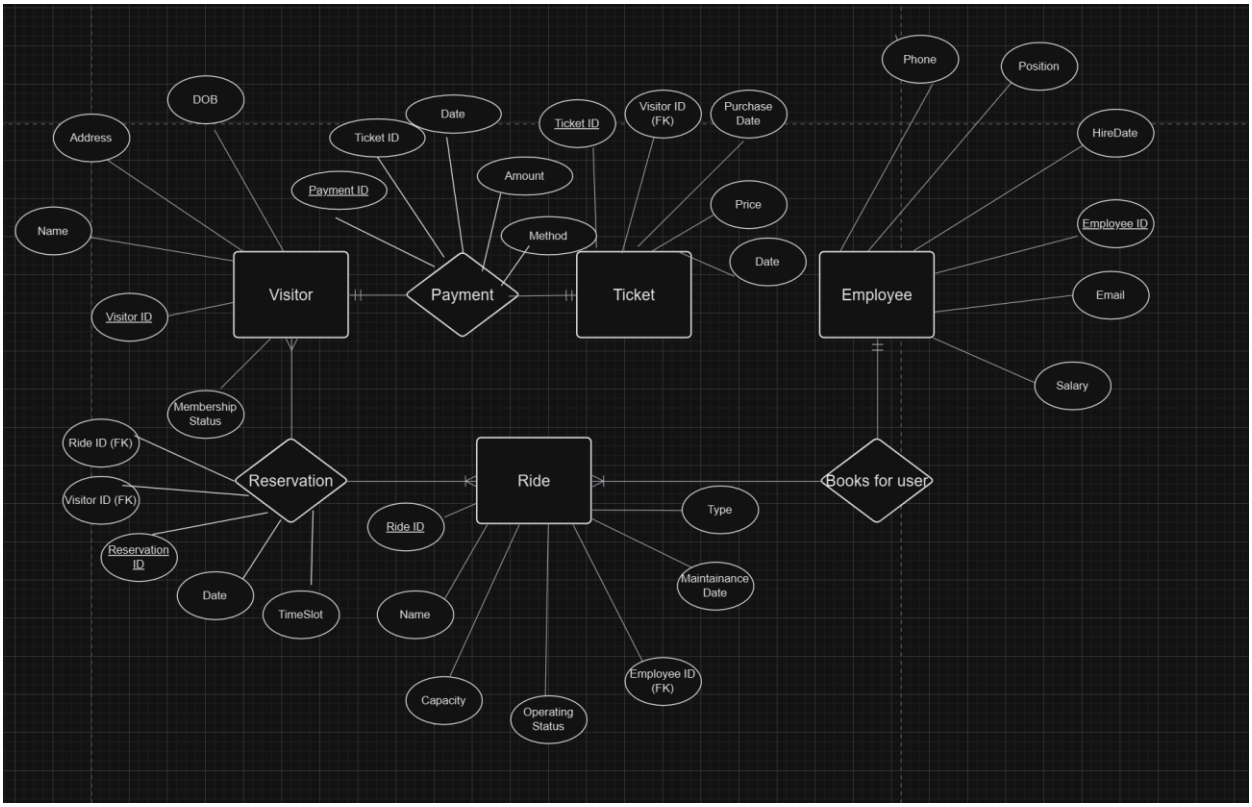
DBMS LAB PROJECT

AMUSEMENT THEME PARK MANAGEMENT SYSTEM

U23CS007

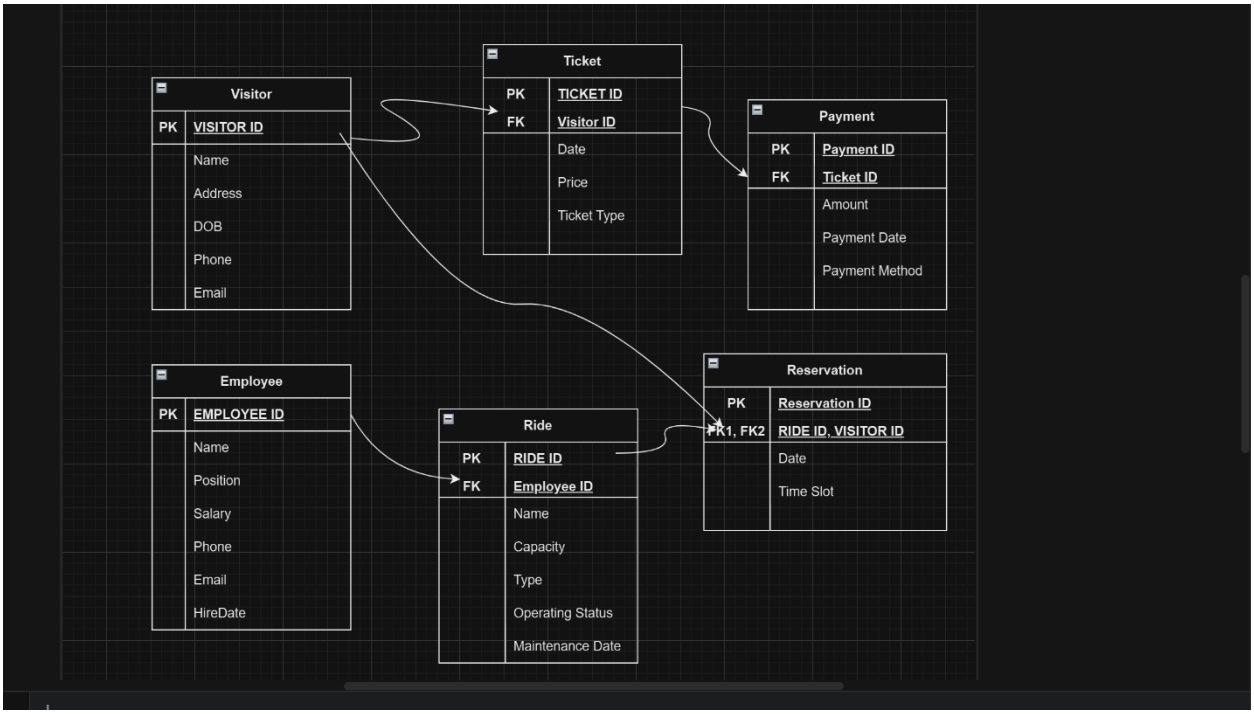
ZALAK NILESHKUMAR PANDIT

1. E-R Model

