EXPOSYS DATA LABS

Project Title

“Sizzling Burger Deals and Blog Application”

Submitted By

Keertana

Master of Computer Applications

BMS Institute of Technology and Management

In fulfilment for the Internship as

FULL STACK DEVELOPER

2023 – 2024

ABSTRACT

‘’Sizzling Burger Deals’’

A online website which sells burger

"Sizzling Burger Deals", an innovative online platform dedicated to delivering a gourmet burger experience to customers. The platform offers a diverse menu featuring high-quality burgers, each crafted with premium ingredients. Users can easily navigate the user-friendly interface to explore menu options, customize their orders, and conveniently place them in the virtual cart. The application also integrates a real-time cart system, allowing users to view their selected items and proceed to a seamless checkout process. The abstract highlights the platform's unique features, emphasizing its commitment to providing a delightful and customized online burger ordering experience.

INDEX

INTRODUCTION

- ## Front-end Project :

Welcome to the flavor-filled world of Sizzling Burger Deals, where your burger experience is tailor-made to suit your cravings. Explore the art of customization by clicking on "Customize," unveiling a palette of ingredients to create your burger masterpiece. Not in the mood for customization? No problem! Swiftly add your favorite burgers to the cart with a click on the "Add to Cart" symbol.

Managing your selections is a breeze with our intuitive cart interface – effortlessly delete items for a seamless ordering experience. When you're ready to savor your creations, a simple tap on "Order" unveils a QR code, revolutionizing payment and ensuring a smooth, hassle-free transaction.

Sizzling Burger Deals is more than an online platform; it's a culinary adventure where your preferences dictate the journey. Delight in a diverse range of burgers, experience easy customization, and indulge in a convenient ordering process that puts you in the driver's seat of your burger escapade.

- ## Back-end Project :

Welcome to the behind-the-scenes magic of Sizzling Burger Deals! Our website not only caters to your burger cravings but also ensures a seamless user journey with a robust backend. When registering, users provide their details through a secure registration form, creating a personalized account. These credentials are securely stored in a MySQL database, enabling hassle-free logins and allowing users to revisit their favorite orders.

Behind the scenes, our Apache Tomcat server facilitates smooth communication, ensuring a responsive experience. Admins wield the power to manage the database, performing essential CRUD (Create, Read, Update, Delete) operations. This backend orchestration guarantees a reliable, secure, and dynamic environment, enhancing your online burger ordering adventure.

EXISTING SYSTEM

The current system lacks user interactivity, restricting users to generic burger purchases without customization options. It operates without login credentials, eliminating personalized experiences. Furthermore, the absence of a payment mechanism and database connectivity means no user data is stored or transactions are facilitated.

PROPOSED SYSTEM

In the proposed system, a dynamic front-end is introduced with user-friendly login and registration features. Users can now customize and add burgers to their carts, fostering a personalized experience. The addition of a secure backend, powered by Java servlets and a MySQL database managed by admin through Apache Tomcat, ensures robust data storage and retrieval. The implementation of a payment option, symbolized by a generated QR code, enhances the overall efficiency of the online burger purchasing process.

ARCHITECTURE

- ## Front-end Project :

The "Sizzling Burger Deals" web application follows a client-server architecture, to create a dynamic and interactive user experience. Here's an overview of the Architecture

Client-Side (Front-End):

- HTML/CSS/JavaScript: The front-end is built using HTML for structure, CSS for styling, and JavaScript for dynamic behaviour. This trio ensures a visually appealing and responsive user interface.
- DOM Manipulation: JavaScript is used to manipulate the Document Object Model (DOM) in real-time, allowing dynamic updates to the user interface based on user interactions such as adding or removing ingredients.

Interaction:

- User Actions: Users interact with the application by adding or removing ingredients, initiating the checkout process, and completing their orders.
- Events: JavaScript event listeners capture user actions (e.g., button clicks), triggering functions that update the DOM and total cost in real-time.

Higher Order Functions:

- Event Listeners: The use of higher-order functions is evident in event listeners. For instance, `foreach` is used to iterate over the elements with the class `.add-button` and attach click event listeners

ES6 Compatibility:

- Modern JavaScript Features: The code incorporates ES6 features like arrow functions, `const` and `let` for variable declarations, and the `query Selector` method for DOM selection.

- **Back-end Project :**

The architecture of the Blog Application project involves various components working together to provide a seamless and efficient experience for users in managing their blogs. Here is a high-level overview of the architecture:

Frontend :

- HTML, CSS, JavaScript: The frontend is developed using standard web technologies, including HTML for markup, CSS for styling, and JavaScript for interactive features.

Backend :

- Java: The backend logic is implemented using java script, a server-side scripting language, to handle dynamic content generation, user authentication, and interactions with the database.
- Server side processing : Java servlets is a server-side components, handling business logic, data processing, and dynamic content generation to create robust and scalable web applications, ensuring seamless integration with front-end technologies.

Database :

- MySql: The project uses MySql as the relational database management system to store and retrieve blog-related data.

Server :

- Web Server (Apachre-tomcat): Responsible for serving web pages and handling requests from clients.

Navigation Button Functionality:

- Create, Read, Edit, Delete (CRUD) Operations: The backend handles CRUD operations for blog posts, allowing users to create, read, edit, and delete their blogs.

METHODOLOGY

Front-end Project

The development of the "Sizzling Burger Deals" web application follows an agile and iterative software development methodology, emphasizing flexibility, collaboration, and incremental improvements. Here's an overview of the methodology applied:

Agile Methodology:

- Iterative Development: The project adopts an iterative approach, with regular increments in functionality. Each iteration focuses on specific features or improvements, allowing for continuous feedback and adjustment

Collaborative Development:

- Cross-Functional Teams: Collaboration is encouraged among cross-functional teams, including designers, frontend developers, and potentially back-end developers. This promotes a holistic approach to development.

Incremental Development:

- Feature Additions: Features are added incrementally, allowing for early releases and rapid deployment. This approach enables users to access new features sooner and provides developers with continuous feedback.

Back-end Project

The development methodology for the “Blog Application” project involves an iterative and incremental approach, emphasizing collaboration between cross-functional teams. A suitable methodology for this project is the “Agile Software Development Methodology”.

Agile Methodology Overview:

Scrum Framework:

Roles:

- Product Owner: Represents stakeholders, defines features, and prioritizes the product backlog.
- Scrum Master: Facilitates the Scrum process, removes impediments, and ensures the team follows Agile principles.
- Development Team: Cross-functional group responsible for delivering potentially shippable increments of the product.

Artifacts:

- Product Backlog: Prioritized list of features and enhancements.
- Sprint Backlog : Subset of the product backlog selected for a sprint.
- Increment : The sum of all completed backlog items at the end of a sprint.

Events:

- Sprint Planning: Defines the work for the sprint.
- Daily Standup: Brief daily meeting for team synchronization.
- Sprint Review: Review of the increment and adaptation of the product backlog
- Sprint Retrospective: Reflects on the past sprint and plans for improvements.

