**RESTOURANT DATABASE**

INTERNSHIP PROJECT REPORT

Submitted to

Zephyr Technologies & Solutions Pvt. Ltd.

5th Floor, Oberle Towers, Balmatta Rd, Bendoor, Mangalore, Karnataka - 575002, India



Submitted By

Sumanth USN: NNS22BC058

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Computer Application

# Dr N.S.A.M First Grade College, Nitte

## ACKNOWLEDGEMENT

We dedicate this page to acknowledge and thank those responsible for the shaping of the project. Without their guidance and help, the experience while constructing the dissertation would not have been so smooth and efficient.

We would like to take the opportunity to express our humble gratitude to the employees

of Zephyr Technologies under whom we executed this project. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depth and helped us to complete the assigned tasks.

Finally, yet importantly, we would like to express our heartfelt thanks to our friends and

classmates for their help and wishes for the successful completion of the project.

## ABOUT THE COMPANY

ZEPHYR TECHNOLOGIES is a software company delivering high-quality, cost-effective reliable result-oriented web and e-commerce solutions on time for a global clientele. professionalism, skill, and expertise are the tools we use to make the web work for your business bringing in maximum return on your investment in the shortest possible time. we have delivered on IT projects of varying complexities for their very demanding and internet clients spread across the globe. They develop unique web solutions that ensure increased efficiency and competitive advantage for your business and thus to your end-users. Their tools are professionalism, skills, and expertise that translate into delivering quality work at every step for any project we undertake. they work towards getting better than the best out of every team member at ZEPHYR TECHNOLOGIES, which means when you hire them all-around quality is assured as you want it. Their advantage quality includes the protection of intellectual for the source codes developed specifically for your business. they do not sell the source codes to third parties and all elements that they create for your web solution belong to you. ZEPHYR TECHNOLOGIES project managers and business analysts place great value on building a clean communication link with you as they consider it the key ingredient for the success of any project at hand.

## ABSTRACT

The main aim and objective of this project were to design and develop a comprehensive web application for managing a restaurant's operations, named "Foodiifyy." Leveraging the MERN stack (MongoDB, Express.js, React.js, Node.js),. The core functionality of "Foodiifyy" revolves around maintaining a detailed and organized database of menu items, customer orders, and user accounts, providing an interactive and user-friendly interface for both customers and administrators.

Through the use of modern web development technologies, the project ensures seamless data management, secure authentication, and dynamic content delivery. MongoDB serves as the backend database, ensuring efficient data storage and retrieval, while Express.js and Node.js handle the server-side logic and API development. React.js powers the frontend, offering a responsive and engaging user experience. The implementation of "Foodiifyy" exemplifies how the MERN stack can be utilized to develop a fully functional, efficient, and maintainable web application tailored to the needs of a restaurant management system.

## TABLE OF CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
| **SI.No** | **Contents** | **Page no.** |  |
| 1.  1.1  1.2  1.3  1.4  1.5  1.6  1.7 | INTRODUCTION  Aim  Objectives  Scope  Problem Statement  Advantages/Disadvantages  Technology Used  System Requirements | 1  1  1  1  2  2  2 |
| 2. | OBJECTIVES | 3 |
| 3. | DATA TABLES | 6 |
| 4. | IMPLEMENTATION |  |
| 5. | SOURCE CODES |  |
| 6. | SNAPSHOTS |  |
| 7. | CONCLUSION |  |
|  |  |  | |

## INTRODUCTION

We have developed a Foodify Restourant Database to get all the information about a restaurant menus in a single place. This system is useful in keeping a record of people and their order. The information is maintained by a single person hence the data is safe. By using this system, a person can get every information about menus orders.

### 1.1 Aim

The main aim of designing this project is to make a convenient website where a person can find information about restaurant menus.

### 1.2 Objectives

The main objective is to keep a record of menu items, customer orders, reviews, and restaurant details on a single website**.**

### 1.3 Scope

The website Foodify! Restourant will be used for maintaining the records of menu items, customer information, and orders in an organized manner. Updating and modifications will be easily achievable using this system.

### 1.4 Problem Definition

There are numerous restaurants offering a wide variety of cuisines and menu items. It's often challenging for customers to browse through the menu, place orders, and find specific dishes based on their preferences or dietary requirements.

