Restaurant Order and Billing Management System

PAYAL PATIL ___ 2542035

SYBSCIT

Established – 1961 Subject: OS DBMS

R. K. TALREJA COLLEGE

SEVA SADAN'SOFARTS, SCIENCE &

COMMERCE ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr./Ms. Payal Vikas Patil of S.Y. Information Technology (SYIT) Roll No. 2542035 has satisfactorily completed the Open Source DataBase Management System Mini Project entitled Restaurant Order and Billing Management System During the academic year 2025 – 2026, as a part of the practical requirement. The project work is found to be satisfactory and is approved for submission.

PROF. INCHARGE **HEAD OF DEPT**

INDEX

1. INTRODUCTION

A database system is a structured way of storing, managing, and retrieving data efficiently. It allows large amounts of information to be organized into tables so that data can be easily accessed, updated, and maintained. Database systems reduce manual work, avoid data duplication, and ensure that information remains accurate and consistent. In today's digital environment, database systems are essential for managing day-to-day operations in many organizations such as banks, hospitals, colleges, and restaurants.

The Restaurant Order and Billing Management System is a database-based system designed to manage restaurant operations including customer information, menu details, order processing, billing, and payment records. In many small and medium-scale restaurants, data is still maintained manually using paper records or basic tools. This manual approach often leads to several problems such as duplicate customer entries, incorrect billing calculations, missing order details, slow retrieval of information, and difficulty in tracking payments. These issues reduce efficiency and can negatively affect customer service.

This project addresses these problem areas by providing a centralized relational database system. All restaurant-related data is stored in a structured format using multiple related tables. Customer details, menu items, orders, billing information, and payments are properly connected using relational database concepts. The system uses database constraints such as primary keys, foreign keys, unique constraints, and not-null conditions to prevent duplicate records and maintain data integrity. Transaction management is also implemented to ensure that important operations such as order booking and bill generation are completed fully and correctly, without partial or inconsistent updates.

MySQL is used as the database management system for this project because it is an open-source, reliable, and widely used relational DBMS. Being open source, MySQL is free to use and does not require expensive licenses, making it suitable for academic projects and small-scale applications. MySQL supports standard SQL commands, strong data integrity constraints, transaction control, and security features. It is also easy to install, simple to learn, and compatible with different platforms such as Windows and Linux.

Another reason for choosing MySQL is its ability to support real-world database concepts such as normalization, joins, views, and stored procedures. These features help students understand how database systems work in practical applications. MySQL's performance and stability make it a popular choice for managing structured data efficiently.

In conclusion, the Restaurant Order and Billing Management System demonstrates how a database system can be used to automate restaurant operations, reduce errors, and improve data management. By using MySQL as an open-source solution, the project provides a simple, cost-effective, and reliable way to manage restaurant data while giving practical exposure to core DBMS concepts.

2. PROBLEM DEFINITION

Many restaurants still depend on manual or unstructured methods to manage customer details, menu information, orders, billing, and payments. These methods include maintaining paper records or basic digital files, which are inefficient and difficult to manage as data volume increases. Such systems require more manual effort and are highly prone to human errors. Due to the absence of a proper database-driven system, restaurants face several challenges, including:

- **Data Redundancy:** The same customer, order, or menu data is entered multiple times, increasing storage and confusion.
- **Data Inconsistency:** Updates made in one place are not reflected in other records, leading to conflicting information.
- **Duplicate Records:** Multiple entries for the same customer or order may exist due to lack of validation.
- **Incorrect Billing Calculations:** Manual billing increases the chances of calculation errors and wrong totals.
- **Partial Data Updates:** Failure during order or billing updates may result in incomplete or invalid records.
- **Poor Data Integrity:** Orders may be stored without proper customer or menu references.
- **Lack of Data Security:** Sensitive information is not protected against unauthorized access.
- **Difficulty in Data Tracking:** Tracking customer orders, payments, and bills becomes time-consuming.
- **Slow Data Retrieval:** Searching, updating, and reporting information takes more time.
- **Limited Reporting Capability:** Manual systems cannot easily generate accurate reports or summaries.
- **Risk of Data Loss:** Paper-based or unstructured digital data can be lost or damaged easily.
- **Scalability Issues:** Manual systems cannot handle increasing data efficiently as the business grows.

These problems highlight the need for a **database-driven restaurant management system**. A relational database using MySQL can reduce redundancy, maintain consistency, improve security, and ensure accurate and reliable data handling through proper constraints and transaction control.

3. OBJECTIVES OF THE PROJECT

The main objectives of the Restaurant Order and Billing Management System are as follows:

- To design and develop a structured database system for managing restaurant operations using MySQL.
- To store and manage customer, menu, order, billing, and payment data efficiently.
- To implement SQL queries for inserting, updating, deleting, and retrieving restaurant data.
- To ensure data accuracy and consistency by applying database constraints such as primary keys, foreign keys, unique constraints, and not-null conditions.
- To prevent duplicate and conflicting records in the database.
- To maintain data integrity during order booking and billing using transaction management.
- To retrieve meaningful information from the database using SQL queries and joins.
- To improve understanding of relational database concepts and their practical implementation.
- To gain hands-on experience with MySQL as an open-source database management system.

4. SCOPE OF THE PROJECT

The Restaurant Order and Billing Management System is designed to manage basic restaurant operations using a database-driven approach. The system can be used in small-scale restaurants, cafés, and food outlets where customer data, menu items, orders, billing, and payments need to be handled efficiently and accurately.

Users of the System

- **Administrator:** Responsible for managing menu items, maintaining the database, and controlling user access.
- **Restaurant Staff:** Handles order placement, bill generation, and payment recording.
- **Students:** Use the system for academic learning and practical understanding of database concepts.
- **Business Users:** Small restaurant owners can use the system to manage daily restaurant operations in a structured manner.

Areas of Use

- **Educational Use:**
  The system is suitable for students to learn database design, SQL queries, constraints, and transaction management as part of academic projects.
- **Business Use:**
  The system can be used by small restaurants to manage orders and billing efficiently while reducing manual work and errors.
- **Administrative Use:**
  Administrators can maintain structured records of customers, menu items, and billing data for better control and organization.

Academic Limitations

- The system is designed mainly for academic and small-scale use.
- Advanced features such as online ordering, live customer interfaces, and external payment gateways are not included.
- The focus is on database design and SQL operations rather than full-scale application development.

Overall, the project scope is limited to demonstrating core database concepts while providing a practical and understandable solution for restaurant order and billing management.

