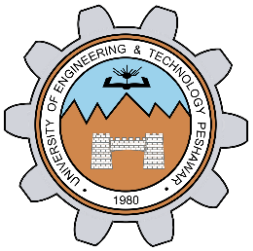
KAZ KITCHEN RESERVATION SYSTEM

PROJECT REPORT



Spring 2025

CSE-403L Database Management System Lab

Group Members:

KHIZRA HAROON (22PWCSE2121)
AREEJ (22PWCSE2206)
HAFIZA ZARLISHT NOOR (22PWCSE2112)

Class Section: **C**

Submitted to:

Engr. Sumayyea Salahuddin

June 20, 2025

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

INTRODUCTION

In today's fast-paced world, efficient reservation systems have become essential for managing hospitality services. The KAZ Kitchen Reservation System is a web-based application developed using Laravel and MySQL that enables customers to reserve tables online. It provides users with a simple interface to select dining services, book a table, and optionally avail add-on services like birthday or anniversary setups.

This system minimizes manual management, reduces overbooking issues, and enhances customer experience by generating a digital reservation ticket and allowing for cancellations.

TECHNOLOGIES USED

- **Frontend:** HTML, CSS, Bootstrap
- **Backend:** Laravel 12.19.0 Framework
- **Database:** MySQL (MariaDB)
- **Local Server:** XAMPP

FINALIZED CONCEPTUAL SCHEMA

ENTITY DESCRIPTION

CUSTOMER	A person who creates an account to reserve a table. Example: Khizra.
RESERVATION	The transaction associated with a customer booking a table, time slot, and service on a specific date. Example: Khizra books Table 5 for 4 guests on May 27, 2025, from 7 PM to 9 PM.
TABLE	A physical dining spot available in the restaurant, each having a specific seating capacity. Example: Table 5, which can seat 4 people.
TIME_SLOT	A predefined time interval during which a reservation can be made. Example: A slot from 7:00 PM to 9:00 PM.
SERVICE	An optional service a customer may select during the reservation, like decoration. Example: "Anniversary Decoration" service with candlelight setup.
CANCELLATION	The record of a reservation being cancelled. Example: Khizra cancels her decorated dinner reservation.

ENTITY RELATIONSHIP DIAGRAM (ERD)