### 1.5ADVANTAGES/DISADVANTAGES

#### 1.5.1 Advantages

* Customers can browse and search for dishes as per their preferences.
* Addition, deletion, and modification of menu items can be done easily.
* Users can find all relevant information about the restaurant in one place.
* Manage the entire order process.
* User-friendly error messages are provided wherever necessary.

#### 1.5.2 Disadvantages

* Security limitations.
* Only a single user (Admin) can modify the menu and other critical data.

**1.6 Technology Used:**

* **Frontend**: React.js
* **Backend**: Node.js, Express.js
* **Database**: MongoDB

**1.7 System Requirements:**

To develop and run the "Foodiifyy" restaurant management system using the MERN stack, the system requirements are as follows:

**Development Environment:**

* **RAM**: Minimum 8 GB
* **Hard Disk**: Minimum 100 GB of free space
* **Processor**: Intel i5 or equivalent
* **Operating System**: Windows 10, macOS, or a Linux distribution **Production Environment (Server):**
* **RAM**: Minimum 4 GB
* **Hard Disk**: Minimum 40 GB of free space (depending on the expected data size, you might need more)
* **Processor**: Intel i3 or equivalent
* **Operating System**: Linux distribution (Ubuntu 18.04 or higher, CentOS 7 or higher) for better performance and stability in production environments **Additional Software Requirements:**
* **Node.js**: Version 14.x or higher
* **npm** (Node Package Manager): Version 6.x or higher
* **MongoDB**: Version 4.x or higher  **React.js**: Latest stable version
* **Express.js**: Latest stable version
* **Browser**: Latest version of Google Chrome, Firefox, or any modern web browser for testing

These requirements ensure that you have a stable and efficient environment for both development and production of the "Foodiifyy" restaurant management system.

## 

## OBJECTIVES

* **Admin Control**: The Admin has complete control over the system, ensuring that only appropriate and valid data is provided and accepted.
* **Purpose**: The basic purpose of designing this project is to keep a record of restaurant menu items, customer information, orders, reviews, and other relevant details in a single web application.
* **Record Keeping**: This project is designed to maintain records of menu items, including customer detailsand their feedback.
* **Data Management**: The system helps in maintaining and analyzing restaurant data in an interactive and efficient manner. This includes managing menu updates, tracking customer orders, and generating reports for better decision-making.

These objectives provide a comprehensive framework for the "Foodiifyy" restaurant management system, ensuring it is user-friendly, efficient, and capable of handling all necessary operations related to running a restaurant.

## DATA TABLES

**Collection:users**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Field Name** | **Data Type** |  | **Description** |
| 1 | \_id | ObjectID |  | Stores user ID (automatically generated by MongoDB) |
| 2 | username | String |  | Stores user name |
| 3 | password | String |  | Stores password entered by user (hashed) |
| 4 | fullname | String |  | Stores full name of user |
| 5 | email | String |  | Stores email address of user |
| 6 | Status | String |  | Stores status of logged in |
|  | **Fig.** | **5.1 User data table** |  | |

**Menu:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Field Name** | **Data Type** | **Description** |
| 1 | \_id | ObjectId | Stores message ID (automatically generated by  MongoDB) |
| 2 | name | ObjectId | Stores the name of the user |
| 3 | email | String | Stores the email address of the user |
| 4 | message | String | Stores the content of the message |
| 5 | order | Array | Stores an array of items ordered, each containing item ID and quantity |
| 6 | date | Date | Stores the date and time when the message was sent |
| 7 | \_\_v | Number | Version key (automatically managed by MongoDB) |

**Fig. 5.1 User Menu Data**

**IMPLEMENTATION**

**Visual Studio Code:** Visual Studio Code, also commonly referred to as VS Code, is a

sourcecode editor by Microsoft for Windows Features include support for debugging.

**HTML:** HTML (Hypertext Markup Language) implementation involves using a text editor to create and structure the content of web pages. It consists of various elements and tags that define the layout, text, links, images, and other media on a webpage. To implement HTML, developers typically start by choosing a suitable text editor, such as Visual Studio Code or Sublime Text. Once the editor is set up, a new HTML file with a ".html" extension is created.

