Course: CSGY-6083-Principles of Database Systems

Section: B

Date of submission: Dec 10, 2023

Team Members

Name: Jie Cheng NetID: jc12300

Name: Yunling Bai NetID: yb2473

Name: Pengfei Zheng NetID: pz2270

Content

Summary

This report will introduce a design of a web-based user interface for business services in a car rental company called WOW (World on Wheels). In this web application, users are able to register and login to perform multiple business activities including booking a car and making payments. This could be achieved by creating programs invoked by a web server that will establish a connection with the database, and then submitting corresponding queries, ultimately presenting the outcomes in a web page.

The first part will provide details of the whole application designs in three aspects. To start with, the outline of the basic view of the relationship between different entities in this system will be specified in database design, which will then be implemented into a database management system. Based on the designed database, the interface will then be implemented through Java. After requests are sent from the client to the server and the stored procedures in the database are called, the results will be shown in the web page. This will be introduced and explained in detail in the web application and backend design.

1. Design Details

1.1 Database Design

1.1.1 ER Diagram

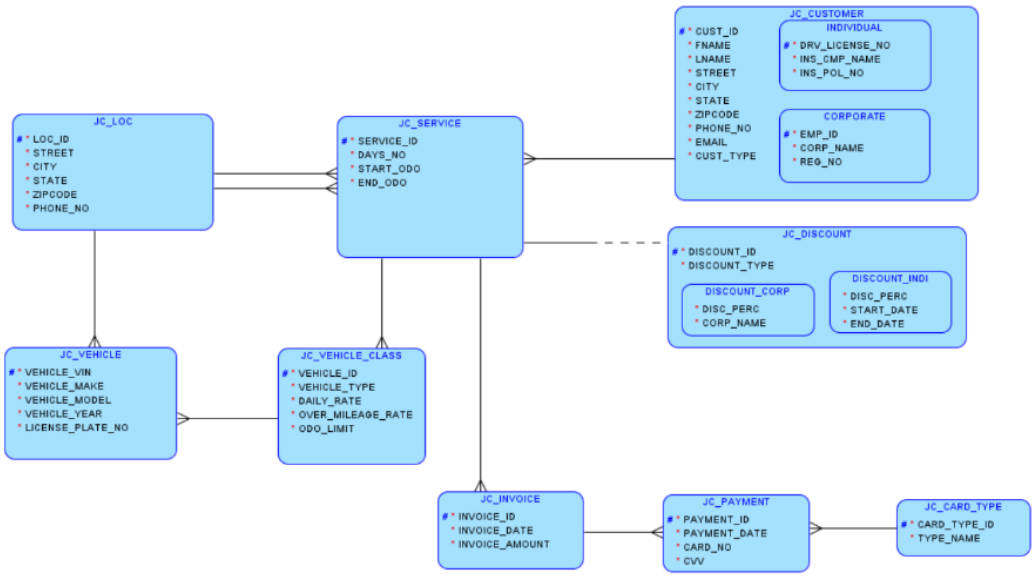
To start with, a low-level view of entities in this system needs to be created to specify their relationships. There are mainly nine entities that exist in the system: location, customer, service, vehicle, vehicle class, invoice, payment, discount, and card type. Customers can be identified by the customer id, and its attributes contain first name, last name, email, phone number, and address that can be divided into street, city, state, and zip code. The customers should also be divided into two subtypes: individual and corporate. For individual customers, they have attributes of driver license number, insurance company name and insurance policy number, while for corporate customers, their corporate name and registration number are stored as well as their employee id. Office locations can be identified by the location id, with specific addresses and phone number stored. For vehicle class, it can be identified by the vehicle id and its attributes include vehicle type, daily rate of service, over mileage rate and odometer limit. In each class, there are multiple vehicles available. They can be identified by their VIN (Vehicle Identification Number) and have attributes including make, model, year and license plate number. In the service entity, it can be specified by the service id, and they include the number of rental days, start and end odometer. In the discount entity identified by discount id, there are two subtypes of corporate and individual ones. Both include discount percentage, but the difference is that corporate discounts are differentiated through companies, thus an attribute of corporation name is needed. For individual discounts, they have validity dates, resulting in the storage of valid start date and valid end date. For the invoice identified by the invoice id, the invoice amount can be generated from the service with the invoice date included. Then the payment entity is identified by the payment id, and it has attributes of payment date, card number and CVV. Since customers can pay an invoice using multiple methods, the card type entity can be identified as card type id and has an attribute of type name.

The relationship between different entities can be summarized as follows.

- Location has a one-to-many relationship with Vehicle (one location can have many vehicles).
- Location has a one-to-many relationship with Service (one location can have many rental services).
- Vehicle Class has a one-to-many relationship with Vehicle (one class can have many vehicles).
- Customer has a one-to-many relationship with Service (one customer can have many rental services).
- Service has a one-to- many relationship with Invoice (each service can have many types of invoices depending on odometer.).
- Invoice has a one-to-many relationship with Payment (one invoice can have many payments).
- Customer has a subtype of Individual (for individual customers).
- Customer has a subtype of Corporate (for corporate customers).