5. REQUIREMENT SPECIFICATION

5.1 Hardware Requirements

The following hardware requirements are necessary to run the Restaurant Order and Billing Management System:

- **Computer / Laptop:**
  A standard desktop computer or laptop capable of running MySQL.

- **Minimum RAM:**
  At least **4 GB RAM** is required for smooth operation of the database system and MySQL Workbench.

- **Storage:**
  Minimum 10 GB free disk space to store the database files, MySQL software, and related project data.

5.2 Software Requirements

The following software tools are required for the development and execution of the Restaurant Order and Billing Management System:

Software	Purpose
MySQL Server	Used to create, store, manage, and secure the restaurant database
MySQL Workbench	Provides a graphical interface for writing, executing, and managing SQL queries
Operating System (Windows / Linux)	Platform required to install and run MySQL Server and related tools
SQL	Query language used to perform database operations such as data insertion, updating, deletion, and retrieval
Web Browser (Optional)	Used for accessing online MySQL documentation and learning resources

6. SYSTEM DESIGN

6.1 System Architecture (Conceptual)

The Restaurant Order and Billing Management System follows a simple database-centered architecture where users interact with the system through SQL queries executed on a MySQL database. The architecture is designed to ensure proper data flow, data integrity, and reliable transaction handling.

In this system, the user such as an administrator or restaurant staff interacts with the database using SQL commands through MySQL Workbench or a similar query interface. The user performs operations like adding customers, placing orders, updating menu items, generating bills, and recording payments by executing SQL queries.

The MySQL Server acts as the core component of the system. It is responsible for storing all restaurant-related data in structured tables such as Customer, Menu, Orders, Billing, and Payment. The server enforces database constraints like primary keys, foreign keys, unique constraints, and not-null conditions to maintain data integrity. It also manages transactions to ensure that multiple related operations, such as order booking and bill generation, are executed safely and completely.

When a user executes an SQL query, it is first sent to the MySQL Server for processing. The server validates the query syntax, checks user privileges, and verifies constraint rules. If the query involves multiple operations, transaction control ensures that either all operations are completed successfully or none are applied. After execution, the MySQL Server returns the result to the user in the form of updated tables or query output.
Overall, this conceptual architecture ensures a smooth interaction between the user and the database system. It provides a reliable and efficient way to manage restaurant data while demonstrating practical use of DBMS concepts such as query execution, constraints, and transaction management.

7. DATABASE DESIGN

The database design of the Restaurant Order and Billing Management System is based on the relational model. The database is divided into multiple related tables to store restaurant data in a structured and organized manner. Each table represents a real-world entity, and relationships between tables are maintained using foreign keys to ensure data integrity and consistency.

7.1 Entity Description

Customer Table
The Customer table stores information related to restaurant customers. It contains details such as customer name and contact number. Each customer is uniquely identified using a customer ID. This table helps in tracking customer orders and billing history while preventing duplicate customer entries.

Menu Table
The Menu table stores information about food items available in the restaurant. It includes item names and their prices. Each menu item is uniquely identified using a menu ID. This table helps maintain consistent pricing and avoids duplicate menu entries.

Orders Table
The Orders table stores details of customer orders. Each order is linked to a specific customer using a foreign key. This table records the order date and helps track multiple orders placed by different customers.

Order_Details Table
The Order_Details table stores detailed information about items included in each order. It links orders with menu items and records the quantity of each item ordered. This table supports a many-to-many relationship between orders and menu items.

Billing Table
The Billing table stores billing information for each order. It contains the total bill amount and billing date. Each bill is associated with a specific order, ensuring accurate bill generation.

Payment Table
The Payment table stores payment-related details such as payment mode and payment status. It is linked to the billing table and helps track whether a bill has been paid or not.

7.2 Table Structure

Customer Table

Attribute	Data Type
customer_id	INT (Primary Key, Auto Increment)
customer_name	VARCHAR**(50)**
contact_no	VARCHAR**(15)**

Menu Table

Attribute	Data Type
menu_id	INT (Primary Key, Auto Increment)
item_name	VARCHAR**(50)**
price	DECIMAL(8,2)

Orders Table

Attribute	Data Type
order_id	INT (Primary Key, Auto Increment)
customer_id	INT (Foreign Key)
order_date	DATETIME

Order_Details Table

Attribute	Data Type
order_detail_id	INT (Primary Key, Auto Increment)
order_id	INT (Foreign Key)
menu_id	INT (Foreign Key)
quantity	INT

Billing Table

Attribute	Data Type
bill_id	INT (Primary Key, Auto Increment)
order_id	INT (Foreign Key)
total_amount	DECIMAL(10,2)
bill_date	DATETIME

Payment Table

Attribute	Data Type
payment_id	INT (Primary Key, Auto Increment)
bill_id	INT (Foreign Key)
payment_mode	VARCHAR(20)
payment_status	VARCHAR(20)

7.3 Constraints Used

Constraints are used in the Restaurant Order and Billing Management System to maintain data accuracy, consistency, and integrity. They ensure that only valid and meaningful data is stored in the database. The following constraints are used extensively in this project:

PRIMARY KEY

The PRIMARY KEY constraint is used to uniquely identify each record in a table. In this project, primary keys are used in all major tables such as Customer, Menu, Orders, Order_Details, Billing, and Payment. Each table has a unique identifier like customer_id, menu_id, or order_id. This ensures that every record can be uniquely accessed and prevents duplicate entries within the same table.

FOREIGN KEY

The FOREIGN KEY constraint is used to establish relationships between different tables. In this project, foreign keys are used to link customer records with orders, orders with order details, orders with billing information, and billing with payment records. This constraint ensures that records in one table always refer to valid records in another table. It prevents invalid or orphan records and maintains logical connections between restaurant data.

