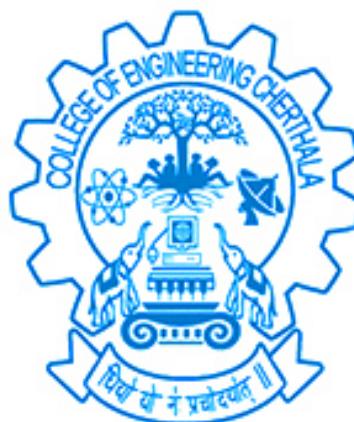


EATWISE
RESTAURANT MANAGEMENT SYSTEM
PROJECT REPORT

Submitted by
ARAVIND A KAMATH (CEC23CS041)
ARJUN MANOJ (CEC23CS043)
ANANDHAKRISHNAN S (CEC23CS033)
ARAVIND SAJEEV (CEC23CS042)

Under the esteemed guidance of
Mrs. Jincy Showri

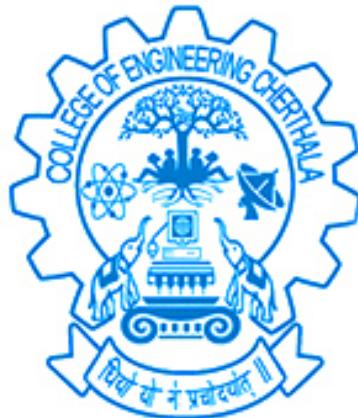
*In partial fulfilment of the requirements for the award of the degree
of
Bachelor of Technology
in
Computer Science and Engineering
of
APJ Abdul Kalam Technological University*



OCTOBER 2025

Department of Computer Science and Engineering
College of Engineering Cherthala
Pallipuram PO, Alappuzha - 688541
Phone: 0478 2553416, Fax: 0478 2552714
<https://www.ectl.ac.in>

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
COLLEGE OF ENGINEERING CHERTHALA
ALAPPUZHA - 688541**



C E R T I F I C A T E

This is to certify that, the project report titled EATWISE RESTAURANT MANAGEMENT SYSTEM is a bonafide record of the CSL333 Database Management Systems Lab - Project presented by ARAVIND A KAMATH (CEC23CS041), ARJUN MANOJ (CEC23CS043), ANANDHAKRISHNAN S (CEC23CS033), ARAVIND SAJEEV (CEC23CS042), fifth semester B. Tech. Computer Science & Engineering students, under our guidance and supervision, in partial fulfilment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of APJ Abdul Kalam Technological University

Internal Supervisor	UG Coordinator & Head of Department	External Supervisor
Mrs. Jincy Showri Assistant Professor Computer Science and Engineering	Mrs. Priya S Professor Computer Science & Engineering	

ACKNOWLEDGEMENT

This work would not have been possible without the support of many people, although not all of them can be mentioned. We wish to express our sincere gratitude to our parents for their love and support.

We would like to thank Dr Jaya V L, our esteemed principal, who has provided us the best facilities for the project completion and presentation. I would also like to thank our Head of Department (HOD), Dr Priya S (Professor, Computer Science & Engineering), our project coordinator Mrs. Jincy Showri (Assistant Professor, Computer Science & Engineering) for the help extended and also for the encouragement and support given to me while working on the project.

We would also like to thank our dear friends for extending their cooperation and encouragement throughout the project work, without which we would never have completed the project this well. Thank you all for your love, support and also for being understanding.

ABSTRACT

In the evolving modern world, technology must be used as a valuable tool to reduce human workload and manual labour. In our project, we have simplified the process of ordering and calculating in restaurant by taking advantage of technology.

In the traditional way, tracking orders, gathering total sum, and other such information proves to be difficult and a bit of hassle. With EatWise, this difficulty is alleviated as it digitalizes this process and it can be performed online. It contains login for both customer and administrator. Customer can order from menu (with quantity), see total/subtotal, delete order. Administrator can modify all the menu items, orders, suborders, user information, reviews.

EatWise method is much more simple and efficient for restaurant management when compared to the traditional way.

CONTENTS

1. INTRODUCTION

2. DESIGN

2.1 ER DIAGRAM

2.2 USECASE DIAGRAM

2.3 SEQUENTIAL DIAGRAM

3. IMPLEMENTATION

3.1 BACKEND

3.1.1 MENU ITEMS

3.1.2 ORDER ITEMS

3.1.3 ORDERS

3.1.4 REVIEWS

3.1.5 USERS

3.2 FRONTEND

3.2.1 HOME

3.2.2 LOGIN

3.2.3 REVIEWS

3.2.4 FOOTER

3.2.5 CUSTOMER LOGIN

3.2.6 ADMINISTRATOR LOGIN

3.3 PROGRAM CODE

4. HARDWARE AND SOFTWARE REQUIREMENTS

5. CONCLUSION

6. REFERENCES

INTRODUCTION

In the traditional way, restaurants manually track orders. There is a lack of centralization. This has a potential to lead to errors such as money sum errors, duplicate orders, mixing orders and more. It is necessary for centralized tracking and management of restaurant orders which uses a single database for all management actions. All the customers and administrators can see the same data in different forms and easily absorbs any changes/modifications and updates it accordingly.

Operations that can be performed:

Admin

- Menu Items - retrieval, addition, deletion, updation
- Order Items (suborder) - retrieval, deletion
- Orders - retrieval, deletion
- Reviews - retrieval, addition, deletion
- Users - retrieval, addition, deletion

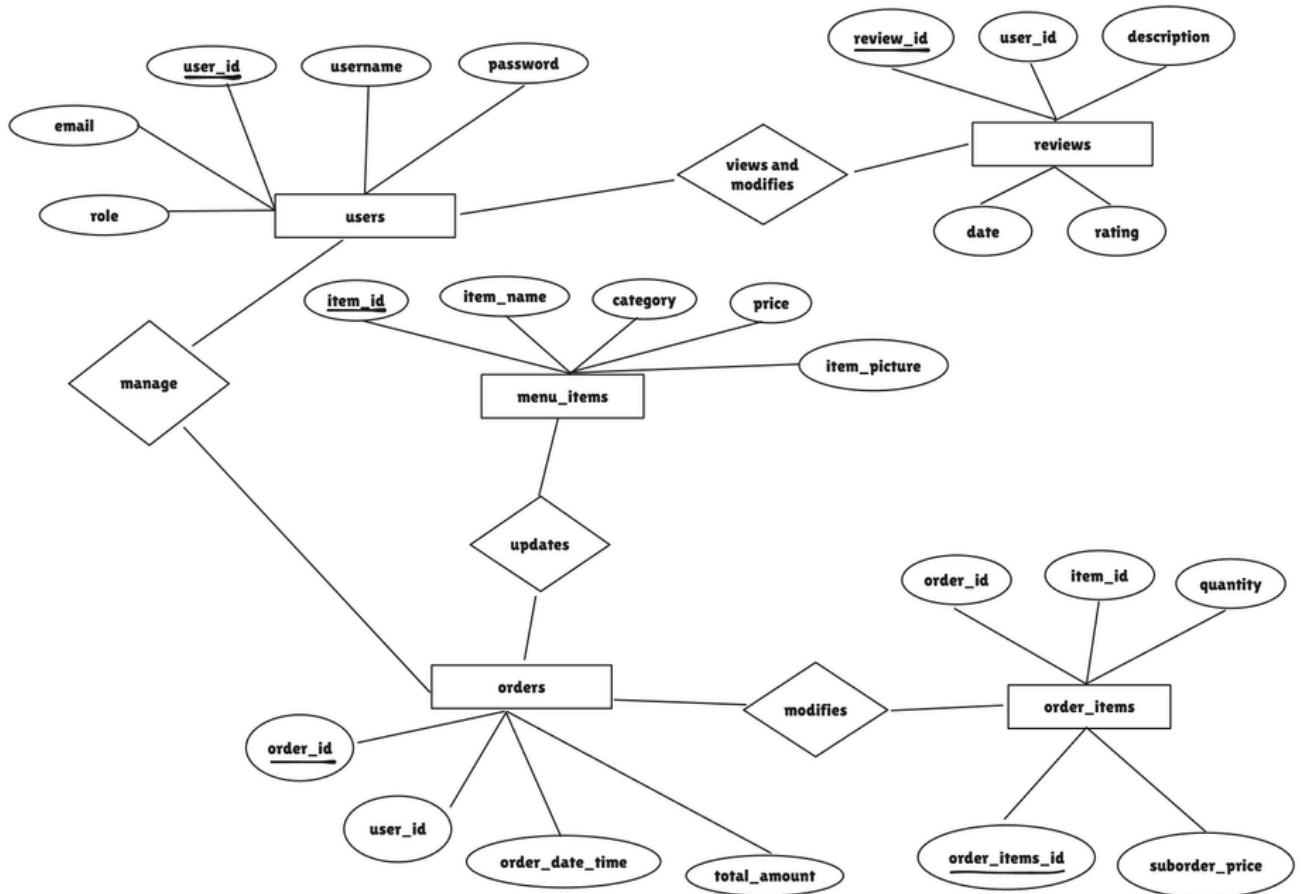
Customer

- Menu Items - retrieval
- Order Items (suborder) - retrieval, addition, deletion
- Reviews - retrieval

DESIGN

2.1 ER Diagram

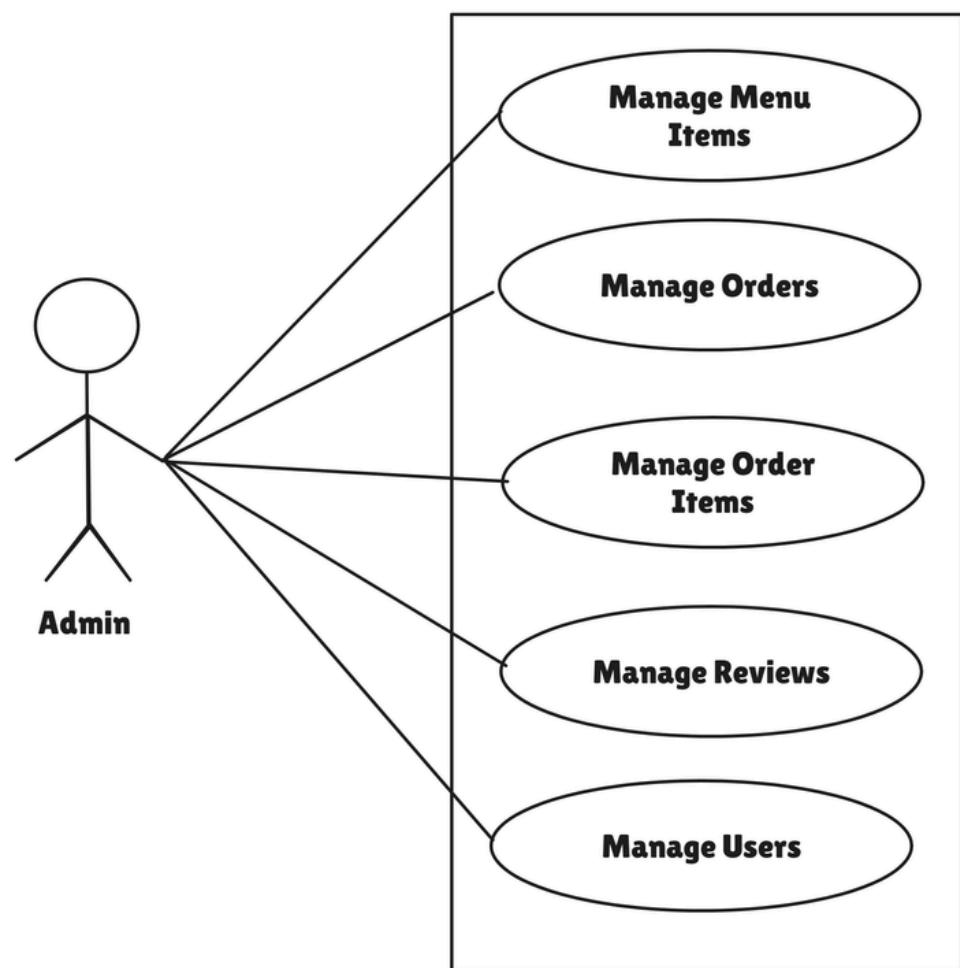
An entity-relationship (ER) diagram is a data-modelling technique that graphically illustrates an information system's entities and the relationships between them.