**CSS:** CSS (Cascading Style Sheets) implementation involves using CSS rules to define the appearance, layout, and style of HTML elements within a web page. It complements HTML by enhancing the presentation of content and creating visually appealing and user-friendly websites.

**MERN Stack:**

To implement the "Foodiifyy" restaurant management system using the MERN stack (MongoDB, Express.js, React.js, Node.js), follow these steps:

1. **Set Up the Project**:
   * + Install Node.js and npm.
     + Set up a new React project using Create React App.  Set up an Express.js server.
2. **Database (MongoDB)**:
   * Install MongoDB and set up a database for storing user data, menu items, and orders.
   * Use Mongoose to define schemas and interact with MongoDB.

**Backend (Node.js and Express.js)**:

* + Create RESTful APIs to handle CRUD operations for menu items, user authentication, and order management.
  + Implement user authentication using JWT (JSON Web Tokens).

**Frontend (React.js)**:

Develop the user interface using React.js, including pages for the menu, login, register, and order

* + Integrate with the backend APIs to fetch and display data.

**Testing and Deployment**:

* + Test the application thoroughly to ensure all functionalities work as expected.
  + Deploy the application using a service like Heroku, AWS, or Vercel for the frontend and backend, and MongoDB Atlas for the database.

By following these steps, you can implement the "Foodiifyy" restaurant management system using the MERN stack, leveraging modern web development practices and tools for a robust and scalable solution.

## SOURCE CODES

## Frontend

## Register.jsx

import React, { useState } from "react";

import { Link, useNavigate } from "react-router-dom"; import axios from "axios"; import './Registerpage.css';

export default function Registerpage() { const [name, setName] = useState(""); const [email, setEmail] = useState(""); const [password, setPassword] = useState("");

const [confirmPassword, setConfirmPassword] = useState(""); // State for confirm password const navigate = useNavigate();

const RegisterUser = async (e) => { e.preventDefault();

if (password !== confirmPassword) {

alert("Passwords do not match. Please try again."); return;

} try {

const response = await axios.post('http://localhost:3000/api/v1/user/', { email: email, password: password, name: name

});

console.log(response); navigate('/login'); } catch (error) {

alert("An error occurred. Please try again.");

}

};

const handleCancel = () => { setName(""); setEmail(""); setPassword("");

setConfirmPassword(""); // Clear confirm password field

};

return (

<div className="sign-in">

<form>

<header className="header1">

<center>

<h1 className="h1">Sign in</h1>

<p className="p1">Please fill in this form to create an account</p>

</center>

</header>

<hr />

<div className="content">

<label>Name</label>

<input type="text" placeholder="Enter your Name" value={name} onChange={(e) => setName(e.target.value)} /> <label>Email</label>

<input type="email" placeholder="Enter your Email" value={email} onChange={(e) => setEmail(e.target.value)} /> <label>Password</label>

<input type="password" placeholder="Enter your password" value={password} onChange={(e) => setPassword(e.target.value)} />

<label>Confirm Password</label>

<input type="password" placeholder="Confirm your password" value={confirmPassword} onChange={(e) => setConfirmPassword(e.target.value)} /> </div>

<div className="div\_btn">

<button type="submit" onClick={handleCancel} className="btn1">Refresh</button>

<button className="btn1" onClick={RegisterUser} type="submit">Register</button>

</div>

<div className="login-link">

<center>Already have an account? <Link to='/login'>Login</Link></center> </div>

</form>

</div>

);

}

## Login.jsx

import React, { useState } from 'react' import axios from 'axios'

import { Link, useNavigate } from 'react-router-dom' import './Loginpage.css'

export default function Loginpage() {

const [email, setEmail] = useState("") const [password, setPassword] = useState("") const navigate = useNavigate() const LoginUser = async (e) => { e.preventDefault()

try {

const response = await axios.post('http://localhost:3000/api/v1/user/login', { email: email,

password: password

});

console.log(response.data.data)

localStorage.setItem('userLogged', JSON.stringify(response.data.data)) alert("Login successful") navigate('/home') } catch (error) {

alert("Invalid email or password, Please try again later!!")