IMPLEMENTATION

Source Code :

Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Webpage Design</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div class="main">
<div class="nabar">
<div class="icon">
```

```
<h2 class="logo">Sizzling Burger Deals</h2>

</div>

<div class="menu">

<ul>

<li><a href="#" onclick="toggleContent('home')"><button class="nav-
btn">HOME</button></a></li>

<li><a href="#" onclick="toggleContent('aboutus')"><button class="nav-btn">ABOUT
US</button></a></li>

<li><a href="#" onclick="toggleContent('contact')"><button class="nav-
btn">CONTACT</button></a></li>

</ul>

</div>

</div>

<div class="content" id="mainContent">

<h1>Burger Delights & <br><span>Online Orders</span> <br>Available</h1>

<p class="par">Explore the world of delicious burgers right at your fingertips.<br>We offer a
wide selection of mouthwatering burgers for you to enjoy.<br>From classic cheeseburgers to
specialty creations,<br>we've got something for everyone.<br> Place your order online and
savor the flavors from the comfort of your home. <br>Quality, taste, and convenience - all in one
bite!</p>

<div class="form">

<form action="main.html">

<h2>Login Here</h2>

<input type="email" name="email" placeholder="Enter Email Here" required>

<input type="password" name="password" placeholder="Enter Password Here" required>

<button class="btnn"><a href="#">Login</a></button>

<p class="link">Don't have an account<br>

<a href="signup.html">Sign up </a> here</a></p>

</form>

</div>

</div>

</div>
```

```
<style>
*{
margin: 0;
padding: 0;
}
.main{
width: 100%;
background: linear-gradient(to top, rgba(0,0,0,0.5)50%,rgba(0,0,0,0.5)50%),
url("./background.jpg");
background-position: center;
background-size: cover;
height: 100vh;
padding-left:20px;
padding:0px;
}
.nabar{
width: 1200px;
height: 100px;
margin: auto;
}
.icon{
width: 200px;
float: left;
height: 70px;
}
.logo{
color: #ff7200;
font-size: 70px;
font-family: Arial;
padding-left: 20px;
```

```
padding:0px;
float: left;
padding-top: 10px;
margin-top: 5px;
white-space: nowrap;
}
.menu {
width: 100%;
text-align: right;
padding-right: 20px;
font-size: 20px;
padding-top: 40px;
cursor:pointer;
margin-top: 5px;
}
ul {
display: inline;
}
ul li {
display: inline;
margin-left: 20px;
}
ul li a{
text-decoration: none;
color: #fff;
font-family: Arial;
font-weight: bold;
transition: 0.4s ease-in-out;
}
ul li a:hover{
```

```
color: #ff7200;
}
.search{
width: 330px;
float: left;
margin-left: 270px;
}
.srch{
font-family: 'Times New Roman';
width: 200px;
height: 40px;
background: transparent;
border: 1px solid #ff7200;
margin-top: 13px;
color: #fff;
border-right: none;
font-size: 16px;
float: left;
padding: 10px;
border-bottom-left-radius: 5px;
border-top-left-radius: 5px;
}
.btn{
width: 100px;
height: 40px;
background: #ff7200;
border: 2px solid #ff7200;
margin-top: 13px;
color: #fff;
font-size: 15px;
```

```
border-bottom-right-radius: 5px;
border-bottom-right-radius: 5px;
transition: 0.2s ease;
cursor: pointer;
}
.btn:hover{
color: #000;
}
.btn:focus{
outline: none;
}
.srch:focus{
outline: none;
}
.nav-btn {
font-size: 15px; /* Adjusted font size */
padding: 15px 20px; /* Adjusted padding */
background-color: #ff7200;
color: #131010;
border: none;
border-radius: 15px;
cursor: pointer;
transition: 0.2s ease;
}
.nav-btn:hover {
background-color: #ffffff;
color: #ff7200;
}
.content{
width: 1200px;
```

```
height: auto;
margin: auto;
color: #fff;
position: relative;
}
.content .par{
padding-left: 20px;
padding-bottom: 25px;
font-family: Arial;
letter-spacing: 1.2px;
line-height: 30px;
}
.content h1{
font-family: 'Times New Roman';
font-size: 50px;
padding-left: 20px;
margin-top: 9%;
letter-spacing: 2px;
}
.content span{
color: #ff7200;
font-size: 65px
}
.form{
width: 250px;
height: 320px;
background: linear-gradient(to top, rgba(0,0,0,0.8)50%,rgba(0,0,0,0.8)50%);
position: absolute;
top: -20px;
left: 950px;
```

```
transform: translate(0%,-5%);
border-radius: 10px;
padding: 25px;
}
.form h2{
width: 220px;
font-family: sans-serif;
text-align: center;
color: #ff7200;
font-size: 22px;
background-color: #fff;
border-radius: 10px;
margin: 2px;
padding: 8px;
}
.form input{
width: 240px;
height: 35px;
background: transparent;
border-bottom: 1px solid #ff7200;
border-top: none;
border-right: none;
border-left: none;
color: #fff;
font-size: 15px;
letter-spacing: 1px;
margin-top: 30px;
font-family: sans-serif;
}
.form input:focus{
```

```
outline: none;
}
::placeholder{
color: #fff;
font-family: Arial;
}
.btnn{
width: 240px;
height: 40px;
background: #ff7200;
border: none;
margin-top: 30px;
font-size: 18px;
border-radius: 10px;
cursor: pointer;
color: #fff;
transition: 0.4s ease;
}
.btnn:hover{
background: #fff;
color: #ff7200;
}
.btnn a{
text-decoration: none;
color: #000;
font-weight: bold;
}
.form .link{
font-family: Arial, Helvetica, sans-serif;
font-size: 17px;
```

```
padding-top: 20px;
text-align: center;
}
.form .link a{
text-decoration: none;
color: #ff7200;
}
.liw{
padding-top: 15px;
padding-bottom: 10px;
text-align: center;
}
.icons a{
text-decoration: none;
color: #fff;
}
.icons ion-icon{
color: #fff;
font-size: 30px;
padding-left: 14px;
padding-top: 5px;
transition: 0.3s ease;
}
.icons ion-icon:hover{
color: #ff7200;
}
</style>

<script>
let originalContent = document.getElementById('mainContent').innerHTML;
```

```
let currentSection = 'home';


function toggleContent(section) {

if (section === currentSection) {

document.getElementById('mainContent').innerHTML = `<div
id="mainContent">${originalContent}</div>`;

currentSection = 'home';

} else {

let newContent = "";

switch (section) {

case 'home':

newContent = "<h1>Welcome to Sizzling Burger Deals <br> Where EveryBite is the Flavour
Explosion</h1>";

break;

case 'aboutus':

newContent = "<h1>Welcome to Sizzling Burger Deals, where passion for burgers meets
culinary artistry. Our journey began with a simple yet ambitious goal — to redefine your burger
experience. At Sizzling Burger Deals, we take pride in crafting mouthwatering burgers that not
only tantalize your taste buds but also tell a story of flavors.</h1>";

break;

case 'contact':

newContent = "<h1>Contact Us</h1><p>Reach out to us at info@sizzlingburgers.com</p>";

break;

default:

break;

}

document.getElementById('mainContent').innerHTML = `<div
id="mainContent">${newContent}</div>`;

currentSection = section;

