PROJECT TITLE:
Restaurant Management System

STUDENT NAME: Madhu Sri Sushmitha Chowdary Nekkanti

STUDENT BITS ID: 2024MT12059

Version Number	Date	Author/Owner	Description of Change
1	18/08/2024	Madhu Sri	Problem Statement and Requirement Specification
2	1/09/2024	Madhu Sri	ER Diagram and Object Model
3	15/09/2024	Madhu Sri	Relational model Mapping from ER/EER to Relational model
4	29/09/2024	Madhu Sri	Implementation of Relational model SQL CODE
5	12/10/2024	Madhu Sri	Application Code which accesses this DB

1. REQUIREMENT SPECIFICATION

a. Problem Statement & Requirements Definition:

RESTAURANT MANAGEMENT SYSTEM

In today's fast-paced culinary landscape, restaurant operations are increasingly complex, requiring efficient coordination between inventory management, customer service, employee scheduling, and financial oversight.

Many restaurants, especially smaller or family-owned establishments, continue to rely on outdated methods such as paper records, manual calculations, and disjointed software systems. These inefficiencies lead to common problems such as overstocking or understocking ingredients, inconsistent service quality, delayed order processing, and difficulty in tracking financial performance.

The lack of an integrated system exacerbates issues like food wastage, customer dissatisfaction due to long wait times, and errors in billing, all of which directly impact the restaurant's bottom line. Additionally, managers often struggle with employee scheduling and payroll management, leading to staffing issues that further hinder smooth operations. The absence of real-time data also limits a restaurant's ability to respond to changing customer preferences and market trends.

Requirements Definition

The Restaurant Management System (RMS) is designed to address these challenges by providing an innovative, all-in-one solution that streamlines every aspect of restaurant operations. This system will replace manual processes with automated workflows, ensuring greater accuracy, efficiency, and real-time visibility into all critical areas of the business.

Key Features and Functionalities:

1. **Integrated Inventory Management:**
    - **Real-time tracking of ingredient levels, automatic reordering based on predefined thresholds, and notifications for expiring stock.**
    - **Predictive analytics to forecast demand based on historical sales data, seasonal trends, and current promotions.**
2. **Order Processing and Table Management:**
    - **A digital order system that syncs with kitchen displays, reducing wait times and minimizing errors.**
    - **Reservation and table management features that optimize seating arrangements and reduce customer wait times.**

3. **Customer Relationship Management (CRM):**
    - **A loyalty program that tracks customer preferences and purchase history, enabling personalized marketing and promotions.**
    - **Feedback collection and analysis to improve customer satisfaction and service quality.**
4. **Employee Scheduling and Payroll:**
    - **Automated scheduling that considers employee availability, labor laws, and peak hours to optimize staffing levels.**
    - **Payroll integration with time-tracking features to ensure accurate compensation and reduce administrative burden.**
5. **Financial Management and Reporting:**
    - **Real-time tracking of sales, expenses, and profit margins, with customizable dashboards for easy monitoring.**
    - **Comprehensive reporting capabilities that provide insights into daily operations, financial performance, and cost-saving opportunities.**
6. **Web Access:**
    - **A friendly interface for managers and staff to access the system on-the-go, allowing for remote monitoring and management.**
    - **Customer-facing features like online ordering, reservations, and loyalty program management via a dedicated website.**
7. **Compliance and Security:**
    - **Built-in compliance checks for health regulations, food safety standards, and labor laws, ensuring the restaurant operates within legal requirements.**
    - **Advanced security protocols to protect sensitive data, including customer information, financial records, and employee details.**

By implementing the RMS, restaurants will experience a significant reduction in operational inefficiencies, leading to improved service quality, better inventory control, and enhanced financial oversight. The system's real-time data capabilities will empower restaurant managers to make informed decisions quickly, adapt to market changes, and ultimately drive profitability. Moreover, the automation of routine tasks will free up staff time, allowing them to focus on delivering exceptional customer experiences.

This innovative approach to restaurant management will not only solve existing problems but also position restaurants to thrive in an increasingly competitive market. The RMS aims to transform the way restaurants operate, creating a seamless, efficient, and customer-centric environment that drives success.

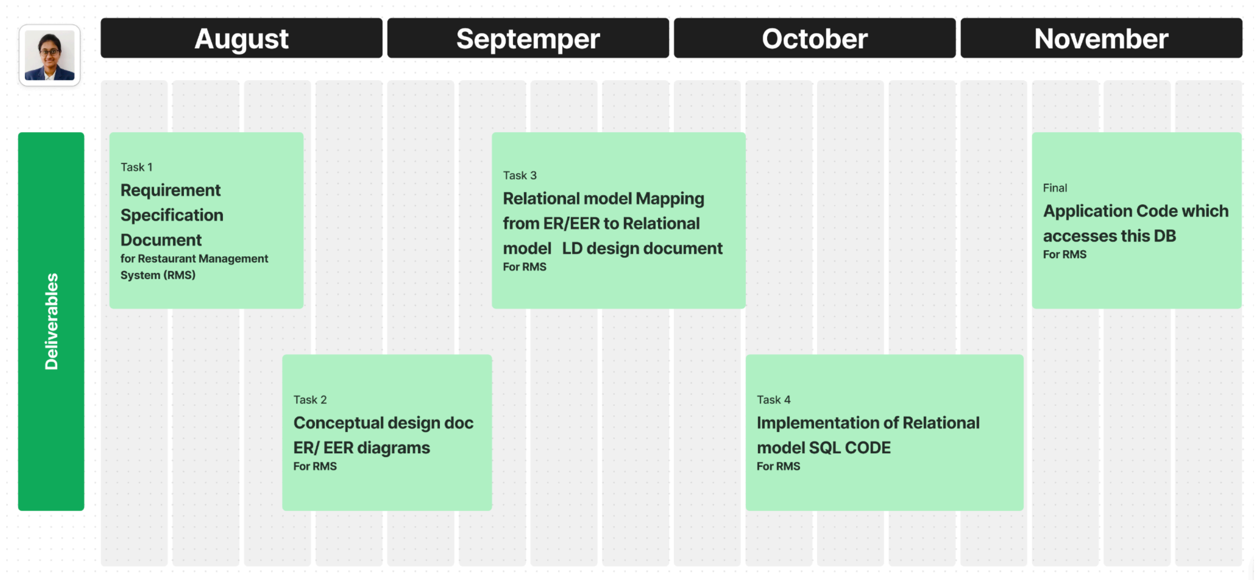
b. Project features identified

Feature ID	Feature Name	About the feature
1	Login	Authentication and success takes you to your respective dashboard based on the privileges.
2	Admin Dashboard	Admin has high privileges, This will give the admin the ability to manage inventory, employees, customers, and view payments.
3	Customer Dashboard	For customers to book tables, view the menu, and place orders.
4	Chef Dashboard	For chef to view and prepare orders.
5	Employee Dashboard	For staff to log in, take orders, and send notifications to the chef.

c. Software and hardware details

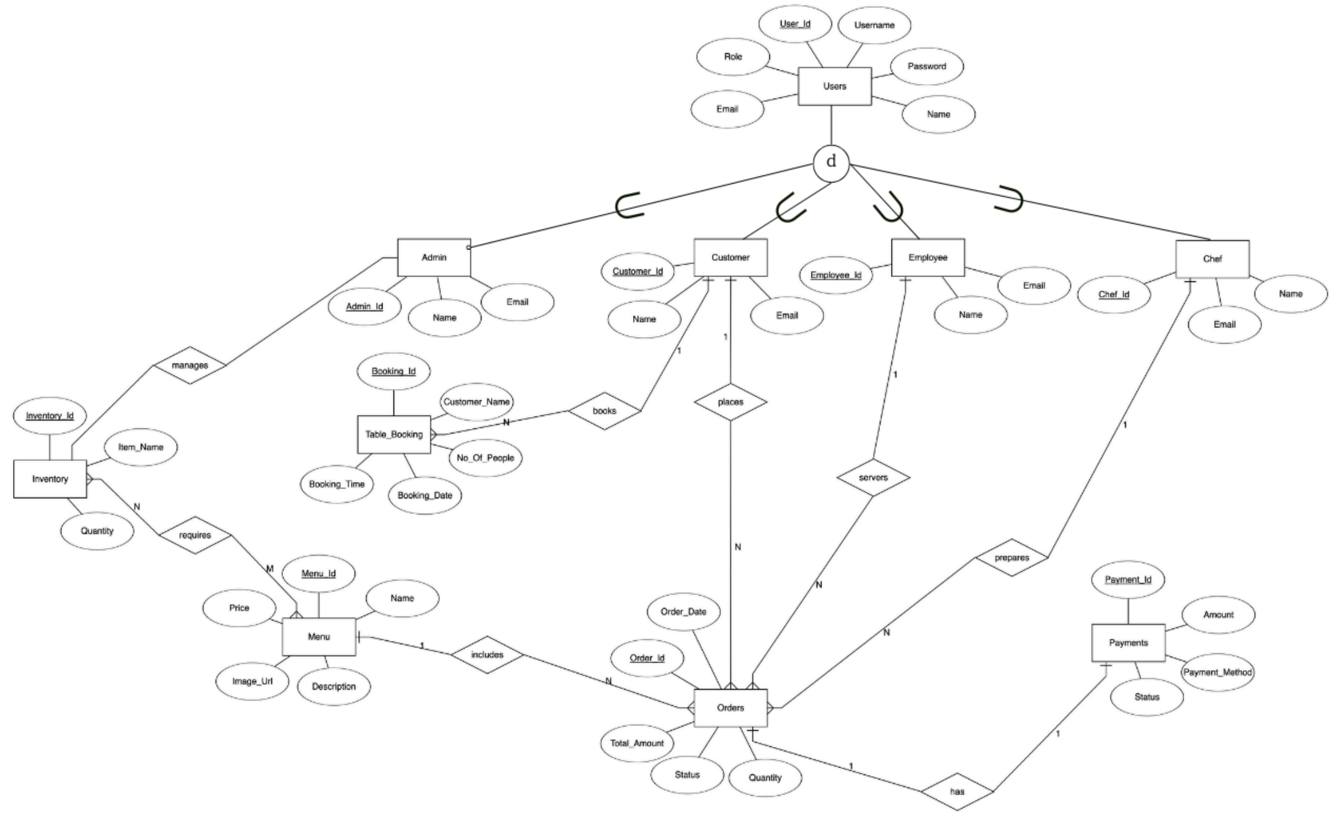
Platform	Desktop-Based Application
FrontEnd/Console	Flask, Python
Backend/server	Flask, Python, SQLite
Database	SQLite
Programming Language : Frontend	Python
Backend/Server: Programming Language	Python

d. Project Plan


August
Septemper
October
November
Deliverables
Task 1
Requirement Specification Document
for Restaurant Management System (RMS)
Task 2
Conceptual design doc ER/ EER diagrams
For RMS
Task 3
Relational model Mapping from ER/EER to Relational model LD design document
For RMS
Task 4
Implementation of Relational model SQL CODE
For RMS
Final
Application Code which accesses this DB
For RMS

