CS-GY 6083-B, Principles of Database Systems

Project Report (Part 2)

WOW (World On Wheels)

Submission Date: May 12, 2022

Submitted to:
Professor Amit Patel

Submitted by:

Aakash Gunda	ga2184
Navya Sravani Jammalamadaka	nsj9072
Sai Vikas Mandadapu	sm9510

Table of Contents

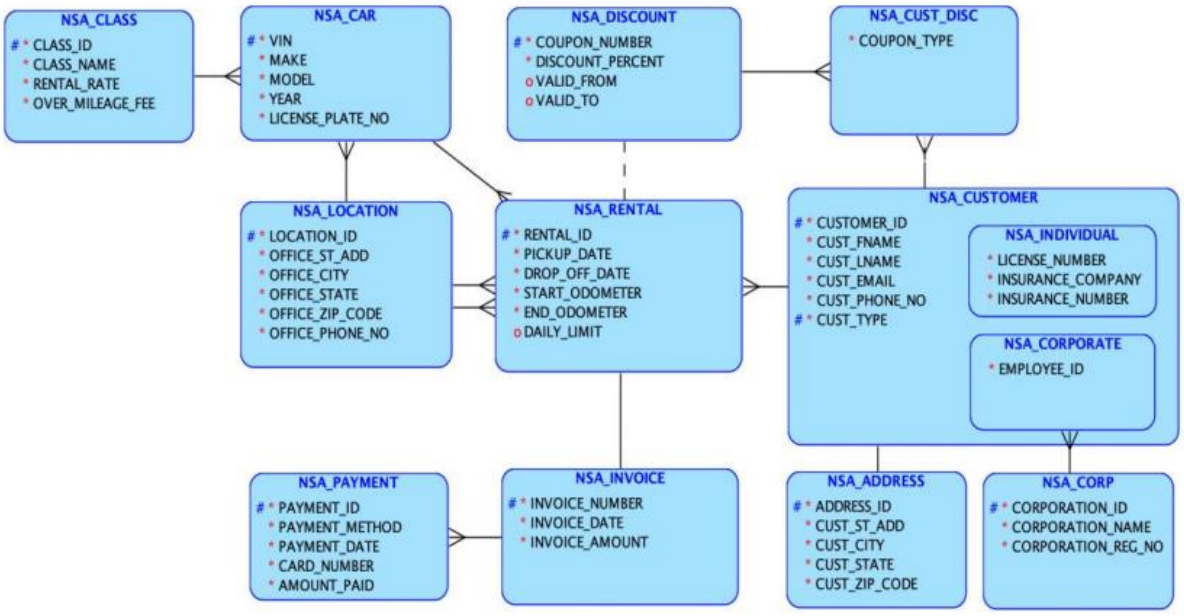
Execute Summary

In this project, we created a relational database management system for WOW (World on wheels). It is a car rental company operating in the United States . WOW maintains both a collection of corporate customers and individual customers that they service. Corporate customers are those who work for corporations which have struck an agreement with WOW Rentals to use their services in exchange for a discounted rental rate. The customer's table lists all of the people who will be renting cars from what time to time and at what locations by connecting to the intersect entities presented in the model. The customer is the passenger who books a car for rent for himself and one or additional passengers. Once a car is booked by the customer, an invoice is generated for him/her, and payments are made against those bills.

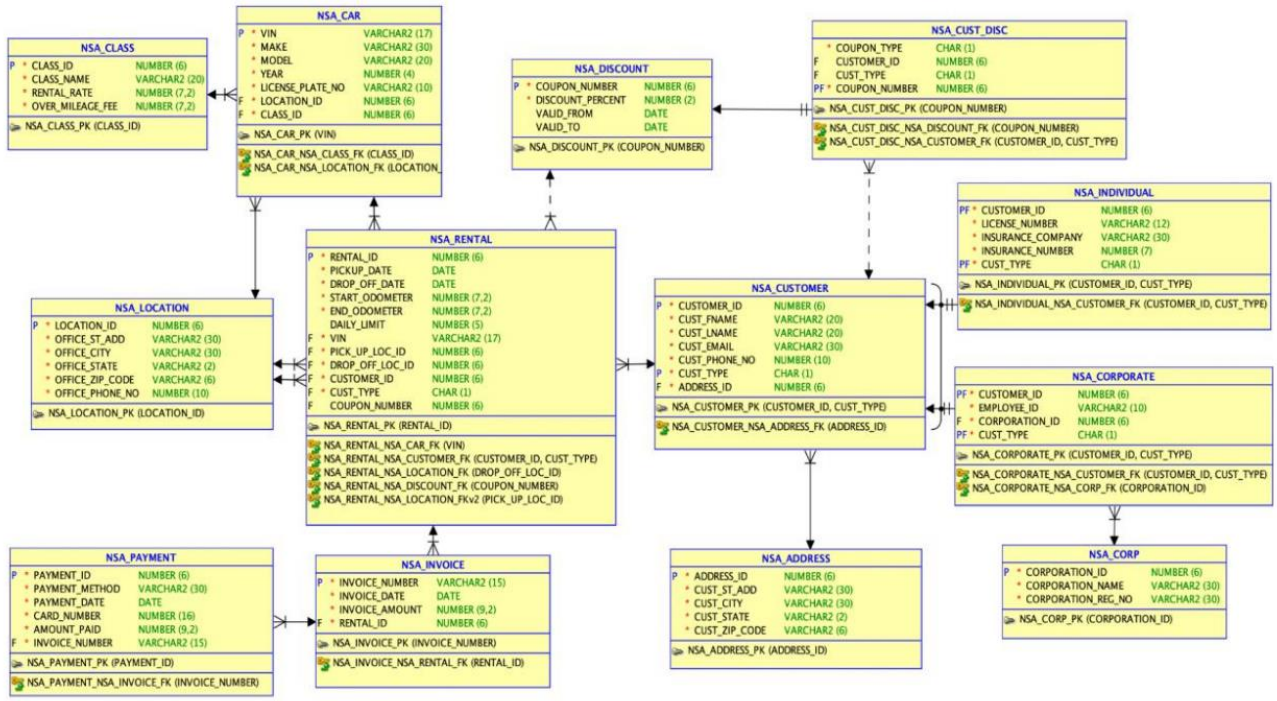
The customers who will be booking the cars are basically grouped into two categories according to our design. They are "Individual customers" and "Corporate customers". Individual customers are those who would need an insurance number and a driving license to get approved for a car rental. Corporate customers are those who would need a corporate identity number and a corporation registration number. All basic details such as Name, Email, Phone number, Address of the customer, and type of customer are collected to ensure a smooth booking process.

During the process of booking a car, the booking_id is taken as a unique attribute that is basically used to identify the customer and the car he booked. The company of the car, model of the car, year of manufacturing, and license plate number give a basic overview of the car to the customer who chooses to rent the car. The customer has to provide with pick-up date and drop-off date. Then, odometer ranges are collected from the car after the customer delivers the car back to the rental company. We have an attribute called daily limit to record if the odometer reading has crossed the limit for that day. Each car is grouped in a class that has a class id and class name. The rental rate is specified in the class entity that explains the rent of each car. Over mileage fee is specified in a similar way accordingly. The invoice is then directed to the customer after calculating the distance he traveled and adding the over mileage fee. Customer can check the available payments through the website and can still a bill multiple smaller amounts and use different credit, debit cards to pay the bill.

Logical Model



Relational Model



Assumptions

There are few assumptions that are made apart from the given business constraints, which are mentioned below:

- We used our initials, NSA_ as prefixes to all the tables.
- Name and address are composite attributes, so we divided them into multiple simple attributes.
- At a single location, there can be multiple classes of cars and a single class car can be at different locations, so many-to-many relations between NSA_LOCATION and NSA_CLASS. It is resolved using an associate entity.
- A car can be rented many times, so there is a one-to-many relationship between NSA_CAR and NSA_RENTAL.
- In NSA_RENTAL, the pick_up_loc_id and drop_off_loc_id are obtained from location_id, which is the primary key of NSA_LOCATION. Hence, there are two connections between NSA_RENTAL and NSA_RENTAL.
- Given, that some rental services are with unlimited mileage, daily_limit is kept optional in NSA_RENTAL.
- Customers can be individual or corporate or both, so it has two subtypes which are distinguished by cust_type. Also, cust_type is made as to the primary key along with customer_id because the same customer can be both individual and corporate.
- Many corporate employees can be from the same corporation, so a separate NSA_CORP entity is created with corporate_id as the primary key.
- There can be some customers who can book more than one car, so there is a one-to-many relationships between NSA_CUSTOMER and NSA_RENTAL.
- Class_id, customer_id, location_id, rental_id, payment_id, address_id, invoice_number, coupon_number are added as primary keys in their respective tables.
- Given, WOW keeps only one address of each customer, so a separate entity NSA_ADDRESS entity is created. There is a one-to-one relationship between NSA_CUSTOMER and NSA_ADDRESS, We made NSA_ADDRESS as a parent.
- Same coupon can be distributed to multiple customers and a customer can have multiple coupons, so a many-to-many relationship between NSA_CUSTOMER and NSA_DISCOUNT. It is resolved by adding an NSA_CUST_DISC intersect entity which has coupon_type attribute.
- Some customers may or may not have a coupon, so it is made optional on one side.

- Since some corporate customers can have coupons with no expiration date, valid_from and valid_to attributes in NSA_DISCOUNT are made optional.
- The relationship between NSA_RENTAL and NSA_DISCOUNT is optional, as the discount may not be applicable to all the rentals.
- A rental service can generate only one invoice, so a one-to-one relationship between NSA_RENTAL and NSA_INVOICE. In the relational model, NSA_RENTAL is made as a parent.
- An invoice amount can be paid using multiple payment methods for a single transaction, such as a gift card applied or using a credit card for the remaining balance, so a one-to-many relationship between NSA_INVOICE and NSA_PAYMENT.
- Further, an optional entity "INSURANCE" can be added if WOW decides to offer insurance to the customer. This can be connected directly to the NSA_INDIVIDUAL entity.
- Added a new NSA_DETAILS entity to store the booking details of the customer when booking a car. These details will be added to NSA_RENTAL after customer returns the car.
- Added a new unique attribute location_name in NSA_LOCATION entity.
- Whenever a customer pays a bill, the invoice_amount is updated in the NSA_INVOICE entity.

Details of Software, Programming Language, and Database

HTML:

HTML is the standard markup language for documents designed to be displayed in a web browser. HTML is used to define the layout, structure, and components of the frontend of a website. Images and other objects, such as interactive forms, can be embedded in web pages using HTML. Tags are used in the code to specify the semantics of page objects. For complete front-end development, it is frequently integrated with other languages such as CSS and JavaScript.