}

}

</script>
```

```
</body>
</html>
```

Sign up page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration</title>
    <style>body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 0;
        color: #fff;
        background-image: url(./signback.jpg);
        background-repeat: no-repeat;
        background-size: cover;
    }
    .container {
        display: flex;
        justify-content: center;
        align-items: center;
        min-height: 100vh;
    }
    .registration-form {
        background-color: #000;
        border-radius: 10px;
        box-shadow: 0px 0px 20px rgba(94, 130, 29, 0.5);
        text-align: center;
```

```
    padding: 30px;
    width: 600px;
}
.registration-form h1 {
    font-size: 36px;
    color: #FF7200;
    margin: 0px;
    margin-bottom: 20px;
    text-transform: uppercase;
}
.registration-form p {
    font-size: 18px;
    color: #ccc;
    margin: 15px 0;
}
.input-group {
    text-align: left;
    margin: 20px 0;
}
.input-group label {
    display: block;
    font-size: 16px;
    color: #f3efed;
    text-align: left;
    margin-bottom: 10px;
}
.input-group input[type="text"],
.input-group input[type="email"],
.input-group input[type="tel"],
.input-group input[type="password"] {
```

```
    width: 100%;
    padding: 15px;
    margin: 10px -5px;
    border: none;
    border-bottom: 2px solid #FF7200;
    font-size: 18px;
    color: #fff;
    background-color: transparent;
    transition: border-color 0.3s;
}
.input-group input[type="text"]:focus,
.input-group input[type="email"]:focus,
.input-group input[type="tel"]:focus,
.input-group input[type="password"]:focus {
    border-color: #ef700e;
}
.input-group input[type="checkbox"] {
    margin-right: 5px;
}
button {
    background-color: #FF7200;
    color: #fff;
    border: none;
    padding: 15px 25px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 18px;
    text-transform: uppercase;
    transition: background-color 0.3s;
}
```

```
    button:hover {
        background-color: #000;
    }
    .registration-form a {
        text-decoration: none;
        color: #FF7200;
        font-weight: 600;
        font-size: 16px;
        transition: color 0.3s;
    }
    .registration-form a:hover {
        color: #000;
    }
     </style>
</head>
<body>
    <div class="container">
        <form class="registration-form">
            <h1>Welcome to Sizzling Burger Deals</h1>
            <p>Create Your Account</p>
            <div class="input-group">
                <label for="firstName">First Name:</label>
                <input type="text" id="firstName" name="firstName" required>
            </div>

            <div class="input-group">
                <label for="lastName">Last Name:</label>
                <input type="text" id="lastName" name="lastName" required>
            </div>
            <div class="input-group">
```

```
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
</div>
<div class="input-group">
    <label for="phone">Phone Number:</label>
    <input type="tel" id="phone" name="phone" required>
</div>
<div class="input-group">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
</div>
<div class="input-group">
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
</div>
<div class="input-group">
    <label for="street">Street Address:</label>
    <input type="text" id="street" name="street" required>
</div>
<div class="input-group">
    <label for="city">City:</label>
    <input type="text" id="city" name="city" required>
</div>
<div class="input-group">
    <label for="state">State/Province:</label>
    <input type="text" id="state" name="state" required>
</div>
<div class="input-group">
    <label for="zip">ZIP/Postal Code:</label>
    <input type="text" id="zip" name="zip" required>
```

```
            </div>
            <div class="input-group">
                <label for="country">Country:</label>
                <input type="text" id="country" name="country" required>
            </div>
            <div class="input-group">
                <input type="checkbox" id="newsletter" name="newsletter" value="yes">
                <label for="newsletter">Yes, I want to receive newsletters.</label>
            </div>
            <div class="input-group">
                <input type="checkbox" id="agree" name="agree" required>
                <label for="agree">By creating an account, you agree to our Terms and Conditions and Privacy Policy.</label>
            </div>
            <button type="submit">Create Account</button>
            <p>Already have an account? <a href="./home.html">Login here</a></p>
        </form>
    </div>
</body>
<script>
    document.addEventListener('DOMContentLoaded', function () {
        const form = document.querySelector('.registration-form');
        form.addEventListener('submit', function (event) {
            event.preventDefault();
            if (validateForm()) {
                storeUserData();
                alert('Account created successfully!');
            }
        });
        function validateForm() {
```

```
    const firstName = document.getElementById('firstName').value.trim();
    const lastName = document.getElementById('lastName').value.trim();
    const email = document.getElementById('email').value.trim();
    const phone = document.getElementById('phone').value.trim();
    const username = document.getElementById('username').value.trim();
    const password = document.getElementById('password').value.trim();
    const street = document.getElementById('street').value.trim();
    const city = document.getElementById('city').value.trim();
    const state = document.getElementById('state').value.trim();
    const zip = document.getElementById('zip').value.trim();
    const country = document.getElementById('country').value.trim();
    if (!firstName || !lastName || !email || !phone || !username || !password ||
        !street || !city || !state || !zip || !country) {
        alert('All fields are required.');
        return false;
    }
    return true;
}
function storeUserData() {
    const userData = {
        firstName: document.getElementById('firstName').value.trim(),
        lastName: document.getElementById('lastName').value.trim(),
        email: document.getElementById('email').value.trim(),
        phone: document.getElementById('phone').value.trim(),
        username: document.getElementById('username').value.trim(),
        password: document.getElementById('password').value.trim(),
        street: document.getElementById('street').value.trim(),
        city: document.getElementById('city').value.trim(),
        state: document.getElementById('state').value.trim(),
        zip: document.getElementById('zip').value.trim(),
```

```
            country: document.getElementById('country').value.trim(),
            newsletter: document.getElementById('newsletter').checked,
            agree: document.getElementById('agree').checked
        };
        localStorage.setItem('userData', JSON.stringify(userData));
    }
  });
</script>
</html>
```

Main page .html :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Delicious Burgers</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #333;
      color: #ff7200;
      margin: 0;
      padding: 0;
    }
    header {
      background-color: #070707;
      padding: 20px;
      text-align: center;
    }
```

```
header h1 {
    margin: 0;
    font-size: 60px;
    font-style: italic;
    color: #ff7200;
}
.cart-icon {
    position: fixed;
    top: 20px;
    right: 20px;
    cursor: pointer;
}
.cart-icon img {
    width: 80px;
}
.container {
    max-width: 1200px;
    margin: 20px auto;
    padding: 20px;
    background-color: #222;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
    border-radius: 5px;
}
.menu {
    display: flex;
    flex-wrap: wrap;
}
.menu-item {
    position: relative;
    margin: 10px;
```

```
    background-color: #333;
    border: 1px solid #ff7200;
    padding: 20px;
    text-align: center;
    border-radius: 10px;
}
.menu-item img {
    max-width: 100%;
    height: 200px;
    object-fit: cover;
    border-radius: 10px;
}
.cus {
    margin-top: 10px;
}
.cus button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    margin-right: 5px;
}
.cart-container {
    position: fixed;
    top: 70px;
    right: 20px;
    background-color: white;
```

```
    border: 1px solid #ccc;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    max-width: 300px;
    width: 100%;
    display: none;
    z-index: 1000;
    border-radius: 5px;
}
.cart-header {
    background-color: #ff7200;
    color: #fff;
    padding: 10px;
    text-align: center;
    font-weight: bold;
}
.cart-items {
    padding: 20px;
}
.cart-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
    border-bottom: 1px solid #ccc;
    cursor: pointer;
    transition: background-color 0.3s;
}
.cart-item:hover {
    background-color: #f5f5f5;
}
```

```
.cart-total {
    padding: 10px;
    font-weight: bold;
    text-align: center;
}
.cart-empty {
    padding: 20px;
    text-align: center;
    color: #777;
}
.remove-item {
    color: red;
    cursor: pointer;
}
.order-now-btn {
    background-color: #ff7200;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    display: block;
    margin: 20px auto;
}
#payment-page {
    display: none;
    max-width: 600px;
    margin: 20px auto;
    padding: 20px;
```

```
    background-color: #222;
    box-shadow: 0 0 10px rgba(172, 144, 144, 0.5);
    border-radius: 5px;
    color: #fff;
}
#qrcode {
    margin-top: 20px;
    text-align: center;
}
#payment-message {
    margin-top: 10px;
    text-align: center;
}
.tooltip {
    position: absolute;
    background-color: #333;
    color: white;
    padding: 5px;
    border-radius: 5px;
    display: none;
}
.modal {
    display: none;
    position: fixed;
    z-index: 1;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
```

