



# Car-rental-system

22.11.2023

## Group no. 9

Vedant Agarwal (2112128)

Deep Saikia (2112129)

Tolemy Kashyap (2112130)

Raviranjan Patel (2112131)

Pragya Annesha Baruah (2112132)

Praptipal Kaur (2112133)

## Title

### **Web-Based Car Rental System: A Comprehensive Solution for Urban Mobility Optimization**

## Problem

Inefficient and fragmented urban mobility necessitates the development of an integrated and user-centric car rental system to enhance accessibility, streamline reservation processes, and contribute to sustainable transportation solutions.

## Problem Description

In the evolving landscape of urban mobility, individuals often face challenges in accessing flexible and efficient transportation solutions. The absence of a streamlined and user-friendly car rental system contributes to inconvenience, inefficiency, and environmental concerns. Current car rental platforms lack comprehensive features, leading to fragmented user experiences, difficulties in reservation management, and a lack of integration with emerging technologies. As a result, users are left with limited options, and the potential for optimizing the utilization of available vehicles remains untapped.

Addressing these challenges requires the development of a robust and innovative car rental system that seamlessly integrates with modern technologies, enhances user experience, optimizes fleet management, and contributes to sustainable urban mobility. The system should provide a user-friendly interface for easy booking, real-time vehicle availability, transparent pricing, and efficient vehicle tracking. Additionally, it should incorporate advanced features such as predictive maintenance, seamless payment options, and integration with smart city infrastructure for improved traffic management.

The aim is to develop a comprehensive car rental solution that not only meets the immediate transportation needs of users but also aligns with the broader goals of sustainability, cost-effectiveness, and adaptability to future technological advancements. The successful implementation of such a system will contribute to the evolution of urban mobility, offering a reliable and eco-friendly alternative to traditional transportation methods.

## Key Challenges

1. **Fleet Management:** Managing a diverse fleet of vehicles, including maintenance, tracking, and allocation of cars, can be complex and time-consuming.
2. **Reservation Handling:** Handling customer reservations, scheduling, and ensuring the availability of vehicles can lead to inefficiencies and errors.
3. **Customer Management:** Tracking customer information, preferences, and maintaining a database for loyalty programs can be a daunting task.
4. **Pricing and Billing:** Setting competitive pricing, calculating rental charges, and handling billing and invoicing can be error-prone.
5. **Vehicle Maintenance:** Regular maintenance and service tracking are essential for ensuring the safety and reliability of the rental fleet.
6. **Inventory Management:** Managing spare parts, tires, and other inventory items for vehicle maintenance can be challenging.
7. **Customer Experience:** Providing a seamless and user-friendly booking and rental experience is essential for customer satisfaction.

## Approach of solving

This report presents a detailed exploration of the design, development, and implementation of a web-based car rental system aimed at addressing the challenges inherent in current urban mobility frameworks. Leveraging a MySQL database, the system provides a multifaceted solution with distinct user roles—admin, office, and user—each contributing to the overall efficiency and effectiveness of the platform. The reports delineates the functionalities of each role, elucidates the database architecture, and expounds on the seamless integration of user-friendly interfaces, fostering a comprehensive and sustainable urban mobility solution.

### I. Introduction:

Urban mobility has become a critical concern in contemporary societies, necessitating innovative solutions to optimize transportation efficiency, reduce environmental impact, and enhance user experience. Urban mobility challenges are escalating with the growing population and increasing urbanization. The proposed web-based car rental system serves as a strategic response to the inefficiencies in current transportation frameworks. By offering a comprehensive solution, the

system aims to optimize transportation, reduce environmental impact, and elevate the overall user experience. The integration of distinct user roles, underpinned by a MySQL database, forms the basis of this solution, ensuring scalability, security, and efficient data management.

## II. System Architecture:

The foundation of the proposed car rental system lies in a robust MySQL database, ensuring data integrity, security, and scalability. The system incorporates three distinct user roles:


- **Admin:** The central authority governing the entire system. Admin has the privilege to manage users, offices, and system configurations. This role ensures the seamless functioning of the platform and oversees the integration of new offices into the system.
- **Office:** Offices, representing car rental entities, have the ability to sign up on the website, post details of their available cars, and manage their inventory. The integration of various offices into the system creates a diverse pool of options for users, enhancing the overall user experience.
- **User:** End-users can sign up, search for available cars based on specific dates and preferences, and make reservations. The web platform acts as a mediator between users and offices, providing a centralized hub for car rental transactions.

The three primary user roles—Admin, Office, and User—are designed to facilitate a hierarchical structure, with the Admin role acting as the central authority, overseeing system-wide operations. The dynamic interaction between these roles ensures a seamless and organized flow of data, contributing to the overall efficiency of the system..