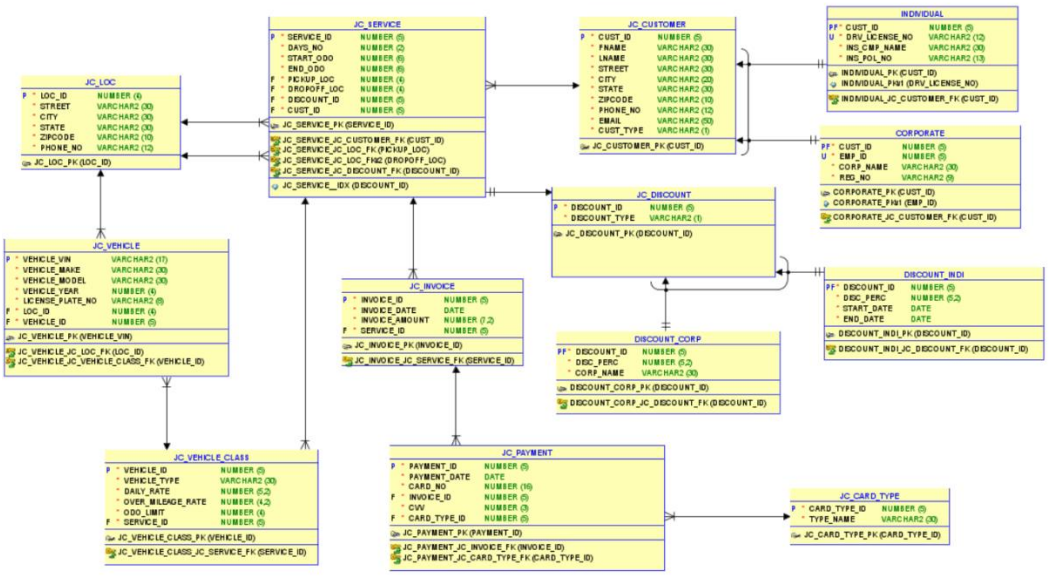
- Service has a one-to-one relationship with Discount (one rental service can only have one discount used and it is optional).

Relationships between these entities are denoted, and the cardinality of the relationships are also detailed in the following diagram.



1.1.2 Relational Model

After the establishment of the ER diagram, we have to translate it into a relational model. For all one-to-many relationships we insert the primary key from the one-side to the many-side and modify the foreign key name. For example, since each location has many vehicles, the primary key should be inserted from the location table to the vehicle table as a foreign key. For the relationships that are many-many, we create a separate table for the relationship with the primary key of both tables. Below is the result of converting the ER diagram into the relational schema.

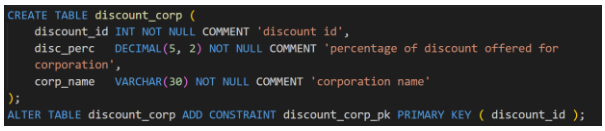


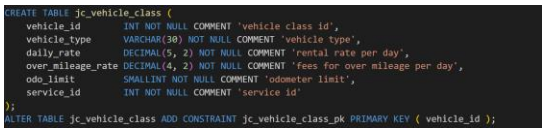
1.1.3 Assumptions and Constraints

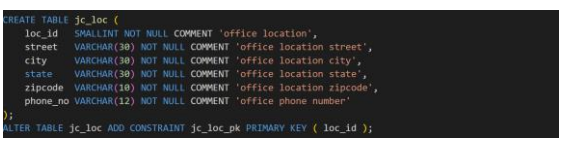
- The Employee ID of the customer and the registration number of the corporation are both unique within a corporate account. Therefore, the employee id can be used as the primary key in the corporate table. The driver license number of the customer is also unique within an individual account, thus similarly, it can be used as the primary key in the individual table.
- Discounts, whether individual or corporate, are applied manually by the staff at the time of renting a vehicle. Because the business case doesn't specify an automated system for applying discounts; hence, it's assumed that staff handles this process. A discount coupon can only be used once, and it cannot be applied retroactively to past rental services, and it is only available within the valid period.
- It's assumed that a customer can have multiple rental services, but only one active rental service at a time.
- Assume that individual customers must provide proof of insurance policy details before renting a vehicle, and this information must be verified and recorded.

1.1.4 Database Setup

Below is the setup for the database with constraints, primary/foreign keys, and type definitions for all attributes for the tables discussed in the relational schema.








```
CREATE TABLE jc_discount (
    discount_id   INT NOT NULL COMMENT 'discount id',
    discount_type VARCHAR(1) NOT NULL COMMENT 'discount type'
);
ALTER TABLE jc_discount
    ADD CONSTRAINT ch_inh_jc_discount CHECK ( discount_type IN ( 'C', 'I' ) );
ALTER TABLE jc_discount ADD CONSTRAINT jc_discount_pk PRIMARY KEY ( discount_id );
```

```
CREATE TABLE jc_service (
    service_id   INT NOT NULL COMMENT 'rent service id',
    days_no      TINYINT NOT NULL COMMENT 'rent service days number',
    start_odo    INT NOT NULL COMMENT 'rent service start odometer',
    end_odo      INT NOT NULL COMMENT 'rent service end odometer',
    pickup_loc   SMALLINT NOT NULL COMMENT 'pickup location',
    dropoff_loc SMALLINT NOT NULL COMMENT 'dropoff location',
    discount_id INT NOT NULL COMMENT 'discount id',
    cust_id      INT NOT NULL
);

CREATE UNIQUE INDEX jc_service__idx ON
    jc_service (
        discount_id
    ASC );
ALTER TABLE jc_service ADD CONSTRAINT jc_service_pk PRIMARY KEY ( service_id );
```