II. CONCEPTUAL DESIGN:

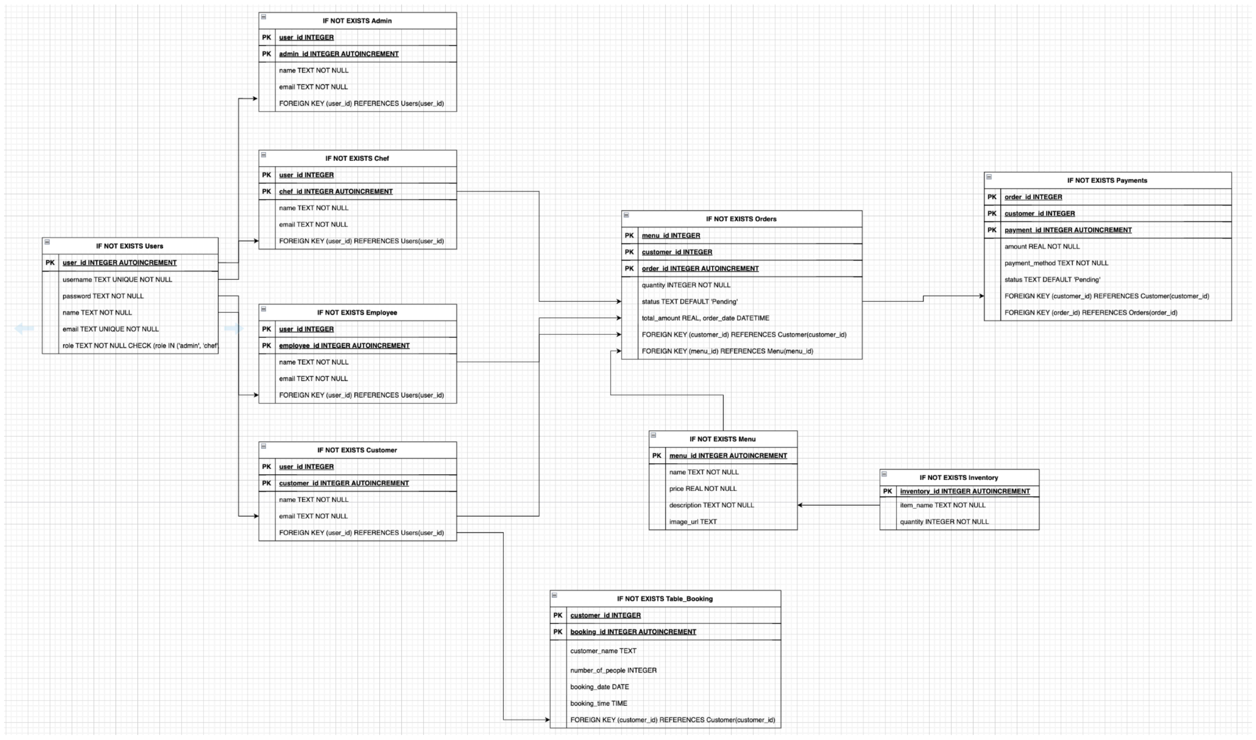
a. Entity Relationship Model



Relationships and Foreign Key Constraints

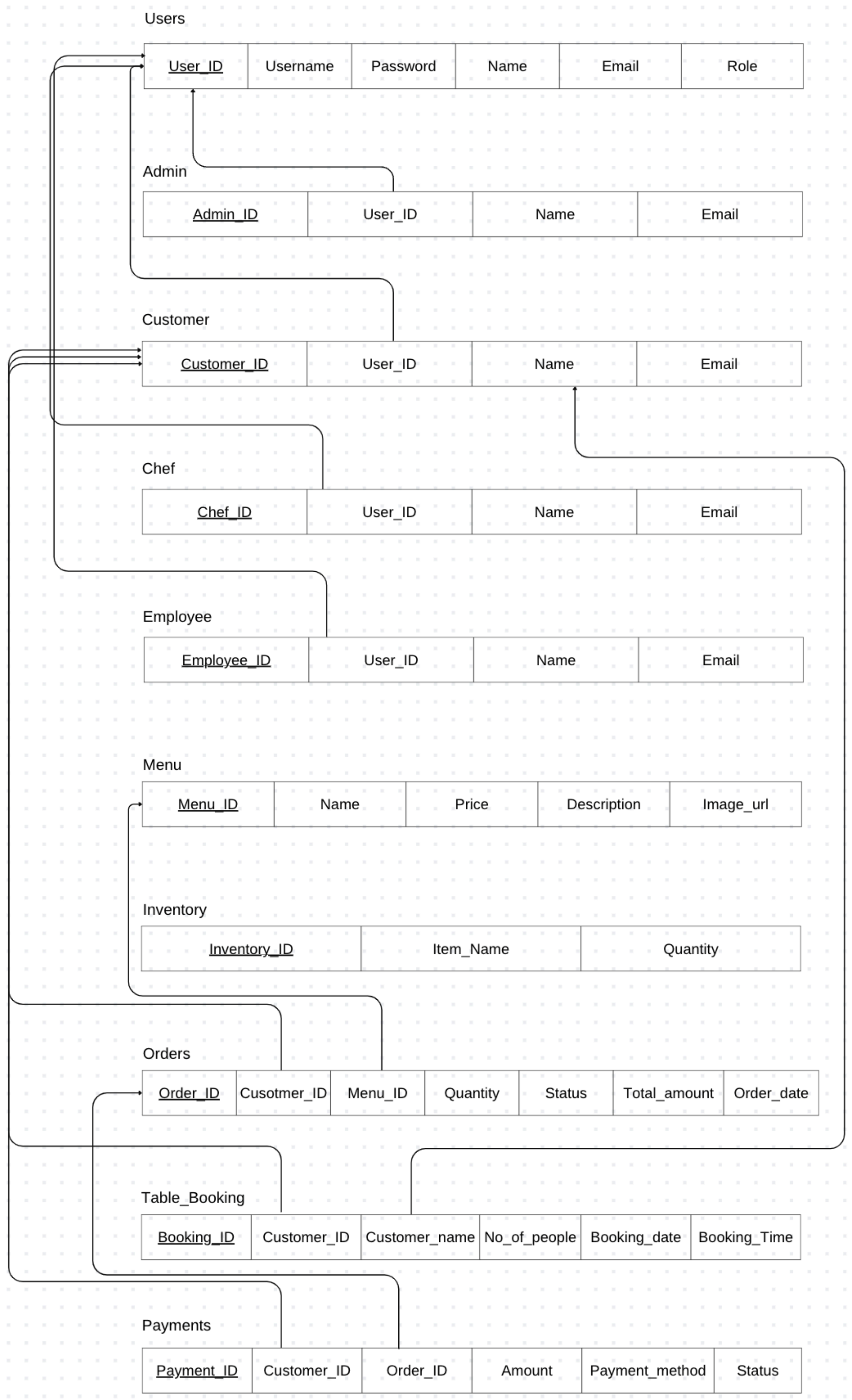
1. **Users ↔ Customer, Chef, Admin, Employee:**
   - `user_id` **in Customer, Chef, Admin, and Employee tables references** `user_id` **in Users.**
2. **Customer ↔ Orders:**
   - `customer_id` **in Orders references** `customer_id` **in Customer (1 relationship, Total Participation on Orders side).**
3. **Customer ↔ Table_Booking:**
   - `customer_id` **in Table_Booking references** `customer_id` **in Customer (1 relationship, Total Participation on Table_Booking side).**
4. **Customer ↔ Payments:**
   - `customer_id` **in Payments references** `customer_id` **in Customer (1 relationship).**
5. **Menu ↔ Orders:**
   - `menu_id` **in Orders references** `menu_id` **in Menu (M relationship, Total Participation on Orders side).**
6. **Orders ↔ Payments:**
   - `order_id` **in Payments references** `order_id` **in Orders (1:1 or N:1 relationship, Total Participation on Payments side).**

b. Object Model


IF NOT EXISTS Admin
PK user_id INTEGER
PK admin_id INTEGER AUTOINCREMENT
name TEXT NOT NULL
email TEXT NOT NULL
FOREIGN KEY (user_id) REFERENCES Users(user_id)
IF NOT EXISTS Chef
PK user_id INTEGER
PK chef_id INTEGER AUTOINCREMENT
name TEXT NOT NULL
email TEXT NOT NULL
FOREIGN KEY (user_id) REFERENCES Users(user_id)
IF NOT EXISTS Users
PK user_id INTEGER AUTOINCREMENT
username TEXT UNIQUE NOT NULL
password TEXT NOT NULL
name TEXT NOT NULL
email TEXT UNIQUE NOT NULL
role TEXT NOT NULL CHECK (role IN ('admin', 'chef'
IF NOT EXISTS Employee
PK user_id INTEGER
PK employee_id INTEGER AUTOINCREMENT
name TEXT NOT NULL
email TEXT NOT NULL
FOREIGN KEY (user_id) REFERENCES Users(user_id)
IF NOT EXISTS Customer
PK user_id INTEGER
PK customer_id INTEGER AUTOINCREMENT
name TEXT NOT NULL
email TEXT NOT NULL
FOREIGN KEY (user_id) REFERENCES Users(user_id)
IF NOT EXISTS Orders
PK menu_id INTEGER
PK customer_id INTEGER
PK order_id INTEGER AUTOINCREMENT
quantity INTEGER NOT NULL
status TEXT DEFAULT 'Pending'
total_amount REAL, order_date DATETIME
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
FOREIGN KEY (menu_id) REFERENCES Menu(menu_id)
IF NOT EXISTS Payments
PK order_id INTEGER
PK customer_id INTEGER
PK payment_id INTEGER AUTOINCREMENT
amount REAL NOT NULL
payment_method TEXT NOT NULL
status TEXT DEFAULT 'Pending'
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
FOREIGN KEY (order_id) REFERENCES Orders(order_id)
IF NOT EXISTS Menu
PK menu_id INTEGER AUTOINCREMENT
name TEXT NOT NULL
price REAL NOT NULL
description TEXT NOT NULL
image_url TEXT
IF NOT EXISTS Inventory
PK inventory_id INTEGER AUTOINCREMENT
item_name TEXT NOT NULL
quantity INTEGER NOT NULL
IF NOT EXISTS Table_Booking
PK customer_id INTEGER
PK booking_id INTEGER AUTOINCREMENT
customer_name TEXT
number_of_people INTEGER
booking_date DATE
booking_time TIME
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)

III LOGICAL DESIGN

a. Relational Database Schema



2024MT12059

Functional Dependencies :

Users(user_id, username, password, name, email, role)
user_id → username, password, name, email, role
username → user_id

Customer(customer_id, user_id, name, email)
customer_id → user_id, name, email
user_id → customer_id

Chef(chef_id, user_id, name, email)
chef_id → user_id, name, email
user_id → chef_id

Admin(admin_id, user_id, name, email)
admin_id → user_id, name, email
user_id → admin_id

Employee(employee_id, user_id, name, email)
employee_id → user_id, name, email
user_id → employee_id

Menu(menu_id, name, price, description, image_url)
menu_id → name, price, description, image_url

Inventory(inventory_id, item_name, quantity)
inventory_id → item_name, quantity
item_name → quantity

Orders(order_id, customer_id, menu_id, quantity, status, total_amount, order_date)
order_id → customer_id, menu_id, quantity, status, total_amount, order_date
customer_id, order_date → order_id

Table_Booking(booking_id, customer_id, customer_name, number_of_people, booking_date, booking_time)
booking_id → customer_id, customer_name, number_of_people, booking_date, booking_time
customer_id, booking_date, booking_time → booking_id

Payments(payment_id, customer_id, order_id, amount, payment_method, status)
payment_id → customer_id, order_id, amount, payment_method, status
order_id → payment_id

b. Normalization

1. Users Table

- **Initial Schema**: `Users(user_id, username, password, name, email, role)`
- **Primary Key**: `user_id`
- **Functional Dependencies**:
  - `user_id → username, password, name, email, role`
- **Analysis**: This table already adheres to 1NF, 2NF, and 3NF because each non-key attribute is directly dependent on the primary key.
- **Final 3NF Schema**: No changes are required.

---

2. Customer Table

- **Initial Schema**: `Customer(customer_id, user_id, name, email)`
- **Primary Key**: `customer_id`
- **Functional Dependencies**:
  - `customer_id → user_id, name, email`
- **Analysis**: Since `customer_id` is the primary key and all non-key attributes depend on it directly, this table is already in 1NF, 2NF, and 3NF.
- **Final 3NF Schema**: No changes are required.

---

3. Chef Table

- **Initial Schema**: `Chef(chef_id, user_id, name, email)`
- **Primary Key**: `chef_id`
- **Functional Dependencies**:
  - `chef_id → user_id, name, email`
- **Analysis**: This table is in 1NF, 2NF, and 3NF as `chef_id` determines all other attributes without transitive dependencies.
- **Final 3NF Schema**: No changes are required.

---

4. Admin Table

- **Initial Schema**: `Admin(admin_id, user_id, name, email)`
- **Primary Key**: `admin_id`
- **Functional Dependencies**:
  - `admin_id → user_id, name, email`
- **Analysis**: This table is in 1NF, 2NF, and 3NF as `admin_id` determines all other attributes directly.
- **Final 3NF Schema**: No changes are required.

---

5. Employee Table

- **Initial Schema**: `Employee(employee_id, user_id, name, email)`
- **Primary Key**: `employee_id`
- **Functional Dependencies**:
  - `employee_id → user_id, name, email`
- **Analysis**: This table is already in 1NF, 2NF, and 3NF since `employee_id` uniquely determines all other attributes.
- **Final 3NF Schema**: No changes are required.

---

6. Menu Table

- **Initial Schema**: `Menu(menu_id, name, price, description, image_url)`
- **Primary Key**: `menu_id`
- **Functional Dependencies**:
  - `menu_id → name, price, description, image_url`
- **Analysis**: This table is already in 1NF, 2NF, and 3NF since `menu_id` uniquely determines all other attributes.
- **Final 3NF Schema**: No changes are required.

---

7. Inventory Table

- **Initial Schema**: `Inventory(inventory_id, item_name, quantity)`
- **Primary Key**: `inventory_id`
- **Functional Dependencies**:
  - `inventory_id → item_name, quantity`
- **Analysis**: This table is already in 1NF, 2NF, and 3NF because `inventory_id` uniquely determines the item details.

- **Final 3NF Schema**: No changes are required.

---

8. Orders Table

- **Initial Schema**: `Orders(order_id, customer_id, menu_id, quantity, status, total_amount, order_date)`
- **Primary Key**: `order_id`
- **Functional Dependencies**:
    - `order_id → customer_id, menu_id, quantity, status, total_amount, order_date`
- **Analysis**: The table is in 1NF (each column contains atomic values) and 2NF (no partial dependency on the primary key).
- However, `total_amount` could be a transitive dependency if calculated as the sum of `quantity * price` for items in the Menu table.
- **Solution**: Remove `total_amount` from the **Orders** table to eliminate the potential transitive dependency.
- Introduce a JSON column, `items`, to store multiple menu items and their quantities for a single order.

Final 3NF Schema:
`Orders(order_id, customer_id, menu_id, quantity, status, order_date)`

Analysis:

- The table is in 1NF (each column contains atomic values) and 2NF (no partial dependency on the primary key).
- However, `total_amount` could be a transitive dependency if calculated as the sum of `quantity * price` for items in the Menu table.

Solution:

- Remove `total_amount` from the **Orders** table to eliminate the potential transitive dependency.
- Introduce a JSON column, `items`, to store multiple menu items and their quantities for a single order.

Explanation of `items` Column Format:

- The `items` column stores a JSON array, with each element containing `menu_id` and `quantity`.

For example, an order with two items (menu ID 1 with quantity 2, and menu ID 3 with quantity 1) would be stored as:

```
[
    {"menu_id": 1, "quantity": 2},
    {"menu_id": 3, "quantity": 1}
]
```

Total Amount Calculation:

- `total_amount` can be calculated dynamically by:
    - Parsing the `items` column,
    - Retrieving prices for each `menu_id` from the **Menu** table,
    - Multiplying each price by its respective `quantity`, and
    - Summing the values for the final total.

Final 3NF Schema Summary:

`Orders(order_id, customer_id, items, status, order_date)`

---

9. Table_Booking Table

- **Initial Schema**: `Table_Booking(booking_id, customer_id, customer_name, number_of_people, booking_date, booking_time)`
- **Primary Key**: `booking_id`
- **Functional Dependencies**:
    - `booking_id → customer_id, customer_name, number_of_people, booking_date, booking_time`
- **Analysis**: The table is in 1NF and 2NF, but `customer_name` is a transitive dependency since it can be derived from the `Customer` table.
- **Solution**: Remove `customer_name` from `Table_Booking`.