CSS:

Cascading Style Sheets (CSS) is a language for creating style sheets. The World Wide Web Consortium first released it in 1996, and it was last updated in 2016. Since its beginnings, this language has undergone very little change. CSS is in charge of the appearance and styling of any web page. It specifies the layout, colors, and fonts of webpages whose contents are described by HTML codes. CSS is used to govern accessibility, content presentation flexibility, and the ability for numerous web pages to share formatting.

JavaScript:

Along with HTML and CSS, another cornerstone technology underpinning the World Wide Web is JavaScript, which is sometimes abbreviated as JS. The majority of today's web pages use JavaScript to give functionality. It also includes a number of libraries that provide various functions to the web pages that are being created. JavaScript is in charge of all event-driven functions. When creating websites, it's frequently utilized for graphical illustrations and functions.

PHP:

PHP stands for Personal Home Page, but it is now known as Hypertext Preprocessor, and it was created in 1994. It's a server-side scripting language that can be integrated into HTML to make web pages more dynamic and interactive. On the other hand, PHP can connect to practically any database with ease. PHP can be run straight from the command line, and there are a number of PHP interpreters available.

SQL:

The acronym SQL stands for Structured Query Language, and it is frequently mispronounced as "sequel." It's a standardized programming language for dealing with data in relational database management systems. It was first introduced in 1974 and has since been used to manage structured data in databases. SQL is a simple language to learn and apply. Many records can be accessed with a single SQL query. Apart from that, unlike previous languages, SQL does not require the specification of indexes for each record to be accessed. DDL (Data Definition Language), DCL (Data Control Language), and DML (Data Management Language) are some of the subtypes (Data Manipulation Language). SQL is available in a variety of versions. We used Oracle SQL for this project.

DDL Code

```
-- SQLINES DEMO *** le SQL Developer Data Modeler
21.4.1.349.1605
-- SQLINES DEMO *** -04-08 09:24:17 EDT
-- SQLINES DEMO *** le Database 21c
-- SQLnsa_addressINES DEMO *** le Database 21c



-- SQLINES DEMO *** no DDL - MDSYS.SDO_GEOMETRY

-- SQLINES DEMO *** no DDL - XMLTYPE

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_address (
    address_id    INT(5) NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT 'This is the unique address id',
    cust_st_add   VARCHAR(30) NOT NULL COMMENT 'This is the
street address of the customer',
    cust_city     VARCHAR(30) NOT NULL COMMENT 'This is the city
in which customer stays',
    cust_state    VARCHAR(2) NOT NULL COMMENT 'This is the state
in whic customer stays',
    cust_zip_code VARCHAR(6) NOT NULL COMMENT 'This is the zip
code of the customer'
);

ALTER TABLE nsa_address AUTO_INCREMENT=10000;

CREATE TABLE nsa_details (
    booking_id        INT(5) NOT NULL AUTO_INCREMENT PRIMARY
KEY,
    pickup_location   VARCHAR(30),
    drop_location     VARCHAR(30),
    pickup_date       DATETIME,
    drop_off_date     DATETIME,
    customer_id       INT(5),
    car_id            INT(5)
);

ALTER TABLE nsa_details AUTO_INCREMENT=12345;

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_car (
```

```
    car_id           INT(5) NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT 'This is the unique car id',
    vin              VARCHAR(17) NOT NULL COMMENT 'This is a
unique vehicle identification number',
    make             VARCHAR(30) NOT NULL COMMENT 'This is the
company that made that car',
    model            VARCHAR(20) NOT NULL COMMENT 'This is the
model of the car',
    year             SMALLINT NOT NULL COMMENT 'This is the year
in which car is manufactured',
    license_plate_no VARCHAR(10) NOT NULL COMMENT 'This the
license plate number of the car',
    location_id      INT(5) NOT NULL,
    class_id         INT(5) NOT NULL
);

ALTER TABLE nsa_car AUTO_INCREMENT=20000;

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_class (
    class_id         INT(5) NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT 'This is the class Id of the car',
    class_name       VARCHAR(20) NOT NULL COMMENT 'This is the
name of the car class',
    rental_rate      DECIMAL(7, 2) NOT NULL COMMENT 'This is the
rental rate per day of the car class',
    over_mileage_fee DECIMAL(7, 2) NOT NULL COMMENT 'This is the
ocer mileage fee for that car class'
);

ALTER TABLE nsa_class AUTO_INCREMENT=30000;

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_corp (
    corporation_id     INT(5) NOT NULL AUTO_INCREMENT PRIMARY
KEY COMMENT 'This is the unique corporation ID',
    corporation_name   VARCHAR(30) NOT NULL COMMENT 'This is the
name of the corporation',
    corporation_reg_no VARCHAR(30) NOT NULL COMMENT 'This is the
registration number of the corporation'
);

ALTER TABLE nsa_corp  AUTO_INCREMENT=40000;

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_corporate (
```

```
    customer_id    INT(5) NOT NULL COMMENT 'This is the unique
customer ID',
    employee_id    VARCHAR(10) NOT NULL COMMENT 'This is the
unique employee ID',
    corporation_id INT(5) NOT NULL,
    cust_type      CHAR(1) NOT NULL COMMENT 'This is the
customer type discriminator'
);


ALTER TABLE nsa_corporate ADD CONSTRAINT nsa_corporate_pk
PRIMARY KEY ( customer_id,

cust_type );

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_cust_disc (
    coupon_type   CHAR(1) NOT NULL COMMENT 'This is the type of
the coupon customer has.',
    customer_id   INT(5),
    cust_type     CHAR(1),
    coupon_number INT(8) NOT NULL
);

ALTER TABLE nsa_cust_disc ADD CONSTRAINT nsa_cust_disc_pk
PRIMARY KEY ( coupon_number );


-- SQLINES LICENSE FOR EVALUATION USE ONLY

CREATE TABLE nsa_customer (
    customer_id   INT(5) NOT NULL AUTO_INCREMENT COMMENT 'This
is the unique customer ID',
    fname    VARCHAR(20) NOT NULL COMMENT 'This is the first
name of the customer',
    lname    VARCHAR(20) NOT NULL COMMENT 'This is the last name
of the customer',
    email    VARCHAR(30) NOT NULL COMMENT 'This is the email id
of the customer',
    password VARCHAR(100) NOT NULL,
    cust_phone_no BIGINT NOT NULL COMMENT 'This is the phone
number of the customer',
    cust_type     CHAR(1) NOT NULL COMMENT 'This is the customer
type discriminator',
    address_id    INT(5) NOT NULL,
    PRIMARY KEY (customer_id,cust_type )
);
```

```
ALTER TABLE nsa_customer
    ADD CONSTRAINT ch_inh_nsa_customer CHECK ( cust_type IN (
'C', 'I' ) );

ALTER TABLE nsa_customer AUTO_INCREMENT = 50000;


-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_discount (
    coupon_number    INT(8) NOT NULL PRIMARY KEY COMMENT 'This
is the unique coupon number',
    discount_percent TINYINT NOT NULL COMMENT 'This is
Percentage of the discount offered.',
    valid_from       DATETIME COMMENT 'This is the data the
coupon is valid from',
    valid_to         DATETIME COMMENT 'This is the data the
coupon is valid to.'
);


-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_individual (
    customer_id       INT(5) NOT NULL COMMENT 'This is the
unique customer ID',
    license_number    VARCHAR(12) NOT NULL COMMENT 'This is the
license number of the individual customer',
    insurance_company VARCHAR(30) NOT NULL COMMENT 'This is the
name of the insurance company',
    insurance_number  INT(5) NOT NULL COMMENT 'This is the
insurance number of the individual customer.',
    cust_type         CHAR(1) NOT NULL COMMENT 'This is the
customer type discriminator'
);

ALTER TABLE nsa_individual ADD CONSTRAINT nsa_individual_pk
PRIMARY KEY ( customer_id,

cust_type );
-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_invoice (
    invoice_number BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT 'This is the Invoice number',
    invoice_date   DATETIME NOT NULL COMMENT 'This is date on
which invoice is generated',
    invoice_amount DECIMAL(9, 2) NOT NULL COMMENT 'This is
amount the customer should pay.',
```

```
    rental_id      INT(5) NOT NULL,
    customer_id    INT(5) NOT NULL
);

ALTER TABLE nsa_invoice AUTO_INCREMENT=1234567890;

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_location (
    location_id     INT(5) NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT 'This is the unique location ID',
    office_st_add   VARCHAR(30) NOT NULL COMMENT 'This is the
street address of the office.',
    office_city     VARCHAR(30) NOT NULL COMMENT 'This is the
city in which office is loacted',
    office_state    VARCHAR(2) NOT NULL COMMENT 'This is the
state in which office is located',
    office_zip_code VARCHAR(6) NOT NULL COMMENT 'This is the Zip
code of the office location.',
    office_phone_no BIGINT NOT NULL COMMENT 'This is the phone
number of the office',
    location_name   VARCHAR(30) NOT NULL
);

ALTER TABLE nsa_location AUTO_INCREMENT=60000;

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_payment (
    payment_id     INT(5) NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT 'This is the unique payment ID',
    payment_method VARCHAR(30) NOT NULL COMMENT 'This is the
method of the payment',
    payment_date   DATETIME NOT NULL COMMENT 'This is the data
on which payment is made',
    card_number    BIGINT NOT NULL COMMENT 'This the card number
used for the payment',
    amount_paid    DECIMAL(9, 2) NOT NULL COMMENT 'This is the
amount for which payment is made',
    invoice_number BIGINT NOT NULL
);

ALTER TABLE nsa_payment AUTO_INCREMENT=70000;

-- SQLINES LICENSE FOR EVALUATION USE ONLY
CREATE TABLE nsa_rental (
    rental_id       INT(5) NOT NULL AUTO_INCREMENT COMMENT 'This
is the unique rental service id',
```

```
    pickup_date     DATETIME NOT NULL COMMENT 'This is the pick
up date of the vehicle',
    drop_off_date   DATETIME NOT NULL COMMENT 'This is the drop
off date of the vehicle',
    start_odometer  DECIMAL(7, 2) NOT NULL COMMENT 'This is the
odometer reading at the pick up time',
    end_odometer    DECIMAL(7, 2) NOT NULL COMMENT 'This is the
drop off time odometer reading',
    daily_limit     INT COMMENT 'This is the daily odometer
limit of the vehicle',
    car_id          INT(5) NOT NULL,
    pick_up_loc_id  INT(5) NOT NULL,
    drop_off_loc_id INT(5) NOT NULL,
    customer_id     INT(5) NOT NULL,
    cust_type       CHAR(1) NOT NULL,
    coupon_number   INT(8),
    PRIMARY KEY (rental_id,customer_id)
);

ALTER TABLE nsa_rental AUTO_INCREMENT=80000;

ALTER TABLE nsa_car
    ADD CONSTRAINT nsa_car_nsa_class_fk FOREIGN KEY ( class_id )
        REFERENCES nsa_class ( class_id );

ALTER TABLE nsa_car
    ADD CONSTRAINT nsa_car_nsa_location_fk FOREIGN KEY (
location_id )
        REFERENCES nsa_location ( location_id );

ALTER TABLE nsa_corporate
    ADD CONSTRAINT nsa_corporate_nsa_corp_fk FOREIGN KEY (
corporation_id )
        REFERENCES nsa_corp ( corporation_id );

ALTER TABLE nsa_corporate
    ADD CONSTRAINT nsa_corporate_nsa_customer_fk FOREIGN KEY (
customer_id,

cust_type )
        REFERENCES nsa_customer ( customer_id,
                                  cust_type );

ALTER TABLE nsa_cust_disc
    ADD CONSTRAINT nsa_cust_disc_nsa_customer_fk FOREIGN KEY (
customer_id,
```

```
cust_type )
        REFERENCES nsa_customer ( customer_id,
                                  cust_type );

ALTER TABLE nsa_cust_disc
    ADD CONSTRAINT nsa_cust_disc_nsa_discount_fk FOREIGN KEY (
coupon_number )
        REFERENCES nsa_discount ( coupon_number );

ALTER TABLE nsa_customer
    ADD CONSTRAINT nsa_customer_nsa_address_fk FOREIGN KEY (
address_id )
        REFERENCES nsa_address ( address_id );

ALTER TABLE nsa_individual
    ADD CONSTRAINT nsa_individual_nsa_customer_fk FOREIGN KEY (
customer_id,

cust_type )
        REFERENCES nsa_customer ( customer_id,
                                  cust_type );

ALTER TABLE nsa_invoice
    ADD CONSTRAINT nsa_invoice_nsa_rental_fk FOREIGN KEY (
rental_id, customer_id)
        REFERENCES nsa_rental ( rental_id, customer_id );

ALTER TABLE nsa_payment
    ADD CONSTRAINT nsa_payment_nsa_invoice_fk FOREIGN KEY (
invoice_number )
        REFERENCES nsa_invoice ( invoice_number );

ALTER TABLE nsa_rental
    ADD CONSTRAINT nsa_rental_nsa_car_fk FOREIGN KEY ( car_id )
        REFERENCES nsa_car ( car_id );

ALTER TABLE nsa_rental
    ADD CONSTRAINT nsa_rental_nsa_customer_fk FOREIGN KEY (
customer_id,

cust_type )
        REFERENCES nsa_customer ( customer_id,
                                  cust_type );

ALTER TABLE nsa_rental
```

```
    ADD CONSTRAINT nsa_rental_nsa_discount_fk FOREIGN KEY (
coupon_number )
        REFERENCES nsa_discount ( coupon_number );

ALTER TABLE nsa_rental
    ADD CONSTRAINT nsa_rental_nsa_location_fk FOREIGN KEY (
drop_off_loc_id )
        REFERENCES nsa_location ( location_id );

ALTER TABLE nsa_rental
    ADD CONSTRAINT nsa_rental_nsa_location_fkv2 FOREIGN KEY (
pick_up_loc_id )
        REFERENCES nsa_location ( location_id );

DELIMITER $$
CREATE TRIGGER arc_fkarc_3_nsa_corporate BEFORE
    INSERT OR UPDATE OF customer_id, cust_type ON nsa_corporate
    FOR EACH ROW
    DECLARE d CHAR(1)//
BEGIN
    -- SQLINES LICENSE FOR EVALUATION USE ONLY
    SELECT
        a.cust_type
    INTO d
    FROM
        nsa_customer a
    WHERE
        a.customer_id = :new.customer_id
        AND a.cust_type = :new.cust_type

    IF ( d IS NULL OR d <> 'C' ) THEN
        raise_application_error(
                                -20223,
                                'FK NSA_CORPORATE_NSA_CUSTOMER_FK
in Table NSA_CORPORATE violates Arc constraint on Table
NSA_CUSTOMER - discriminator column CUST_TYPE doesn''t have
value ''C'''
        )
    END IF

    DECLARE EXIT HANDLER FOR not found BEGIN
        NULL
    END
    DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
        RESIGNAL
    END
END
```

```
/
DELIMITER ;

DELIMITER $$
CREATE OR REPLACE TRIGGER arc_fkarc_3_nsa_individual BEFORE
    INSERT OR UPDATE OF customer_id, cust_type ON nsa_individual
    FOR EACH ROW
    DECLARE d CHAR(1)//
BEGIN
    -- SQLINES LICENSE FOR EVALUATION USE ONLY
    SELECT
        a.cust_type
    INTO d
    FROM
        nsa_customer a
    WHERE
        a.customer_id = :new.customer_id
        AND a.cust_type = :new.cust_type//

    IF ( d IS NULL OR d <> 'I' ) THEN
        raise_application_error(
                                -20223,
                                'FK
NSA_INDIVIDUAL_NSA_CUSTOMER_FK in Table NSA_INDIVIDUAL violates
Arc constraint on Table NSA_CUSTOMER - discriminator column
CUST_TYPE doesn''t have value ''I'''
        )//
    END IF//

    DECLARE EXIT HANDLER FOR not found BEGIN
        NULL//
    END//
    DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
        RESIGNAL//
    END//
END;
/
DELIMITER ;

-- SQLINES DEMO *** per Data Modeler Summary Report:
--
-- SQLINES DEMO ***                        13
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                        29
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
```

```
-- SQLINES DEMO *** DY                      0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         2
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***  TYPE                   0
-- SQLINES DEMO ***  TYPE                   0
-- SQLINES DEMO ***  TYPE BODY              0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO *** EGMENT                  0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO *** ED VIEW                 0
-- SQLINES DEMO *** ED VIEW LOG             0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
--
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
--
-- SQLINES DEMO ***                         0
--
-- SQLINES DEMO ***                         0
-- SQLINES DEMO *** A                       0
-- SQLINES DEMO *** T                       0
--
-- SQLINES DEMO ***                         0
-- SQLINES DEMO ***                         0
```