## III. User Interaction:

The user experience is paramount in the success of the car rental system. Upon signing up, users gain access to a user-friendly interface that allows them to search for available cars based on specified dates and preferences. The system provides a comprehensive list of cars from various offices, enabling users to make informed choices. User interaction is a pivotal aspect of the car rental system, emphasizing a user-friendly interface to enhance accessibility and usability. The system employs sophisticated logic to present users with a diverse range of options, fostering an informed decision-making process. The user interface is designed to be intuitive and responsive, ensuring a positive and efficient experience.

## IV. Reservation Process:



Once a user selects a car, the reservation process is initiated. Users can choose to pay immediately or opt for deferred payment. Payment transactions are securely processed through the platform, and funds are directed to the respective office. This approach ensures financial transparency and trust in the system. The reservation process is a critical juncture in the user journey. Users, having selected a car, are presented with the option to pay immediately or defer payment. This flexibility accommodates varying user preferences and financial situations. The payment gateway ensures the confidentiality and integrity of financial transactions.

## V. Office Management:


Offices, representing car rental entities, are equipped with a dedicated dashboard for efficient management of their fleet. This dashboard allows offices to update information on car availability, review reservations, and access performance analytics. Effective communication channels between offices and users are embedded in the system, ensuring timely updates and facilitating a smooth exchange of information. The integration of analytics empowers offices to make data-driven decisions, contributing to the overall optimization of the car rental process.

In conclusion, the proposed web-based car rental system represents a significant stride towards overcoming challenges in urban mobility. By integrating user-centric design, robust database architecture the system addresses the inefficiencies of current transportation frameworks. The integration of distinct user roles, each with its specific functionalities, ensures a cohesive and efficient system that caters to the diverse needs of administrators, car rental offices, and end-users alike.

## Frontend:

Frontend design in the context of the proposed web-based car rental system is crucial for creating an intuitive and visually engaging user interface. It serves as the bridge between users and the system's functionalities, influencing the overall user experience. The frontend design focuses on translating the complexity of the system's architecture into a seamless, user-friendly interface. Through a well-crafted design, users, including administrators, offices, and end-users, can interact with the system effortlessly.

The frontend design begins with an emphasis on user roles, ensuring that each role has a distinct and purposeful interface. For administrators, this means a centralized and efficient dashboard for overseeing system operations. Offices are provided with a dedicated space to manage their fleet and reservations, while end-users experience a user-friendly platform for browsing, selecting, and reserving cars.



The design prioritizes responsiveness, ensuring a consistent and optimal experience across various devices. Clear navigation, visually appealing elements, and an intuitive layout contribute to a positive user experience. Integration of real-time updates and interactive features enhances engagement, providing users with up-to-date information on car availability and facilitating seamless transactions.

We are using the following technologies for implementing the frontend:

1. **HTML5:** Utilized to structure and present content, HTML5 forms the backbone of the frontend, ensuring semantic and organized layout for the car rental system.
2. **CSS3:** Applied for styling and layout enhancements, CSS3 enhances the visual appeal and responsiveness of the user interface, providing a seamless and aesthetic design.
3. **JavaScript:** Employs client-side scripting for dynamic functionalities, enhancing user interactivity and enabling real-time updates within the car rental system.
4. **Ajax:** Integrated for asynchronous data exchange with the server, Ajax optimizes the user experience by facilitating seamless, dynamic content retrieval without requiring a full page reload.


## Backend:

Now let's delve into the intricacies of the backend infrastructure powering a web-based car rental system. Employing a stack comprising Node.js, Express, JSON Web Tokens (JWT), Bcrypt, and MySQL, the backend is engineered for scalability, security, and efficient database management. This comprehensive overview explores the role of each component in handling user requests, utilizing MySQL DML queries for seamless data manipulation, and underscores the significance of backend technologies in ensuring a reliable and secure car rental platform.

### 1. Introduction:

The backend of a web-based car rental system plays a pivotal role in handling user requests, managing data, and ensuring the overall functionality and security of the platform. This note focuses on the technology stack employed in the backend, elucidating the synergy between Node.js, Express, JWT, Bcrypt, and MySQL. The integration of these technologies creates a robust and scalable foundation, ensuring a seamless user experience and efficient data management.

### 2. Node.js and Express:



Node.js serves as the runtime environment, providing a non-blocking, event-driven architecture that enhances the system's scalability. Paired with Express, a minimalist web application framework for Node.js, the backend benefits from streamlined route handling, middleware support, and a modular structure. This combination accelerates the development process and optimizes the handling of HTTP requests, making it well-suited for the real-time interactions inherent in a car rental system.

### **3. JSON Web Tokens (JWT):**

JWT is employed for secure and stateless user authentication. By generating JSON-encoded tokens, JWT enables the backend to verify the authenticity of user requests without the need for sessions or cookies. This lightweight and efficient authentication mechanism enhances the security of the car rental system, safeguarding user data and mitigating the risk of unauthorized access.