Final 3NF Schema:
`Table_Booking(booking_id, customer_id, number_of_people, booking_date, booking_time)`

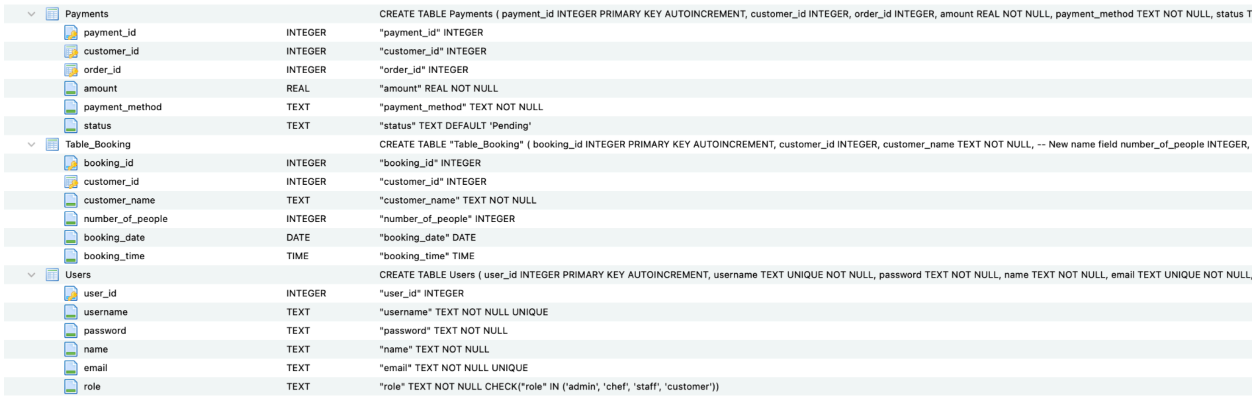
---

10. Payments Table

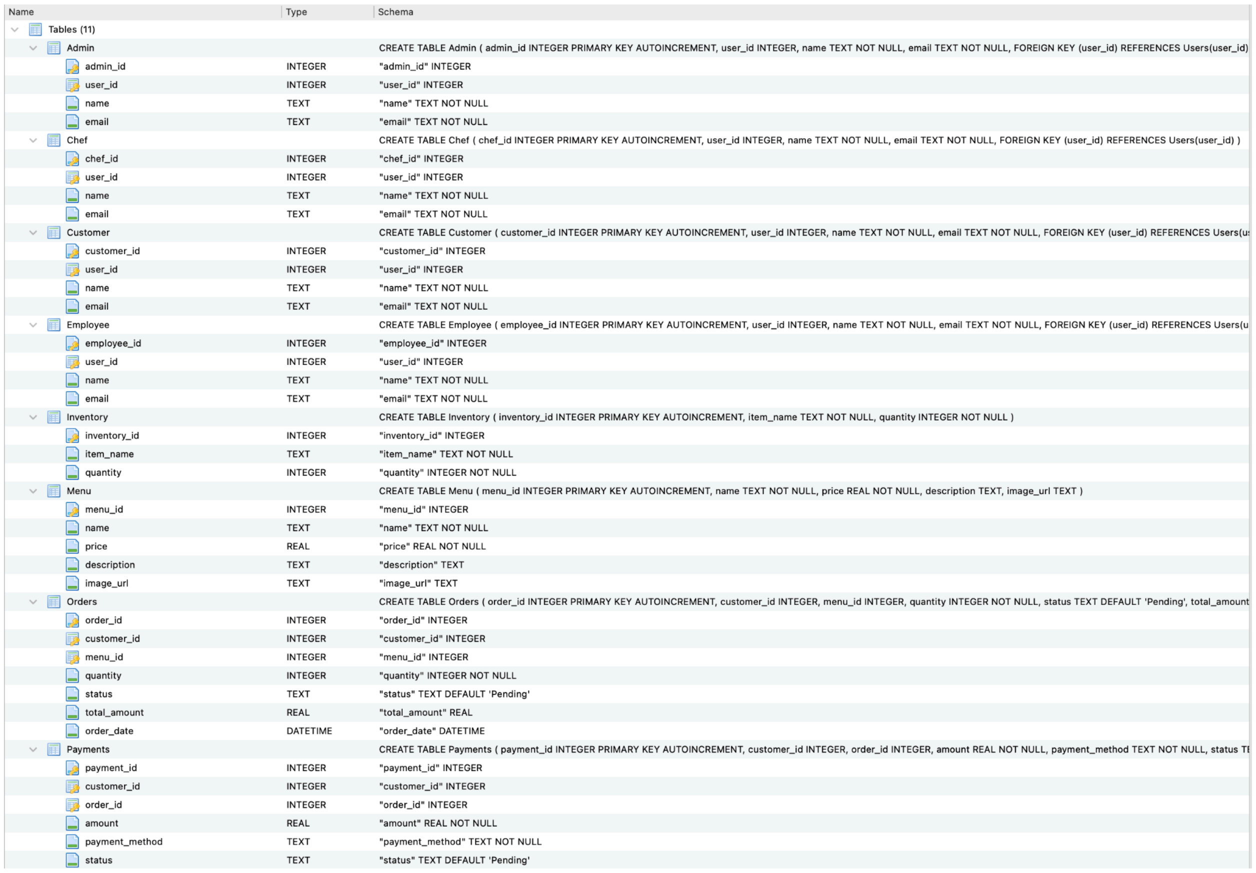
- **Initial Schema**: `Payments(payment_id, order_id, amount, payment_method, status)`
- **Primary Key**: `payment_id`
- **Functional Dependencies**:
  - `payment_id → order_id, amount, payment_method, status`
- **Analysis**: This table is in 1NF, 2NF, and 3NF because `payment_id` determines all other attributes directly.
- **Final 3NF Schema**: No changes are required.

c. Data Dictionary

Table definition

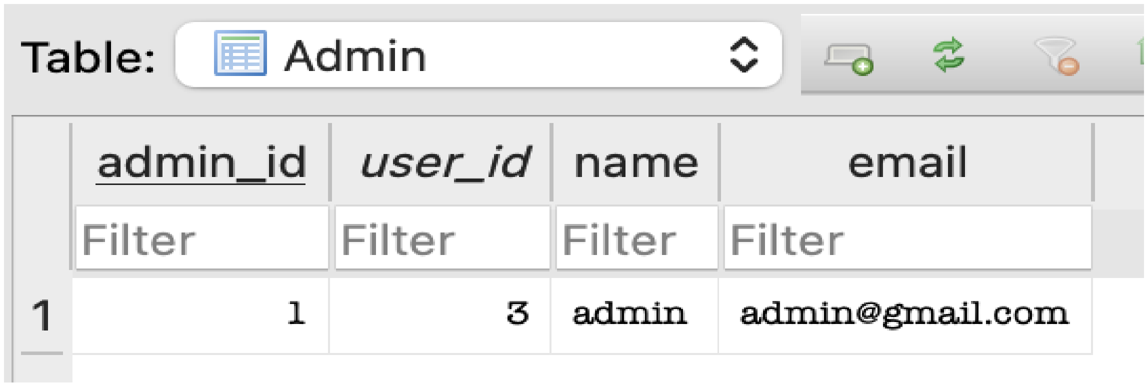


Payments		CREATE TABLE Payments (payment_id INTEGER PRIMARY KEY AUTOINCREMENT, customer_id INTEGER, order_id INTEGER, amount REAL NOT NULL, payment_method TEXT NOT NULL, status T
payment_id	INTEGER	"payment_id" INTEGER
customer_id	INTEGER	"customer_id" INTEGER
order_id	INTEGER	"order_id" INTEGER
amount	REAL	"amount" REAL NOT NULL
payment_method	TEXT	"payment_method" TEXT NOT NULL
status	TEXT	"status" TEXT DEFAULT 'Pending'
Table_Booking		CREATE TABLE "Table_Booking" (booking_id INTEGER PRIMARY KEY AUTOINCREMENT, customer_id INTEGER, customer_name TEXT NOT NULL, -- New name field number_of_people INTEGER,
booking_id	INTEGER	"booking_id" INTEGER
customer_id	INTEGER	"customer_id" INTEGER
customer_name	TEXT	"customer_name" TEXT NOT NULL
number_of_people	INTEGER	"number_of_people" INTEGER
booking_date	DATE	"booking_date" DATE
booking_time	TIME	"booking_time" TIME
Users		CREATE TABLE Users (user_id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT UNIQUE NOT NULL, password TEXT NOT NULL, name TEXT NOT NULL, email TEXT UNIQUE NOT NULL,
user_id	INTEGER	"user_id" INTEGER
username	TEXT	"username" TEXT NOT NULL UNIQUE
password	TEXT	"password" TEXT NOT NULL
name	TEXT	"name" TEXT NOT NULL
email	TEXT	"email" TEXT NOT NULL UNIQUE
role	TEXT	"role" TEXT NOT NULL CHECK("role" IN ('admin', 'chef', 'staff', 'customer'))



Name	Type	Schema
Tables (11)		
Admin		CREATE TABLE Admin (admin_id INTEGER PRIMARY KEY AUTOINCREMENT, user_id INTEGER, name TEXT NOT NULL, email TEXT NOT NULL, FOREIGN KEY (user_id) REFERENCES Users(user_id)
admin_id	INTEGER	"admin_id" INTEGER
user_id	INTEGER	"user_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
email	TEXT	"email" TEXT NOT NULL
Chef		CREATE TABLE Chef (chef_id INTEGER PRIMARY KEY AUTOINCREMENT, user_id INTEGER, name TEXT NOT NULL, email TEXT NOT NULL, FOREIGN KEY (user_id) REFERENCES Users(user_id))
chef_id	INTEGER	"chef_id" INTEGER
user_id	INTEGER	"user_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
email	TEXT	"email" TEXT NOT NULL
Customer		CREATE TABLE Customer (customer_id INTEGER PRIMARY KEY AUTOINCREMENT, user_id INTEGER, name TEXT NOT NULL, email TEXT NOT NULL, FOREIGN KEY (user_id) REFERENCES Users(u
customer_id	INTEGER	"customer_id" INTEGER
user_id	INTEGER	"user_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
email	TEXT	"email" TEXT NOT NULL
Employee		CREATE TABLE Employee (employee_id INTEGER PRIMARY KEY AUTOINCREMENT, user_id INTEGER, name TEXT NOT NULL, email TEXT NOT NULL, FOREIGN KEY (user_id) REFERENCES Users(u
employee_id	INTEGER	"employee_id" INTEGER
user_id	INTEGER	"user_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
email	TEXT	"email" TEXT NOT NULL
Inventory		CREATE TABLE Inventory (inventory_id INTEGER PRIMARY KEY AUTOINCREMENT, item_name TEXT NOT NULL, quantity INTEGER NOT NULL)
inventory_id	INTEGER	"inventory_id" INTEGER
item_name	TEXT	"item_name" TEXT NOT NULL
quantity	INTEGER	"quantity" INTEGER NOT NULL
Menu		CREATE TABLE Menu (menu_id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, price REAL NOT NULL, description TEXT, image_url TEXT)
menu_id	INTEGER	"menu_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
price	REAL	"price" REAL NOT NULL
description	TEXT	"description" TEXT
image_url	TEXT	"image_url" TEXT
Orders		CREATE TABLE Orders (order_id INTEGER PRIMARY KEY AUTOINCREMENT, customer_id INTEGER, menu_id INTEGER, quantity INTEGER NOT NULL, status TEXT DEFAULT 'Pending', total_amount
order_id	INTEGER	"order_id" INTEGER
customer_id	INTEGER	"customer_id" INTEGER
menu_id	INTEGER	"menu_id" INTEGER
quantity	INTEGER	"quantity" INTEGER NOT NULL
status	TEXT	"status" TEXT DEFAULT 'Pending'
total_amount	REAL	"total_amount" REAL
order_date	DATETIME	"order_date" DATETIME
Payments		CREATE TABLE Payments (payment_id INTEGER PRIMARY KEY AUTOINCREMENT, customer_id INTEGER, order_id INTEGER, amount REAL NOT NULL, payment_method TEXT NOT NULL, status TE
payment_id	INTEGER	"payment_id" INTEGER
customer_id	INTEGER	"customer_id" INTEGER
order_id	INTEGER	"order_id" INTEGER
amount	REAL	"amount" REAL NOT NULL
payment_method	TEXT	"payment_method" TEXT NOT NULL
status	TEXT	"status" TEXT DEFAULT 'Pending'

Data contents



	admin_id	user_id	name	email
	Filter	Filter	Filter	Filter
1	1	3	admin	admin@gmail.com

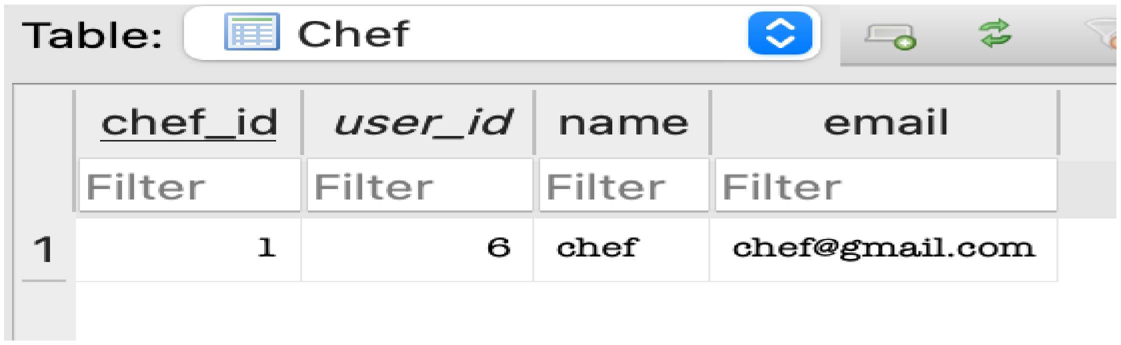


Table: Chef

	chef_id	user_id	name	email
1	1	6	chef	chef@gmail.com

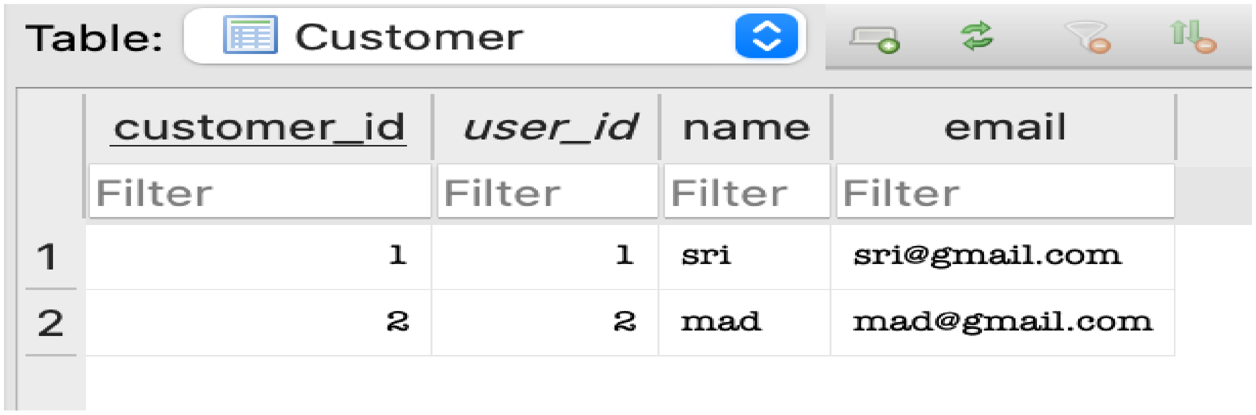


Table: Customer

	customer_id	user_id	name	email
1	1	1	sri	sri@gmail.com
2	2	2	mad	mad@gmail.com

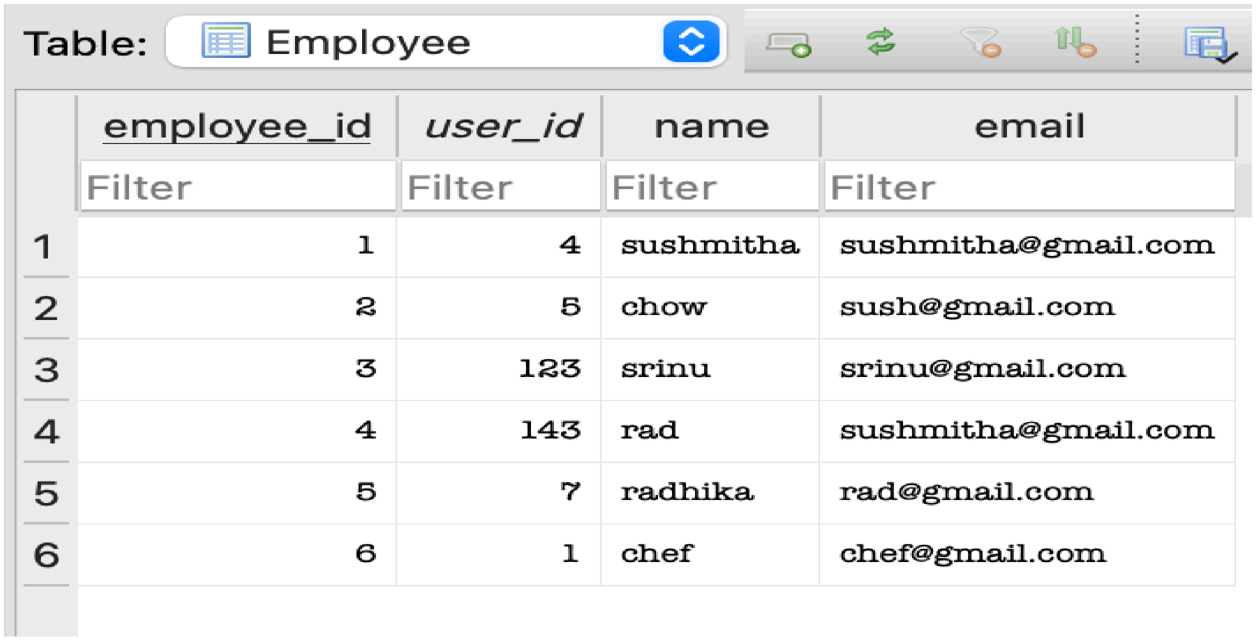


Table: Employee

	employee_id	user_id	name	email
1	1	4	sushmitha	sushmitha@gmail.com
2	2	5	chow	sush@gmail.com
3	3	123	srinu	srinu@gmail.com
4	4	143	rad	sushmitha@gmail.com
5	5	7	radhika	rad@gmail.com
6	6	1	chef	chef@gmail.com