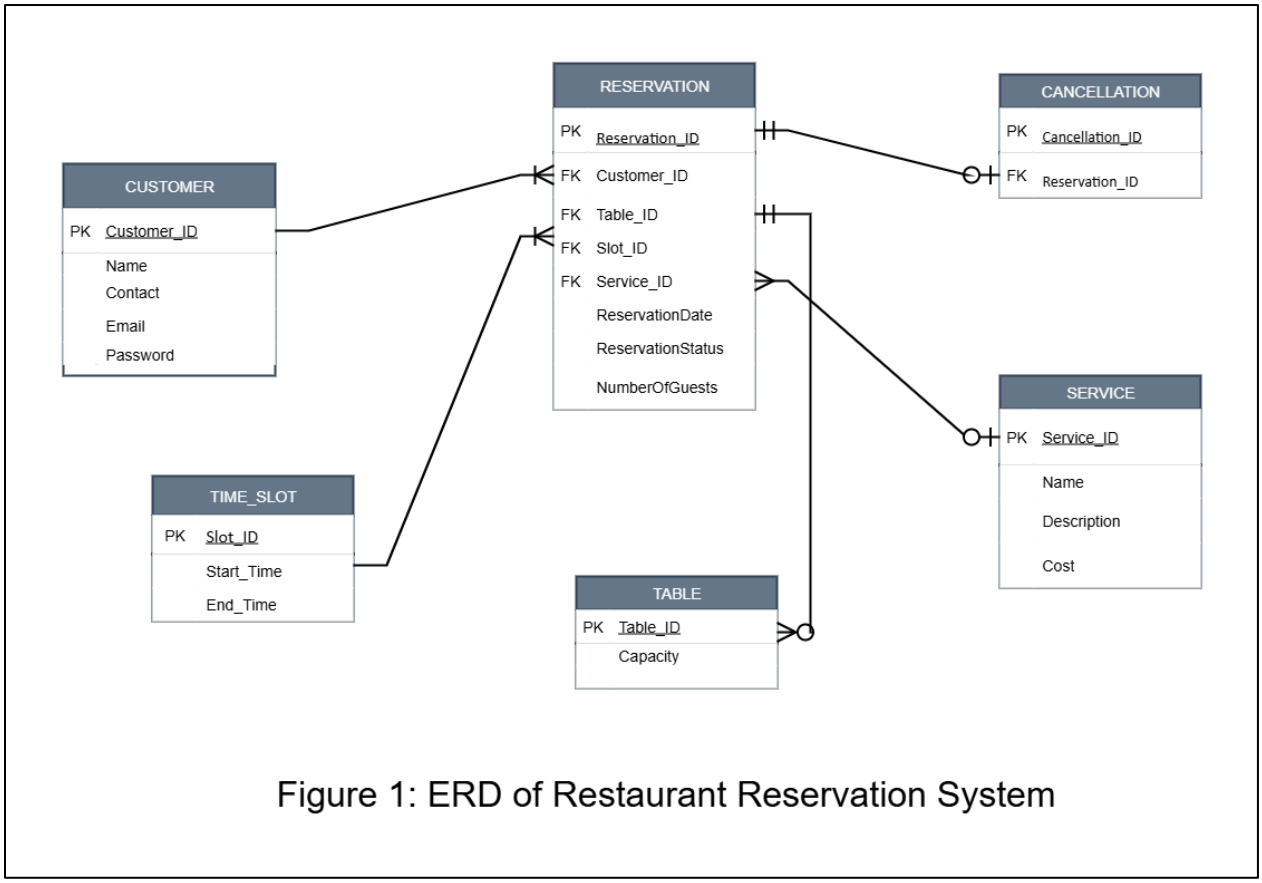
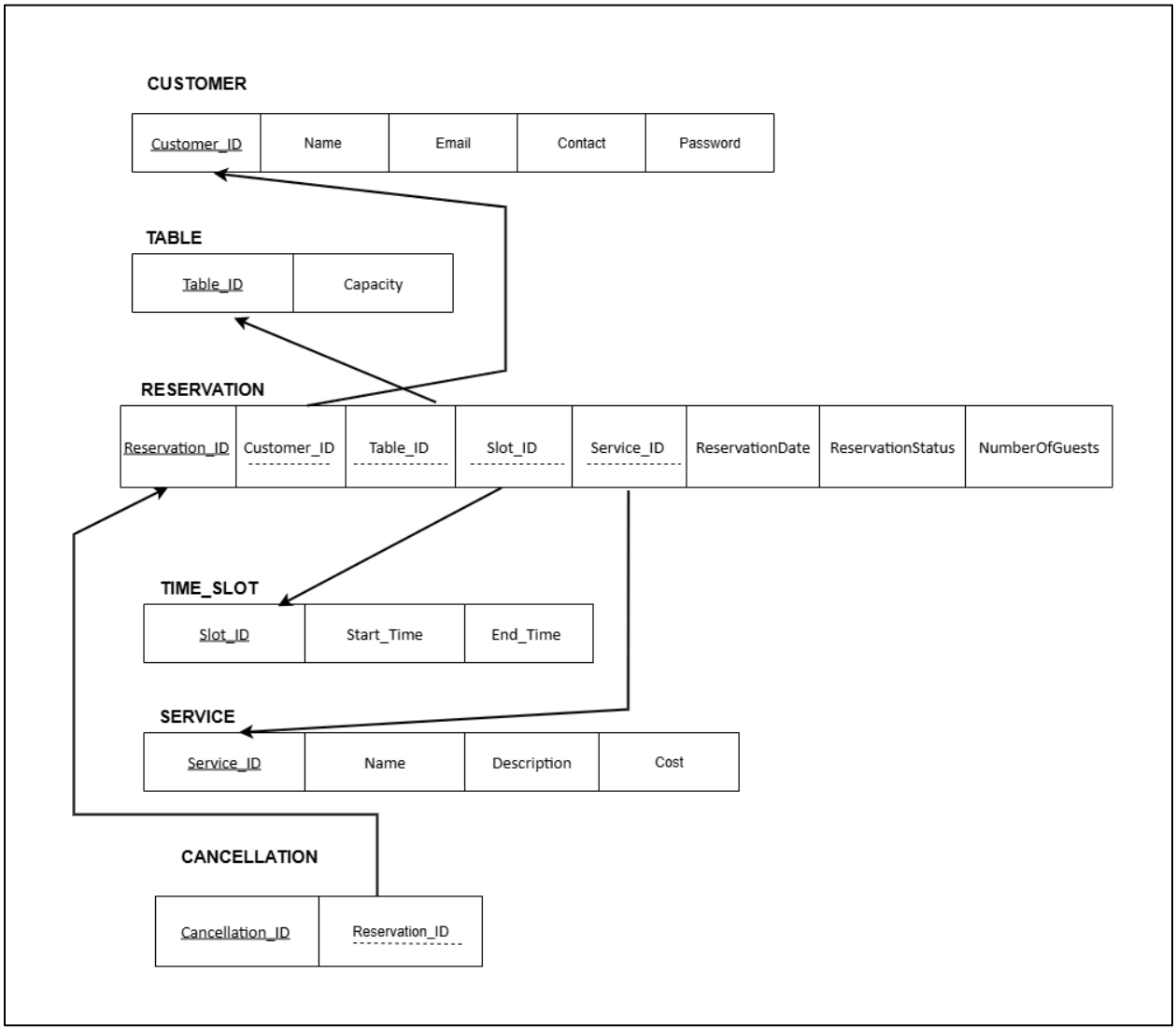


Figure 1: ERD of Restaurant Reservation System

RELATIONAL SCHEMA



NORMALIZATION TO 3NF

Each of the following relations has been analyzed to ensure it meets the requirements of the Third Normal Form (3NF).
This involves verifying that each non-key attribute is fully functionally dependent on the primary key, and that there are no transitive dependencies.

1. **CUSTOMER**
   - **1NF**: All fields are atomic.
   - **2NF**: Only one primary key (Customer_ID), so no partial dependencies.
   - **3NF**: No transitive dependencies.
     Already in 3NF

2. **TABLE**
   - **1NF**: all fields are atomic.
   - **2NF**: Single primary key (Table_ID), so no partial dependencies.
   - **3NF**: Capacity depends only on Table_ID.
     Already in 3NF

3. **TIME_SLOT**
   - **1NF**: Atomic fields.
   - **2NF**: Single key, so no partial dependencies.
   - **3NF**: No transitive dependencies.
     Already in 3NF

4. **SERVICE**
   - **1NF**: Atomic fields.
   - **2NF**: Single PK, so no partial dependency.
   - **3NF**: No transitive dependency.
     Already in 3NF

5. **RESERVATION**
   - **1NF**: All fields are atomic.
   - **2NF**: Reservation_ID is the primary key, and all non-key attributes depend entirely on it.
   - **3NF**: No transitive dependency among non-key attributes.
     Already in 3NF

6. **CANCELLATION**
   - **1NF**: Atomic fields.
   - **2NF**: Single PK (Cancellation_ID), so no partial dependency.
   - **3NF**: Every attribute depends on the cancellation.
     Already in 3NF

