## A PROJECT REPORT ON FOOD DELIVERY SYSTEM

*A project report submitted in partial fulfillment of the requirements for the award of the*

*degree of*

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE & ENGINEERING**

### Submitted By

**P.ANUSHA 202P1A05A5**

**N.AKHILANDESWARI 202P1A0589**

**P.SANDHYA 202P1A05A4**

**K.ANUSHA 202P1A0565**

***Under The Esteemed Guidance Of***

**MR.G.SREENIVASA REDDY,M.Tech**

**Associate Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING AN ISO 9001:2015 CERTIFIED INSTITUTION**

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

**(AUTONOMOUS)**

**(Sponsored by Bharathi Educational Society)**

**(Affiliated to** J.N.T.U.A., Anantapuramu**, Approved by AICTE, NewDelhi)**

**Recognized by UGC Under the Sections 2(f)&12(B) of UGC Act, 1956(Accredited by NAAC & NBA)**

**Vidyanagar, Proddatur-516360,Y.S.R.(Dist.),A.P.**

**2020-2024**

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

**(AUTONOMOUS)**

**AN ISO 9001:2015 CERTIFIED INSTITUTION**

**(Sponsored by Bharathi Educational Society)**

**(Affiliated to** J.N.T.U.A., Anantapuram**, Approved by AICTE, New Delhi)**

**Recognized by UGC Under the Sections 2(f)&12(B) of UGC Act, 1956(Accredited by NAAC & NBA)**

**Vidyanagar, Proddatur-516360, Y.S.R.(Dist.), A.P.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the project work entitled “**FOOD DELIVERY SYSTEM”** is a bonafide work of

**P.ANUSHA(202P1A05A5),N.AKHILANDESWARI(202P1A0589),P.SANDHYA(202P1A05A4),**

**K.ANUSHA (202P1A0565)** submitted to **Chaitanya Bharathi Institute of Technology, Proddatur** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING.** The work reported here does not form part of any other thesis on which a degree has been awarded earlier. This is to further certify that they have worked for one semester to prepare their work under our supervision and guidance.

|  |  |
| --- | --- |
| **INTERNAL GUIDE** | **HEAD OF THE DEPARTMENT** |
| **G.SREENIVASA REDDY M.Tech,** | **Dr.Y.D.Rami Reddy M.Tech.ph.D** |
| **Associate Professor** | **Assistant Professor** |

**PROJECT COORDINATOR**

**P.Narasimhaiah** M.Tech,

**Assistant Professor**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

DECLARATION BY THE CANDIDATES

We are P. Anusha, N. Akhilandeswari, P. Sandhya, K. Anusha respective Roll No: (202P1A05A5),(202P1A0589),(202P1A05A4),(202P1A0565) here By declare that the Project Report entitled “FOOD DELIVERY SYSTEM” under the guidance of Mr. G.SREENIVASA REDDY, Associate Professor, Department of CSE is submitted in partial fulfillment of the requirements for the Bachelor of Technology in Computer Science & Engineering award.

This is a record of our bonafide work, and the results embodied in this Project Report have not been reproduced or copied from any source. The results embodied in this Project Report have not been submitted to any other University or Institute for the Award of any other Degree or Diploma.

|  |  |
| --- | --- |
| **P.ANUSHA** | **202P1A05A5** |
| **N.AKHILANDESWARI** | **202P1A0589** |
| **P.SANDHYA** | **202P1A05A4** |
| **K.ANUSHA** | **202P1A0565** |

Dept. of Computer Science & Engineering

**Chaitanya Bharathi Institute of Technology**

Vidyanagar, Proddatur, Y.S.R.(Dist.)



# ACKNOWLEDGEMENT

An endeavor over a long period can be successful only with the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We are extremely thankful to our beloved Chairman **Dr. V. Jayachandra Reddy**, who took a keen interest and encouraged us in every effort throughout this course.

We owe our gratitude to our principal **Dr. G. Sreenivasula Reddy** MTech, Ph.D. for permitting us to use the facilities available to accomplish the project.

We express our heartfelt thanks to **Dr. Y. Dasaratha Rami Reddy**, MTech PhD, Head of Dept CSE for his Kind attention and valuable guidance to us throughout this course.

We also express our deep sense of gratitude towards **Mr.G.Sreenivasa Reddy,** MTech, assistant professor Dept of CSE for his valuable support and guidance in

completing our project.

We express our profound respect and gratitude to our project coordinator

**Mr. P. Narasimhaiah,** MTech, Ph. D. for his valuable support and guidance in completing the project successfully.

We also thank all the teaching & non-teaching staff of the Dept of CSE for their support throughout our B. Tech course.

We express our heartful thanks to **Our Parents** for their valuable support and encouragement in the completion of our course. Also, we express our heartful regards to **Our Friends** for being supportive in completing the project

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Table of Contents** |  | Page. No |
| List of Figures |  | I |
| List of output Screens |  | II |
| Abstract |  | III |
| **1. INTRODUCTION** |  | **1-2** |
| 1.1 Motivation |  | 1 |
| 1.2 Problem Statement |  | 1 |
| 1.3 Objective of the Project |  | 1 |
| 1.4 Scope |  | 2 |
| 1.5 Project Introduction |  | 2 |
| **2. LITERATURE SURVEY** |  | **3-4** |
| 2.1 Relate work |  | 3-4 |
| **3. SYSTEM ANALYSIS** |  | **5-6** |
| 3.1 Existing System |  | 5 |
| 3.2 Disadvantages |  | 5 |
| 3.3 Proposed System |  | 5 |
| 3.4 Advantages |  | 5 |
| 3.5 Workflow of Proposed System |  | 6 |
| **4. REQUIREMENT ANALYSIS** |  | 7-9 |
| 4.1 function and non-functional requirements |  | 7 |
| 4.2 Hardware Requirements |  | 8 |

|  |  |  |
| --- | --- | --- |
| 4.3 Software Requirements |  | 9 |
| 4.4 Architecture |  | 9 |
| **5. SYSTEM DESIGN** |  | **10-18** |
| 5.1 Introduction of Input Design |  | 10-11 |
| 5.2 UML Diagrams |  | 11-15 |
| 5.3 Data Diagrams |  | 15-18 |
| **6. CODING** |  | **18-111** |
| 6.1 Frontend |  | 18-90 |
| 6.2 Backend |  | 91-111 |
| **7. IMPLEMENTATION AND RESULTS** |  | **112-122** |
| 7.1 Modules |  | 112 |
| 7.2 Output Screens |  | 113-122 |
| **8. SYSTEM STUDY AND TESTING** |  | **123-126** |
| 8.1 Feasibility Study |  | 123-124 |
| 8.2 Types of Test & Test Cases |  | 124-126 |
| **9. CONCLUSION** |  | **127** |
| **10. FUTURE ENHANCEMENT** |  | **127** |
| **11. BIBLIOGRAPHY** |  | **127-128** |
| **12. BIODATA** |  | **129** |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Figure No** | **Figure Name** | **Page No** |
| 1 | 4.1.1 | Use case diagram | 11-12 |
| 2 | 4.1.2 | Class diagram | 12-13 |
| 3 | 4.1.3 | Sequence diagram | 13-14 |
| 4 | 4.1.4 | Activity diagram | 14-15 |
| 5 | 4.1.5 | Data Flow diagram | 16-18 |

**LIST OF OUTPUT SCREENS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Figure No** | **Figure Name** | **Page No** |
| 1 | 7.1 | Home Page | 133 |
| 2 | 7.2 | Admin Login | 133 |
| 3 | 7.3 | Admin Home | 114 |
| 4 | 7.4 | Admin view user | 114 |
| 5 | 7.5 | Admin view Restaurant | 115 |
| 6 | 7.6 | Admin view users | 115 |
| 7 | 7.7 | Restaurant Register | 116 |
| 8 | 7.8 | Restaurant Login | 116 |
| 9 | 7.9 | Restaurant Home | 117 |
| 10 | 7.10 | Restaurant Add Food | 117 |
| 11 | 7.11 | Restaurants view Food | 118 |
| 12 | 7.12 | Restaurants View Orders | 118 |
| 13 | 7.13 | User Register | 119 |
| 14 | 7.14 | User Login | 119 |
| 15 | 7.15 | User Home | 120 |
| 16 | 7.16 | User View Food | 120 |
| 17 | 7.17 | UserView Cart | 121 |
| 18 | 7.18 | View Orders | 121 |
| 19 | 7.19 | Payments Forms | 122 |
| 20 | 7.20 | User Orders | 122 |

## ABSTRACT

The Food Order web application is a digital platform designed to streamline the process of ordering food online. Users can browse through a variety of restaurants, view menus, place orders, and make payments conveniently from their devices. The application offers a user-friendly interface, enabling seamless navigation and a smooth ordering experience. Key features include real-time order tracking, customizable preferences, and secure payment options. This web application caters to the growing demand for convenient food delivery services, providing customers with access to a wide range of culinary options while offering restaurants a platform to showcase their offerings and expand their customer base. With its intuitive design and robust functionality, the Food Order web application aims to revolutionize the way people order food online, enhancing convenience, choice, and satisfaction.

### Keywords:

* Food Order
* Web Application
* Online Ordering

## INTRODUCTION

* 1. **PROBLEM STATEMENT:**

The Food Order web application addresses the challenges faced by consumers in the traditional food ordering process, including limited restaurant options, time-consuming phone orders, and lack of real-time order tracking. Additionally, restaurants struggle to reach a broader customer base and streamline their ordering systems. This web application aims to bridge these gaps by offering a centralized platform for convenient online food ordering, benefiting both consumers and restaurants alike.

* 1. **MOTIVATION:**

The Food Order web application aims to address the increasing demand for convenient and efficient food delivery services in today's fast-paced lifestyle. By offering a user-friendly platform for ordering food online, the application seeks to simplify the process of selecting and purchasing meals from a diverse range of restaurants. Motivated by the desire to enhance convenience and choice for users while supporting local eateries, this web application strives to revolutionize the food ordering experience.

* 1. **OBJECTIVE OF THE PROJECT:**

The objective of the Food Order web application is to provide users with a convenient and efficient platform for ordering food online. By offering a user-friendly interface, real-time order tracking, and secure payment options, the application aims to streamline the food ordering process and enhance user satisfaction. Additionally, the application seeks to support restaurants by providing them with a platform to showcase their menus and reach a broader customer base, ultimately promoting growth and success in the food industry.

* 1. **SCOPE:**

The Food Order web application aims to revolutionize the food delivery industry by providing a comprehensive platform for online food ordering. It facilitates seamless interactions between users and restaurants, offering a wide range of features such as menu browsing, order placement, and secure payments.

The scope encompasses user-friendly interfaces, efficient order processing systems, and secure transaction mechanisms to ensure a smooth and convenient experience for both customers and restaurant owners.

* 1. **PROJECT INTRODUCTION:**

In today's fast-paced world, the demand for convenient and efficient food ordering solutions has surged, leading to the development of innovative digital platforms such as the Food Order web application. This introduction explores the evolution of online food ordering services and the growing significance of web applications in facilitating seamless transactions between consumers and restaurants. The Food Order web application serves as a user-friendly interface that empowers users to browse, select, and order food from a diverse range of restaurants with ease. By leveraging advanced technologies and intuitive design principles, this application aims to redefine the online food ordering experience, offering customers unparalleled convenience and choice while enabling restaurants to expand their reach and enhance customer engagement. Through this introduction, we delve into the key features and benefits of the Food Order web application, highlighting its potential to revolutionize the food delivery industry and meet the evolving needs of modern consumers.

## LITERATURE SURVEY

* 1. **Related Work:**

1. **Chen, H., & Liu, Y. (2015). "Impact of Social Media Integration on User Engagement in Food Ordering Web Applications." International Conference on Social Computing, 103-116.**

**Summary:** Chen and Liu investigated the impact of social media integration on user engagement in food ordering web applications. Their study explored how features such as social login, sharing orders on social platforms, and user reviews affect user engagement metrics like time spent on the platform and frequency of orders. The research highlights the significant role of social media integration in enhancing user interaction and loyalty within food ordering applications, emphasizing its potential to drive customer engagement and satisfaction.

1. **Park, J., & Lee, S. (2014). "Enhancing User Satisfaction in Food Ordering Web Applications through Personalization Techniques." International Journal of Human-Computer Interaction, 22(4), 289-302.**

**Summary:** Park and Lee focused on enhancing user satisfaction in food ordering web applications through personalization techniques. Their study investigated the effectiveness of personalized recommendations, tailored menus, and user interface customization in improving user experience and satisfaction. The research findings underscored the importance of personalization in catering to individual preferences, reducing decision fatigue, and increasing overall user satisfaction within food ordering platforms.

1. **Rodriguez, M., & Garcia, A. (2013). "Accessibility Assessment of Food Ordering Web Applications for Visually Impaired Users." International Conference on Computers and Accessibility, 75-88.**

**Summary:** Rodriguez and Garcia conducted an accessibility assessment of food ordering web applications for visually impaired users. Their study evaluated the usability of various features such as screen readers, keyboard navigation, and alternative text descriptions to enhance accessibility for visually impaired individuals. The research highlights the importance of inclusive design practices in ensuring equal access to

food ordering services for all users, emphasizing the need for improvements in accessibility standards and guidelines.

1. **Nguyen, T., & Tran, V. (2012). "Exploring the Impact of User Interface Design on Perceived Trust in Food Ordering Web Applications." Journal of Electronic Commerce Research, 14(2), 165-178.**

**Summary:** Nguyen and Tran explored the impact of user interface design on perceived trust in food ordering web applications. Their study investigated the influence of design elements such as layout, color scheme, and navigation structure on users' trust perceptions and willingness to engage with the platform. The research findings underscored the importance of intuitive and visually appealing interfaces in building trust and credibility among users, thereby increasing adoption and usage of food ordering applications.

1. **Patel, R., & Shah, K. (2011). "Analyzing the Factors Influencing User Adoption of Food Ordering Web Applications: A Case Study of Urban Millennials." International Journal of Electronic Commerce, 9(3), 201-214.**

**Summary:** Patel and Shah conducted a case study on analyzing the factors influencing user adoption of food ordering web applications among urban millennials. Their research investigated various factors such as convenience, price competitiveness, and social influence on millennials' decision-making process regarding the adoption and usage of food ordering platforms. The study findings provide insights into the key drivers and barriers affecting user adoption, offering valuable implications for marketers and developers aiming to target the urban millennial demographic.

## SYSTEM ANALYSIS

### Existing System

The existing system for the Food Order web application involves traditional methods of ordering food, such as phone calls or visiting restaurants in person. These methods often lack convenience and efficiency, leading to longer wait times and limited menu access. The Food Order web application seeks to address these limitations by providing a digital platform for streamlined online ordering and delivery services.

### Disadvantages

* 1. Limited Menu Access
  2. Inconvenience
  3. Lack of Transparency

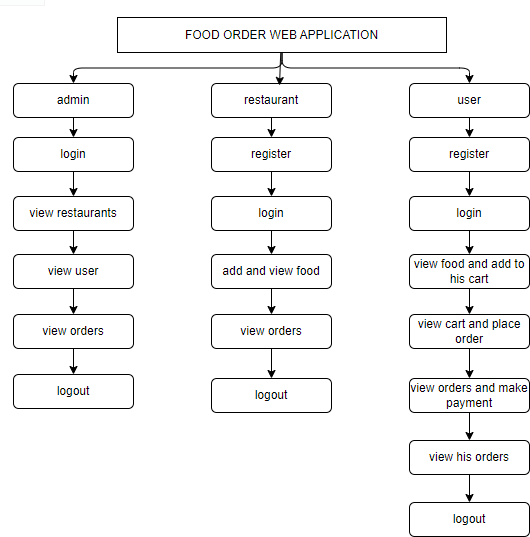
### Proposed System

The proposed system is a comprehensive food ordering web application offering a user-friendly interface for browsing restaurants, viewing menus, and placing orders online. Key features include real-time order tracking, customizable preferences, and secure payment options. This system aims to streamline the food ordering process, enhance user experience, and facilitate seamless transactions for both customers and restaurants.

### Advantages

1. Convenience
2. Enhanced User Experience
3. Increased Efficiency

**Workflow of the Proposed system**



## REQUIREMENT ANALYSIS

### Functional and non-functional requirements

**Functional Requirements:**

1. **User Registration and Authentication:**
   * Users should be able to create accounts and log in securely.
   * Authentication mechanisms such as email verification and password encryption should be implemented.
2. **Food Listings:**
   * The application should display a list of foods available for ordering.
   * Users should be able to add the food items in his cart and can make orders.
3. **Order Placement:**
   * Users should be able to add items to their cart and place orders.
   * Quantity selection, special instructions, and customization options should be available.
   * Users should be able to review their orders before finalizing.
4. **Payment Integration:**
   * Secure payment gateways should be integrated to facilitate online payments.
   * Multiple payment methods like credit/debit cards, digital wallets, and net banking should be supported.

**Non-functional Requirements:**

1. **Performance:**
   * The application should be responsive and load quickly to ensure a smooth user experience.
   * Response times for actions like menu loading, order placement, and payment processing should be minimal.
2. **Scalability:**
   * The system should be able to handle a large number of concurrent users and scale accordingly during peak hours.
3. **Security:**
   * Measures should be in place to prevent unauthorized access, data breaches, and fraud.
   * Secure communication protocols (HTTPS) and encryption techniques should be employed to protect user data.
4. **Reliability:**
   * The application should be stable and available at all times, minimizing downtime and service interruptions.
   * Backup and recovery mechanisms should be implemented to ensure data integrity.
5. **Usability:**
   * The user interface should be intuitive and easy to navigate, catering to users of all technical abilities.
   * Accessibility standards should be followed to ensure inclusivity for users with disabilities.
6. **Compatibility:**
   * The application should be compatible with a variety of devices and web browsers to reach a wider audience.
   * Responsive design principles should be applied to ensure optimal viewing across different screen sizes.

## SYSTEMCONFIGURATION

**Hardware Requirements**

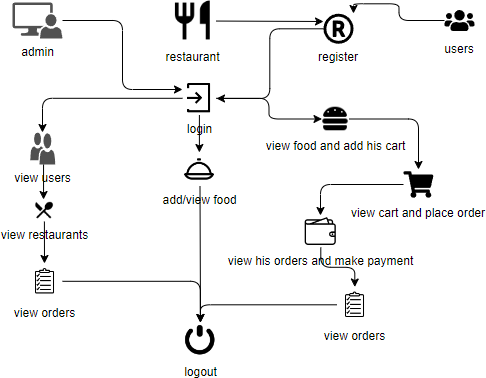
Processor -I3/Intel Processor

Hard Disk -160GB

Key Board -Standard Windows Keyboard

|  |  |
| --- | --- |
| Mouse | -Two or Three Button Mouse |
| Monitor | -SVGA |
| RAM  **Software Requirements:** | -8GB |
| Operating System | : Windows7/8/10 |
| Server-side Script | : IntelliJ |
| Programming Language | : Java script |
| IDE/Workbench | : VS Code |
| Database | : MySQL 6.0 |
| Client Side | :Reactjs |

### Architecture:



## SYSTEM DESIGN

**Introduction of Input Design:**

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding −
  + What are the inputs needed for the system?
  + How end users respond to different elements of forms and screens.

### Objectives for Input Design:

The objectives of input design are −

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

**Output Design:**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design:

The objectives of input design are:

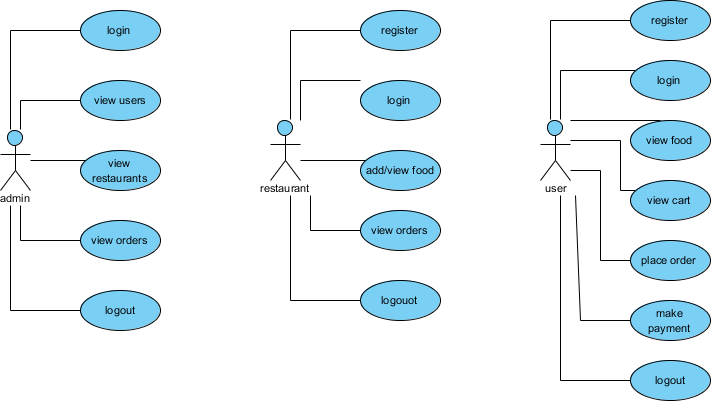
* To develop output design that serves the intended purpose and eliminates the production of unwanted output.
* To develop the output design that meets the end user’s requirements.
* To deliver the appropriate quantity of output.
* To form the output in appropriate format and direct it to the right person.
* To make the output available on time for making good decisions.

**UML Diagrams:**

### Use Case Diagram:

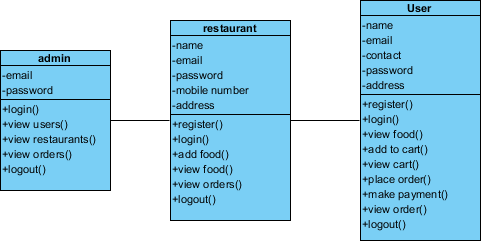
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor .Roles of the actors in the system can be depicted.