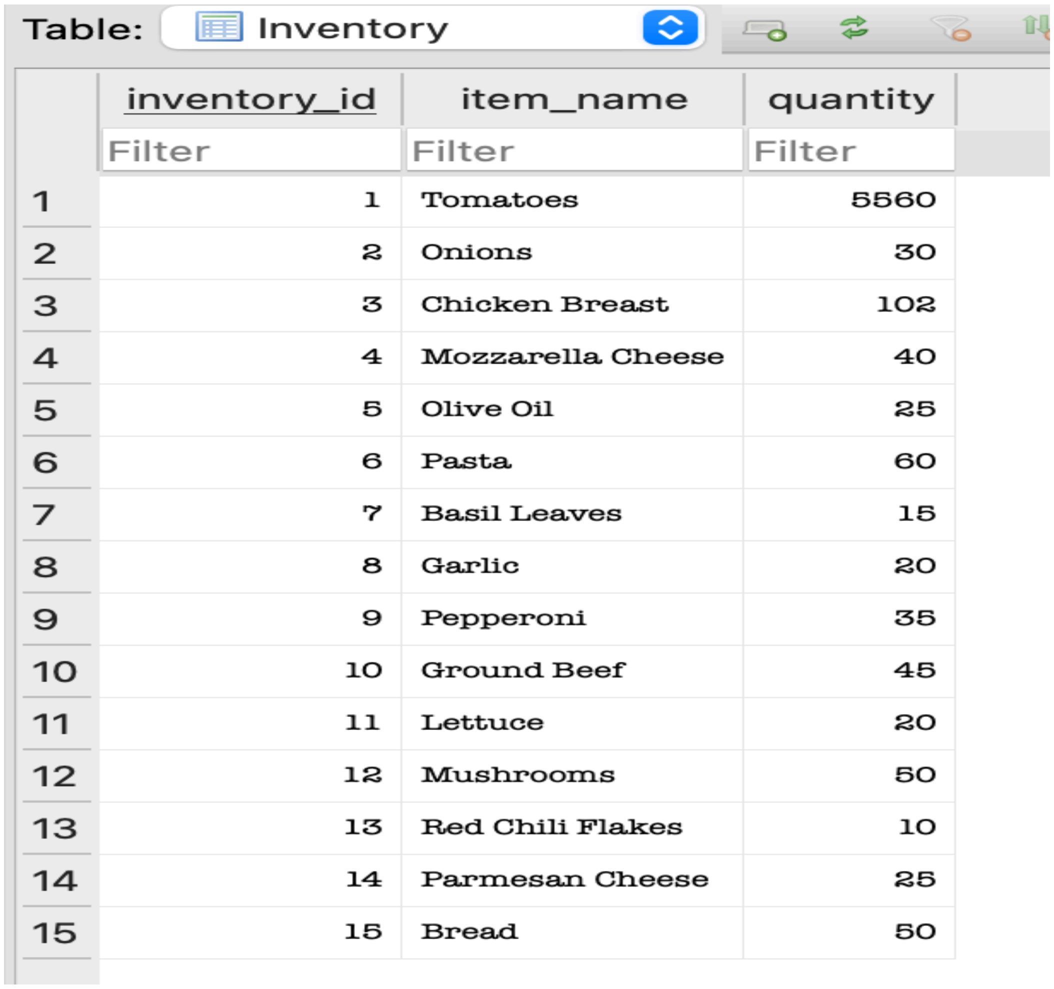


Table: Inventory

	inventory_id	item_name	quantity
	Filter	Filter	Filter
1	1	Tomatoes	5560
2	2	Onions	30
3	3	Chicken Breast	102
4	4	Mozzarella Cheese	40
5	5	Olive Oil	25
6	6	Pasta	60
7	7	Basil Leaves	15
8	8	Garlic	20
9	9	Pepperoni	35
10	10	Ground Beef	45
11	11	Lettuce	20
12	12	Mushrooms	50
13	13	Red Chili Flakes	10
14	14	Parmesan Cheese	25
15	15	Bread	50

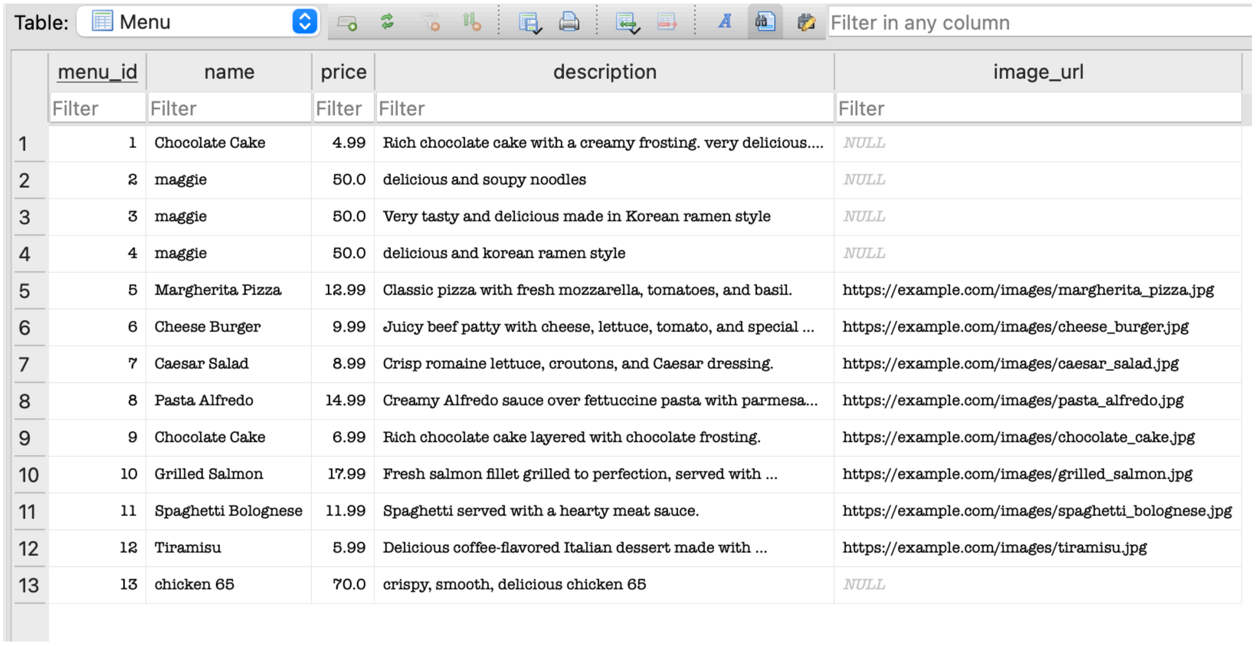


Table: Menu

	menu_id	name	price	description	image_url
1	1	Chocolate Cake	4.99	Rich chocolate cake with a creamy frosting. very delicious....	NULL
2	2	maggie	50.0	delicious and soupy noodles	NULL
3	3	maggie	50.0	Very tasty and delicious made in Korean ramen style	NULL
4	4	maggie	50.0	delicious and korean ramen style	NULL
5	5	Margherita Pizza	12.99	Classic pizza with fresh mozzarella, tomatoes, and basil.	https://example.com/images/margherita_pizza.jpg
6	6	Cheese Burger	9.99	Juicy beef patty with cheese, lettuce, tomato, and special ...	https://example.com/images/cheese_burger.jpg
7	7	Caesar Salad	8.99	Crisp romaine lettuce, croutons, and Caesar dressing.	https://example.com/images/caesar_salad.jpg
8	8	Pasta Alfredo	14.99	Creamy Alfredo sauce over fettuccine pasta with parmesa...	https://example.com/images/pasta_alfredo.jpg
9	9	Chocolate Cake	6.99	Rich chocolate cake layered with chocolate frosting.	https://example.com/images/chocolate_cake.jpg
10	10	Grilled Salmon	17.99	Fresh salmon fillet grilled to perfection, served with ...	https://example.com/images/grilled_salmon.jpg
11	11	Spaghetti Bolognese	11.99	Spaghetti served with a hearty meat sauce.	https://example.com/images/spaghetti_bolognese.jpg
12	12	Tiramisu	5.99	Delicious coffee-flavored Italian dessert made with ...	https://example.com/images/tiramisu.jpg
13	13	chicken 65	70.0	crispy, smooth, delicious chicken 65	NULL

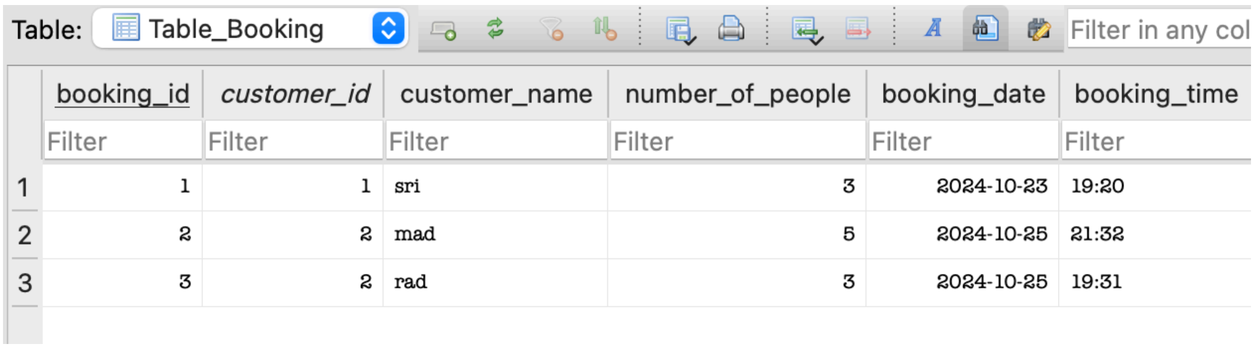


Table: Table_Booking

	booking_id	customer_id	customer_name	number_of_people	booking_date	booking_time
1	1	1	sri	3	2024-10-23	19:20
2	2	2	mad	5	2024-10-25	21:32
3	3	2	rad	3	2024-10-25	19:31

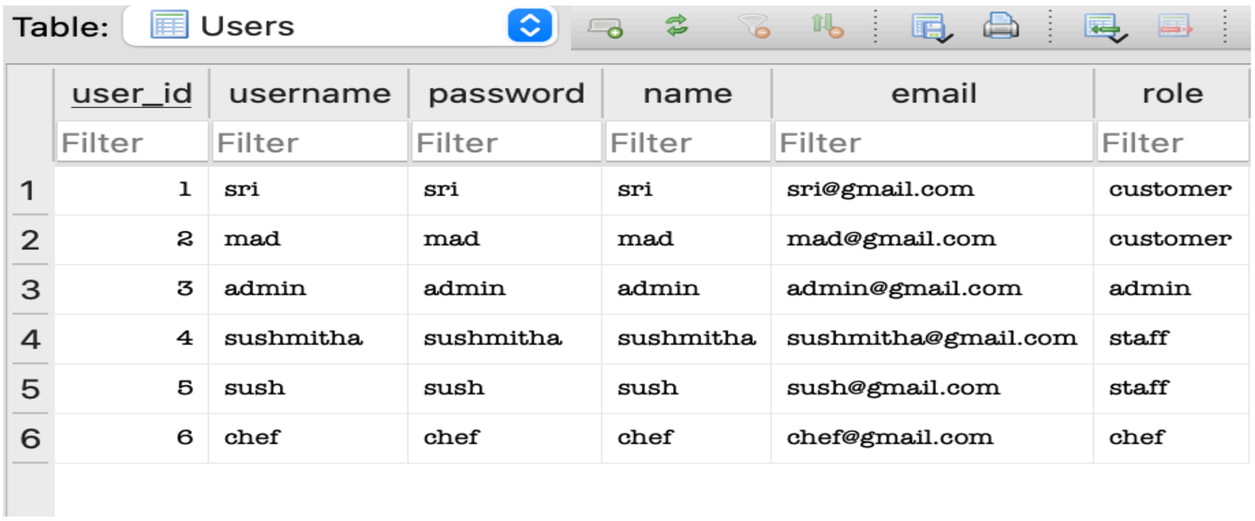


Table: Users

	user_id	username	password	name	email	role
1	1	sri	sri	sri	sri@gmail.com	customer
2	2	mad	mad	mad	mad@gmail.com	customer
3	3	admin	admin	admin	admin@gmail.com	admin
4	4	sushmitha	sushmitha	sushmitha	sushmitha@gmail.com	staff
5	5	sush	sush	sush	sush@gmail.com	staff
6	6	chef	chef	chef	chef@gmail.com	chef