### **4. Bcrypt:**

Security is paramount in any web-based system, particularly when dealing with sensitive user information. Bcrypt is utilized for password hashing, ensuring that user passwords are stored securely in the database. Its adaptive hashing algorithm and salting mechanism add an extra layer of protection against common vulnerabilities, such as brute-force attacks, contributing to the overall robustness of the backend security infrastructure.

### **5. MySQL Database Access:**

MySQL is employed as the relational database management system (RDBMS), providing a structured and scalable solution for storing and retrieving data. The backend communicates with the MySQL database using Data Manipulation Language (DML) queries, facilitating the seamless manipulation of data based on user requests. This relational database structure ensures data integrity, consistency, and reliability, critical aspects in managing the dynamic information associated with car rentals.

### **6. Handling User Requests:**

User requests are managed through Express routes, each designed to handle specific functionalities such as user authentication, car availability queries, reservation processing, and more. The integration of Node.js allows for asynchronous processing, enabling the system to handle multiple concurrent user requests efficiently. JWT authentication ensures that only authorized users can access sensitive functionalities, enhancing the overall security of the system.

### **7. MySQL DML Queries:**

MySQL DML queries form the backbone of the backend's interaction with the database. These queries are strategically crafted to retrieve, insert, update, or delete data based on user actions. For instance, when a user searches for available cars on specific dates, the backend formulates DML queries to retrieve relevant data from the database. The use of parameterized queries enhances security by preventing SQL injection attacks, showcasing a meticulous approach to data manipulation.

## 8. Conclusion:

In conclusion, the backend infrastructure of the web-based car rental system, orchestrated through Node.js, Express, JWT, Bcrypt, and MySQL, embodies a comprehensive and sophisticated approach to handling user requests and managing data. The synergistic integration of these technologies ensures scalability, security, and efficiency. The use of MySQL DML queries exemplifies a meticulous strategy for manipulating data, underpinning the seamless and dynamic functionality of the car rental platform.

## Results:

### Database:

The database, designed and implemented in MySQL contains multiple tables as mentioned below.

#### 1. Admin Table:

- **Columns:**
  - email (VARCHAR(64)): Unique identifier for admin login.
  - password (VARCHAR(256)): Hashed password for admin login.
- **Constraints:**
  - admin\_pk (PRIMARY KEY): Primary key constraint on the email column.
- **Purpose:**
  - This table stores information about administrators who have access to manage the entire car rental system.

#### 2. Customer Table:

- **Columns:**
  - ssn (CHAR(6)): Social Security Number, serves as a unique identifier for customers.
  - fname (VARCHAR(32)): First name of the customer.
  - lname (VARCHAR(32)): Last name of the customer.
  - email (VARCHAR(64)): Unique email identifier for customer login.



- phone\_no (CHAR(11)): Unique phone number identifier for customer contact.
  - password (VARCHAR(256)): Hashed password for customer login.
  - wallet (REAL): Wallet balance for financial transactions.
  - **Constraints:**
    - customer\_pk (PRIMARY KEY): Primary key constraint on the ssn column.
  - **Purpose:**
    - Stores details of registered customers who can access and utilize the car rental services.
3. **Credit Card Table:**
- **Columns:**
    - card\_no (CHAR(16)): Unique identifier for credit cards.
    - holder\_name (VARCHAR(64)): Name of the cardholder.
    - cvv (CHAR(3)): Card Verification Value.
    - exp\_date (DATE): Expiry date of the credit card.
  - **Constraints:**
    - credit\_card\_pk (PRIMARY KEY): Primary key constraint on the card\_no column.
  - **Purpose:**
    - Contains information about credit cards used for payment transactions by customers.
4. **Customer Credit Table:**
- **Columns:**
    - ssn (CHAR(6)): Foreign key referencing the Customer table.
    - card\_no (CHAR(16)): Foreign key referencing the Credit Card table.
  - **Constraints:**
    - customer\_credit\_pk (PRIMARY KEY): Primary key constraint on ssn and card\_no.
    - customer\_credit\_customer\_fk (FOREIGN KEY): Foreign key constraint referencing Customer table.
    - customer\_credit\_card\_fk (FOREIGN KEY): Foreign key constraint referencing Credit Card table with CASCADE on DELETE.
  - **Purpose:**
    - Establishes the relationship between customers and their associated credit cards.
5. **Office Table:**
- **Columns:**
    - office\_id (INT): Auto-incremented identifier for offices.
    - name (VARCHAR(32)): Name of the office.
    - email (VARCHAR(64)): Unique email identifier for office login.

- phone\_no (CHAR(11)): Unique phone number identifier for office contact.
- password (VARCHAR(256)): Hashed password for office login.
- country (VARCHAR(64)): Country where the office is located.
- city (VARCHAR(64)): City where the office is located.
- building\_no (VARCHAR(16)): Building number of the office.
- **Constraints:**
  - office\_pk (PRIMARY KEY): Primary key constraint on the office\_id column.
- **Purpose:**
  - Stores information about car rental offices, including contact details and login credentials.