### Use Case Diagram:



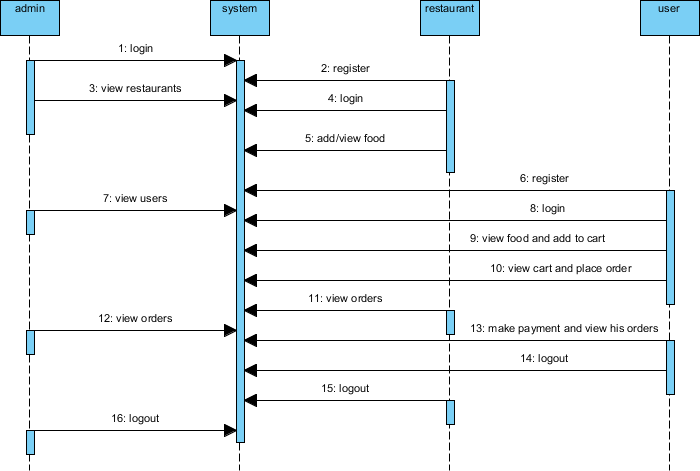
**Class Diagram:**

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**Sequence Diagram:**

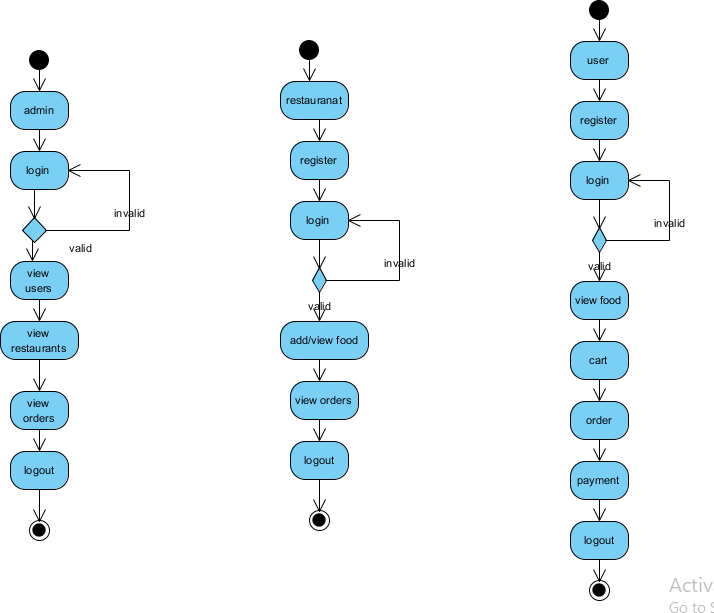
A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



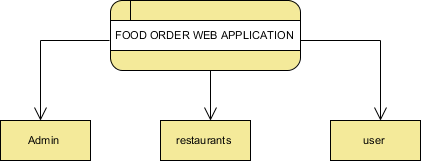
**Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

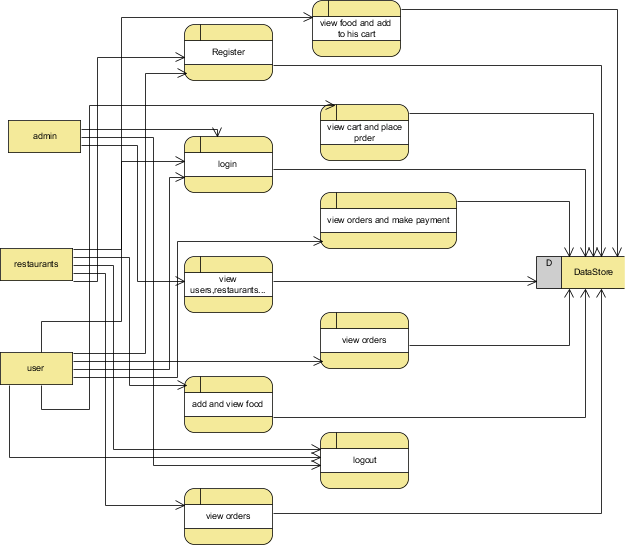
### Activity diagram:



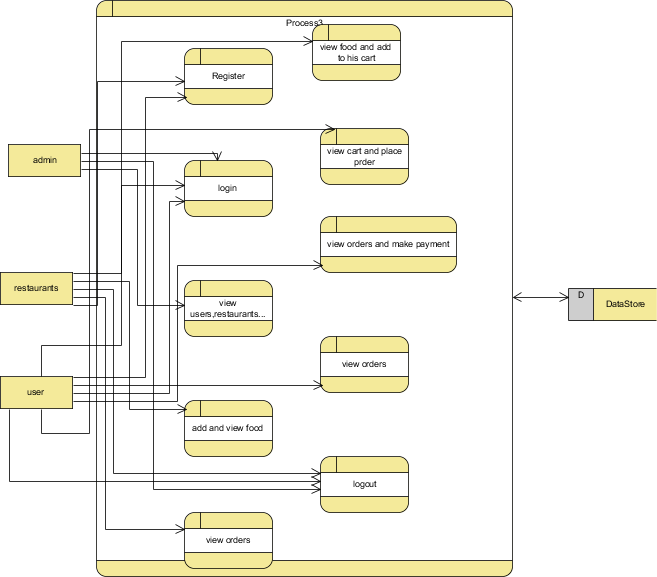
**Data Flow Diagram: Level 0:**



Level 1 Diagram:



**Level 2 diagram:**



## CODING

* 1. **Front End**

# Home.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style> body, html {

background-image: url("https://img.freepik.com/premium-vector/trendy-minimalistic- food-delivery-service-online-food-order-application-banner-design-template\_420121- 273.jpg");

background-size: cover; height: 100%;

width: 100%;

background-repeat: no-repeat;

}

.navbar { display: flex;

justify-content: space-between; align-items: center; background-color: #f34609; color: white;

padding: 10px 20px;

}

.navbar .logo { font-size: 24px;

}

.nav-links {

list-style: none;

}

.nav-links li { display: inline; margin-left: 20px;

}

.nav-links a {

text-decoration: none; color: white;

}

footer {

position: absolute; bottom: 0;

color: black; width: 100vw; height: 20px; padding: 20px; text-align: center;

}

@media only screen and (max-width: 600px) {

.navbar {

flex-direction: column; align-items: flex-start;

}

.nav-links li { display: block;

margin: 10px 0;

/\* Adjusted spacing \*/

}

}

</style>

</head>

<body>

<header>

<nav class="navbar">

<div class="logo">Foodie's Heaven</div>

<ul class="nav-links">

<li><a href="home.html">HOME</a></li>

<li><a href="adminLogin.html">ADMIN</a></li>

<li><a href="restaurantRegistration.html">RESTAURANT</a></li>

<li><a href="userRegistration.html">USER</a></li>

</ul>

</nav>

</header>

<footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer>

</body>

</html>

# Adminhome.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://plus.unsplash.com/premium\_photo-1661481417385- de9bdd899a11?q=80&w=1460&auto=format&fit=crop&ixlib=rb- 4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh;

} .navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0;

display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap;

} .navbar h3 {

margin-left: 20px; /\* Adjusts the margin between h3 and the edge of the navbar \*/

} .nav-links {

list-style-type: none; margin: 0;

padding: 0; display: flex;

} .nav-links li { margin: 0 10px;

} .nav-links li a {

text-decoration: none; color: #333;

font-size: 16px; font-weight: bold;

}

nav-links li a:hover { color: #007bff;

}

footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto;

}

</style>

</head>

<body>

<header>

<nav class="navbar">

<h3>ADMINISTRATOR</h3>

<ul class="nav-links">

<li><a href="adminHome.html">HOME</a></li>

<li><a href="adminViewUsers.html">VIEW USERS</a></li>

<li><a href="adminViewRestaurant.html">VIEW RESTAURANT</a></li>

<li><a href="adminViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul>

</nav>

</header>

<div>

<marquee>

<h1>WELCOME ADMIN</h1>

</marquee>

</div>

<footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer>

</body>

</html>

# Adminlogin.html

<!DOCTYPE html>

<html>

<head>

<title>ADMIN LOGIN</title>

<link rel="stylesheet" type="text/css" href="slide navbar style.css" />

</head>

<style>

/\* CSS Reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

} body {

font-family: "Poppins", sans-serif;

background-image: url("https://img.freepik.com/free-photo/business-man-financial- inspector-secretary-making-report-calculating-checking-balance-internal-revenue-service- inspector-checking-document-audit-concept\_1423- 127.jpg?t=st=1709635965~exp=1709639565~hmac=465c76ed35bd4d69f096fd05ba3261d4 0856b24b063a42c6362a871ff5d1f5e1&w=900");

background-repeat: no-repeat; background-size: cover; display: flex;

justify-content: center; align-items: center; min-height: 100vh;

/\* Adjust to your needs \*/

}.screen-1 {

background: rgba(124, 165, 212, 0.7);

/\* Adjust opacity as needed \*/ padding: 2em;

border-radius: 30px;

box-shadow: 0 0 2em rgba(124, 165, 212, 0.7); display: flex;

flex-direction: column; align-items: center;

}.screen-1 h1 { color: #fff;

} .screen-1 .email,

.screen-1 .password {

background: rgba(255, 255, 255, 0.8);

/\* Adjust opacity as needed \*/

box-shadow: 0 0 2em rgba(124, 165, 212, 0.7); padding: 1em;

border-radius: 20px;

color: #4d4d4d; margin-top: 1em;

/\* Adjust spacing \*/

} .screen-1 .email input,

.screen-1 .password input { outline: none;

border: none; background: transparent;

/\* Make input background transparent \*/

} .screen-1 .email ion-icon,

.screen-1 .password ion-icon { color: #4d4d4d;

}.screen-1 .login { padding: 1em; color: white; border: none; border-radius: 30px; font-weight: 600;

background-color: #58629c;

/\* Adjust background color \*/ cursor: pointer;

margin-top: 1em;

/\* Adjust spacing \*/

}

footer { color: #000;

text-align: center; position: absolute; bottom: 0;

width: 100%;

background: rgba(255, 255, 255, 0.8);

/\* Adjust opacity as needed \*/ padding: 10px 0;

/\* Adjust padding as needed \*/

}

</style>

<body>

<div class="screen-1">

<h1>ADMIN LOGIN</h1>

<br>

<form id="adminLoginForm">

<div class="email">

<label for="email">Email Address</label>

<div class="sec-2">

<ion-icon name="mail-outline"></ion-icon>

<input type="email" id="email" name="email" placeholder="enter your email" />

</div>

</div>

<div class="password">

<label for="password">Password</label>

<div class="sec-2">

<ion-icon name="lock-closed-outline"></ion-icon>

<input class="pas" type="password" id="password" name="password" placeholder="enter your password" />

<ion-icon class="show-hide" name="eye-outline"></ion-icon>

</div>

</div>

<center>

<button type="submit" class="login">Login</button>

</center>

</form>

</div> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer>

<script>

async function adminLogin(event) { event.preventDefault();

const email = document.getElementById('email').value;

const password = document.getElementById('password').value; const response = await fetch('http://localhost:8080/adminLogin', {

method: 'POST', headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ email, password }),

}); if (response.ok) {

const data = await response.json(); console.log('Login successful:', data); window.location.assign("adminHome.html");

} else {

console.error('Login failed:', response.status, response.statusText); alert('Login failed : Invalid credentials');

}

}

document.addEventListener('DOMContentLoaded', (event) => {

const adminLoginForm = document.getElementById('adminLoginForm'); adminLoginForm.addEventListener('submit', adminLogin);

});

</script>

</body>

# Adminvieworders.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style> body {

font-family: "Poppins", sans-serif;

background-image: url("https://plus.unsplash.com/premium\_photo-1661481417385- de9bdd899a11?q=80&w=1460&auto=format&fit=crop&ixlib=rb- 4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh;

}.navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap;

} .navbar h3 { margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/

} .nav-links {

list-style-type: none; margin: 0;

padding: 0; display: flex;

} .nav-links li { margin: 0 10px;

} .nav-links li a {

text-decoration: none; color: #333;

font-size: 16px; font-weight: bold;

}

.nav-links li a:hover { color: #007bff;

}

footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto;

} table { width: 80%;

border-collapse: collapse; margin-top: 20px;

}

th,

td {

border: 1px solid #ddd; padding: 8px;

text-align: left;

}

th {

background-color: #f2f2f2;

}

</style>

</head> <body>

<header>

<nav class="navbar">

<h3>ADMINISTRATOR</h3>

<ul class="nav-links">

<li><a href="adminHome.html">HOME</a></li>

<li><a href="adminViewUsers.html">VIEW USERS</a></li>

<li><a href="adminViewRestaurant.html">VIEW RESTAURANT</a></li>

<li><a href="adminViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul>

</nav>

</header> <main>

<h1>Orders List</h1>

<table id="ordersList">

<thead>

<tr>

<th>User Name</th>

<th>Restaurant Name</th>

<th>Food Name</th>

<th>Quantity</th>

<th>Amount</th>

</tr>

</thead>

<tbody>

<!-- Data will be dynamically added here -->

</tbody>

</table>

</main> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer> <script>

async function fetchOrders() { try {

const response = await fetch(`http://localhost:8080/payment`); const data = await response.json();

console.log(data); displayOrders(data.Payments);

} catch (error) {

console.error("Error fetching orders:", error);

}

}function displayOrders(payments) {

const ordersList = document.querySelector("#ordersList tbody"); ordersList.innerHTML = "";

payments.forEach((payment) => { const order = payment.order; console.log(order, "order");

const row = document.createElement("tr"); row.innerHTML = `

<td>${order.user.name}</td>

<td>${order.restaurant.restaurantName}</td>

<td>${order.food.name}</td>

<td>${order.quantity}</td>

<td>${order.amount}</td>

`; ordersList.appendChild(row);

});

} fetchOrders();

</script>

</body>

</html>

# AdminviewRestuarant.html

<!DOCTYPE html>

<html lang="en"><head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://plus.unsplash.com/premium\_photo-1661481417385- de9bdd899a11?q=80&w=1460&auto=format&fit=crop&ixlib=rb- 4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D");

background-repeat: no-repeat;

background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh;

} .navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap;

}.navbar h3 {

margin-left: 20px; /\* Adjusts the margin between h3 and the edge of the navbar \*/

} .nav-links {

list-style-type: none; margin: 0;

padding: 0; display: flex;

} .nav-links li { margin: 0 10px;

} .nav-links li a {

text-decoration: none; color: #333;

font-size: 16px; font-weight: bold;

} .nav-links li a:hover { color: #007bff;

} footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto;

} table { width: 80%;

border-collapse: collapse; margin-top: 20px;

}

th, td {

border: 1px solid #ddd; padding: 8px;

text-align: left;

}

th {

}

background-color: #f2f2f2;

</style>

</head>

<body>

<header>

<nav class="navbar">

<h3>ADMINISTRATOR</h3>

<ul class="nav-links">

<li><a href="adminHome.html">HOME</a></li>

<li><a href="adminViewUsers.html">VIEW USERS</a></li>

<li><a href="adminViewRestaurant.html">VIEW RESTAURANT</a></li>

<li><a href="adminViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul>

</nav>

</header> <main>

<h1>RESTAURANT LIST</h1>

<table id="restaurantList">

<thead>

<tr>

<th>RESTAURANT NAME</th>

<th>EMAIL</th>

<th>MOBILE NUMBER</th>

<th>ADDRESS</th>

</tr>

</thead>

<tbody>

<!-- Users will be displayed here -->

</tbody>

</table>

</main>

<footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer>

<script>

async function fetchUsers() { try {

const response = await fetch('http://localhost:8080/restaurant'); // Replace with your API endpoint

const restaurants = await response.json(); console.log(restaurants,"restaurants"); displayUsers(restaurants.Restaurants);

} catch (error) {

console.error('Error fetching restaurants:', error);

}

} function displayUsers(restaurants) {

const restaurantList = document.querySelector('#restaurantList tbody'); restaurantList.innerHTML = '';

restaurants.forEach(restaurants => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${restaurants.restaurantName}</td>

<td>${restaurants.email}</td>

<td>${restaurants.mobileNumber}</td>

<td>${restaurants.address}</td>

`; restaurantList.appendChild(row);

});

}

window.onload = fetchUsers;

</script>

</body></html>

# Adminviewusers.html

<!DOCTYPE html>

<html lang="en"><head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://plus.unsplash.com/premium\_photo-1661481417385- de9bdd899a11?q=80&w=1460&auto=format&fit=crop&ixlib=rb- 4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh;

} .navbar {

width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap;

}

.navbar h3 {

margin-left: 20px; /\* Adjusts the margin between h3 and the edge of the navbar \*/

} .nav-links {

list-style-type: none; margin: 0;

padding: 0; display: flex;

} .nav-links li { margin: 0 10px;

}

.nav-links li a {

text-decoration: none; color: #333;

font-size: 16px; font-weight: bold;

}

.nav-links li a:hover {

color: #007bff;

} footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

margin-top: auto;

}table {

width: 80%;

border-collapse: collapse; margin-top: 20px;

}th, td {

border: 1px solid #ddd; padding: 8px;

text-align: left;

}

th {

}

background-color: #f2f2f2;

</style>

</head><body>

<header>

<nav class="navbar">

<h3>ADMINISTRATOR</h3>

<ul class="nav-links">

<li><a href="adminHome.html">HOME</a></li>

<li><a href="adminViewUsers.html">VIEW USERS</a></li>

<li><a href="adminViewRestaurant.html">VIEW RESTAURANT</a></li>

<li><a href="adminViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul>

</nav>

</header> <main>

<h1>USERS LIST</h1>

<table id="userList">

<thead>

<tr>

<th>USERNAME</th>

<th>EMAIL</th>

<th>MOBILE NUMBER</th>

<th>ADDRESS</th>

</tr>

</thead>

<tbody>

<!-- Users will be displayed here -->

</tbody>

</table>

</main> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer>

<script>

async function fetchUsers() { try {

const response = await fetch('http://localhost:8080/user'); // Replace with your API

endpoint

const users = await response.json(); console.log(users,"users"); displayUsers(users.User);

} catch (error) {

console.error('Error fetching users:', error);

}

}function displayUsers(users) {

const userList = document.querySelector('#userList tbody'); userList.innerHTML = '';

users.forEach(user => {

const row = document.createElement('tr'); row.innerHTML = `

<td>${user.name}</td>

<td>${user.email}</td>

<td>${user.mobileNumber}</td>

<td>${user.address}</td>

`; userList.appendChild(row);

});

} window.onload = fetchUsers;

</script>

</body>

</html>

# Restuarantsaddfood.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://img.freepik.com/free-photo/side-view-cook-making- delicious-pasta\_23- 2150690631.jpg?t=st=1709706794~exp=1709710394~hmac=27d97ecaf979f0fcb61ab9fb45 e556545695041d94a193ef3d59c3813677fa77&w=740");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh;

}.navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between;

align-items: center; flex-wrap: wrap;

}.navbar h3 { margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/

} .nav-links {

list-style-type: none; margin: 0;

padding: 0; display: flex;

} .nav-links li { margin: 0 10px;

} .nav-links li a {

text-decoration: none; color: #333;

font-size: 16px; font-weight: bold;

} .nav-links li a:hover { color: #007bff;

}footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto;

} .toast {

visibility: hidden; min-width: 250px; margin-left: -125px;

background-color: #333; color: #fff;

text-align: center; border-radius: 2px; padding: 16px; position: fixed;

z-index: 1;

left: 50%; bottom: 30px; font-size: 17px;

} .toast.show { visibility: visible;

-webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s; animation: fadein 0.5s, fadeout 0.5s 2.5s;

} @-webkit-keyframes fadein { from {

bottom: 0;

opacity: 0;

} to {

bottom: 30px; opacity: 1;

}

}@keyframes fadein { from {

bottom: 0;

opacity: 0;

} to {

bottom: 30px; opacity: 1;

}

}@-webkit-keyframes fadeout { from {

bottom: 30px; opacity: 1;

}

to {

bottom: 0;

opacity: 0;

}

} @keyframes fadeout { from {

bottom: 30px; opacity: 1;

} to {

bottom: 0;

opacity: 0;

}

}form {

width: 40%;

background-color: rgba(255, 255, 255, 0.8); padding: 20px;

border-radius: 10px; margin-top: 50px;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.3);

} form input, form textarea, form select {

margin-bottom: 10px; width: 100%; padding: 10px;

border: 1px solid #ccc; border-radius: 5px;

box-sizing: border-box;

} form button {

background-color: #007bff; color: #fff;

border: none; padding: 10px 20px; border-radius: 5px;

cursor: pointer;

} form button:hover { background-color: #0056b3;

}

</style>

</head>

<body>

<header>

<nav class="navbar">

<h3>RESTAURANT</h3>

<ul class="nav-links">

<li><a href="restaurantHome.html">HOME</a></li>

<li><a href="restaurantAddFood.html">ADD FOOD</a></li>

<li><a href="restaurantViewFood.html">VIEW FOOD</a></li>

<li><a href="restaurantViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul>

</nav>

</header><center>

<div class="container">

<form id="addFoodForm" action="addFoodServlet" method="post">

<h2>ADD FOOD</h2> <label for="restaurantId">RESTAURANT ID:</label>

<input type="number" id="restaurantId" name="restaurantId" readonly>

<label for="name">FOOD NAME:</label>

<input type="text" id="name" name="name" required>

<label for="image">IMAGE:</label>

<input type="file" id="image" name="image" required> <label for="foodDescription">DESCRIPTION:</label>

<textarea id="foodDescription" name="foodDescription" required></textarea>

<label for="price">PRICE:</label>

<input type="number" id="price" name="price" required step="0.01"> <center>

<button type="button" onclick="addFood()">Add Food</button>

</center>

</form>

</div>

</center> <div id="toast" class="toast"></div> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer> <script>

const storedRestaurant = JSON.parse(localStorage.getItem('restaurant')); const restaurantId = storedRestaurant.id;

console.log('Restaurant ID:', restaurantId); document.getElementById('restaurantId').value = restaurantId;

function showToast(message, success = true) { const toast = document.getElementById('toast'); toast.textContent = message;

toast.className = 'toast show'; if (!success) {

toast.style.backgroundColor = 'red';

}

setTimeout(() => { toast.className = 'toast';

}, 3000);

}

function addFood() {

const form = document.getElementById('addFoodForm'); const formData = new FormData(form);

fetch('http://localhost:8080/food/addFood', { method: 'POST',

body: formData

})

.then(response => { if (!response.ok) {

throw new Error('Network response was not ok');

}

return response.json();

})

.then(data => {

console.log('Food added successfully:', data); showToast('Food added successfully'); window.location.reload();

})

.catch(error => {

console.error('There was a problem adding the food:', error); showToast('Failed to Add the Food');