}

}

const handleCancel = () => {

setEmail("") setPassword("") navigate(-1) // This will navigate back to the previous page }

return (

<div className='login sign-in'>

<form>

<header className="header1">

<center>

<h1 id="h1">Login</h1>

<p id="p1">Please login to existing account</p>

</center>

</header>

<hr />

<div className="content">

<label>Email:</label>

<input type="email" placeholder="Enter your Email:" value={email} onChange={(e) => setEmail(e.target.value)} /> <label>Password:</label>

<input type="password" placeholder="Enter your password" value={password} onChange={(e) => setPassword(e.target.value)} />

<div className="div\_btn">

<button type="button" onClick={handleCancel} className="btn1">Cancel</button> <button className="btn1" onClick={LoginUser} type="submit">Login</button><br

/><br />

</div>

<div className="login-link">

<center>Create a new account <Link to="/">Register</Link></center>

</div>

</div>

</form>

</div>

)

}

## Navbar.jsx

import React from "react";

import { Link, useLocation } from "react-router-dom"; import './Navbar.css';

import { Link as ScrollLink } from 'react-scroll';

import { FaHome, FaUtensils, FaShoppingCart } from 'react-icons/fa'; // Importing icons

export default function Navbar({ toggleCart, isCartOpen }) { const location = useLocation();

return (

<div className="nav-dev" id="home">

<nav className="nav">

<ul>

<li>

<ScrollLink id="link" to="home" smooth={true} duration={500} className="navlink">

<FaHome className="icon" />

</ScrollLink>

</li>

</ul> <ul>

<li>

<ScrollLink id="link" to="menu" smooth={true} duration={500} className="navlink">

<FaUtensils className="icon" />

</ScrollLink>

</li>

</ul>

<ul>

<li>

<Link onClick={toggleCart} className="nav-link">

<FaShoppingCart className="icon" />

</Link>

</li>

</ul>

</nav>

</div>

);

}

## Homepage.jsx

import React, { useState } from 'react';

import Navbar from '../../Components/Navbar/Navbar'; // Make sure the path to Navbar is correct import './Homepage.css';

import dessertImage from './images/img1.jpeg'; // Correct relative path to the image import images from './images/special\_combos.png';

import imag2Image from './images/img2.jpeg'; import imag3Image from './images/img3.jpeg';

import imag4Image from './images/img4.jpeg'; import imag5Image from './images/fish.jpg'; import imag6Image from './images/img6.jpeg'; import imag7Image from './images/mommos.avif'; import imag8Image from './images/img8.jpeg'; import imag11Image from './images/img11.jpeg'; import { useNavigate } from 'react-router-dom'; import { Link as ScrollLink } from 'react-scroll';

export default function Homepage() { const navigate = useNavigate(); const [selectedItems, setSelectedItems] = useState([]); const [popMessage, setPopMessage] = useState(null);

const [showMiniForm, setShowMiniForm] = useState(false); // State to control showing the mini form

const handleOrderClick = (event) => {

const h2Text = event.target.closest('.box\_info').querySelector('h2').innerText;

setPopMessage(`You have selected: ${h2Text}`);

setTimeout(() => setPopMessage(null), 500); // Remove pop message after 0.5 seconds setSelectedItems((prevItems) => [...prevItems, h2Text]);

};

const handleRemoveItem = (indexToRemove) => {

setSelectedItems((prevItems) => prevItems.filter((\_, index) => index !== indexToRemove));

};

const handleProceedToContact = () => {

navigate('/contact', { state: { order: selectedItems } }); };

const toggleCart = () => {

setShowMiniForm(!showMiniForm); // Toggle showing the mini form

};

return ( <>

<Navbar toggleCart={toggleCart} isCartOpen={showMiniForm} />

<section className="home\_page">

<div className="home\_content">

<div><img className='div\_img' src={images} alt="Description of the image" /></div>

<div className='home\_info'>

<h1 id='home\_h1'>Welcome to Foodify!</h1>

<p id='home\_p'>At Foodify, we believe that every meal should be an experience to savor. Our chefs are passionate about crafting dishes that not only satisfy your hunger but also delight your senses. Whether you're here for a casual lunch, a family dinner, or a special celebration, we've got something to make every moment unforgettable. .</p> <li>