## 6. Car Table:

- **Columns:**
  - plate\_id (VARCHAR(8)): Unique identifier for the car.
  - model (VARCHAR(32)): Model of the car.
  - make (VARCHAR(32)): Make of the car.
  - year (YEAR): Manufacturing year of the car.
  - price (REAL): Rental price of the car.
  - registration\_date (DATE): Registration date of the car.
  - office\_id (INT): Foreign key referencing the Office table.
- **Constraints:**
  - car\_pk (PRIMARY KEY): Primary key constraint on the plate\_id column.
  - car\_office\_fk (FOREIGN KEY): Foreign key constraint referencing Office table with CASCADE on DELETE.
- **Purpose:**
  - Contains details about individual cars available for rental, including pricing and location.

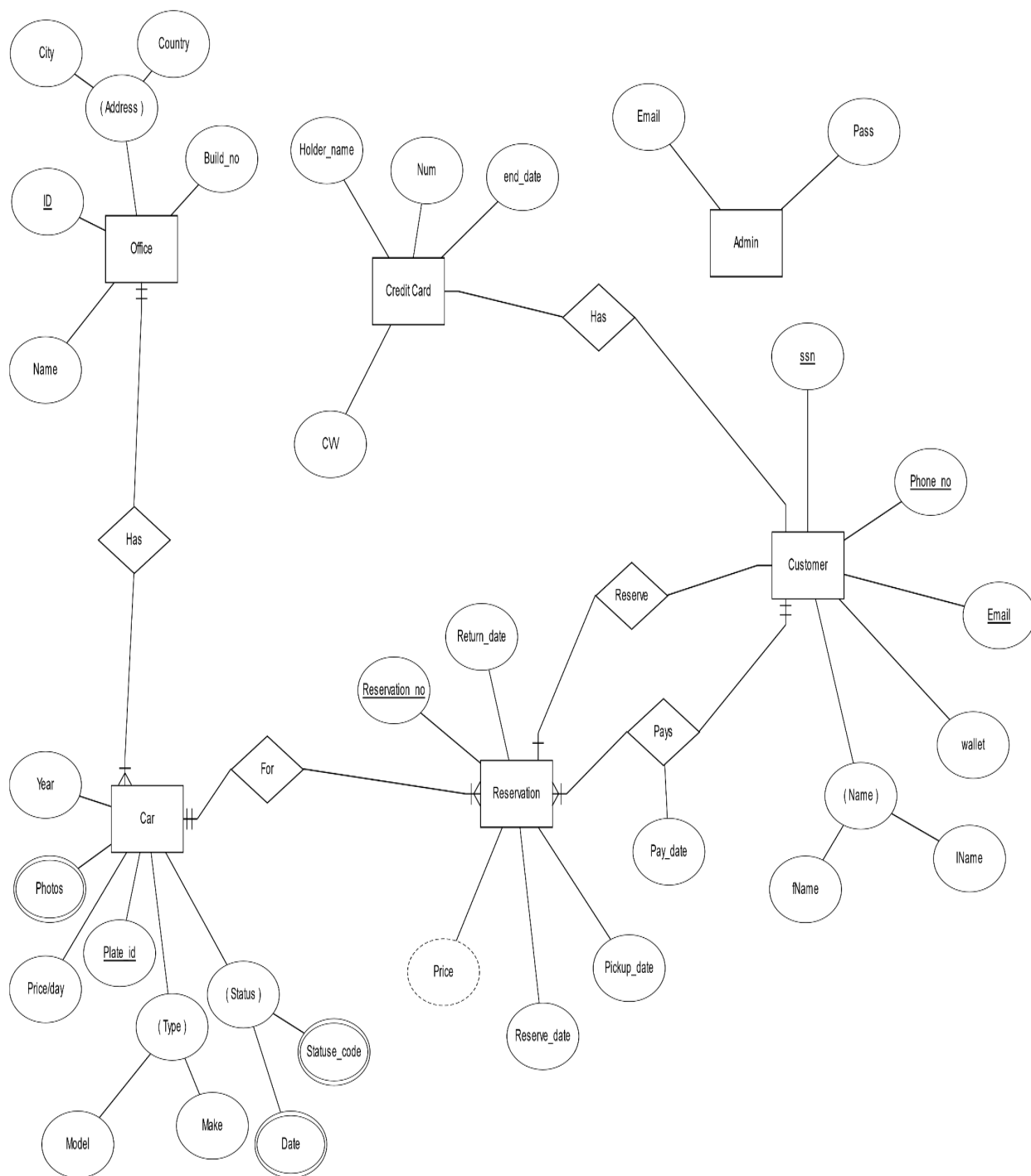
## 7. Reservation Table:

- **Columns:**
  - reservation\_no (INT): Auto-incremented identifier for reservations.
  - ssn (CHAR(6)): Foreign key referencing the Customer table.
  - plate\_id (VARCHAR(8)): Foreign key referencing the Car table.
  - reserve\_date (DATE): Date of reservation.
  - pickup\_date (DATE): Date of car pickup.
  - return\_date (DATE): Date of car return.
  - payment\_date (DATE): Date of payment.
- **Constraints:**
  - reservation\_pk (PRIMARY KEY): Primary key constraint on reservation\_no column.