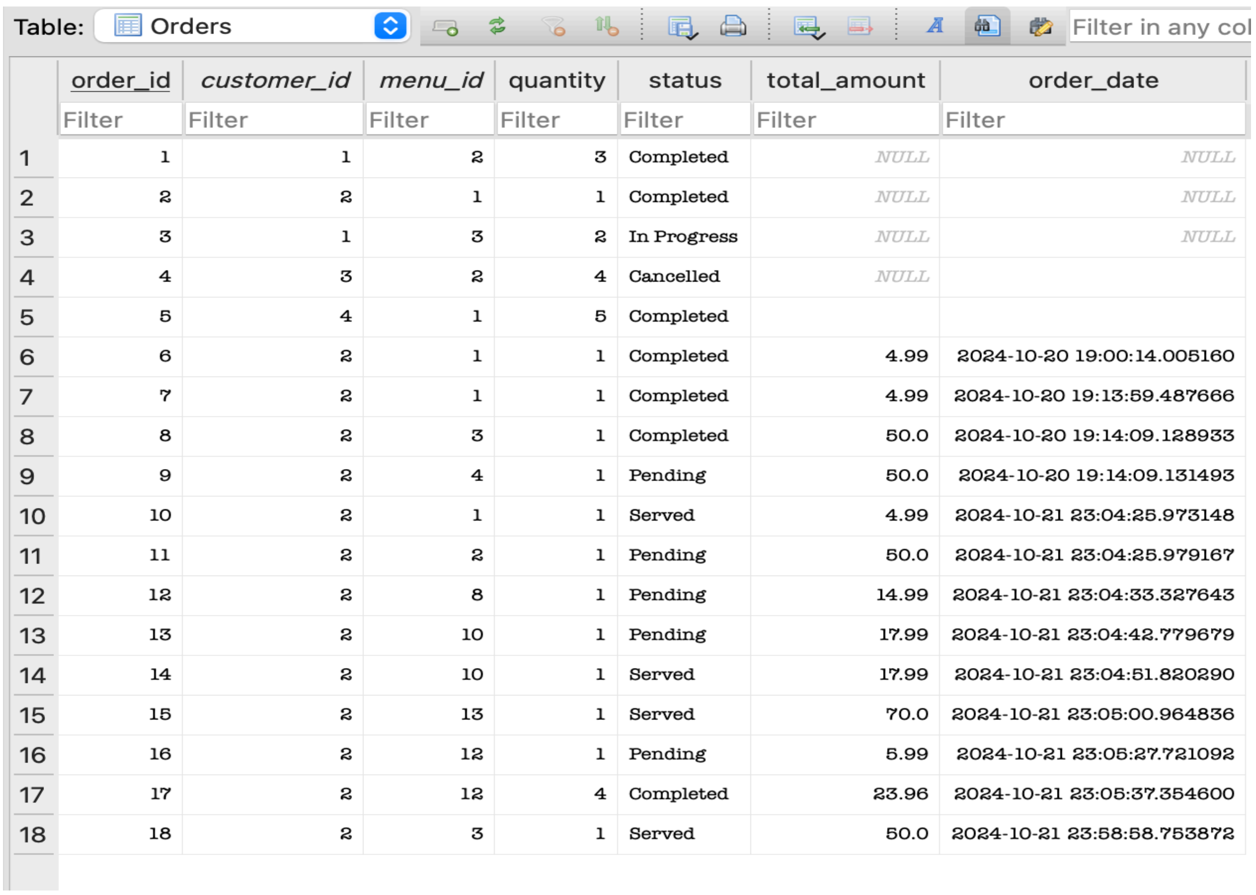


Table: Orders

Filter in any col

	order_id	customer_id	menu_id	quantity	status	total_amount	order_date
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	2	3	Completed	NULL	NULL
2	2	2	1	1	Completed	NULL	NULL
3	3	1	3	2	In Progress	NULL	NULL
4	4	3	2	4	Cancelled	NULL	
5	5	4	1	5	Completed		
6	6	2	1	1	Completed	4.99	2024-10-20 19:00:14.005160
7	7	2	1	1	Completed	4.99	2024-10-20 19:13:59.487666
8	8	2	3	1	Completed	50.0	2024-10-20 19:14:09.128933
9	9	2	4	1	Pending	50.0	2024-10-20 19:14:09.131493
10	10	2	1	1	Served	4.99	2024-10-21 23:04:25.973148
11	11	2	2	1	Pending	50.0	2024-10-21 23:04:25.979167
12	12	2	8	1	Pending	14.99	2024-10-21 23:04:33.327643
13	13	2	10	1	Pending	17.99	2024-10-21 23:04:42.779679
14	14	2	10	1	Served	17.99	2024-10-21 23:04:51.820290
15	15	2	13	1	Served	70.0	2024-10-21 23:05:00.964836
16	16	2	12	1	Pending	5.99	2024-10-21 23:05:27.721092
17	17	2	12	4	Completed	23.96	2024-10-21 23:05:37.354600
18	18	2	3	1	Served	50.0	2024-10-21 23:58:58.753872

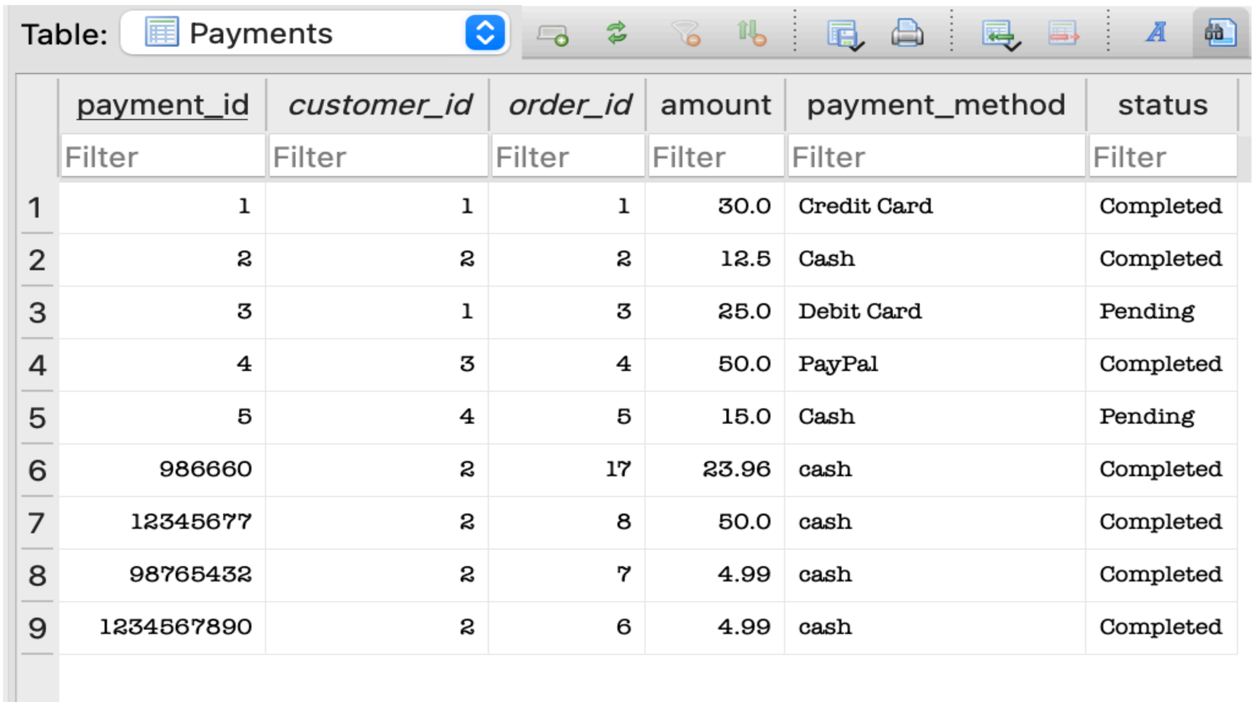


Table: Payments

	payment_id	customer_id	order_id	amount	payment_method	status
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	30.0	Credit Card	Completed
2	2	2	2	12.5	Cash	Completed
3	3	1	3	25.0	Debit Card	Pending
4	4	3	4	50.0	PayPal	Completed
5	5	4	5	15.0	Cash	Pending
6	986660	2	17	23.96	cash	Completed
7	12345677	2	8	50.0	cash	Completed
8	98765432	2	7	4.99	cash	Completed
9	1234567890	2	6	4.99	cash	Completed

IV PHYSICAL DESIGN

1. SQL Statements

```
CREATE TABLE Users (

    user_id INTEGER PRIMARY KEY AUTOINCREMENT,

    username TEXT UNIQUE NOT NULL,

    password TEXT NOT NULL,

    name TEXT NOT NULL,

    email TEXT UNIQUE NOT NULL,

    role TEXT NOT NULL CHECK (role IN ('admin', 'chef', 'staff',
'customer'))

);


CREATE TABLE Customer (

    customer_id INTEGER PRIMARY KEY AUTOINCREMENT,

    user_id INTEGER UNIQUE,

    name TEXT NOT NULL,

    email TEXT NOT NULL,

    FOREIGN KEY (user_id) REFERENCES Users(user_id)

);


CREATE TABLE Chef (

    chef_id INTEGER PRIMARY KEY AUTOINCREMENT,

    user_id INTEGER UNIQUE,
    name TEXT NOT NULL,

    email TEXT NOT NULL,

    FOREIGN KEY (user_id) REFERENCES Users(user_id)

);


CREATE TABLE Admin (

    admin_id INTEGER PRIMARY KEY AUTOINCREMENT,

    user_id INTEGER UNIQUE,

    name TEXT NOT NULL,

    email TEXT NOT NULL,

    FOREIGN KEY (user_id) REFERENCES Users(user_id)

);


CREATE TABLE Employee (

    employee_id INTEGER PRIMARY KEY AUTOINCREMENT,

    user_id INTEGER UNIQUE,

    name TEXT NOT NULL,

    email TEXT NOT NULL,

    FOREIGN KEY (user_id) REFERENCES Users(user_id)

);


CREATE TABLE Menu (

    menu_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,

    price REAL NOT NULL,

    description TEXT NOT NULL,

    image_url TEXT

);


CREATE TABLE Inventory (

    inventory_id INTEGER PRIMARY KEY AUTOINCREMENT,

    item_name TEXT NOT NULL,

    quantity INTEGER NOT NULL

);


CREATE TABLE Orders (

    order_id INTEGER PRIMARY KEY AUTOINCREMENT,

    customer_id INTEGER,

    menu_id INTEGER,

    quantity INTEGER NOT NULL,

    status TEXT DEFAULT 'Pending',

    order_date DATETIME,

    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),

    FOREIGN KEY (menu_id) REFERENCES Menu(menu_id)

);
```

```
CREATE TABLE Orders (
    order_id INTEGER PRIMARY KEY AUTOINCREMENT,
    customer_id INTEGER,
    items TEXT NOT NULL,
    status TEXT DEFAULT 'Pending',
    total_amount REAL,
    order_date DATETIME,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Table_Booking (
    booking_id INTEGER PRIMARY KEY AUTOINCREMENT,
    customer_id INTEGER,
    number_of_people INTEGER,
    booking_date DATE,
    booking_time TIME,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Payments (
    payment_id INTEGER PRIMARY KEY AUTOINCREMENT,
    order_id INTEGER,
    amount REAL NOT NULL,
    payment_method TEXT NOT NULL,

    status TEXT DEFAULT 'Pending',

    FOREIGN KEY (order_id) REFERENCES Orders(order_id)

);
```

2. Indexes

To improve performance, indexes are created on frequently searched columns:

```
CREATE INDEX idx_user_username ON Users(username);
CREATE INDEX idx_order_customer ON Orders(customer_id);
CREATE INDEX idx_inventory_item ON Inventory(item_name);
```

3. Triggers

Triggers maintain database integrity and automate certain processes.

Trigger to Update Inventory on Order Creation

This trigger adjusts inventory quantities when an order is placed.

```
CREATE TRIGGER adjust_inventory_on_order
AFTER INSERT ON Orders
FOR EACH ROW
BEGIN
    DECLARE item_id INT;
    DECLARE item_qty INT;
    DECLARE menu_id INT;

    -- Parse items JSON and update inventory based on quantity
    DECLARE cursor_items CURSOR FOR
        SELECT menu_id, quantity FROM JSON_TABLE(NEW.items, "$[*]"
        COLUMNS(menu_id INT PATH "$.menu_id", quantity INT PATH
"$.quantity"));

    OPEN cursor_items;
    FETCH cursor_items INTO menu_id, item_qty;

    WHILE (FETCH_STATUS = 0) DO
        UPDATE Inventory
        SET quantity = quantity - item_qty
        WHERE item_name = (SELECT name FROM Menu WHERE menu_id =
menu_id);

        FETCH cursor_items INTO menu_id, item_qty;
    END WHILE;

    CLOSE cursor_items;
END;
```

> **Explanation**: This trigger checks the ordered items' quantities and deducts them from the inventory. JSON parsing may vary by SQL system support.

Trigger to Update Payment Status

This trigger marks an order as "Completed" once the payment status is "Paid."

```
CREATE TRIGGER update_order_on_payment
AFTER UPDATE ON Payments
FOR EACH ROW
WHEN NEW.status = 'Paid'
BEGIN
    UPDATE Orders
    SET status = 'Completed'
    WHERE order_id = NEW.order_id;
END;
```

> **Explanation**: Automates order completion when payment is finalized.

4. Stored Procedures

Stored procedures streamline complex operations like placing orders or updating inventory.