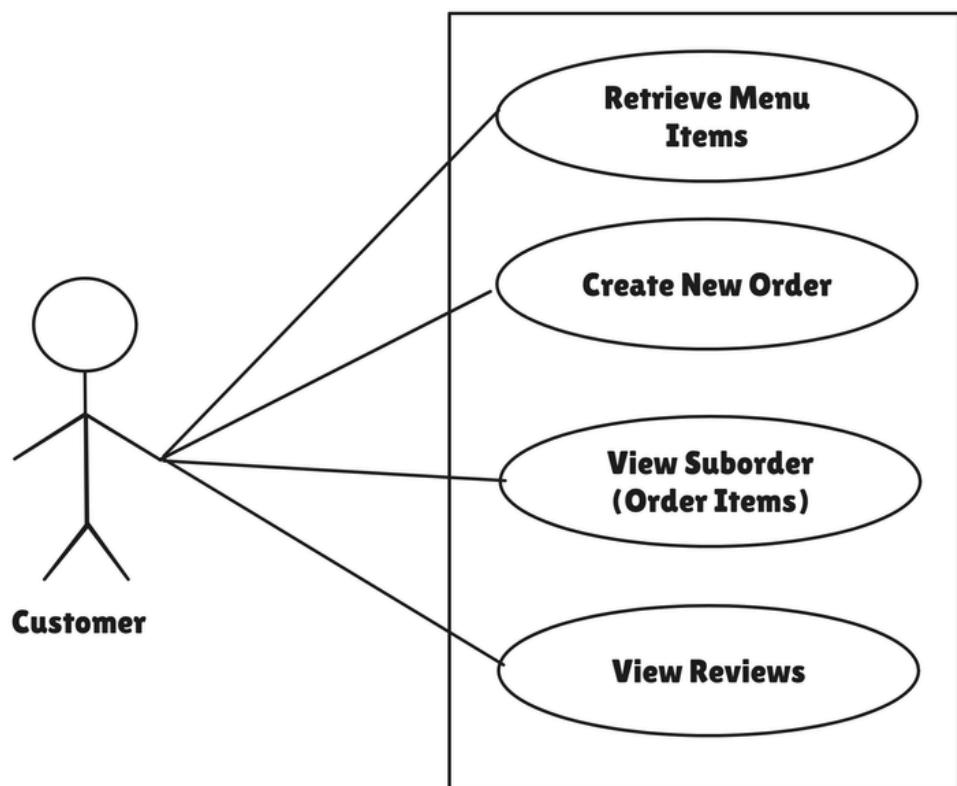
2.2 USECASE DIAGRAM

It is a graphic depiction of the interaction among the elements of EatWise Restaurant Management System

ADMIN

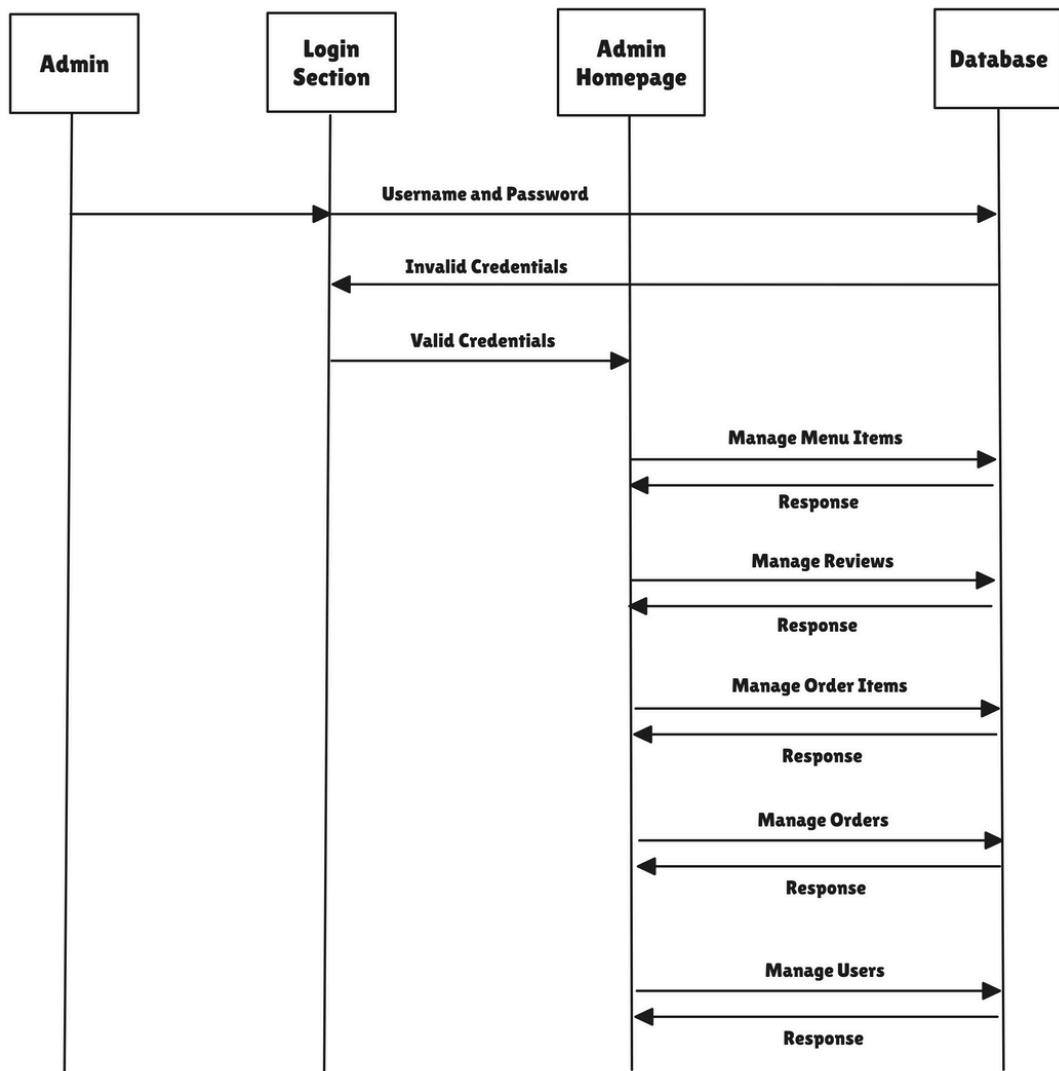


CUSTOMER

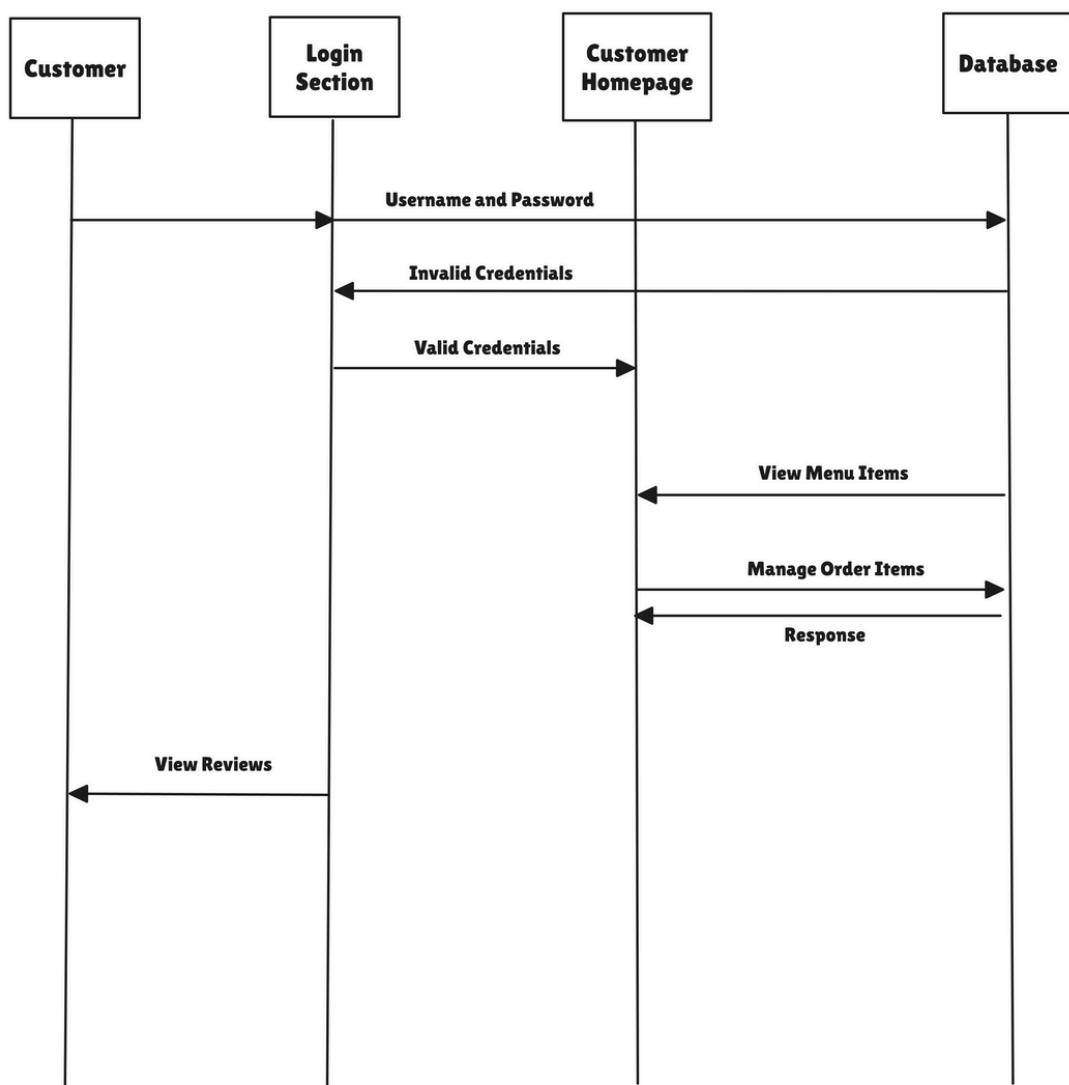


2.3 SEQUENTIAL DIAGRAM

ADMIN



CUSTOMER



IMPLEMENTATION

3.1 BACKEND

3.1.1 MENU ITEMS

The screenshot shows the MySQL Workbench interface with the 'menu_items' table selected in the 'restaurant_schema' database. The table has columns: item_id (int PK), item_name (varchar(45)), category (varchar(45)), price (int), and item_picture (varchar(100)). The result grid displays four rows of data:

item_id	item_name	category	price	item_picture
1	Puri	Food	40	https://www.sproutsfoods.com/wp-content/uploads/...
2	Kadale Curry	Food	40	https://www.sproutsfoods.com/wp-content/uploads/2021/0...
3	Pasta	Food	25	https://pescadofoods.com/wp-content/uploads/2023/02/...
4	Chicken Briyani	Food	170	https://theroyaltyroyal.com/wp-content/uploads/2023/10/...

The SQL pane contains the query: `SELECT * FROM restaurant_schema.menu_items`. The status bar shows the session was active from 09:31 PM on 20-Oct-25.

3.1.2 ORDER ITEMS

The screenshot shows the MySQL Workbench interface with the 'order_items' table selected in the 'restaurant_schema' database. The table has columns: order_items_id (int AI PK), order_id (int), item_id (int), quantity (int), and suborder_price (float). The result grid displays three rows of data:

order_items_id	order_id	item_id	quantity	suborder_price
30	11	1	1	40
31	11	4	4	680

The SQL pane contains the query: `SELECT * FROM restaurant_schema.order_items`. The status bar shows the session was active from 09:32 PM on 20-Oct-25.