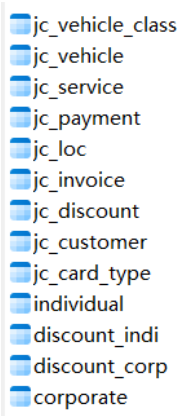
```
CREATE TABLE jc_customer (
    cust_id    INT NOT NULL COMMENT 'customer id',
    fname      VARCHAR(30) NOT NULL COMMENT 'customer first name',
    lname      VARCHAR(30) NOT NULL COMMENT 'customer last name',
    street     VARCHAR(30) NOT NULL COMMENT 'customer street',
    city       VARCHAR(20) NOT NULL COMMENT 'customer city',
    state      VARCHAR(30) NOT NULL COMMENT 'customer state',
    zipcode    VARCHAR(10) NOT NULL COMMENT 'customer zipcode',
    phone_no   VARCHAR(12) NOT NULL COMMENT 'customer phone number',
    email      VARCHAR(50) NOT NULL COMMENT 'customer email',
    cust_type VARCHAR(1) NOT NULL COMMENT 'customer type'
);
ALTER TABLE jc_customer
    ADD CONSTRAINT ch_inh_jc_customer CHECK ( cust_type IN ( 'C', 'I' ) );
ALTER TABLE jc_customer ADD CONSTRAINT jc_customer_pk PRIMARY KEY ( cust_id );
```

```
ALTER TABLE corporate
    ADD CONSTRAINT corporate_jc_customer_fk FOREIGN KEY ( cust_id )
        REFERENCES jc_customer ( cust_id );

ALTER TABLE discount_corp
    ADD CONSTRAINT discount_corp_jc_discount_fk FOREIGN KEY ( discount_id )
        REFERENCES jc_discount ( discount_id );

ALTER TABLE discount_indi
    ADD CONSTRAINT discount_indi_jc_discount_fk FOREIGN KEY ( discount_id )
        REFERENCES jc_discount ( discount_id );

ALTER TABLE individual
    ADD CONSTRAINT individual_jc_customer_fk FOREIGN KEY ( cust_id )
        REFERENCES jc_customer ( cust_id );

ALTER TABLE jc_invoice
    ADD CONSTRAINT jc_invoice_jc_service_fk FOREIGN KEY ( service_id )
        REFERENCES jc_service ( service_id );

ALTER TABLE jc_payment
    ADD CONSTRAINT jc_payment_jc_card_type_fk FOREIGN KEY ( card_type_id )
        REFERENCES jc_card_type ( card_type_id );

ALTER TABLE jc_payment
    ADD CONSTRAINT jc_payment_jc_invoice_fk FOREIGN KEY ( invoice_id )
        REFERENCES jc_invoice ( invoice_id );
```

```
ALTER TABLE jc_service
    ADD CONSTRAINT jc_service_jc_customer_fk FOREIGN KEY ( cust_id )
        REFERENCES jc_customer ( cust_id );

ALTER TABLE jc_service
    ADD CONSTRAINT jc_service_jc_discount_fk FOREIGN KEY ( discount_id )
        REFERENCES jc_discount ( discount_id );

ALTER TABLE jc_service
    ADD CONSTRAINT jc_service_jc_loc_fk FOREIGN KEY ( pickup_loc )
        REFERENCES jc_loc ( loc_id );

ALTER TABLE jc_service
    ADD CONSTRAINT jc_service_jc_loc_fkv2 FOREIGN KEY ( dropoff_loc )
        REFERENCES jc_loc ( loc_id );

ALTER TABLE jc_vehicle_class
    ADD CONSTRAINT jc_vehicle_class_jc_service_fk FOREIGN KEY ( service_id )
        REFERENCES jc_service ( service_id );

ALTER TABLE jc_vehicle
    ADD CONSTRAINT jc_vehicle_jc_loc_fk FOREIGN KEY ( loc_id )
        REFERENCES jc_loc ( loc_id );

ALTER TABLE jc_vehicle
    ADD CONSTRAINT jc_vehicle_jc_vehicle_class_fk FOREIGN KEY ( vehicle_id )
        REFERENCES jc_vehicle_class ( vehicle_id );
```

1.1.5 List of Tables

After implementing the data description language code into MySQL, the list of tables could be shown. After populating some sample data into the tables using data manipulation language code, we can clearly observe the data in the following visual interface.



vehicle_vin	vehicle_make	vehicle_model	vehicle_year	license_plate_no	loc_id	vehicle_id
18802705293939440	Nissan	Altima	2023	JKL04100	2	10
20635667483605990	Toyota	Camry	2023	GHI78900	4	6
25242520074326360	Honda	Elantra	2023	JKL01200	3	3
34645433723964390	Toyota	Elantra	2023	JKL04420	2	9
36417030095670650	Toyota	Elantra	2020	GHI71010	2	2
38068192468764550	Honda	Elantra	2022	GHI74020	2	4
39787974838173440	Hyundai	Accord	2020	JKL02400	3	7
40061061306428900	Toyota	Elantra	2021	ABC18800	2	8
56473604679613320	Honda	Accord	2022	DEF18105	3	7
56710278506513240	Hyundai	Accord	2023	GHI46020	4	2
64548180238807220	Toyota	Camry	2020	DEF45600	1	1
70108202227182780	Nissan	Camry	2022	DEF50542	1	5
77807017992587660	Honda	Accord	2022	ABC12300	1	2
84759324439018280	Honda	Camry	2022	ABC17600	4	1
88558226746779550	Honda	Elantra	2021	JKL05000	4	5
91044619254230110	Hyundai	Camry	2021	DEF40273	5	6
93630501216869070	Hyundai	Altima	2020	JKL00570	3	5
93972144626475160	Honda	Accord	2021	GHI74055	1	4
96000604123026110	Toyota	Altima	2021	JKL04050	1	6
96476313716194310	Honda	Altima	2023	ABC10573	3	1