Procedure to Place Order

This procedure handles placing an order and updating related records.

```
CREATE PROCEDURE PlaceOrder(
    IN cust_id INT,
    IN items JSON,
    IN order_status TEXT
)
BEGIN
    DECLARE total_amount REAL DEFAULT 0;

    -- Insert Order
    INSERT INTO Orders (customer_id, items, status, order_date)
    VALUES (cust_id, items, order_status, NOW());

    -- Update Inventory based on ordered items
    CALL UpdateInventory(items);
END;
```

Procedure to Update Inventory

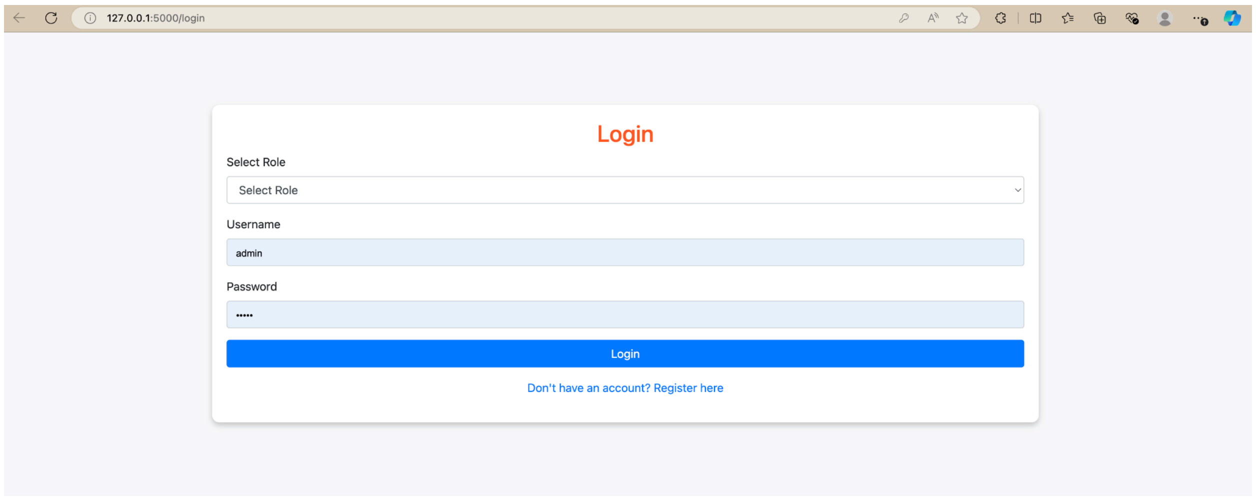
```
CREATE PROCEDURE UpdateInventory(IN items JSON)
BEGIN
    DECLARE item_id INT;
    DECLARE item_qty INT;

    DECLARE cursor_items CURSOR FOR
        SELECT menu_id, quantity FROM JSON_TABLE(items, "$[*]"
        COLUMNS(menu_id INT PATH "$.menu_id", quantity INT PATH
"$.quantity"));

    OPEN cursor_items;
    FETCH cursor_items INTO item_id, item_qty;
    WHILE (FETCH_STATUS = 0) DO
        UPDATE Inventory
        SET quantity = quantity - item_qty
        WHERE item_name = (SELECT name FROM Menu WHERE menu_id =
item_id);

        FETCH cursor_items INTO item_id, item_qty;
    END WHILE;

    CLOSE cursor_items;
END;
```

4. Front End

a. Login Page

The Login Page provides a secure entry point for users to access their respective dashboards.
Key Features:

- **User Authentication:** Users can log in using their registered email and password.
- **Role-based Access Control:** Depending on the user's role (Admin, Chef, Staff, Customer), they are directed to the corresponding dashboard.
- **Error Handling:** Provides feedback for incorrect credentials and other login issues.
- **Session Tracking**: Maintains user sessions, ensuring users remain logged in while interacting with their dashboard.

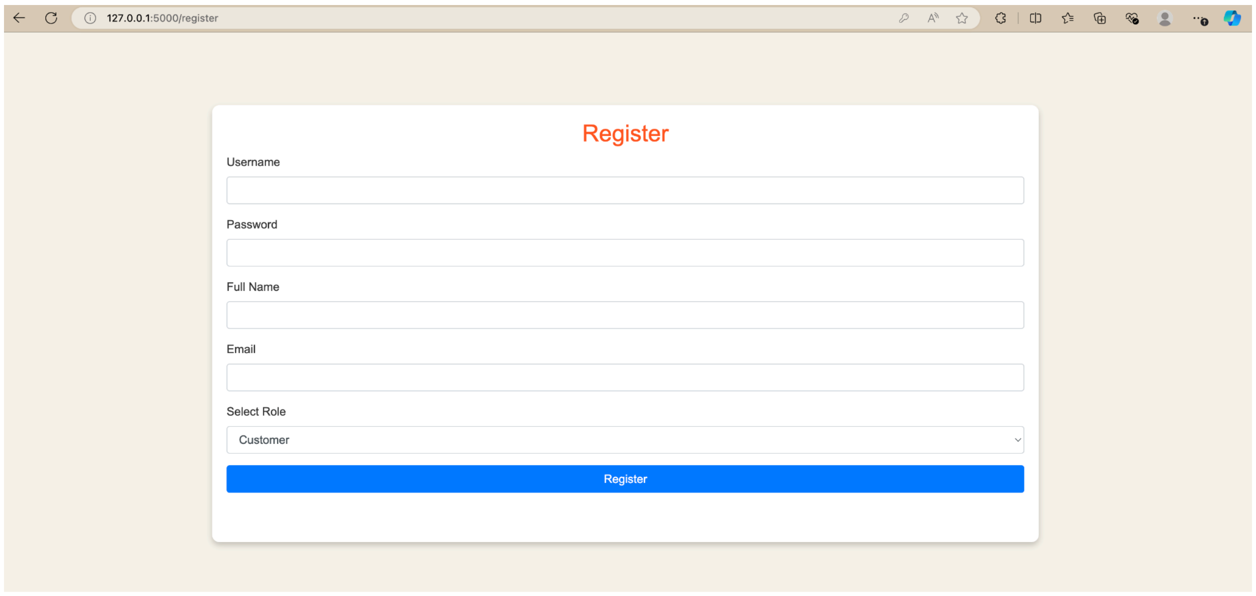


b. Registration Page

The Registration Page allows new users to create accounts to access the system.

Key Features:

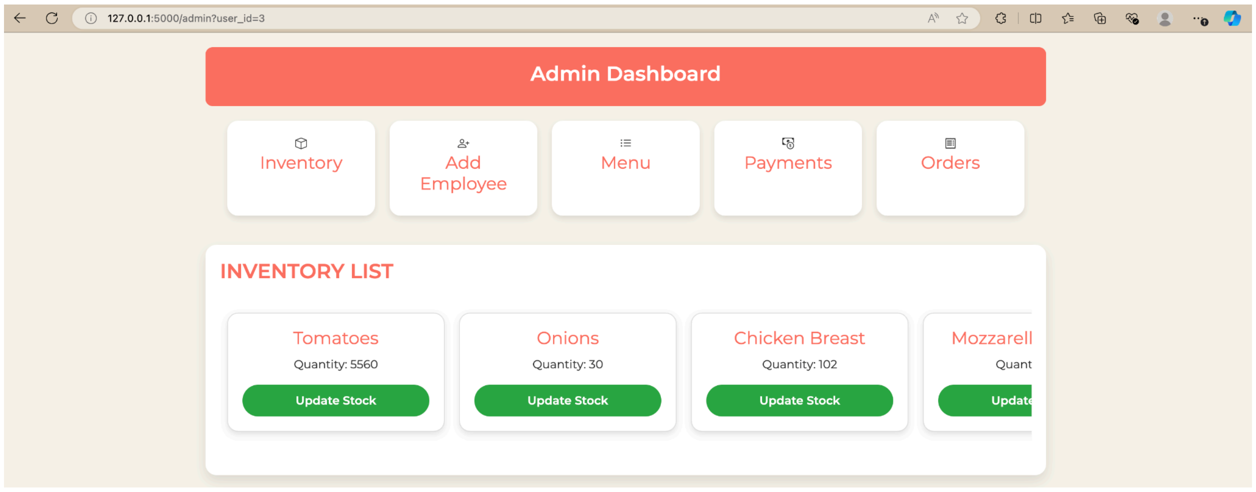
- **User Information Collection: Users can enter their name, email, password, and role (if applicable) during registration.**
- **Validation: The system validates input fields for completeness and correctness (e.g., email format).**
- **User Creation: Upon successful validation, the user account is created, allowing them to log in immediately.**
- **Feedback Mechanism: Users receive confirmation of successful registration or errors in the registration process.**
- **Error Handling**: Alerts users to any missing or incorrectly entered information, guiding them through the registration process.

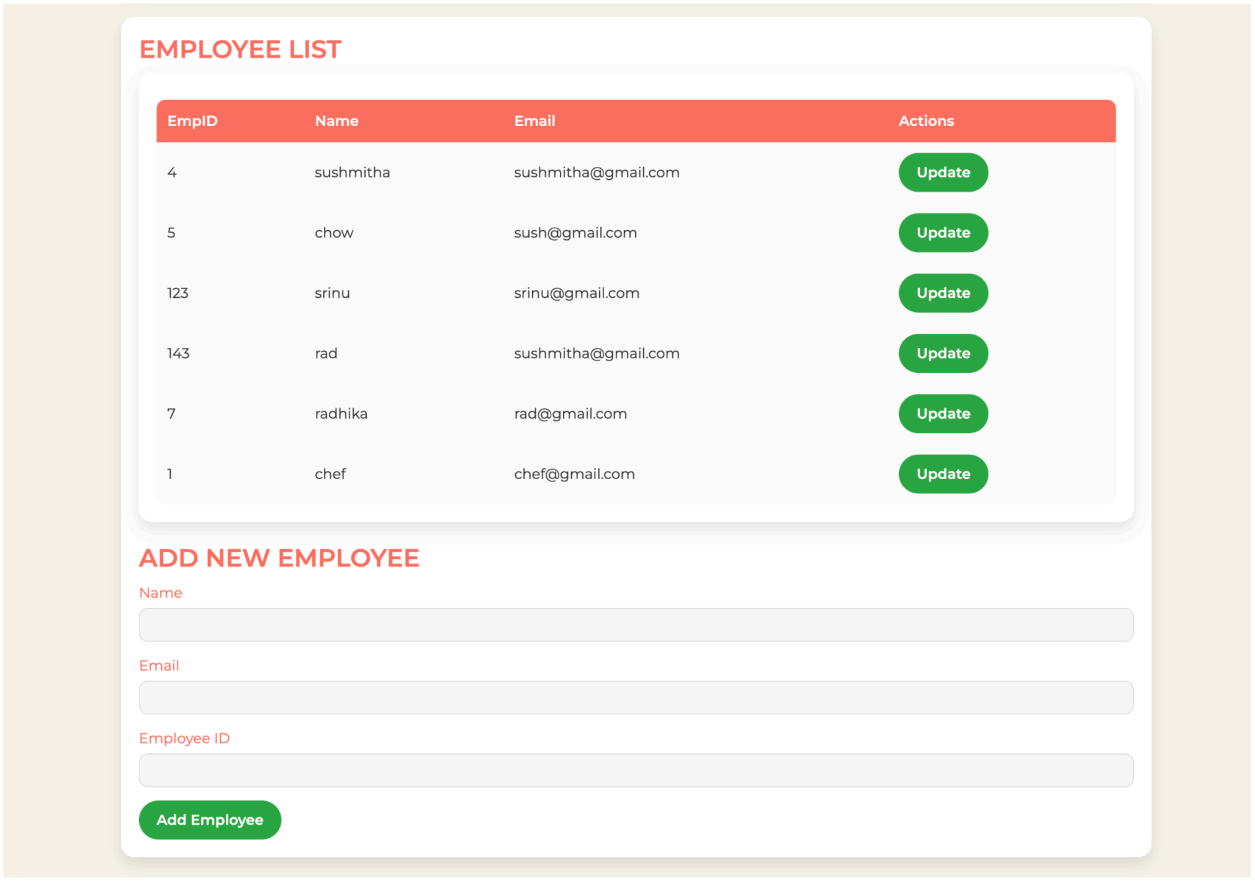


c. Admin Dashboard

The Admin Dashboard provides administrative control over the restaurant's operations, enabling the management of various components of the system.
Key Features:

- **User Management:** Admins can add, edit, or remove users (staff, chefs, customers) from the system.
- **Menu Management:** Admins can add new menu items, update existing items, and delete items from the menu.
- **Inventory Management:** The dashboard allows tracking and updating of inventory levels for food and beverage items.
- **Order Management:** Admins can view and manage all customer orders, including their status (Pending, Prepared, Served, Completed).
- **Reporting:** Admins can generate reports on sales, inventory usage, and employee performance to make informed decisions.
- **Notifications:** Admins can view alerts for critical system updates or inventory levels.




EMPLOYEE LIST

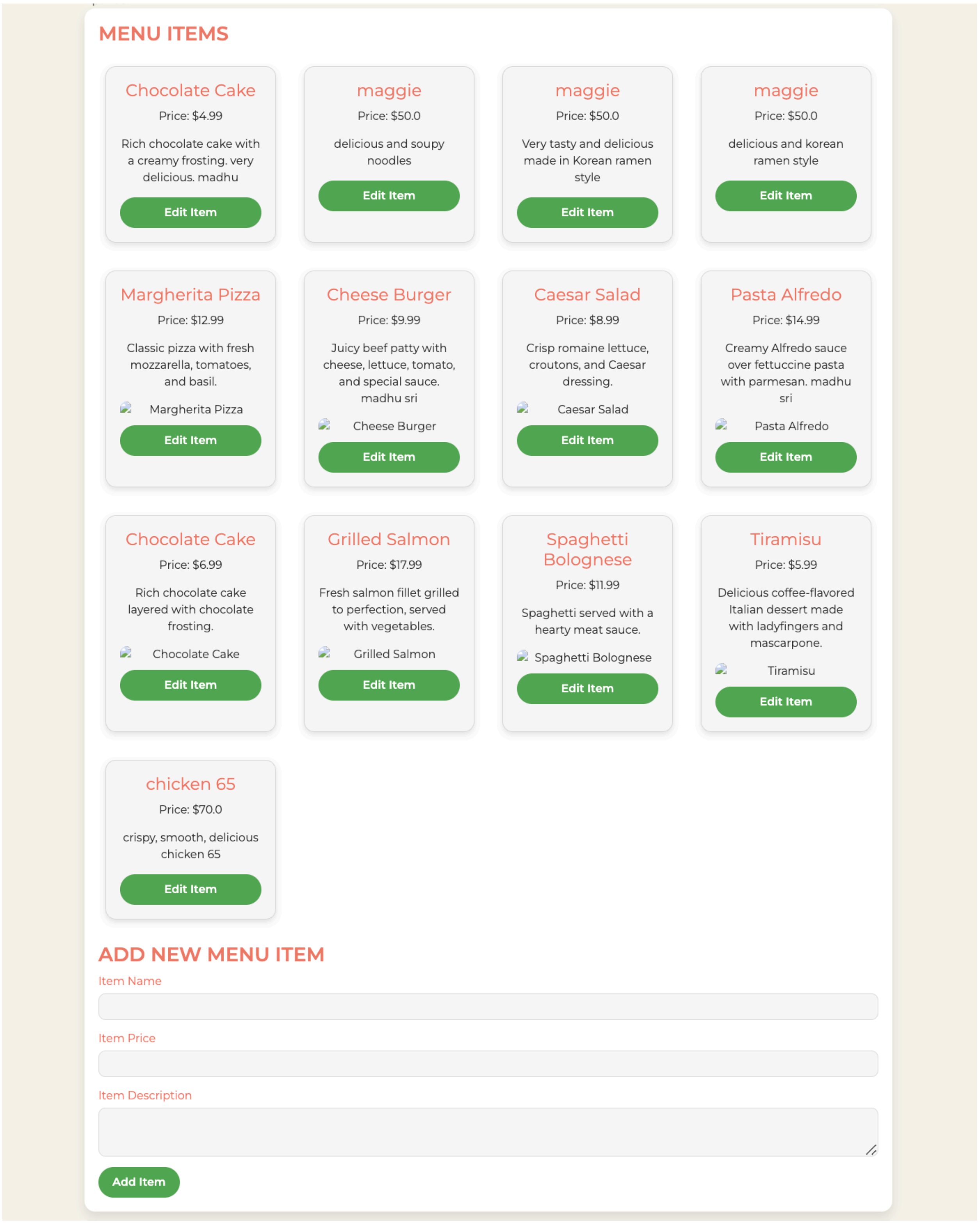
EmpID	Name	Email	Actions
4	sushmitha	sushmitha@gmail.com	Update
5	chow	sush@gmail.com	Update
123	srinu	srinu@gmail.com	Update
143	rad	sushmitha@gmail.com	Update
7	radhika	rad@gmail.com	Update
1	chef	chef@gmail.com	Update