DATABASE AND TABLES

Database Name

```
MariaDB [(none)]> create database restaurant_db;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> use restaurant_db;
Database changed
```

Table: Customer

```
MariaDB [restaurant_db]> create table Customer (
    -> Customer_ID int auto_increment primary key,
    -> Name varchar(100),
    -> Contact varchar(20),
    -> Email varchar(100),
    -> Password varchar(100));
Query OK, 0 rows affected (0.039 sec)
```

```
MariaDB [restaurant_db]> describe customer;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| Customer_ID | int(11)      | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(100) | YES  |     | NULL    |                |
| Contact     | varchar(20)  | YES  |     | NULL    |                |
| Email       | varchar(100) | YES  |     | NULL    |                |
| Password    | varchar(100) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

Table: TableInfo

```
MariaDB [restaurant_db]> create table TableInfo (
    -> Table_ID int auto_increment primary key,
    -> Capacity int);
Query OK, 0 rows affected (0.062 sec)
```

```
MariaDB [restaurant_db]> describe TableInfo;
+----------+---------+------+-----+---------+----------------+
| Field    | Type    | Null | Key | Default | Extra          |
+----------+---------+------+-----+---------+----------------+
| Table_ID | int(11) | NO   | PRI | NULL    | auto_increment |
| Capacity | int(11) | YES  |     | NULL    |                |
+----------+---------+------+-----+---------+----------------+
2 rows in set (0.046 sec)
```

Table: Time_Slot

```
MariaDB [restaurant_db]> create table Time_Slot (
    -> Slot_ID int auto_increment primary key,
    -> Start_Time TIME,
    -> End_Time TIME);
Query OK, 0 rows affected (0.051 sec)
```

```
MariaDB [restaurant_db]> describe Time_Slot;
+------------+---------+------+-----+---------+----------------+
| Field      | Type    | Null | Key | Default | Extra          |
+------------+---------+------+-----+---------+----------------+
| Slot_ID    | int(11) | NO   | PRI | NULL    | auto_increment |
| Start_Time | time    | YES  |     | NULL    |                |
| End_Time   | time    | YES  |     | NULL    |                |
+------------+---------+------+-----+---------+----------------+
```

Table: Service

```
MariaDB [restaurant_db]> create table Service (
    -> Service_ID int auto_increment primary key,
    -> Name varchar(100),
    -> Description TEXT,
    -> Cost Decimal(10,2));
Query OK, 0 rows affected (0.054 sec)
```

```
MariaDB [restaurant_db]> describe Service;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| Service_ID  | int(11)       | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(100)  | YES  |     | NULL    |                |
| Description | text          | YES  |     | NULL    |                |
| Cost        | decimal(10,2) | YES  |     | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
4 rows in set (0.040 sec)
```

Table: Reservation

```
MariaDB [restaurant_db]> create table Reservation (
    -> Reservation_ID int auto_increment primary key,
    -> Customer_ID int,
    -> Table_ID int,
    -> Slot_ID int,
    -> Service_ID int,
    -> ReservationDate DATE,
    -> ReservationStatus ENUM('Confirmed', 'Cancelled'),
    -> NumberOfGuests int,
    -> FOREIGN KEY (Customer_ID) references Customer(Customer_ID),
    -> FOREIGN KEY (Table_ID) references TableInfo(Table_ID),
    -> FOREIGN KEY (Slot_ID) references Time_Slot(Slot_ID),
    -> FOREIGN KEY (Service_ID) references Service(Service_ID));
Query OK, 0 rows affected (0.093 sec)
```

```
MariaDB [restaurant_db]> describe Reservation;
+------------------+-----------------------------+------+-----+---------+----------------+
| Field            | Type                        | Null | Key | Default | Extra          |
+------------------+-----------------------------+------+-----+---------+----------------+
| Reservation_ID   | int(11)                     | NO   | PRI | NULL    | auto_increment |
| Customer_ID      | int(11)                     | YES  | MUL | NULL    |                |
| Table_ID         | int(11)                     | YES  | MUL | NULL    |                |
| Slot_ID          | int(11)                     | YES  | MUL | NULL    |                |
| Service_ID       | int(11)                     | YES  | MUL | NULL    |                |
| ReservationDate  | date                        | NO   |     | NULL    |                |
| ReservationStatus | enum('Confirmed','Cancelled') | YES  |     | NULL    |                |
| NumberOfGuests   | int(11)                     | YES  |     | NULL    |                |
+------------------+-----------------------------+------+-----+---------+----------------+
8 rows in set (0.048 sec)
```

Table: Cancellation

```
MariaDB [restaurant_db]> create table Cancellation (
    -> Cancellation_ID int auto_increment primary key,
    -> Reservation_ID int UNIQUE,
    -> FOREIGN KEY (Reservation_ID) REFERENCES Reservation(Reservation_ID));
Query OK, 0 rows affected (0.061 sec)
```

```
MariaDB [restaurant_db]> describe Cancellation;
+-----------------+---------+------+-----+---------+----------------+
| Field           | Type    | Null | Key | Default | Extra          |
+-----------------+---------+------+-----+---------+----------------+
| Cancellation_ID | int(11) | NO   | PRI | NULL    | auto_increment |
| Reservation_ID  | int(11) | YES  | UNI | NULL    |                |
+-----------------+---------+------+-----+---------+----------------+
2 rows in set (0.046 sec)
```

TRIGGERS USED

Trigger 1: Prevent Double Booking

This checks if a confirmed reservation already exists for the same table and time on the same date. If yes, it throws an error and blocks the insertion.