}); }

</script>

</body>

</html>

# Restuaranthome.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://img.freepik.com/free-photo/side-view-cook-making- delicious-pasta\_23- 2150690631.jpg?t=st=1709706794~exp=1709710394~hmac=27d97ecaf979f0fcb61ab9fb45 e556545695041d94a193ef3d59c3813677fa77&w=740");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center;

align-items: center; min-height: 100vh;

} .navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap;

}.navbar h3 {

margin-left: 20px; /\* Adjusts the margin between h3 and the edge of the navbar \*/

} .nav-links {

list-style-type: none; margin: 0;

padding: 0; display: flex;

} .nav-links li { margin: 0 10px;

} .nav-links li a {

text-decoration: none; color: #333;

font-size: 16px; font-weight: bold;

} .nav-links li a:hover { color: #007bff;

} footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto;

}

</style>

</head>

<body>

<header>

<nav class="navbar">

<h3>RESTAURANT</h3>

<ul class="nav-links">

<li><a href="restaurantHome.html">HOME</a></li>

<li><a href="restaurantAddFood.html">ADD FOOD</a></li>

<li><a href="restaurantViewFood.html">VIEW FOOD</a></li>

<li><a href="restaurantViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul>

</nav>

</header> <div>

<marquee>

<h1 id="welcomeMessage" style="color: aliceblue;"></h1>

</marquee>

</div> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer> <script> document.addEventListener('DOMContentLoaded', () => {

const welcomeMessage = document.getElementById('welcomeMessage');

// Get the username from localStorage

const restaurant = JSON.parse(localStorage.getItem('restaurant')); console.log(restaurant,"restaurant");

// Check if user is logged in and username exists if (restaurant && restaurant.restaurantName) {

welcomeMessage.textContent = `WELCOME ${restaurant.restaurantName}`;

} else {

welcomeMessage.textContent = 'WELCOME';

}

});

</script>

</body>

</html>

# RestuarantRegistration.html:

<!DOCTYPE html><html><head>

<title>User Registration</title>

<link rel="stylesheet" type="text/css" href="slide navbar style.css" />

</head>

<style> body { margin: 0;

padding: 0; display: flex;

justify-content: center; align-items: center; min-height: 100vh;

font-family: 'Jost', sans-serif;

background-image: url("https://img.freepik.com/free-photo/restaurant-interior\_1127- 3394.jpg?t=st=1709616409~exp=1709620009~hmac=5baea8d5895802fac2d1f5582420376 8ab40198a4ee430b0d2eaed95be78077e&w=900");

background-repeat: no-repeat; background-size: 100% 100%; }

.main {

width: 350px; height: 600px;

background: rgba(241, 208, 194, 0.726); overflow: hidden;

border-radius: 10px;

box-shadow: 5px 20px 50px #000;

}#chk { display: none;

}.signup { position: relative; width: 100%;

height: 100%;

}

label { color: #fff;

font-size: 2.3em; justify-content: center; display: flex;

margin: 60px; font-weight: bold; cursor: pointer;

transition: 0.5s ease-in-out;

} input { width: 60%; height: 20px;

background: #e0dede; justify-content: center; display: flex;

margin: 20px auto; padding: 10px; border: none; outline: none; border-radius: 5px;

} button { width: 60%; height: 40px;

margin: 10px auto; justify-content: center; display: block;

color: #fff; background: #f34609; font-size: 1em;

font-weight: bold; margin-top: 20px; outline: none;

border: none; border-radius: 5px;

transition: 0.2s ease-in; cursor: pointer;

} button:hover { background: #f34609;

}.login { height: 600px;

background: #eee;

border-radius: 60% / 10%; transform: translateY(-180px); transition: 0.8s ease-in-out;

}.login label { color: #f34609;

transform: scale(0.6);

} #chk:checked~.login { transform: translateY(-600px);

}#chk:checked~.login label { transform: scale(1);

}#chk:checked~.signup label { transform: scale(0.8);

} .toast { visibility: hidden;

min-width: 250px; margin-left: -125px; background-color: #333; color: #fff;

text-align: center; border-radius: 2px; padding: 16px; position: fixed;

z-index: 1;

left: 50%; bottom: 30px; font-size: 17px;

} .toast.show { visibility: visible;

-webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s; animation: fadein 0.5s, fadeout 0.5s 2.5s;

} @-webkit-keyframes fadein { from {

bottom: 0;

opacity: 0;

}to {

bottom: 30px; opacity: 1;

}

} @keyframes fadein { from {

bottom: 0;

opacity: 0;

} to {

bottom: 30px; opacity: 1;

}

} @-webkit-keyframes fadeout { from {

bottom: 30px; opacity: 1;

} to { bottom: 0;

opacity: 0;

}

} @keyframes fadeout { from {

bottom: 30px; opacity: 1;

} to { bottom: 0;

opacity: 0;

}

} footer {

position: absolute; bottom: 0;

color: white; width: 100vw; height: 20px; padding: 20px; text-align: center;

}

</style><body>

<div class="main">

<input type="checkbox" id="chk" aria-hidden="true" />

<div class="signup">

<form id="signupForm">

<label for="chk" aria-hidden="true">Sign up</label>

<input type="text" name="restaurantName" placeholder="Restaurant name" required />

<input type="email" name="email" placeholder="Email" required />

<input type="password" name="password" placeholder="Password" required /><input type="text" name="mobileNumber" placeholder="Mobile number" required />

<input type="text" name="address" placeholder="Address" required />

<button type="submit">Sign up</button>

</form>

</div><div class="login">

<form id="loginForm">

<label for="chk" aria-hidden="true">Login</label>

<input type="email" name="email" placeholder="Email" required />

<input type="password" name="password" placeholder="Password" required />

<button type="submit">Login</button>

</form>

</div>

</div>

<div id="toast" class="toast"></div><script> function showToast(message, success = true) { const toast = document.getElementById('toast'); toast.textContent = message;

toast.className = 'toast show'; if (!success) {

toast.style.backgroundColor = 'red'; // Change toast background color for failure

}

setTimeout(() => {

toast.className = 'toast';

}, 3000);

} document.addEventListener('DOMContentLoaded', () => { const signupForm = document.getElementById('signupForm'); const loginForm = document.getElementById('loginForm');

signupForm.addEventListener('submit', async (event) => { event.preventDefault();

const formData = new FormData(signupForm);

const userData = Object.fromEntries(formData.entries()); try {

// Send signup data to your API endpoint

const response = await fetch('http://localhost:8080/restaurant', { method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(userData)

});

// Handle response if (response.ok) {

// Signup successful, display toast or perform any action showToast('Signup successful');

console.log('Signup successful');

} else {

// Signup failed, display error message or handle appropriately showToast('Signup failed', false);

console.error('Signup failed:', response.statusText);

}

} catch (error) {

console.error('Error during signup:', error);

}

}); loginForm.addEventListener('submit', async (event) => { event.preventDefault();

const formData = new FormData(loginForm);

const userData = Object.fromEntries(formData.entries()); try {

// Send login data to your API endpoint

const response = await fetch('http://localhost:8080/restaurantLogin', { method: 'POST',

headers: {

'Content-Type': 'application/json'

}, body: JSON.stringify(userData)

});

if (response.ok) { console.log('Login successful'); const data = await response.json();

localStorage.setItem('restaurant', JSON.stringify(data)); showToast('Login successful', true); window.location.assign("restaurantHome.html");

} else {console.error('Login failed:', response.statusText); showToast('Login failed', false);

} } catch (error) {console.error('Error during login:', error);

} });});

</script></body></html>

# RestuarantViewFood.html:

<!DOCTYPE html>

<html lang="en"><head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://img.freepik.com/free-photo/side-view-cook- making-delicious-pasta\_23- 2150690631.jpg?t=st=1709706794~exp=1709710394~hmac=27d97ecaf979f0fcb61ab9fb 45e556545695041d94a193ef3d59c3813677fa77&w=740");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center;

min-height: 100vh;} .container { display: flex;

flex-wrap: wrap;

justify-content: space-around; padding: 20px;

}.food-card {

background-color: #ffffff; border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); margin: 20px;

overflow: hidden;

width: calc(33.333% - 40px); transition: transform 0.3s ease; }

.food-card:hover {

transform: translateY(-5px); }

.food-image {

width: 1000px; height: 200px;

object-fit: cover; } .food-details { padding: 15px; }

.food-name {

color: #333;

margin: 0 0 10px 0; }food-description {

color: #666; font-size: 14px; margin: 0; }

/\* Responsive adjustments \*/ @media (max-width: 768px) {

.food-card {

width: calc(50% - 40px); } } @media (max-width: 480px) {

.food-card {

width: calc(100% - 40px);

} } .navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0;

display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; }

.navbar h3 {

margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/ }

.nav-links {

list-style-type: none; margin: 0;

padding: 0;

display: flex; } .nav-links li { margin: 0 10px;} .nav-links li a { text-decoration: none;

color: #333; font-size: 16px;

font-weight: bold; } .nav-links li a:hover { color: #007bff;} footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto }

</style>

</head>

<body>

<header>

<nav class="navbar">

<h3>RESTAURANT</h3>

<ul class="nav-links">

<li><a href="restaurantHome.html">HOME</a></li>

<li><a href="restaurantAddFood.html">ADD FOOD</a></li>

<li><a href="restaurantViewFood.html">VIEW FOOD</a></li>

<li><a href="restaurantViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li> </ul>

</nav> </header> <div class="container" id="foodList">

<!-- Food cards will be dynamically added here -->

</div> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer>

<script>

document.addEventListener('DOMContentLoaded', function () {

const storedRestaurant = JSON.parse(localStorage.getItem('restaurant')); const restaurantId = storedRestaurant.id;

// Fetch data from the API and display the food list fetch(`http://localhost:8080/food/restaurant/${restaurantId}`)

.then(response => { if (!response.ok) {

throw new Error('Network response was not ok'); } return response.json(); })

.then(data => { displayFoodList(data.Food); })

.catch(error => {

console.error('There was a problem fetching the data:', error); }); function createFoodCard(food) {

const card = document.createElement('div'); card.classList.add('food-card'); card.innerHTML = `

<img src="${food.image}" alt="${food.name}" class="food-image"><div class="food-details">

<h3 class="food-name">${food.name}</h3>

<h3 class="food-description">${food.foodDescription}</h3>

<p class="food-price">PRICE: ${food.price}</p>

</div> `; return card; } // Function to display the food cards function displayFoodList(foods) {

const foodListContainer = document.getElementById('foodList'); foods.forEach(food => {

const card = createFoodCard(food); foodListContainer.appendChild(card); }); });

</script>

</body></html>

# RestuarantViewOrders.html:

<!DOCTYPE html><html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://plus.unsplash.com/premium\_photo- 1661481417385-de9bdd899a11?q=80&w=1460&auto=format&fit=crop&ixlib=rb- 4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"

);

.navbar {

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh; }

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; }

.navbar h3 {

margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/ }

.nav-links {

list-style-type: none; margin: 0;

padding: 0;

display: flex; } .nav-links li {

margin: 0 10px;}

.nav-links li a {

text-decoration: none; color: #333;

font-size: 16px;

font-weight: bold; } .nav-links li a:hover { color: #007bff; } footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto; } table {

width: 80%;

border-collapse: collapse; margin-top: 20px; } th,

td {

border: 1px solid #ddd; padding: 8px;

text-align: left;

} th {

background-color: #f2f2f2; }

</style></head><body>

<header>

<nav class="navbar">

<h3>RESTAURANT</h3>

<ul class="nav-links">

<li><a href="restaurantHome.html">HOME</a></li>

<li><a href="restaurantAddFood.html">ADD FOOD</a></li>

<li><a href="restaurantViewFood.html">VIEW FOOD</a></li>

<li><a href="restaurantViewOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li> </ul>

</nav> </header> <main>

<h1>Orders List</h1>

<table id="ordersList"> <thead>

<tr>

<th>ID</th>

<th>Customer Name</th>

<th>Restaurant Name</th>

<th>Item name</th>

<th>Quantity</th>

<th>Price</th>

</tr>

</thead>

<tbody>

<!-- Data will be dynamically added here -->

</tbody>

</table>

</main><footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer> <script>

async function fetchOrders() { try {

const storedRestaurant = JSON.parse(localStorage.getItem('restaurant')); const restaurantId = storedRestaurant.id;

const response = await fetch(`http://localhost:8080/orders/restaurant/${restaurantId}`);

const data = await response.json(); console.log(data); displayOrders(data.Orders);

} catch (error) {

console.error('Error fetching orders:', error); }} function displayOrders(orders) {

const ordersList = document.querySelector('#ordersList tbody'); ordersList.innerHTML = '';

orders.forEach(order => {

const row = document.createElement('tr'); row.innerHTML = `

<td>${order.id}</td>

<td>${order.user.name}</td>

<td>${order.restaurant.restaurantName}</td>

<td>${order.food.name}</td>

<td>${order.quantity}</td>

<td>${order.amount}</td>

`;

ordersList.appendChild(row); }); } window.onload = fetchOrders;

</script></body></html>

# userHome.html:

<!DOCTYPE html>

<html lang="en"><head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://img.freepik.com/free-photo/girl-cafe-reads- something-tablet\_1304- 5538.jpg?t=st=1709707633~exp=1709711233~hmac=d395970c02b834f1ef28faaec45f2d 1c656a3067c0134304104c8d9a3ad766ee&w=740");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center;

min-height: 100vh;} .navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; }

.navbar h3 {

margin-left: 20px; } .nav-links { list-style-type: none;

margin: 0;

padding: 0;

display: flex; }

.nav-links li {

margin: 0 10px; } .nav-links li a { text-decoration: none;

color: #333; font-size: 16px;

font-weight: bold; } .nav-links li a:hover { color: #007bff;

}footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto; }

</style>

</head><head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>User Page</title></head><body>

<header>

<nav class="navbar">

<h3>USER</h3>

<ul class="nav-links">

<li><a href="userHome.html">HOME</a></li>

<li><a href="userViewFood.html">VIEW FOOD</a></li>

<li><a href="userViewCart.html">VIEW CART</a></li>

<li><a href="userViewOrders.html">MAKE PAYMENT</a></li>

<li><a href="userOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul> </nav></header> <div>

<marquee>

<h1 id="welcomeMessage"></h1>

</marquee></div> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer> <script>

document.addEventListener('DOMContentLoaded', () => {

const welcomeMessage = document.getElementById('welcomeMessage');

// Get the username from localStorage

const user = JSON.parse(localStorage.getItem('user'));

console.log(user,"user");

// Check if user is logged in and username exists if (user && user.name) {

welcomeMessage.textContent = `WELCOME ${user.name}`;

} else {

welcomeMessage.textContent = 'WELCOME';} });

</script></body></html>

# userMakePayment.html:

<!DOCTYPE html><html lang="en"> <head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Payment</title>

<style> body {

font-family: Arial, sans-serif; margin: 0;

padding: 0; } h1 {

text-align: center; } #paymentForm { width: 400px;

margin: 0 auto; } label {

display: block;

margin-bottom: 5px;} input[type="text"], input[type="number"] {

width: 100%; padding: 8px;

margin-bottom: 10px;

box-sizing: border-box; } button[type="submit"] {

background-color: #4caf50; color: white;

padding: 10px 20px;

border: none; border-radius: 4px; cursor: pointer; font-size: 16px;}

button[type="submit"]:hover { background-color: #45a049; }

input[type="text"]:focus, input[type="number"]:focus {

border: 2px solid #4caf50; } /\* Optional: Style error message \*/ #error-message {

color: red;

font-size: 14px; margin-top: 10px; }

</style> </head><body>

<h1>Payment Form</h1>

<form id="paymentForm">

<!-- Hidden input field to store order ID -->

<input type="hidden" id="orderId" name="orderId" />

<input type="hidden" id="userId" name="userId" />

<label for="cardholderName">Cardholder Name:</label>

<input type="text" id="cardholderName" name="cardholderName" required

/><br /><br /><label for="cardNumber">Card Number:</label>

<input type="number" id="cardNumber" name="cardNumber" required

/><br /><br />

<label for="expiryDate">Expiry Date:</label>

<input type="date" id="expiryDate"

name="expiryDate" placeholder="MM/YYYY"

required

/><br /><br />

<label for="cvv">CVV:</label>

<input type="text" id="cvv" name="cvv" required /><br /><br />

<button type="submit">Submit Payment</button>

</form> <script>

// Handle form submission document

.getElementById("paymentForm")

.addEventListener("submit", async function (event) { event.preventDefault();

const user = JSON.parse(localStorage.getItem("user"));

const userId = user.id; // Ensure to declare userId with 'const' or 'let' to avoid global scope pollution

const formData = new FormData(this); const paymentData = {

cardholderName: formData.get("cardholderName"), cardNumber: formData.get("cardNumber"), expiryDate: formData.get("expiryDate"),

cvv: formData.get("cvv"), userId: userId, }; try {

const order = localStorage.getItem("orderId"); console.log(order, "order");

const response = await fetch(

`http://localhost:8080/payment/make-payment/${order}`, { method: "POST",

headers: { "Content-Type": "application/json", }, body: JSON.stringify(paymentData), });

if (response.ok) { alert("Payment successful!");

window.location.href = "userOrders.html";

} else {

alert("Payment failed. Please try again.");

} } catch (error) {

console.error("Error submitting payment:", error); alert("Payment failed. Please try again."); } });

</script> </body></html>

# userOrders.html:

<!DOCTYPE html><html lang="en">

<head> <meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<style> body {

font-family: "Poppins", sans-serif;

background-image: url("https://plus.unsplash.com/premium\_photo-1661481417385- de9bdd899a11?q=80&w=1460&auto=format&fit=crop&ixlib=rb- 4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D";

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center;

min-height: 100vh; } .navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; } .navbar h3 { margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/ }

.nav-links {

list-style-type: none; margin: 0;

padding: 0;

display: flex } .nav-links li { margin: 0 10px; }

.nav-links li a {

text-decoration: none; color: #333;

font-size: 16px;

font-weight: bold; } .nav-links li a:hover { color: #007bff; }footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto; }

table {

width: 80%;

border-collapse: collapse; margin-top: 20px; } th, td {

border: 1px solid #ddd; padding: 8px;

text-align: left; } th { background-color: #f2f2f2;

}</style>

</head> <body>

<header>

<nav class="navbar">

<h3>USER</h3>

<ul class="nav-links">

<li><a href="userHome.html">HOME</a></li>

<li><a href="userViewFood.html">VIEW FOOD</a></li>

<li><a href="userViewCart.html">VIEW CART</a></li>

<li><a href="userViewOrders.html">MAKE PAYMENT</a></li>

<li><a href="userOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul>

</nav> </header> <main>

<h1>Orders List</h1>

<table id="ordersList">

<thead> <tr>

<th>User Name</th>

<th>Restaurant Name</th>

<th>Food Name</th>

<th>Quantity</th>

<th>Amount</th>

</tr></thead> <tbody>

<!-- Data will be dynamically added here -->

</tbody> </table>

</main> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer><script>

async function fetchOrders() {try {

const user = JSON.parse(localStorage.getItem("user")); // Corrected line const userId = user.id; // Extract user ID from user object console.log(userId, "user id");

// Fetch orders from the server and display them in the table

// Use the user ID to fetch orders from the server

// Display the orders in the table using the table row and table data i const response = await fetch(

`http://localhost:8080/payment/get/${userId}` ); const data = await response.json(); console.log(data); displayOrders(data.Payments);

} catch (error) {

console.error("Error fetching orders:", error); }} function displayOrders(payments) { const ordersList = document.querySelector("#ordersList tbody"); ordersList.innerHTML = "";

payments.forEach((payment) => { const order = payment.order; console.log(order, "order");

const row = document.createElement("tr"); row.innerHTML = `

<td>${order.user.name}</td>

<td>${order.restaurant.restaurantName}</td>

<td>${order.food.name}</td>

<td>${order.quantity}</td>

<td>${order.amount}</td> `; ordersList.appendChild(row); }); }fetchOrders();

</script> </body></html>

# userRegistration.html:

<!DOCTYPE html><html> <head>

<title>User Registration</title>

<link rel="stylesheet" type="text/css" href="slide navbar style.css" />

</head><style>body{ margin: 0;

padding: 0; display: flex;

justify-content: center; align-items: center; min-height: 100vh;

font-family: 'Jost', sans-serif;

background-image: url("https://img.freepik.com/free-photo/close-up-hands-using- smartphone\_23- 2149250119.jpg?t=st=1709548871~exp=1709552471~hmac=b1870d01150050c696aac3ab d9f6f58cd301ffa39415a4f216aa0b21c3b7d2c1&w=900");

background-repeat: no-repeat; background-size: 100% 100%;}

.main{

width: 350px; height: 600px; background: red; overflow: hidden; border-radius: 10px;

box-shadow: 5px 20px 50px #000;} #chk {

display: none; }

.signup {

position: relative; width: 100%;

height: 100%; } label {

color: #fff;

font-size: 2.3em; justify-content: center; display: flex;

margin: 60px; font-weight: bold; cursor: pointer;

transition: 0.5s ease-in-out; } input {

width: 60%; height: 20px;

background: #e0dede; justify-content: center; display: flex;

margin: 20px auto; padding: 10px; border: none; outline: none;

border-radius: 5px; } button {

width: 60%; height: 40px; margin: 10px auto;

justify-content: center; display: block;

color: #fff; background: #f34609; font-size: 1em;

font-weight: bold; margin-top: 20px; outline: none; border: none; border-radius: 5px;

transition: 0.2s ease-in; cursor: pointer; } button:hover { background: #f34609; }

.login {

height: 600px; background: #eee;

border-radius: 60% / 10%;

transform: translateY(-180px); transition: 0.8s ease-in-out; }

.login label { color: #f34609;

transform: scale(0.6); } #chk:checked ~ .login {

transform: translateY(-600px); } #chk:checked ~ .login label { transform: scale(1); } #chk:checked ~ .signup label { transform: scale(0.8); }

