CAR RENTAL SYSTEM

GROUP 2

Elvis Lutomia-1049070

George Mulu-1049459

Maryann Chege-1049085

Maxwell Muguna-1049417

Mitchel Wawira-1049080

NOVEMBER 2024

Introduction

Overview

The purpose of this project was to create a system that is able to handle customer and car data. In this way booking of cars is streamlined. The system is also to help in processing payments, manage car rental operations, track car inventory and generate business reports.

Rationale

The proposed car rental system aims to modernize and streamline car rental operations through a comprehensive digital solution. This system will replace the current manual processes that have become inefficient and error prone.

A manual car rental system faces the following challenges:

- Manual booking has led to high redundancy of data as there are cases of double bookings
- Paper based record keeping causes data inconsistencies
- Delayed revenue reporting and financial tracking.

The benefits of this system will be:

Operational

- Reduced booking errors
- Real time vehicle availability tracking
- Automated maintenance alerts
- Streamlined payment processing

Customer

- Enhanced customer experience
- Improved communication

Business

- Reduced operational costs
- Improved decision making through analytics
- Better resource allocation

The scope the system will encompass is vehicle inventory management, booking and reservation system, payment processing and report generation

Objectives

The primary purpose of this system is to :

- Automate rental operations,
- Improve customer service delivery,
- Enhance fleet management,
- Streamline financial operations
- Enable data driven decision making

System Design

ER diagram

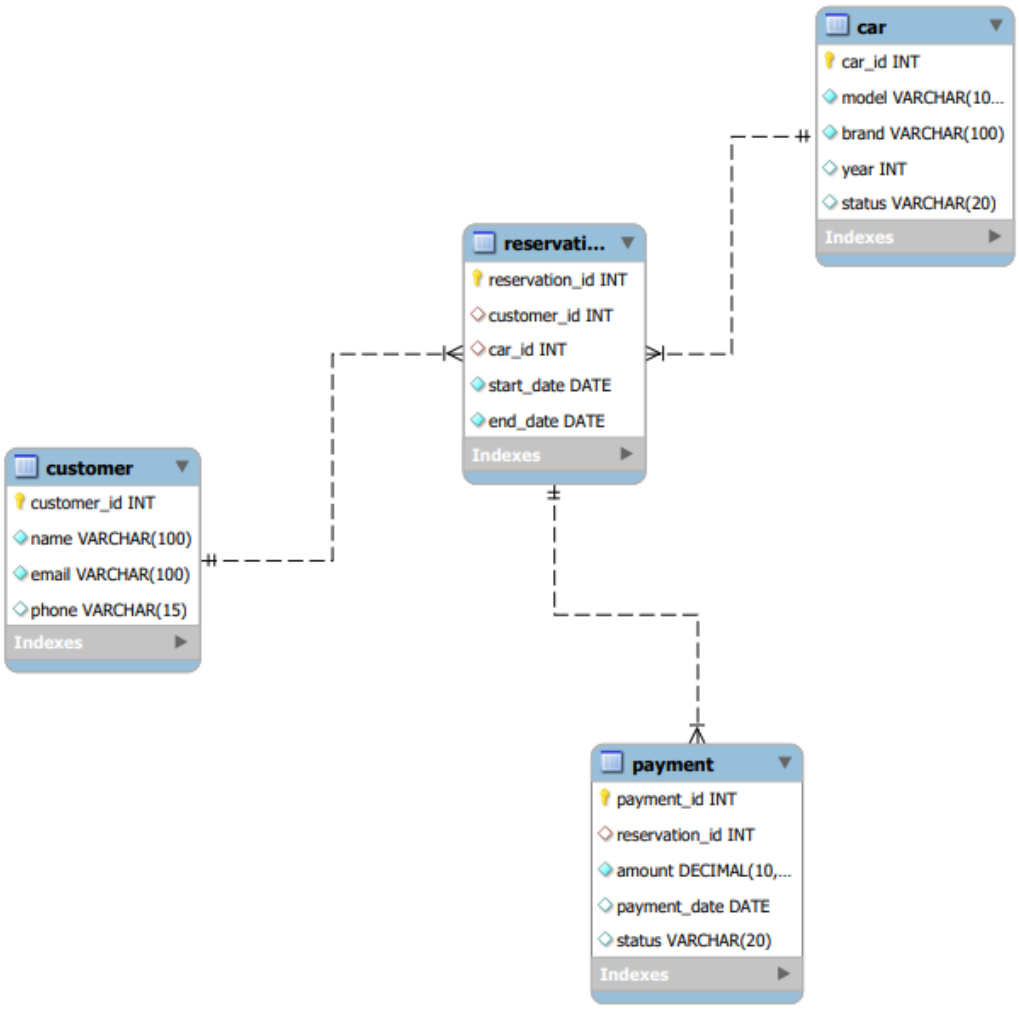

Figure 1 ER diagram

Table structures

```sql
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(15)
);

CREATE TABLE car (
    car_id INT PRIMARY KEY,
    model VARCHAR(100) NOT NULL,
    brand VARCHAR(100) NOT NULL,
    year INT NOT NULL,
    status VARCHAR(20) NOT NULL
);

CREATE TABLE reservation (
    reservation_id INT PRIMARY KEY,
    customer_id INT NOT NULL,
    car_id INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    FOREIGN KEY (car_id) REFERENCES car(car_id)
);

CREATE TABLE payment (
    payment_id INT PRIMARY KEY,
    reservation_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    payment_date DATE NOT NULL,
    status VARCHAR(20) NOT NULL,
    FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)
);
```

