

Table of Contents

- **Introduction -----**
- **Uses of This Project**
- **Background**
- **DBMS for Restaurant Management System _____**
- **Addressing the Requirements**
- **Relationships and Mapping**
- **ER Diagram**
- **Mapping ER Model to Relational Model**
- **Structured Query Language (SQL)**
- **Table Creation**
- **Data Insertion**
- **Overall tables of Library Management System**
- **Summary**

❖ Introduction

This project implements a Database Management System (DBMS) designed to modernize operations in a restaurant, replacing traditional pen-and-paper methods and disjointed digital records. The system aims to resolve common challenges in areas such as order handling, inventory tracking, payment processing, and staff coordination. Ultimately, the system promotes smoother workflows and enables more responsive customer service.

Problem Statement:

Currently, many restaurants rely on:

- Manual record-keeping (e.g., handwritten orders, cash registers).
- Disconnected tools (e.g., spreadsheets for inventory, point-of-sale systems for billing).

This leads to:

- **Data Duplication:** Orders logged separately by waiters, kitchen staff, and accounts.
- **Update Delays:** Menu changes or price updates require manual synchronization across systems.
- **Reporting Challenges:** Sales analytics and inventory reports are compiled manually, risking errors.
- **No Centralized Backup:** Critical data (orders, customer history) is vulnerable to loss.
- **Security Gaps:** Sensitive payment/customer data lacks role-based access controls.

❖ Uses of This Project:

This project can be used for:

- Real-time order tracking from kitchen to billing.
- Automated inventory updates with supplier integration.
- Unified customer/payment records to enhance service.
- Digital backups and security to protect business data.
- Synchronizes operations between kitchen staff, waiters, and billing, reducing delays and confusion.
- Enables quick updates to menu items and prices across all connected systems in real-time.

By implementing a digital database solution, the system significantly improves operational accuracy, enhances efficiency, and establishes a robust platform for business expansion.

❖ Background

A **Restaurant Management System (RMS)** is software designed to efficiently handle the core operations of a restaurant. The system helps automate tasks such as order processing, table management, inventory control, billing, and staff coordination. By streamlining these processes, the RMS enhances service quality, reduces human error, and improves overall operational efficiency.

1. Database Management System: MySQL

MySQL is an open-source **relational database management system (RDBMS)** chosen for several key advantages:

- It uses Structured Query Language (SQL), which ensures it works smoothly with a wide range of software.
- The system is known for its simple installation process and ease of use.
- Its cross-platform compatibility allows it to function efficiently on various devices and operating systems.
- Being license-free, it offers a budget-friendly option for developers and businesses.
- Its open-source nature has fostered a strong user community, offering abundant resources for support and problem-solving.

2. MySQL Workbench

MySQL Workbench is an advanced and user-friendly graphical interface tool used to manage MySQL databases. It is commonly utilized by database administrators, developers, and analysts for handling a variety of database-related tasks. With its rich set of features, MySQL Workbench streamlines and simplifies the overall database management process. Below is a summary of its main functions:

❖ DBMS for Restaurant Management System

The development of the database schema for the Restaurant Management System begins with the Entity-Relationship Model (ER Model). This model is then translated into the Relational Model used in an RDBMS.

Entity-Relationship Model:

The Entity-Relationship Model (ER Model) is used to define the structure of the database for the Restaurant Management System. It includes the following components:

1. Entity:

- A thing or object in the real world that is distinguishable from other objects.
- Example: Student, Teacher, Book.

2. **Entity Set:**
 - A collection of similar types of entities.
 - Example: All students form the `Student` entity set.
3. **Attributes:**
 - Properties or characteristics of entities.
 - Example: A `Student` may have attributes like `ID`, `Name`, `Age`.
4. **Primary Key:**
 - An attribute (or set of attributes) that uniquely identifies each entity in the entity set.
5. **Relationship:**
 - Association among two or more entities.
 - Example: A `Student` *enrolls in* a `Course`.
6. **Relationship Set:**
 - A set of relationships of the same type.
 - Example: All enrollments between students and courses.
7. **Cardinality:**
 - Describes how many entities can be related (1:1, 1:N, N:M).
 - Example: A student *enrolls in* many courses (1:N).
8. **ER Diagram:**
 - A graphical representation of entities, attributes, and relationships using symbols:
 - Rectangle: Entity
 - Ellipse: Attribute
 - Diamond: Relationship
 - Line: Connection

❖ Addressing the Requirements

The initial step in building the ER Model is identifying the key entities relevant to restaurant operations.