```
MariaDB [restaurant_db]> DELIMITER $$
MariaDB [restaurant_db]> create trigger prevent_double_booking
    -> BEFORE INSERT ON reservation
    -> FOR EACH ROW
    -> BEGIN
    ->     IF EXISTS (
    ->         SELECT 1 FROM reservation
    ->         WHERE Table_ID = NEW.Table_ID
    ->           AND Slot_ID = NEW.Slot_ID
    ->           AND ReservationDate = NEW.ReservationDate
    ->           AND ReservationStatus = 'Confirmed'
    ->     ) THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'This table is already booked for the same time slot and date.';
    -> END IF;
    -> END$$
Query OK, 0 rows affected (0.056 sec)
```

Trigger 2: Auto-Update Status on Cancellation

When a new row is added to Cancellation, this updates the related reservation status.

```
MariaDB [restaurant_db]> DELIMTER $$
    -> CREATE TRIGGER update_status_on_cancellation
    -> AFTER INSERT ON Cancellation
    -> FOR EACH ROW
    -> UPDATE Reservation
    -> SET ReservationStatus = 'Cancelled'
    -> WHERE Reservation_ID = NEW.Reservation_ID;
    -> WHERE Reservation_ID = NEW.Reservation_ID;
    -> END$$
Query OK, 0 rows affected (0.037 sec)
```

ON DELETE CASCADE FOREIGN KEY CONSTRAINTS

This ensures related rows get deleted when a parent is deleted (like Cancellation when Reservation is deleted).

```
MariaDB [restaurant_db]> ALTER TABLE Cancellation
    -> ADD CONSTRAINT fk_res_cancel
    -> FOREIGN KEY (Reservation_ID) REFERENCES Reservation(Reservation_ID)
    -> ON DELETE CASCADE;
Query OK, 2 rows affected (0.182 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

SAMPLE QUERIES

1. **SHOW TABLES;**
   Shows all the tables in the database.

```
MariaDB [restaurant_db]> show tables;
+-------------------------+
| Tables_in_restaurant_db |
+-------------------------+
| cancellation            |
| customer                |
| reservation             |
| service                 |
| tableinfo               |
| time_slot               |
+-------------------------+
6 rows in set (0.001 sec)
```

2. **SELECT * FROM Customer;**
   Fetches all rows and all columns from the Customer table.

```
MariaDB [restaurant_db]> select * from Customer;
+-------------+---------------+-------------+--------------------------+----------+
| Customer_ID | Name          | Contact     | Email                    | Password |
+-------------+---------------+-------------+--------------------------+----------+
|           1 | Khizra Haroon | 03419084519 | khizra.haroon3@gmail.com | 1422     |
|           2 | Areej         | 03018977431 | areej123@gmail.com       | abc123   |
|           3 | Zarlisht Noor | 03028867400 | zarlishtkhan@gmail.com   | try456   |
|           4 | Ali           | 03312330541 | alykhn@gmail.com         | 123456   |
|           5 | Bruno         | 03318911221 | bruno@gmail.com          | bruno123 |
+-------------+---------------+-------------+--------------------------+----------+
5 rows in set (0.001 sec)
```

3. **SELECT * FROM Reservation;**
   Fetches all rows and all columns from the Reservation table.

```
MariaDB [restaurant_db]> select * from reservation;
+----------------+-------------+----------+---------+------------+-----------------+------------------+----------------+
| Reservation_ID | Customer_ID | Table_ID | Slot_ID | Service_ID | ReservationDate | ReservationStatus | NumberOfGuests |
+----------------+-------------+----------+---------+------------+-----------------+------------------+----------------+
|              1 |           1 |        2 |       1 |          1 | 2025-06-01      | Cancelled         |              2 |
|              2 |           2 |        3 |       2 |          2 | 2025-06-02      | Confirmed         |              4 |
|              3 |           3 |        1 |       3 |       NULL | 2025-06-03      | Cancelled         |              2 |
|              4 |           4 |        4 |       4 |          3 | 2025-06-04      | Confirmed         |              6 |
|              5 |           5 |        5 |       5 |          5 | 2025-06-05      | Confirmed         |              8 |
+----------------+-------------+----------+---------+------------+-----------------+------------------+----------------+
5 rows in set (0.000 sec)
```

4. **SELECT * FROM TableInfo;**
   Fetches all rows and all columns from the TableInfo table.

```
MariaDB [restaurant_db]> select * from TableInfo;
+----------+----------+
| Table_ID | Capacity |
+----------+----------+
|        1 |        2 |
|        2 |        3 |
|        3 |        6 |
|        4 |        8 |
|        5 |       10 |
+----------+----------+
5 rows in set (0.001 sec)
```

5. **SELECT * FROM Time_Slot;**
   Fetches all rows and all columns from the Time_Slot table.

```
MariaDB [restaurant_db]> select * from Time_Slot;
+---------+------------+----------+
| Slot_ID | Start_Time | End_Time |
+---------+------------+----------+
|       1 | 18:00:00   | 20:00:00 |
|       2 | 20:00:00   | 21:00:00 |
|       3 | 21:00:00   | 22:00:00 |
|       4 | 22:00:00   | 24:00:00 |
|       5 | 17:00:00   | 18:00:00 |
|       6 | 17:00:00   | 18:00:00 |
+---------+------------+----------+
```

6. **SELECT * FROM Service;**
   Fetches all rows and all columns from the Service table.

```
MariaDB [restaurant_db]> select * from Service;
+------------+-------------------+-------------+---------+
| Service_ID | Name              | Description | Cost    |
+------------+-------------------+-------------+---------+
|          1 | Birthday Setup    | NULL        | 5000.00 |
|          2 | Anniversary Decor | NULL        | 5000.00 |
|          3 | Valentine Setup   | NULL        | 4500.00 |
|          4 | Corporate Setup   | NULL        | 5500.00 |
|          5 | Candlelight Dinner| NULL        | 6000.00 |
+------------+-------------------+-------------+---------+
5 rows in set (0.000 sec)
```