cust_id	emp_id	corp_name	reg_no
1	101	Acme Corporation	123456789
2	102	Initech	987654321
3	103	Wayne Enterprises	098765432
4	104	LexCorp	321654987
5	105	Stark Industries	765432198
6	106	Umbrella Corporation	321456789
7	107	Google	987321654
8	108	Microsoft	098216543
9	109	Apple	216549873
10	110	Amazon	654321987

discount_id	disc_perc	corp_name
1	10.00	Acme Corporation
2	15.00	Initech
3	20.00	Wayne Enterprises
4	25.00	LexCorp
5	30.00	Stark Industries
6	35.00	Umbrella Corporation
7	40.00	Google
8	45.00	Microsoft
9	50.00	Apple
10	55.00	Amazon

discount_id	disc_perc	start_date	end_date
11	60.00	2023-08-01 00:00:00	2023-11-21 00:00:00
12	65.00	2023-09-10 00:00:00	2023-12-25 00:00:00
13	15.00	2023-10-01 00:00:00	2023-11-18 00:00:00
14	25.00	2023-10-08 00:00:00	2023-11-24 00:00:00
15	12.00	2023-10-15 00:00:00	2023-11-25 00:00:00
16	35.00	2023-10-23 00:00:00	2023-11-29 00:00:00
17	11.00	2023-11-01 00:00:00	2024-01-21 00:00:00
18	18.00	2023-11-01 00:00:00	2024-01-27 00:00:00
19	22.00	2023-11-01 00:00:00	2024-01-25 00:00:00
20	16.00	2023-11-01 00:00:00	2024-02-12 00:00:00

cust_id	drv_license_no	ins_cmp_name	ins_pol_no
11	67890	American Family Insurance	0165000498730
12	01234	Mercury Insurance	0543000219870
13	56789	Nationwide General Insurance	0894500061230
14	90123	Erie Insurance Exchange	9076594643210
15	23456	SFMA Insurance Company	1234567000890
16	78901	PM Insurance Company	9076500043210
17	12340	GG Insurance Company	0980650004320
18	67895	APC Insurance Company	2105490008730
19	85231	USAAC Insurance Company	6543219000870
20	23450	NM Insurance Company	8894561200030

cust_id	fname	lname	street	city	state	zipcode	phone_no	email	cust_type
1	David	Doe	123 Main Street	Los Angeles	CA	14886	123-456-7890	David.Doe@example.co	C
2	Jane	Brown	456 Elm Street	Jersey City	NJ	59641	987-654-3210	Jane.Brown@example.cc	C
3	Mary	Williams	789 Oak Street	San Diego	CA	23233	987-453-3210	Mary.Williams@example	C
4	Linda	Smith	1 SAGE PL	Manhanttan	NY	16642	456-789-0183	Linda.Smith@example.c	C
5	Patricia	Brown	900 AVENUE C	Fresno	CA	69840	456-754-0123	Patricia.Brown@example	C
6	Jane	Garcia	301 HULSE LANDING RD	Modesto	CA	54530	987-654-8643	Jane.Garcia@example.cc	C
7	William	Williams	1125 ROUTE 112	Riverside	CA	84818	789-012-3456	William.Williams@examp	C
8	David	Brown	700 GARNSEY RD	Long Beach	CA	90106	789-012-8466	David.Brown@example.c	C
9	Susan	Williams	8 TWISTING DR	Anaheim	CA	41109	184-456-7890	Susan.Williams@exampl	C
10	William	Jones	9162 HIGH BRIDGE RD	Irvine	CA	68815	789-012-4256	William.Jones@example	C
11	Emily	Davis	2436 Naples Avenue	Panama City	FL	32405	133-456-7890	Emily.Davis@example.cc	I
12	Ryan	Mitchell	1030 E Main St	Benton Harbor	MI	49022	438-654-3210	Ryan.Mitchell@example	I
13	Olivia	Garcia	244 W 109TH ST	Brooklyn	NY	10025	450-654-3210	Olivia.Garcia@example.c	I
14	Ethan	Anderson	1561 SHERIDAN AVE	Manhanttan	NY	16642	843-789-0123	Ethan.Anderson@examp	I
15	Ava	Taylor	301 CAROLINE ST	Fresno	CA	69840	456-108-0123	Ava.Taylor@example.co	I
16	Benjamin	Martinez	5400 DUNNING AVE	Modesto	CA	54530	987-435-3210	Benjamin.Martinez@exa	I
17	Sophia	Brown	73 LINDEN ST	Riverside	CA	84818	789-012-4851	Sophia.Brown@example	I
18	Jackson	Wright	30 SAINT JOHN ST	Long Beach	CA	90106	789-012-4350	Jackson.Wright@examp	I
19	Mia	Robinson	482 VAN DUZER ST	Anaheim	CA	41109	123-456-1571	Mia.Robinson@example	I
20	Noah	Adams	723 BROWN ST	Irvine	CA	68815	789-012-4287	Noah.Adams@example.	I