Entities of the Restaurant Management System:

An entity represents a real-world object that exists independently. It could be a person, location, item, or concept. The primary entities in a Restaurant Management System include:

Entities:

- Customer
- Kitchen Staff
- Payment Processor
- Waiter
- Orders
- Supplier

Attributes:

Entity	Attributes
Customer	CustomerId (Primary Key) C_Name C_Email C_Phone
Kitchen Staff	KStaffId (Primary Key) KS_Name KS_Role KS_Salary KS_Phone
Payment Processor	PaymentId (Primary Key) OrderId Amount PType PDate
Waiter	WaiterId (Primary Key) W_Name W_Salary W_Phone WHiringDate
Orders	OrderId (Primary Key) CustomerId Waiter_Id OrderTime TableNo OrderStatus OrderTotal

Supplier	SupplierId (Primary Key) S_Name S_Phone S_Address S_Email
----------	---

❖ Relationships and Mapping:

1. Waiter and Customer

Relationship: One waiter can serve multiple customers.

Types: 1: N

2. Payment Processor and Order

Relationship: Each order payment uses one payment processor but one processor can handle many orders.

Types: 1: N

3. Waiter and Orders

Relationship: One waiter can take many orders, but each order is taken by one waiter.

Types: 1: N

4. Orders and Kitchen Staff

Relationship: Many orders can be prepared by one kitchen staff member but each order is assigned to one staff member.