ADD NEW EMPLOYEE

Name

Email

Employee ID



MENU ITEMS

Chocolate Cake
Price: $4.99
Rich chocolate cake with a creamy frosting. very delicious. madhu
Edit Item

maggie
Price: $50.0
delicious and soupy noodles
Edit Item

maggie
Price: $50.0
Very tasty and delicious made in Korean ramen style
Edit Item

maggie
Price: $50.0
delicious and korean ramen style
Edit Item

Margherita Pizza
Price: $12.99
Classic pizza with fresh mozzarella, tomatoes, and basil.
Margherita Pizza
Edit Item

Cheese Burger
Price: $9.99
Juicy beef patty with cheese, lettuce, tomato, and special sauce. madhu sri
Cheese Burger
Edit Item

Caesar Salad
Price: $8.99
Crisp romaine lettuce, croutons, and Caesar dressing.
Caesar Salad
Edit Item

Pasta Alfredo
Price: $14.99
Creamy Alfredo sauce over fettuccine pasta with parmesan. madhu sri
Pasta Alfredo
Edit Item

Chocolate Cake
Price: $6.99
Rich chocolate cake layered with chocolate frosting.
Chocolate Cake
Edit Item

Grilled Salmon
Price: $17.99
Fresh salmon fillet grilled to perfection, served with vegetables.
Grilled Salmon
Edit Item

Spaghetti Bolognese
Price: $11.99
Spaghetti served with a hearty meat sauce.
Spaghetti Bolognese
Edit Item

Tiramisu
Price: $5.99
Delicious coffee-flavored Italian dessert made with ladyfingers and mascarpone.
Tiramisu
Edit Item

chicken 65
Price: $70.0
crispy, smooth, delicious chicken 65
Edit Item

ADD NEW MENU ITEM

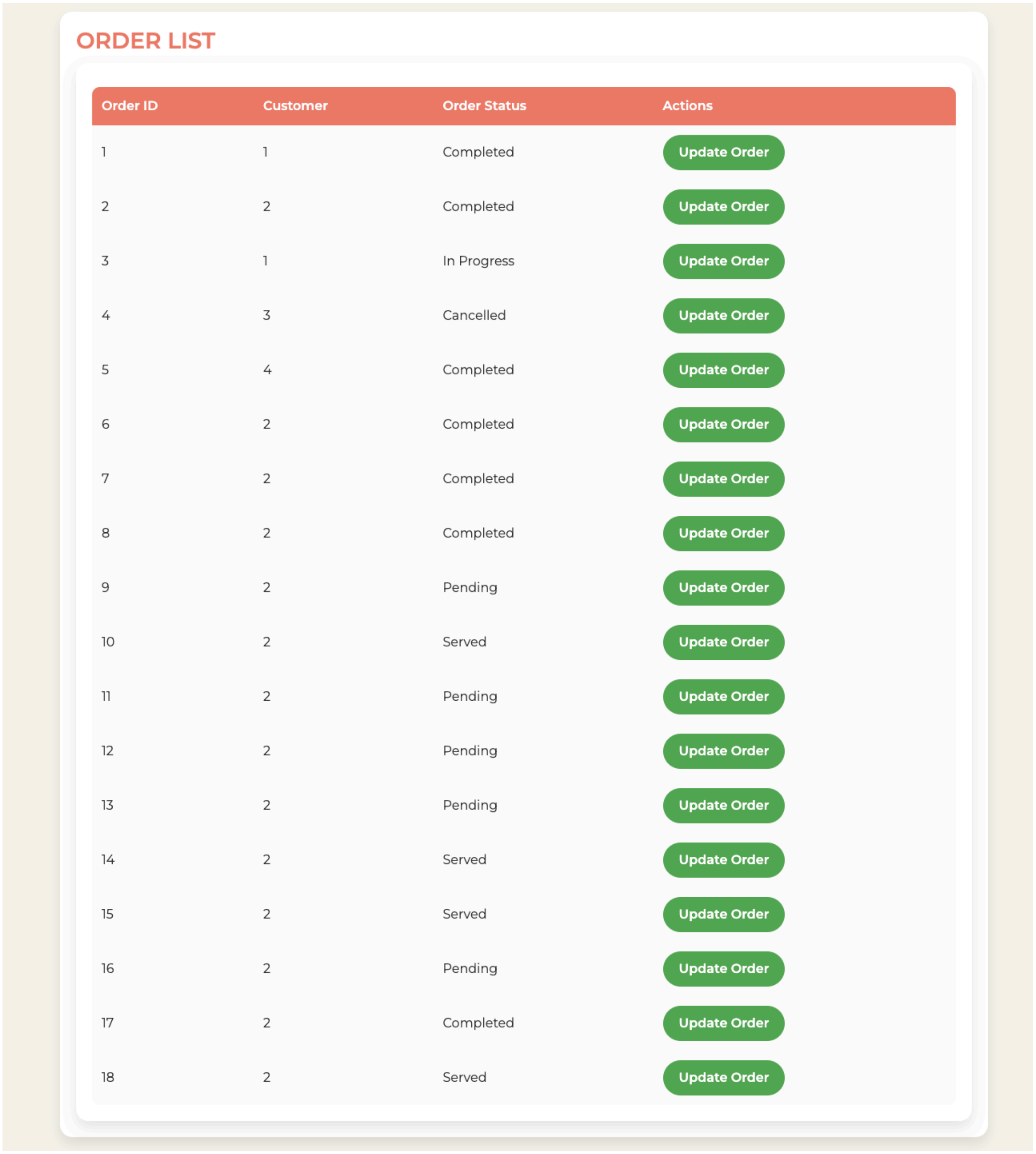
Item Name

Item Price

Item Description

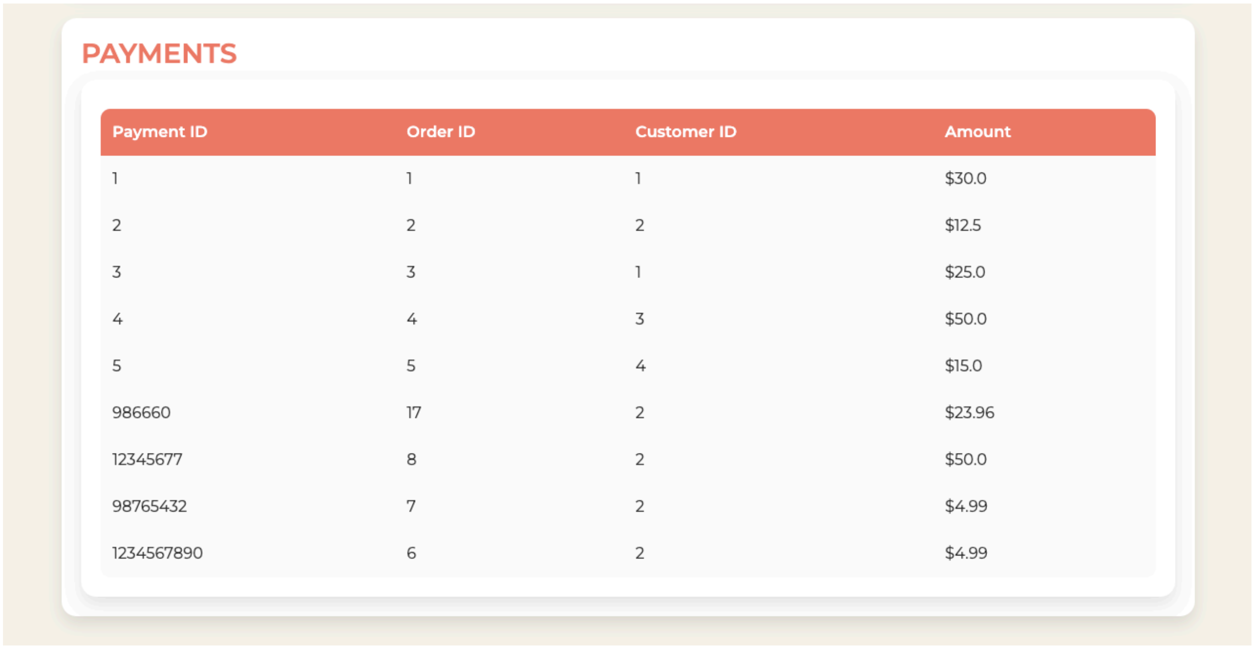
Add Item

2024MT12059



ORDER LIST

Order ID	Customer	Order Status	Actions
1	1	Completed	Update Order
2	2	Completed	Update Order
3	1	In Progress	Update Order
4	3	Cancelled	Update Order
5	4	Completed	Update Order
6	2	Completed	Update Order
7	2	Completed	Update Order
8	2	Completed	Update Order
9	2	Pending	Update Order
10	2	Served	Update Order
11	2	Pending	Update Order
12	2	Pending	Update Order
13	2	Pending	Update Order
14	2	Served	Update Order
15	2	Served	Update Order
16	2	Pending	Update Order
17	2	Completed	Update Order
18	2	Served	Update Order



PAYMENTS

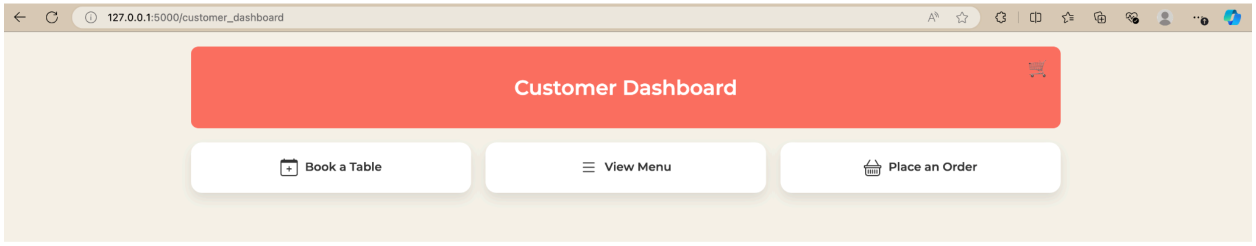
Payment ID	Order ID	Customer ID	Amount
1	1	1	$30.0
2	2	2	$12.5
3	3	1	$25.0
4	4	3	$50.0
5	5	4	$15.0
986660	17	2	$23.96
12345677	8	2	$50.0
98765432	7	2	$4.99
1234567890	6	2	$4.99

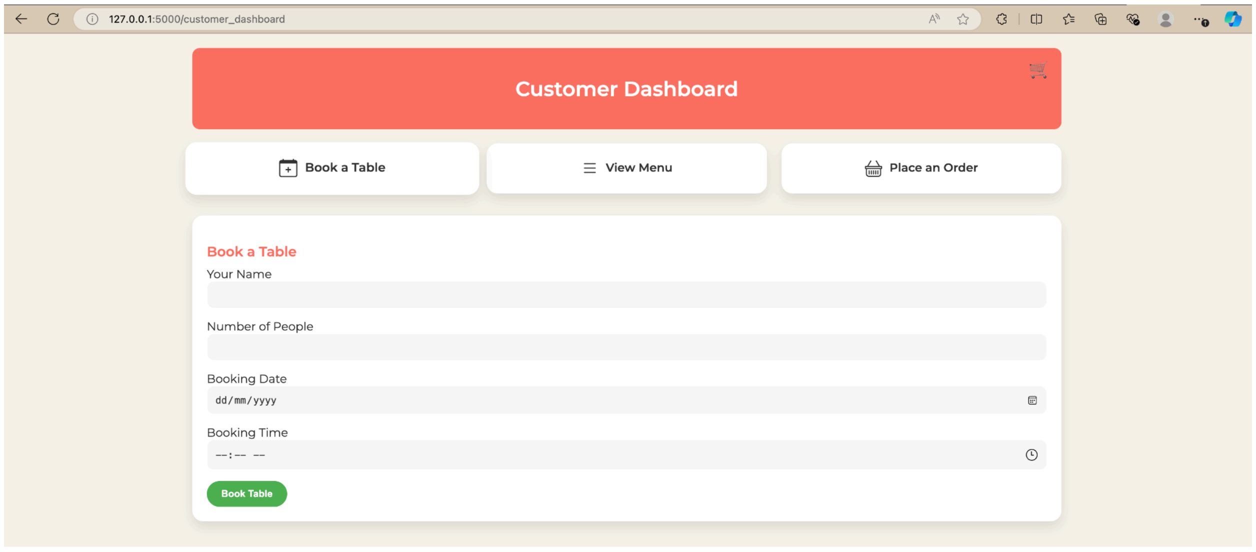
d. Customer Dashboard

The Customer Dashboard provides a user-friendly interface for customers to manage their dining experience.

Key Features:

- **Menu Browsing:** Customers can view the full menu with images, descriptions, and prices.
- **Table Booking:** Customers can book tables for dine-in experiences.
- **Order Placement:** Customers can add items to their cart and place orders for takeout or delivery.
- **Notifications:** Alerts for special offers, order status updates, and promotions.