discount_id	discount_type
1	C
2	C
3	C
4	C
5	C
6	C
7	C
8	C
9	C
10	C
11	I
12	I
13	I
14	I
15	I
16	I
17	I
18	I
19	I
20	I

card_type_id	type_name
1	Visa
2	Mastercard
3	American Express
4	Discover
5	JCB
6	Diners Club
7	Carte Blanche
8	China UnionPay
9	Maestro
10	Electron

loc_id	street	city	state	zipcode	phone_no
1	201 PROSPECT ST	Jersey City	NJ	07306	735-428-8435
2	456 Elm Street	Manhattan	NY	10013	542-416-1864
3	3881 SEDGWICK AVE	San Diego	CA	92101	743-410-3542
4	1 BARBARA RD	Modesto	OH	95354	150-348-8020
5	107 CLARENDON RD	Riverside	MN	92501	848-181-4610
6	1 VERNON ST	Anaheim	GA	92803	105-543-3570
7	150 BRABANT ST	Irvine	FL	92612	135-654-3418
8	6 W 107TH ST	Fresno	CT	93701	843-420-2345
9	35 TRENT CT	Long Beach	SD	90801	405-345-3458
10	13 Main Street	Los Angeles	LA	90802	435-802-1505

invoice_id	invoice_date	invoice_amount	service_id
1	2023-11-01 00:00:00	286.05	20
2	2023-11-01 00:00:00	260.32	18
3	2023-11-02 00:00:00	458.65	19
4	2023-11-03 00:00:00	364.14	1
5	2023-11-05 00:00:00	176.61	2
6	2023-11-05 00:00:00	412.26	17
7	2023-11-07 00:00:00	424.74	16
8	2023-11-07 00:00:00	120.17	3
9	2023-11-07 00:00:00	365.60	4
10	2023-11-08 00:00:00	401.10	15
11	2023-11-09 00:00:00	491.33	9
12	2023-11-10 00:00:00	390.93	5
13	2023-11-10 00:00:00	486.97	6
14	2023-11-10 00:00:00	351.70	13
15	2023-11-11 00:00:00	251.70	14
16	2023-11-12 00:00:00	428.99	7
17	2023-11-13 00:00:00	183.92	8
18	2023-11-14 00:00:00	114.53	12
19	2023-11-14 00:00:00	362.90	11
20	2023-11-15 00:00:00	172.14	10

vehicle_id	vehicle_type	daily_rate	over_mileage_rate	odo_limit	service_id
1	Economy	50.00	1.00	100	1
2	Compact	40.00	1.00	150	2
3	Midsize	55.00	2.00	200	3
4	Fullsize	65.00	2.00	250	4
5	SUV	70.00	2.00	300	5
6	Van	50.00	2.00	350	6
7	Luxury	75.00	3.00	400	7
8	Sports	60.00	1.00	200	8
9	Exotic	70.00	4.00	500	9
10	Unlimited	50.00	2.00	200	10

service_id	days_no	start_odo	end_odo	pickup_loc	dropoff_loc	discount_id	cust_id
1	2	10000	10500	1	2	1	1
2	1	12000	12700	3	4	2	2
3	2	14000	14500	5	6	3	3
4	6	16000	16700	7	8	4	4
5	2	18000	18500	9	10	5	5
6	3	20000	20700	1	3	6	6
7	5	22000	22700	3	5	7	7
8	3	24000	24700	5	10	8	8
9	2	26000	26700	7	9	9	9
10	7	28000	29000	4	1	10	10
11	2	30000	30800	5	7	11	11
12	5	50000	52000	2	8	12	12
13	2	58000	58500	9	1	13	13
14	4	84000	86000	8	1	14	14
15	2	95000	96500	4	7	15	15
16	7	18000	19000	10	1	16	16
17	2	10000	10800	5	7	17	17
18	5	20000	22000	2	8	18	18
19	2	78000	78500	9	1	19	19
20	4	64000	66000	8	1	20	20

payment_id	payment_date	card_no	invoice_id	cvv	card_type_id
1	2023-11-01 00:00:00	5592056333172992	1	123	4
2	2023-11-12 00:00:00	6544036909269729	2	456	5
3	2023-11-07 00:00:00	5901389979804023	3	789	3
4	2023-11-12 00:00:00	7101689179311568	4	167	1
5	2023-11-10 00:00:00	1765842134041871	5	943	2
6	2023-11-08 00:00:00	6624503305630504	6	640	4
7	2023-11-09 00:00:00	4722663911676210	7	701	5
8	2023-11-12 00:00:00	9279689068045021	8	940	3
9	2023-11-11 00:00:00	4002158299984822	9	852	2
10	2023-11-13 00:00:00	5769850775546851	10	744	2
11	2023-11-16 00:00:00	1635375209274466	11	726	5
12	2023-11-19 00:00:00	7967389041059884	12	127	5
13	2023-11-20 00:00:00	3252814464085864	13	712	6
14	2023-11-21 00:00:00	1575499649453264	14	987	4
15	2023-11-16 00:00:00	8407612688356883	15	554	5
16	2023-11-18 00:00:00	7483665849065060	16	637	8
17	2023-11-17 00:00:00	7728705484790868	17	998	5
18	2023-11-20 00:00:00	9414730357026336	18	734	3
19	2023-11-16 00:00:00	2495898899739702	19	100	7
20	2023-11-17 00:00:00	7173884435759535	20	855	5
21	2023-11-18 00:00:00	9816574641576445	15	633	7
22	2023-11-19 00:00:00	5378327876357835	18	348	8
23	2023-11-20 00:00:00	7837453787534873	19	168	5
24	2023-11-21 00:00:00	7832453787537837	20	866	9
25	2023-11-22 00:00:00	7854563783735743	20	597	5