NOT NULL

The NOT NULL constraint is used to ensure that important fields always contain values. In this project, attributes such as customer name, contact number, menu item name, price, and quantity are defined as NOT NULL. This ensures that incomplete or meaningless records are not stored in the database and that essential information is always available

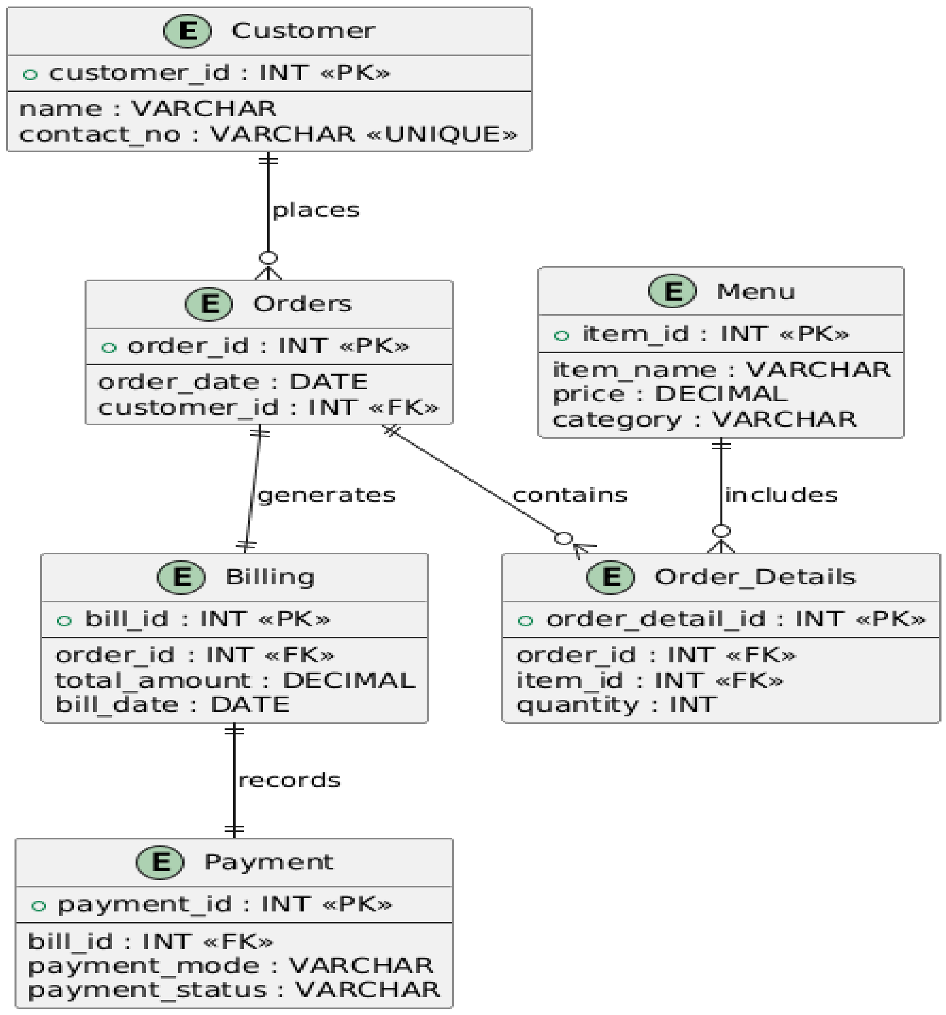
UNIQUE

The UNIQUE constraint is used to prevent duplicate entries in specific fields. In this project, it is mainly used for customer contact numbers and menu item names. This ensures that the same customer or menu item is not stored multiple times in the database, reducing redundancy and confusion.

8. UML DIAGRAMS

In this project, various UML diagrams such as ER Diagram, Use Case Diagram, Activity Diagram, and Sequence Diagram are used. Each diagram explains a different aspect of the system including database structure, user interaction, workflow of operations, and query execution flow.

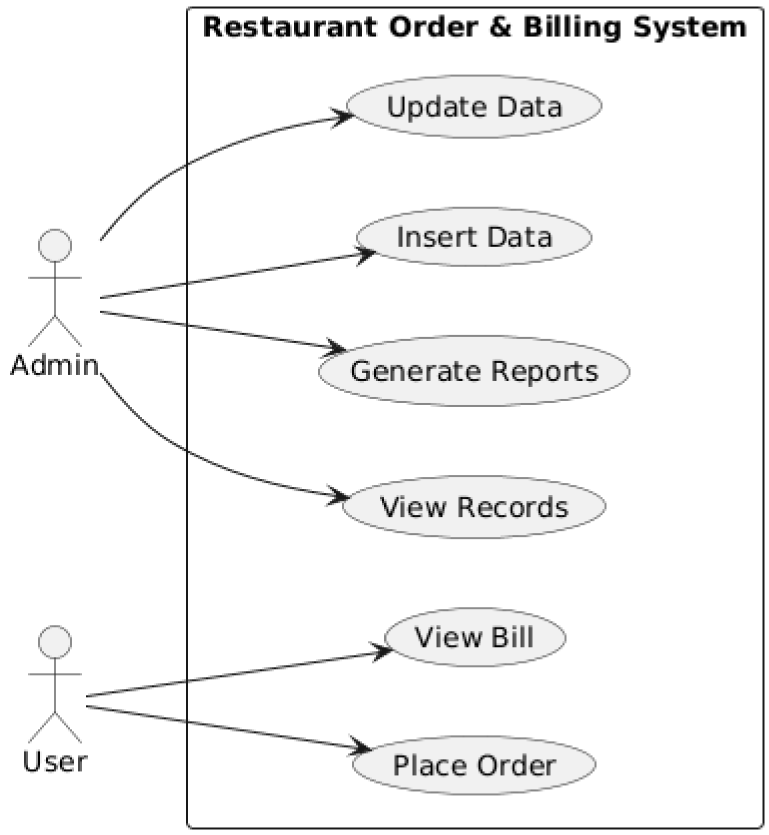
8.1 ER Diagram



- Entities: Customer, Menu, Orders, Order_Details, Billing, Payment
- Relationships between entities using foreign keys

The ER diagram represents the entities used in the Restaurant Order and Billing Management System and shows the relationships between them. It explains how customer, order, menu, billing, and payment data are connected using primary and foreign keys