7. **SELECT * FROM Cancellation;**
   Fetches all rows and all columns from the Cancellation table.

```
MariaDB [restaurant_db]> select * from cancellation;
+-----------------+----------------+
| Cancellation_ID | Reservation_ID |
+-----------------+----------------+
|               2 |              1 |
|               1 |              3 |
+-----------------+----------------+
2 rows in set (0.000 sec)
```

8. View All Confirmed Reservations:

```
MariaDB [restaurant_db]> SELECT * FROM Reservation WHERE ReservationStatus = 'Confirmed';
+----------------+-------------+----------+---------+------------+-----------------+-------------------+----------------+
| Reservation_ID | Customer_ID | Table_ID | Slot_ID | Service_ID | ReservationDate | ReservationStatus | NumberOfGuests |
+----------------+-------------+----------+---------+------------+-----------------+-------------------+----------------+
|              2 |           2 |        3 |       2 |          2 | 2025-06-02      | Confirmed         |              4 |
|              4 |           4 |        4 |       4 |          3 | 2025-06-04      | Confirmed         |              6 |
|              5 |           5 |        5 |       5 |          5 | 2025-06-05      | Confirmed         |              8 |
+----------------+-------------+----------+---------+------------+-----------------+-------------------+----------------+
3 rows in set (0.001 sec)
```

9. Show Upcoming Reservations:

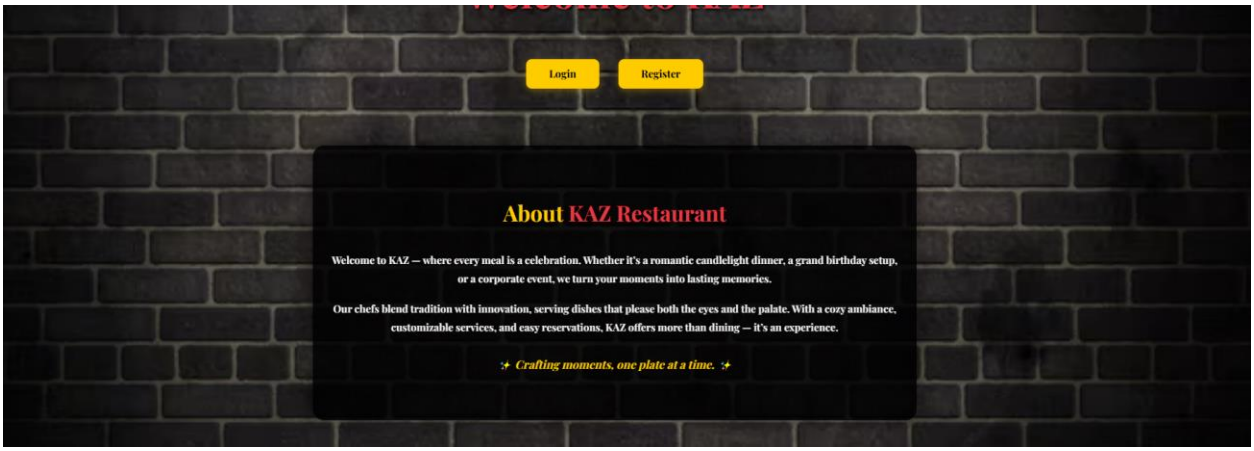
```
MariaDB [restaurant_db]> SELECT *
    -> FROM Reservation
    -> WHERE ReservationStatus = 'Confirmed'
    ->   AND ReservationDate >= CURDATE()
    -> ORDER BY ReservationDate, Slot_ID;
+----------------+-------------+----------+---------+------------+-----------------+-------------------+----------------+
| Reservation_ID | Customer_ID | Table_ID | Slot_ID | Service_ID | ReservationDate | ReservationStatus | NumberOfGuests |
+----------------+-------------+----------+---------+------------+-----------------+-------------------+----------------+
|              2 |           2 |        3 |       2 |          2 | 2025-06-02      | Confirmed         |              4 |
|              4 |           4 |        4 |       4 |          3 | 2025-06-04      | Confirmed         |              6 |
|              5 |           5 |        5 |       5 |          5 | 2025-06-05      | Confirmed         |              8 |
+----------------+-------------+----------+---------+------------+-----------------+-------------------+----------------+
3 rows in set (0.001 sec)
```

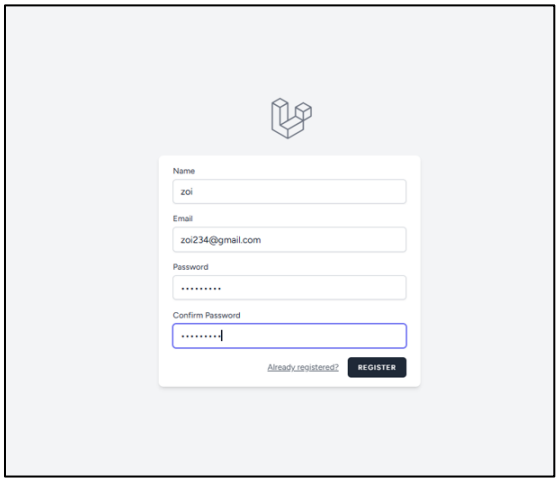
LARAVEL IMPLEMENTATION

STEPS TAKEN:

1. Created a Laravel project and configured the ".env" for MySQL
2. Created models for all entities: Customer, Reservation, Service, TableInfo, TimeSlot, Cancellation.
3. Developed views:
   - Home page with restaurant name and login.
   - Page showing available tables and services.
   - Reservation form with date, slot, service.
   - Ticket view after successful reservation.
   - Option to cancel the reservation.
4. Backend:
   - Controllers: ReservationController, PaymentController, CustomerController.
   - Routes configured in `web.php`.
   - Logic implemented to check availability, generate a ticket, and handle cancellations.