1.1.6 Triggers and Procedures

Since customers and discounts should be both presented in two types: corporate or individual, we could create useful triggers to help identify the correct customer or discount type.

```
DROP TRIGGER IF EXISTS arc_fkarc_4_discount_indi;

DELIMITER //
CREATE TRIGGER arc_fkarc_4_discount_indi BEFORE INSERT ON discount_indi
FOR EACH ROW
BEGIN
    DECLARE d VARCHAR(1);

    SELECT discount_type INTO d
    FROM jc_discount a
    WHERE a.discount_id = NEW.discount_id;

    IF (d IS NULL OR d <> 'I') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'FK DISCOUNT_INDI_JC_DISCOUNT_FK in Table
        DISCOUNT_INDI violates Arc constraint on Table JC_DISCOUNT -
        discriminator column DISCOUNT_TYPE doesn''t have value ''I''';
    END IF;
END;
//
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS arc_fkarc_4_discount_corp;

DELIMITER //
CREATE TRIGGER arc_fkarc_4_discount_corp BEFORE INSERT ON discount_corp
    FOR EACH ROW
BEGIN
    DECLARE d VARCHAR(1);

    SELECT discount_type INTO d
    FROM jc_discount a
    WHERE a.discount_id = NEW.discount_id
    LIMIT 1;

    IF (d IS NULL OR d <> 'C') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'FK DISCOUNT_CORP_JC_DISCOUNT_FK in Table
        DISCOUNT_CORP violates Arc constraint on Table JC_DISCOUNT -
        discriminator column DISCOUNT_TYPE doesn''t have value ''C''';
    END IF;
END;
//
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS arc_fkarc_3_corporate;

DELIMITER //
CREATE TRIGGER arc_fkarc_3_corporate BEFORE INSERT ON corporate
    FOR EACH ROW
BEGIN
    DECLARE d VARCHAR(1);

    SELECT cust_type INTO d
    FROM jc_customer a
    WHERE a.cust_id = NEW.cust_id
    LIMIT 1;

    IF (d IS NULL OR d <> 'C') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'FK CORPORATE_JC_CUSTOMER_FK in Table
        CORPORATE violates Arc constraint on Table JC_CUSTOMER -
        discriminator column CUST_TYPE doesn''t have value ''C''';
    END IF;
END;
//
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS arc_fkarc_3_individual;

DELIMITER //
CREATE TRIGGER arc_fkarc_3_individual BEFORE INSERT ON individual
    FOR EACH ROW
BEGIN
    DECLARE d VARCHAR(1);

    SELECT cust_type INTO d
    FROM jc_customer a
    WHERE a.cust_id = NEW.cust_id
    LIMIT 1;

    IF (d IS NULL OR d <> 'I') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'FK INDIVIDUAL_JC_CUSTOMER_FK in Table
        INDIVIDUAL violates Arc constraint on Table JC_CUSTOMER -
        discriminator column CUST_TYPE doesn''t have value ''I''';
    END IF;
END;
//
DELIMITER ;
```

For each rental service provided, an invoice with Invoice Date, and Invoice Amount should be generated when the vehicle is returned. Thus, a database trigger to calculate the invoice amount is necessary. In simple terms, the total invoice amount could be represented as the sum of rental fee and extra fee. The rental fee could be calculated as the product of the daily rate of service and the number of rental days while the extra fee is composed of the product of the over mileage fee and the extra part of miles subtracted from the total odometer limit (daily limit times rental days).

```
DROP TRIGGER IF EXISTS calculate_invoice;

DELIMITER //
CREATE TRIGGER calculate_invoice
BEFORE INSERT ON jc_invoice
FOR EACH ROW
BEGIN
    DECLARE v_daily_rate        DECIMAL(5, 2);
    DECLARE v_over_mileage_rate DECIMAL(4, 2);
    DECLARE v_extra_miles       INT;

    SELECT vc.daily_rate, vc.over_mileage_rate
    INTO v_daily_rate, v_over_mileage_rate
    FROM jc_vehicle_class vc
    WHERE vc.service_id = (SELECT service_id FROM jc_service WHERE service_id = NEW.service_id);

    SELECT GREATEST(end_odo - start_odo - (days_no * vc.odo_limit), 0)
    INTO v_extra_miles
    FROM jc_service
    WHERE service_id = NEW.service_id;

    SET NEW.invoice_amount = (days_no * v_daily_rate) + (v_extra_miles * v_over_mileage_rate);
END;
//
DELIMITER ;
```

1.2 Web Application Design

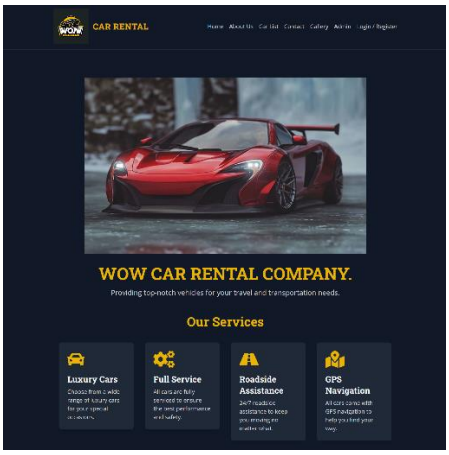
According to project requirements, we design a web application for car rental which has two main interfaces: a customer interface and an administrator interface. In this section, this application will be presented in these two function parts.