```
            background-color: rgba(0, 0, 0, 0.5);
        }
        .modal-content {
            background-color: #222;
            margin: 10% auto;
            padding: 20px;
            border: 1px solid #ff7200;
            border-radius: 5px;
            color: #fff;
        }
        .close {
            color: #ff7200;
            float: right;
            font-size: 30px;
            font-weight: bold;
            cursor: pointer;
        }
        .close:hover {
            color: #fff;
        }
    </style>
    <!-- Include the QR code library -->
    <script src="https://cdn.rawgit.com/davidshimjs/qrcodejs/gh-pages/qrcode.min.js"></script>
</head>
<body>
    <header>
        <h1>Sizzling Burger Deals</h1>
        <div class="cart-icon" onclick="toggleCart()">
            <img src="https://cdn.pixabay.com/photo/2016/10/10/14/46/icon-1728552_1280.jpg"
alt="Cart">
```

```
    </div>
  </header>
  <div id="customize-tooltip" class="tooltip">
    <h3>Customize your Burger</h3>
    <div id="customization-options">
      <!-- Customize options will be dynamically added here -->
    </div>
  </div>
  <div class="container">
    <h2>Our Menu</h2>
    <div class="menu">
      <div class="menu-item">
        <img src="https://s23209.pcdn.co/wp-content/uploads/2022/07/220602_DD_The-Best-Ever-Cheeseburger_267-500x500.jpg"
          alt="Cheese Burger">
        <h3>Cheese Burger</h3>
        <p>Price: ₹300</p>
        <div class="cus">
          <button onclick="addToCart('Cheese Burger', 300)">Add to Cart</button>
          <button onclick="customizeBurger('Cheese Burger', 300)">Customize</button>
        </div>
      </div>
      <div class="menu-item">
        <img src="https://www.vegrecipesofindia.com/wp-content/uploads/2020/12/burger-recipe-4.jpg"
          alt="Veggie Burger">
        <h3>Veggie Burger</h3>
        <p>Price: ₹350</p>
        <div class="cus">
          <button onclick="addToCart('Veggie Burger', 350)">Add to Cart</button>
```

```
            <button onclick="customizeBurger('Veggie Burger', 350)">Customize</button>
        </div>
    </div>
    <div class="menu-item">
        <img src="https://natashaskitchen.com/wp-content/uploads/2020/06/Chicken-Sandwich-7.jpg"
            alt="Chicken Burger">
        <h3>Chicken Burger</h3>
        <p>Price: ₹250</p>
        <div class="cus">
            <button onclick="addToCart('Chicken Burger', 250)">Add to Cart</button>
            <button onclick="customizeBurger('Chicken Burger', 250)">Customize</button>
        </div>
    </div>
    <div class="menu-item">
        <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS7DlMgj-APCF5zuqIYYvUBZLKk74XLai-eJuZg-fUW6g&s"
            alt="Mushroom Burger">
        <h3>Mushroom Burger</h3>
        <p>Price: ₹180</p>
        <div class="cus">
            <button onclick="addToCart('Mushroom Burger', 180)">Add to Cart</button>
            <button onclick="customizeBurger('Mushroom Burger', 180)">Customize</button>
        </div>
    </div>
    <div class="menu-item">
        <img src="https://b.zmtcdn.com/data/pictures/chains/1/19733081/c29b03fda54d263bfdd4604dc0913469.jpg"
            alt="Gourmet Burger">
        <h3>Gourmet Burger</h3>
```

```
            <p>Price: ₹380</p>
            <div class="cus">
                <button onclick="addToCart('Gourmet Burger', 380)">Add to Cart</button>
                <button onclick="customizeBurger('Gourmet Burger', 380)">Customize</button>
            </div>
        </div>
    </div>
</div>
<div id="cart-container" class="cart-container">
    <div class="cart-header">Your Cart</div>
    <div id="cart-items" class="cart-items">
        <!-- Cart items will be dynamically added here -->
    </div>
    <button class="order-now-btn" onclick="orderNow()">Order Now</button>
    <div class="cart-total">Total: ₹0.00</div>
    <div class="cart-empty" id="cart-empty">Your cart is empty.</div>
</div>
<div id="payment-page">
    <h2>Payment Page</h2>
    <p>Total Amount: ₹0.00</p>
    <div id="qrcode"></div>
    <p id="payment-message">Scan the QR code and pay using any UPI app.</p>
    <button onclick="goBack()">Go Back</button>
</div>
<script>
    const menuData = [
        { name: 'Cheese Burger', price: 300 },
        { name: 'Veggie Burger', price: 350 },
        { name: 'Chicken Burger', price: 250 },
        { name: 'Mushroom Burger', price: 180 },
```

```
        { name: 'Gourmet Burger', price: 380 }
    ];
    const menuContainer = document.querySelector('.menu');
    const cartItemsContainer = document.getElementById('cart-items');
    const cartEmptyMessage = document.getElementById('cart-empty');
    const cartTotal = document.querySelector('.cart-total');
    const paymentPage = document.getElementById('payment-page');
    const cartItemsArray = [];
    function toggleCart() {
        const cartContainer = document.getElementById('cart-container');
        cartContainer.style.display = (cartContainer.style.display === 'none' ||
cartContainer.style.display === '') ? 'block' : 'none';
    }
    function addToCart(burgerName, price) {
        const cartItem = document.createElement('div');
        cartItem.classList.add('cart-item');
        cartItem.innerHTML = `
            <span>${burgerName} - ₹${price}</span>
            <span class="remove-item" onclick="removeFromCart(this)">X</span>
        cartItemsContainer.appendChild(cartItem);
        cartItemsArray.push({ name: burgerName, price: price });
        if (cartItemsContainer.children.length === 1) {
            cartEmptyMessage.style.display = 'none';
        }
        const total = calculateTotal();
        cartTotal.textContent = `Total: ₹${total.toFixed(2)}`;
    }
    function calculateTotal() {
        let total = 0;
        cartItemsArray.forEach(item => {
```

```
            total += item.price;
        });
        return total;
    }
    function orderNow() {
        const total = calculateTotal();
        if (total > 0) {
            document.getElementById('cart-container').style.display = 'none';
            document.getElementById('payment-page').style.display = 'block';
            paymentPage.querySelector('p').textContent = `Total Amount: ₹${total.toFixed(2)}`;
            generateQRCode(total.toFixed(2));
        } else {
            alert('Add items to the cart before placing an order.');
        }
    }
    function removeFromCart(element) {
        const index =
Array.from(element.parentNode.parentNode.children).indexOf(element.parentNode);
        removeFromCartArray(index);
        element.parentNode.remove();
        const total = calculateTotal();
        cartTotal.textContent = `Total: ₹${total.toFixed(2)}`;
        if (cartItemsContainer.children.length === 0) {
            cartEmptyMessage.style.display = 'block';
        }
    }
    function removeFromCartArray(index) {
        cartItemsArray.splice(index, 1);
    }
    function generateQRCode(amount) {
```