Figure 2 Table structure

SQL SCHEMA

```sql
CREATE TABLE `car` (
  `car_id` INT NOT NULL AUTO_INCREMENT,
  `model` VARCHAR(100) NOT NULL,
  `brand` VARCHAR(100) NOT NULL,
  `year` INT NULL DEFAULT NULL,
  `status` VARCHAR(20) NULL DEFAULT 'available',
  PRIMARY KEY (`car_id`))

CREATE TABLE `customer` (
  `customer_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `phone` VARCHAR(15) NULL DEFAULT NULL,
  PRIMARY KEY (`customer_id`),
  UNIQUE INDEX `email` (`email` ASC) VISIBLE)

CREATE TABLE `reservation` (
  `reservation_id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT NULL DEFAULT NULL,
  `car_id` INT NULL DEFAULT NULL,
  `start_date` DATE NOT NULL,
  `end_date` DATE NOT NULL,
  PRIMARY KEY (`reservation_id`),
  INDEX `customer_id` (`customer_id` ASC) VISIBLE,
  INDEX `car_id` (`car_id` ASC) VISIBLE,
  FOREIGN KEY (`customer_id`)
    REFERENCES `customer` (`customer_id`)
    ON DELETE CASCADE,
  FOREIGN KEY (`car_id`)
    REFERENCES `car` (`car_id`)
    ON DELETE SET NULL)

```

Figure 3 Sql Schema

```
33  CREATE TABLE `payment` (
34    `payment_id` INT NOT NULL AUTO_INCREMENT,
35    `reservation_id` INT NULL DEFAULT NULL,
36    `amount` DECIMAL(10,2) NOT NULL,
37    `payment_date` DATE NULL DEFAULT NULL,
38    `status` VARCHAR(20) NULL DEFAULT 'pending',
39    PRIMARY KEY (`payment_id`),
40    INDEX `reservation_id` (`reservation_id` ASC) VISIBLE,
41    FOREIGN KEY (`reservation_id`)
42      REFERENCES `reservation` (`reservation_id`)
43      ON DELETE CASCADE)
44
```

Figure 4 Continuation of sql schema

Implementation

Create queries

To create tables you'll use the following create queries:

```sql
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(15)
);

CREATE TABLE car (
    car_id INT PRIMARY KEY,
    model VARCHAR(100) NOT NULL,
    brand VARCHAR(100) NOT NULL,
    year INT NOT NULL,
    status VARCHAR(20) NOT NULL
);

CREATE TABLE reservation (
    reservation_id INT PRIMARY KEY,
    customer_id INT NOT NULL,
    car_id INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    FOREIGN KEY (car_id) REFERENCES car(car_id)
);

CREATE TABLE payment (
    payment_id INT PRIMARY KEY,
    reservation_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    payment_date DATE NOT NULL,
    status VARCHAR(20) NOT NULL,
    FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)
);
```

Figure 5 Create queries

Read Queries

To generate various reports based on data input the in the tables you have a variety of select queries below you can use to generate them.

```sql
-- Get customer details
SELECT * FROM customer;

--List all available cars
SELECT car_id, brand, model, year
FROM car
WHERE status = 'available';

--Get all active reservations with customer and car details
SELECT
    r.reservation_id,
    c.name AS customer_name,
    car.brand,
    car.model,
    r.start_date,
    r.end_date
FROM reservation r
JOIN customer c ON r.customer_id = c.customer_id
JOIN car ON r.car_id = car.car_id
WHERE r.end_date >= CURRENT_DATE;

--Get payment history for a specific customer
SELECT
    p.payment_id,
    p.amount,
    p.payment_date,
    p.status,
    car.brand,
    car.model
FROM payment p
JOIN reservation r ON p.reservation_id = r.reservation_id
JOIN car ON r.car_id = car.car_id
WHERE r.customer_id = [customer_id];
```

Figure 6 Read queries

```sql
-- Find total revenue by car
SELECT
    car.brand,
    car.model,
    SUM(p.amount) as total_revenue
FROM car
JOIN reservation r ON car.car_id = r.car_id
JOIN payment p ON r.reservation_id = p.reservation_id
WHERE p.status = 'completed'
GROUP BY car.car_id, car.brand, car.model;

-- Get customers with active reservations
SELECT DISTINCT
    c.customer_id,
    c.name,
    c.email,
    c.phone
FROM customer c
JOIN reservation r ON c.customer_id = r.customer_id
WHERE r.end_date >= CURRENT_DATE;

--Find overdue payments
SELECT
    p.payment_id,
    c.name AS customer_name,
    p.amount,
    p.payment_date,
    p.status
FROM payment p
JOIN reservation r ON p.reservation_id = r.reservation_id
JOIN customer c ON r.customer_id = c.customer_id
WHERE p.status = 'pending'
AND p.payment_date < CURRENT_DATE;
```

