DATABASE DESIGN REPORT

Group Members:

Baja, Arlene Joy

Mabini, Anjo

Pabilonia, Karylle

Perez, Froilan

Privaldos, Eugene

Fajardo, Zackery Alline

TABLE OF CONTENTS

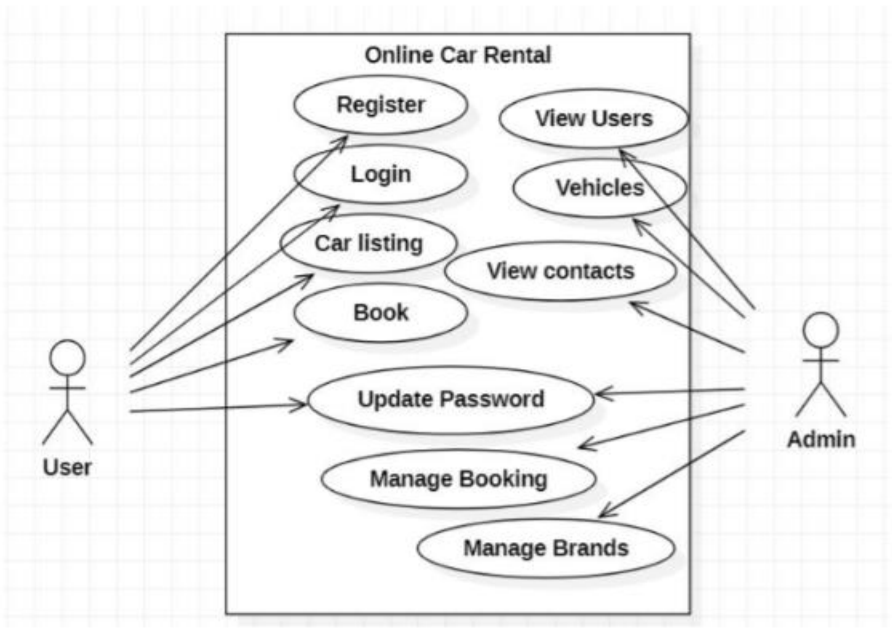
INTRODUCTION

We have chosen to produce a Car Rental system. In our system, Customer can rent a car based on make and a model. Our system provides customer to have different pick-up and drop off locations and will impose late fee if the rental car is returned beyond the return date and time.

A single car renting or reservation company needs to keep track of records of their cars and also of the people that will be renting the car. Our world has become a place where there is a lot of technological development. Every single thing done physically has been transformed into computerized form. Here goes the Car Renting System, with the use of the latest technologies used to develop a system, this system will be fast and reliable for the company. Car renting or reservation is very important for every people who are planning to travel across their city because of different occasions. Therefore, it is proposed to have a system that can be used to provide booking and management to make easier for both of them.

So for the system, The system will be build using the latest Web Development tools such as PHP and Microsoft SQL. The system will be getting the data of the customer's details, car owner details and the car details that they will rent. The system will help them by find the car rental base on the criteria that the customer like. This system primarily aims to secure the integrity of customer's data and also the car owner.

USER CASE DIAGRAM



REQUIREMENTS

A. Car rental agency should have collection of cars.
B. Customer, based on his location and car category preferences, rents a car.
C. Based on his location and car category preferences, list of cars available to rent will be shown along with available date and time (from and to).
D. Customer will select a car from the suggestions and should be able to reserve it for rent.
E. Billing is generated when a car is returned.
F. Once the car is returned it becomes available for the booking.
G. Car price is decided or made by the car owner.