List of Tables

Your SQL query has been executed successfully.

```
1 show tables;
```

Tables_in_nsa_project
nsa_address
nsa_car
nsa_class
nsa_corp
nsa_corporate
nsa_cust_disc
nsa_customer
nsa_details
nsa_discount
nsa_individual
nsa_invoice
nsa_location
nsa_payment
nsa_rental

Record Counts of Each Table

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_address;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
10

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_car;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
29

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_class;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
10

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_corp;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
5

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_corporate;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
5

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_customer;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
10

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_cust_disc;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
7

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_details;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
10

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_discount;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
10

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_individual;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
5

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_invoice;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
10

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_location;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
11

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_payment;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
11

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM nsa_rental;
```

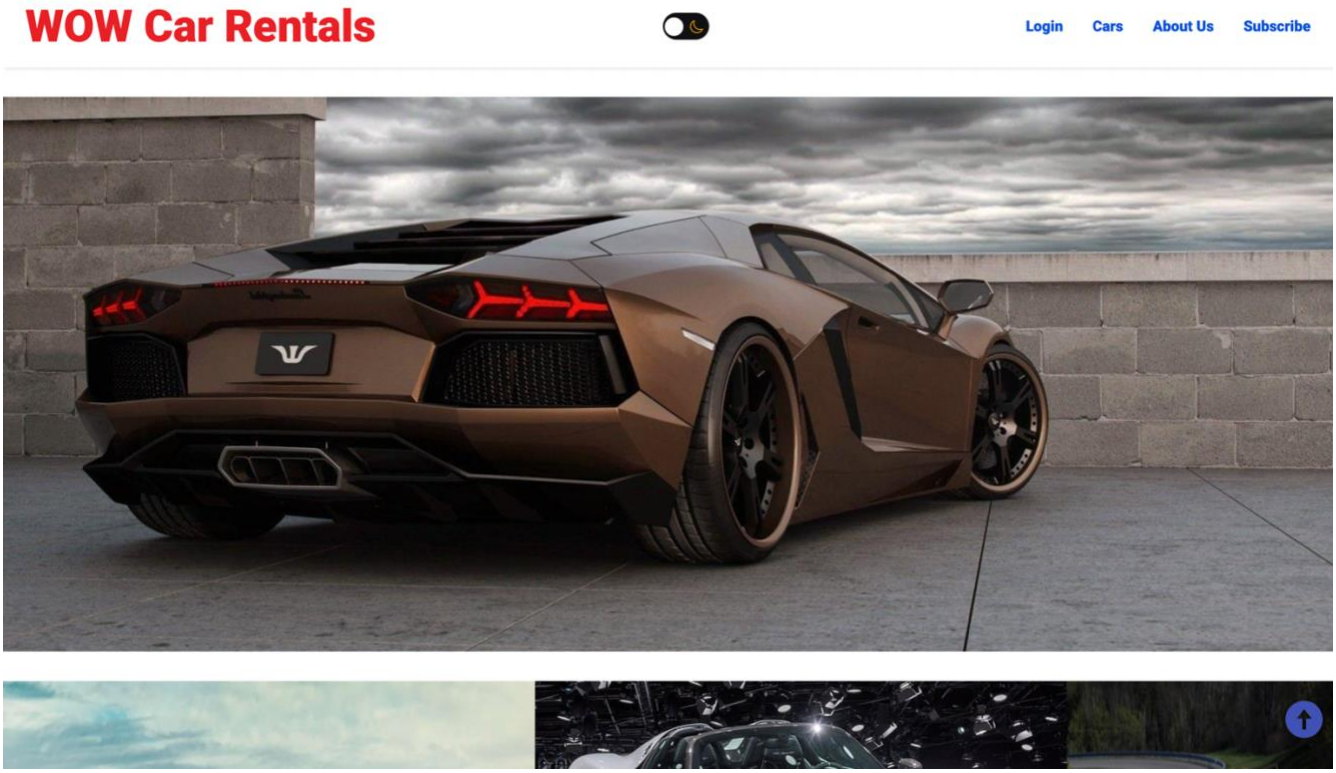
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

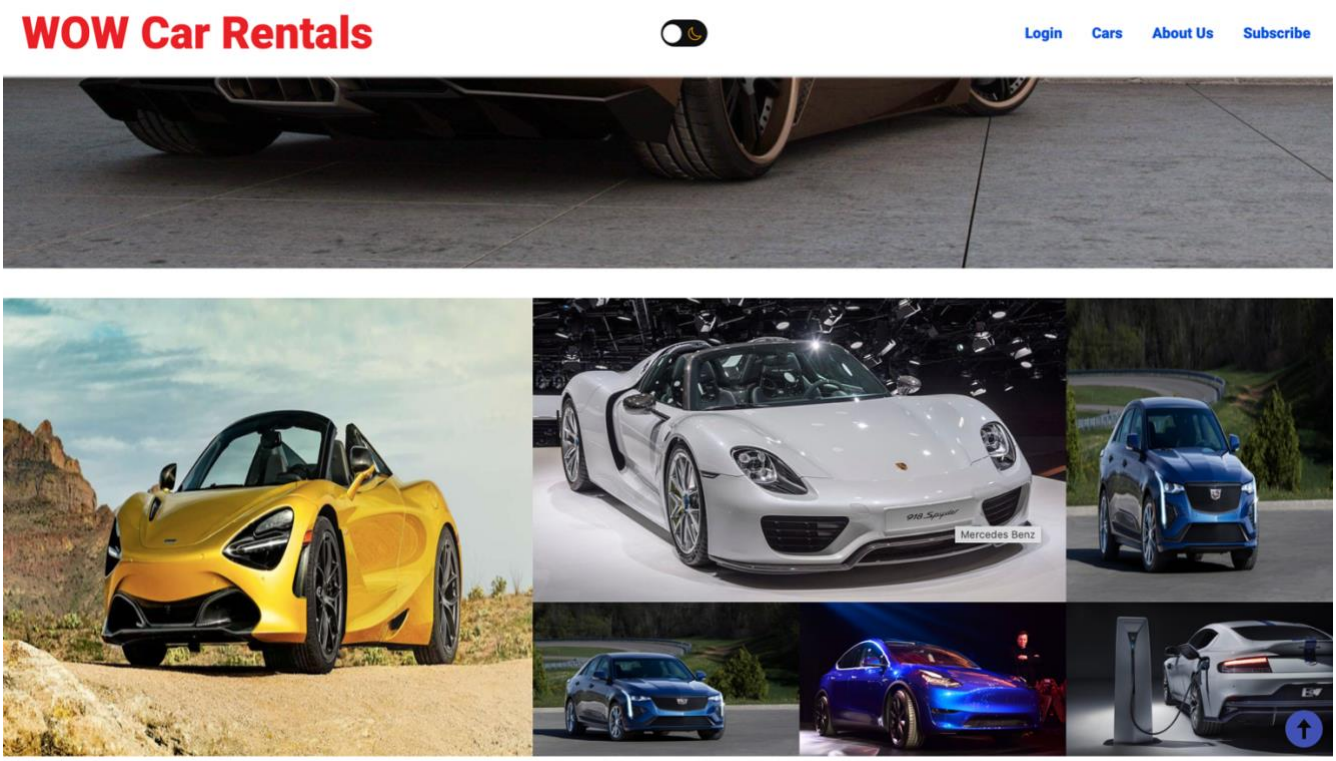