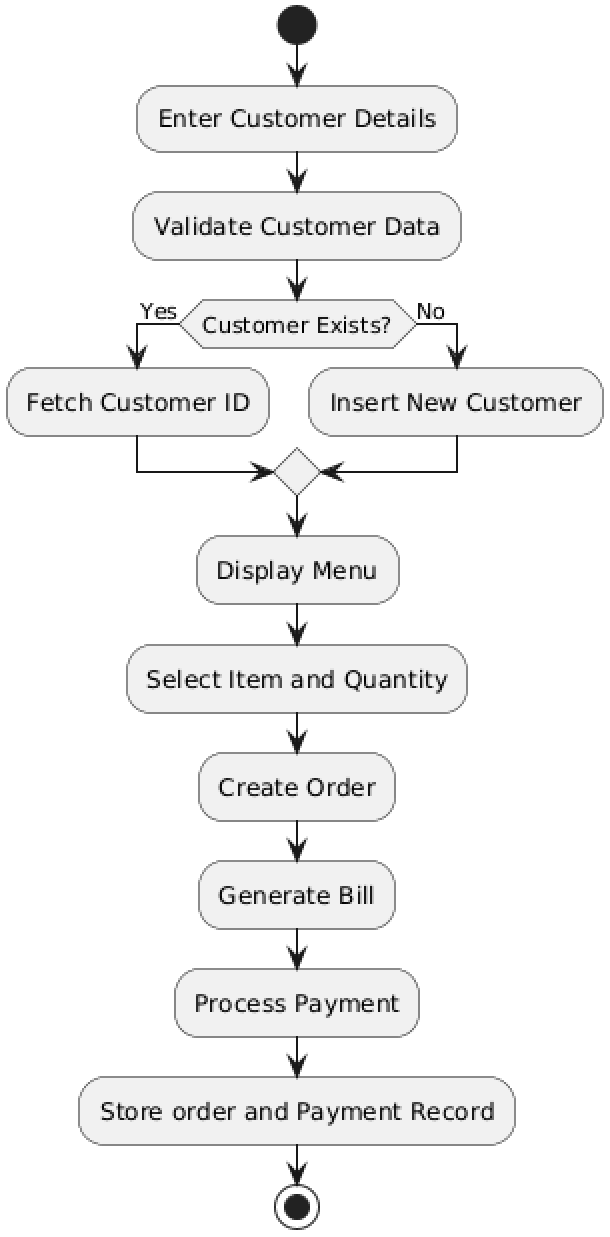
8.2 Use Case Diagram



- The Use Case Diagram represents the interaction between the actors and the Restaurant Order & Billing System.
- It identifies two main actors: Admin and User.

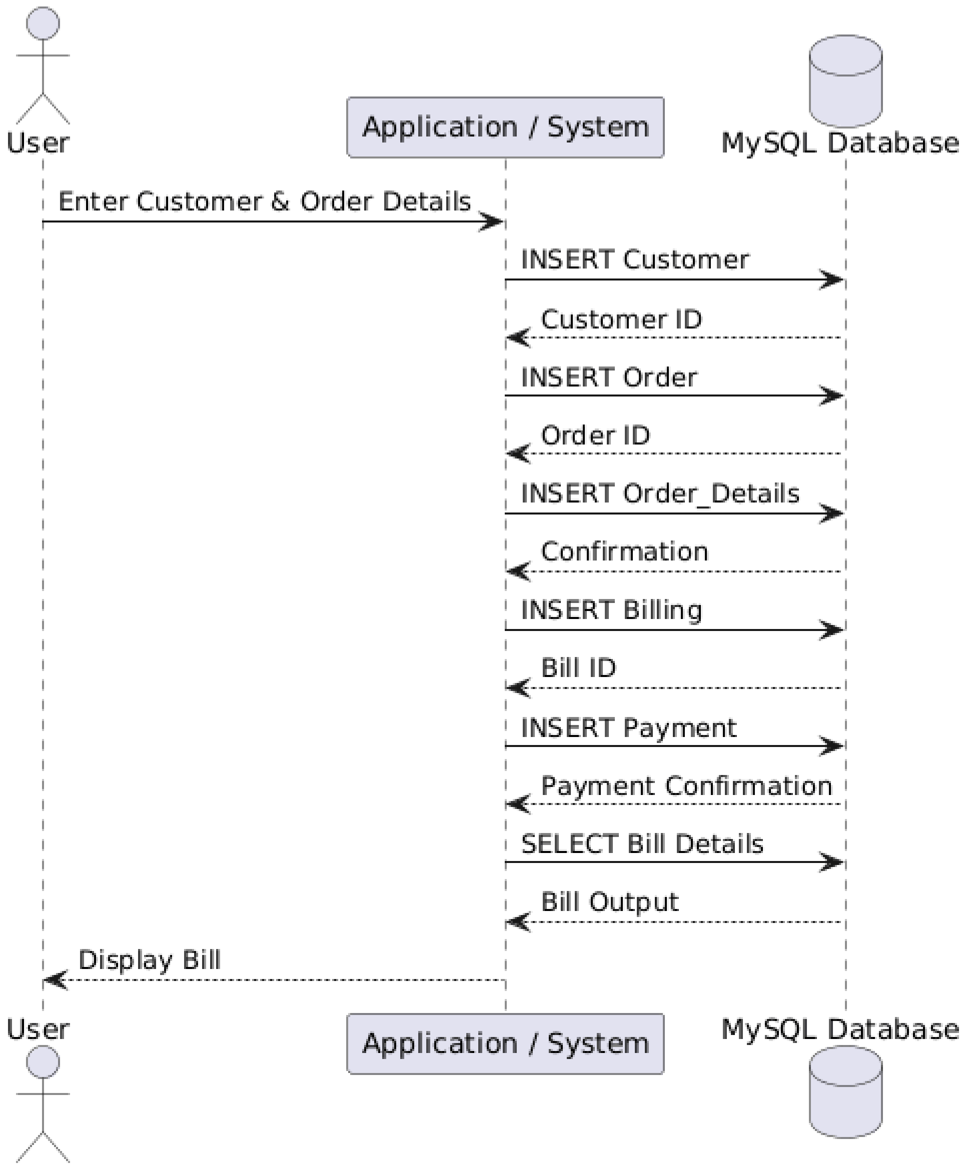
The Use Case Diagram shows the interaction between users and the system. It represents the different operations such as inserting, updating, viewing data, and generating reports performed by the admin and staff.

8.3 Activity Diagram



The Activity Diagram represents the workflow of database operations in the system. It shows the step-by-step process from order placement to billing and payment.

8.4 Sequence Diagram



The Sequence Diagram illustrates the query execution flow in the system. It shows how user requests are processed by the system and how the database responds during order and billing operations.