Customer Dashboard

Book a Table | View Menu | Place an Order

Book a Table

Your Name

Number of People

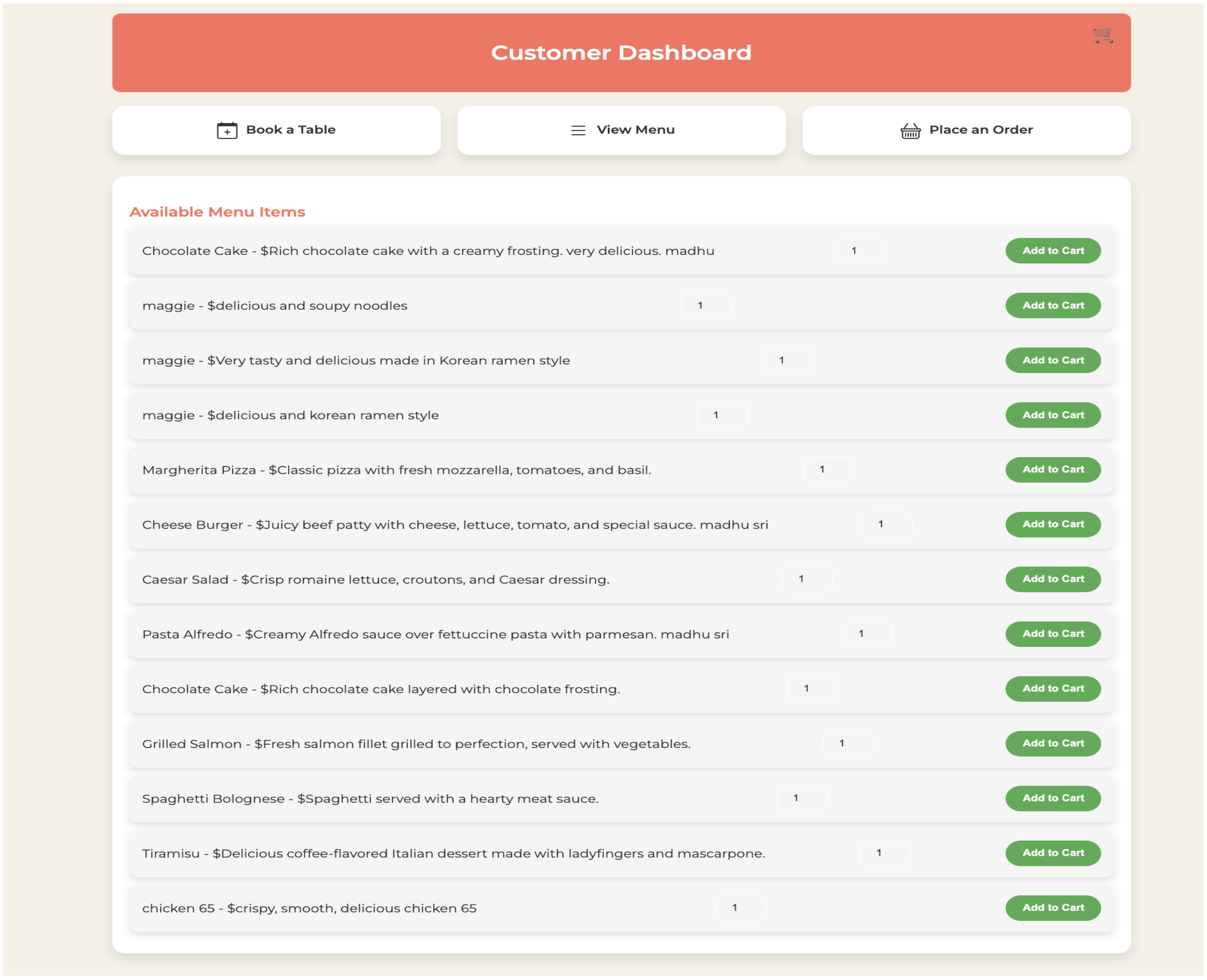
Booking Date

dd/mm/yyyy

Booking Time

--:-- --

Book Table



Customer Dashboard

Book a Table | View Menu | Place an Order

Available Menu Items

Chocolate Cake - $Rich chocolate cake with a creamy frosting. very delicious. madhu 1 Add to Cart

maggie - $delicious and soupy noodles 1 Add to Cart

maggie - $Very tasty and delicious made in Korean ramen style 1 Add to Cart

maggie - $delicious and korean ramen style 1 Add to Cart

Margherita Pizza - $Classic pizza with fresh mozzarella, tomatoes, and basil. 1 Add to Cart

Cheese Burger - $Juicy beef patty with cheese, lettuce, tomato, and special sauce. madhu sri 1 Add to Cart

Caesar Salad - $Crisp romaine lettuce, croutons, and Caesar dressing. 1 Add to Cart

Pasta Alfredo - $Creamy Alfredo sauce over fettuccine pasta with parmesan. madhu sri 1 Add to Cart

Chocolate Cake - $Rich chocolate cake layered with chocolate frosting. 1 Add to Cart

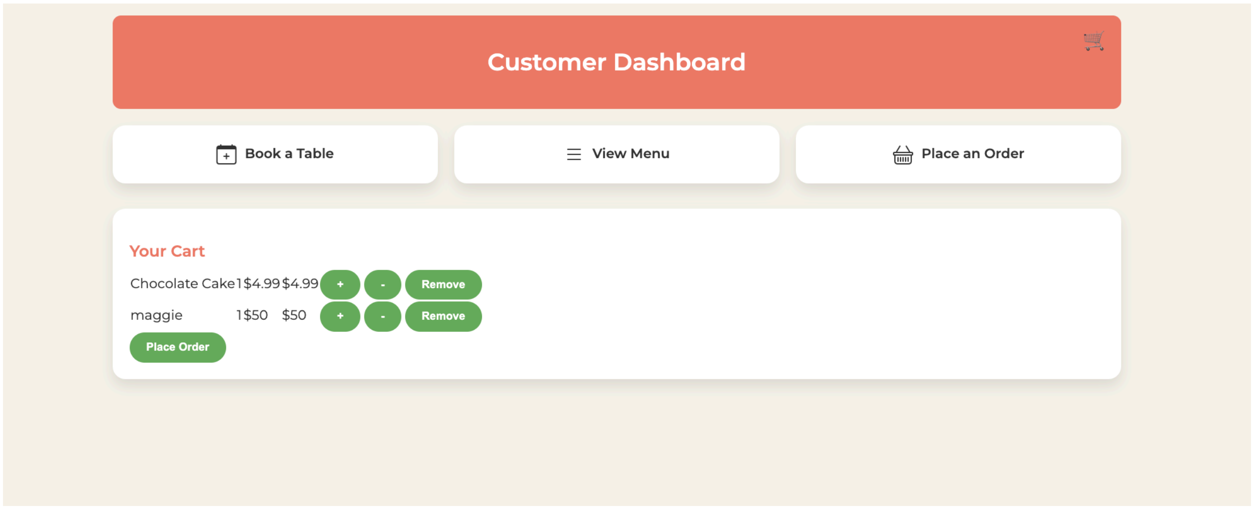
Grilled Salmon - $Fresh salmon fillet grilled to perfection, served with vegetables. 1 Add to Cart

Spaghetti Bolognese - $Spaghetti served with a hearty meat sauce. 1 Add to Cart

Tiramisu - $Delicious coffee-flavored Italian dessert made with ladyfingers and mascarpone. 1 Add to Cart

chicken 65 - $crispy, smooth, delicious chicken 65 1 Add to Cart

2024MT12059

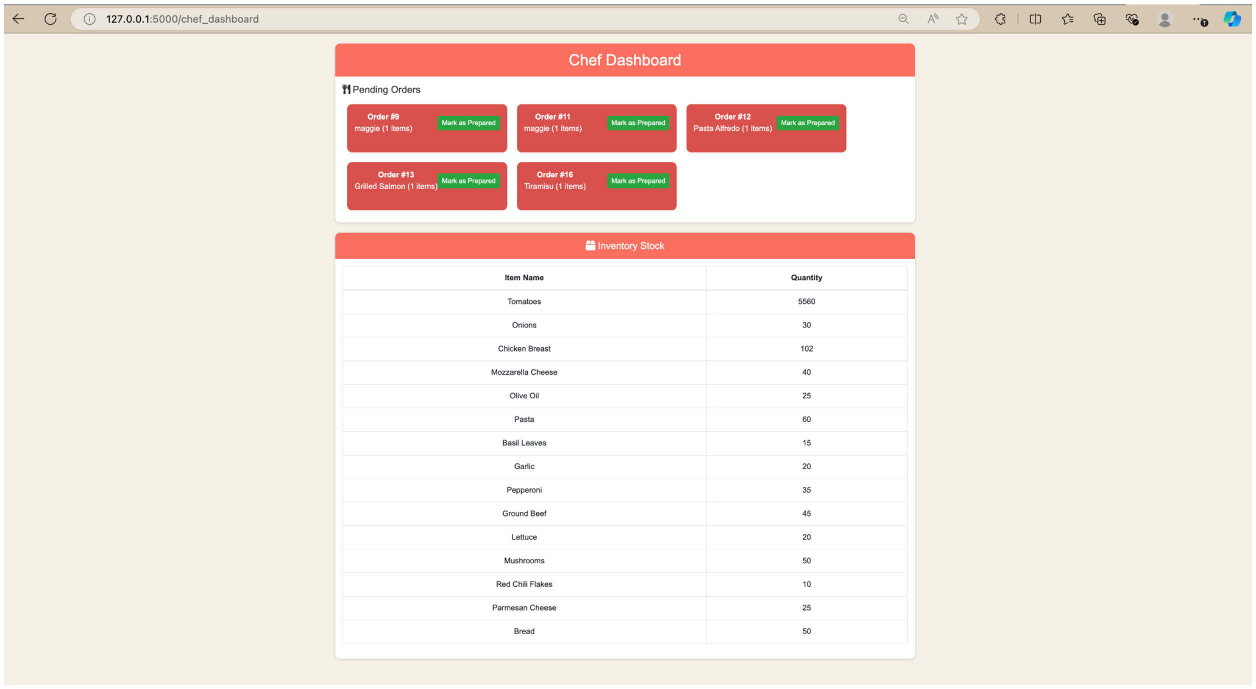


Chocolate Cake	1	$4.99	$4.99
maggie	1	$50	$50

e. Chef Dashboard

The Chef Dashboard is designed for kitchen staff to manage food preparation and order fulfillment effectively.
Key Features:

- **View Prepared Orders:** Chefs can see a list of prepared orders that need to be served.
- **Order Notifications:** Notifications for new orders and updates on order status (e.g., marked as prepared).
- **Order Management:** Chefs can update the status of orders once they are prepared, signaling to the staff that the orders are ready to be served.
- **Menu Access:** Chefs have access to the menu items to ensure they know what to prepare based on customer orders.



Order #9
maggie (1 items)

Order #11
maggie (1 items)

Order #12
Pasta Alfredo (1 items)

Order #13
Grilled Salmon (1 items)

Order #16
Tiramisu (1 items)

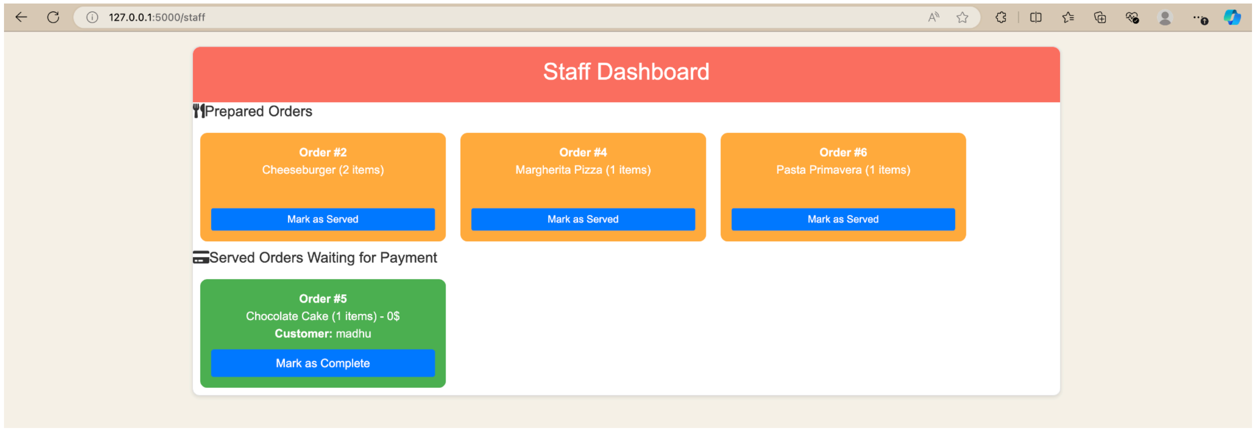
Inventory Stock

Item Name	Quantity
Tomatoes	5560
Onions	30
Chicken Breast	102
Mozzarella Cheese	40
Olive Oil	25
Pasta	60
Basil Leaves	15
Garlic	20
Pepperoni	35
Ground Beef	45
Lettuce	20
Mushrooms	50
Red Chili Flakes	10
Parmesan Cheese	25
Bread	50

f. Staff Dashboard

The Staff Dashboard supports restaurant staff in managing orders and customer interactions efficiently.
Key Features:

- **Pending Orders:** Staff can view a list of prepared orders that need to be served.
- **Order Management:** Staff can mark orders as served and completed after payment is received.
- **Customer Interaction:** Staff can access customer details and order history to assist them better.
- **Notifications:** Staff receive alerts for new orders, changes in order status, and customer requests.
- **Payment Processing:** Staff can process payments for the orders through various methods (credit card, QR code) and update the order as completed.



127.0.0.1:5000/staff

Staff Dashboard

Prepared Orders

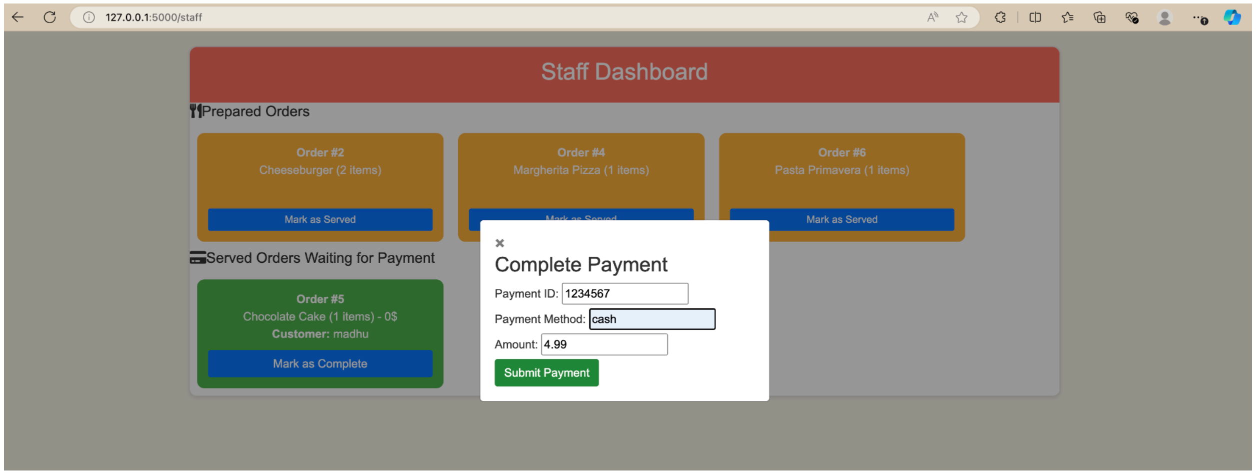
Order #2
Cheeseburger (2 items)
Mark as Served

Order #4
Margherita Pizza (1 items)
Mark as Served

Order #6
Pasta Primavera (1 items)
Mark as Served

Served Orders Waiting for Payment

Order #5
Chocolate Cake (1 items) - 0$
Customer: madhu
Mark as Complete



127.0.0.1:5000/staff

Staff Dashboard

Complete Payment

Payment ID: 1234567
Payment Method: cash
Amount: 4.99
Submit Payment

V FINAL DOC

a. Demo Video of the RMS:

https://drive.google.com/file/d/1rXTOuuDdBzYTakQrBQqLANCsanzD3_CS/view?usp=sharin

b. Complete project folder link with code :

https://drive.google.com/drive/folders/1VoGLzDXy6PZSN_KuX4rBeQ1gkZCbLoCr?usp=sharing