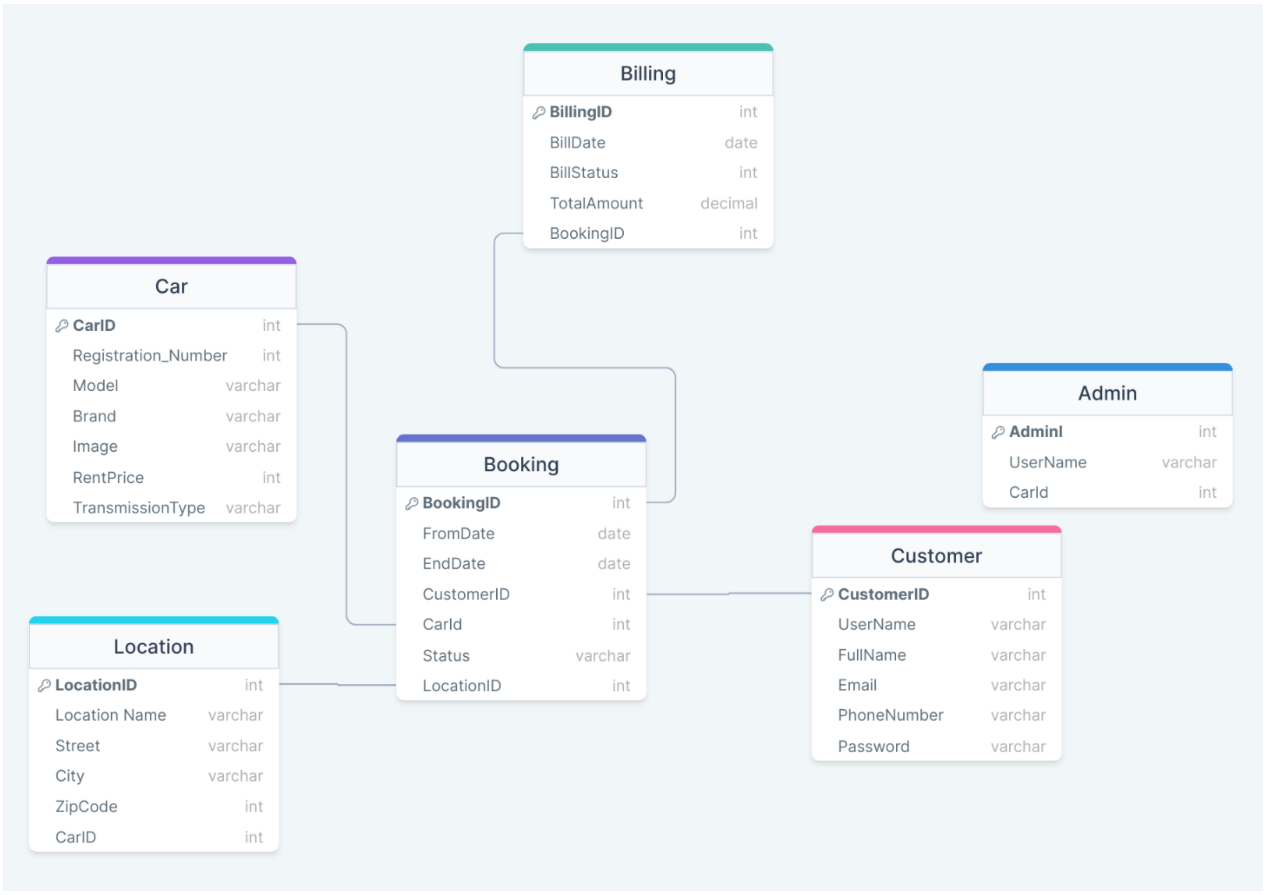
ENTITIES

- **Customer**
  - Customer will be the one who is using car rental system for reserving a car. He can be a member of the system or a non-member of the system. Member of the system will have Customer ID (this is auto generated). This entity stores information about the customer, such as name, username that they wanted, email, and phone number.
- **Car**
  - Car entity will have list of cars available in the system. Each car will be associated with a car category and car will have attributes like make, model, image, transmission type and registration number. Car will also have separate flag to check the availability of the car (will be in another table).
- **Location**
  - Location entity here denotes the pickup and drop off location of the car. Customer can pick up the car from the particular location and can have same or different drop off location. Location will have attributes like Location id, name, street, city, and zip code.
- **Booking**
  - Each car reservation will be monitored in the entity called booking. Booking will have attributes like booking id, from date and time of booking and due return date and time and actual return date and time of the booking, and booking status.
- **Billing**
  - When a customer returns a car, a bill will be generated on the particular booking. Billing have attributes like Bill ID, bill date, bill status, and total amount.