- reservation\_customer\_fk (FOREIGN KEY): Foreign key constraint referencing Customer table.
  - reservation\_car\_fk (FOREIGN KEY): Foreign key constraint referencing Car table with CASCADE on DELETE.
  - **Purpose:**
    - Stores information about customer reservations, including pickup and return dates, payment status, and associated cars.
8. **Car Photos Table:**
- **Columns:**
    - plate\_id (VARCHAR(8)): Foreign key referencing the Car table.
    - photo (VARCHAR(512)): Photo URL of the car.
  - **Constraints:**
    - car\_photos\_pk (PRIMARY KEY): Primary key constraint on plate\_id and photo columns.
    - car\_photos\_fk (FOREIGN KEY): Foreign key constraint referencing Car table with CASCADE on DELETE.
  - **Purpose:**
    - Stores URLs of photos associated with each car.
9. **Car Status Table:**
- **Columns:**
    - plate\_id (VARCHAR(8)): Foreign key referencing the Car table.
    - status\_code (SMALLINT): Status code indicating car availability.
    - status\_date (DATETIME): Date and time of the status update.
  - **Constraints:**
    - car\_status\_pk (PRIMARY KEY): Primary key constraint on plate\_id, status\_code, and status\_date columns.
    - car\_status\_fk (FOREIGN KEY): Foreign key constraint referencing Car table with CASCADE on DELETE.
  - **Purpose:**
    - Records the status of each car, whether available, under maintenance, being cleaned, or rented.

Each table serves a distinct purpose within the car rental system, and their relationships are carefully defined through foreign key constraints, ensuring data integrity and coherence throughout the database. This relational schema provides a solid foundation for efficiently managing customer data, office details, car information, reservations, and associated status updates.

The ER diagram for the database is given below:



## Features:

### Sign-In/Sign-Up:

In our envisioned web-based car rental system, known as 'CarRentalServices,' we have identified three pivotal stakeholders: administrators, offices, and users. Each of these entities requires a secure and seamless process for both sign-in and sign-up functionalities. To achieve this, our system relies on carefully crafted mechanisms for credential verification and registration.

#### Credential Verification:

For sign-in authentication, the system leverages specific procedures to search and match credentials for administrators, offices, and users:

1. **Admin Sign-In:**
  - The system queries the database to validate an administrator's credentials based on their email.
2. **User Sign-In:**
  - User authentication involves querying the customer database, ensuring a secure match with the provided email.
3. **Office Sign-In:**
  - Offices are authenticated by querying the office database and verifying the provided email.

#### User and Office Sign-Up:

For the sign-up processes of both users and offices, the system utilizes distinct procedures to securely insert relevant information into the database:

1. **User Sign-Up:**
  - The system captures user information by inserting credit card details, customer details, and establishing a link between the customer and their credit card.
2. **Office Sign-Up:**
  - Office registration involves the secure insertion of office details, ensuring the confidentiality of sensitive information.

#### Security Measures:

The sign-up process prioritizes the secure storage of sensitive information, such as user and office details, and the association of users with their respective credit card information. Notably, our system adheres to industry-standard security practices:

- **JWT (JSON Web Token):**
  - JWTs are employed for generating secure authentication tokens. These tokens validate the identity of users during the sign-in process, enhancing overall system security.
- **Bcrypt for Password Hashing:**
  - User password security is ensured through Bcrypt, a robust hashing algorithm. Bcrypt transforms passwords into irreversible hashes, safeguarding user credentials from potential breaches.

By incorporating these security measures, 'CarRentalServices' ensures the confidentiality and integrity of user credentials, providing a trustworthy and resilient platform for administrators, offices, and users alike.

## Admin Menu:

### 1. Get Most Profitable Office:

- **Description:** This feature enables the admin to identify the office that has generated the highest revenue. It calculates the total earnings for each office by summing the product of the rental duration and car price for all completed reservations, then selects the office with the highest total revenue.
- **Usage:** The SQL query involves joining the **reservation**, **car**, and **office** tables, utilizing the duration of rentals and car prices to calculate the total earnings for each office. The result is then sorted, and the office with the highest total revenue is retrieved.

### 2. Get Most Rented Model:

- **Description:** This feature provides insights into the most frequently rented car model. It counts the occurrences of each car model in completed reservations and identifies the model with the highest count.
- **Usage:** The SQL query involves joining the **reservation** and **car** tables, grouping the results by car model, and ordering them by count in descending order. The model with the highest count is then retrieved.