<ScrollLink id="link" to="menu" smooth={true} duration={500}

className="home\_btn">Order</ScrollLink>

</li>

</div>

</div>

</section>

<div className='menu\_page' id='menu'>

{showMiniForm && (

<div className="mini-window-overlay">

<div className="mini-window">

<h3>Selected Items</h3>

<ul>

{selectedItems.map((item, index) => (

<li key={index}>

{item} <button onClick={() => handleRemoveItem(index)}>Remove</button>

</li>

))}

</ul>

<button id="finish\_btn" onClick={handleProceedToContact}>Finish Order</button> <button className="go\_back\_btn" onClick={toggleCart}>Go Back</button>

</div>

</div>

)}

<div className="home-container">

{popMessage && (

<div className="pop-message">

{popMessage}

</div>

)}

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img1' src={dessertImage} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Masala Dosa</h2>

<p>Masala Dosa is one of the most popular South Indian breakfast dishes served in restaurants and tiffin centres.</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img2' src={imag11Image} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Biriyani</h2>

<p>Biryani is a mixed rice dish, mainly popular in South Asia. It is made with rice, some type of meat</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img3' src={imag2Image} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Chicken Tikka </h2>

<p>Chicken Tikka are boneless pieces of chicken,threaded on a metal skewer and cooked on live charcoal.</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img4' src={imag3Image} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Chicken Popcorn</h2>

<p>Popcorn chicken is a dish consisting of small, bite-sized pieces of chicken (about the size of popped corn kernels) that have been breaded and fried.</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img5' src={imag4Image} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Pizza</h2>

<p>pizza, dish of Italian origin consisting of a flattened disk of bread dough topped with some combination of olive oil, oregano, tomato, olives, mozzarella or other cheese, and many other ingredients,</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img6' src={imag5Image} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Fried Fish</h2>

<p>Fried fish is any fish or shellfish that has been prepared by frying. Often, the fish is covered in batter, egg and breadcrumbs, flour, or herbs and spices before being fried and served, often with a slice of lemon</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img7' src={imag6Image} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Burger</h2>

<p>A burger is a patty of ground beef grilled and placed between two halves of a bun.

Slices of raw onion, lettuce,and other ingredients add flavor.</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img8' src={imag7Image} alt="Description of the

<div className='box\_info'>

<h2>Mommos</h2>

<p>It is believed that momos originated in Tibet and were introduced to Nepal by

Tibetan immigrants who settled in the Kathmandu Valley.</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

<div className="box box1" >

<div className='img\_info'>

<div><img className='box\_img img9' src={imag8Image} alt="Description of the image" /></div>

<div className='box\_info'>

<h2>Noodels</h2>

<p>Noodle, a cooked egg-and-flour paste prominent in European and Asian cuisine, generally distinguished from pasta by its elongated ribbonlike form.</p>

<button className='dev\_btn' onClick={handleOrderClick}>Add to Cart</button>

</div>

</div>

</div>

</div>

</div>

</>

); }

## Backend Controllers contactCtrl.js

// controllers/contactController.js

const Contact = require('../models/contact');

const submitMessage = async (req, res) => { const { name, email, message, order } = req.body; try { const newContact = new Contact({ name, email, message, order }); await newContact.save(); res.status(200).json({ message: 'Message sent successfully' }); } catch (error) { console.error('Error saving message:', error); res.status(500).json({ message: 'Error sending message' }); } }; module.exports = submitMessage;

## userCtrl.js

const bcryptjs=require('bcryptjs')

const User = require('../models/user')

//bcryptjs-used for hashing password const Login=async(req,res)=>{

const body=req.body

const findUser=await User.findOne({email:body.email})

const comparePassword=await bcryptjs.compare(body.password,findUser.password) if(!comparePassword){

return res.status(401).json({errorMessage:'Invalid Password'}) }

res.send({message:'Success', data:findUser}) }

const Register=async (req,res)=>{

// Data which we getting from user in req,body const body=req.body const hashedpassword= await bcryptjs.hash(req.body.password,10) const saveData= await User.create({

name:body.name, email:body.email, // saving the hashed password password: hashedpassword

})//saves in mongo database

res.send({message:'Success',data:saveData}

module.exports={

Login,

Register

}

## Database Connection.js

const mongoose = require('mongoose')

function RunServer() {

try {

mongoose.connect('mongodb://localhost:27017/douts'); console.log('mongodb connected');