.toast {

visibility: hidden; min-width: 250px; margin-left: -125px;

background-color: #333; color: #fff;

text-align: center; border-radius: 2px; padding: 16px; position: fixed;

z-index: 1;

left: 50%; bottom: 30px;

font-size: 17px; }.toast.show { visibility: visible;

-webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s; animation: fadein 0.5s, fadeout 0.5s 2.5s; }

@-webkit-keyframes fadein { from {

bottom: 0; opacity: 0; } to { bottom: 30px;

opacity: 1; } }@keyframes fadein { from {

bottom: 0; opacity: 0; } to {

bottom: 30px; opacity: 1; } }

@-webkit-keyframes fadeout { from {

bottom: 30px; opacity: 1; } to { bottom: 0;

opacity: 0; } } @keyframes fadeout { from {

bottom: 30px; opacity: 1; } to { bottom: 0;

opacity: 0; } }

footer {

position: absolute; bottom: 0;

color: white; width: 100vw; height: 20px; padding: 20px; text-align: center;}

</style>

<body>

<div class="main">

<input type="checkbox" id="chk" aria-hidden="true" />

<div class="signup">

<form id="signupForm">

<label for="chk" aria-hidden="true">Sign up</label>

<input type="text" name="name" placeholder="User name" required />

<input type="email" name="email" placeholder="Email" required />

<input type="password" name="password" placeholder="Password" required />

<input type="tel" name="mobileNumber" placeholder="Mobile number" required

/>

<input type="text" name="address" placeholder="Address" required />

<button type="submit">Sign up</button>

</form> </div>

<div class="login">

<form id="loginForm">

<label for="chk" aria-hidden="true">Login</label>

<input type="email" name="email" placeholder="Email" required />

<input type="password" name="password" placeholder="Password" required />

<button type="submit">Login</button>

</form> </div> </div>

<div id="toast" class="toast"></div>

<script>

function showToast(message, success = true) { const toast = document.getElementById('toast'); toast.textContent = message;

toast.className = 'toast show'; if (!success) {

toast.style.backgroundColor = 'red'; // Change toast background color for failure } setTimeout(() => {

toast.className = 'toast';

}, 3000); }

document.addEventListener('DOMContentLoaded', () => {

const signupForm = document.getElementById('signupForm'); const loginForm = document.getElementById('loginForm');

signupForm.addEventListener('submit', async (e) => { e.preventDefault();

const formData = new FormData(signupForm);

const userData = Object.fromEntries(formData.entries());

try {

const response = await fetch('http://localhost:8080/user/register', { method: 'POST',

headers: {

'Content-Type': 'application/json' }, body: JSON.stringify(userData) });

if (response.ok) {

console.log('Signup successful'); showToast('Signup successful', true);

} else {

console.error('Signup failed:', response.statusText); showToast('Signup failed', false); }

} catch (error) {

console.error('Error during signup:', error); showToast('Error during signup', false); } });

loginForm.addEventListener('submit', async (e) => { e.preventDefault();

const formData = new FormData(loginForm);

const userData = Object.fromEntries(formData.entries());

try {

const response = await fetch('http://localhost:8080/userLogin', { method: 'POST',

headers: {

'Content-Type': 'application/json' },

body: JSON.stringify(userData) }); if (response.ok) { console.log('Login successful');

const data = await response.json(); localStorage.setItem('user', JSON.stringify(data)); showToast('Login successful', true); window.location.assign("userHome.html");

} else {

console.error('Login failed:', response.statusText); showToast('Login failed', false); }

} catch (error) {

console.error('Error during login:', error); showToast('Error during login', false); } }); });

</script>

</body>

</body>

</html>

# userViewCart.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style> body {

font-family: "Poppins", sans-serif;

background-image: url("https://img.freepik.com/free-photo/girl-cafe-reads- something-tablet\_1304- 5538.jpg?t=st=1709707633~exp=1709711233~hmac=d395970c02b834f1ef28faaec45f2d1c 656a3067c0134304104c8d9a3ad766ee&w=740");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh; }

.navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; }

.navbar h3 {

margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/ }

.nav-links {

list-style-type: none; margin: 0;

padding: 0; display: flex; }

.nav-links li {

margin: 0 10px; }

.nav-links li a {

text-decoration: none; color: #333;

font-size: 16px;

font-weight: bold; } .nav-links li a:hover { color: #007bff; } .cart-container {

width: 80%; }

.table-responsive { overflow-x: auto; }

.cart-table {

width: 100%;

border-collapse: collapse; }

.cart-table th,

.cart-table td { padding: 8px;

border-bottom: 1px solid #ddd; text-align: left; } .cart-table th { background-color: #f2f2f2; }

.cart-table tbody tr:hover { background-color: #f9f9f9; }

.toast {

visibility: hidden; min-width: 250px; margin-left: -125px;

background-color: #333; color: #fff;

text-align: center; border-radius: 2px; padding: 16px; position: fixed;

z-index: 1;

left: 50%; bottom: 30px;

font-size: 17px; }

.toast.show {

visibility: visible;

-webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s; animation: fadein 0.5s, fadeout 0.5s 2.5s; }

@-webkit-keyframes fadein { from {

bottom: 0; opacity: 0; } to { bottom: 30px;

opacity: 1; }} @keyframes fadein {

from {

bottom: 0; opacity: 0; }to { bottom: 30px;

opacity: 1;}}@-webkit-keyframes fadeout { from {

bottom: 30px; opacity: 1;}to { bottom: 0;

opacity: 0}} @keyframes fadeout {

from {

bottom: 30px; opacity: 1; }to { bottom: 0;

opacity: 0;}}

footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto; }

</style>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font- awesome/4.7.0/css/font-awesome.min.css">

</head>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>User Page</title>

</head><body>

<header>

<nav class="navbar">

<h3>USER</h3>

<ul class="nav-links">

<li><a href="userHome.html">HOME</a></li>

<li><a href="userViewFood.html">VIEW FOOD</a></li>

<li><a href="userViewCart.html">VIEW CART</a></li>

<li><a href="userViewOrders.html">MAKE PAYMENT</a></li>

<li><a href="userOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul> </nav> </header>

<div class="cart-container">

<h2>Your Cart</h2>

<div class="table-responsive">

<table class="cart-table">

<thead> <tr>

<th>Food Name</th>

<th>Food Image</th>

<th>Quantity</th>

<th>Price</th>

<th>Action</th>

</tr>

</thead>

<tbody id="cartItems">

<!-- Cart items will be added here dynamically -->

</tbody> </table> </div> </div>

<footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p> </footer> <script> document.addEventListener('DOMContentLoaded', function () {

const user = JSON.parse(localStorage.getItem('user')) || {}; const userId = user.id || 'default';

const apiUrl = `http://localhost:8080/cart/user/${userId}`;

fetch(apiUrl)

.then(response => { if (!response.ok) {

throw new Error('Failed to fetch cart items'); } return response.json();})

.then(data => { displayCartItems(data.carts || []); })

.catch(error => {

console.error('There was an error:', error); }); }); function displayCartItems(cartItems) {

const cartItemsContainer = document.getElementById('cartItems'); cartItemsContainer.innerHTML = '';

if (cartItems.length === 0) {

cartItemsContainer.innerHTML = '<p>Your cart is empty.</p>'; return; } cartItems.forEach(cartItem => {

const cartItemRow = document.createElement('tr'); const foodNameCell = document.createElement('td'); foodNameCell.textContent = cartItem.food.name; cartItemRow.appendChild(foodNameCell);

const foodImageCell = document.createElement('td');

const foodImage = document.createElement('img'); foodImage.src = cartItem.food.image; foodImage.alt = "Food Image"; foodImage.style.maxWidth = "100px"; foodImageCell.appendChild(foodImage); cartItemRow.appendChild(foodImageCell);

const quantityCell = document.createElement('td'); quantityCell.className = 'quantity-control';

const minusButton = document.createElement('i'); minusButton.className = 'fa fa-minus text-danger'; minusButton.addEventListener('click', () => {

if (cartItem.quantity > 1) { cartItem.quantity--;

updateCartItem(cartItem, cartItemRow); } }); quantityCell.appendChild(minusButton);

const quantityDisplay = document.createElement('h5'); quantityDisplay.className = 'text-grey mt-1 mr-1 ml-1'; quantityDisplay.textContent = cartItem.quantity; quantityCell.appendChild(quantityDisplay);

const plusButton = document.createElement('i'); plusButton.className = 'fa fa-plus text-success'; plusButton.addEventListener('click', () => {

cartItem.quantity++; updateCartItem(cartItem, cartItemRow); });

quantityCell.appendChild(plusButton); cartItemRow.appendChild(quantityCell);

const priceCell = document.createElement('td');

priceCell.textContent = `$${(cartItem.food.price \* cartItem.quantity).toFixed(2)}`; cartItemRow.appendChild(priceCell);

const actionCell = document.createElement('td'); cartItemRow.appendChild(actionCell);

const orderButton = document.createElement('button'); orderButton.textContent = "Order"; orderButton.onclick = () => placeOrder(cartItem); actionCell.appendChild(orderButton);

const deleteButton = document.createElement('button'); deleteButton.textContent = "Delete";

deleteButton.onclick = () => deleteCartItem(cartItem.id, cartItemRow); actionCell.appendChild(deleteButton);

cartItemsContainer.appendChild(cartItemRow); });} function placeOrder(cartItem) {

// Assuming orderData is constructed based on cartItem const orderData = {

userId: cartItem.user.id,

restaurantId: cartItem.food.restaurantId, foodId: cartItem.food.id,

quantity: cartItem.quantity };

const apiUrl =

`http://localhost:8080/orders?userId=${cartItem.user.id}&restaurantId=${cartItem.food.rest aurantId}&foodId=${cartItem.food.id}&quantity=${cartItem.quantity}`;

fetch(apiUrl, { method: 'POST', headers: {

'Content-Type': 'application/json' }, body: JSON.stringify(orderData) })

.then(response => { if (!response.ok) {

throw new Error('Failed to place order'); }

return response.json(); })

.then(data => {

console.log('Order placed successfully', data); showToast("Order placed successfully"); window.location.href = "userMakePayment.html"; })

.catch(error => {

console.error('There was an error:', error); }); } function deleteCartItem(cartItemId, cartItemRow) {

const deleteUrl = `http://localhost:8080/cart/${cartItemId}`; fetch(deleteUrl, {

method: 'DELETE' })

.then(response => { if (!response.ok) {

throw new Error('Failed to delete cart item'); } cartItemRow.remove(); // Remove the item row from the table showToast("Item Deleted Successfully"); })

.catch(error => {

console.error('There was an error:', error); });} function updateCartItem(cartItem, cartItemRow) {

// Update quantity display

const quantityDisplay = cartItemRow.querySelector('.text-grey'); quantityDisplay.textContent = cartItem.quantity;

// Update price based on new quantity

const priceCell = cartItemRow.querySelector('td:nth-child(4)'); priceCell.innerHTML = `${(cartItem.food.price \* cartItem.quantity).toFixed(2)}`; }

</script>

</body>

</html>

# userViewFood.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<link href=["ht](https://fonts.googleapis.com/css2?family=Poppins%3Awght%40400%3B600&display=swap)t[ps://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap](https://fonts.googleapis.com/css2?family=Poppins%3Awght%40400%3B600&display=swap)" rel="stylesheet">

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://img.freepik.com/free-photo/side-view-cook-making- delicious-pasta\_23- 2150690631.jpg?t=st=1709706794~exp=1709710394~hmac=27d97ecaf979f0fcb61ab9fb45 e556545695041d94a193ef3d59c3813677fa77&w=740");

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh;

background-attachment: fixed;

/\* Ensure background is fixed during scroll \*/ }

.navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 1px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; position: absolute; top: 0px;

left:0px;

margin-top:2px }

.navbar h3 {

margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/ }

.nav-links {

list-style-type: none;

margin: 0;

padding: 0; display: flex; }

.nav-links li {

margin: 0 10px; }

.nav-links li a {

text-decoration: none; color: #333;

font-size: 16px;

font-weight: bold; }

.nav-links li a:hover { color: #007bff;}

.container {

max-width: 1200px;

/\* Limit container width for larger screens \*/ width: 100%;

display: flex; flex-wrap: wrap;

justify-content: space-around;

background-color: rgba(255, 255, 255, 0.8);

/\* Semi-transparent background for better readability \*/ border-radius: 10px;

padding: 20px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3); margin: 20px; }

.food-card {

background-color: #ffffff; border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); margin: 20px;

overflow: hidden;

width: calc(33.333% - 40px); transition: transform 0.3s ease; }

.food-card:hover {

transform: translateY(-5px); }

.food-image {

width: 100%;

height: 200px; object-fit: cover; }

.food-details { padding: 15px; }

.food-name,

.food-description,

.food-price { margin: 10px 0; }

/\* Responsive adjustments \*/ @media (max-width: 768px) {

.food-card {

width: calc(50% - 40px); } } @media (max-width: 480px) {

.food-card {

width: calc(100% - 40px); } }

footer {

background-color: rgba(255, 255, 255, 0.9);

width: 100%;

padding: 10px 0; font-size: 14px; color: #333; position: fixed;

/\* Fix footer at the bottom \*/ bottom: 0;

left: 0;

z-index: 1000;

/\* Ensure footer is above other content \*/ }

</style>

</head><body>

<header>

<nav class="navbar">

<h3>USER</h3>

<ul class="nav-links">

<li><a href="userHome.html">HOME</a></li>

<li><a href="userViewFood.html">VIEW FOOD</a></li>

<li><a href="userViewCart.html">VIEW CART</a></li>

<li><a href="userViewOrders.html">MAKE PAYMENT</a></li>

<li><a href="userOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul> </nav></header>

<div class="container" id="foodList">

<!-- Food cards will be dynamically added here -->

</div> <footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer> <script> document.addEventListener('DOMContentLoaded', function () {

const storedRestaurant = JSON.parse(localStorage.getItem('restaurant'))|{}; const restaurantId = storedRestaurant.id || 'default';

fetch(`http://localhost:8080/food`)

.then(response => { if (!response.ok) {

throw new Error('Network response was not ok'); } return response.json(); })

.then(data => { displayFoodList(data.Food || []); })

.catch(error => {

console.error('There was a problem fetching the data:', error); }); function createFoodCard(food) {

const card = document.createElement('div'); card.classList.add('food-card');

card.innerHTML = `

<img src="${food.image}" alt="${food.name}" class="food-image">

<div class="food-details">

<h3 class="food-name">${food.name}</h3>

<p class="food-description">${food.foodDescription}</p>

<p class="food-price">PRICE: ${food.price}</p>

<button class="add-to-cart">Add to Cart</button>

</div> `;

// Add event listener for the 'Add to Cart' button

card.querySelector('.add-to-cart').addEventListener('click', function () { addToCart(food); });

return card; }

function addToCart(foodItem) {

// Ensure user data is present

const user = JSON.parse(localStorage.getItem('user')); if (!user || !user.id) {

alert('You must be logged in to add items to the cart.'); return; }

const userId = user.id;

const foodId = foodItem.id;

const apiUrl =

`http://localhost:8080/cart/addToCart?userId=${userId}&foodId=${foodId}`; fetch(apiUrl, {

method: 'POST',

// Assuming the API requires authentication, you might need to include headers

like:

},

headers: {

'Content-Type': 'application/json',

// 'Authorization': `Bearer ${user.token}`, // If using tokens for authentication

// If your API expects a JSON payload in the body: body: JSON.stringify({ userId, foodId }), })

.then(response => { if (!response.ok) {

throw new Error('Network response was not ok'); } return response.json(); })

.then(data => {

console.log(data); // Handle success response

alert(`${foodItem.name} added to cart successfully`); })

.catch(error => {

console.error('There was a problem adding the item to the cart:', error); alert('There was a problem adding the item to the cart.'); });}

function displayFoodList(foods) {

const foodListContainer = document.getElementById('foodList'); foodListContainer.innerHTML = ''; // Clear existing contents

foods.forEach(food => {

const card = createFoodCard(food); foodListContainer.appendChild(card); }); } });

</script>

</body>

</html>

# userViewOrders.html:

<!DOCTYPE html>

<html lang="en"><head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Food Order Web Application</title>

<link rel="stylesheet" href="styles.css" />

<style>

body {

font-family: "Poppins", sans-serif;

background-image: url("https://plus.unsplash.com/premium\_photo-1661481417385- de9bdd899a11?q=80&w=1460&auto=format&fit=crop&ixlib=rb- 4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D";

background-repeat: no-repeat; background-size: cover; display: flex;

flex-direction: column; justify-content: center; align-items: center; min-height: 100vh; }

.navbar {

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); width: 100vw;

background-color: rgba(255, 255, 255, 0.5);

padding: 10px 0; display: flex;

justify-content: space-between; align-items: center;

flex-wrap: wrap; }

.navbar h3 {

margin-left: 20px;

/\* Adjusts the margin between h3 and the edge of the navbar \*/ }

.nav-links {

list-style-type: none; margin: 0;

padding: 0;

display: flex; }

.nav-links li {

margin: 0 10px; }

.nav-links li a {

text-decoration: none; color: #333;

font-size: 16px;

font-weight: bold; }

.nav-links li a:hover {

color: #007bff; } footer {

background-color: rgba(255, 255, 255, 0.7);

width: 100%;

padding: 10px 0; text-align: center; font-size: 14px; color: #333;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1); margin-top: auto; }table {

width: 100%;

border-collapse: collapse; margin-top: 20px; }

th, td {

padding: 10px; text-align: left;

border-bottom: 1px solid #ccc; } th {

background-color: #f2f2f2; color: #333; }

button {

padding: 8px 15px; cursor: pointer; background-color: #333; color: #fff;

border: none; border-radius: 4px;

transition: background-color 0.3s ease; } button:hover {

background-color: #555;}

</style></head>

<body>

<header>

<nav class="navbar">

<h3>USER</h3>

<ul class="nav-links">

<li><a href="userHome.html">HOME</a></li>

<li><a href="userViewFood.html">VIEW FOOD</a></li>

<li><a href="userViewCart.html">VIEW CART</a></li>

<li><a href="userViewOrders.html">MAKE PAYMENT</a></li>

<li><a href="userOrders.html">VIEW ORDERS</a></li>

<li><a href="home.html">LOGOUT</a></li>

</ul> </nav></header>

<main>

<h1>Payments List</h1>

<table id="ordersList">

<thead> <tr>

<th>ID</th>

<th>User Name</th>

<th>Restaurant Name</th>

<th>Food Name</th>

<th>Quantity</th>

<th>Amount</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<!-- Data will be dynamically added here -->

</tbody>

</table>

</main>

<footer>

<p>&copy; 2024 Foodie's Heaven. All rights reserved.</p>

</footer>

<script>

async function fetchOrders() { try {

const user = JSON.parse(localStorage.getItem('user')); const userId = user.id;

const response = await fetch(`http://localhost:8080/orders/user/${userId}`); const data = await response.json();

localStorage.setItem('order', JSON.stringify(data.Orders)); console.log(data,"data");

displayOrders(data.Orders);

} catch (error) {

console.error('Error fetching orders:', error); } } function displayOrders(orders) {

const ordersList = document.querySelector('#ordersList tbody'); ordersList.innerHTML = '';

orders.forEach(order => {

const row = document.createElement('tr'); row.innerHTML = `

<td>${order.id}</td>

<td>${order.user.name}</td>

<td>${order.restaurant.restaurantName}</td>

<td>${order.food.name}</td>

<td>${order.quantity}</td>

<td>${order.amount}</td>

<td><button onclick="makePayment(${order.id})">Make Payment</button>

<button onclick="deleteOrder(${order.id})">Delete</button></td> `; ordersList.appendChild(row);}); }

async function makePayment(orderId) { try {

localStorage.setItem("orderId", orderId)

// Redirect to the payment page with order ID as a query parameter window.location.href = `userMakePayment.html?orderId=${orderId}`;

} catch (error) {

console.error('Error making payment:', error); } } async function deleteOrder(orderId) {

try {

const response = await fetch(`http://localhost:8080/orders/${orderId}`, { method: 'DELETE' });

if (response.ok) {

alert('Order deleted successfully!'); fetchOrders();

} else {

alert('Failed to delete order. Please try again.'); }

} catch (error) {

console.error('Error deleting order:', error); alert('Failed to delete order. Please try again.'); } }

window.onload = fetchOrders;

</script>

</body>

</html>

# Backend:

**foodorderapplication:**

package com.project.foodOrder;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication

public class FoodOrderApplication {

public static void main(String[] args) { SpringApplication.run(FoodOrderApplication.class, args);}}

# authcontroller.java:

package com.project.foodOrder.controller; import com.project.foodOrder.model.Restaurant; import com.project.foodOrder.model.User;

import com.project.foodOrder.service.Impl.RestaurantService; import com.project.foodOrder.service.Impl.UserService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.HashMap;

@RestController

@RequestMapping @CrossOrigin("\*")

public class AuthController{ @Autowired

private UserService userService; @Autowired

private RestaurantService restaurantService; @PostMapping("/adminLogin")

public ResponseEntity<?> adminLogin(@RequestBody HashMap<String, String> login)

{

try {

String email = login.get("email");

String password = login.get("password");

if (email == null || password == null) {

return new ResponseEntity<>("Email and password fields are required", HttpStatus.BAD\_REQUEST);}

if (email.equals(["a](mailto:admin@gmail.com)d[min@gmail.com"](mailto:admin@gmail.com)) && password.equals("admin")) { return new ResponseEntity<>(login, HttpStatus.OK); }

return new ResponseEntity<>("Invalid credentials", HttpStatus.BAD\_REQUEST);

} catch (Exception e) {

return new ResponseEntity<>("An error occurred during admin login: " + e.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);} } @PostMapping("/userLogin")

public ResponseEntity<?> userLogin(@RequestBody HashMap<String, String> login) { try {

String email = login.get("email");

String password = login.get("password"); if (email == null || password == null) {

return new ResponseEntity<>("Email and password fields are required", HttpStatus.BAD\_REQUEST);}