3.1.3 ORDERS

The screenshot shows the MySQL Workbench interface with the 'orders' table selected in the schema browser. The table has four columns: order_id (int AI PK), user_id (int), order_date_time (datetime), and total_amount (float). A single row is displayed in the result grid:

order_id	user_id	order_date_time	total_amount
11	5	2025-10-20 13:28:58	720

The SQL pane contains the query: `SELECT * FROM restaurant_schema.orders`. The output pane shows the execution log with three entries:

- 12 21:31:43 SELECT * FROM restaurant_schema.menu_items LIMIT 0, 1000
- 13 21:32:12 SELECT * FROM restaurant_schema.order_items LIMIT 0, 1000
- 14 21:32:32 SELECT * FROM restaurant_schema.orders LIMIT 0, 1000

3.1.4 REVIEWS

The screenshot shows the MySQL Workbench interface with the 'reviews' table selected in the schema browser. The table has five columns: review_id (int AI PK), user_id (int), description (text), date (date), and rating (int). Multiple rows are displayed in the result grid:

review_id	user_id	description	date	rating
1	1	One of the best experiences I ha...	2025-09-28	5
2	2	First time I ordered online and...	2025-09-29	4
3	3	Indian food always have great fla...	2025-09-27	4
4	4	Tasty food! If you haven't tried ...	2025-09-29	5
5	5	Authentic Kerala flavor in everyt...	2025-09-25	4
6	6	Always a great meal, highly recomme...	2025-09-28	5
7	7	Hands down the best place I've e...	2025-09-21	5
8	8	Amazing hospitality and delicious...	2025-09-27	5
9	9	I can't stop thinking about the C...	2025-09-25	5

The SQL pane contains the query: `SELECT * FROM restaurant_schema.reviews`. The output pane shows the execution log with three entries:

- 13 21:32:12 SELECT * FROM restaurant_schema.order_items LIMIT 0, 1000
- 14 21:32:32 SELECT * FROM restaurant_schema.orders LIMIT 0, 1000
- 15 21:32:46 SELECT * FROM restaurant_schema.reviews LIMIT 0, 1000

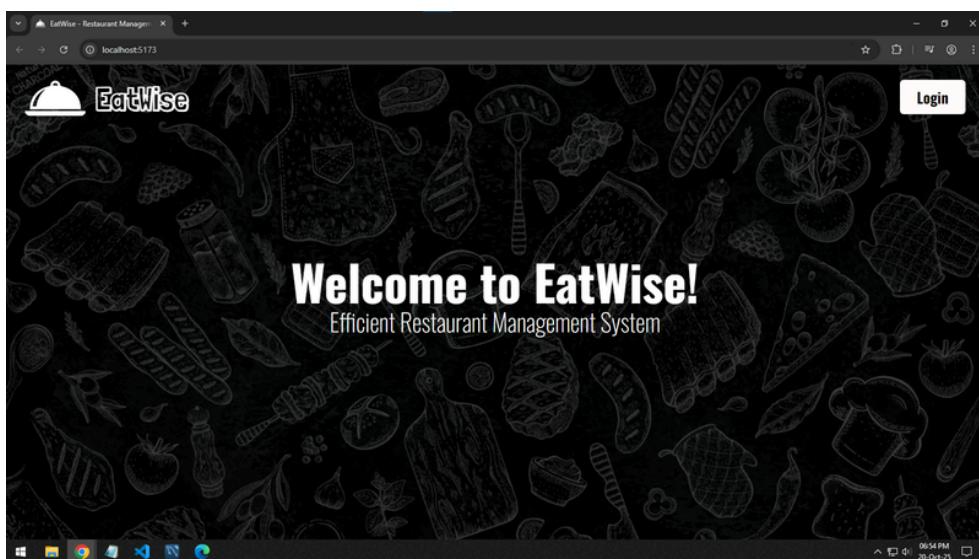
3.1.5 USERS

The screenshot shows the MySQL Workbench interface with the 'users' table selected in the schema browser. The table has columns: user_id (int AI PK), username (varchar(45)), password (varchar(45)), email (varchar(45)), and role (enum). The data grid displays 11 rows of user information, including names like Arvind, and Arvind, along with their respective roles (Customer or Admin).

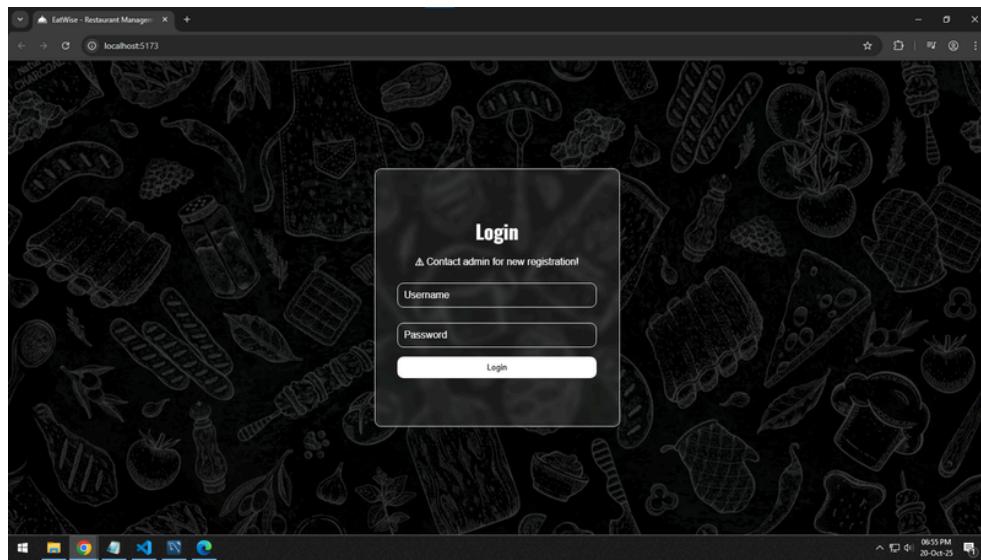
user_id	username	password	email	role
1	Arvind Karthik	Arvind2005	arvindkarthik@gmail.com	Admin
2	Arun Mehta	Arvin2005		Customer
3	Aravind Sapev	Arvind2005		Customer
4	Anandakrishnan S	Anandhu2005		Customer
5	Sarvesh Arvind	Sarvesh2005		Customer
6	Sarvesh Krishan S	Sarvesh2005		Customer
7	Senthil Selvaraj	Senthil2005		Customer
8	Unnilkanthan A	Unnilkanthan2005		Customer
9	Auf Subar	Auf2005		Customer
11	Jeng Shewi	Jeng2005		Admin

3.2 FRONTEND

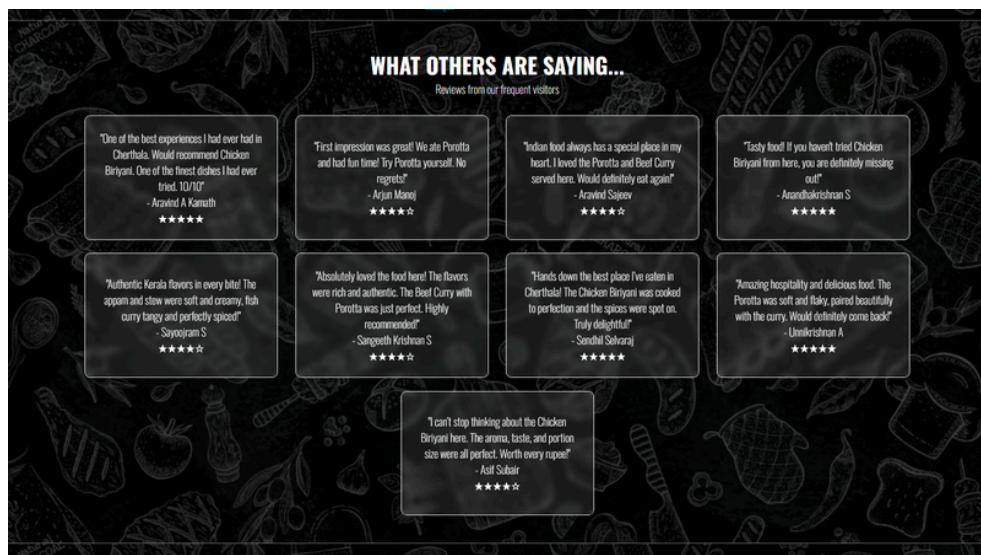
3.2.1 HOME



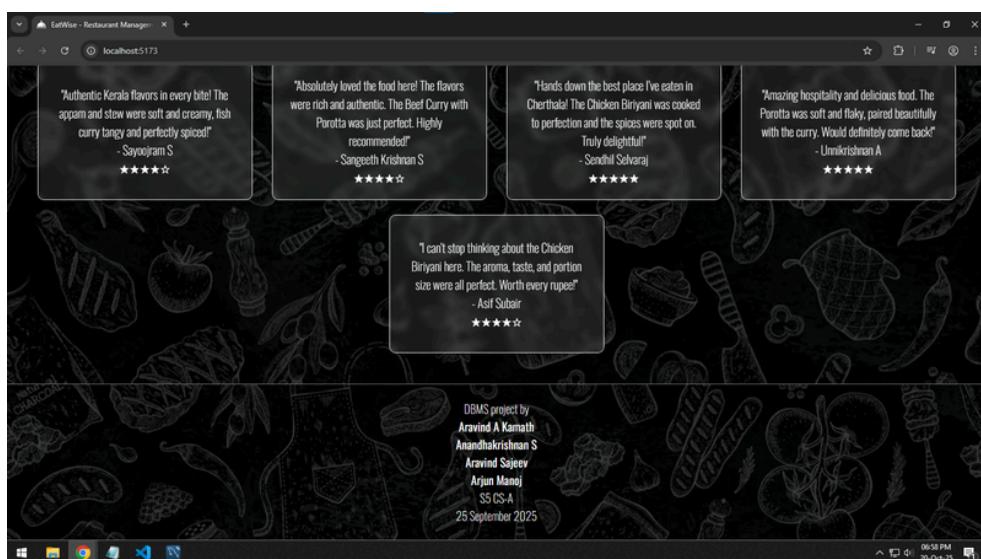
3.2.2 LOGIN



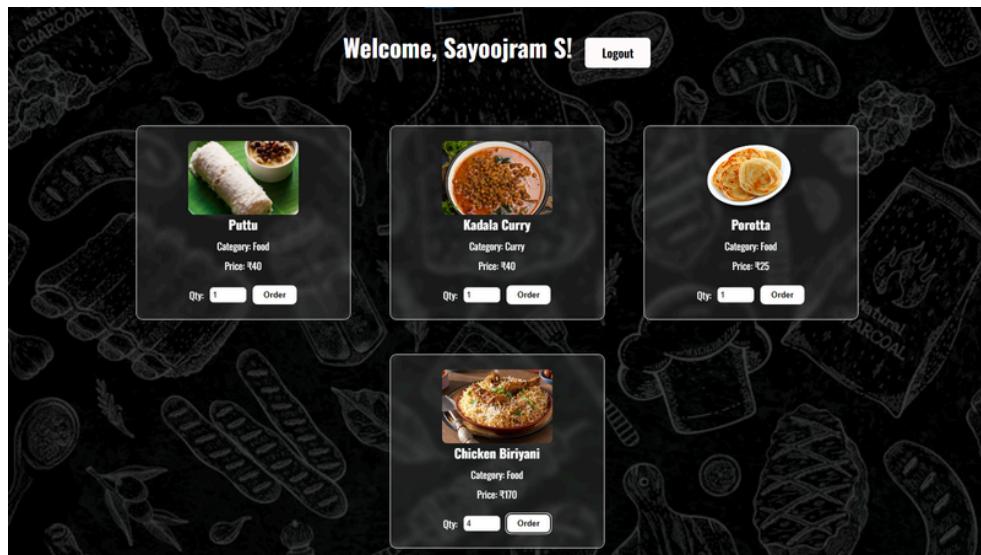
3.2.3 REVIEWS



3.2.4 FOOTER



3.2.5 CUSTOMER LOGIN



Let's Go - Restaurant Manager X

localhost:5173/customer

Puttu
Qty: 1 [Order](#)

Kadala Curry
Qty: 1 [Order](#)

Porotta
Qty: 8 [Order](#)

Chicken Biriyani
Category: Food
Price: ₹170
Qty: 4 [Order](#)