} catch (error) { console.log(error.message)

}

}

module.exports = RunServer //exploring the function RunServer

## Loginpage Contact.js

// models/contact.js

const mongoose = require('mongoose');

const contactSchema = new mongoose.Schema({ name: {

type: String, required: true

}, email: {

type: String, required: true

},

message: { type: String, required: true

}, order: {

type: [String], // Array of strings to store order items

required: true

}, date: { type: Date,

default: Date.now

}

});

const Contact = mongoose.model('Contact', contactSchema); module.exports = Contact;

# user.js

const mongoose=require('mongoose') const userSchema= new mongoose.Schema({ name:{ type:String, required:true

}, email:{ type:String,

required:true

},

password:{ type:String, required:true

},

}

) const User= mongoose.model('user',userSchema) module.exports=User;

# Registerpage contactRoutes.js

const express=require('express')

const submitMessage = require('../controllers/contactCtrl')

const router=express.Router()

router.post('/contact',submitMessage)

module.exports=router;

# userRoutes.js

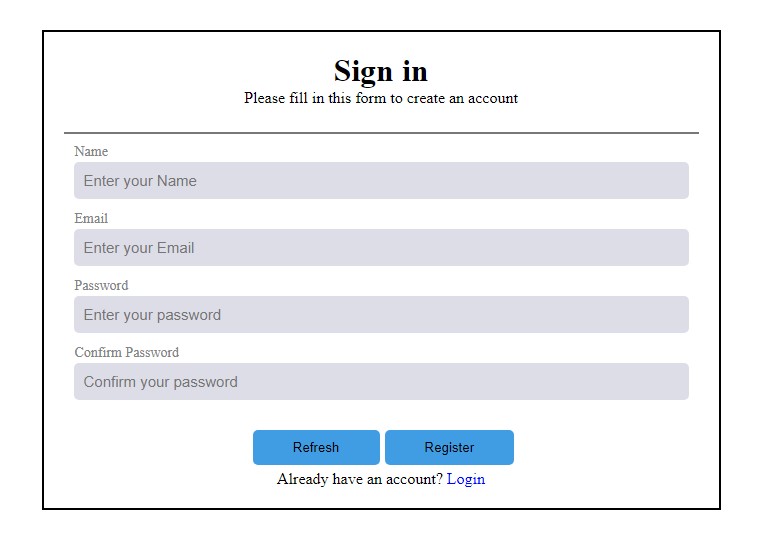
const express=require('express')

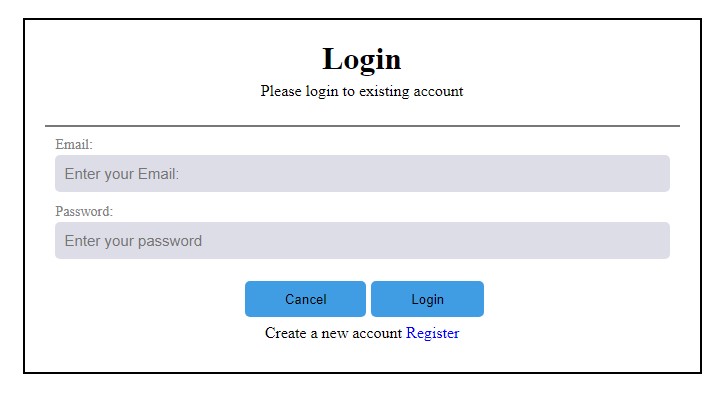
const { Register, Login } = require('../controllers/userCtrl')