### 3. Get Most Rented Make:

- **Description:** Similar to the previous feature, this functionality identifies the most popular car make based on the frequency of rentals. It counts the occurrences of each car make in completed reservations and selects the make with the highest count.
- **Usage:** The SQL query involves joining the **reservation** and **car** tables, grouping the results by car make, and ordering them by count in descending order. The make with the highest count is then retrieved.

#### 4. Get Car Status on a Day:

- **Description:** This feature allows the admin to retrieve the current status of all cars on a specified day. It provides vital information on whether a car is available, undergoing maintenance, being cleaned, or rented.
- **Usage:** The system employs a complex SQL query involving the **car\_status** and **car** tables, utilizing the latest status entries for each car to ensure accurate and up-to-date information.

#### 5. Payment in Period:

- **Description:** This feature allows the admin to retrieve information on all completed reservations within a specified payment period. It includes details on the reservation, such as the reservation number, customer information, car details, and the calculated revenue for each reservation based on the rental duration and car price.
- **Usage:** The SQL query involves joining the **reservation** and **car** tables, filtering results based on the payment date falling within the specified period. The revenue for each reservation is calculated using the duration of the rental and the car price.

#### 6. Reservation at a Certain Period:

- **Description:** This functionality enables the admin to view all reservations made within a specified period. It provides comprehensive details on each reservation, including customer information and car details.
- **Usage:** The SQL query involves joining the **reservation**, **customer**, and **car** tables, filtering results based on the reservation date falling within the specified period.

#### 7. Customer Reservation Search:

- **Description:** This feature allows the admin to search for reservations made by a specific customer. It provides detailed information on each reservation, including

customer details, car information, and the calculated revenue for each reservation based on the rental duration and car price.

- **Usage:** The SQL query involves joining the `reservation`, `customer`, and `car` tables, filtering results based on the customer's Social Security Number (SSN). The revenue for each reservation is calculated using the duration of the rental and the car price.

## 8. Reservation at a Certain Period for a Car:

- **Description:** This feature enables the admin to retrieve all reservations for a specific car within a specified period. It includes details on each reservation, encompassing customer information and reservation dates.
- **Usage:** The SQL query involves joining the `reservation`, `customer`, and `car` tables, filtering results based on both the car's plate ID and the reservation date falling within the specified period.

## 9. Advanced Search:

- **Description:** This feature in the 'CarRentalServices' system provides the admin with a robust tool for conducting an advanced search on reservations. It allows the admin to filter and retrieve reservation details based on a variety of criteria, including car model, make, year, plate ID, customer details (SSN, first name, last name, email, phone number), and reservation date.
- **Implementation:** The incoming request includes parameters for the search criteria, such as model, make, year, etc. The server-side logic dynamically constructs SQL queries based on these parameters, joining the `reservation`, `car`, and `customer` tables to gather comprehensive information. The conditions for the search are added to the queries based on the provided parameters.
- **Query Construction:** Two SQL queries are constructed and combined using a `UNION ALL` operation. The first query performs a left join on the `reservation`, `car`, and `customer` tables, while the second query performs a right join. The conditions for the search are added to both queries based on the provided parameters, ensuring a thorough search regardless of whether the reservation, car, or customer information is the primary focus. The final combined query is executed on the database, and the result is sent back as a response.



## Office Menu:

### 1. Add a Car:

- **Description:** This feature allows the office to add a new car to its inventory. The system inserts relevant details such as the plate ID, model, make, year, price, and links the car to the specific office.
- **Implementation:** The system executes three SQL queries to add a new car: one to insert the car details into the `car` table, another to initialize the car's status in the `car_status` table, and the last one to add a default photo for the car in the `car_photos` table.

### 2. Delete a Car:

- **Description:** This feature enables the office to remove a car from its inventory. The system first checks if the specified car belongs to the logged-in office, and if so, it deletes the car from the `car` table.
- **Implementation:** The system executes a SELECT query to retrieve the office ID associated with the specified car's plate ID. If the office ID matches the logged-in office, a DELETE query is executed to remove the car from the `car` table.

### 3. Change status of a Car:

- **Description:** This feature allows the office to update the status of its cars. The system first checks if the specified car belongs to the logged-in office and, if so, inserts a new status entry into the `car_status` table.
- **Implementation:** The system executes a SELECT query to retrieve the office ID associated with the specified car's plate ID. If the office ID matches the logged-in office, an INSERT query is executed to update the car's status in the `car_status` table.

### 4. Office Reservations Search:

- **Description:** This feature allows the office to search for reservations associated with its office. The system retrieves comprehensive information on each reservation, including customer details and calculated revenue.
- **Implementation:** The system executes a SELECT query, joining the `reservation`, `car`, `office`, and `customer` tables. The results are filtered based on the office ID, ensuring that only reservations related to the logged-in office are retrieved.

## 5. Get all the Cars operating under the office:

- **Description:** This feature retrieves information on cars associated with the logged-in office. It includes details on the current status of each car and is filtered based on the office ID.
- **Implementation:** The system executes a SELECT query, joining the `car_status` and `car` tables. The results are filtered based on the office ID, ensuring that only cars belonging to the logged-in office and with current status information are retrieved.