Your Orders

Order Item ID	Item Name	Quantity	Suborder Price
30	Puttu	1	₹40
31	Chicken Biriyani	4	₹680

06:59 PM 20-Oct-25

3.2.6 ADMINISTRATOR LOGIN

Welcome, Jincy Showri! [Logout](#)

Menu Items

ID	Item	Description	Category	Price
1	Puttu	A traditional South Indian breakfast dish made from rice flour.	Food	₹40
2	Kadala Curry	A creamy, spicy curry made with chickpeas.	Curry	₹40
3	Porotta	A soft, round bread stuffed with various fillings.	Food	₹25
4	Chicken Biriyani	A flavorful rice dish with chicken and spices.	Food	₹110

[Delete](#) [Update](#) [Add New Item](#)

Reviews Management

ID	User	Description	Date	Rating	Delete
1	Aravind A Kanath	One of the best experiences I had ever had in Cherthala. Would recommend Chicken Biriyani. One of the finest dishes I had ever tried. 10/10	2025-09-27 18:30:00.000Z	5	Delete
2	Arjun Manoj	Firrt impression was great! We ate Porotta and had fun time! Try Porotta yourself. No regrets!	2025-09-28 18:30:00.000Z	4	Delete
3	Aravind Sajeet	Indian food always has a special place in my heart. I loved the Porotta and Beet Curry served here. Would definitely eat again!	2025-09-26 18:30:00.000Z	4	Delete
4	Anandakrishnan S	Tasty food if you haven't tried Chicken Biriyani from here, you are definitely missing out!	2025-09-28 18:30:00.000Z	5	Delete
5	Sayegram S	Authentic Kerala flavors in every bite! The appam and stew were soft and creamy, Beet curry tangy and perfectly spiced!	2025-09-24 18:30:00.000Z	4	Delete
6	Sangeth Krishnan S	Absolutely loved the food here! The flavors were rich and authentic. The Beet Curry with Porotta was just perfect. Highly recommended!	2025-09-24 18:30:00.000Z	4	Delete
7	Senthil Selvaraj	Hands down the best place I've eaten in Cherthala! The Chicken Biriyani was cooked to perfection and the spices were spot on. Truly delightful!	2025-09-20 18:30:00.000Z	5	Delete
8	Umeshkrishnan A	Amazing hospitality and delicious food. The Porotta was soft and fluffy, paired beautifully with the curry. Would definitely come back!	2025-09-26 18:30:00.000Z	5	Delete
9	Aatif Sabair	I can't stop thinking about the Chicken Biriyani here. The aroma, taste, and portion size were all perfect. Worth every rupee!	2025-09-24 18:30:00.000Z	4	Delete

[Add New Item](#)

Reviews Management

ID	User	Description	Date	Rating	Delete
1	Aravind A Kanath	One of the best experiences I had ever had in Cherthala. Would recommend Chicken Biriyani. One of the finest dishes I had ever tried. 10/10	2025-09-27 18:30:00.000Z	5	Delete
2	Arjun Manoj	Firrt impression was great! We ate Porotta and had fun time! Try Porotta yourself. No regrets!	2025-09-28 18:30:00.000Z	4	Delete
3	Aravind Sajeet	Indian food always has a special place in my heart. I loved the Porotta and Beet Curry served here. Would definitely eat again!	2025-09-26 18:30:00.000Z	4	Delete
4	Anandakrishnan S	Tasty food if you haven't tried Chicken Biriyani from here, you are definitely missing out!	2025-09-28 18:30:00.000Z	5	Delete
5	Sayegram S	Authentic Kerala flavors in every bite! The appam and stew were soft and creamy, Beet curry tangy and perfectly spiced!	2025-09-24 18:30:00.000Z	4	Delete
6	Sangeth Krishnan S	Absolutely loved the food here! The flavors were rich and authentic. The Beet Curry with Porotta was just perfect. Highly recommended!	2025-09-24 18:30:00.000Z	4	Delete
7	Senthil Selvaraj	Hands down the best place I've eaten in Cherthala! The Chicken Biriyani was cooked to perfection and the spices were spot on. Truly delightful!	2025-09-20 18:30:00.000Z	5	Delete
8	Umeshkrishnan A	Amazing hospitality and delicious food. The Porotta was soft and fluffy, paired beautifully with the curry. Would definitely come back!	2025-09-26 18:30:00.000Z	5	Delete
9	Aatif Sabair	I can't stop thinking about the Chicken Biriyani here. The aroma, taste, and portion size were all perfect. Worth every rupee!	2025-09-24 18:30:00.000Z	4	Delete

[Add New Item](#)

Orders Management

Order ID	User ID	Order Date Time	Total Amount	Delete
11	5	2025-10-20 07:58:58.000Z	720	Delete

Order Items Management

Order Item ID	Order ID	Item ID	Quantity	Suborder Price	Delete
30	11	1	1	600	Delete
31	11	4	4	600	Delete

Order Item Management

Order Item ID	Order ID	Suborder Price	Delete
30	11	600	Delete
31	11	600	Delete

To exit full screen, press and hold Esc

Users Management

ID	Username	Password	Email	Role	Delete
1	Aravind A Kanath	Aravind2005	aravindanath1@gmail.com	Admin	Delete
2	Arjun Manoj	Arjun2005		Customer	Delete
3	Aravind Sajeet	AravindS2005		Customer	Delete
4	Anandakrishnan S	Anandhu2005		Customer	Delete
5	Sayegram S	Sayegram2005		Customer	Delete
6	Sangeth Krishnan S	Sangeth2005		Customer	Delete
7	Senthil Selvaraj	Senthil2005		Customer	Delete
8	Umeshkrishnan A	Umesh2005		Customer	Delete
9	Aatif Sabair	Aatif2005		Customer	Delete
10	Jincy Showri	Jincy2025		Admin	Delete

[Add User](#)

DBMS project by
Aravind A Kanath
Anandakrishnan S
Aravind Sajeet
Arjun Manoj
SS CSA
25 September 2025

3.3 PROGRAM CODE

PROGRAM FILE STRUCTURE

The image shows a file explorer interface with two panes. The left pane displays the directory structure of the 'RESTAURANT-MANAGEMENT-SYSTEM' project. The right pane lists various files and assets.

Left Pane (Project Structure):

- RESTAURANT-MANAGEMENT-SYSTEM
 - backend
 - node_modules
 - index.js
 - package-lock.json
 - package.json
 - frontend
 - node_modules
 - public
 - favicon.svg
 - src
 - assets
 - abstract_background_image.jpg
 - cloche_icon.png
 - pages
 - Add.jsx
 - AdminLanding.jsx
 - Credits.jsx
 - CustomerLanding.jsx
 - Hero.jsx
 - Homepage.jsx
 - Login.jsx
 - MenuItems.jsx
 - ProtectedRoute.jsx
 - Reviews.jsx
 - Update.jsx
 - App.jsx
 - # index.css
 - main.jsx

index.css

```
/* Reset default styles */
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}

/* General styles */
html {
scroll-behavior: smooth;
}

::webkit-scrollbar {
width: 5px;
}

::webkit-scrollbar-track {
background-color: rgb(11, 11, 11);
}

::webkit-scrollbar-thumb {
background-color: hsl(0, 0%, 100%);
border-radius: 1px;
}

a {
color: inherit;
text-decoration: none;
}

/* Homepage */
.page-background {
background-color: #181818;
height: 100vh;
color: white;
background-image: url("./assets/abstract_background_image.jpg");
background-size: cover;
position: relative;
}

.navbar {
display: flex;
flex-direction: row;
padding: 25px 30px 25px 30px;
gap: 10px;
justify-content: space-between;
}

.logo-div h1 {
font-family: "Londrina Shadow", sans-serif;
color: white;
font-size: 3.25rem;
}

.logo-img {
width: 100px;
}

.logo-div {
display: flex;
flex-direction: row;
align-items: center;
justify-content: center;
gap: 1rem;
}

.navbar button {
background-color: #fffffd;
border: none;
color: black;
border-radius: 6px;
padding: 10px 28px;
font-family: "Oswald", sans-serif;
font-size: 1.5rem;
font-weight: 600;
transition: all 0.3s ease;
cursor: pointer;
}

.navbar button:hover {
background-color: #929292;
}

.hero-text {
font-family: "Oswald", sans-serif;
text-align: center;
position: absolute;
font-size: 2.5rem;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
width: 100%;
z-index: 1;
line-height: 4.1rem;
}

.hero-text p {
font-weight: 200;
}

/* Login Page */
.login-section {
margin: 0;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
background-size: cover;
background-image: url("./assets/abstract_background_image.jpg");
}

.glass-container {
width: 400px;
height: 420px;
position: relative;
display: flex;
align-items: center;
justify-content: center;
z-index: 1;
background: rgba(255, 255, 255, 0.1);
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
border-radius: 10px;
border: 1px solid #fff;
}

.glass-container::before {
content: "";
position: absolute;
width: 100%;
height: 100%;
border-radius: 10px;
background: linear-gradient(45deg, transparent, #fff, transparent);
filter: blur(5px);
-webkit-backdrop-filter: blur(5px);
z-index: -1;
}

.login-box {
width: 325px;
margin: 0 auto;
text-align: center;
margin: 0;
}

.login-section h2 {
font-size: 2rem;
color: #fff;
```

```

font-family: "Oswald", sans-serif;
}

.login-section form {
display: flex;
flex-direction: column;
}

.login-section input {
padding: 10px;
margin-top: 25px;
border: none;
border-radius: 10px;
background: transparent;
border: 1px solid #fff;
color: #fff;
font-size: 1rem;
width: 100%;
}

.login-section input::placeholder {
color: #fff;
}

.login-section input:focus {
outline: none;
}

.login-section button {
background: #fff;
color: black;
padding: 10px;
border: none;
border-radius: 10px;
cursor: pointer;
margin-top: 15px;
transition: all 0.3s ease;
}

.login-section button:hover {
background: transparent;
color: white;
outline: 1px solid #fff;
}

.login-section p {
font-family: 'Poppins', sans-serif;
color: #fff;
margin-top: 15px;
}

/* Reviews Section */
.reviews-section {
color: white;
display: flex;
flex-direction: column;
font-family: 'Oswald', sans-serif;
padding: 50px 20px;
background-size: cover;
background-image: url("./assets/abstract_background_image.jpg");
}

.reviews-section h2 {
text-align: center;
font-size: 2.5rem;
}

.reviews-section p {
text-align: center;
font-size: 1.25rem;
font-weight: 200;
}

.reviews-container {
margin-top: 10px;
display: flex;
flex-wrap: wrap;
align-items: center;
justify-content: center;
text-align: center;
}

.review-card {
width: 350px;
height: 225px;
position: relative;
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
z-index: 1;
background: rgba(255, 255, 255, 0.1);
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
border-radius: 10px;
border: 1px solid #fff;
padding: 20px 25px;
margin: 1.5rem 1rem 0 1rem;
}

.review-card::before {
content: "";
position: absolute;
width: 100%;
height: 100%;
border-radius: 10px;
background-color: #fff;
border: 1px solid #fff;
border-width: 0 1px 1px 0;
border-style: solid;
border-collapse: collapse;
border-top-left-radius: 0;
border-bottom-left-radius: 0;
border-top-right-radius: 10px;
border-bottom-right-radius: 10px;
z-index: -1;
}

/* Credits Section */
.credits-section {
color: white;
display: flex;
flex-direction: column;
font-family: 'Oswald', sans-serif;
padding: 25px 20px;
font-size: 1.2rem;
background-size: cover;
background-image: url("./assets/abstract_background_image.jpg");
text-align: center;
}

.light-emphasis {
font-weight: 200;
}

/* Customer Landing Page */
.customer-landing-section {
color: white;
font-family: 'Oswald', sans-serif;
padding-bottom: 50px;
min-height: 100vh;
background-size: cover;
background-image: url("./assets/abstract_background_image.jpg");
}

.customer-landing-section-header {
display: flex;
gap: 20px;
align-items: center;
justify-content: center;
padding: 30px 0;
}

.customer-landing-section-header h2 {
font-size: 2.5rem;
}

.customer-landing-section-header button {
background-color: #fffffd;
border: none;
color: black;
border-radius: 6px;
padding: 10px 28px;
font-family: "Oswald", sans-serif;
}