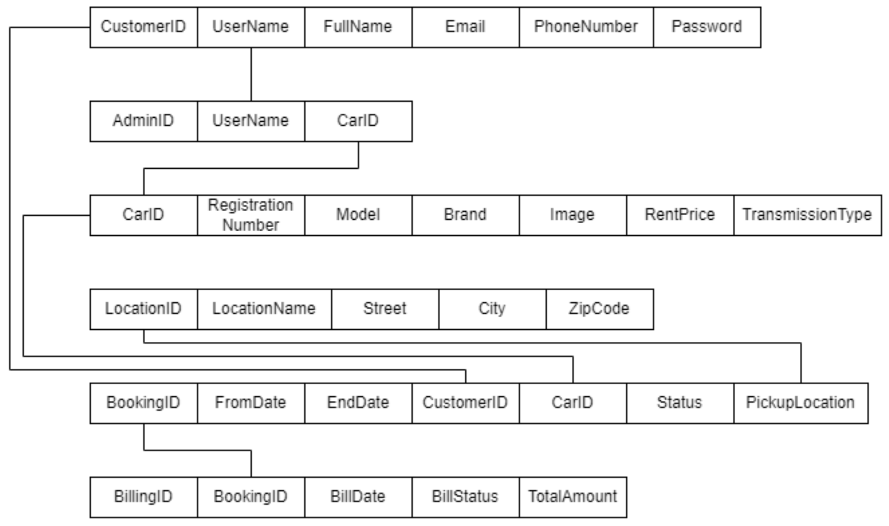
RELATIONS

1. **Customer/Admin to Car**
   - ➢ Customer can let their own car to be rented.
2. **Car to Location**
   - ➢ Customer will be picking up or dropping the car in a particular location. So, cars will be present at a location.
3. **Booking to Billing**
   - ➢ Once customer returns a car bill will be generated for each booking
4. **Booking to Location**
   - ➢ Customer can drop off rental car in a particular location. The relation name is ‘Pickup location’.
5. **Customer to Car to Booking**
   - ➢ Customer will select car for rent. So the customer will be related to the both car and the booking.

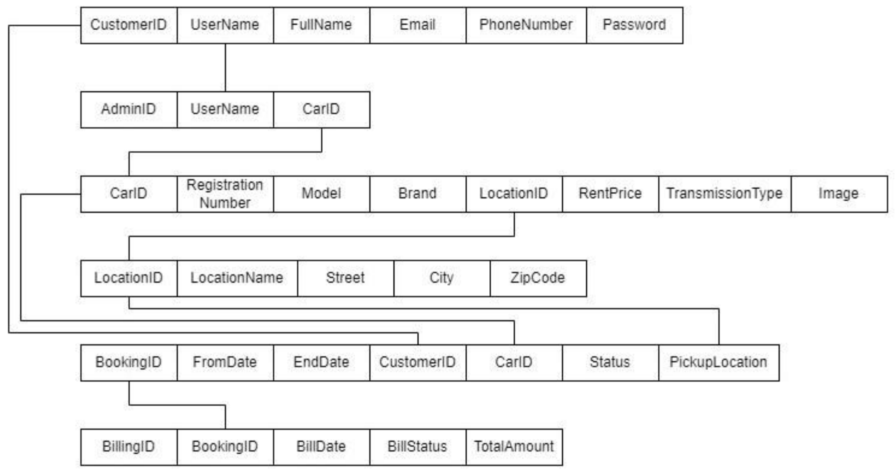
ERD



RELATIONAL SCHEMA



Plan 1



Plan 2

SQL STATEMENTS

```sql
CREATE DATABASE RentCar2; --Can have different name

CREATE TABLE Customer(
    CustomerID INT NOT NULL
    CONSTRAINT PK_CustomerID PRIMARY KEY IDENTITY(1,1),
    UserName VARCHAR(255) NOT NULL,
    FullName VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    PhoneNumber VARCHAR(255) NOT NULL,
    [Password] VARCHAR(125) NOT NULL,
);

CREATE TABLE [Location](
    LocationID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    LocationName VARCHAR(255) NOT NULL,
    Street VARCHAR(255) NOT NULL,
    City VARCHAR(255) NOT NULL,
    ZipCode INT NOT NULL
);


CREATE TABLE Car(
    CarID INT NOT NULL
    CONSTRAINT PK_CarID PRIMARY KEY IDENTITY(1,1),
    Registration_Number BIGINT NOT NULL,
    Model VARCHAR(255) NOT NULL,
    Brand VARCHAR(255) NOT NULL,
    CarImage VARCHAR(255) NOT NULL,
    RentPrice DECIMAL(10,2) NOT NULL,
    TransmissionType VARCHAR(255) NOT NULL,

);

CREATE TABLE Booking(
    BookingID INT NOT NULL
    CONSTRAINT PK_BookingID PRIMARY KEY IDENTITY(1,1),
    FromDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    CustomerID INT NOT NULL,
    CarID INT NOT NULL,
    [Status] VARCHAR(255) NOT NULL,
    LocationID INT NOT NULL,
  FOREIGN KEY(CustomerID)
    REFERENCES [Customer] (CustomerID),
  FOREIGN KEY(CarID)
    REFERENCES [Car] (CarID),
  FOREIGN KEY(LocationID)
    REFERENCES [Location] (LocationID),
);


CREATE TABLE [Admin](
    AdminID INT NOT NULL
    CONSTRAINT PK_AdminID PRIMARY KEY IDENTITY(1,1),
    UserName VARCHAR(255) NOT NULL,
  Password VARCHAR(255) NOT NULL,
    CarID INT NOT NULL,
  FOREIGN KEY(CarID)
    REFERENCES [Car] (CarID),
);


CREATE TABLE Billing(
    BillingID INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    BillDate DATE NOT NULL,
    BillStatus INT NOT NULL,
    TotalAmount DECIMAL(10,2) NOT NULL,
    BookingID INT NOT NULL,

  FOREIGN KEY(BookingID)
    REFERENCES [Booking] (BookingID),
);
```