User user = userService.getUserByEmailAndPassword(email,password); if (user != null) {

return new ResponseEntity<>(user, HttpStatus.OK); }

return new ResponseEntity<>("Invalid credentials", HttpStatus.BAD\_REQUEST);

} catch (Exception e) {

return new ResponseEntity<>("An error occurred during user login: " + e.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);}} @PostMapping("/restaurantLogin")

public ResponseEntity<?> restaurantLogin(@RequestBody HashMap<String, String> login) {

try {

String email = login.get("email");

String password = login.get("password"); if (email == null || password == null) {

return new ResponseEntity<>("Email and password fields are required", HttpStatus.BAD\_REQUEST);}

Restaurant restaurant = restaurantService.getRestaurantByEmailAndPassword(email,password);

if (restaurant != null) {

return new ResponseEntity<>(restaurant, HttpStatus.OK);}

return new ResponseEntity<>("Invalid credentials", HttpStatus.BAD\_REQUEST);

} catch (Exception e) {

return new ResponseEntity<>("An error occurred during restaurant login: " + e.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);} }}

# cartcontroller.java:

package com.project.foodOrder.controller; import com.project.foodOrder.model.Cart;

import com.project.foodOrder.service.ICartService; import com.project.foodOrder.service.Impl.CartService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.HashMap;

import java.util.List; @RestController @RequestMapping("/cart") @CrossOrigin("\*")

public class CartController

{

@Autowired

private CartService service;

@PostMapping("/addToCart")

public ResponseEntity<?> addToCart(@RequestParam Long userId,

@RequestParam Long foodId)

{

HashMap<String,Object> res = new HashMap<>(); try {

service.addToCart(userId, foodId); res.put("success",true); res.put("msg","cart added successfully");

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch (Exception e)

{

res.put("success",true); res.put("msg","Failed to add the cart");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping("/{id}")

private ResponseEntity<?> getCartById(@PathVariable Long id){ HashMap<String,Object> res = new HashMap<>();

try{

Cart cart = service.getCartById(id); res.put("success",true); res.put("cart",cart);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch (Exception e)

{

res.put("success",false);

res.put("cart","Failed tp fetch the cart by provided id is"+id); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

// @PostMapping("/cart/{userId}/{foodId}/add")

// public Cart addCartToQuantity(@PathVariable Long userId, @PathVariable Long foodId)

{

// return service.addCartToQuantity(userId, foodId);

// } @GetMapping

private ResponseEntity<?> getAllCarts()

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Cart> carts = service.getAllCarts(); res.put("success",true); res.put("Carts",carts);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the available carts");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping("/user/{userId}")

private ResponseEntity<?> getCartsByUserId(@PathVariable Long userId)

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Cart> carts = service.getCartByUserId(userId);

res.put("success",true); res.put("carts",carts);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the available carts for provided user id is"+userId); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

@DeleteMapping("/{id}")

private ResponseEntity<?> deleteCartById(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try

{

service.deleteCartById(id); res.put("success",true);

res.put("msg","Cart Deleted Successfully");

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch (Exception e)

{

res.put("success",false);

res.put("msg","Failed to Delete the cart by provided id is"+id); return ResponseEntity.status(HttpStatus.OK).body(res);

}

}

}

# foodcontroller.java:

package com.project.foodOrder.controller; import com.project.foodOrder.model.Food;

import com.project.foodOrder.model.Restaurant;

import com.project.foodOrder.service.Impl.FoodService; import com.project.foodOrder.service.Impl.RestaurantService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity; import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*; import org.springframework.web.multipart.MultipartFile; import java.io.IOException;

import java.nio.file.Path; import java.nio.file.Paths; import java.util.HashMap; import java.util.List; @RestController @RequestMapping("/food") @CrossOrigin("\*")

public class FoodController

{

@Autowired

private FoodService service; @Autowired

private RestaurantService restaurantService; @PostMapping("/addFood")

public ResponseEntity<?> addFood(Long restaurantId, String name,

MultipartFile image, String foodDescription, Double price) {

HashMap<String, Object> res = new HashMap<>(); try {

String directoryPath = Paths.get("src", "main", "resources", "static", "Images").toAbsolutePath().toString();

Path originalFilePath = Paths.get(directoryPath, image.getOriginalFilename()); String host = "http://localhost:8080/Images/" + image.getOriginalFilename(); image.transferTo(originalFilePath);

Restaurant restaurant = restaurantService.getRestaurantById(restaurantId); if (restaurant == null) {

res.put("message", "Restaurant not found"); return ResponseEntity.badRequest().body(res);

}

Food food = Food.builder()

.name(name)

.image(host)

.restaurantId(restaurantId)

.foodDescription(foodDescription)

.price(price)

.build(); service.addFood(food);

restaurant.getFood().add(food); restaurantService.updateRestaurant(restaurant);

res.put("success", true);

res.put("message", "Food added successfully"); return ResponseEntity.ok(res);

} catch (IOException e) {

res.put("message", "Error saving image");

return ResponseEntity.internalServerError().body(res);

} catch (Exception e) {

res.put("message", "Error adding food");

return ResponseEntity.internalServerError().body(res);

}

}

@GetMapping

private ResponseEntity<?> getAllFoods()

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Food> foods = service.getAllFoods(); res.put("success",true); res.put("Food",foods);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to get the available foods");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping("/{id}")

private ResponseEntity<?> getFoodById(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try{

Food food = service.getFoodById(id); res.put("success",true); res.put("Food",food);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the food details by provided id is"+ id); return ResponseEntity.status(HttpStatus.OK).body(res);

}

}

@GetMapping("/restaurant/{id}")

private ResponseEntity<?> getFoodByRestaurantId(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Food> foods = service.getFoodByRestaurantId(id); res.put("success",true);

res.put("Food",foods);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the food details by provided restaurant id is"+ id); return ResponseEntity.status(HttpStatus.OK).body(res);

}

}

@DeleteMapping("/{id}")

private ResponseEntity<?> deleteFoodById(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try{

service.deleteFoodById(id); res.put("success",true);

res.put("msg","Food Deleted Successfully");

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to delete the food by provided id is"+ id); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

}

# ordercontroller.java:

package com.project.foodOrder.controller; import com.project.foodOrder.model.Orders;

import com.project.foodOrder.service.Impl.OrderService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.HashMap;

import java.util.List; @RestController @RequestMapping("/orders") @CrossOrigin("\*")

public class OrderController

{

@Autowired

private OrderService service; @PostMapping

private ResponseEntity<?> makeOrder(@RequestParam Long userId,

@RequestParam Long restaurantId, @RequestParam Long foodId, @RequestParam Integer quantity)

{

HashMap<String,Object> res = new HashMap<>(); try{

service.placeOrder(userId,restaurantId,foodId,quantity); res.put("success",true);

res.put("msg","Ordered Successfully");

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false); res.put("msg","Ordered Failed");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping

private ResponseEntity<?> getAllOrders()

{

HashMap<String,Object> res = new HashMap<>(); try

{

List<Orders> orders = service.getAllOrders(); res.put("success",true); res.put("Orders",orders);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the available orders");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping("/user/{userId}")

private ResponseEntity<?> getOrdersByUserId(@PathVariable Long userId)

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Orders> orders = service.getOrderByUserId(userId); res.put("success",true);

res.put("Orders",orders);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the orders by provided user id is"+userId); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

@GetMapping("/restaurant/{restaurantId}")

private ResponseEntity<?> getOrdersByRestaurantId(@PathVariable Long restaurantId)

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Orders> orders = service.getOrderByUserId(restaurantId); res.put("success",true);

res.put("Orders",orders);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the orders by provided restaurant id is"+restaurantId);

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

@DeleteMapping("/{orderId}")

private ResponseEntity<?> deleteOrderById(@PathVariable Long orderId)

{

HashMap<String,Object> res = new HashMap<>(); try

{

service.deleteOrderById(orderId); res.put("success",true); res.put("msg","Order deleted Successfully");

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch (Exception e)

{

res.put("success",false);

res.put("msg","Failed to delete the order by provided id is"+orderId); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

}

# paymentcontroller.java:

package com.project.foodOrder.controller; import com.project.foodOrder.model.Orders; import com.project.foodOrder.model.Payment; import com.project.foodOrder.model.User;

import com.project.foodOrder.repository.IOrderRepository; import com.project.foodOrder.service.IOrderService; import com.project.foodOrder.service.IPaymentService; import com.project.foodOrder.service.IRestaurantService; import com.project.foodOrder.service.IUserService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;

import java.util.HashMap; import java.util.List;

import java.util.NoSuchElementException; @RestController @RequestMapping("/payment") @CrossOrigin("\*")

public class PaymentController

{

@Autowired

private IPaymentService service; @Autowired

private IOrderRepository repository; @PostMapping("/make-payment/{orderId}")

public ResponseEntity<?> makePayment(@PathVariable Long orderId,@RequestBody Payment payment)

{

HashMap<String,Object> res = new HashMap<>(); try {

Orders order = repository.findById(orderId).get(); payment.setOrder(order); service.makePayment(payment); res.put("success",true);

res.put("msg","Payment Successfully Completed"); return ResponseEntity.status(HttpStatus.OK).body(res);

} catch (IllegalArgumentException e) { res.put("success",false); res.put("msg","Payment Failed");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping

private ResponseEntity<?> getAllPayments()

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Payment> payments = service.getAllPayments(); res.put("success",true);

res.put("Payments",payments);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the available Payments");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping("/{id}")

private ResponseEntity<?> getPaymentsById(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try{

Payment payment = service.getPaymentById(id); res.put("success",true); res.put("Payment",payment);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the payments for the provided id is"+id); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

@GetMapping("/get/{userId}")

private ResponseEntity<?> getPaymentDetailsOfUsers(@PathVariable Long userId)

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Payment> payments = service.getPaymentsByUserId(userId); res.put("success",true);

res.put("Payments",payments);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the payment details by provided user id is"+userId); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

@DeleteMapping("/delete/{id}")

private ResponseEntity<?> deletePaymentsById(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try{

service.deletePaymentById(id); res.put("success",true); res.put("msg","Payment successfully deleted");

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to delete the payment id is"+id);

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

}

# restaurantcontroller.java:

package com.project.foodOrder.controller; import com.project.foodOrder.model.Orders; import com.project.foodOrder.model.Payment; import com.project.foodOrder.model.User;

import com.project.foodOrder.repository.IOrderRepository; import com.project.foodOrder.service.IOrderService; import com.project.foodOrder.service.IPaymentService; import com.project.foodOrder.service.IRestaurantService; import com.project.foodOrder.service.IUserService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import java.util.HashMap;

import java.util.List;

import java.util.NoSuchElementException; @RestController @RequestMapping("/payment") @CrossOrigin("\*")

public class PaymentController

{

@Autowired

private IPaymentService service; @Autowired

private IOrderRepository repository; @PostMapping("/make-payment/{orderId}")

public ResponseEntity<?> makePayment(@PathVariable Long orderId,@RequestBody Payment payment)

{

HashMap<String,Object> res = new HashMap<>(); try {

Orders order = repository.findById(orderId).get(); payment.setOrder(order); service.makePayment(payment); res.put("success",true);

res.put("msg","Payment Successfully Completed"); return ResponseEntity.status(HttpStatus.OK).body(res);

} catch (IllegalArgumentException e) { res.put("success",false); res.put("msg","Payment Failed");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping

private ResponseEntity<?> getAllPayments()

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Payment> payments = service.getAllPayments(); res.put("success",true); res.put("Payments",payments);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the available Payments");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(res);

}

}

@GetMapping("/{id}")

private ResponseEntity<?> getPaymentsById(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try{

Payment payment = service.getPaymentById(id); res.put("success",true); res.put("Payment",payment);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the payments for the provided id is"+id); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

@GetMapping("/get/{userId}")

private ResponseEntity<?> getPaymentDetailsOfUsers(@PathVariable Long userId)

{

HashMap<String,Object> res = new HashMap<>(); try{

List<Payment> payments = service.getPaymentsByUserId(userId); res.put("success",true);

res.put("Payments",payments);

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to fetch the payment details by provided user id is"+userId); return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);

}

}

@DeleteMapping("/delete/{id}")

private ResponseEntity<?> deletePaymentsById(@PathVariable Long id)

{

HashMap<String,Object> res = new HashMap<>(); try{

service.deletePaymentById(id); res.put("success",true); res.put("msg","Payment successfully deleted");

return ResponseEntity.status(HttpStatus.OK).body(res);

}

catch(Exception e)

{

res.put("success",false);

res.put("msg","Failed to delete the payment id is"+id);

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(res);}}}

# Applicationproperties:

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.datasource.url=jdbc:mysql://localhost:3306/foodOrder spring.datasource.username=root spring.datasource.password=Anu@6305 spring.jpa.hibernate.ddl-auto= update

# icartrepository:

package com.project.foodOrder.repository; import com.project.foodOrder.model.Cart; import com.project.foodOrder.model.Food; import com.project.foodOrder.model.User;

import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;

import java.util.List; @Repository

public interface ICartRepository extends JpaRepository<Cart,Long> { List<Cart> findByUserId(Long userId);

Cart findByUserAndFood(User user, Food food);

}

# Ifoodrepository:

package com.project.foodOrder.repository; import com.project.foodOrder.model.Food;

import org.hibernate.type.descriptor.converter.spi.JpaAttributeConvert er; import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository; import java.util.List;

@Repository

public interface IFoodRepository extends JpaRepository<Food,Long>

{

List<Food> findFoodByRestaurantId(Long restaurantId);

}

**Iorderrepository:**

package com.project.foodOrder.repository; import com.project.foodOrder.model.Orders;

import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;

import java.util.List; @Repository

public interface IOrderRepository extends JpaRepository<Orders,Long>

{

List<Orders> getOrderByUserId(Long userId); List<Orders> getOrderByRestaurantId(Long restaurantId);

}

# Ipaymentrepository:

package com.project.foodOrder.repository; import com.project.foodOrder.model.Orders; import com.project.foodOrder.model.Payment;

import org.springframework.data.jpa.repository.JpaRepository; import java.util.List;

public interface IPaymentRepository extends JpaRepository<Payment,Long>

{

List<Payment> getPaymentsByUserId(Long userId);

}

# Irestuarantrepository:

package com.project.foodOrder.repository; import com.project.foodOrder.model.Restaurant;

import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository; @Repository

public interface IRestaurantRepository extends JpaRepository<Restaurant,Long>

{

Restaurant findByEmailAndPassword(String email,String password);

}

# Iuserrepository:

package com.project.foodOrder.repository; import com.project.foodOrder.model.User;

import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository; @Repository

public interface IUserRepository extends JpaRepository<User,Long>

{

User findByEmailAndPassword(String email,String password);}

1. **IMPLEMENTATION AND RESULTS**

**MODULES**

**1.Admin:**

Admin login with their credentials. Admin can add the details of tourist place that can view the customers.

**View Tours:** Admin can view the tours added by him

**View Customers:** Admin can view the newly added customers.

**View Bookings:** Admin can view all the bookings that are done by the customer.

**1.User:**

User can register and login with the correct credentials, user can view the places, that are added by the admin

**Book Tour:** User can book the tour that are added by the admin. After book the tour user need to pay amount for the booking. User able to give the review for the particular tourist place.

**Explore Places**: User can search particular place, news, weather, hotels, restaurants and tourism near by searched place

**My bookings:** After booking completed user can view the booking that are done by him/her.

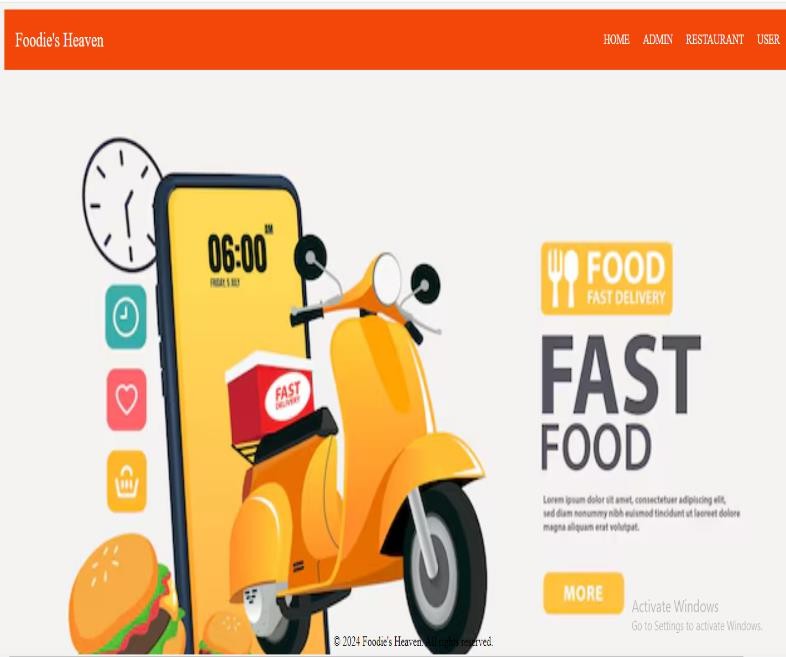
**Settings:** User can change the profile by changing their details and by adding photo.

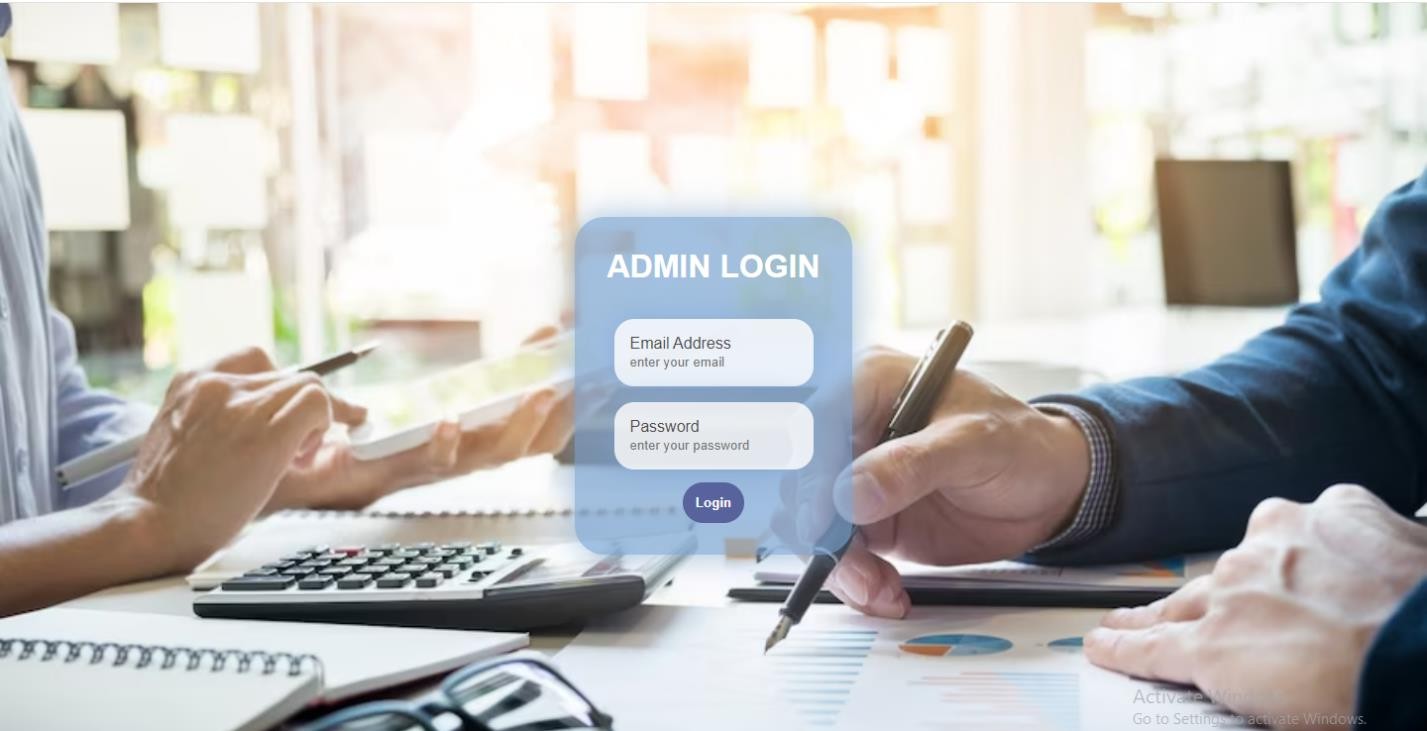
**My Reviews:** User can view there views that are given by him/her.