9. SQL IMPLEMENTATION
Code:

```
CREATE DATABASE Restaurant_System;
USE Restaurant_System;

CREATE TABLE Customer (
   customer_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_name VARCHAR(50) NOT NULL,
   contact_no VARCHAR(15) NOT NULL UNIQUE
);

CREATE TABLE Menu (
   menu_id INT AUTO_INCREMENT PRIMARY KEY,
   item_name VARCHAR(50) NOT NULL UNIQUE,
   price DECIMAL(8,2) NOT NULL
);

CREATE TABLE Orders (
   order_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_id INT NOT NULL,
   order_date DATE NOT NULL,
   FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE Order_Details (
   order_detail_id INT AUTO_INCREMENT PRIMARY KEY,
   order_id INT NOT NULL,
   menu_id INT NOT NULL,
   quantity INT NOT NULL,
   FOREIGN KEY (order_id) REFERENCES Orders(order_id),
   FOREIGN KEY (menu_id) REFERENCES Menu(menu_id)
);

CREATE TABLE Billing (
   bill_id INT AUTO_INCREMENT PRIMARY KEY,
   order_id INT NOT NULL,
   total_amount DECIMAL(10,2) NOT NULL,
   bill_date DATE NOT NULL,
   FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);

CREATE TABLE Payment (
   payment_id INT AUTO_INCREMENT PRIMARY KEY,
   bill_id INT NOT NULL,
   payment_method VARCHAR(20) NOT NULL,
   payment_status VARCHAR(20) NOT NULL,
   FOREIGN KEY (bill_id) REFERENCES Billing(bill_id)
);


INSERT INTO Customer (customer_name, contact_no) VALUES
('Amit Sharma','9123456789'),
('Priya Verma','9234567890'),
('Rahul Mehta','9345678901'),
('Sneha Patil','9456789012'),
('Karan Singh','9567890123');

SELECT * FROM Customer;

INSERT INTO Menu (item_name, price) VALUES
('Veg Burger',80),
('Cheese Pizza',200),
('Pasta',150),
('Paneer Thali',180),
('Masala Dosa',90);

SELECT * FROM Menu;

INSERT INTO Orders (customer_id, order_date) VALUES
(1,'2026-02-20'),
(2,'2026-02-21'),
(3,'2026-02-22'),
(4,'2026-02-23'),
(5,'2026-02-24');

SELECT * FROM Orders;

INSERT INTO Order_Details (order_id, menu_id, quantity) VALUES
(1,1,2),
(2,2,1),
(3,3,3),
(4,4,2),
(5,5,1);

SELECT * FROM Order_Details;

INSERT INTO Billing (order_id, total_amount, bill_date)
SELECT
    od.order_id,
    SUM(m.price * od.quantity),
    '2026-03-05'
FROM Order_Details od
JOIN Menu m ON od.menu_id = m.menu_id
GROUP BY od.order_id;

SELECT * FROM Billing;

INSERT INTO Payment (bill_id, payment_method, payment_status) VALUES
(1,'Cash','Paid'),
(2,'Online','Paid'),
(3,'Card','Paid'),
(4,'Cash','Paid'),
(5,'Online','Paid');

SELECT * FROM Payment;

SELECT
    b.bill_id,
    c.customer_name,
    m.item_name,
    od.quantity,
    m.price,
    b.total_amount,
    p.payment_method
FROM Customer c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Order_Details od ON o.order_id = od.order_id
JOIN Menu m ON od.menu_id = m.menu_id
JOIN Billing b ON o.order_id = b.order_id
JOIN Payment p ON b.bill_id = p.bill_id;
```

9.1 Database Creation

The database is created using the CREATE DATABASE command. After creation, the USE command is used to select the database for further operations.

```
CREATE DATABASE Restaurant_System;
USE Restaurant_System;
```

9.2 Table Creation

Tables are created using the CREATE TABLE command with proper primary keys and foreign key relationships to maintain data integrity.

Customer Table

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(50) NOT NULL,
    contact_no VARCHAR(15) UNIQUE
);
```

Menu Table

```
CREATE TABLE Menu (
    menu_id INT PRIMARY KEY AUTO_INCREMENT,
    item_name VARCHAR(50) NOT NULL,
    price DECIMAL(10,2) NOT NULL
);
```

Orders Table

```
CREATE TABLE Orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    order_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);
```

Order_Details Table

```
CREATE TABLE Order_Details (
    order_detail_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT,
    menu_id INT,
    quantity INT,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (menu_id) REFERENCES Menu(menu_id)
);
```

Billing Table

```
CREATE TABLE Billing (
   bill_id INT PRIMARY KEY AUTO_INCREMENT,
   order_id INT,
   total_amount DECIMAL(10,2),
   bill_date DATE,
   FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
```

Payment Table

```
CREATE TABLE Payment (
   payment_id INT PRIMARY KEY AUTO_INCREMENT,
   bill_id INT,
   payment_method VARCHAR(30),
   payment_status VARCHAR(20),
   FOREIGN KEY (bill_id) REFERENCES Billing(bill_id)
);
```

9.3 Data Insertion

Data is inserted using the INSERT INTO command.

Insert into Customer

```
INSERT INTO Customer (customer_name, contact_no)
VALUES ('Amit Sharma', '9876543210');
```

Insert into Menu

```
INSERT INTO Menu (item_name, price)
VALUES ('Veg Burger', 80.00);
```

Insert into Orders

```
INSERT INTO Orders (customer_id, order_date)
VALUES (1, '2026-02-20');
```

Insert into Order_Details

```
INSERT INTO Order_Details (order_id, menu_id, quantity)
VALUES (1, 1, 2);
```

Insert into Billing

```
INSERT INTO Billing (order_id, total_amount, bill_date)
VALUES (1, 160.00, '2026-02-20');
```

Insert into Payment

```
INSERT INTO Payment (bill_id, payment_method, payment_status)
VALUES (1, 'Cash', 'Paid');
```

9.4 Data Retrieval

Data is retrieved using the SELECT command with JOIN operations to combine multiple tables.

Display Complete Bill Summary

```
SELECT c.customer_name,
       m.item_name,
       od.quantity,
       b.total_amount
FROM Customer c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Order_Details od ON o.order_id = od.order_id
JOIN Menu m ON od.menu_id = m.menu_id
JOIN Billing b ON o.order_id = b.order_id;
```

9.5 Advanced Queries

Advanced queries are used to generate meaningful reports and summaries.

GROUP BY

Used to group records and calculate totals.

```
SELECT order_id, SUM(total_amount) AS Total_Bill
FROM Billing
GROUP BY order_id;
```

HAVING

Used to filter grouped results.

```
SELECT order_id, SUM(total_amount) AS Total_Bill
FROM Billing
GROUP BY order_id
HAVING SUM(total_amount) > 200;
```

Subqueries

Used to retrieve data based on another query.