```

```

font-size: 1.2rem;
font-weight: 600;
transition: all 0.3s ease;
cursor: pointer;
}

.customer-landing-section-header button:hover {
background-color: #929292;
}

.menu-items-container {
display: flex;
flex-wrap: wrap;
justify-content: center;
align-items: flex-start;
gap: 2rem;
margin: 40px 0;
}

.text-aligner {
text-align: center;
}

.font-changer {
font-family: 'Oswald', sans-serif;
font-size: 1.2rem;
}

.menu-item-card {
width: 350px;
position: relative;
display: flex;
flex-direction: column;
align-items: center;
justify-content: flex-start;
z-index: 1;
background: rgba(255, 255, 255, 0.1);
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
border-radius: 10px;
border: 1px solid #fff;
padding: 24px 20px;
margin: 1.5rem 1rem 0 1rem;
}

.menu-item-card::before {
content: "";
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
border-radius: 10px;
backdrop-filter: blur(5px);
-webkit-backdrop-filter: blur(5px);
z-index: -1;
}

.menu-item-card img {
width: 180px;
height: 120px;
object-fit: cover;
border-radius: 8px;
}

.menu-item-card h3 {
font-size: 1.3rem;
margin-bottom: 8px;
}

.menu-item-card p {
font-size: 1rem;
margin-bottom: 8px;
}

.menu-item-actions {
display: flex;
align-items: center;
gap: 10px;
}

margin-top: 12px;
}

.menu-item-card button {
background: #fff;
color: black;
padding: 8px 18px;
border: none;
border-radius: 8px;
cursor: pointer;
font-weight: 600;
transition: all 0.2s;
}

.menu-item-card button:hover {
background: #929292;
color: white;
}

.order-items-container {
margin: 40px auto 0 auto;
max-width: 900px;
background: rgba(255,255,255,0.08);
border-radius: 10px;
box-shadow: 0 2px 8px rgba(0,0,0,0.08);
padding: 24px;
min-height: 120px;
max-height: 400px;
overflow-y: auto;
color: white;
position: relative;
}

.order-items-table {
width: 100%;
border-collapse: collapse;
color: white;
background: transparent;
}

.order-items-table th, .order-items-table td {
padding: 10px;
border-bottom: 1px solid #fff;
text-align: center;
}

.order-items-table th {
background: rgba(255,255,255,0.12);
}

/* Admin Landing Page */
.admin-landing-section {
color: white;
font-family: 'Oswald', sans-serif;
padding-bottom: 20px;
min-height: 100vh;
background-size: cover;
background-image: url("./assets/abstract_background_image.jpg");
}

.admin-landing-section-header {
display: flex;
gap: 20px;
align-items: center;
justify-content: center;
padding: 30px 0;
}

.admin-landing-section-header h2 {
font-size: 2.5rem;
}

.admin-landing-section-header button {
background-color: #fffffd;
border: none;
color: black;
border-radius: 6px;
padding: 10px 28px;
}

```

```

font-family: "Oswald", sans-serif;
font-size: 1.2rem;
font-weight: 600;
transition: all 0.3s ease;
cursor: pointer;
}

.admin-landing-section-header button:hover {
background-color: #929292;
}

.admin-menu-items-container {
display: flex;
flex-wrap: wrap;
justify-content: center;
align-items: flex-start;
gap: 2rem;
margin: 10px 0;
}

.admin-menu-item-card {
width: 350px;
position: relative;
display: flex;
flex-direction: column;
align-items: center;
justify-content: flex-start;
z-index: 1;
background: rgba(255, 255, 255, 0.1);
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
border-radius: 10px;
border: 1px solid #fff;
padding: 24px 20px;
margin: 1.5rem 1rem 0 1rem;
}

.admin-menu-item-card::before {
content: "";
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
border-radius: 10px;
backdrop-filter: blur(5px);
-webkit-backdrop-filter: blur(5px);
z-index: -1;
}

.admin-menu-item-card img {
width: 180px;
height: 200px;
object-fit: cover;
border-radius: 8px;
margin-bottom: 10px;
}

.admin-menu-item-card h3 {
font-size: 1.3rem;
margin-bottom: 8px;
}

.admin-menu-item-card p {
font-size: 1rem;
margin-bottom: 2px;
}

.admin-menu-item-actions {
display: flex;
align-items: center;
gap: 10px;
margin-top: 15px;
}

.admin-menu-item-card button {
background: #fff;
color: black;
padding: 8px 18px;
border: none;
border-radius: 8px;
cursor: pointer;
font-weight: 600;
transition: all 0.2s;
}

.admin-menu-item-card button:hover {
background: #929292;
color: white;
}

.admin-menu-item-actions button,
.admin-menu-items-container + div button {
background-color: #ffffdf;
border: none;
color: black;
border-radius: 6px;
padding: 10px 28px;
font-family: "Oswald", sans-serif;
font-size: 1.2rem;
font-weight: 600;
transition: all 0.3s ease;
cursor: pointer;
}

.admin-menu-item-actions button:hover,
.admin-menu-items-container + div button:hover {
background-color: #929292;
color: white;
}

.admin-form {
display: flex;
gap: 10px;
margin: 20px 0;
background: rgba(255,255,255,0.08);
border-radius: 10px;
box-shadow: 0 2px 8px rgba(0,0,0,0.08);
padding: 24px;
align-items: center;
justify-content: center;
}

.admin-form input,
.admin-form select {
background: rgba(255,255,255,0.15);
border: 1px solid #fff;
border-radius: 8px;
padding: 10px 16px;
color: white;
font-size: 1rem;
transition: box-shadow 0.2s;
box-shadow: 0 2px 6px rgba(0,0,0,0.08);
min-width: 180px;
height: 44px;
margin-bottom: 0;
}

.admin-form select {
background: rgba(30,41,59,0.25);
color: #fff;
border: 1px solid #fff;
border-radius: 8px;
padding: 10px 16px;
font-size: 1rem;
box-shadow: 0 2px 6px rgba(0,0,0,0.08);
min-width: 180px;
height: 44px;
}

.admin-form select option {
background: #222;
color: #fff;
}

.admin-form label.admin-role-label {
color: white;
font-size: 1rem;
}

```

```

min-width: 60px;
text-align: left;
margin-right: 0;
margin-bottom: 0;
height: 44px;
display: flex;
align-items: center;
}

.admin-form button {
background: #32a852;
color: white;
border: none;
border-radius: 8px;
padding: 10px 24px;
font-weight: 600;
cursor: pointer;
transition: background 0.2s;
min-width: 120px;
height: 44px;
display: flex;
align-items: center;
justify-content: center;
}

.admin-form button:hover {
background: #218c3a;
}

.admin-landing-section input::placeholder {
color: white !important;
}

/* Add New Items page */
.add-section {
min-height: 100vh;
display: flex;
align-items: center;
justify-content: center;
background-image:
url('../src/assets/abstract_background_image.jpg');
background-size: cover;
color: white;
}

.add-glass-container {
width: 420px;
font-family: 'Oswald', sans-serif;
padding: 36px 32px;
background: rgba(255,255,255,0.12);
border-radius: 16px;
box-shadow: 0 4px 16px rgba(0,0,0,0.12);
border: 1px solid #fff;
position: relative;
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
z-index: 1;
}

.add-glass-container::before {
content: "";
position: absolute;
width: 100%;
height: 100%;
border-radius: 16px;
backdrop-filter: blur(8px);
-webkit-backdrop-filter: blur(8px);
z-index: -1;
}

.add-glass-container h2 {
font-size: 2rem;
margin-bottom: 24px;
text-align: center;
}

.add-form {
display: flex;
flex-direction: column;
gap: 18px;
width: 100%;
align-items: center;
}

.add-form input,
.add-form select {
width: 90%;
background: rgba(255,255,255,0.18);
border: 1px solid #fff;
border-radius: 8px;
padding: 12px 16px;
color: white;
font-size: 1rem;
margin-bottom: 0;
}

.add-form input:focus,
.add-form select:focus {
outline: none;
box-shadow: 0 0 0 2px #929292;
}

.add-form button {
background: #32a852;
color: white;
border: none;
border-radius: 8px;
padding: 12px 24px;
font-weight: 600;
cursor: pointer;
transition: background 0.2s;
margin-top: 10px;
}

.add-form button:hover {
background: #218c3a;
}

.add-form input::placeholder {
color: #fff;
opacity: 1;
}

/* Update menu items style */
.update-section {
min-height: 100vh;
display: flex;
align-items: center;
justify-content: center;
background-image:
url('../src/assets/abstract_background_image.jpg');
background-size: cover;
color: white;
}

.update-glass-container {
width: 420px;
font-family: 'Oswald', sans-serif;
padding: 36px 32px;
background: rgba(255,255,255,0.12);
border-radius: 16px;
box-shadow: 0 4px 16px rgba(0,0,0,0.12);
border: 1px solid #fff;
position: relative;
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
z-index: 1;
}

.update-glass-container::before {
content: "";
}

```

```
position: absolute;
width: 100%;
height: 100%;
border-radius: 16px;
backdrop-filter: blur(8px);
-webkit-backdrop-filter: blur(8px);
z-index: -1;
}

.update-glass-container h2 {
font-size: 2rem;
margin-bottom: 24px;
text-align: center;
}

.update-form {
display: flex;
flex-direction: column;
gap: 18px;
width: 100%;
align-items: center;
}

.update-form input,
.update-form select {
width: 90%;
background: rgba(255,255,255,0.18);
border: 1px solid #fff;
border-radius: 8px;
padding: 12px 16px;
color: white;
font-size: 1rem;
margin-bottom: 0;
}

.update-form input:focus,
.update-form select:focus {
outline: none;
box-shadow: 0 0 0 2px #929292;
}

.update-form button {
background: #32a852;
color: white;
border: none;
border-radius: 8px;
padding: 12px 24px;
font-weight: 600;
cursor: pointer;
transition: background 0.2s;
margin-top: 10px;
}

.update-form button:hover {
background: #218c3a;
}

.update-form input::placeholder {
color: #ffff;
opacity: 1;
}

.update-back-btn {
background-color: #fffffd;
border: none;
color: black;
border-radius: 6px;
padding: 10px 28px;
font-family: "Oswald", sans-serif;
font-size: 1.2rem;
font-weight: 600;
transition: all 0.3s ease;
cursor: pointer;
margin-top: 18px;
text-align: center;
text-decoration: none;
display: inline-block;
}

.update-back-btn:hover {
background-color: #929292;
color: white;
}
```

main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
<StrictMode>
  <App />
</StrictMode>,
)
```

App.jsx