These features empower the office with functionalities to manage its car inventory, monitor reservations, and update the status of its cars within the 'CarRentalServices' system.

## User Menu:

### 1. Add a Reservation:

- **Description:** This feature allows users to make a reservation for a car. The system inserts relevant details such as the user's SSN, the car's plate ID, pickup and return dates, and, if applicable, the payment date.
- **Implementation:** Two SQL queries handle this functionality. The first query is for paid reservations, which includes the payment date in the `reservation` table. The second query is for unpaid reservations, where the payment date remains null until the user makes the payment. Additionally, two queries update the car's status to mark it as rented during the reservation period.

### 2. Reservation Payment (If Not Done During Reservation):

- **Description:** This feature allows users to make a payment for a reservation if it wasn't completed during the reservation process. It updates the payment date in the `reservation` table.
- **Implementation:** The system executes an UPDATE query to set the payment date to the current date for the specified reservation number. This ensures that the system records the payment and finalizes the reservation.

### 3. Show Available Cars Using User-Provided Filters:

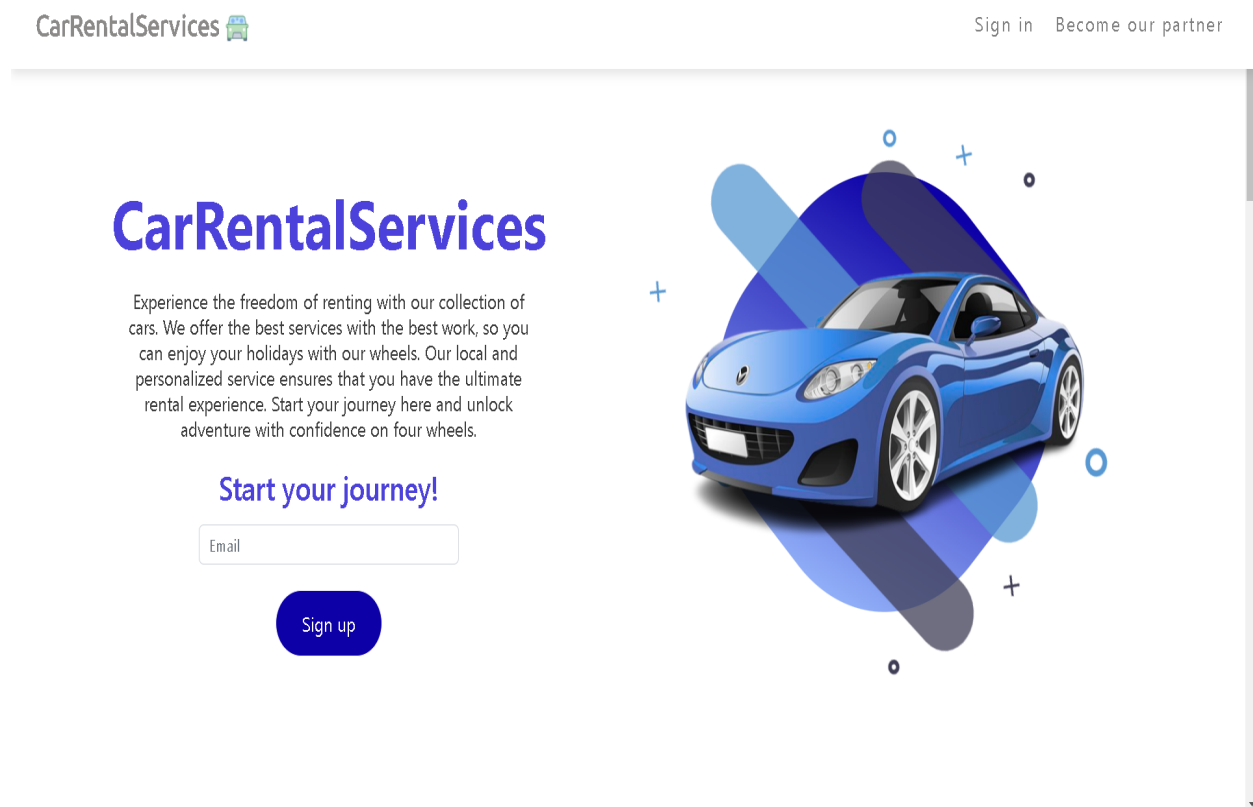
- **Description:** This feature allows authorized customers to search for available cars based on their preferences, including pickup and return dates, car model, make, location, and rental office details.