```
SELECT customer_name
FROM Customer
WHERE customer_id IN (SELECT customer_id FROM Orders);
```

Views

Views simplify complex queries by storing them as virtual tables.

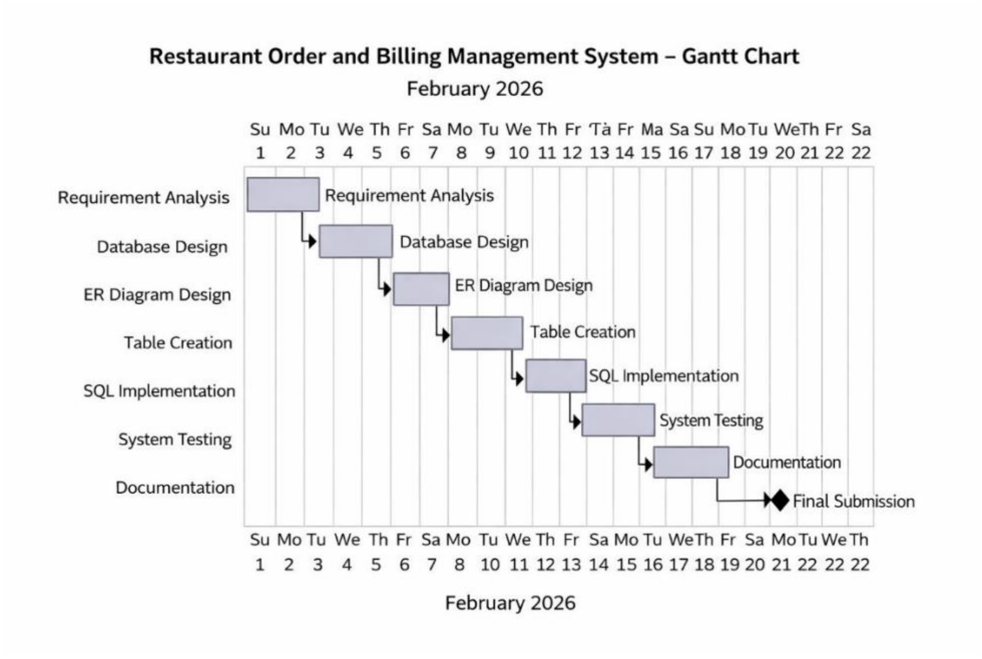
```
CREATE VIEW Bill_View AS
SELECT c.customer_name,
       b.total_amount
FROM Customer c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Billing b ON o.order_id = b.order_id;
```

To view data from the view:

```
SELECT * FROM Bill_View;
```

Gantt Chart Description:

The Gantt Chart illustrates the project schedule and time distribution for the development of the Restaurant Order and Billing Management System. It clearly represents the sequence of activities such as requirement analysis, database design, ER diagram design, table creation, SQL implementation, system testing, and documentation.
Each task is assigned a specific duration and arranged in a logical order to ensure smooth workflow and timely completion of the project. The chart helps in monitoring progress, managing time effectively, and ensuring that the project is completed within the planned schedule.



• The Gantt Chart represents the timeline and overall schedule of the Restaurant Order and Billing Management System project.

• It shows the sequence of development phases including requirement analysis, database design, ER diagram design, table creation, SQL implementation, system testing, and documentation.

• Each activity is assigned a specific duration to ensure systematic and organized project execution.

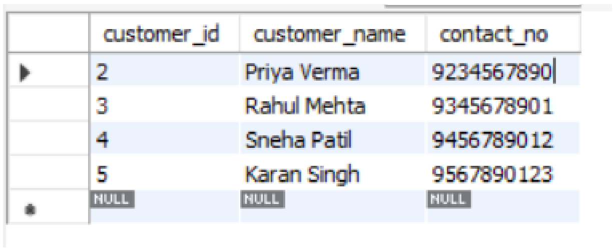
• The chart helps in tracking project progress and monitoring task completion at every stage.

• It ensures effective time management and timely submission of the project.

10. SYSTEM TESTING AND RESULT

System testing was carried out to ensure that the Restaurant Order and Billing Management System works correctly and produces accurate results. Different SQL queries were executed and verified to check the correctness of operations, data validity, and final output.

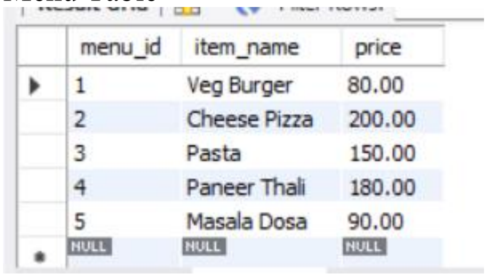
Customer Table



	customer_id	customer_name	contact_no
▶	2	Priya Verma	9234567890
	3	Rahul Mehta	9345678901
	4	Sneha Patil	9456789012
	5	Karan Singh	9567890123
*	NULL	NULL	NULL

This output displays the records stored in the Customer table. It confirms that customer details such as customer ID, name, and contact number have been successfully inserted and stored without duplication due to the UNIQUE constraint.

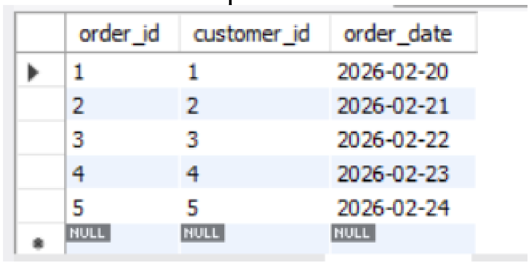
Menu Table



	menu_id	item_name	price
▶	1	Veg Burger	80.00
	2	Cheese Pizza	200.00
	3	Pasta	150.00
	4	Paneer Thali	180.00
	5	Masala Dosa	90.00
*	NULL	NULL	NULL

This output shows the list of food items available in the restaurant along with their respective prices. It verifies that menu data has been inserted correctly and that each item has a unique identifier.

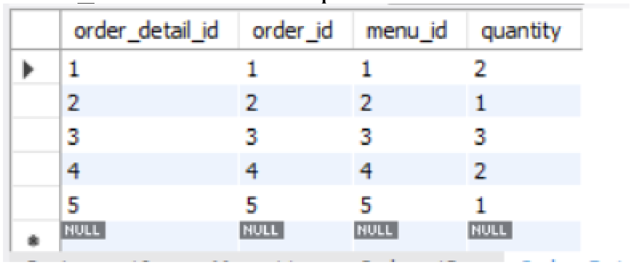
Orders Table Output



	order_id	customer_id	order_date
▶	1	1	2026-02-20
	2	2	2026-02-21
	3	3	2026-02-22
	4	4	2026-02-23
	5	5	2026-02-24
*	NULL	NULL	NULL

This output represents the orders placed by customers. It confirms that each order is linked to a valid customer through the foreign key relationship and that order dates are recorded correctly.

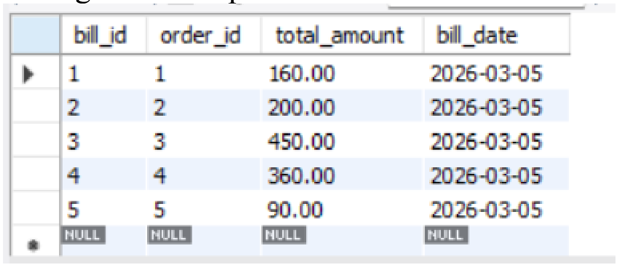
Order_Details Table Output



order_detail_id	order_id	menu_id	quantity
1	1	1	2
2	2	2	1
3	3	3	3
4	4	4	2
5	5	5	1
NULL	NULL	NULL	NULL

This output displays the detailed information of each order, including menu items and quantity ordered. It verifies the many-to-many relationship between Orders and Menu tables.

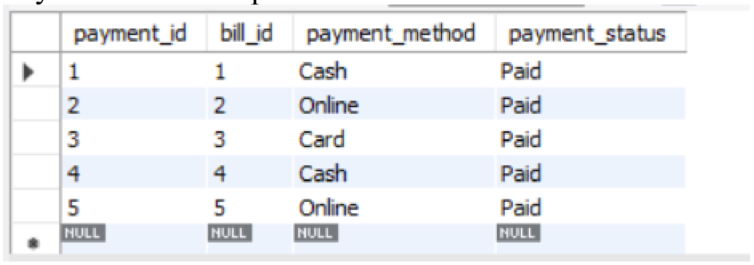
Billing Table Output



bill_id	order_id	total_amount	bill_date
1	1	160.00	2026-03-05
2	2	200.00	2026-03-05
3	3	450.00	2026-03-05
4	4	360.00	2026-03-05
5	5	90.00	2026-03-05
NULL	NULL	NULL	NULL

This output shows the generated bill for each order along with the calculated total amount. It confirms that billing information is generated accurately based on item price and quantity.

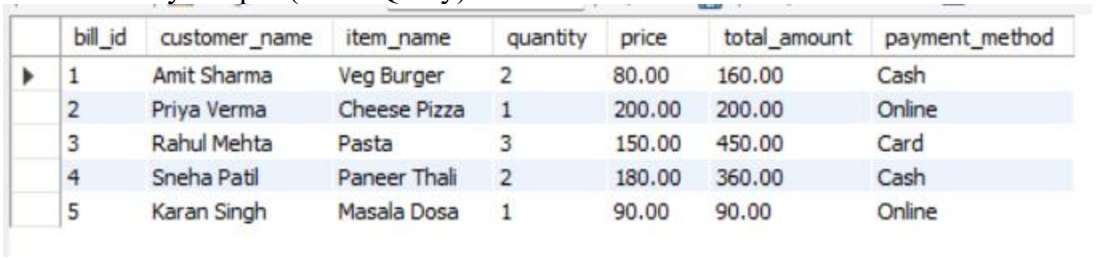
Payment Table Output



payment_id	bill_id	payment_method	payment_status
1	1	Cash	Paid
2	2	Online	Paid
3	3	Card	Paid
4	4	Cash	Paid
5	5	Online	Paid
NULL	NULL	NULL	NULL

This output displays the payment details for each bill, including payment method and payment status. It verifies that each payment record is correctly linked to its corresponding bill.

Bill Summary Output (JOIN Query)



bill_id	customer_name	item_name	quantity	price	total_amount	payment_method
1	Amit Sharma	Veg Burger	2	80.00	160.00	Cash
2	Priya Verma	Cheese Pizza	1	200.00	200.00	Online
3	Rahul Mehta	Pasta	3	150.00	450.00	Card
4	Sneha Patil	Paneer Thali	2	180.00	360.00	Cash
5	Karan Singh	Masala Dosa	1	90.00	90.00	Online

This output presents the final bill summary by combining data from multiple related tables using JOIN operations. It confirms that customer details, ordered items, total amount, and payment method are retrieved accurately

11. SECURITY, BACKUP AND RECOVERY

11.1 Security

Security is an important aspect of the Restaurant Order and Billing Management System because it handles sensitive data such as customer contact details, order history, billing amounts, and payment information. Proper security mechanisms are required to protect this data from unauthorized access, modification, or deletion.

In this system, security is ensured using MySQL's built-in authentication and access control features. Access to the database is restricted through username and password authentication. Only authorized users such as the administrator or restaurant staff are allowed to log in and perform database operations.

User privileges can be assigned using SQL commands such as GRANT and REVOKE to control what actions a user can perform. For example:

- The administrator may have full privileges (CREATE, INSERT, UPDATE, DELETE).
- Restaurant staff may only be allowed to insert orders and view billing details.
- Unauthorized users are prevented from modifying or deleting important records.

In addition, database constraints such as PRIMARY KEY, FOREIGN KEY, UNIQUE, and NOT NULL help maintain data integrity. These constraints prevent duplicate entries, invalid references, and incomplete records, ensuring that the system stores only valid and consistent data.

Thus, authentication, access control, and integrity constraints collectively ensure confidentiality, integrity, and reliability of restaurant data.

11.2 Backup

Backup is essential to protect the Restaurant Order and Billing Management System from accidental data loss, system crashes, hardware failure, or human errors such as incorrect deletion of records.

In this system, database backups can be created using the mysqldump utility provided by MySQL. The mysqldump command generates a complete backup file containing:

- Database structure (tables, constraints, relationships)
- All inserted records (customer, menu, orders, billing, and payment data)

Regular backups ensure that important restaurant data such as customer details, order history, billing records, and payment information can be restored whenever required. Periodic backup is especially important for business continuity and reliability.

11.3 Recovery

Recovery refers to restoring the database from a previously created backup file. In case of data loss or system failure, the database can be restored using the backup created by mysqldump. This process helps recover customer details, order records, and billing information, allowing the system to continue functioning without permanent data loss

- All tables (Customer, Menu, Orders, Order_Details, Billing, and Payment)
- All stored records
- All relationships and constraints

The recovery process ensures that restaurant operations can continue without permanent loss of important data. It guarantees business continuity and maintains system reliability.

12. FUTURE SCOPE AND CONCLUSION

12.1 Future Scope

The Restaurant Order and Billing Management System developed in this project provides a strong foundation for managing restaurant operations using a relational database. However, the system can be further enhanced and expanded by adding advanced features and integrating modern technologies.

One major improvement would be the development of a web-based or mobile-based interface. Currently, the system operates through SQL queries in MySQL Workbench for academic purposes. In the future, a user-friendly graphical interface can be developed using technologies such as HTML, CSS, JavaScript, or any backend framework. This would allow restaurant staff to place orders, generate bills, and record payments through an interactive dashboard instead of manually executing SQL commands.