STORED PROCEDURE

```sql
--DISPLAYING CAR
CREATE PROCEDURE SP_LIST_CAR
AS
SELECT * FROM Car;
--INSERTING CAR
CREATE PROCEDURE SP_INSERT_CAR
    @p_Registration_Number AS BIGINT
    ,@p_Model AS VARCHAR(255)
    ,@p_Brand AS VARCHAR(255)
    ,@p_TransmissionType AS VARCHAR(255)
    ,@p_RentPrice AS DECIMAL(10,2)
    ,@p_CarImage AS VARCHAR(255)
AS
INSERT INTO Car
    (Registration_Number, Model, Brand, TransmissionType, RentPrice,
CarImage)
VALUES
    (@p_Registration_Number, @p_Model, @p_Brand, @p_TransmissionType,
@p_RentPrice, @p_CarImage)

--DELETING A CAR
CREATE PROCEDURE SP_DELETE_CAR
    @p_CarID INT
AS
DELETE FROM Car
    WHERE CarID = @p_CarID

--INSERTING USER
CREATE PROCEDURE SP_INSERT_CUSTOMER
    @p_UserName AS VARCHAR(255)
    ,@p_FullName AS VARCHAR(255)
    ,@p_Email AS VARCHAR(255)
    ,@p_PhoneNumber AS VARCHAR(255)
    ,@p_Password AS VARCHAR(255)
AS
INSERT INTO Customer
    (UserName, FullName, Email, PhoneNumber, [Password])
VALUES
    (@p_UserName, @p_FullName, @p_Email, @p_PhoneNumber, @p_Password)

--USER LOGIN
CREATE PROCEDURE SP_CUSTOMER_LOGIN
    @p_Email AS VARCHAR(255)
AS
SELECT * FROM Customer WHERE Email = @p_Email

--USER EDIT/UPDATE
CREATE PROCEDURE SP_CUSTOMER_UPDATE
    @p_UserName AS VARCHAR(255)
    ,@p_FullName AS VARCHAR(255)
    ,@p_Email AS VARCHAR(255)
    ,@p_PhoneNumber AS VARCHAR(255)
    ,@p_CustomerID AS INT
AS
UPDATE Customer
```

```sql
SET UserName = @p_UserName,
      FullName = @p_FullName,
      Email = @p_Email,
      PhoneNumber = @p_PhoneNumber
  WHERE CustomerID = @p_CustomerID;


--DISPLAY USER
CREATE PROCEDURE SP_CUSTOMER_DISPLAY
    @p_UserId AS INT
AS
SELECT * FROM Customer WHERE CustomerID = @p_UserId

--DELETE USER
CREATE PROCEDURE SP_CUSTOMER_DELETE
    @p_CustomerID INT
AS
DELETE FROM Customer
    WHERE CustomerID = @p_CustomerID
```

JOIN TABLE

```sql
--JOINING TABLE BOOKING AND BILLING
SELECT *
FROM Booking
INNER JOIN Billing ON Booking.[BookingID] = Billing.[BookingID]
WHERE Booking.BookingID = 32;


--JOINING TABLE CUSTOMER AND BOOKING
SELECT *
FROM Booking
INNER JOIN Billing ON Booking.[BookingID] = Billing.[BookingID]
WHERE Booking.CustomerID = 3;
```