```
        document.getElementById("qrcode").innerHTML = "";
        const qrcode = new QRCode(document.getElementById("qrcode"), {
            text: `upi://pay?pa=your-upi-id@domain&pn=Your%20Name&mc=YourMerchantCode&tid=YourTransactionId&tr=YourOrderId&tn=Payment%20for%20Burgers&am=${amount}&cu=INR`,
            width: 128,
            height: 128
        });
    }
    function goBack() {
        document.getElementById('cart-container').style.display = 'block';
        document.getElementById('payment-page').style.display = 'none';
    }
  const customizeTooltip = document.getElementById('customize-tooltip');
  const customizationOptions = document.getElementById('customization-options');
  let currentCustomizationBurger = '';
  function customizeBurger(burgerName, price) {
    currentCustomizationBurger = burgerName;
    customizeTooltip.style.display = 'block';
    const options = getCustomizationOptions(burgerName);
    populateCustomizationOptions(options);
    const submitButton = document.createElement('button');
    submitButton.textContent = 'Submit';
    submitButton.addEventListener('click', addToCartCustom);
    customizationOptions.appendChild(submitButton);
    submitButton.dataset.price = price;
  }
  function getCustomizationOptions(burgerName) {
    const burgerOptions = {
        'Cheese Burger': [
```

```
            { name: 'Extra Cheese', price: 20 },
            { name: 'Extra Bacon', price: 30 },            ],
        'Veggie Burger': [
        { name: 'Extra Vegetables', price: 20 },
        { name: 'Extra Sauces', price: 30 },
        ],
        'Chicken Burger': [
        { name: 'Extra Chicekn patty', price: 20 },
        { name: 'Extra Cheese', price: 30 },
        ],
        'Mushroom Burger': [
        { name: 'Extra Mushroom Patty', price: 20 },
        { name: 'Extra Tomato', price: 30 },
        ],
        'Gourmet Burger': [
        { name: 'Extra Cheese', price: 20 },
        { name: 'Extra Sauces', price: 30 },
        ],
    };
    return burgerOptions[burgerName] || [];
}
function populateCustomizationOptions(options) {
    customizationOptions.innerHTML = '';
    options.forEach(option => {
        const checkbox = document.createElement('input');
        checkbox.type = 'checkbox';
        checkbox.name = 'customization-option';
        checkbox.value = option.name;
        checkbox.dataset.price = option.price;
        checkbox.addEventListener('change', updateTotal);
```

```
            const label = document.createElement('label');
            label.innerHTML = `${option.name} (+₹${option.price.toFixed(2)})`;
            customizationOptions.appendChild(checkbox);
            customizationOptions.appendChild(label);
            customizationOptions.appendChild(document.createElement('br'));
        });
    }
    function updateTotal() {
        const total = calculateCustomizationTotal();
        customizationOptions.querySelector('button').dataset.total = total;
    }
    function calculateCustomizationTotal() {
        const checkboxes = document.querySelectorAll('input[name="customization-
option"]:checked');
        let total = 0;
        checkboxes.forEach(checkbox => {
            total += parseFloat(checkbox.dataset.price);
        });
        return total;
    }
    function addToCartCustom() {
        const selectedOptions = getSelectedCustomizationOptions();
        const total = parseFloat(customizationOptions.querySelector('button').dataset.total);
        addToCart(currentCustomizationBurger, total, selectedOptions);
        customizeTooltip.style.display = 'none';
    }
    function getSelectedCustomizationOptions() {
        const checkboxes = document.querySelectorAll('input[name="customization-
option"]:checked');
        const selectedOptions = [];
```

```
        checkboxes.forEach(checkbox => {
            selectedOptions.push({ name: checkbox.value, price: parseFloat(checkbox.dataset.price)
});
        });
        return selectedOptions;
    }
    const customizeButtons = document.querySelectorAll('.cus button:nth-child(2)');
    customizeButtons.forEach((button, index) => {
        button.addEventListener('click', function () {
            const burgerName = menuData[index].name;
            const price = menuData[index].price;
            customizeBurger(burgerName, price);
        });
    });
    function customizeBurger(burgerName, price) {
        currentCustomizationBurger = burgerName;
        const customizeButton = event.target;
        const cartImage = document.querySelector('.cart-icon img');
        const imageRect = cartImage.getBoundingClientRect();
        const imageBottom = imageRect.bottom + window.scrollY;
        const imageLeft = imageRect.left + window.scrollX;
        const leftOffset = 150; // You can adjust this offset as needed
        customizeTooltip.style.top = `${imageBottom + leftOffset}px`;
        customizeTooltip.style.left = `${imageLeft - leftOffset}px`;
        customizeTooltip.style.display = 'block';
        const options = getCustomizationOptions(burgerName);
        populateCustomizationOptions(options);
        const submitButton = document.createElement('button');
        submitButton.textContent = 'Submit';
        submitButton.addEventListener('click', addToCartCustom);
```

```
        customizationOptions.appendChild(submitButton);
        submitButton.dataset.price = price;
    }
    </script>
</body>
</html>
```

Backend code :

```
package Keertana.controller;
import java.io.IOException;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebServlet;
import varshini.dao.dao;
import varshini.dto.dto;
@WebServlet("/insert")
public class controller  extends GenericServlet;
        @Override
        public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException {
String id=req.getParameter("id");
int id1=Integer.parseInt(id);
String email=req.getParameter("email");
String phone=req.getParameter("phone");
String username=req.getParameter("username");
String password=req.getParameter("password");
String  street=req.getParameter("street");
String city=req.getParameter("city");
String state=req.getParameter("state");
```

```
String zip=req.getParameter("zip");
String country=req.getParameter("country");
Long phone1=Long.parseLong(phone);
 int zip1=Integer.parseInt(zip);
 dto dto=new dto();
dto.setId(id1);
 dto.setEmail(email);
 dto.setPhno(phone1);
 dto.setUser_name(username);
 dto.setPassword(password);
 dto.setState(state);
 dto.setCity(city);
 dto.setZip(zip1);
 dto.setCountry(country);
 dao dao=new dao();
 dao.insert(dto);
          }
}


package Keertana.controller;
import java.io.IOException;
import java.util.List;
import javax.servlet.GenericServlet;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebServlet;
import Keertna.dao.dao;
```

```
import keertana.dto.dto;
@WebServlet("/delete")
public class delete extends GenericServlet{
	@Override
	public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
		String id=req.getParameter("id");
		int cid=Integer.parseInt(id);
		dao dao=new dao();
		dao.delete(cid);
	}
	}
package Keertana.controller;
import java.io.IOException;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebServlet;
import varshini.dao.dao;
@WebServlet("/fetch")
public class fetch extends GenericServlet {
	@Override
	public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
		// TODO Auto-generated method stub
		String id=req.getParameter("id");
		int id1=Integer.parseInt(id);
		 dao d=new dao();
		Object o= d.fetch(id1);
```