+ Options

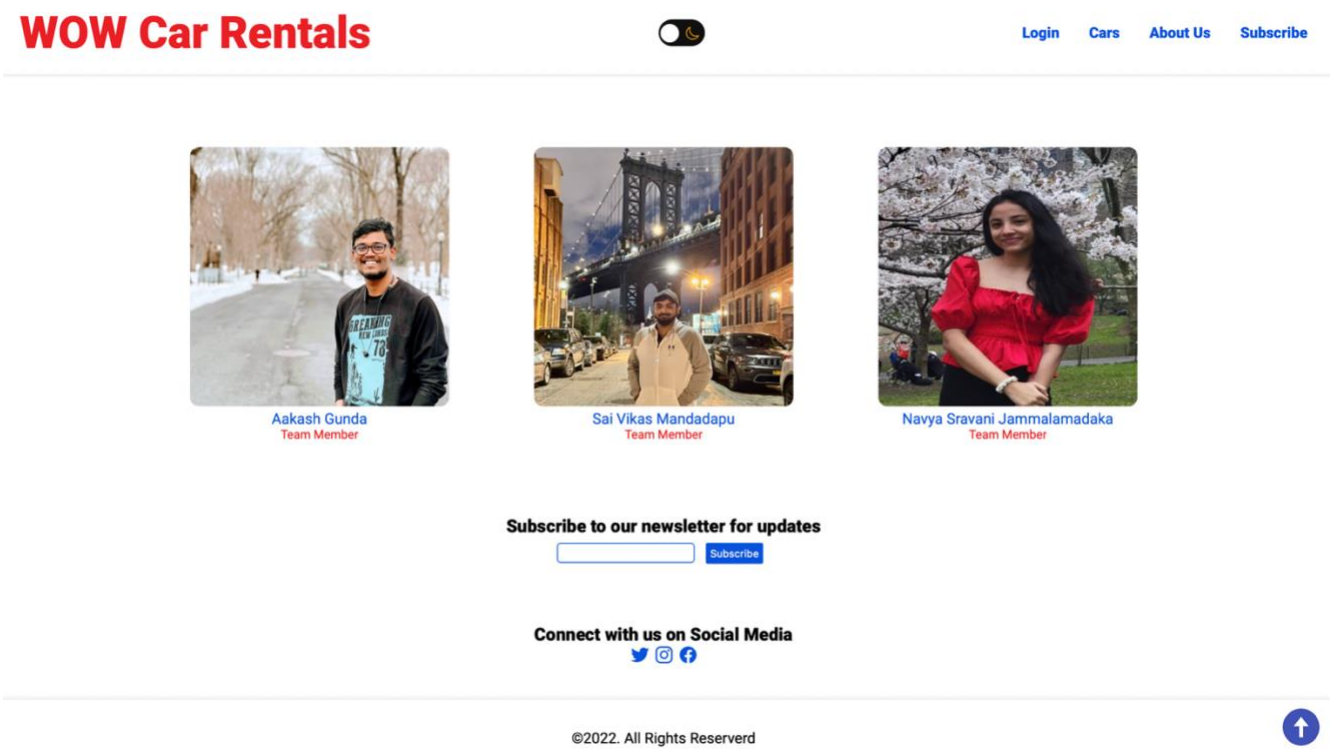
COUNT(*)
10

Screenshots of some sessions, pages, menus of our Web Application

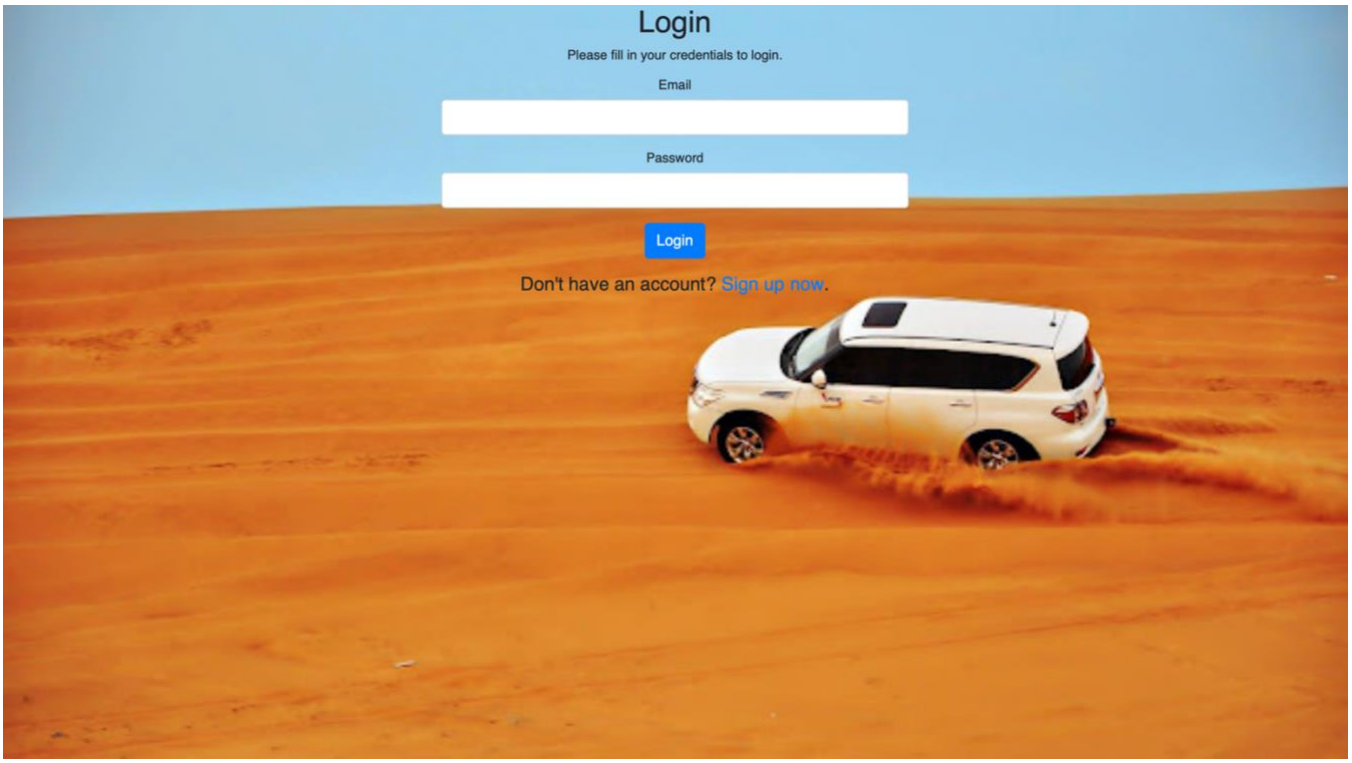
Welcome Page:



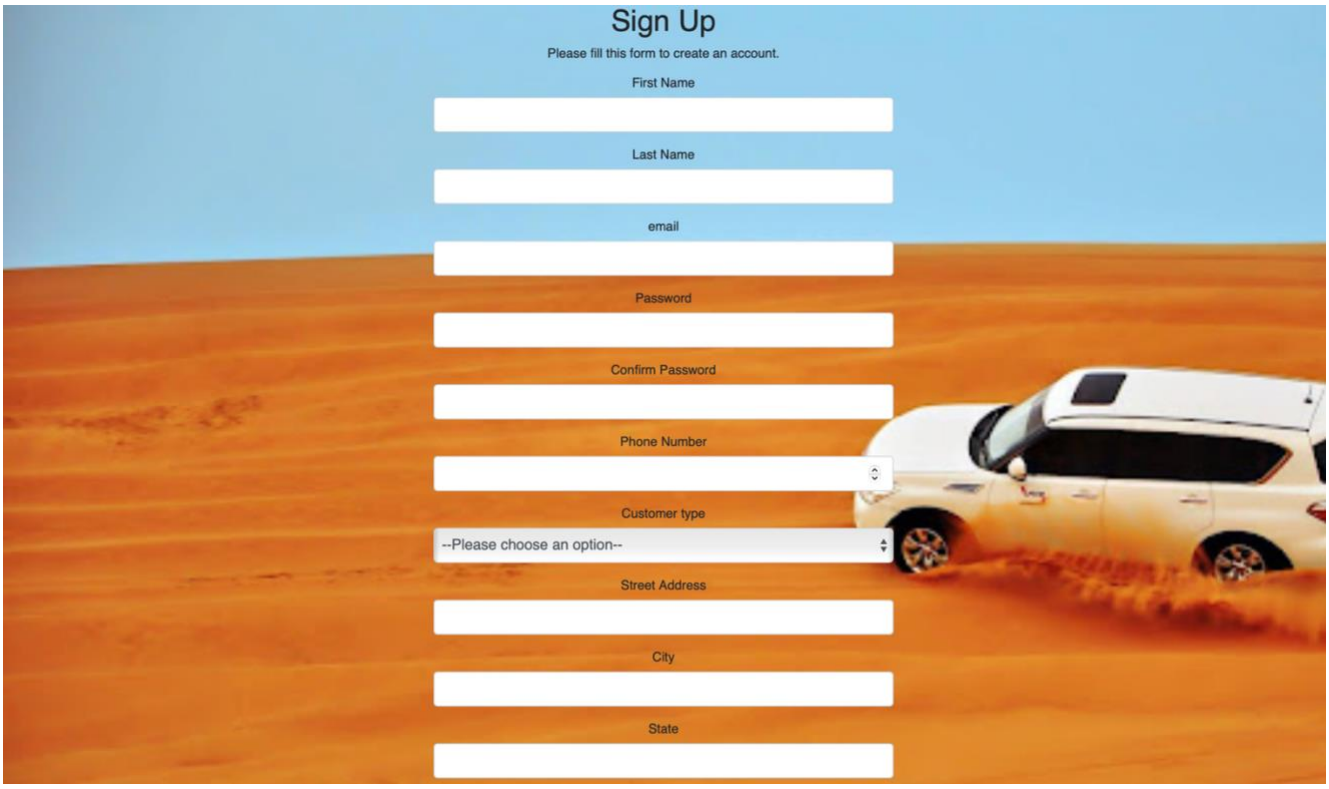



WOW Car Rentals
Login
Cars
About Us
Subscribe
Aakash Gunda
Team Member
Sai Vikas Mandadapu
Team Member
Navya Sravani Jammalamadaka
Team Member
Subscribe to our newsletter for updates
Subscribe
Connect with us on Social Media
©2022. All Rights Reserverd

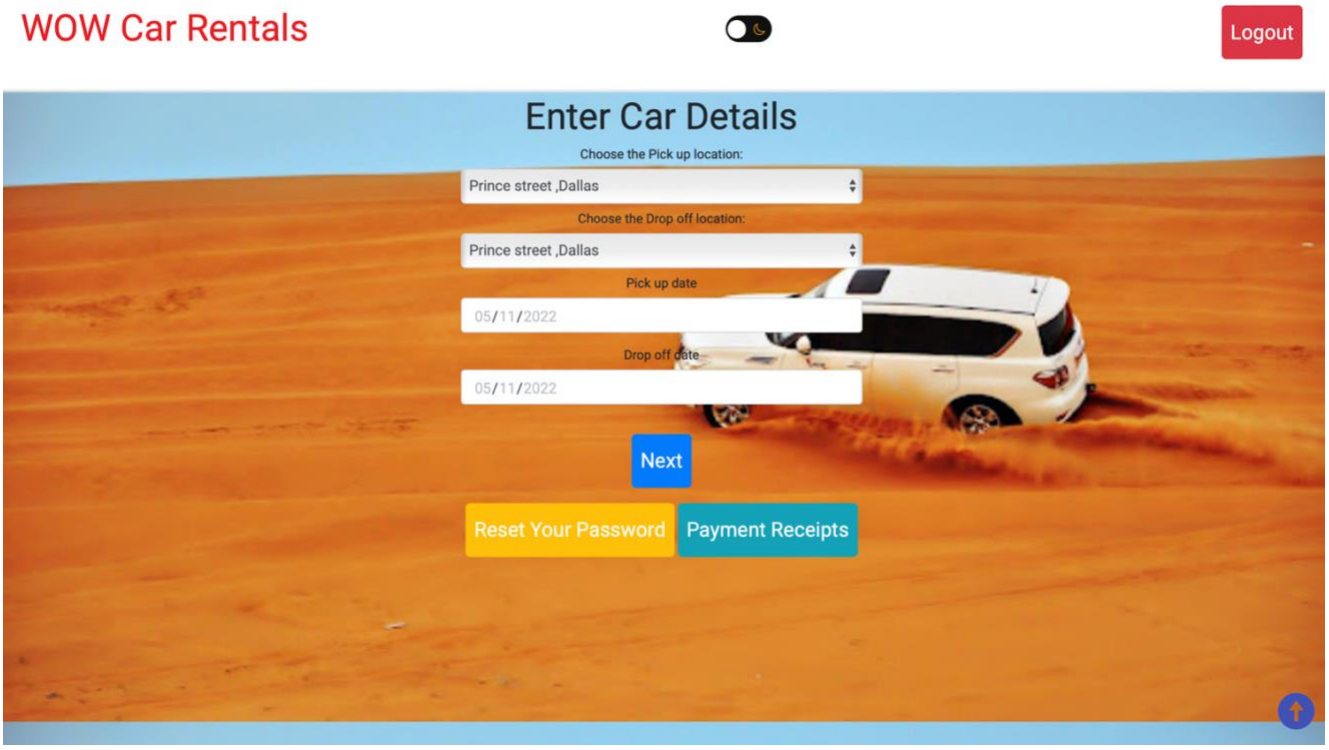
Login page:


Login
Please fill in your credentials to login.
Email
Password
Login
Don't have an account? Sign up now.

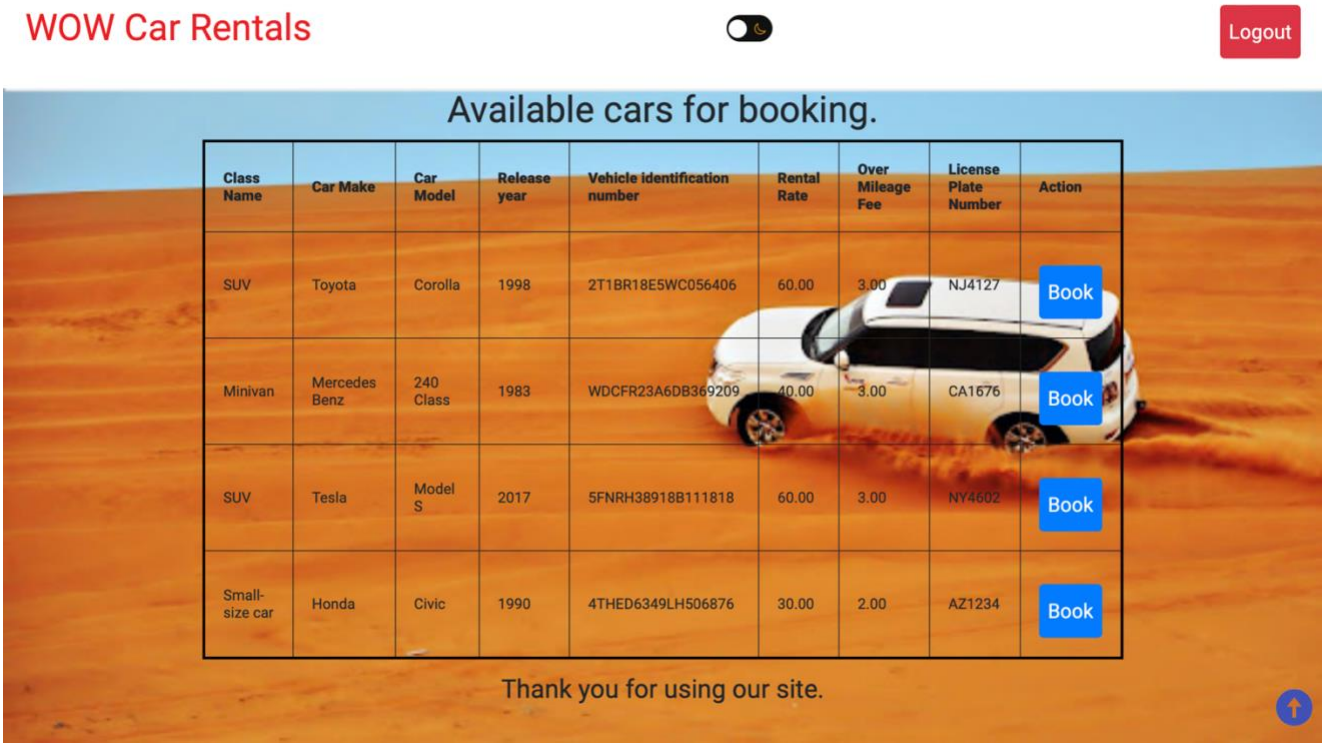
Registration page:



Booking details page:

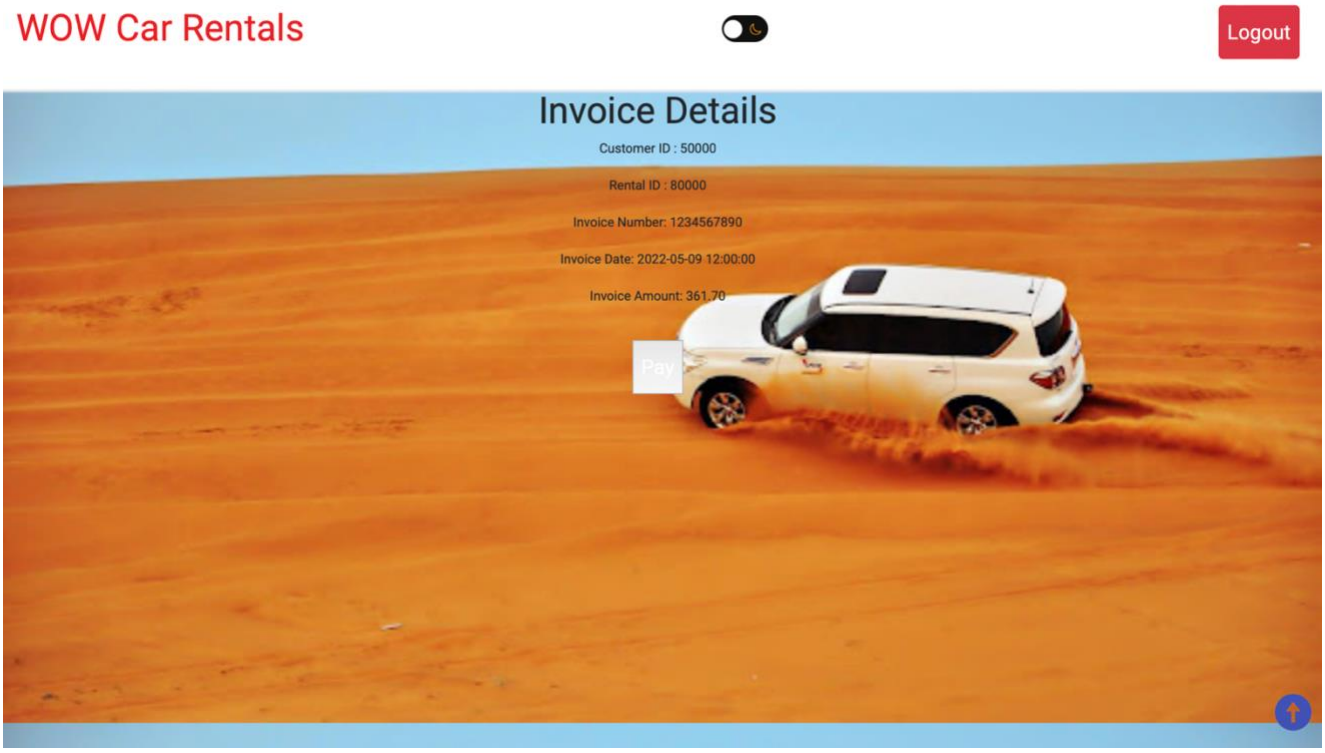


Available cars page:

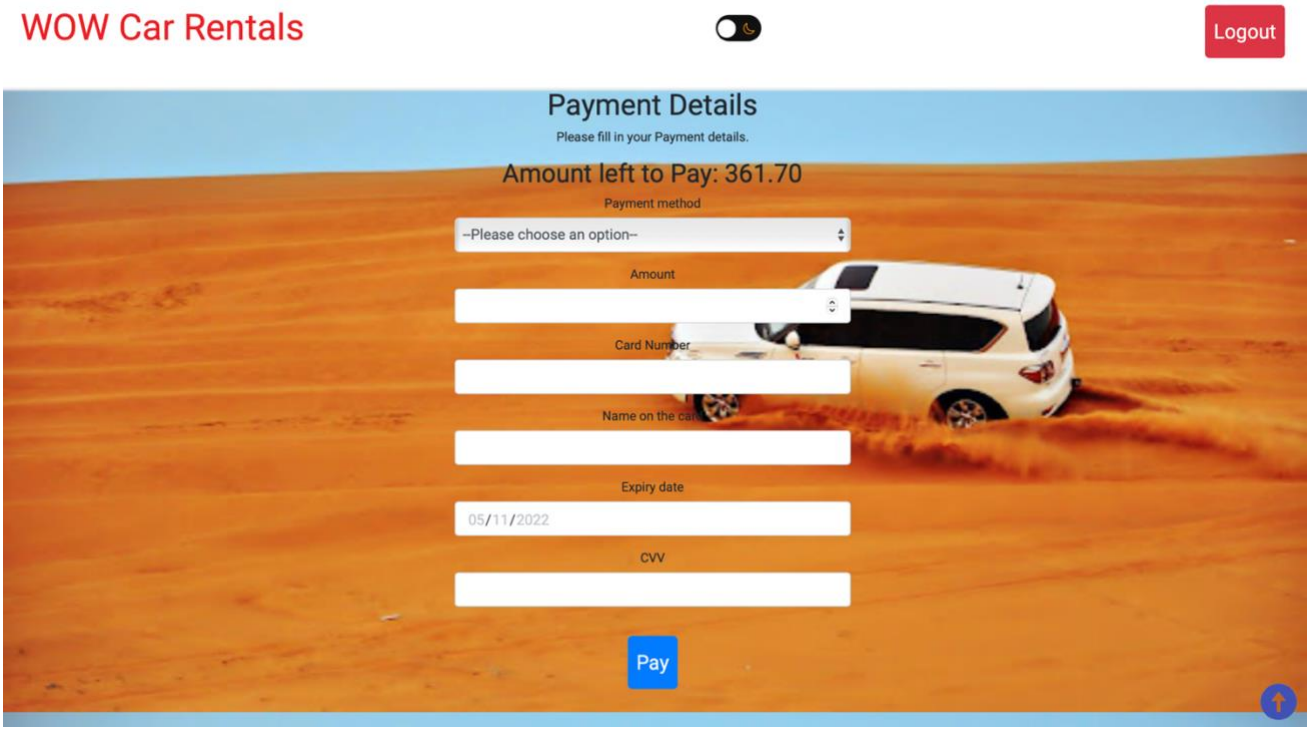


Class Name	Car Make	Car Model	Release year	Vehicle identification number	Rental Rate	Over Mileage Fee	License Plate Number	Action
SUV	Toyota	Corolla	1998	2T1BR18E5WC056406	60.00	3.00	NJ4127	Book
Minivan	Mercedes Benz	240 Class	1983	WDCFR23A6DB369209	40.00	3.00	CA1676	Book
SUV	Tesla	Model S	2017	5FNRH38918B111818	60.00	3.00	NY4602	Book
Small-size car	Honda	Civic	1990	4THED6349LH506876	30.00	2.00	AZ1234	Book

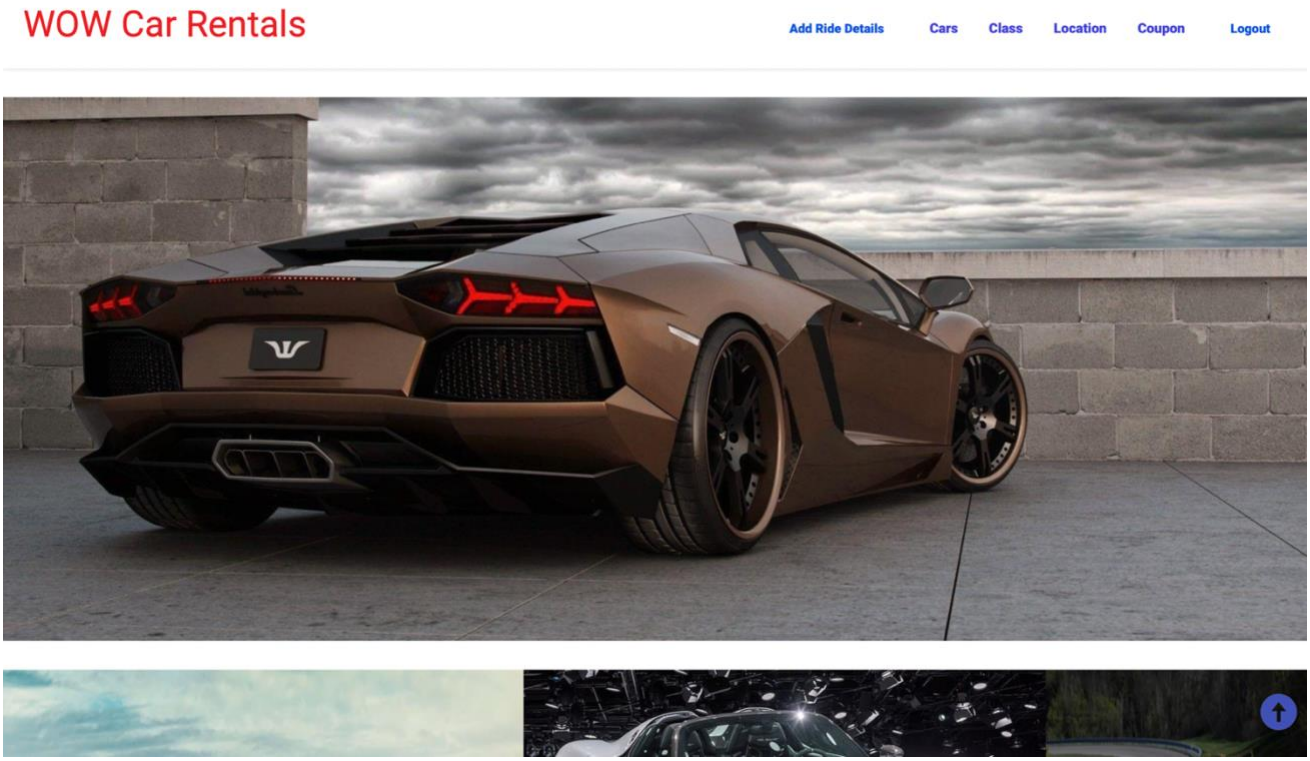
Invoice page:



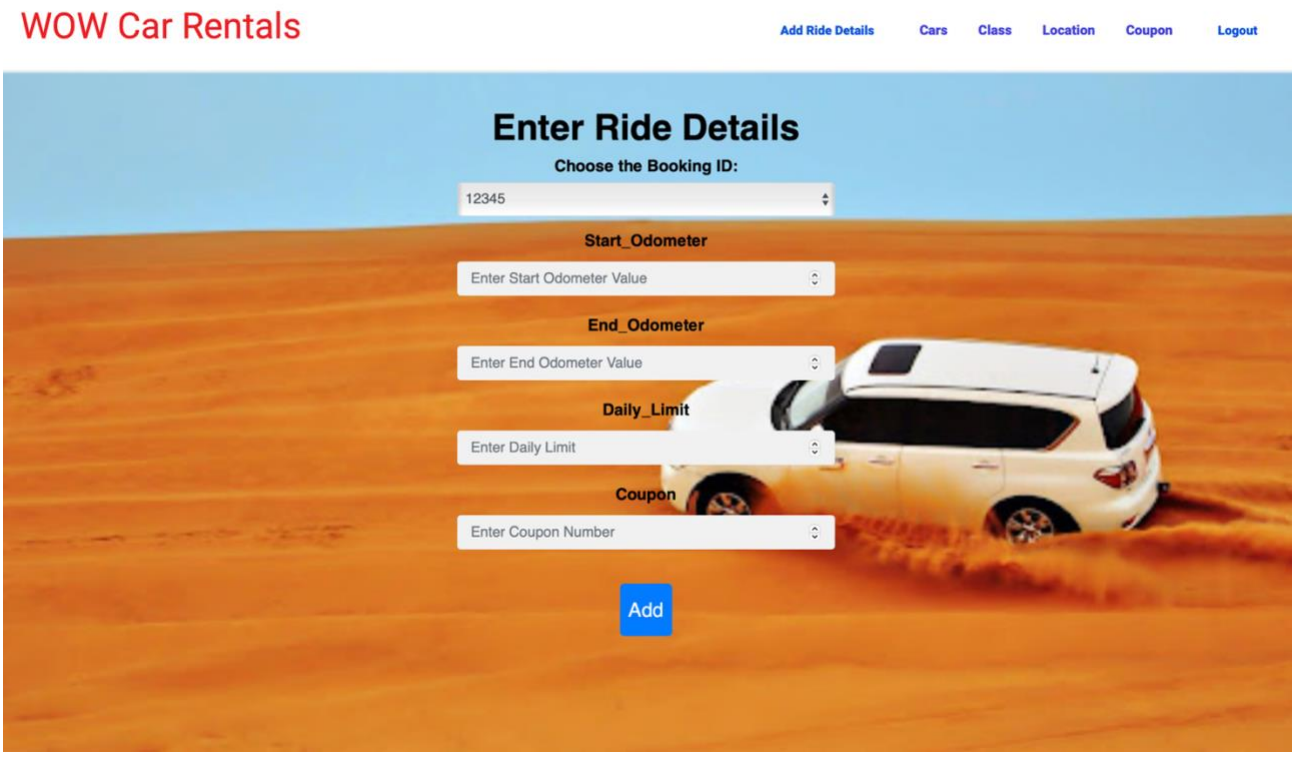
Payment page:



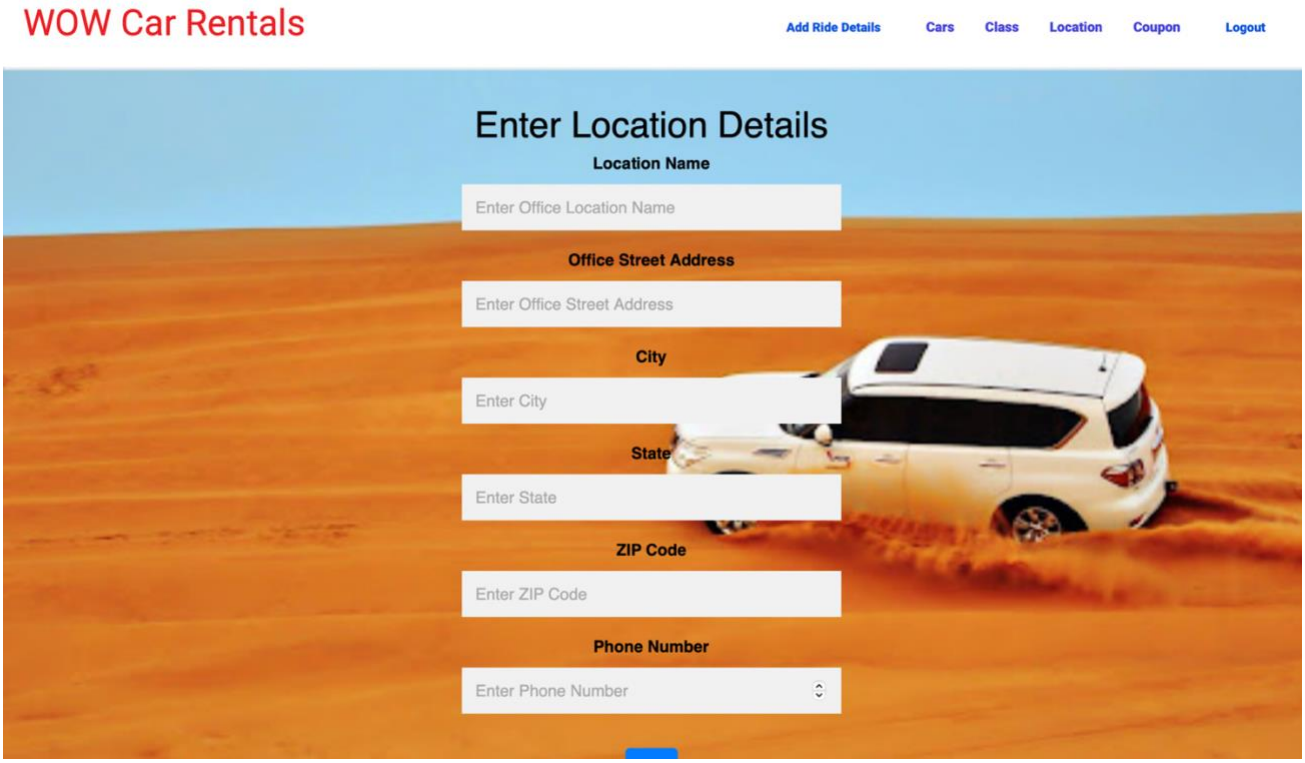
Admin main page:



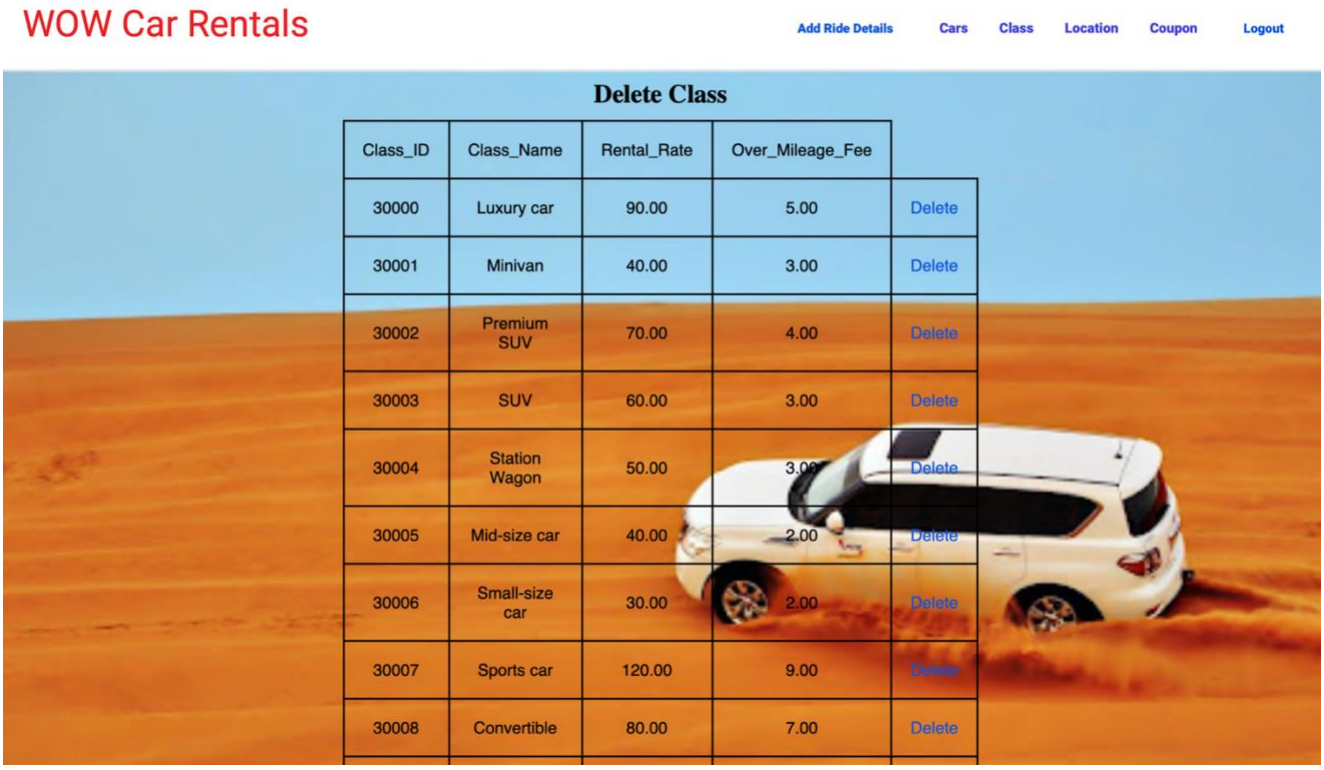
Add ride details page:



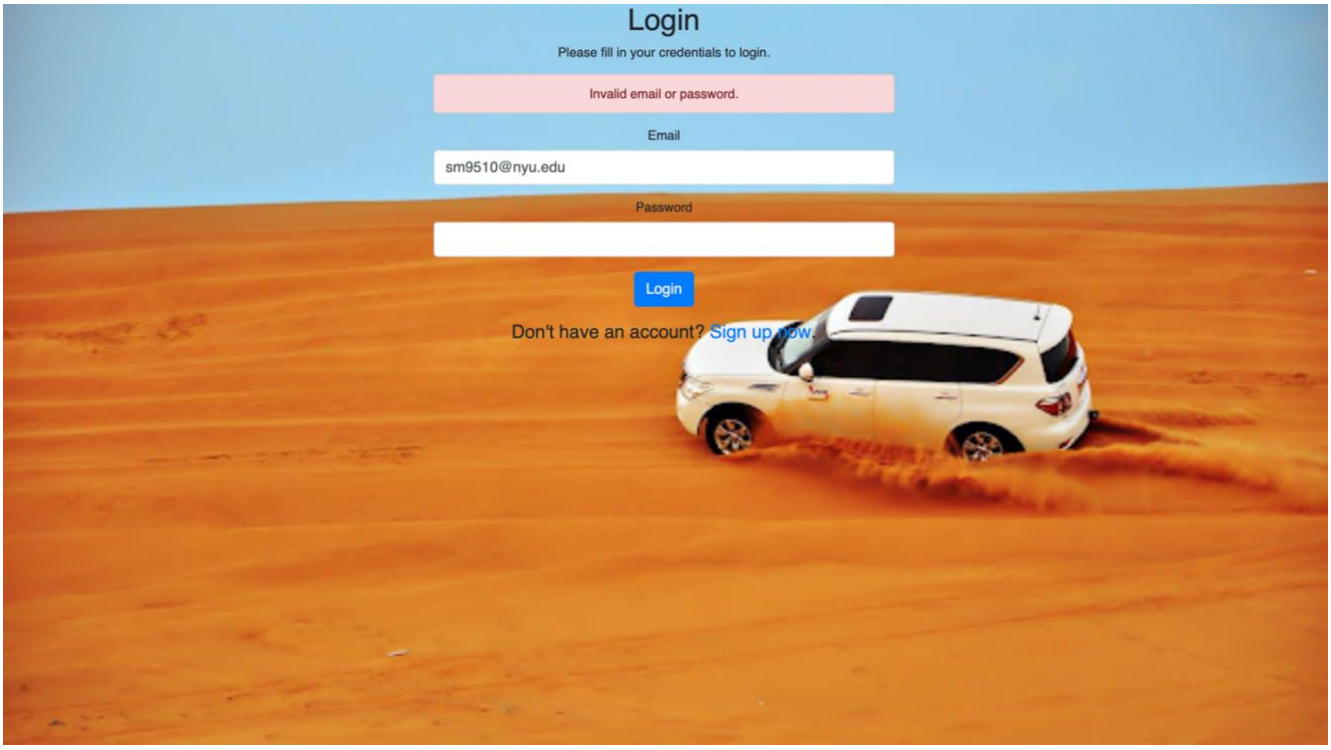
Add Location page:



Delete class page:

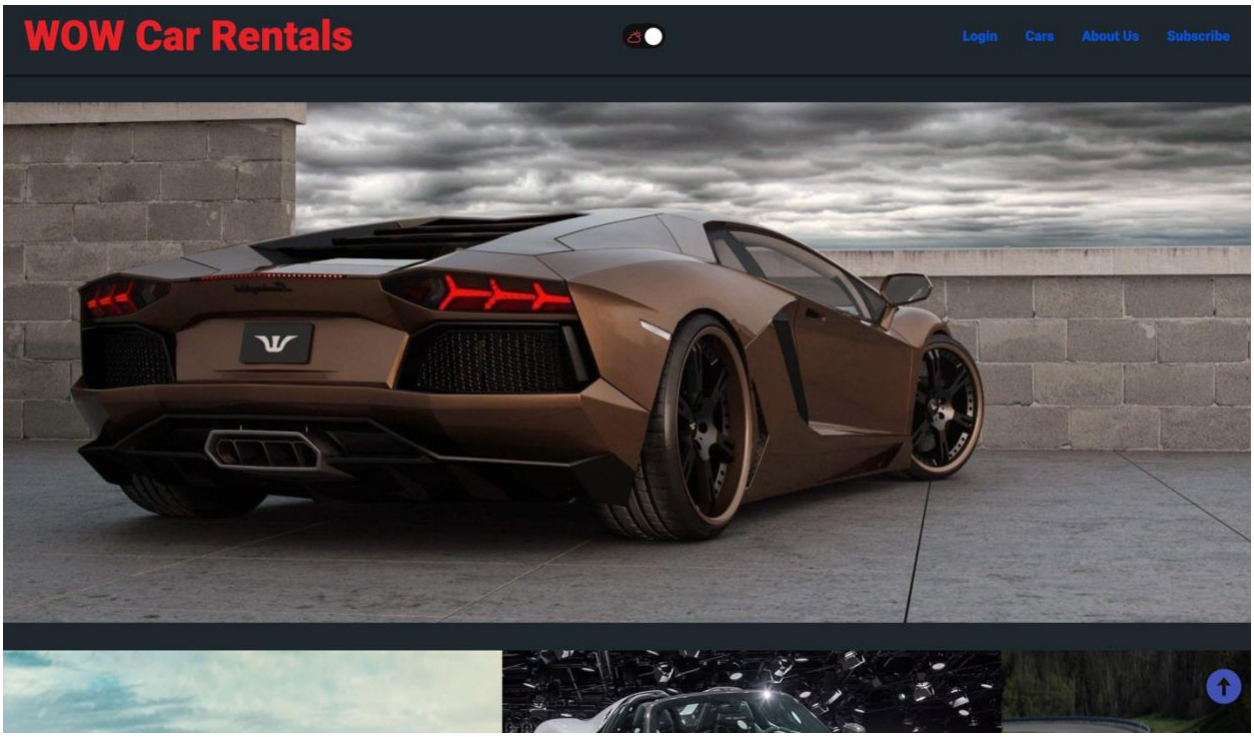


Invalid credentials:



Additional Features

Night Mode:



Security Features

Password Encryption:

The password is encrypted at backend. Our backend script is generating a hashed password using a password_hash function in php.

Form Validation:

All the forms will be validated, and user won't be able to enter the data if in the wrong input format. In the registration page, we are alerted by a message when we leave any field blank, or if passwords don't match, or if they are too simple and small, or if they are too similar to the username, or if we don't enter a valid email id. In the login page also, if we have entered incorrect details, we will be alerted via messages. Same thing happens in all the other pages where customers are employees required to enter data.

SQL Injection:

To prevent SQL Injection, we used MYSQLi prepared statements. In which a SQL query is prepared with empty values as placeholders, then variables are bind to the placeholders by stating each variable, along with its type and finally query is executed.

Cross-Site Scripting:

Htmlspecialchars() function is used for the prevention of Cross-Site Scripting which converts special characters to HTML entities.

User Authentication:

We have also made sure the login and registration page redirect to the user dashboard if the user is already logged in.

Lessons Learned

Being novices in the area of web page design and development it was initially very tough to decide what software to use and understand how to use them. After deciding on the software, it was also challenging to write code for the first few pages, but it got a bit easier once we started organizing and distributing the work among ourselves. One thing that we could have done better was to use the features like SQL injection and XSS from the get-go as it doubled our work when we did it after completing scripting for a few pages. What went well was that once we figured this out the rest of the process went very smoothly with us discussing the template code for the main page with all the features and headers thought out and then splitting the work. Time was a constraint especially in the second part of the project as all of us have different courses with different deadlines and commitments and it was challenging to set realistic deadlines and sync our timing for any meetings.

Business analysis with 6 SQLs using your project data

Q1) Table joins with at least 3 tables in join

Showing rows 0 - 9 (10 total, Query took 0.0030 seconds.)