```
import './index.css'; /* This single CSS file is used for the entire website. Styles are also provided inline in some cases */
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import ProtectedRoute from './pages/ProtectedRoute';
import Homepage from './pages/Homepage';
import CustomerLanding from './pages/CustomerLanding';
import AdminLanding from './pages/AdminLanding';
import Add from './pages/Add';
import Update from './pages/Update';

const App = () => {
  return (
    <BrowserRouter>
      <Routes>
        {/* Displays <Homepage /> if '/' is the path - it is the main starting project for this website for a user */}
        <Route path="/" element={<Homepage />} />
        {/* Allows login as customer and admin and allows admin to add and update menu_items table via two separate routes '/add' and '/update/:id' */}
        <Route path="/customer" element={
          <ProtectedRoute allowedRole="Customer">
            <CustomerLanding />
          </ProtectedRoute>
        } />
        <Route path="/admin" element={
          <ProtectedRoute allowedRole="Admin">
            <AdminLanding />
          </ProtectedRoute>
        } />
        <Route path="/add" element={
          <ProtectedRoute allowedRole="Admin">
            <Add />
          </ProtectedRoute>
        } />
        <Route path="/update/:id" element={
          <ProtectedRoute allowedRole="Admin">
            <Update />
          </ProtectedRoute>
        } />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Update.jsx

```
import axios from "axios";
import React, { useState } from "react";
import { Link, useLocation, useNavigate } from "react-router-dom";

const Update = () => {
  // Hook for navigation and getting the current location
  const navigate = useNavigate();
  const location = useLocation();

  // This is for getting the id of the item to be updated from the URL
  const itemId = location.pathname.split("/")[2];

  // State to hold the item details
  const [item, setItem] = useState({
    item_name: "",
    category: "",
    price: "",
    item_picture: ""
  });

  // Fetch the current item details when the component mounts
  React.useEffect(() => {
    axios.get(`http://localhost:8800/menu_items/${itemId}`)
      .then(res => setItem(res.data))
      .catch(err => console.log(err));
  }, [itemId]);

  // Handle input changes
  const handleChange = e => {
    setItem(prev => ({ ...prev, [e.target.name]: e.target.value }));
  };

  // Handle form submission to update the item
  const handleSubmit = async e => {
    e.preventDefault();
    try {
      await axios.put(`http://localhost:8800/menu_items/${itemId}`, item);
      navigate("/admin");
    } catch (err) {
      alert("Update failed");
    }
  };

  return (
    <section className="update-section">
      <div className="update-glass-container">
        <h2>Update Menu Item</h2>
        <form className="update-form" onSubmit={handleSubmit}>
          <input
            type="text"
            name="item_name"
            placeholder="Item Name"
            value={item.item_name}
            onChange={handleChange}
            required
          />
          <input
            type="text"
            name="category"
            placeholder="Category"
            value={item.category}
            onChange={handleChange}
            required
          />
          <input
            type="number"
            name="price"
            placeholder="Price"
            value={item.price}
            onChange={handleChange}
            required
          />
        </form>
      </div>
    </section>
  );
}
```

```

<input
type="text"
name="item_picture"
placeholder="Image URL"
value={item.item_picture}
onChange={handleChange}
/>
<button type="submit" className="font-changer">Update
Item</button>
</form>
<Link to="/admin" className="update-back-btn">Back</Link>
</div>
</section>
);
};

export default Update;

```

Reviews.jsx

```

import { useEffect, useState } from "react";
import axios from 'axios';

const Reviews = () => {
// State to hold reviews data
const [reviews, setReviews] = useState([]);

// Fetch reviews from the backend when the component mounts
useEffect(() => {
  const fetchReviews = async () => {
    try {
      const res = await axios.get("http://localhost:8800/reviews");
      setReviews(res.data);
    } catch (err) {
      console.log(err);
    }
  };
  fetchReviews();
}, []);

// Function to render star ratings
const renderStars = (rating) => {
  const rounded = Math.round(rating); // rounds 4.1 to 4, 4.7 to 5
  return Array.from({ length: 5 }, (_, i) =>
    <span key={i}>i < rounded ? '★' : '☆'</span>
  );
};

return (
  <section className="reviews-section">
    <h2>WHAT OTHERS ARE SAYING...</h2>
    <p>Reviews from our frequent visitors</p>
    <div className="reviews-container">
      {reviews.map((review) => (
        <div className="review-card" key={review.review_id}>
          <p className="review-card-description">{review.description}</p>
          <p className="review-author">- {review.username}</p>
          <p className="rating-value">{renderStars(review.rating)}</p>
        </div>
      )));
    </div>
  </section>
);
};

export default Reviews;

```

ProtectedRoute.jsx

```

import { Navigate } from 'react-router-dom';

const ProtectedRoute = ({ children, allowedRole }) => {
  const user = JSON.parse(localStorage.getItem('user'));
  if (!user) {
    return <Navigate to="/" replace />;
  }
  if (allowedRole && user.role !== allowedRole) {
    return <Navigate to="/" replace />;
  }
  return children;
};

export default ProtectedRoute;

```

MenuItems.jsx

```

import React from 'react'
import { useState, useEffect } from 'react'
import axios from 'axios'
import { Link, useNavigate } from 'react-router-dom'

const MenuItems = () => {
// State to hold menu items
const [menuitems, setMenuitems] = useState([]);

// Fetch menu items from the backend when the component mounts
useEffect(() => {
  const fetchMenuitems = async () => {
    try {
      const res = await axios.get("http://localhost:8800/menu_items");
      setMenuitems(res.data);
    } catch (err) {
      console.log(err);
    }
  };
  fetchMenuitems();
}, []);

// Function to handle deletion of a menu item
const handleClick = async (id) => {
  try {
    await axios.delete("http://localhost:8800/menu_items/" + id);
    // To refresh the page after deletion
    window.location.reload();
  } catch (err) {
    console.log(err);
  }
}

return (
  <div>
    <h1 style={{textAlign: 'center', marginTop: "15px", fontSize: "2rem"}>Menu Items</h1>
    <div className="admin-menu-items-container">
      {menuitems.map((item) => (
        <div className="admin-menu-item-card" key={item.item_id}>
          {item.item_picture && (
            <img src={item.item_picture} alt={'Picture of menu item ${item.item_name}'} />
          )}
          <h3>{item.item_name}</h3>
          <p>ID: {item.item_id}</p>
          <p>Category: {item.category}</p>
          <p>Price: ₹{item.price}</p>
          <div className="admin-menu-item-actions">
            <button className='delete-button' onClick={() => handleClick(item.item_id)}>Delete</button>

```

```

    <button className='update-button'><Link to=
   ={`/update/${item.item_id}`}>Update</Link></button>
  </div>
</div>
)}
</div>
<div style={{textAlign: 'center', marginTop: '2rem'}}>
  <button><Link to="/add">Add New Item</Link></button>
</div>
</div>
)
</div>
}

export default MenuItems

```

Login.jsx

```

import { useState } from 'react';
import { useNavigate } from 'react-router-dom';

const Login = () => {
  // State variables for form inputs and error message
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  // Handle login form submission
  const handleSubmit = async (e) => {
    e.preventDefault();
    setError("");
    try {
      const res = await fetch('http://localhost:8800/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username, password })
      });
      const data = await res.json();
      if (!res.ok) {
        setError(data.error || 'Login failed');
        return;
      }
      localStorage.setItem('user', JSON.stringify(data));
      if (data.role === 'Admin') {
        navigate('/admin');
      } else {
        navigate('/customer');
      }
    } catch (err) {
      setError('Server error');
    }
  };
}

return (
  <section className="login-section" id="login-section">
    <div className="glass-container">
      <div className="login-box">
        <h2>Login</h2>
        <p>Contact admin for new registration!</p>
        <form onSubmit={handleSubmit}>
          <input
            type="text"
            id="username"
            name="username"
            required
            placeholder="Username"
            value={username}
            onChange={e => setUsername(e.target.value)}
          />

```

```

<input
  type="password"
  id="password"
  name="password"
  required
  placeholder="Password"
  value={password}
  onChange={e => setPassword(e.target.value)}
/>
<button type="submit">Login</button>
</form>
/* Display error message in red in case of error */
{error && <p style={{ color: 'red', marginTop: '10px' }}>{error}</p>}
</div>
</div>
</section>
);
}

```

export default Login;

Homepage.jsx

```

import Hero from './Hero';
import Login from './Login';
import Reviews from './Reviews';
import Credits from './Credits';

const Homepage = () => {
  return (
    // Using React fragments (<></>) to avoid unnecessary divs and
    // passing multiple elements
    <>
      /* Hero, Login, Reviews, Credits are separately written in their
       respective .jsx files. Clean horizontal rule (HR) is added between
       these for aesthetics */
      <Hero />
      <hr style={{ height: '1px', border: '0 solid gray' }} />
      <Login />
      <hr style={{ height: '1px', border: '0 solid gray' }} />
      <Reviews />
      <hr style={{ height: '1px', border: '0 solid gray' }} />
      <Credits />
      <hr style={{ height: '1px', border: '0 solid gray' }} />
    </>
  );
}

export default Homepage;

```

Hero.jsx

```

import cloche_icon from "../assets/cloche_icon.png"; // Import the
// cloche or dish icon image from assets for logo

const Hero = () => {
  return (
    <section className="homepage page-background">
      <div className="navbar">
        <div className="logo-div">
          <img src={cloche_icon} alt="EatWise Logo" className="logo-img"/>
          <h1>EatWise</h1>
        </div>
        <a href="#login-section">
          <button type="button">Login</button>
        </a>
      </div>
    </section>
  );
}

export default Hero;

```

```

<div className="hero-text">
  <h1>Welcome to EatWise!</h1>
  <p>Efficient Restaurant Management System</p>
</div>
</section>
)
}

export default Hero

// Refresh order items
fetch(`http://localhost:8800/order_items?user_id=${user.user_id}`)
.then(res => res.json())
.then(data => setOrderItems(data));
})
.catch(err => {
alert('Network error');
console.error('Network error:', err);
});
};

return (
<section className="customer-landing-section">
<div className="customer-landing-section-header">
<h2>Welcome, {user?.username}!</h2>
<button onClick={handleLogout} style={{marginTop: '20px'}}>Logout</button>
</div>

<div className="menu-items-container">
{menuItems.map(item => (
<div className="menu-item-card" key={item.item_id}>
{item.item_picture}
? <img src={item.item_picture} alt={item.item_name} />
: null)
<h3>{item.item_name}</h3>
<p>Category: {item.category}</p>
<p>Price: ₹{item.price}</p>
<div className="menu-item-actions">
<label htmlFor={`quantity-${item.item_id}`}>Qty:</label>
<input
  type="number"
  min="1"
  value={quantities[item.item_id] || 1}
  onChange={(e => handleQuantityChange(item.item_id, e.target.value))}
  style={{ width: '60px', borderRadius: '6px', border: '1px solid #ccc', padding: '4px' }}
  id={`quantity-${item.item_id}`}
/>
<button onClick={() => handleOrder(item)}>Order</button>
</div>
</div>
))
</div>

<div className="order-items-container">
<h2>Your Orders</h2>
<table className="order-items-table">
<thead>
<tr>
<th>Order Item ID</th>
<th>Item Name</th>
<th>Quantity</th>
<th>Suborder Price</th>
</tr>
</thead>
<tbody>
{orderItems.map(order => (
<tr key={order.order_items_id}>
<td>{order.order_items_id}</td>
<td>{order.item_name}</td>
<td>{order.quantity}</td>
<td>₹{order.suborder_price}</td>
<td>
<button
  style={{ background: '#a83232', color: 'white', border: 'none', borderRadius: '6px', padding: '6px 12px', cursor: 'pointer' }}
  onClick={() => {
    fetch(`http://localhost:8800/order_items/${order.order_items_id}`, {
      method: 'DELETE'
    })
    .then(res => res.json())
    .then(data => {
      if (data.success) {
        setOrderItems(data);
      }
    })
  }}
/>
</td>
))
</tbody>
</table>

```

CustomerLanding.jsx

```

import { useNavigate } from 'react-router-dom';
import { useEffect, useState } from 'react';

const CustomerLanding = () => {
  // Navigation hook to redirect users to different routes
  const navigate = useNavigate();
  // State variables to hold menu items, order items, and quantities
  const [menuItems, setMenuItems] = useState([]);
  const [orderItems, setOrderItems] = useState([]);
  const [quantities, setQuantities] = useState({});
  // To store logged-in user details from localStorage
  const user = JSON.parse(localStorage.getItem('user'));

  // During logout, remove user from localStorage and navigate to '/' or
  <Homepage />
  const handleLogout = () => {
    localStorage.removeItem('user');
    navigate('/');
  };

  // Retrieve menu items from backend and store in state variable
  useEffect(() => {
    fetch('http://localhost:8800/menu_items')
      .then(res => res.json())
      .then(data => setMenuItems(data));
  }, []);

  // Retrieve order items for the logged-in user and store in state
  // variable
  useEffect(() => {
    if (user) {
      fetch(`http://localhost:8800/order_items?user_id=${user.user_id}`)
        .then(res => res.json())
        .then(data => setOrderItems(data));
    }
  }, [user]);

  // Handle quantity change
  const handleQuantityChange = (itemId, value) => {
    setQuantities(q => ({ ...q, [itemId]: value }));
  };

  // Handle order button
  const handleOrder = (item) => {
    const quantity = parseInt(quantities[item.item_id] || 1);
    if (!user) return;
    fetch('http://localhost:8800/order_items', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        user_id: user.user_id,
        item_id: item.item_id,
        quantity,
        price: item.price
      })
    })
    .then(async res => {
      const data = await res.json();
      if (!res.ok || !data.success) {
        alert('Order failed: ' + (data.error || 'Unknown error'));
        console.error('Order error:', data);
        return;
      }
    })
  };
}

export default Hero