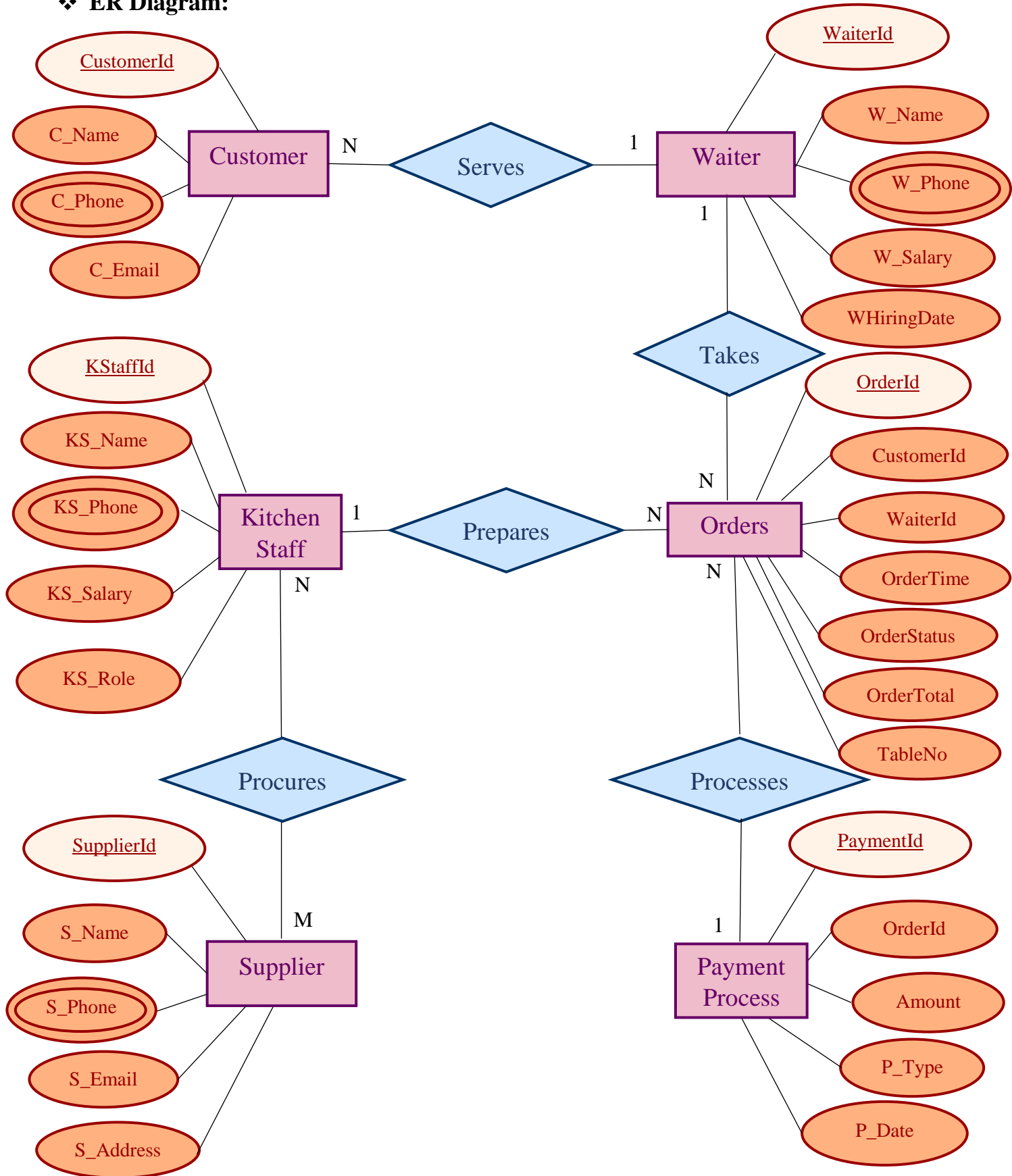
Types: N:1

5. Kitchen Staff and Supplier

Relationship: One kitchen staff can place orders to multiple suppliers for ingredients, one supplier may serve multiple kitchen staff.

Types: N:M

❖ ER Diagram:



❖ Mapping ER Model to Relational Model: Final Relation Schemas

Customer

<u>CustomerId</u>	C_Name	C_Email	C_Phone

Kitchen Staff

<u>KStaffId</u>	KS_Name	KS_Role	KS_Salary	KS_Phone

Payment Processor

<u>PaymentId</u>	OrderId	Amount	P_Type	P_Date

Waiter

<u>WaiterId</u>	W_Name	W_Salary	W_Phone	WHiringDate

Orders

<u>OrderId</u>	CustomerId	WaiterId	OrderTime	TableNo	OrderStatus	OrderTotal

Supplier

<u>SupplierId</u>	S_Name	S_Phone	S_Address	S_Email

❖ Structured Query Language (SQL):

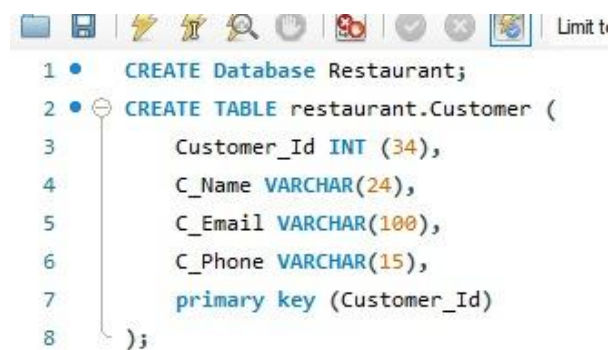
We have used the MySQL implementation of SQL for our project. The main SQL commands that we used are:

DDL- Data Definition Language: In DDL, we have used CREATE commands to library database and tables.

DML-Data Manipulation Language: In DDL, we have used INSERT, UPDATE, DELETE commands.

❖ Table Creation:

- Customer Table:



```
1 • CREATE Database Restaurant;
2 • CREATE TABLE restaurant.Customer (
3     Customer_Id INT (34),
4     C_Name VARCHAR(24),
5     C_Email VARCHAR(100),
6     C_Phone VARCHAR(15),
7     primary key (Customer_Id)
8 );
```

Table: customer

Columns:

<u>Customer_Id</u>	int PK
C_Name	varchar(24)
C_Email	varchar(100)
C_Phone	varchar(15)

Field Types					
#	Field	Schema	Table	Type	Character Set
1	Customer_Id	restaurant	customer	INT	binary
2	C_Name	restaurant	customer	VARCHAR	utf8mb4
3	C_Email	restaurant	customer	VARCHAR	utf8mb4
4	C_Phone	restaurant	customer	VARCHAR	utf8mb4

Fig-1: Source Code for Customer Table

○ Kitchen Staff Table:

```
9 • ○ CREATE TABLE restaurant.KitchenStaff (  
10     KStaffId INT (23),  
11     KS_Name VARCHAR(100),  
12     KS_Role VARCHAR(50),  
13     KS_Salary DECIMAL(10, 2),  
14     KS_Phone VARCHAR(20),  
15     primary key (KStaffId)  
16 );
```

Table: kitchenstaff

Columns:

<u>KStaffId</u>	int PK
KS_Name	varchar(100)
KS_Role	varchar(50)
KS_Salary	decimal(10,2)
KS_Phone	varchar(20)