```
SELECT nsa_customer.fname as "First Name", nsa_customer.lname as "Last Name", nsa_car.make as "Make",
nsa_car.model as "Model", nsa_rental.pickup_date as "Pick-up Date", nsa_rental.drop_off_date as "Drop-off Date"
FROM nsa_rental
JOIN nsa_customer ON nsa_customer.Customer_ID=nsa_rental.Customer_ID
JOIN nsa_car ON nsa_car.car_id=nsa_rental.car_id;
```

Enable foreign key checks

Go Cancel

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

First Name	Last Name	Make	Model	Pick-up Date	Drop-off Date
Aakash	Gunda	Mercedes Benz	240 Class	2022-05-10 00:00:00	2022-05-17 00:00:00
Navya	j	Audi	A4	2022-05-10 00:00:00	2022-05-12 00:00:00
sai vikas	mandadapu	Tesla	Model S	2022-05-12 00:00:00	2022-05-20 00:00:00
sachin	tendulkar	Audi	A4	2022-05-12 00:00:00	2022-05-19 00:00:00
Navya	j	Honda	Cr V	2022-05-21 00:00:00	2022-05-22 00:00:00
rohit	sharma	Ford	Aspire	2022-05-17 00:00:00	2022-05-20 00:00:00
rohit	sharma	Volvo	S80	2022-05-16 00:00:00	2022-05-27 00:00:00
virat	kohli	Ford	Edge	2022-05-21 00:00:00	2022-05-27 00:00:00
sai	m	Tesla	Model S	2022-05-18 00:00:00	2022-05-26 00:00:00
sai	m	Audi	A4	2022-05-15 00:00:00	2022-05-16 00:00:00

For the admin this query returns the details of customer name, the car they booked along with its pickup and drop off date for tracking purposes.

Q2) Multi-row subquery

Showing rows 0 - 2 (3 total, Query took 0.0018 seconds.)