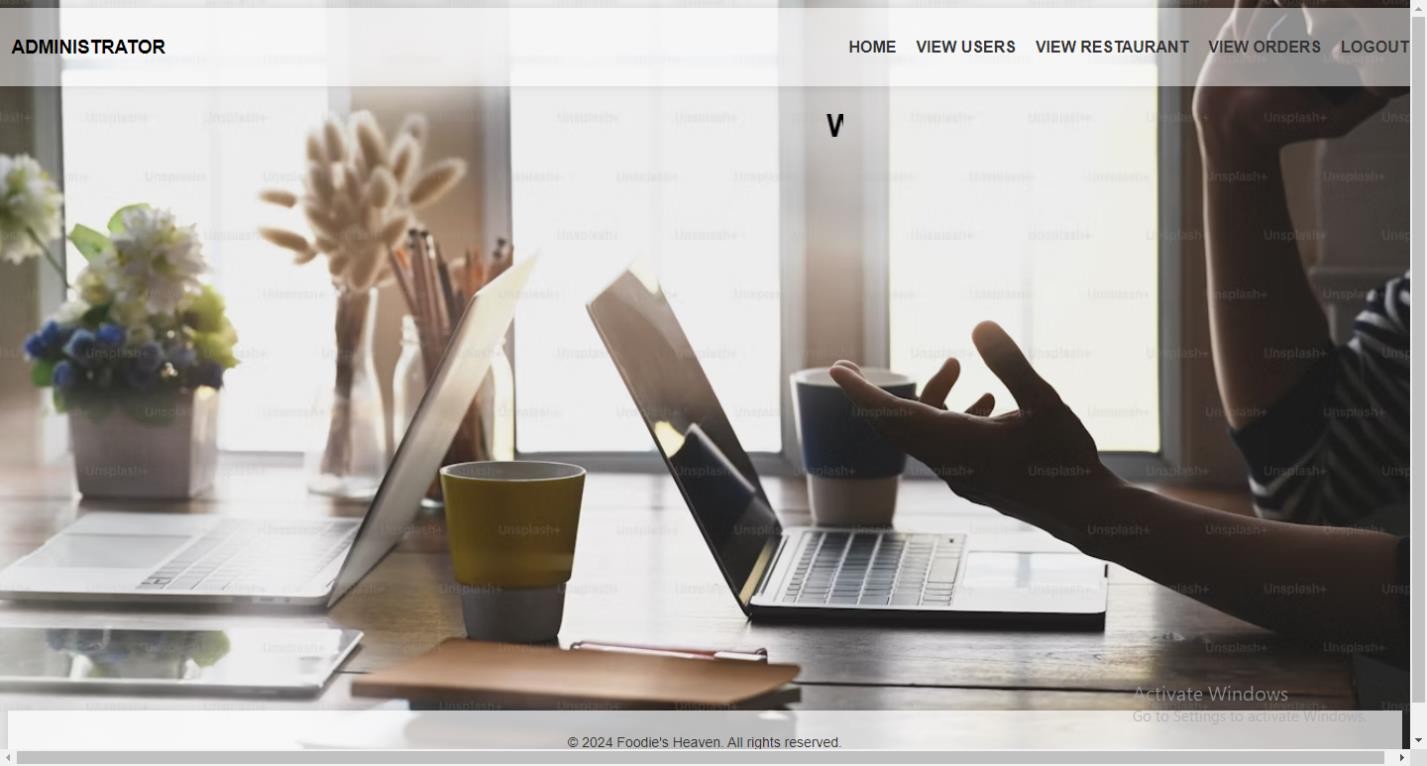
**Billing:** User can able to view the billing details.

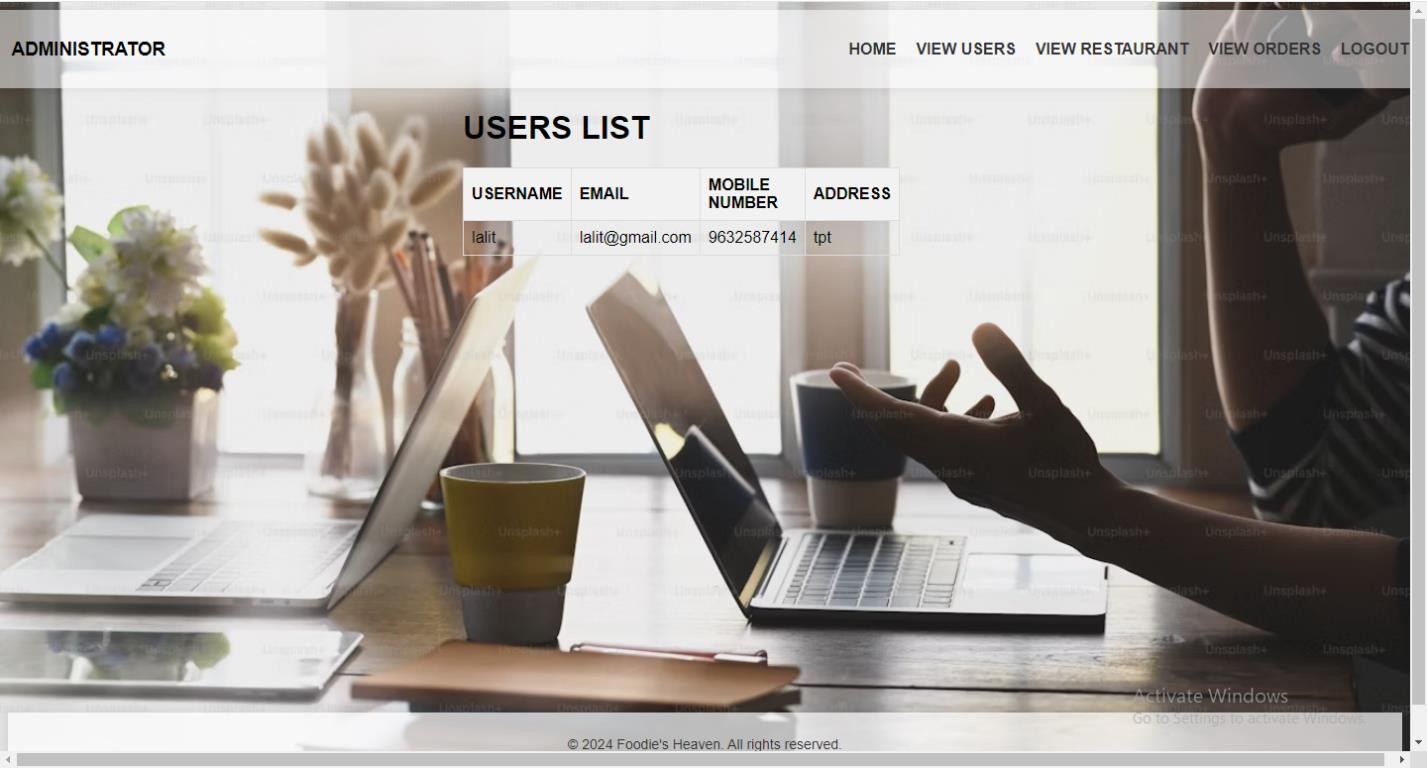
### Results:

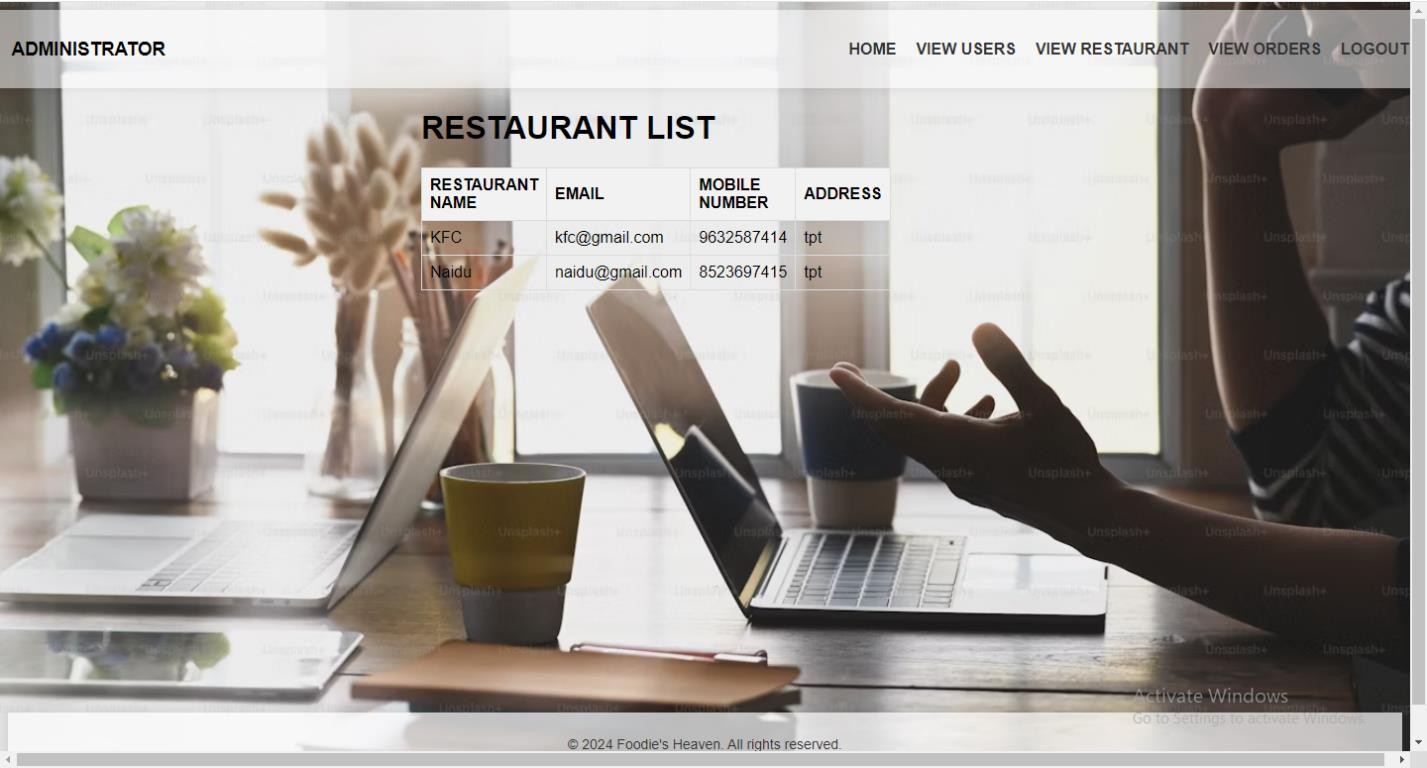
**7.1 Home Page:**

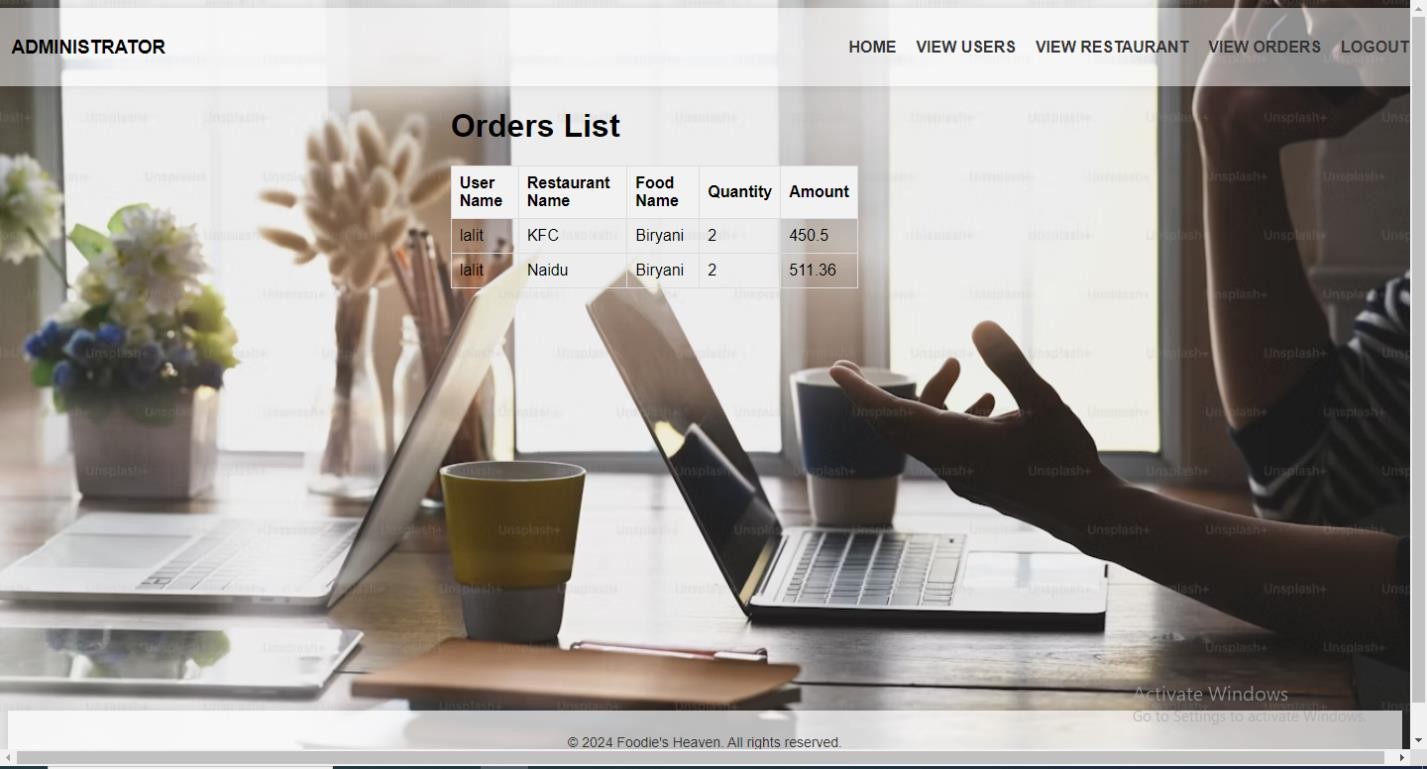


**7.2 Admin login:**

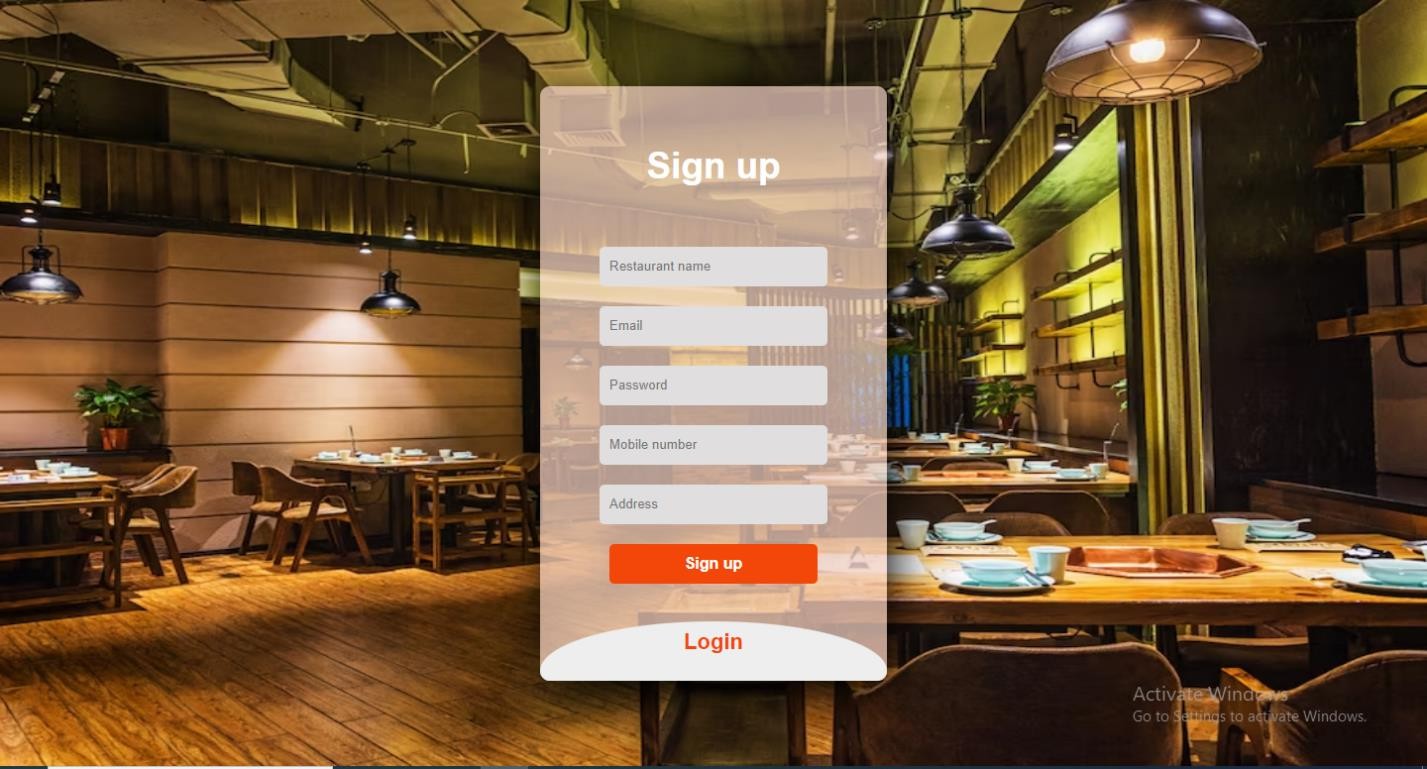
7.3 Admin Home:

**7.4 Admin View Users:**

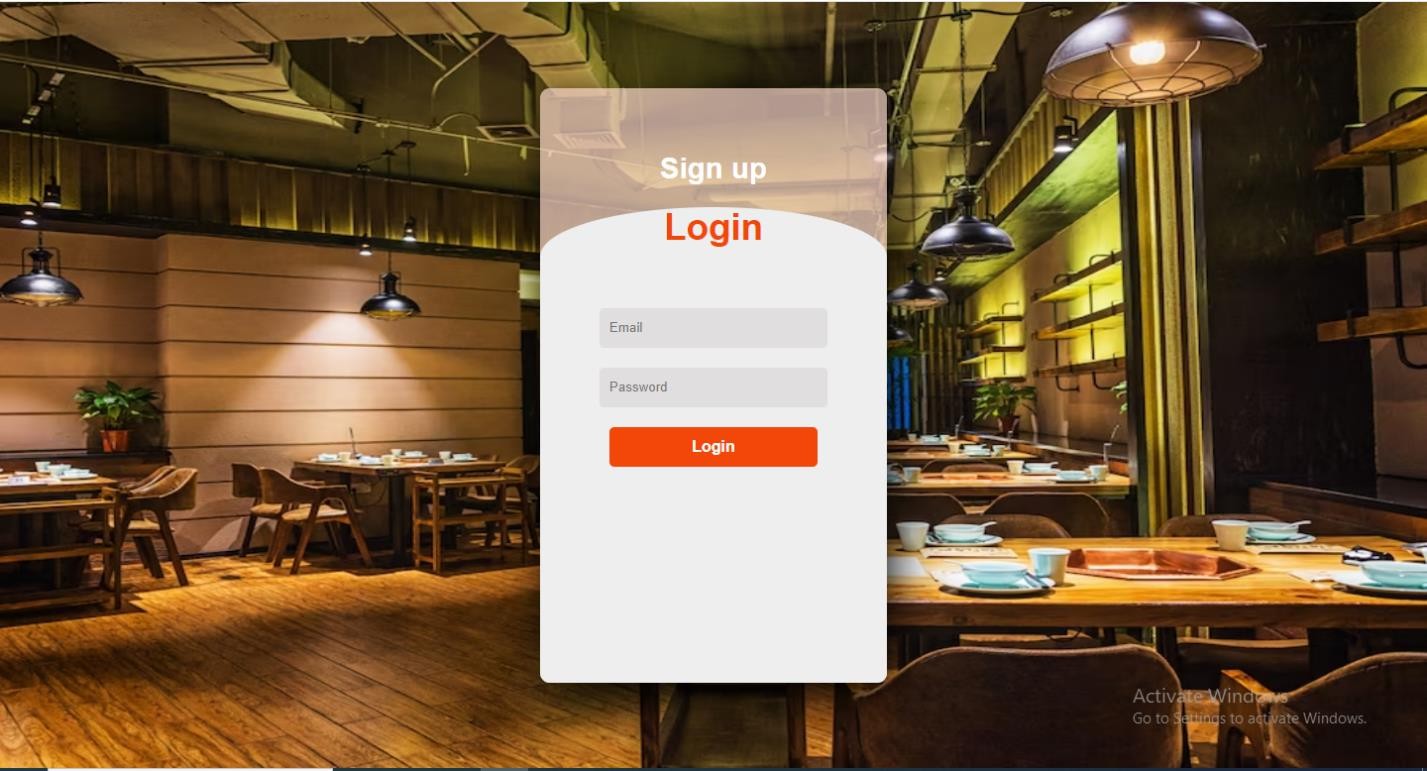
**7.5 Admin view restuarants:**

**7.6 Admin view users:**

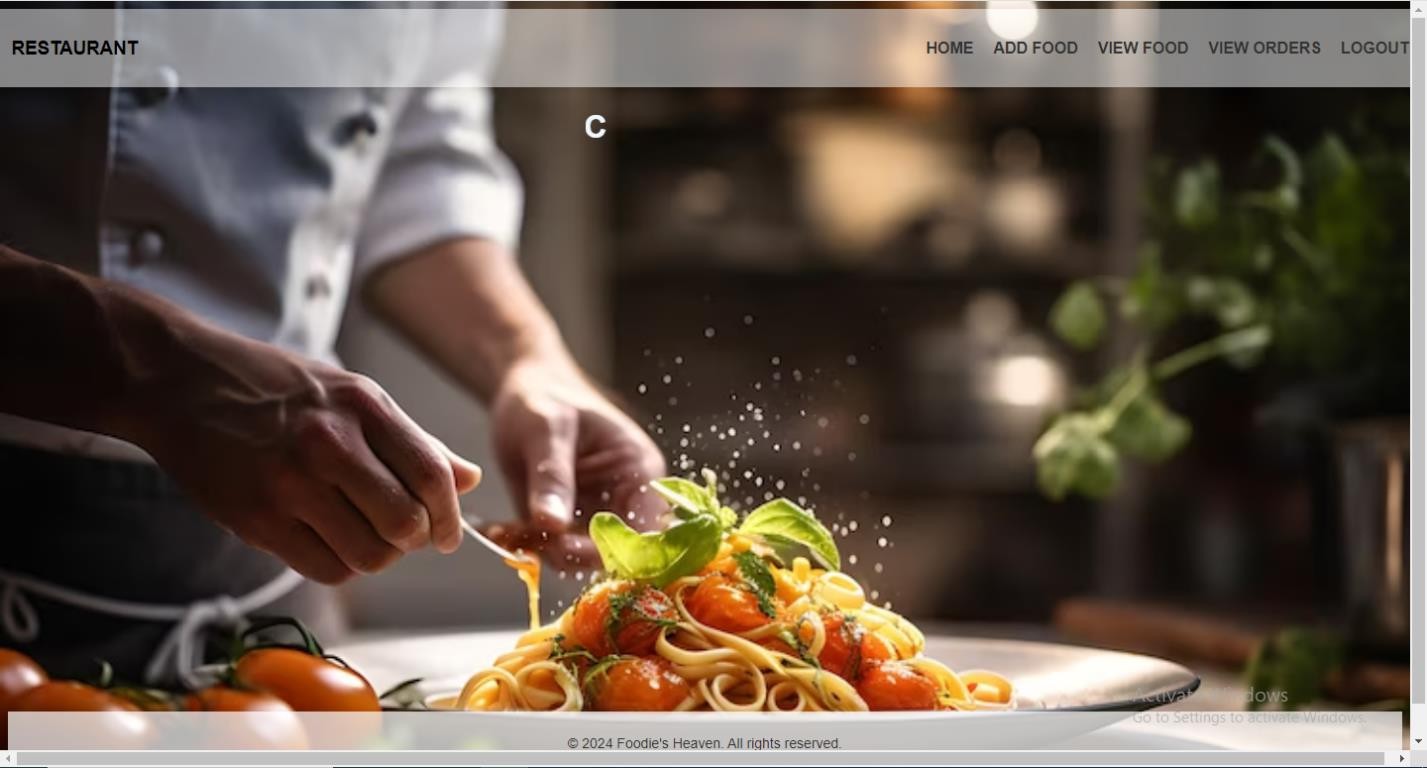
**7.7 Restaurant register:**

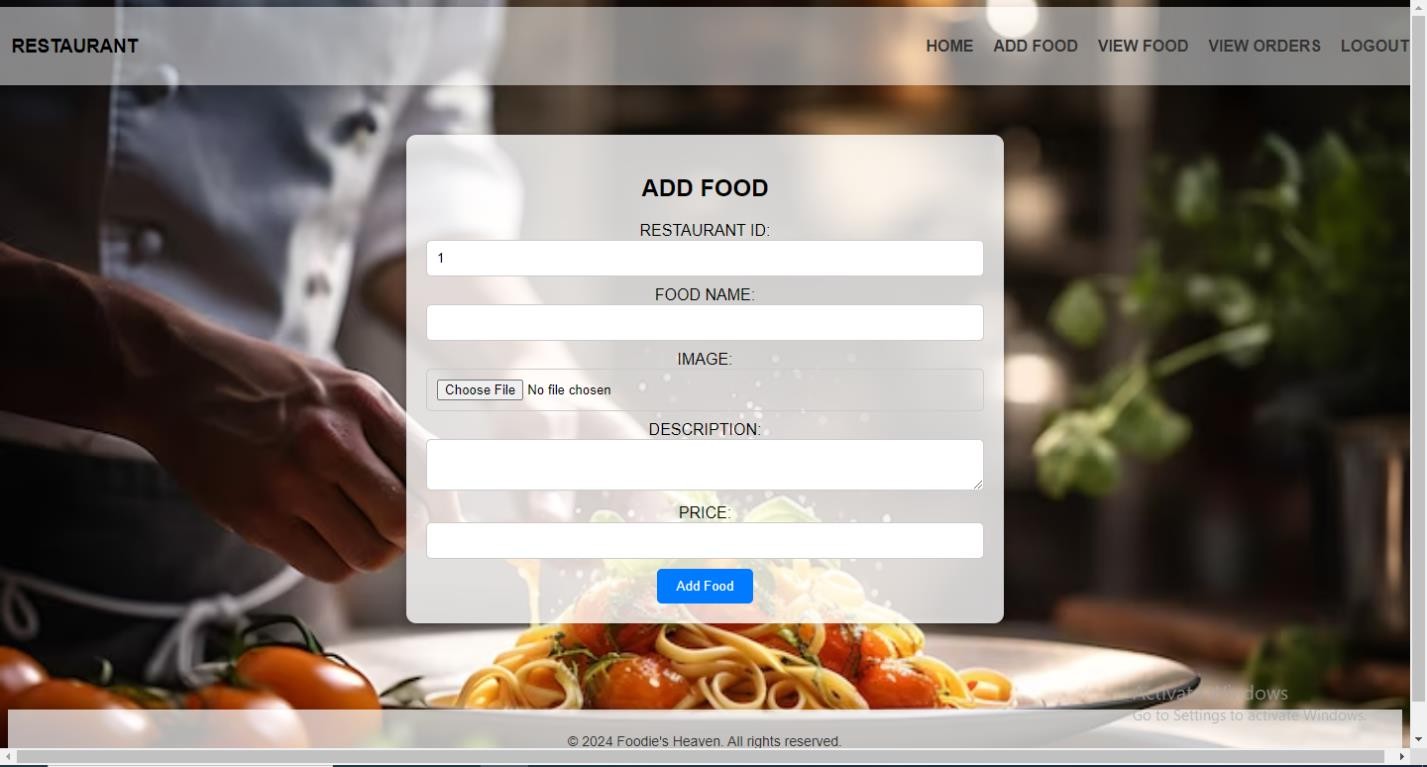


**7.8 Restaurant Login:**

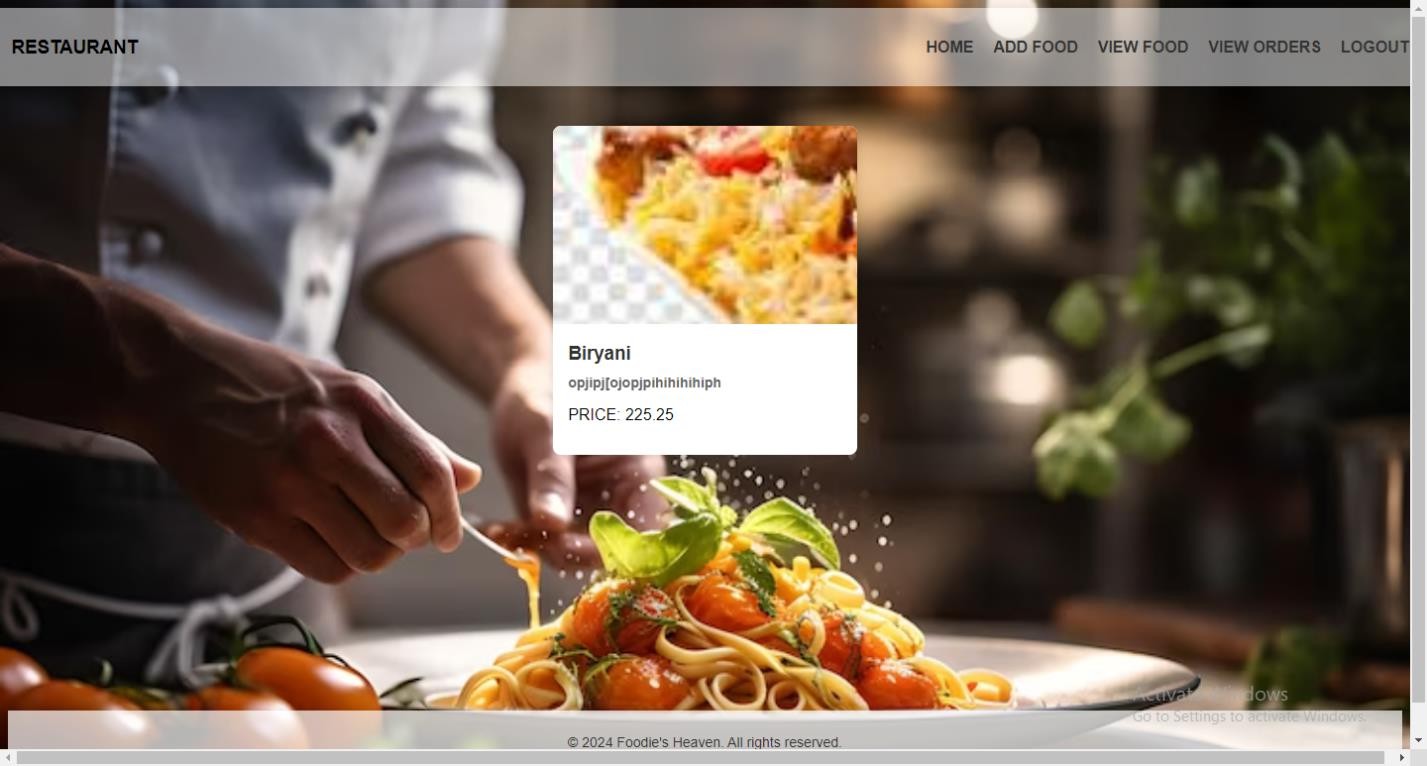


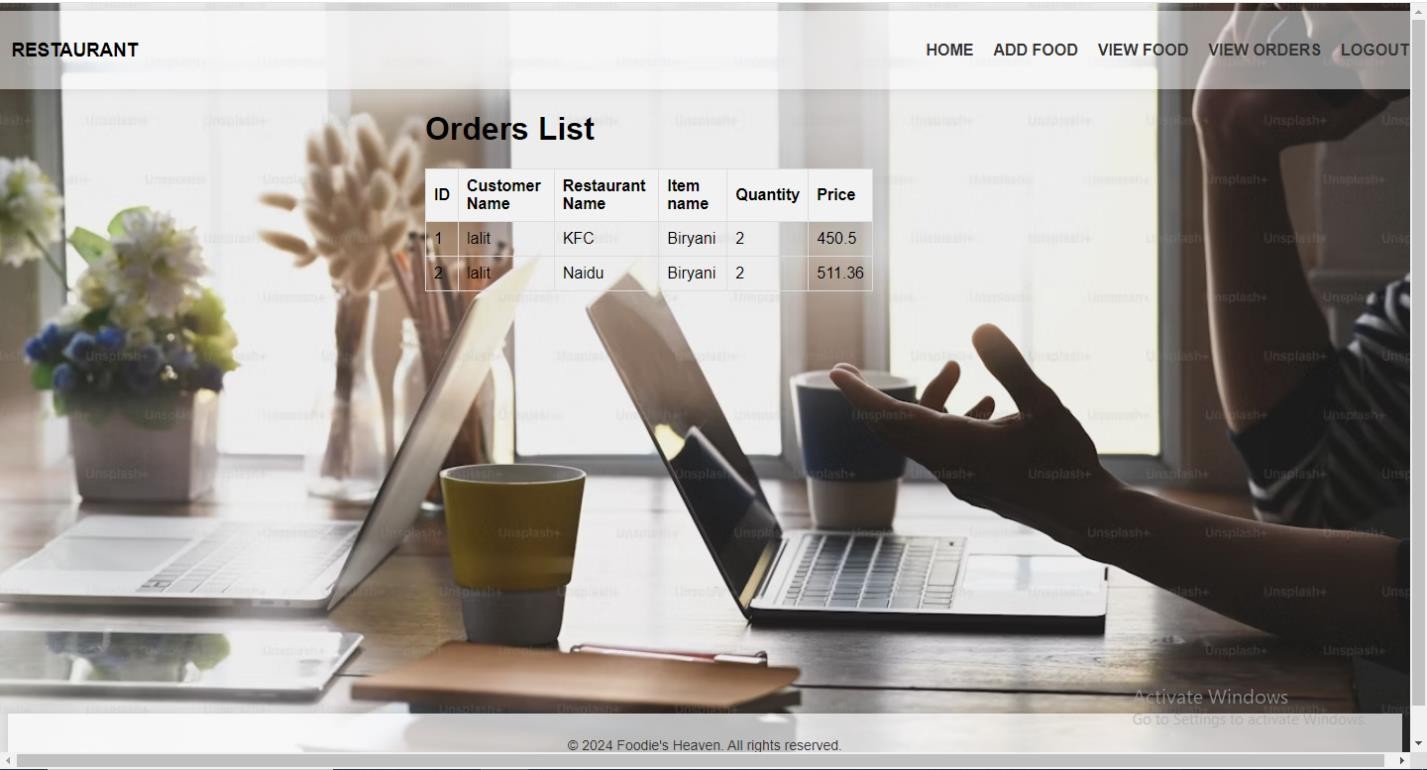
**7.9 Restaurant Home:**



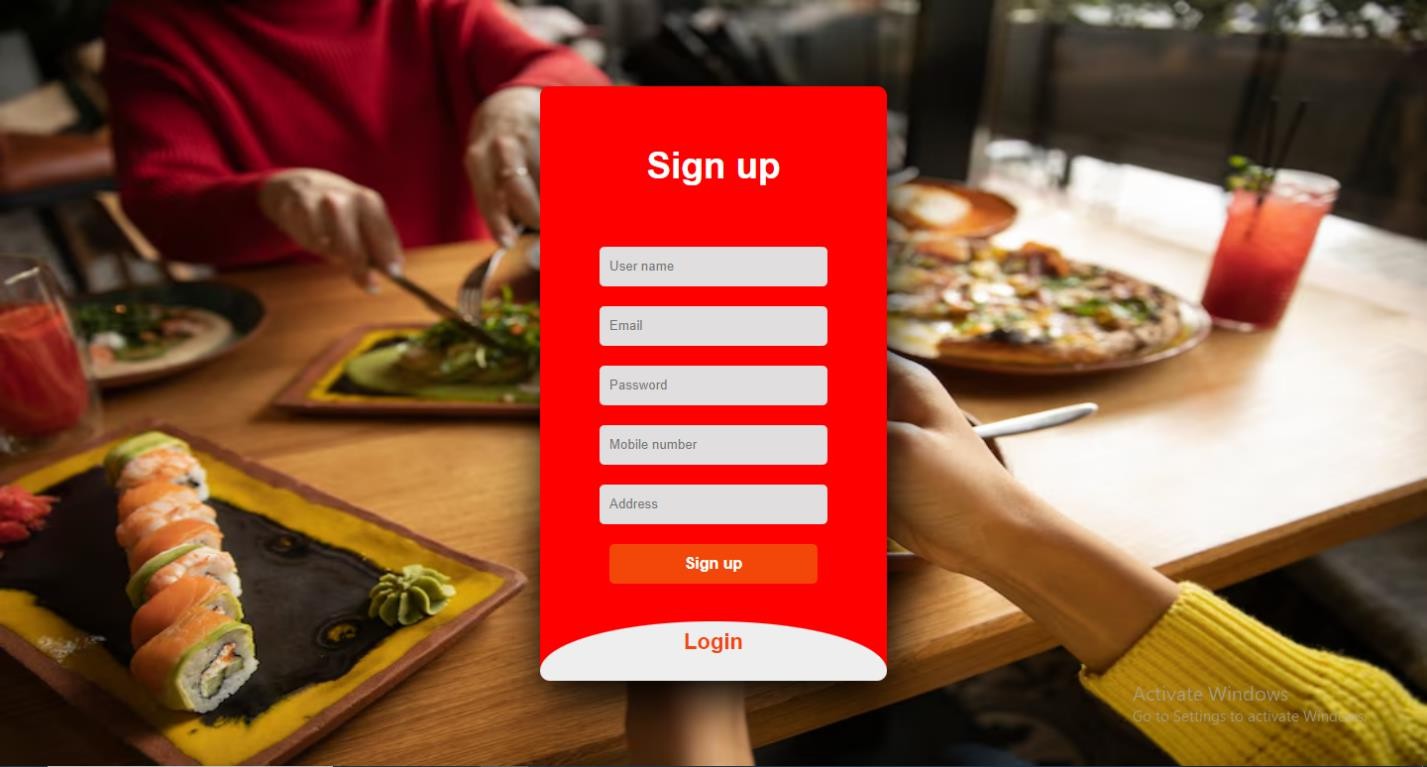
**7.10 Restaurant add food:**

**7.11 Restaurant view food:**

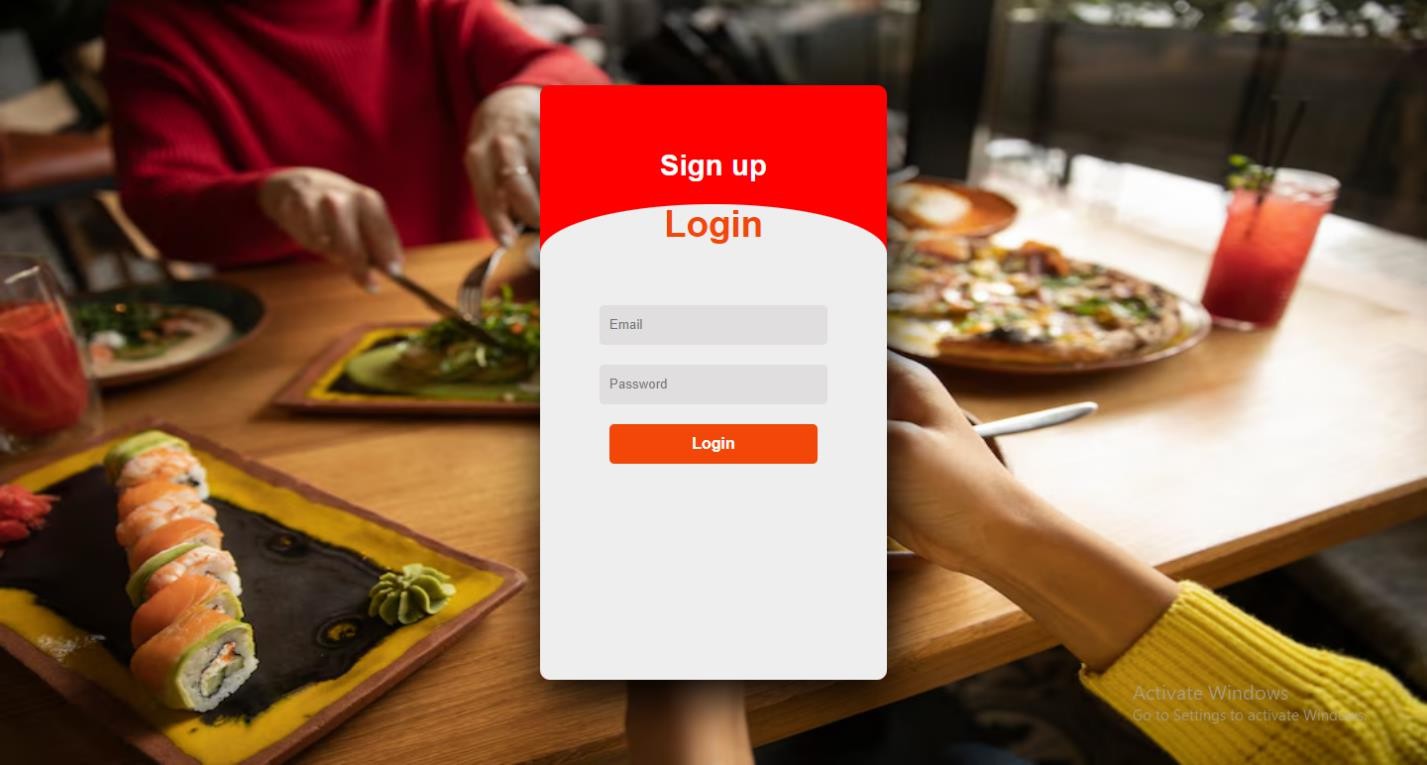


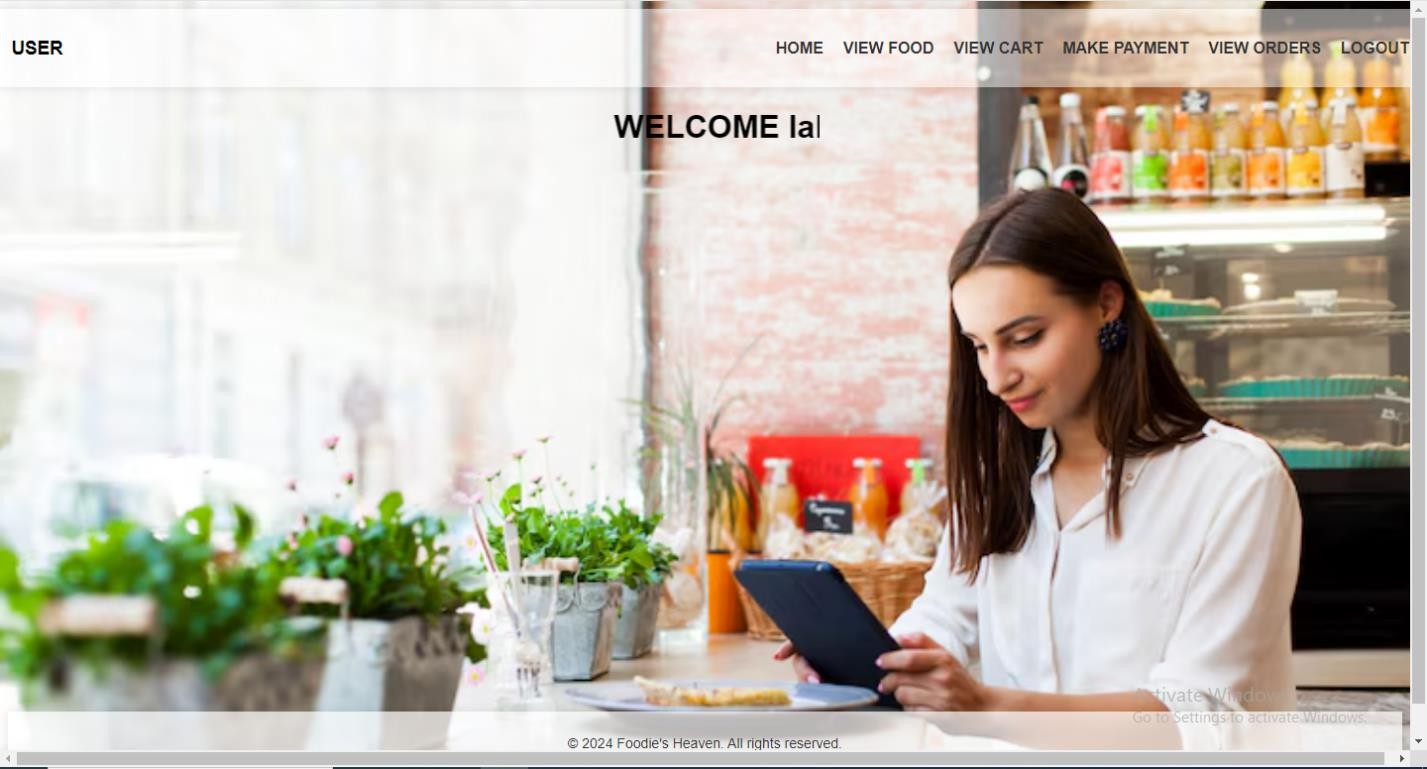
**7.12 Restaurant view orders:**

**7.13 User Register:**

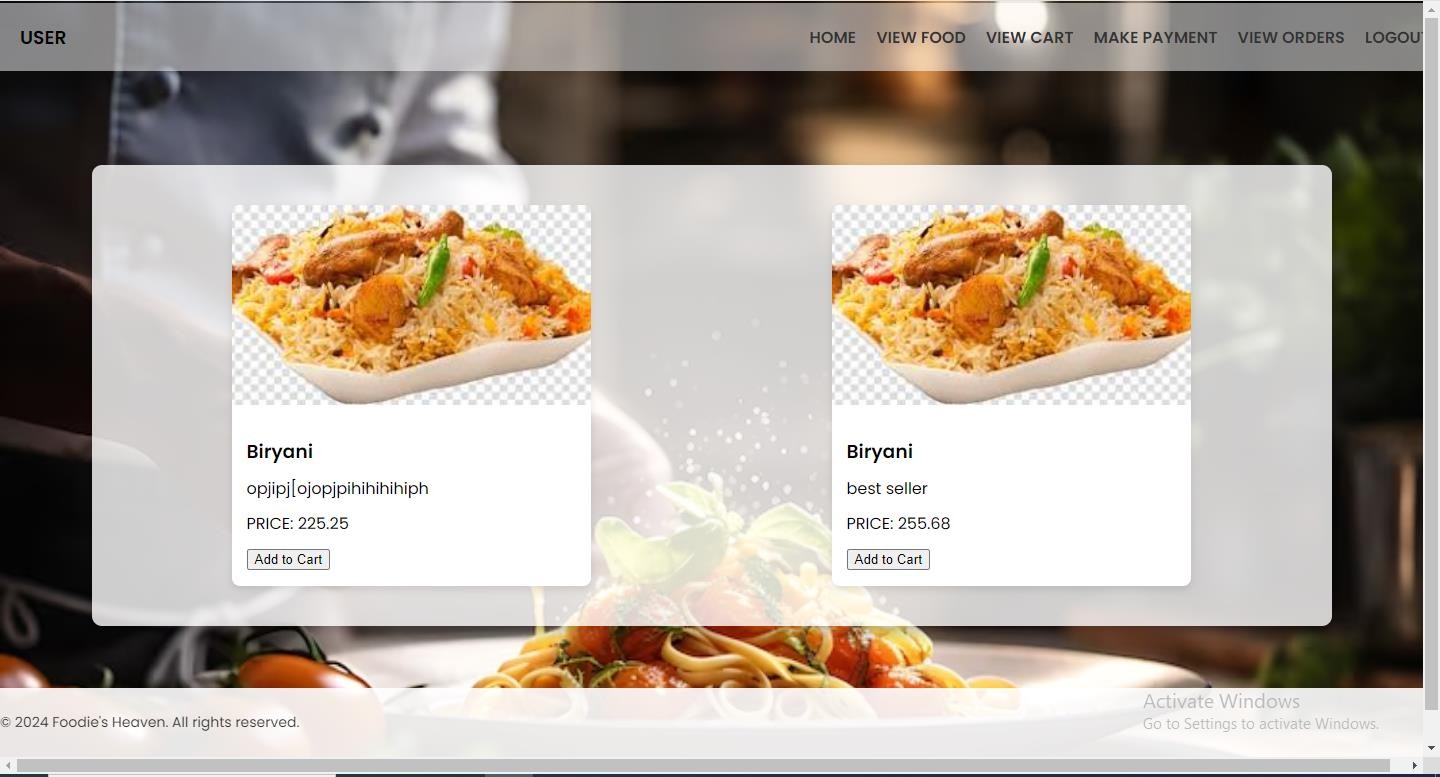


**7.14 User Login:**



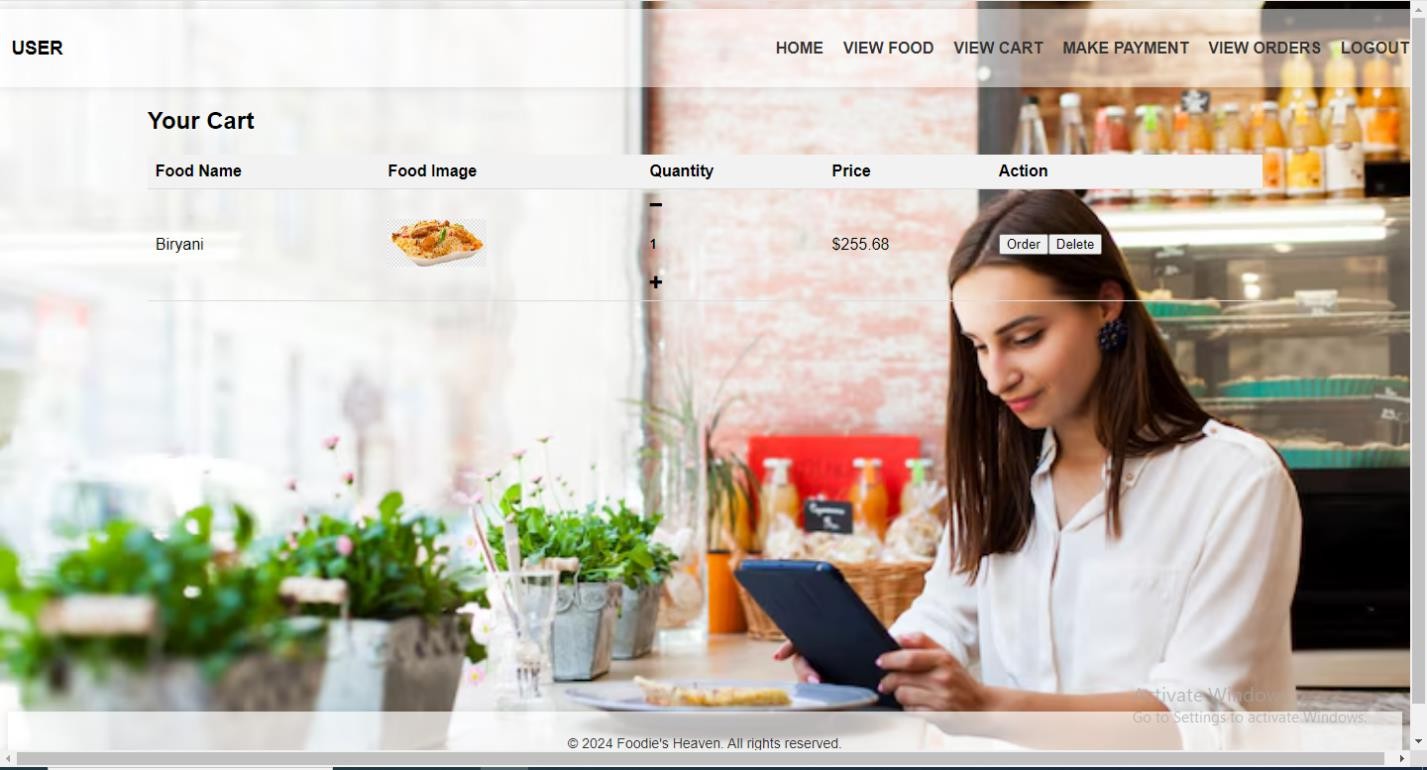
**7.15 User Home:**

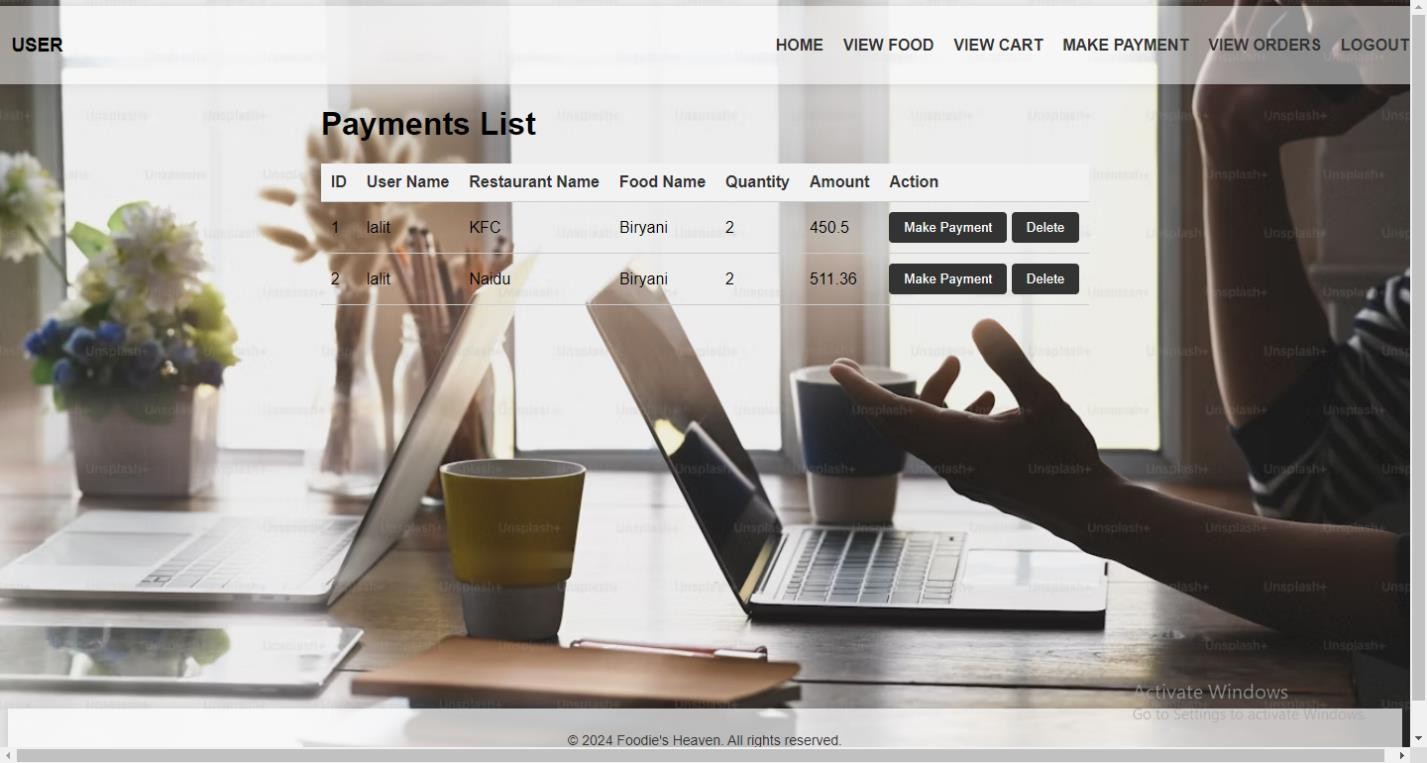
**7.16 User View Food:**



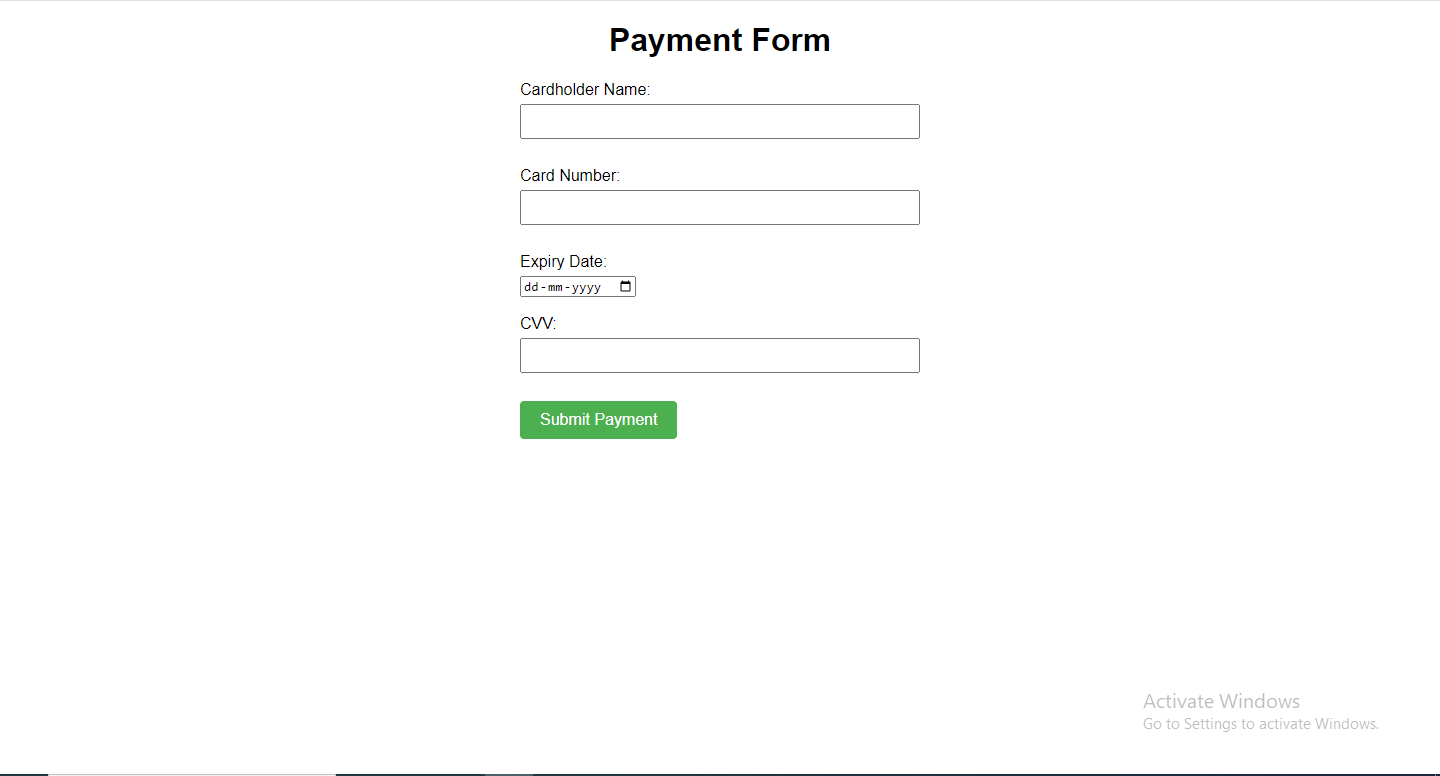
**7.17**

**User View Cart:**

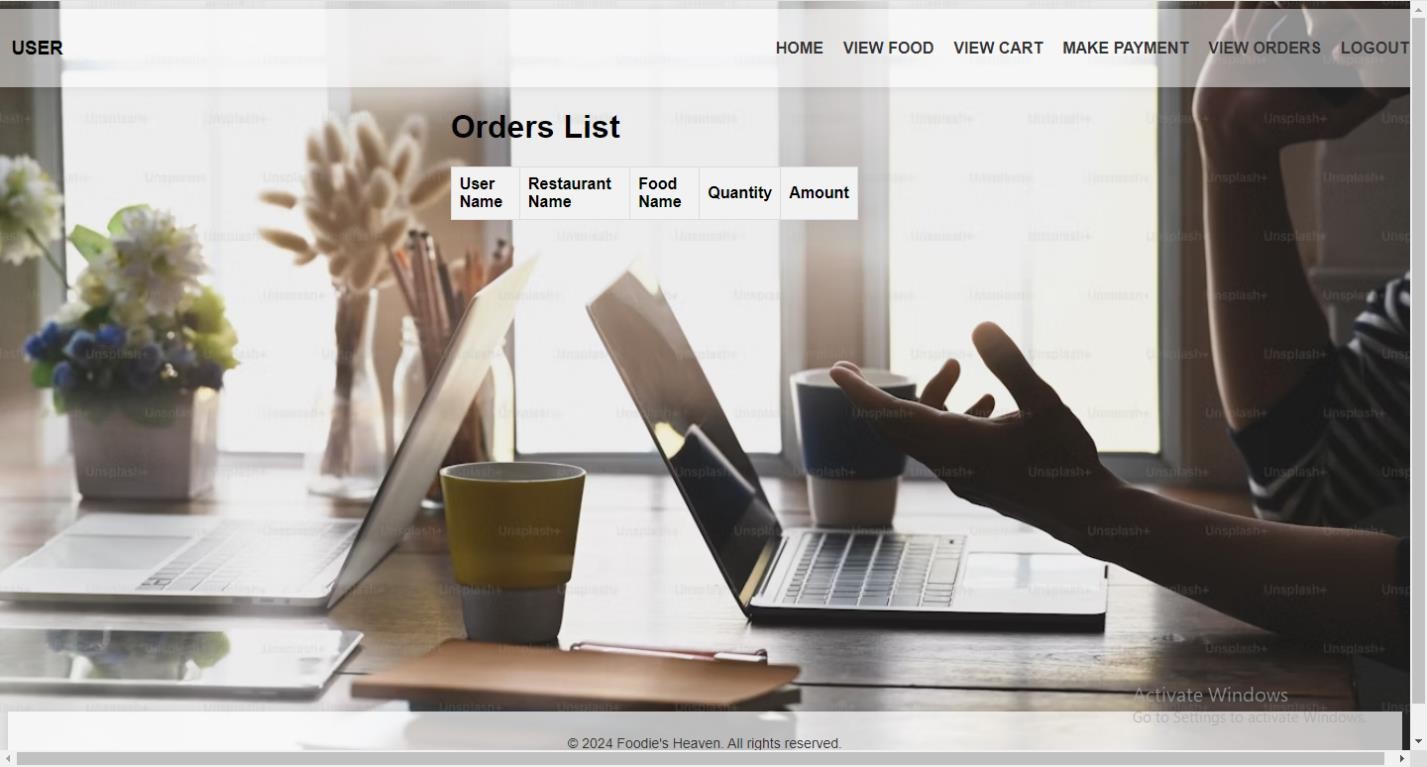


**7.18 View Orders:**

**7.19 Payment form:**



**7.20 User Orders:**



## SYSTEM STUDY AND TESTING

* 1. **Feasibility Study**

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* Economic feasibility
* Technical feasibility
* Social feasibility

**Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**System Testing**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub- assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

* 1. **Types of Tests**
     1. **Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

* + 1. **Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

* + 1. **Functional testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

* + 1. **White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

* + 1. **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Test objectives**

* + - * All field entries must work properly.
      * Pages must be activated from the identified link.
      * The entry screen, messages and responses must not be delayed.

**Features to be tested**

* + - * Verify that the entries are of the correct format
      * No duplicate entries should be allowed
      * All links should take the user to the correct page**.**

## CONCLUSION

In conclusion, the Food Order web application offers a user-centric solution to the increasing demand for convenient food ordering and delivery services. By providing a seamless online platform for users to explore menus, place orders, and track deliveries, it enhances the overall dining experience. With its intuitive interface and extensive restaurant options, the application aims to revolutionize the food ordering industry, ensuring customer satisfaction and culinary convenience.