```
                res.getWriter().print(o);
        }
}
package keertna.dao;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import varshini.dto.dto;
public class dao {
        EntityManagerFactory f = Persistence.createEntityManagerFactory("dev");
        EntityManager m1 = f.createEntityManager();
        EntityTransaction t = m1.getTransaction();
        public void insert(dto d)
        {
                t.begin();
                m1.persist(d);
                t.commit();
        }
        public String delete(int id) {
        dto d=m1.find(dto.class, id);
        if(d!=null)
        {
                t.begin();
                m1.remove(d);
                t.commit();
                return "data deleted";
        }
        else
        {
```

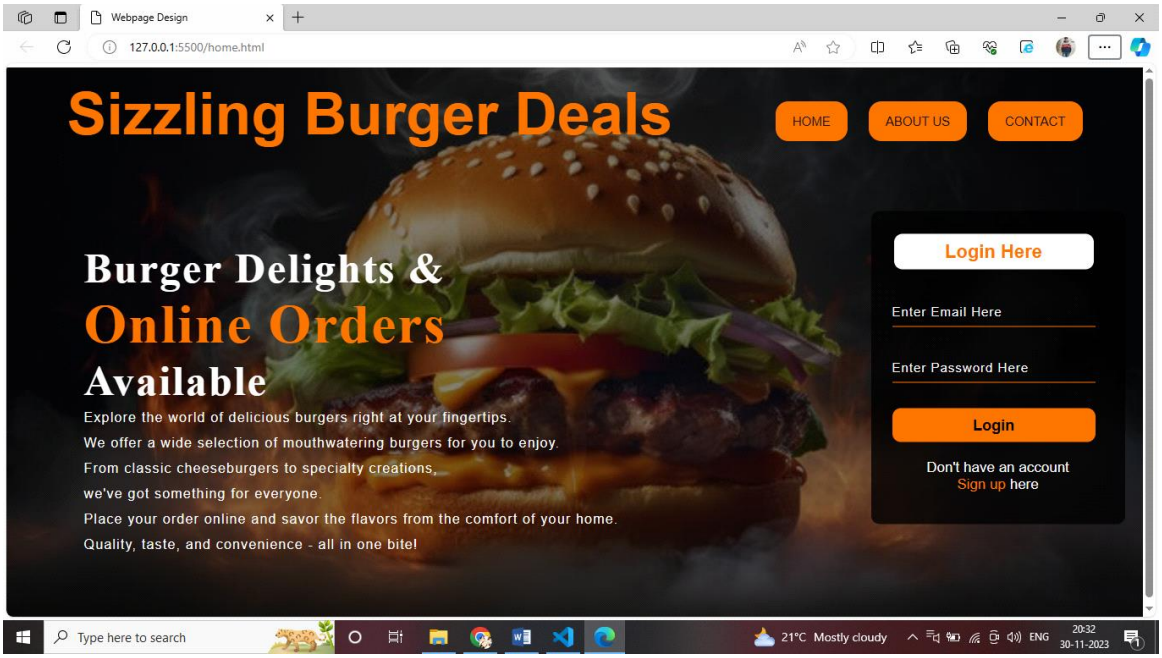
```
		return "no data found";
	}
	}
	public Object fetch(int id1) {
		dto d=m1.find(dto.class , id1);
		if(d!=null)
		{
		System.out.println(d.getId());
		System.out.println(d.getCity());
		System.out.println(d.getCountry());
		System.out.println(d.getEmail());
		System.out.println(d.getPassword());
		System.out.println(d.getPhno());
		System.out.println(d.getState());
		System.out.println(d.getZip());
		return "data fetched";
		}
		else
		{
			return "no data found";
		}
	}
}