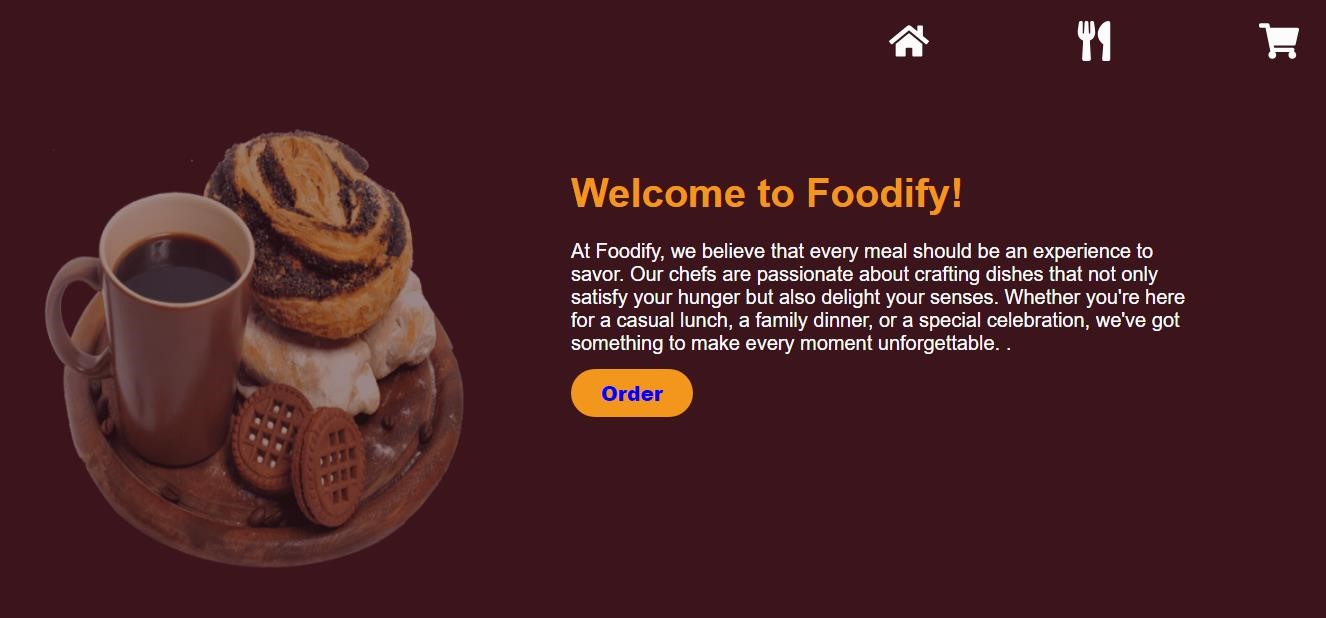
const router=express.Router() router.post('/',Register)