```

```

// Refresh order items
fetch('http://localhost:8800/order_items?user_id=${user.user_id}`)
.then(res => res.json())
.then(data => setOrderItems(data));
} else {
  alert('Delete failed');
}
});
})
>Delete</button>
</td>
</tr>
))}
</tbody>
</table>
</div>
</section>
);
};

export default CustomerLanding;

```

Credits.jsx

```

const Credits = () => {
  return (
    <section className='credits-section'>
      <p>DBMS project by</p>
      <p>Aravind A Kamath</p>
      <p>Anandhakrishnan S</p>
      <p>Aravind Sajeev</p>
      <p>Arjun Manoj</p>
      <p>S5 CS-A</p>
      <p>25 September 2025</p>
    </section>
  )
}

export default Credits

```

AdminLanding.jsx

```

import { useNavigate } from "react-router-dom";
import { useEffect, useState } from "react";
import MenuItems from "./MenuItems";
import Credits from "./Credits";

const AdminLanding = () => {
  // Navigation hook
  const navigate = useNavigate();

  // Logout function for removing user from local storage and
  // redirecting to home
  const handleLogout = () => {
    localStorage.removeItem("user");
    navigate("/");
  };

  // Get user from local storage
  const user = JSON.parse(localStorage.getItem("user"));

  // Reviews state
  const [reviews, setReviews] = useState([]);
  useEffect(() => {
    fetch("http://localhost:8800/reviews")
      .then((res) => res.json())
      .then((data) => setReviews(data));
  }, []);

  // Delete review function
  const handleDeleteReview = (id) => {
    fetch(`http://localhost:8800/reviews/${id}`, { method: "DELETE" })
      .then((res) => res.json())
      .then((data) => {
        if (data.success) {
          setReviews((reviews) => reviews.filter((r) => r.review_id !== id));
        }
      });
  };
}

export default AdminLanding;

```

```

} else {
  alert("Delete failed");
}
});

// Users state
const [users, setUsers] = useState([]);
useEffect(() => {
  fetch("http://localhost:8800/users")
    .then((res) => res.json())
    .then((data) => setUsers(data));
}, []);

// Delete user function
const handleDeleteUser = (id) => {
  fetch(`http://localhost:8800/users/${id}`, { method: "DELETE" })
    .then((res) => res.json())
    .then((data) => {
      if (data.success) {
        setUsers((users) => users.filter((u) => u.user_id !== id));
      } else {
        alert("Delete failed");
      }
    });
};

// Orders Management state
const [orders, setOrders] = useState([]);
useEffect(() => {
  fetch("http://localhost:8800/orders")
    .then((res) => res.json())
    .then((data) => setOrders(data));
}, []);

// Delete order function
const handleDeleteOrder = (id) => {
  fetch(`http://localhost:8800/orders/${id}`, { method: "DELETE" })
    .then((res) => res.json())
    .then((data) => {
      if (data.success) {
        setOrders((orders) => orders.filter((o) => o.order_id !== id));
      } else {
        alert("Delete failed");
      }
    });
};

// Order Items Management state
const [orderItems, setOrderItems] = useState([]);
useEffect(() => {
  fetch("http://localhost:8800/order_items?all=true")
    .then((res) => res.json())
    .then((data) => setOrderItems(data));
}, []);

// Delete order item function
const handleDeleteOrderItem = (id) => {
  fetch(`http://localhost:8800/order_items/${id}`, { method: "DELETE" })
    .then((res) => res.json())
    .then((data) => {
      if (data.success) {
        setOrderItems((items) => {
          items.filter((oi) => oi.order_items_id !== id);
        });
      } else {
        alert("Delete failed");
      }
    });
};

return (
  <>
  <section className="admin-landing-section">
    <div className="admin-landing-section-header">
      <h2>Welcome, {user?.username!}</h2>
      <button onClick={handleLogout}>Logout</button>
    </div>
  </section>
)
}

```

```

<hr
style={{ height: "1px", borderWidth: "0", backgroundColor: "gray" }}
/>
<div className="admin-landing-section-content">
<div className="admin-landing-menu-items" style={{marginBottom: '20px'}}>
<MenuItems />
</div>
<hr
style={{ height: "1px", borderWidth: "0", backgroundColor: "gray" }}
/>
{/* Reviews Management Section */}
<div style={{ marginTop: "10px", padding: "0 30px" }}>
<h2 className="text-aligner" style={{ fontSize: "2rem" }}>Reviews Management</h2>
<table
style={{
width: "100%",
color: "white",
background: "rgba(255,255,255,0.08)",
borderRadius: "10px",
marginTop: "20px",
}}
>
<thead>
<tr>
<th className="text-aligner">ID</th>
<th className="text-aligner">User</th>
<th className="text-aligner">Description</th>
<th className="text-aligner">Date</th>
<th className="text-aligner">Rating</th>
<th className="text-aligner">Delete</th>
</tr>
</thead>
<tbody>
{reviews.map((r) => (
<tr key={r.review_id}>
<td className="text-aligner">{r.review_id}</td>
<td className="text-aligner">{r.username}</td>
<td className="text-aligner">{r.description}</td>
<td className="text-aligner">{r.date}</td>
<td className="text-aligner">{r.rating}</td>
<td className="text-aligner">
<button
onClick={() => handleDeleteReview(r.review_id)}
style={{
background: "#a83232",
color: "white",
border: "none",
borderRadius: "6px",
padding: "6px 12px",
cursor: "pointer",
}}
>
Delete
</button>
</td>
</tr>
))}
</tbody>
</table>
/* Add Review Form */
<form
className="admin-form"
onSubmit={async (e) => {
e.preventDefault();
const form = e.target;
const body = {
user_id: form.user_id.value,
description: form.description.value,
date: form.date.value,
rating: form.rating.value,
};
const res = await fetch("http://localhost:8800/reviews", {
method: "POST",
headers: { "Content-Type": "application/json" },
body: JSON.stringify(body),
});
const data = await res.json();
if (data.success) {
form.reset();
fetch("http://localhost:8800/reviews")
.then((res) => res.json())
.then((data) => setReviews(data));
} else {
alert(data.error ? `Add failed: ${data.error}` : "Add failed");
}
}}
>
<input
name="user_id"
type="number"
placeholder="User ID"
required
/>
<input
name="description"
type="text"
placeholder="Description"
required
/>
<input name="date" type="date" required />
<input
name="rating"
type="number"
min="1"
max="5"
placeholder="Rating"
required
/>
<button type="submit" className="font-changer">Add Review</button>
</form>
</div>
<hr
style={{ height: "1px", borderWidth: "0", backgroundColor: "gray" }}
/>
/* Orders Management Section */
<div style={{ marginTop: "15px", marginBottom: "20px", padding: "0 30px" }}>
<h2 className="text-aligner" style={{ fontSize: "2rem" }}>Orders Management</h2>
<table
style={{
width: "100%",
color: "white",
background: "rgba(255,255,255,0.08)",
borderRadius: "10px",
marginTop: "20px",
}}
>
<thead>
<tr>
<th className="text-aligner">Order ID</th>
<th className="text-aligner">User ID</th>
<th className="text-aligner">Order Date Time</th>
<th className="text-aligner">Total Amount</th>
<th className="text-aligner">Delete</th>
</tr>
</thead>
<tbody>
{orders.map((o) => (
<tr key={o.order_id}>
<td className="text-aligner">{o.order_id}</td>
<td className="text-aligner">{o.user_id}</td>
<td className="text-aligner">{o.order_date_time}</td>
<td className="text-aligner">{o.total_amount}</td>
<td className="text-aligner">
<button
onClick={() => handleDeleteOrder(o.order_id)}
style={{
background: "#a83232",
color: "white",
border: "none",
borderRadius: "6px",
padding: "6px 12px",
cursor: "pointer",
}}
>
Delete
</button>
</td>
</tr>
))
)}
</tbody>
</table>

```

```

    >
  Delete
</button>
</td>
</tr>
)})
</tbody>
</table>
</div>
<hr
style={{ height: "1px", borderWidth: "0", backgroundColor: "gray" }}>
/* Order Items Management Section */
<div style={{ marginTop: "15px", marginBottom: "20px", padding: "0px 30px" }}>
  <h2 className="text-aligner" style={{ fontSize: "2rem" }}>Order
  Items Management</h2>
  <table
  style={{ width: "100%", color: "white", background: "rgba(255,255,255,0.08)", borderRadius: "10px", marginTop: "20px", }}>
    <thead>
      <tr>
        <th className="text-aligner">Order Item ID</th>
        <th className="text-aligner">Order ID</th>
        <th className="text-aligner">Item ID</th>
        <th className="text-aligner">Quantity</th>
        <th className="text-aligner">Suborder Price</th>
        <th className="text-aligner">Delete</th>
      </tr>
    </thead>
    <tbody>
      {orderItems.map((oi) => (
        <tr key={oi.order_items_id}>
          <td className="text-aligner">{oi.order_items_id}</td>
          <td className="text-aligner">{oi.order_id}</td>
          <td className="text-aligner">{oi.item_id}</td>
          <td className="text-aligner">{oi.quantity}</td>
          <td className="text-aligner">{oi.suborder_price}</td>
          <td className="text-aligner">
            <button
              onClick={() => handleDeleteOrderItem(oi.order_items_id)}
              style={{ background: "#a83232", color: "white", border: "none", borderRadius: "6px", padding: "6px 12px", cursor: "pointer", }}>
              Delete
            </button>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
</div>

<hr
style={{ height: "1px", borderWidth: "0", backgroundColor: "gray" }}>
/* Users Management Section */
<div style={{ marginTop: "15px", padding: "0 30px" }}>
  <h2 className="text-aligner" style={{ fontSize: "2rem" }}>Users
  Management</h2>
  <table
  style={{ width: "100%", color: "white", background: "rgba(255,255,255,0.08)", borderRadius: "10px", }}>
    <thead>
      <tr>
        <th>ID</th>
        <th>Username</th>
        <th>Password</th>
        <th>Email</th>
        <th>Role</th>
        <th>Delete</th>
      </tr>
    </thead>
    <tbody>
      {users.map((u) => (
        <tr key={u.user_id}>
          <td>{u.user_id}</td>
          <td>{u.username}</td>
          <td>{u.password}</td>
          <td>{u.email || "-"}</td>
          <td>{u.role}</td>
          <td>
            <button
              onClick={() => handleDeleteUser(u.user_id)}
              style={{ background: "#a83232", color: "white", border: "none", borderRadius: "6px", padding: "6px 12px", cursor: "pointer", }}>
              Delete
            </button>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
  /* Add User Form */
  <form
    className="admin-form"
    onSubmit={async (e) => {
      e.preventDefault();
      const form = e.target;
      const body = {
        username: form.username.value,
        password: form.password.value,
        email: form.email.value,
        role: form.role.value,
      };
      const res = await fetch("http://localhost:8800/users", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(body),
      });
      const data = await res.json();
      if (data.success) {
        form.reset();
        fetch("http://localhost:8800/users")
          .then((res) => res.json())
          .then((data) => setUsers(data));
      } else {
        alert(data.error ? `Add failed: ${data.error}` : "Add failed");
      }
    }}
  >
    <input
      name="username"
      type="text"
      placeholder="Username"
      required
    />
    <input
      name="password"
      type="text"
      placeholder="Password"
      required
    />

```

```

<input name="email" type="email" placeholder="Email" />
<select
name="role"
required
style={{
background: "rgba(30,41,59,0.15)",
color: "white",
border: "1px solid #ccc",
borderRadius: "8px",
padding: "8px 12px",
fontSize: "1rem",
}}
>
<option value="" disabled>
Role
</option>
<option value="Admin">Admin</option>
<option value="Customer">Customer</option>
</select>
<button type="submit" className="font-changer">Add
User</button>
</form>
</div>
</div>
</section>
<hr
style={{ height: "1px", borderWidth: "0", backgroundColor: "gray" }}>
</>
<Credits />
</>
);
};

export default AdminLanding;