```
select class_name "Class Name", rental_rate "Rental rate per day"
from nsa_class
where rental_rate > ANY (select 20*over_mileage_fee+ rental_rate from nsa_class);
```

Enable foreign key checks

Go Cancel

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Class Name	Rental rate per day
Edit Copy Delete	Luxury car	90.00
Edit Copy Delete	Sports car	120.00
Edit Copy Delete	Convertible	80.00

Check all *With selected:* Edit Copy Delete Export

This query gives us the details of the classes which would cost more just in the daily rental rate, than some other classes would in that class's daily rental rate + 20 mile worth of over-mileage, So customer can choose a class which will cost them less when they exceed the daily mileage limit.

Q3) Correlated subquery

Showing rows 0 - 2 (3 total, Query took 0.0030 seconds.)

```
SELECT fname as "First Name", lname as "Last Name"
FROM nsa_customer c
WHERE NOT EXISTS (select r.customer_id from nsa_rental r where c.customer_id = r.customer_id);
```

Enable foreign key checks

Go Cancel

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	First Name	Last Name
Edit Copy Delete	Ms	dhoni
Edit Copy Delete	R	pant
Edit Copy Delete	Joseph	A

This query shows the customers who created an account but never booked a car. So that admin can send promotional emails targeting these customers.

Q4) SET operator query

Showing rows 0 - 1 (2 total, Query took 0.0018 seconds.)