Field Types						
#	Field	Schema	Table	Type	Character Set	Disj
1	KStaffId	restaurant	kitchenstaff	INT	binary	
2	KS_Name	restaurant	kitchenstaff	VARCHAR	utf8mb4	
3	KS_Role	restaurant	kitchenstaff	VARCHAR	utf8mb4	
4	KS_Salary	restaurant	kitchenstaff	DECIMAL	binary	
5	KS_Phone	restaurant	kitchenstaff	VARCHAR	utf8mb4	

Fig - 2: Source Code for Kitchen Staff Table

- Waiter Table:

```

26 ● ○ CREATE TABLE restaurant.Waiter (
27     Waiter_Id INT (34),
28     W_Name VARCHAR(43),
29     W_Salary DECIMAL(10, 2),
30     W_Phone VARCHAR(20),
31     WHiringDate DATE,
32     PRIMARY KEY ( Waiter_Id )
33 );

```

Table: waiter

Columns:

<u>Waiter_Id</u>	int PK
W_Name	varchar(43)
W_Salary	decimal(10,2)
W_Phone	varchar(20)
WHiringDate	date

Field Types						
#	Field	Schema	Table	Type	Character Set	
1	Waiter_Id	restaurant	waiter	INT	binary	
2	W_Name	restaurant	waiter	VARCHAR	utf8mb4	
3	W_Salary	restaurant	waiter	DECIMAL	binary	
4	W_Phone	restaurant	waiter	VARCHAR	utf8mb4	
5	WHiringDate	restaurant	waiter	DATE	binary	

Fig - 3: Source Code for Waiter Table

- Payment Processor Table:

```

18 • CREATE TABLE restaurant.Payment_Processor (
19     PaymentId INT (45),
20     OrderId INT (45),
21     Amount DECIMAL(10, 2),
22     PType VARCHAR(50),
23     PDate DATE,
24     primary key (PaymentId)
25 );

```

Table: **payment_processor**

Columns:

PaymentId	int PK
OrderId	int
Amount	decimal(10,2)
PType	varchar(50)
PDate	date

Field Types					
#	Field	Schema	Table	Type	Character Set
1	PaymentId	restaurant	payment_processor	INT	binary
2	OrderId	restaurant	payment_processor	INT	binary
3	Amount	restaurant	payment_processor	DECIMAL	binary
4	PType	restaurant	payment_processor	VARCHAR	utf8mb4
5	PDate	restaurant	payment_processor	DATE	binary

Fig - 4: Source Code for Payment Processor Table

○ Orders Table:

```

34 • CREATE TABLE restaurant.Orders (
35     OrderId INT (34),
36     CustomerId INT (34),
37     Waiter_Id INT (34),
38     OrderTime DATETIME,
39     TableNo INT (2),
40     OrderStatus VARCHAR(50),
41     OrderTotal INT (10),
42     PRIMARY KEY ( OrderId)
43 );

```

Table: orders

Columns:

<u>OrderId</u>	int PK
CustomerId	int
Waiter_Id	int
OrderTime	datetime
TableNo	int
OrderStatus	varchar(50)
OrderTotal	int

Field Types					
#	Field	Schema	Table	Type	Character Set
1	OrderId	restaurant	orders	INT	binary
2	CustomerId	restaurant	orders	INT	binary
3	Waiter_Id	restaurant	orders	INT	binary
4	OrderTime	restaurant	orders	DATETIME	binary
5	TableNo	restaurant	orders	INT	binary

Fig - 5: Source Code for Orders Table

○ Suppliers Table:

```
CREATE TABLE restaurant.Supplier (  
    SupplierId INT (45),  
    S_Name VARCHAR(56),  
    S_Phone VARCHAR(20),  
    S_Address VARCHAR(25),  
    S_Email VARCHAR(10),  
    PRIMARY KEY (SupplierId)  
);
```

Table: supplier

Columns:

SupplierId	int
S_Name	varchar(56)
S_Phone	varchar(20)
S_Address	varchar(25)
S_Email	varchar(100)

Field Types					
#	Field	Schema	Table	Type	Character Set
1	SupplierId	restaurant	supplier	INT	binary
2	S_Name	restaurant	supplier	VARCHAR	utf8mb4
3	S_Phone	restaurant	supplier	VARCHAR	utf8mb4
4	S_Address	restaurant	supplier	VARCHAR	utf8mb4
5	S_Email	restaurant	supplier	VARCHAR	utf8mb4

Fig – 6: Source Code for Suppliers Table

❖ **Overall Tables of Library Database:**

Customer Table
Kitchen Staff Table
Payment Processor Table
Waiter Table
Orders Table
Supplier Table
Total 6 Tables