LANDING PAGE – LOGIN / REGISTER

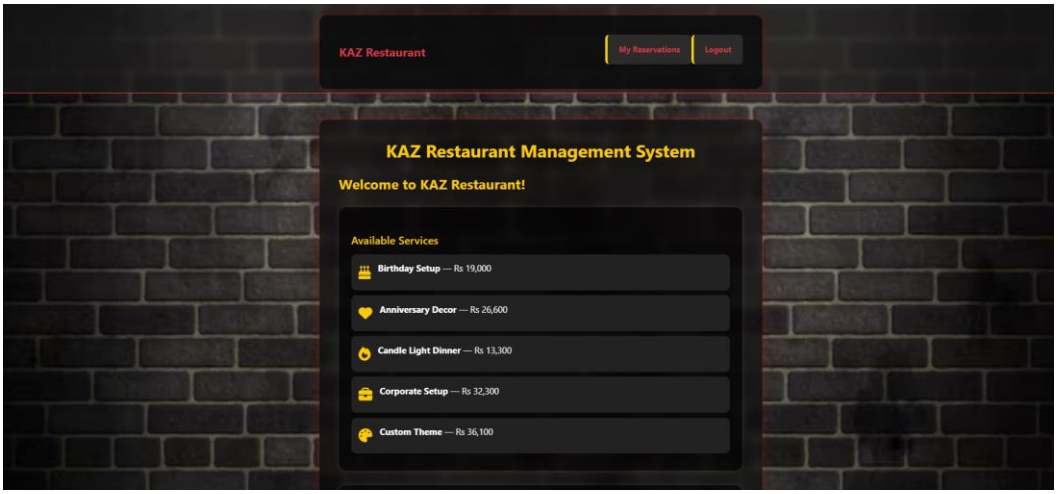




USER HOME DASHBOARD

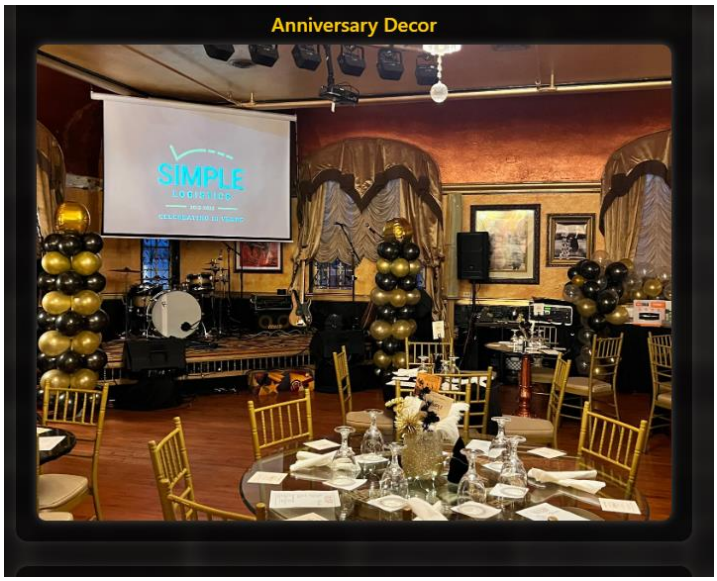
You can view available services with their costs and table availability, along with their capacity.