```
SELECT class_name as "Class Name", rental_rate as "Rental rate", over_mileage_fee as "Over mileage fee" FROM nsa_class where rental_rate > 80
UNION
SELECT class_name as "Class Name", rental_rate as "Rental rate", over_mileage_fee as "Over mileage fee" FROM nsa_class where over_mileage_fee > 7;
```

Enable foreign key checks

Go Cancel

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

Class Name	Rental rate	Over mileage fee
Luxury car	90.00	5.00
Sports car	120.00	9.00

This query gives the list of class whose rental rates or over mileage fees is higher than a maximum number of classes.

Q5) Query within line view or WITH clause

Showing rows 0 - 2 (3 total, Query took 0.0016 seconds.)

```
With DailyLimitNull AS
(Select rental_id as "Rental Id", cust_type as "Customer type",daily_limit as Daily_Limit , nsa_car.make as "Make", nsa_car.model as "Model" from nsa_rental
join nsa_car where nsa_car.car_id = nsa_rental.car_id)
Select * from DailyLimitNull where Daily_Limit is Null;
```

Enable foreign key checks

Go Cancel

[Edit inline] [Edit] [Create PHP code]

+ Options

	Rental Id	Customer type	Daily_Limit	Make	Model
Edit Copy Delete	80004	C	NULL	Honda	Cr V
Edit Copy Delete	80008	C	NULL	Tesla	Model S
Edit Copy Delete	80009	C	NULL	Audi	A4

This query provides us the information about cars whose daily limit is Null.

Q6) TOP-N query

Showing rows 0 - 4 (5 total, Query took 0.0014 seconds.)

```
select fname as "First Name", lname as "Last Name", invoice_amount as "Invoice Amount"
from nsa_customer join nsa_invoice on nsa_invoice.customer_id = nsa_customer.customer_id
order by nsa_invoice.invoice_amount DESC
LIMIT 5;
```

Enable foreign key checks

Go Cancel

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

First Name	Last Name	Invoice Amount
rohit	sharma	1019.20
sai	m	480.00
Aakash	Gunda	361.70
sai vikas	mandadapu	132.00
virat	kohli	56.80

This query gives a list of top five customers with most amount to pay.