- **Implementation:** Utilizing an HTTP POST endpoint restricted to authorized customers, the system constructs a dynamic SQL query, joining relevant tables for comprehensive car information.
- **Conditions and Query Building:** The system constructs a dynamic SQL query based on the provided parameters and conditions. The main query involves joining the `car_status`, `car`, `office`, and `car_photos` tables to gather comprehensive information about the cars. A subquery is used to filter out cars that are already reserved during the specified pickup and return dates. Additional conditions are added to the query based on the provided parameters. For example, if the user specifies a particular car model, make, city, country, office name, or office building number, these conditions are appended to the query.
- **Filtering Available Cars:** The query executes on the database, filtering out cars that are not available during the specified pickup and return dates. It also filters cars based on the additional conditions provided by the user. The results are further filtered to include only cars with a status code of 0, indicating they are currently available. The endpoint responds with a list of available cars meeting the user's criteria. The details include car status, model, make, office details, and photos.

## Log out:


Last but not the least, there is a logout button for all stakeholders through which they can sign out of the system. The login-logout feature (i.e. remembering the logged-in user) in our web-based car rental system is seamlessly implemented using `cookie-parser`. When users decide to end their session, the system employs `cookie-parser` to efficiently clear the authentication cookie associated with their session. This mechanism ensures that the user's access privileges are promptly revoked, enhancing the overall security of the platform. The use of `cookie-parser` simplifies the logout process, providing users with a straightforward and reliable means to log out. This approach contributes to a positive user experience by ensuring a swift and secure logout, preventing any potential risks associated with lingering sessions.

## Some screenshots of the user interface:



CarRentalServices 

## Register Your Office

CarRentalServices 

Admin

Reports


Payments



Reservations

Cars Status

Customer Reservations

Car Reservations



Statistics  

Most rented model

Audi

This model was rented by 2 customers

Most rented make

A4

This make was rented by 1 customers


Most Profitable Office

Lambo office

This office has successfully gained 300 \$

Search

Advanced Search

CarRentalServices 

Back

Advanced Search

Make

Model

Year


Plate ID

Customer SSN

Customer First Name

Customer Last Name

Customer E-Mail

CarRentalServices 

Uber office

Log Out


Your current reservations

Res NO.	Plate ID	Customer Name	Res Date	Pickup Date	Return Date	Total Revenue
9	31742611	Ziad hassan	2023-11-8	2022-1-24	2022-1-27	200
10	97785008	Leonel Messi	2023-11-8	2022-1-25	2022-1-28	58

Your current cars

Add Car

Plate ID	Status	Registration Date	Make	Model	Year	Price/Day	
31742611	Available	2023-11-07	Model S	Tesla	2021	50	-
97785008	Available	2023-11-07	Tucson	Hyundai	2018	14.5	-

CarRentalServices 

Cristiano Ronaldo

Log Out

## Reserve a Car

Any

Any

Audi

Alfa Romeo

BMW

Tesla

Jaguar

Mercedes

Nissan

Mini

Infiniti

Land Rover

Kia

Honda

Chevrolet

Any

Any

Up Date

dd / mm / yyyy


Return Date:

dd / mm / yyyy

Show Available Cars

☒ Pay Now

Reserve

CarRentalServices 

## Add new Car

Plate ID

model

make

year

Price/Day


Photo Link

Add

Add Photo

## Conclusion:

In the dynamic realm of urban mobility solutions, the development and implementation of a web-based car rental system stand as a testament to the convergence of cutting-edge



technologies, user-centric design, and robust backend infrastructure. The comprehensive integration of HTML5, CSS3, JavaScript, Ajax, Node.js, Express, JWT, Bcrypt, and MySQL collectively forms a sophisticated ecosystem, addressing the multifaceted challenges inherent in contemporary transportation frameworks.

The frontend design, crafted with HTML5, CSS3, and JavaScript, prioritizes user experience, offering an intuitive interface that seamlessly connects users with the intricacies of the car rental system. Through responsive design principles and dynamic functionalities facilitated by Ajax, users navigate effortlessly, benefiting from real-time updates and a visually appealing layout that enhances their interaction with the platform.

On the backend, Node.js and Express provide a scalable and efficient runtime environment, facilitating the handling of user requests with agility. The inclusion of JWT and Bcrypt ensures a secure and reliable authentication mechanism, safeguarding user data and fortifying the system against potential security threats. Meanwhile, the MySQL database, accessed through carefully constructed DML queries, serves as the foundation for structured data management, ensuring integrity and reliability in the storage and retrieval of information.

Together, these frontend and backend components synergize to create a holistic solution for urban mobility challenges. Users, whether administrators, offices, or end-users, experience a seamless, secure, and intuitive platform for managing, reserving, and renting cars. The real-time nature of the system, powered by Ajax, enhances the overall efficiency and responsiveness, contributing to a positive user experience.

As urban landscapes continue to evolve, this web-based car rental system, with its forward-looking architecture, establishes a precedent for scalable, secure, and user-friendly solutions. The integration of emerging technologies, commitment to sustainability, and adaptability to future advancements position the system at the forefront of innovation in the field of urban mobility. This project not only addresses the current challenges in transportation but also lays the groundwork for continued innovation and enhancement, underscoring the importance of technology in shaping the future of urban mobility solutions.