VIEW AVAILABLE SERVICES



Birthday Setup

Anniversary Decor

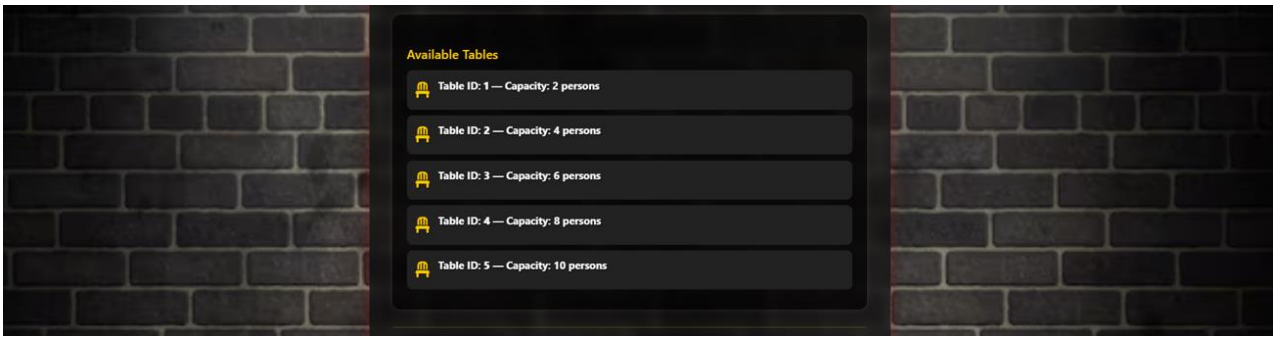


Candle Light Dinner

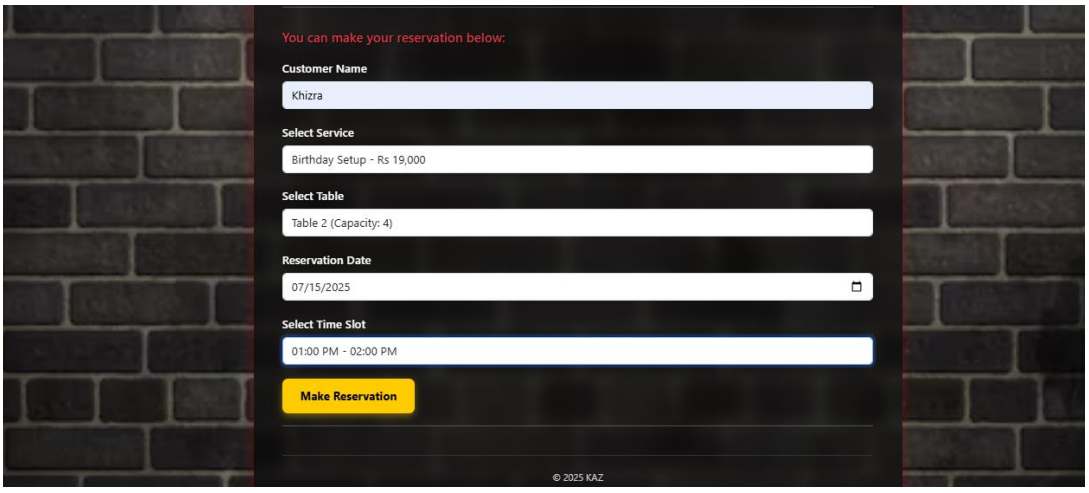
Corporate Setup

Custom Theme

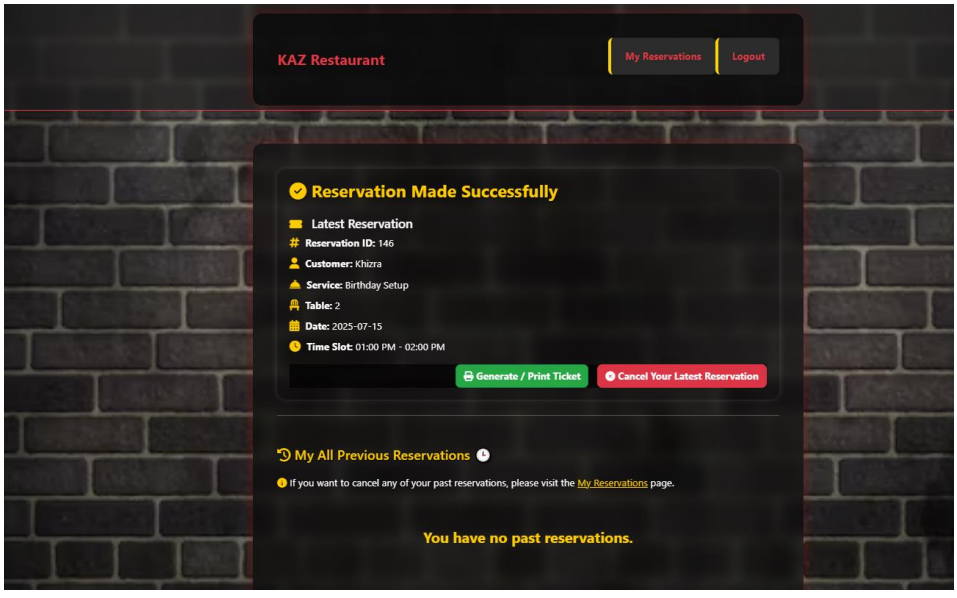
VIEW TABLE AVAILABILITY



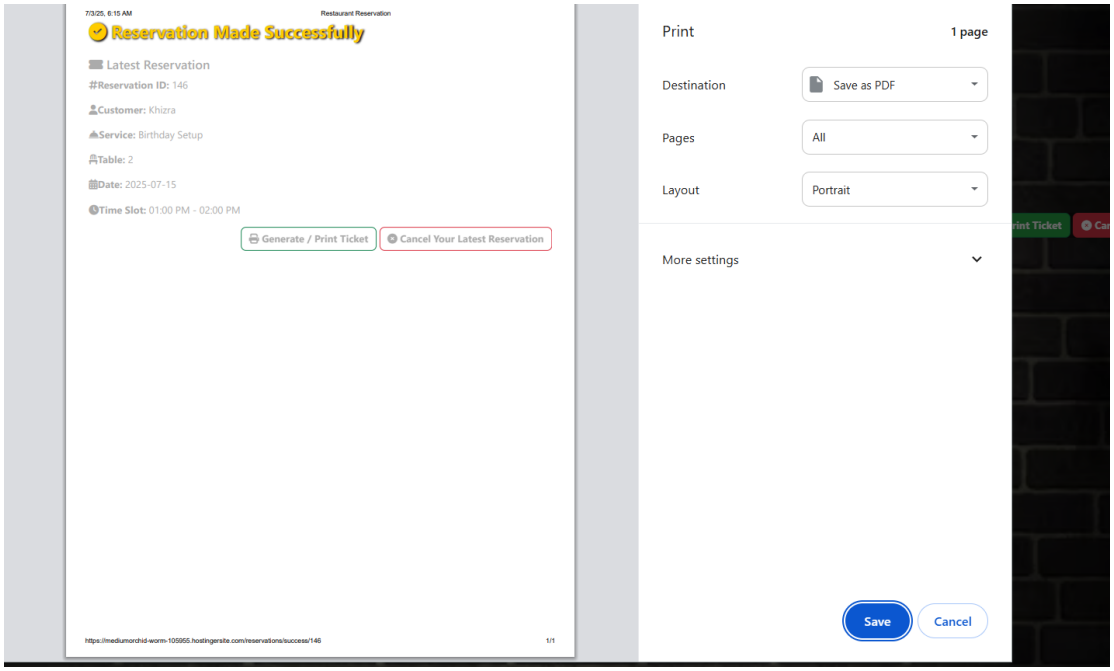
MAKE A RESERVATION FORM



SUCCESSFUL RESERVATION

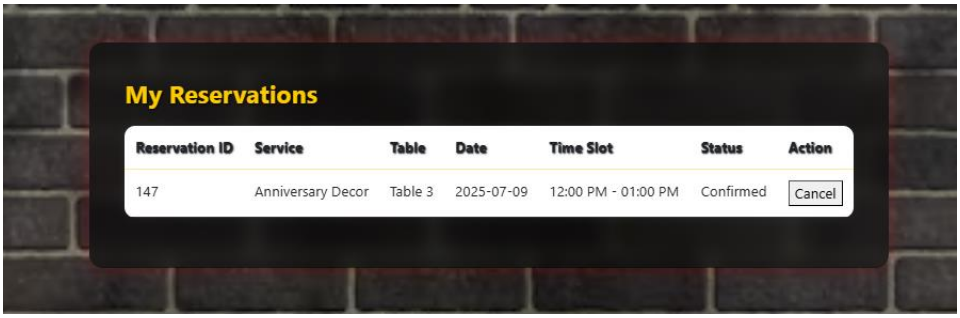


RESERVATION CONFIRMATION TICKET

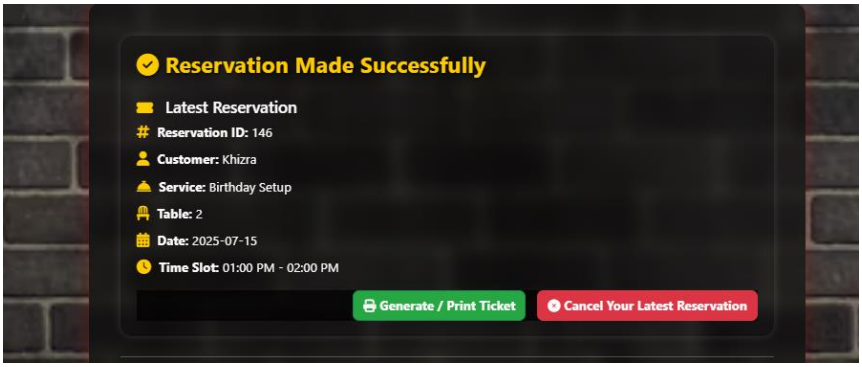


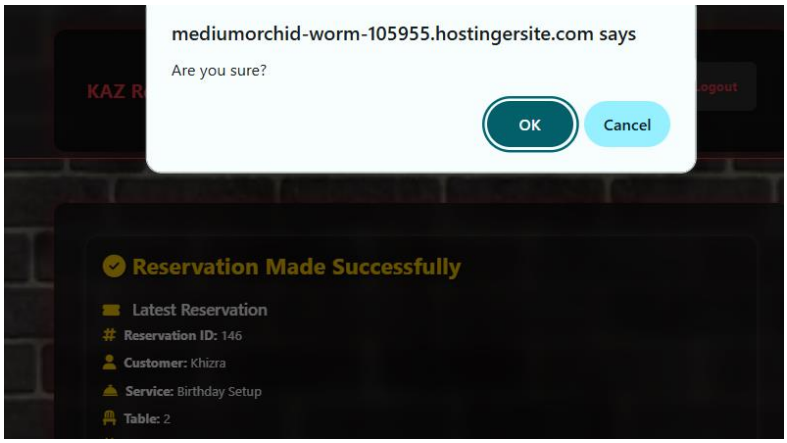
MY RESERVATIONS PAGE

It shows all the reservations made by the customer.

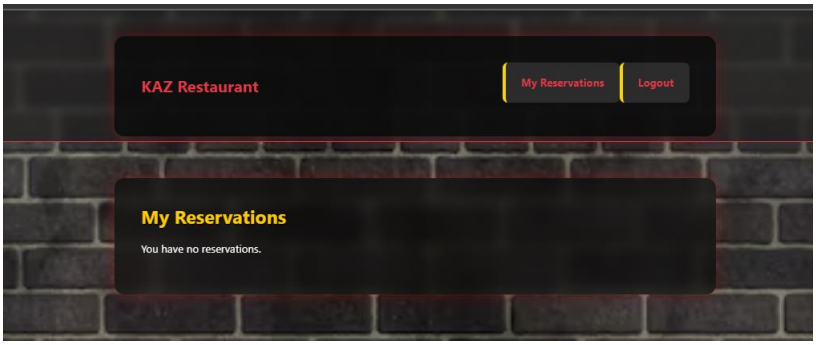


CANCELLING RESERVATION





Cancellation deletes the reservation from my reservations page



LOGOUT PAGE

FUTURE ENHANCEMENTS

1. **Full-Fledged Payment Gateway Integration**
   In the future, the system can be enhanced with real payment gateway integration (e.g., Credit Card or EasyPaisa) to allow secure online transactions at the time of booking.

CONCLUSION

The KAZ Restaurant Reservation System successfully demonstrates the application of database normalization, SQL triggers, and Laravel MVC architecture to build a dynamic reservation system. It handles reservations, cancellations, and service payments efficiently. With further enhancements like online payments and automated notifications, this system can serve as a robust solution for real-world restaurant operations.

DEPLOYMENT LINK

https://mediumorchid-worm-105955.hostingersite.com/

REFERENCES

[1] Perplexity, "Perplexity AI," *www.perplexity.ai*, 2025. https://www.perplexity.ai/

[2] A. Kumar, "Restaurant Table Booking System project in PHP | online Restaurant Table Booking project," *PHPGurukul*, Jun. 05, 2023. https://phpgurukul.com/restaurant-table-booking-system-using-php-and-mysql/

[3] "How to Design a Database for Online Restaurant Reservation and Food Delivery," *GeeksforGeeks*, Mar. 12, 2024. https://www.geeksforgeeks.org/how-to-design-a-database-for-online-restaurant-reservation-and-food-delivery/

[4] "ER Diagram for Restaurant reservation system | Creately," *Creately.com*, 2025. https://creately.com/diagram/example/il8aa6292/er-diagram-for-restaurant-reservation-system (accessed May 24, 2025).