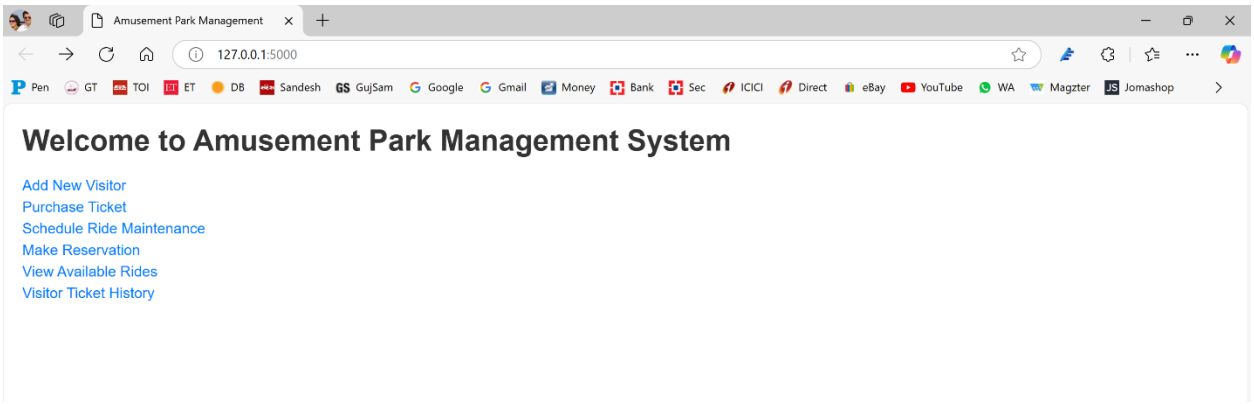


The database has 5 relations:

2. Relational Model:

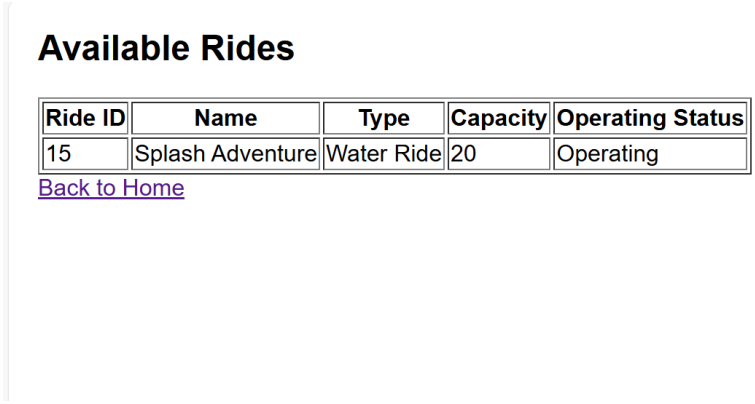


When you enter our app/website, these are the functionalities you get:



Two views are created

1. Ride View

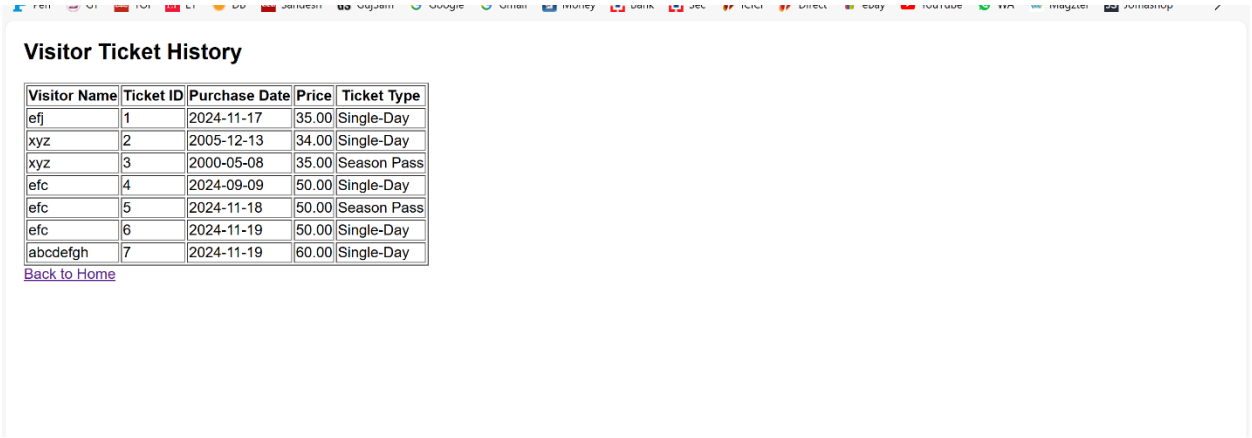


Available Rides

Ride ID	Name	Type	Capacity	Operating Status
15	Splash Adventure	Water Ride	20	Operating

Back to Home

3. View Ticket History



Visitor Ticket History

Visitor Name	Ticket ID	Purchase Date	Price	Ticket Type
efj	1	2024-11-17	35.00	Single-Day
xyz	2	2005-12-13	34.00	Single-Day
xyz	3	2000-05-08	35.00	Season Pass
efc	4	2024-09-09	50.00	Single-Day
efc	5	2024-11-18	50.00	Season Pass
efc	6	2024-11-19	50.00	Single-Day
abcdefgh	7	2024-11-19	60.00	Single-Day

Back to Home

I used 3 procedures:

1. Add New Visitor
2. Purchase Ticket
3. Schedule Ride Maintenance

1. Add new Visitor


```
DELIMITER //
CREATE PROCEDURE AddNewVisitor (
    IN p_Name VARCHAR(100),
    IN p_Address VARCHAR(255),
    IN p_Phone VARCHAR(15),
    IN p_Email VARCHAR(100),
    IN p_DateOfBirth DATE,
    IN p_MembershipStatus ENUM('Regular', 'Silver', 'Gold', 'Platinum')
)
BEGIN
    INSERT INTO Visitor (Name, Address, Phone, Email, DateOfBirth, MembershipStatus)
    VALUES (p_Name, p_Address, p_Phone, p_Email, p_DateOfBirth, p_MembershipStatus);
END //

DELIMITER;
```

2. Purchase Ticket


```
0
1     DELIMITER //
2     CREATE PROCEDURE RecordTicketPurchase(
3         IN p_VisitorID INT,
4         IN p_PurchaseDate DATE,
5         IN p_Price DECIMAL(8,2),
6         IN p_TicketType ENUM('Single-Day', 'Season Pass')
7     )
8     BEGIN
9         INSERT INTO Ticket (VisitorID, PurchaseDate, Price, TicketType)
0         VALUES (p_VisitorID, p_PurchaseDate, p_Price, p_TicketType);
1     END //
2     DELIMITER ;
3
4     DELIMITER //
5
```

3. Schedule Ride Maintenance

```
114     DELIMITER //
115
116 •   CREATE PROCEDURE ScheduleRideMaintenance(
117         IN p_RideID INT,
118         IN p_MaintenanceDate DATE
119     )
120     BEGIN
121         UPDATE Ride
122         SET OperatingStatus = 'Closed', MaintenanceDate = p_MaintenanceDate
123         WHERE RideID = p_RideID;
124     END //
125
126     DELIMITER ;
127
```

2 Triggers are also created:

1. After Ticket Insert
2. Before Reservation Insert

```
127
128     DELIMITER //
129 •   CREATE TRIGGER AfterTicketInsert
130     AFTER INSERT ON Ticket
131     FOR EACH ROW
132     BEGIN
133         INSERT INTO Payment (TicketID, Amount, PaymentDate, PaymentMethod)
134         VALUES (NEW.TicketID, NEW.Price, NEW.PurchaseDate, 'Credit Card');
135     END //
136
```

```
138 CREATE TRIGGER BeforeReservationInsert
139 BEFORE INSERT ON Reservation
140 FOR EACH ROW
141 BEGIN
142     DECLARE current_reservations INT;
143     DECLARE ride_capacity INT;
144
145     SELECT COUNT(*) INTO current_reservations
146     FROM Reservation
147     WHERE RideID = NEW.RideID AND ReservationDate = NEW.ReservationDate AND TimeSlot = NEW.TimeSlot;
148
149     SELECT Capacity INTO ride_capacity
150     FROM Ride
151     WHERE RideID = NEW.RideID;
152
153     IF current_reservations >= ride_capacity THEN
154         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ride capacity exceeded for the selected time slot.';
155     END IF;
156 END //
157
158 select * from Visitor;
159
160
```

The code for the Database and Python file (
for backend and connecting MySQL and frontend)

Is attached