package keertana.dto;
import javax.persistence.Entity;
import javax.persistence.Id;
@Entity
public class dto {
```

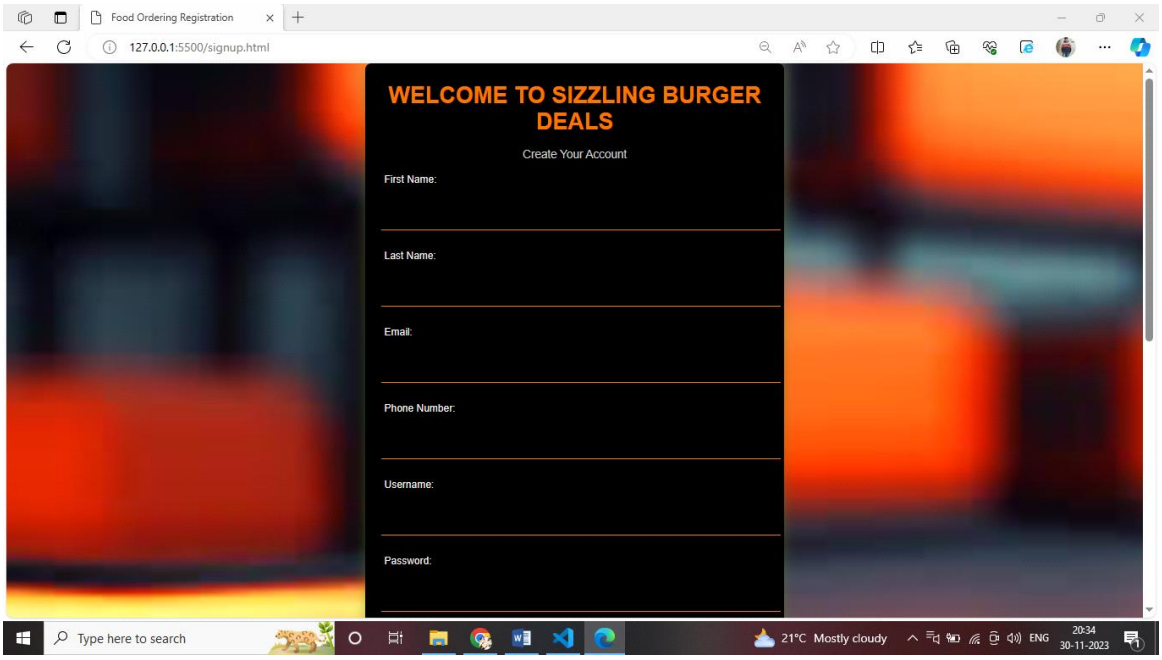
```
	@Id
	private int id;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	@Override
	public String toString() {
		return "dto [id=" + id + ", email=" + email + ", phno=" + phno + ", user_name="
+ user_name + ", password="
				+ password + ", address=" + address + ", city=" + city + ", state="
+ state + ", zip=" + zip
				+ ", country=" + country + "]";
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public long getPhno() {
		return phno;
	}
	public void setPhno(long phno) {
		this.phno = phno;
	}
	public String getUser_name() {
		return user_name;
```

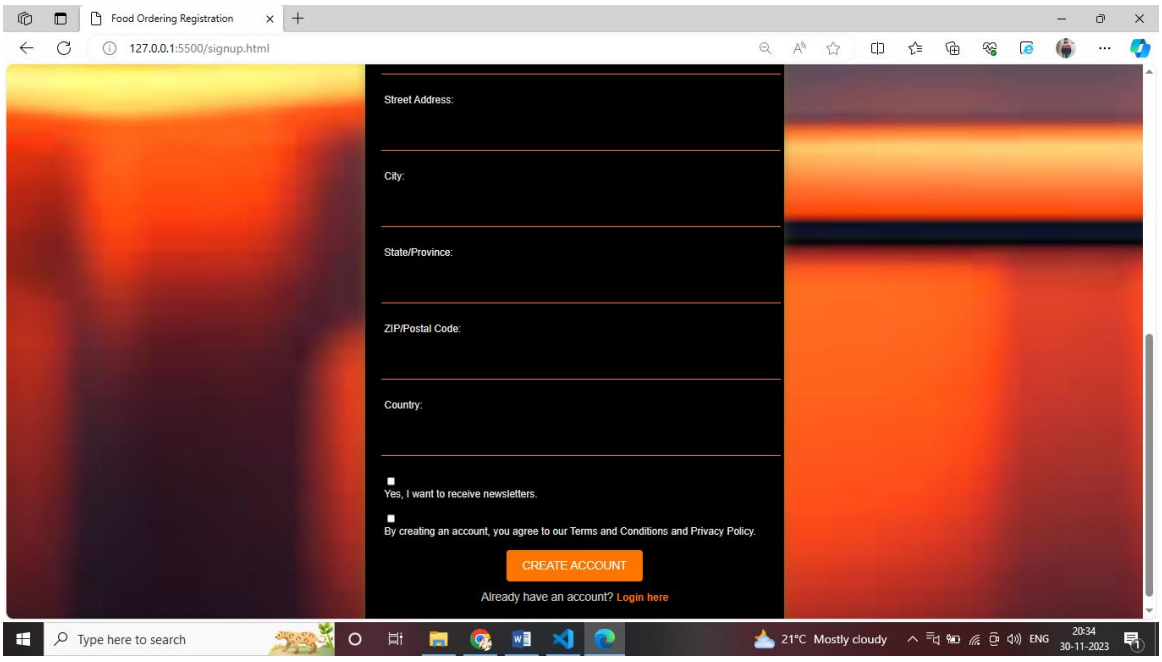
```
}
public void setUser_name(String user_name) {
	this.user_name = user_name;
}
public String getPassword() {
	return password;
}
public void setPassword(String password) {
	this.password = password;
}
public String getAddress() {
	return address;
}
public void setAddress(String address) {
	this.address = address;
}
public String getCity() {
	return city;
}
public void setCity(String city) {
	this.city = city;
}
public String getState() {
	return state;
}
public void setState(String state) {
	this.state = state;
}
public int getZip() {
	return zip;
```

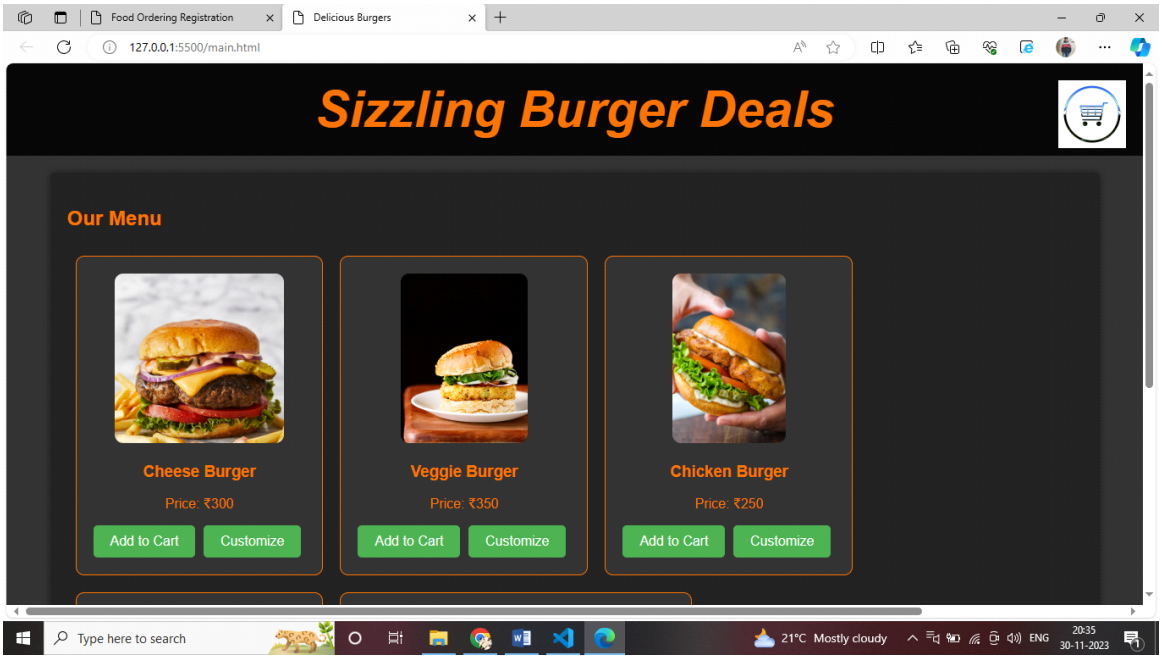
```
	}
	public void setZip(int zip) {
		this.zip = zip;
	}
	public String getCountry() {
		return country;
	}
	public void setCountry(String country) {
		this.country = country;
	}
	private String email;
	private long phno;
	private String user_name;
	private String password;
	private String address;
	private String city;
	private String state;
	private int zip;
	private String country;
	private String street;
}
```

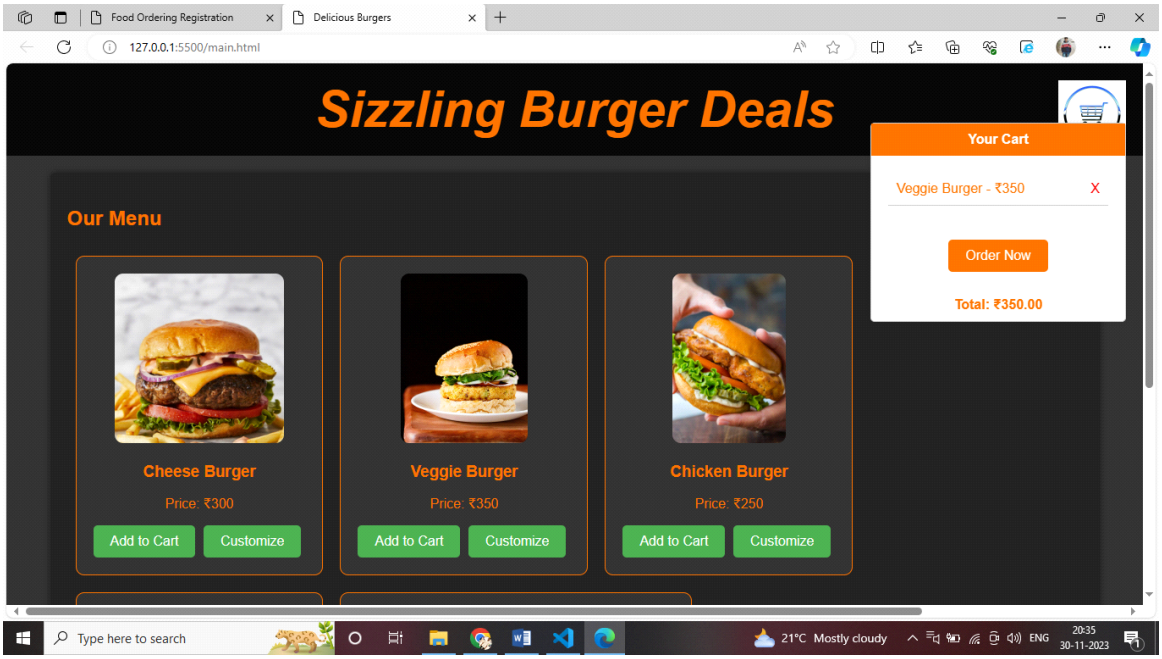
SCREENSHOTS

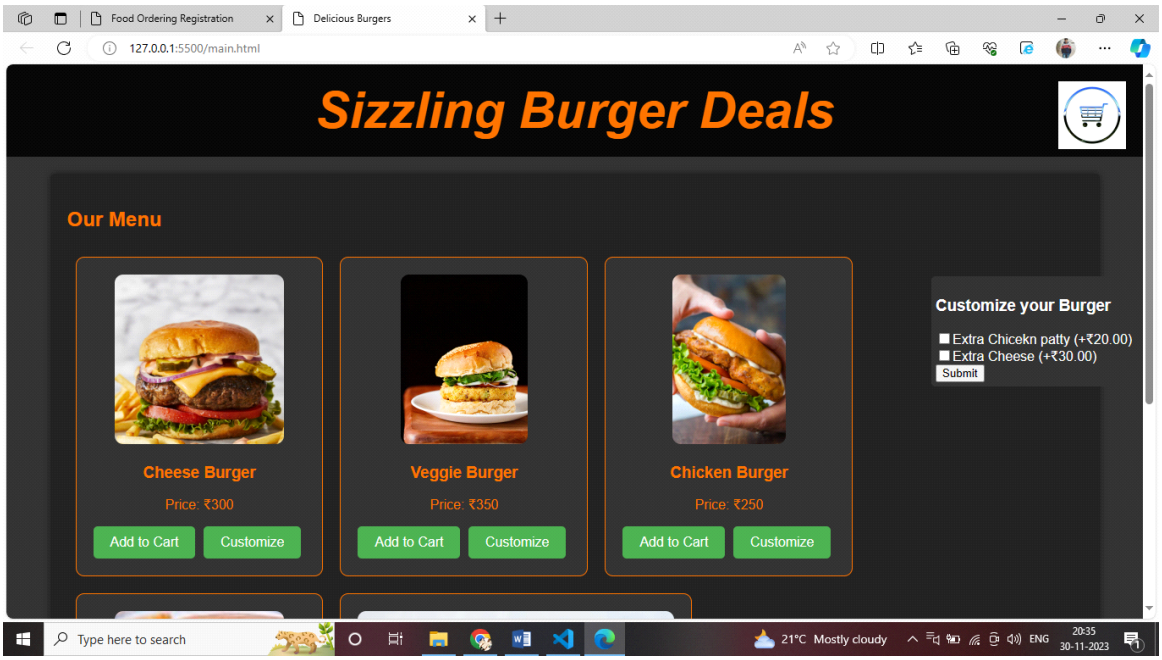


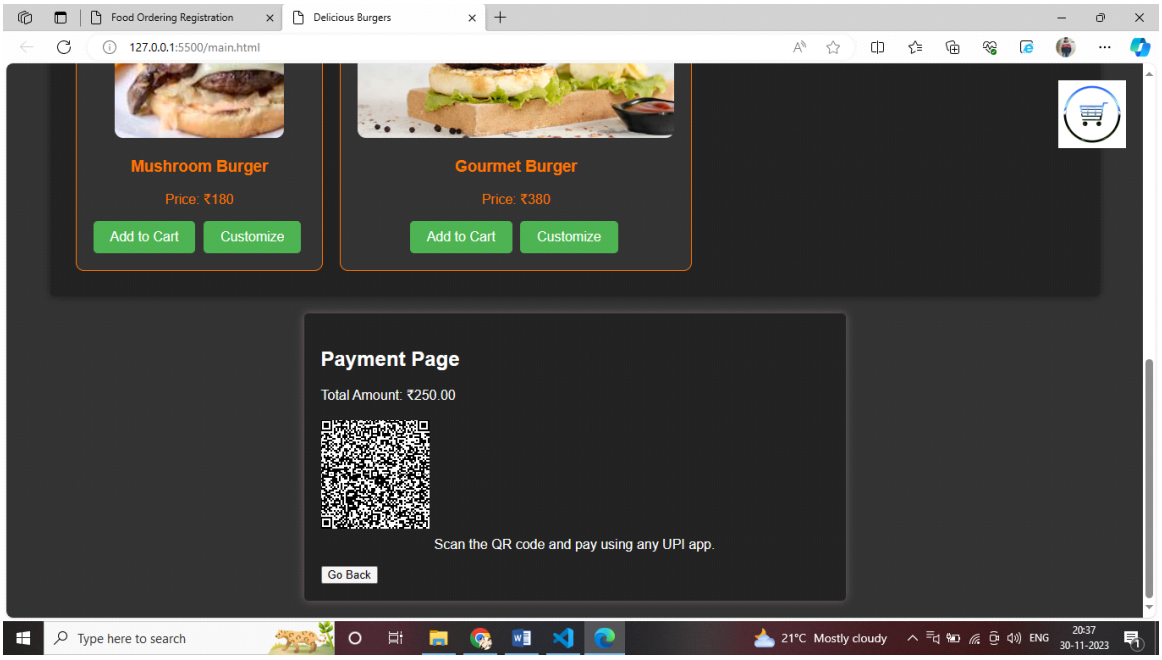

Food Ordering Registration
127.0.0.1:5500/signup.html
WELCOME TO SIZZLING BURGER DEALS
Create Your Account
First Name:
Last Name:
Email:
Phone Number:
Username:
Password:
Type here to search
21°C Mostly cloudy
ENG
20:34
30-11-2023


Food Ordering Registration
127.0.0.1:5500/signup.html
Street Address:
City:
State/Province:
ZIP/Postal Code:
Country:
Yes, I want to receive newsletters.
By creating an account, you agree to our Terms and Conditions and Privacy Policy.
CREATE ACCOUNT
Already have an account? Login here
Type here to search
21°C Mostly cloudy
ENG
20:34
30-11-2023


Food Ordering Registration
Delicious Burgers