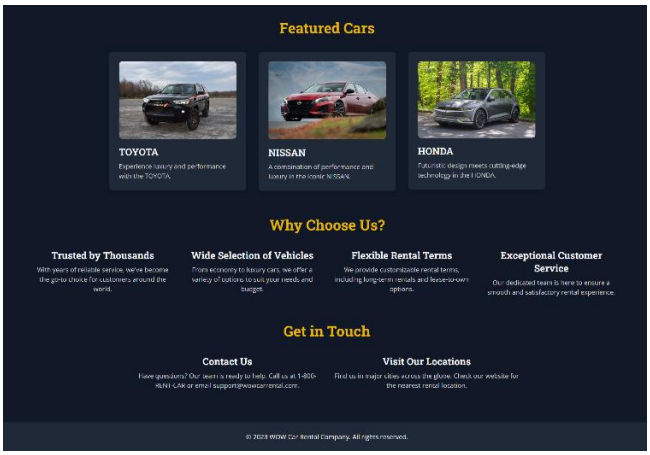
1.2.1 Customer Interface

The customer interface is the main way for users to interact with the application. The home page of the customer interface contains a menu with several functional options which is shown below.

- **Home**

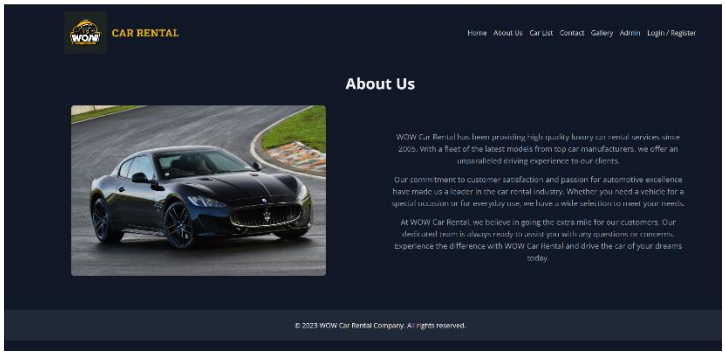
The top of the page features the application's logo, along with a menu that allows users to navigate to different sections of the application. The menu includes options for logging in or registering, viewing vehicles, and logging in as an administrator. Below the menu, the home page features a section that highlights the featured vehicles. These vehicles are typically new or popular models in promotion. The bottom of the home page features a section that provides information about the car rental company, such as its contact information, company's advantages, and address information of local stores.





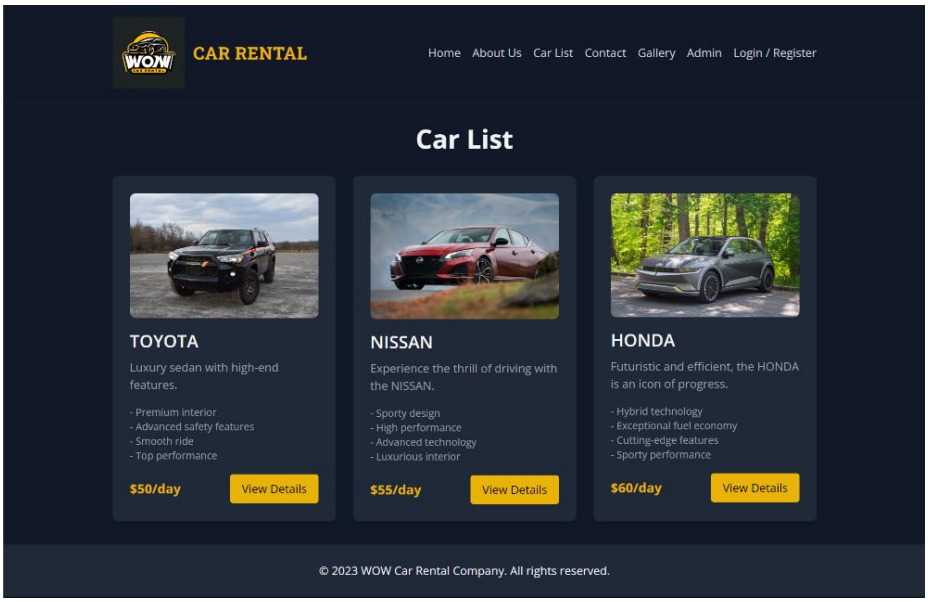
- **About US**

The "About Us" page is designed to help users get information about our company briefly. It provides users with a good understanding of the company's history, mission, and values.



