.query.id) {
46        const item = await Order.findById(req.query.id);
47        if (item) {
48          item.orderStatus = req.query.status;
49          await item.save();
50          res.status(201).json(item);
51          return;
52        }
53      }
54      res.status(500).json({ message: 'Error deleting restaurants' });
55    } catch (error) {
56      console.log(error);
57      res.status(500).json({ message: 'Error deleting restaurants' });
58    }
59  }
```

```
Handling GET request for getting orders
ter.get('/orders', async (req, res) => {
  await mongoConnect();
  try {
    if (req.query.restaurantId) {
      const products = await Order.find({ restaurantId: req.query.restaurantId });
      res.json(products);
      return;
    } else {
      const products = await Order.find();
      res.json(products);
      return;
    }
  } catch (error) {
    res.status(500).json({ message: 'Error fetching orders' });
  }
}
```

## Technology (Framework, Languages)

Framework: MERN

Language: Javascript, React, MongoDB, Express, NodeJs

## Github Repository

Link: <https://github.com/rhythm6677/khawahobe.git>

## Individual Contribution

ID	Name	Contribution
21101164	Saimum Reza Siam	Module-3 and Module-4 (contributed in all modules)
21301581	Nafiz Ahmed Rhythm	Module-1 and Module-2 (contributed in all modules)