Figure 7 Read queries continuation

Update Queries

```sql
--Update a customer's information
UPDATE customer
SET name = 'Max',
    email = 'max@gmail.com',
    phone = '0712345678'
WHERE customer_id = 2;

--Update a car's Dates
UPDATE car
SET status = 'Unavailable'
WHERE car_id = 4;

--update a Reservation's Status
UPDATE reservation
SET start_date = '2024-01-10',
    end_date = '2024-01-25'
WHERE reservation_id = 3;

--Update a payment status
UPDATE payment
SET status = 'Paid'
WHERE payment_id = 4;

-- Update a Payment amount
UPDATE payment
SET amount = 15000
WHERE payment_id = 6;

--Update a customer assigned to Reservation
UPDATE reservation
SET customer_id = 2
WHERE reservation_id = 5;

```

To update sets of data in the tables you can use the queries above.

Figure 8 Update queries

Delete Queries

To delete specific entries made in the tables you can use the queries above:

```sql
-- Delete from reservation
-- Delete a specific reservation
DELETE FROM reservation WHERE reservation_id = 10;

-- Delete all reservations for a specific customer
DELETE FROM reservation WHERE customer_id = 1;

-- Delete all reservations for a specific car
DELETE FROM reservation WHERE car_id = 2;

-- Delete from customer
-- Delete a specific customer
DELETE FROM customer WHERE customer_id = 1;

-- Delete from car
-- Delete a specific car
DELETE FROM car WHERE car_id = 1;

```

Figure 9 Delete queries

Advanced queries

Below are advanced queries to output specific data based on your needs from the different tables

```sql
-- Total reservation days by a car
SELECT c.car_id,
       c.model,
       c.brand,
       SUM(DATEDIFF(r.end_date, r.start_date) + 1) AS total_reserved_days
FROM car c
LEFT JOIN reservation r ON c.car_id = r.car_id
GROUP BY c.car_id, c.model, c.brand
ORDER BY total_reserved_days DESC;

-- Cars reserved by multiple customers
SELECT r.car_id,
       c.model,
       c.brand,
       COUNT(DISTINCT r.customer_id) AS unique_customers
FROM reservation r
JOIN car c ON r.car_id = c.car_id
GROUP BY r.car_id, c.model, c.brand
HAVING unique_customers > 1;

-- Customers without reservations
SELECT c.customer_id,
       c.name,
       c.email,
       c.phone
FROM customer c
LEFT JOIN reservation r ON c.customer_id = r.customer_id
WHERE r.reservation_id IS NULL;
```

Figure 10 Procedure

```
        -- Update car status
        UPDATE car SET status = 'reserved' WHERE car_id = p_car_id;

        COMMIT;
        SELECT 'Reservation created successfully' AS message;
    ELSE
        ROLLBACK;
        SELECT 'Car not available for selected dates' AS message;
    END IF;
END //

-- Cancel reservation
CREATE PROCEDURE CancelReservation(
    IN p_reservation_id INT
)
BEGIN
    START TRANSACTION;

    UPDATE car c
    JOIN reservation r ON c.car_id = r.car_id
    SET c.status = 'available'
    WHERE r.reservation_id = p_reservation_id;

    DELETE FROM reservation WHERE reservation_id = p_reservation_id;

    COMMIT;
END //

DELIMITER ;
```

Figure 11 Continuation of procedure

Testing and validation

The database underwent testing to ensure that it works properly. In our test results we found that we were able to add new entries in the database tables. We could also delete and update the data. All queries processed data in record time proving to be reliable. We then proceeded in generating reports based on a number of criteria and got our desired output. We can therefore confirm that the database is fully functional and streamlined to be a efficient and thorough.

Conclusion and Recommendations

The proposed car rental system aims to modernize and streamline vehicle rental operations through an extensive digital solution. The current manual processes have become inefficient and error prone. This has resulted in data redundancy, poor customer service, inaccurate financial analytics and poor fleet management.

The primary purpose of this system is to automate rental operations, improve customer service, enhance data driven decisions through accurate financial analytics and enhance fleet management. The expected benefits include reduced booking errors, real-time vehicle availability tracking, automated maintenance alerts, streamlined payment processing, and enhanced customer experience.

From a business perspective the system is projected to increase revenue through better fleet utilization, reduce operational costs and enhance a company’s competitive advantage

Future improvements to the system to enhance its capabilities and stay ahead of the competition include mobile app development, scheduled maintenance and insurance management

This will help in continued improvement in customer satisfaction and drive sustainable business growth

Appendices