127.0.0.1:5500/main.html
Sizzling Burger Deals
Our Menu
Cheese Burger
Price: ₹300
Add to Cart
Customize
Veggie Burger
Price: ₹350
Add to Cart
Customize
Chicken Burger
Price: ₹250
Add to Cart
Customize
Type here to search
21°C Mostly cloudy
ENG
20:35
30-11-2023


Food Ordering Registration
Delicious Burgers
127.0.0.1:5500/main.html
Sizzling Burger Deals
Your Cart
Veggie Burger - ₹350
X
Order Now
Total: ₹350.00
Our Menu
Cheese Burger
Price: ₹300
Add to Cart
Customize
Veggie Burger
Price: ₹350
Add to Cart
Customize
Chicken Burger
Price: ₹250
Add to Cart
Customize
Type here to search
21°C Mostly cloudy
ENG
20:35
30-11-2023


Food Ordering Registration
Delicious Burgers
127.0.0.1:5500/main.html
Sizzling Burger Deals
Our Menu
Cheese Burger
Price: ₹300
Add to Cart
Customize
Veggie Burger
Price: ₹350
Add to Cart
Customize
Chicken Burger
Price: ₹250
Add to Cart
Customize
Customize your Burger
Extra Chicekn patty (+₹20.00)
Extra Cheese (+₹30.00)
Submit
Type here to search
21°C Mostly cloudy
ENG
20:35
30-11-2023


Food Ordering Registration
Delicious Burgers
127.0.0.1:5500/main.html
Mushroom Burger
Price: ₹180
Add to Cart
Customize
Gourmet Burger
Price: ₹380
Add to Cart
Customize
Payment Page
Total Amount: ₹250.00
Scan the QR code and pay using any UPI app.
Go Back
Type here to search
21°C Mostly cloudy
ENG
20:37
30-11-2023

CONCLUSION

Front-end Project :

The “Sizzling Burger Deals “ front-end implementation features a user-friendly interface with secure login credentials and registration forms. The inclusion of a diverse burger selection provides users with the choice of instant ordering or personalized customization. The seamless addition of burgers to the cart, coupled with an efficient order process, culminates in the generation of a QR code, simplifying and expediting the payment experience for an overall convenient online burger shopping journey.

Back-end Project :

The “Sizzling Burger Deals” backend infrastructure ensures a secure and organized user experience. The registration details submitted by users are efficiently stored in a MySQL database, managed through Java servlets. The robust server-side architecture, powered by Apache Tomcat, operates seamlessly to handle user data, guaranteeing data integrity and security. Admin-exclusive database management ensures controlled interactions, enhancing overall system reliability and maintenance and we have implemented the CRUD operations that the only the Data Can be managed by the admin itself There will ne interaction of user without the database.