Another enhancement could be the implementation of online ordering functionality. Customers could place orders directly through a website or mobile application, and the system would automatically store the data in the database. This would enable real-time order processing and automatic bill generation, improving efficiency and customer satisfaction.
Advanced reporting and analytics features can also be added. The system can generate:

- Daily, weekly, and monthly sales reports
- Customer order history reports
- Revenue summaries
- Most frequently ordered menu items
- Peak business hours analysis

These reports would help restaurant management analyze sales trends and make better business decisions. For example, management can identify popular dishes, adjust pricing strategies, or introduce promotional offers based on customer preferences.
Integration with digital payment gateways such as UPI, debit/credit cards, and online wallets can also be included in future versions. This would allow secure and automated payment processing within the system.

Additionally, inventory management can be incorporated to track stock levels of raw materials. The system can automatically update inventory whenever an order is placed and notify the administrator when stock is low.

Security features can also be enhanced by implementing role-based access control, data encryption, and audit logs to track user activities.
In conclusion, while the current system successfully demonstrates core DBMS concepts such as table creation, relationships, constraints, joins, and transaction management, it has significant potential for expansion into a full-scale, real-world restaurant management application.

12.2 Conclusion

The Restaurant Order and Billing Management System successfully demonstrates the practical design and implementation of a relational database system using MySQL. The primary objective of the project was to develop a structured and efficient database solution to manage restaurant operations such as customer management, menu handling, order processing, billing, and payment recording. All these objectives have been successfully achieved through proper database design and SQL implementation.

The system is designed using multiple related tables such as Customer, Menu, Orders, Order_Details, Billing, and Payment. These tables are interconnected using primary keys and foreign keys to maintain strong relationships and ensure data consistency. By applying constraints such as PRIMARY KEY, FOREIGN KEY, UNIQUE, and NOT NULL, the system prevents duplicate entries, invalid references, and incomplete records. This ensures high data integrity and reliability.

SQL queries were effectively used to perform various operations including data insertion, retrieval, grouping, joining multiple tables, and generating final bill summaries. The implementation of JOIN operations demonstrates how data from multiple related tables can be combined to generate meaningful outputs such as detailed billing reports. Advanced concepts like GROUP BY and aggregate functions were used to calculate total bill amounts accurately. The project also highlights the importance of transaction management in maintaining database consistency. By understanding the role of COMMIT and ROLLBACK, the system ensures that important operations such as order placement and billing are completed safely without partial updates.

Through this project, practical knowledge of database normalization, table relationships, constraint enforcement, query execution, and data validation was gained. It provided hands-on experience with MySQL Workbench and enhanced understanding of real-world database applications.

The use of MySQL as an open-source relational database management system demonstrates its effectiveness, reliability, and scalability in developing database-driven applications. Overall, the project successfully fulfills its intended purpose and provides a strong foundation for expanding into a full-scale restaurant management system in the future.

13. REFERENCES

The following references were used for understanding database concepts, SQL queries, and MySQL implementation during the development of the **Restaurant Order and Billing Management System**.

MySQL Official Documentation:

The official MySQL documentation was referred to for learning SQL syntax, database creation, table structures, constraints, transaction management, and query execution. It provided accurate and detailed information about MySQL features, commands, and functions used in this project.
Website: https://dev.mysql.com/doc/

Prescribed Textbooks:

Academic textbooks related to Database Management Systems (DBMS) were used to understand theoretical concepts such as relational database models, normalization, primary and foreign keys, constraints, entity relationships, and database design principles.

Online Learning Sources:

Various online tutorials, educational websites, and learning platforms were used to understand the practical implementation of SQL queries and MySQL Workbench. These resources helped in learning joins, GROUP BY clauses, views, aggregate functions, and transaction commands used in the Restaurant Order and Billing Management System.
Examples:

- www.w3schools.com
- www.geeksforgeeks.org
- www.tutorialspoint.com

College Study Materials and Notes:

Lecture notes, classroom demonstrations, practical sessions, and guidance provided by faculty members were also referred to during the development of the Restaurant Order and Billing Management System.

14. GLOSSARY

DBMS (Database Management System):
A DBMS is software used to create, manage, and maintain databases. It allows users to store, retrieve, and manipulate data efficiently. In this project, MySQL is used as the DBMS to manage restaurant data such as customers, menu items, orders, billing, and payments.

SQL (Structured Query Language):
SQL is a standard language used to communicate with the database. It is used to create tables, insert data, update records, delete data, and retrieve information. In this system, SQL is used to manage customer details, menu records, orders, and billing operations.

Primary Key:
A Primary Key is a unique identifier for each record in a table. It ensures that no duplicate values exist and helps to uniquely identify each record.
Example: customer_id in the Customer table and order_id in the Orders table.

Foreign Key:
A Foreign Key is a field in one table that refers to the Primary Key of another table. It is used to maintain relationships between tables.
Example: customer_id in the Orders table links to the Customer table.

MySQL
MySQL is an open-source relational database management system used to store and manage structured data. It supports SQL queries, constraints, and transaction management. In this project, MySQL Workbench is used to implement the Restaurant Order and Billing Management System.

Table:
A table is a structured collection of related data organized into rows and columns. Each table stores specific information such as Customer details, Menu items, Orders, Billing records, or Payment information.

Record (Row):
A record is a single row in a table that contains complete information about one entity. For example, one customer entry in the Customer table represents one record.

Query:
A query is a request made to the database to retrieve or manipulate data. Queries are written using SQL commands.
Example: The SELECT query is used to retrieve customer or billing details from the database.

View:
A View is a virtual table created using a SELECT query from one or more tables. It helps simplify complex queries and display selected data such as a final bill summary combining Customer, Orders, and Payment information.

Transaction:
A transaction is a group of SQL operations executed as a single unit. It ensures that all operations are completed successfully. Commands such as COMMIT and ROLLBACK are used to manage transactions and maintain data consistency.

Billing:
Billing refers to the process of calculating the total amount for ordered menu items based on price and quantity. The Billing table stores the total bill amount and bill date for each order.

Payment:
Payment refers to the process of recording how a customer pays for their order. The Payment table stores payment method (Cash, Card, Online) and payment status.