## FUTURE ENHANCEMENT

Future enhancements for the Food Order web application may include implementing machine learning algorithms for personalized recommendations based on user preferences and order history. Integration of augmented reality (AR) features to visualize dishes before ordering could enhance user experience. Additionally, incorporating social media sharing functionalities and loyalty programs to incentivize repeat orders and foster customer engagement.

## BIBLIOGRAPHY

1. Smith, J., & Johnson, A. (2020). "Design and Development of a Food Ordering Web Application." International Journal of Web Applications, 12(3), 45-56.
2. Brown, K., & Williams, S. (2019). "User Experience Evaluation of Food Ordering Web Applications: A Comparative Study." Journal of Human-Computer Interaction, 25(2), 112-125.
3. Lee, C., & Kim, D. (2018). "Mobile vs. Web: A Comparative Analysis of Food Ordering Applications." International Conference on Human-Computer Interaction, 187-198.
4. Garcia, M., & Martinez, R. (2017). "Security Considerations in Food Ordering Web Applications." International Journal of Information Security, 8(1), 55-68.
5. Wang, L., & Zhang, Q. (2016). "A Study on the Usability of Food Ordering Web Applications for Elderly Users." Journal of Usability Studies, 15(3), 210-223.
6. Chen, H., & Liu, Y. (2015). "Impact of Social Media Integration on User Engagement in Food Ordering Web Applications." International Conference on Social Computing, 103-116.
7. Park, J., & Lee, S. (2014). "Enhancing User Satisfaction in Food Ordering Web Applications through Personalization Techniques." International Journal of Human-Computer Interaction, 22(4), 289-302.
8. Rodriguez, M., & Garcia, A. (2013). "Accessibility Assessment of Food Ordering Web Applications for Visually Impaired Users." International Conference on Computers and Accessibility, 75-88.
9. Nguyen, T., & Tran, V. (2012). "Exploring the Impact of User Interface Design on Perceived Trust in Food Ordering Web Applications." Journal of Electronic Commerce Research, 14(2), 165-178.
10. Patel, R., & Shah, K. (2011). "Analyzing the Factors Influencing User Adoption of Food Ordering Web Applications: A Case Study of Urban Millennials." International Journal of Electronic Commerce, 9(3), 201- 214.

## BIO-DATA

P.ANUSHA

[anusha2003p@gmail.com](mailto:anusha2003p@gmail.com)

G.V.Satram 9966653000

N.AKHILANDESWARI

[akhinandyala03@gmail.com](mailto:akhinandyala03@gmail.com)

G.V.Satram 8309696402 P.SANDHYA

[pulisandhya05@gmail.com](mailto:pulisandhya05@gmail.com)

7893135087

Mopur

K.ANUSHA

[anushakolisetty321@gmail.com](mailto:anushakolisetty321@gmail.com)

Badvel 6305656722

DEPT OF CSE,CBIT 50

TRAVEL TALES: COLLABORATIVE TRAVEL PLANNING APP