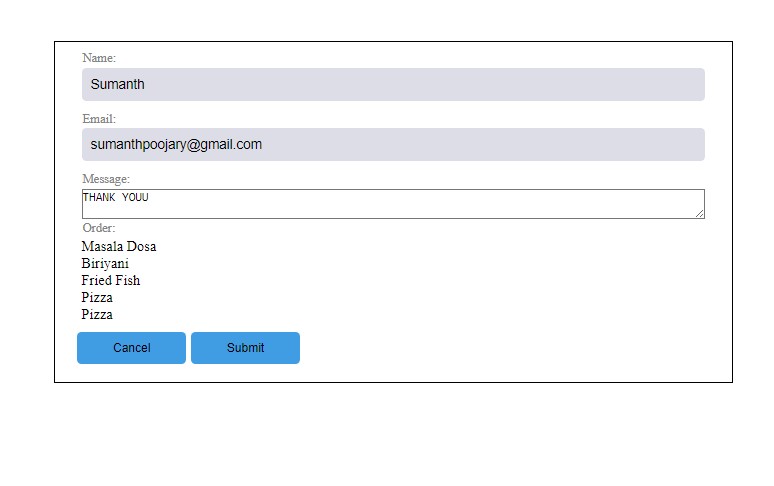
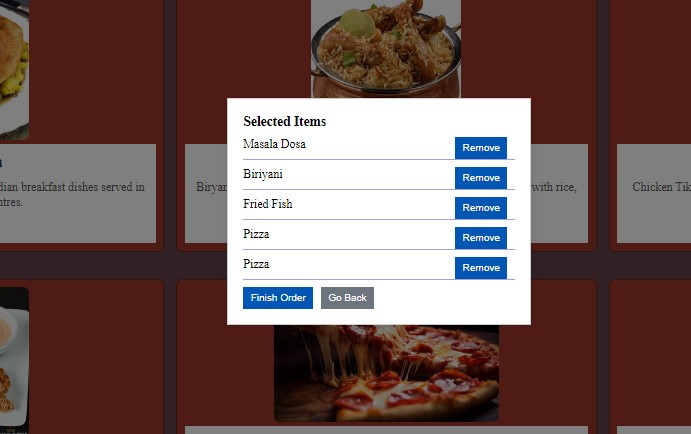
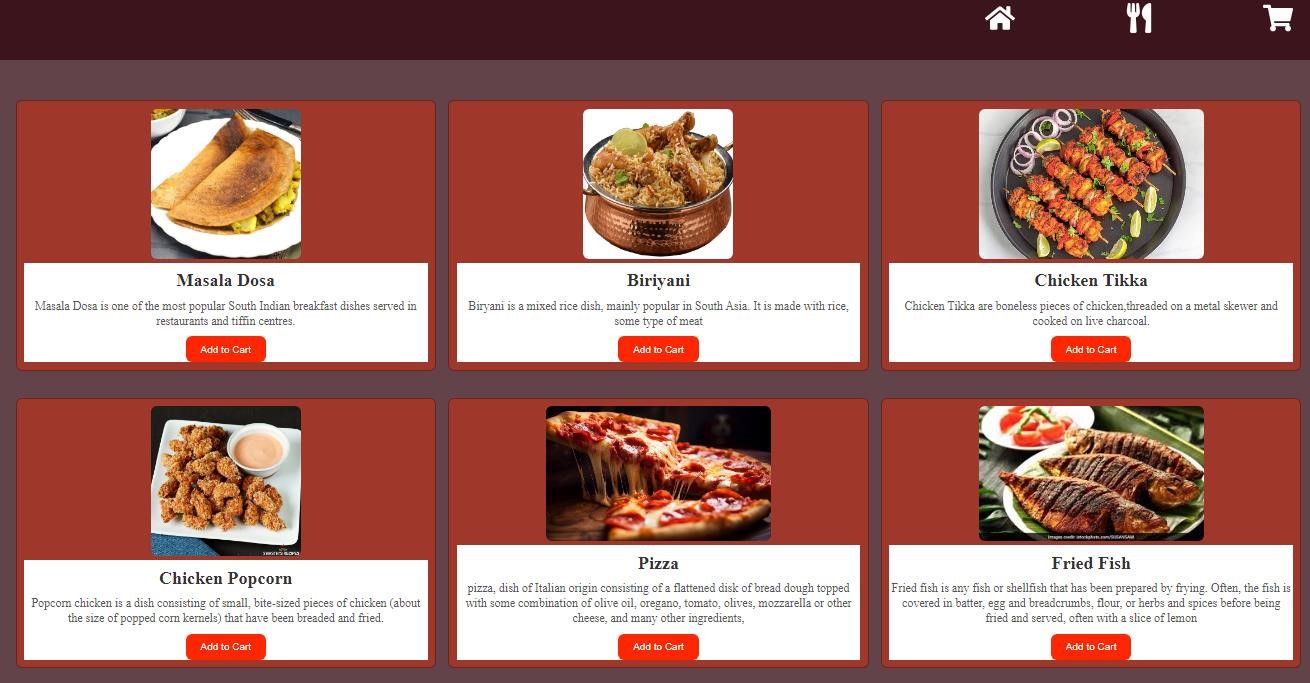
router.post('/login',Login)

module.exports=router;

**SNAPSHOTS** 







## CONCLUSION

This project showcases the integration of various technologies to meet the primary objectives of providing admin control, maintaining accurate records, and offering an interactive platform for data management and analysis. Through the use of RESTful APIs, secure user authentication, and a dynamic frontend, "Foodiifyy" is positioned as a valuable tool for managing restaurant operations. The system is designed to be maintainable and extensible, allowing for future growth and enhancements, thus exemplifying best practices in modern web development and the potential of the MERN stack in building comprehensive web solutions.

The implementation of the "Foodiifyy" restaurant management system demonstrates the effective use of the MERN stack (MongoDB, Express.js, React.js, Node.js) in creating a modern, scalable, and efficient web application. By leveraging the strengths of each component in the stack, the system provides a seamless user experience, robust data management, and secure authentication mechanisms. The project encompasses a comprehensive approach to web development, integrating HTML for content structure, CSS for design, and JavaScript for interactivity and backend operations, ensuring a cohesive and functional application.

\*\*\*\*\*\*\*\*\*\*\*\*