```

Add.jsx

```

import { useState } from 'react'; // Importing useState hook from React
import { useNavigate } from 'react-router-dom'; // Importing
useNavigate hook from react-router-dom for navigation
import axios from 'axios'; // Importing axios for making HTTP requests

const Add = () => {
// State to hold the new menu item details
const [menulitem, setMenulitem] = useState({
item_id: '',
item_picture: '',
item_name: '',
category: '',
price: null,
});

// Hook for navigation
const navigate = useNavigate();

// Function to handle input changes and update the state
const handleChange = (e) => {
setMenulitem((prev) => ({ ...prev, [e.target.name]: e.target.value }));
};

// Function to send updated data to backend server
const handleClick = async (e) => {
e.preventDefault(); // To prevent refreshing of the page when
clicking the button
try {
await axios.post("http://localhost:8800/menu_items", menulitem);
navigate("/admin");
} catch (err) {
console.log(err);
}
}

return (
<section className="add-section">
<div className="add-glass-container">
<h2>Add New Item</h2>
<form className="add-form">
<input type="text" placeholder="Item ID" onChange=
{handleChange} name='item_id' required />

```

index.js (backend)

```
import express from 'express';
import mysql from 'mysql2';
import cors from 'cors';
const app = express();

// Code to connect to MySQL database
// If there is an authentication problem, ALTER USER 'root'@'localhost'
// IDENTIFIED WITH mysql_native_password BY 'your_password';
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'Aravind2005',
  database: 'restaurant_schema'
});

// Without middleware, you cannot send json in the body of a post
// request
app.use(express.json());
// Without CORS middleware, you cannot make requests from
// frontend to backend. You can also specify domain names here like
// localhost:3000
app.use(cors());

// Basic route to check if backend is running
app.get('/', (req, res) => {
  res.json("Hello this is the backend!");
});

// Login route
app.post('/login', (req, res) => {
  const { username, password } = req.body;
  const q = "SELECT * FROM users WHERE username = ? AND
password = ?";
  db.query(q, [username, password], (err, results) => {
    if (err) return res.status(500).json({ error: 'Database error' });
    if (results.length === 0) {
      return res.status(401).json({ error: 'Invalid credentials' });
    }
    // Return role and user info
    const user = results[0];
    return res.json({ role: user.role, user_id: user.user_id, username:
user.username });
  });
});

// Retrieve all users
app.get('/users', (req, res) => {
  const q = "SELECT * FROM users";
  db.query(q, (err, data) => {
    if (err) return res.status(500).json({ error: err });
    return res.json(data);
  });
});

// Delete user by id
app.delete('/users/:id', (req, res) => {
  const user_id = req.params.id;
  const q = "DELETE FROM users WHERE user_id = ?";
  db.query(q, [user_id], (err, data) => {
    if (err) return res.status(500).json({ error: err });
    return res.json({ success: true });
  });
});

// Admin's menu_items management routes
// Retrieve all menu items
app.get('/menu_items', (req, res) => {
  const q = "SELECT * FROM menu_items";
  db.query(q, (err, data) => {
    if (err) return res.json(err);
    return res.json(data);
  });
});

// Create new menu item
app.post('/menu_items', (req, res) => {
  const q = "INSERT INTO menu_items ('item_id', 'item_name',
'category', 'price', 'item_picture') VALUES (?)";
  const values = [req.body.item_id, req.body.item_name,
req.body.category, req.body.price, req.body.item_picture];
  db.query(q, [values], (err, data) => {
    if (err) return res.json(err);
    return res.json("Menu item created successfully!");
  });
});

// Delete menu item by id
app.delete('/menu_items/:id', (req, res) => {
  const menuitemId = req.params.id;
  const q = "DELETE FROM menu_items WHERE item_id = ?";
  db.query(q, [menuitemId], (err, data) => {
    if (err) return res.json(err);
    return res.json("Menu item deleted successfully!");
  });
});

// Update menu item by id
app.put('/menu_items/:id', (req, res) => {
  const menuitemId = req.params.id;
  const q = "UPDATE menu_items SET `item_name` = ?, `category` =
?, `price` = ?, `item_picture` = ? WHERE item_id = ?";
  const values = [
    req.body.item_name,
    req.body.category,
    req.body.price,
    req.body.item_picture
  ];
  db.query(q, [...values, menuitemId], (err, data) => {
    if (err) return res.json(err);
    return res.json("Menu item updated successfully!");
  });
});

// Customer/visitor/guest reviews routes
// Rerieve all reviews (for admin)
app.get('/reviews', (req, res) => {
  const q = "SELECT reviews.*, users.username FROM reviews JOIN
users ON reviews.user_id = users.user_id";
  db.query(q, (err, data) => {
    if (err) return res.status(500).json({ error: err });
    return res.json(data);
  });
});

// Delete review by id
app.delete('/reviews/:id', (req, res) => {
  const review_id = req.params.id;
  const q = "DELETE FROM reviews WHERE review_id = ?";
  db.query(q, [review_id], (err, data) => {
    if (err) return res.status(500).json({ error: err });
    return res.json({ success: true });
  });
});

// Get a single menu item by id
app.get('/menu_items/:id', (req, res) => {
  const menuitemId = req.params.id;
  const q = "SELECT * FROM menu_items WHERE item_id = ?";
  db.query(q, [menuitemId], (err, data) => {
    if (err) return res.status(500).json({ error: err });
    if (data.length === 0) return res.status(404).json({ error: "Menu item
not found" });
    return res.json(data[0]);
  });
});

// Add new review
app.post('/reviews', (req, res) => {
  const { user_id, description, date, rating } = req.body;
  if (!user_id || !description || !date || !rating) {
    return res.status(400).json({ error: 'Missing required fields' });
  }
});
```

```

    });

    const q = "INSERT INTO reviews (user_id, description, date, rating)
VALUES (?, ?, ?, ?)";
    db.query(q, [user_id, description, date, rating], (err, result) => {
      if (err) return res.status(500).json({ error: err });
      if (result && result.affectedRows > 0) {
        return res.json({ success: true });
      } else {
        return res.status(500).json({ error: 'Insert failed' });
      }
    });
  });

  // Add new user
  app.post('/users', (req, res) => {
    const { username, password, email, role } = req.body;
    if (!username || !password || !role) {
      return res.status(400).json({ error: 'Missing required fields' });
    }
    // Check for duplicate username
    const checkQ = "SELECT * FROM users WHERE username = ?";
    db.query(checkQ, [username], (err, results) => {
      if (err) return res.status(500).json({ error: err });
      if (results.length > 0) {
        return res.status(409).json({ error: 'Username already exists' });
      }
      const q = "INSERT INTO users (username, password, email, role)
VALUES (?, ?, ?, ?)";
      db.query(q, [username, password, email, role], (err, result) => {
        if (err) return res.status(500).json({ error: err });
        if (result && result.affectedRows > 0) {
          return res.json({ success: true });
        } else {
          return res.status(500).json({ error: 'Insert failed' });
        }
      });
    });
  });

  // Add order item (creates order if needed)
  app.post('/order_items', (req, res) => {
    const { user_id, item_id, quantity, price } = req.body;
    if (!user_id || !item_id || !quantity || !price) {
      return res.status(400).json({ error: 'Missing required fields' });
    }
    // Find or create an order for today
    const now = new Date();
    const order_date_time = now.toISOString().slice(0, 19).replace('T', ' ');
    const total_amount = price * quantity;
    // Check if an order exists for this user today
    const findOrderQ = "SELECT order_id FROM orders WHERE user_id
= ? AND DATE(order_date_time) = CURDATE()";
    db.query(findOrderQ, [user_id], (err, orders) => {
      if (err) return res.status(500).json({ error: err });
      let order_id;
      if (orders.length > 0) {
        order_id = orders[0].order_id;
        insertOrderItem(order_id);
      } else {
        // Create new order
        const createOrderQ = "INSERT INTO orders (user_id,
order_date_time, total_amount) VALUES (?, ?, ?)";
        db.query(createOrderQ, [user_id, order_date_time, total_amount],
(err, result) => {
          if (err) return res.status(500).json({ error: err });
          order_id = result.insertId;
          insertOrderItem(order_id);
        });
      }
      function insertOrderItem(order_id) {
        const suborder_price = price * quantity;
        const q = "INSERT INTO order_items (order_id, item_id, quantity,
suborder_price) VALUES (?, ?, ?, ?)";
        db.query(q, [order_id, item_id, quantity, suborder_price], (err,
data) => {
          if (err) return res.status(500).json({ error: err });
          return res.json({ success: true });
        });
      }
    });
  });

  // Delete order item
  app.delete('/order_items/:id', (req, res) => {
    const order_items_id = req.params.id;
    const q = "DELETE FROM order_items WHERE order_items_id = ?";
    db.query(q, [order_items_id], (err, data) => {
      if (err) return res.status(500).json({ error: err });
      return res.json({ success: true });
    });
  });

  // Retrieve all orders (for admin)
  app.get('/orders', (req, res) => {
    const q = "SELECT * FROM orders";
    db.query(q, (err, data) => {
      if (err) return res.status(500).json({ error: err });
      return res.json(data);
    });
  });

  // Delete order by id
  app.delete('/orders/:id', (req, res) => {
    const order_id = req.params.id;
    const q = "DELETE FROM orders WHERE order_id = ?";
    db.query(q, [order_id], (err, data) => {
      if (err) return res.status(500).json({ error: err });
      return res.json({ success: true });
    });
  });

  // Retrieve all order_items (for admin) or for a user
  app.get('/order_items', (req, res) => {
    if (req.query.all === 'true') {
      const q = "SELECT order_items_id, order_id, item_id, quantity,
suborder_price FROM order_items";
      db.query(q, (err, data) => {
        if (err) return res.status(500).json({ error: err });
        return res.json(data);
      });
    } else if (req.query.user_id) {
      const user_id = req.query.user_id;
      const q = `SELECT oi.order_items_id, oi.order_id, oi.item_id,
oi.quantity, oi.suborder_price, mi.item_name
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN menu_items mi ON oi.item_id = mi.item_id
WHERE o.user_id = ?`;
      db.query(q, [user_id], (err, data) => {
        if (err) return res.status(500).json({ error: err });
        return res.json(data);
      });
    } else {
      return res.status(400).json({ error: 'Missing query parameter' });
    }
  });

  // Start the server
  app.listen(8800, () => { console.log('Connected to backend!'); });
}

```

4. HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS

- CPU: Dual-core (2 logical cores) 1.6 GHz or better
- RAM: 4 GB (2 GB free after OS and background processes)
- Disk: 2 GB free (project + node_modules + Database storage); SSD recommended but not required
- Network: Local network or loopback (localhost) access; no special bandwidth required

SOFTWARE REQUIREMENTS

- Operating System: Windows 10/11, macOS 10.15+, or any modern Linux distribution
- Database Software: MySQL (MySQL Community Server 8.x recommended)
- NodeJS v16+ and npm
- Modern Web Browser (Google Chrome or Mozilla Firefox recommended)

5. CONCLUSION

The world has become a place where there is a lot of technological development, where every single thing done physically has been transformed into a digitized or computerized form. This project is in the restaurant/food/hotel business domain.

Our website takes this approach to the next level by providing a lot of features to both customers and administrators to manage their restaurant system efficiently. Orders, Reviews, Menu, Suborder, Users - all this information is centralized for maximum convenience and efficiency. This website eliminates the need for manual labour of fetching orders and updating them. Everyone in the restaurant - whether it is waiter, cashier, customer can modify and interact with this website, creating a hassle-free experience for everyone.

In the customer side, customer has a landing page after successful login which allows customer to see the menu items, create order and view suborders. For administrator side, all the relations can be viewed and modified

6. REFERENCES

1. Learning ReactJS, ExpressJS, NodeJS, MySQL - YouTube
(<https://www.youtube.com>)
2. Glitches, bug fixes and doubts - Google (<https://www.google.com>)