- **Car List**

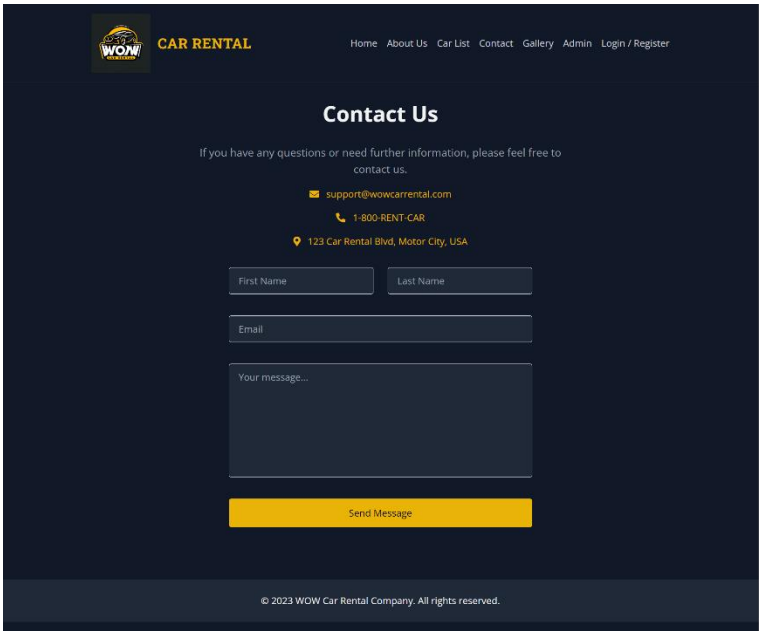
The Car list page displays a list of vehicles by presenting pictures, main features and rent price for each vehicle. In this interface, users can quickly understand the vehicle information and choose the vehicle they are interested in. Then, they can obtain more details about the vehicle by clicking on the button under the information box.



The further page provides more information about the vehicle, such as its make, model, year, price, and availability. At the bottom of the vehicle's details page is a booking form. Users can use this form to book the vehicle. The booking form requires users to provide their name, contact information, and rental information, such as the dates and locations they want to rent the vehicle.

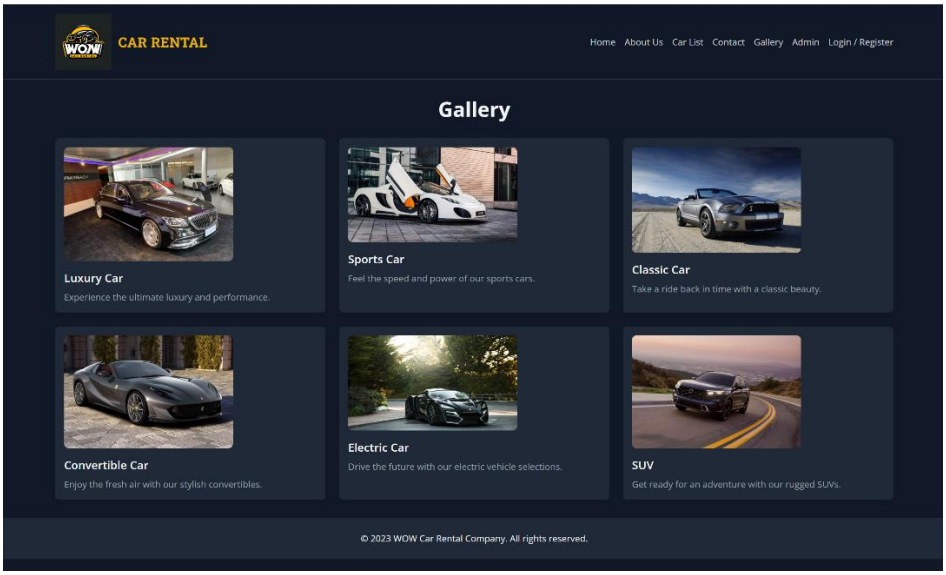
- **Contact**

This page provides a message form for users to contact Customer Service.



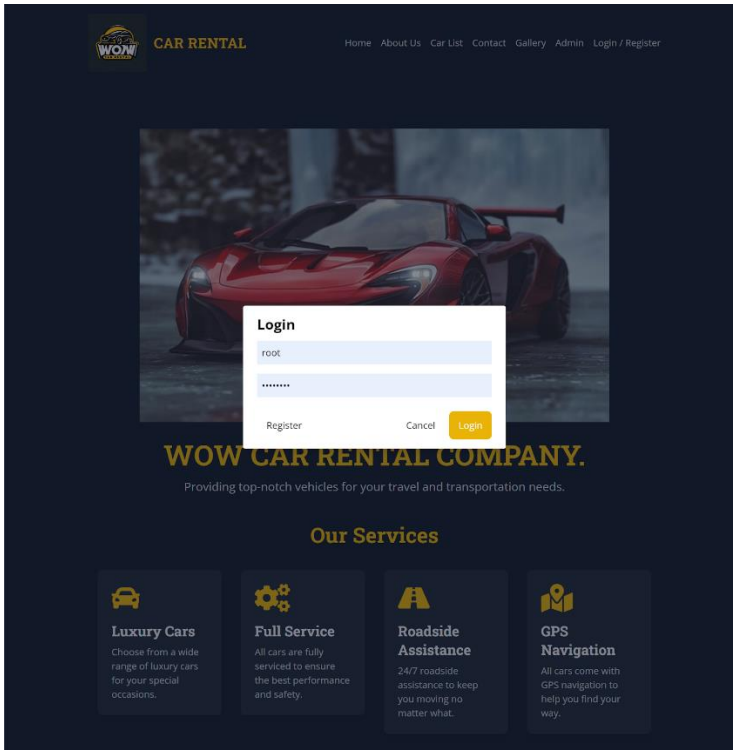
- **Gallery**

Gallery is designed to demonstrate vehicles according to different vehicle classes, including Luxury Car, Sports Car, SUV etc.



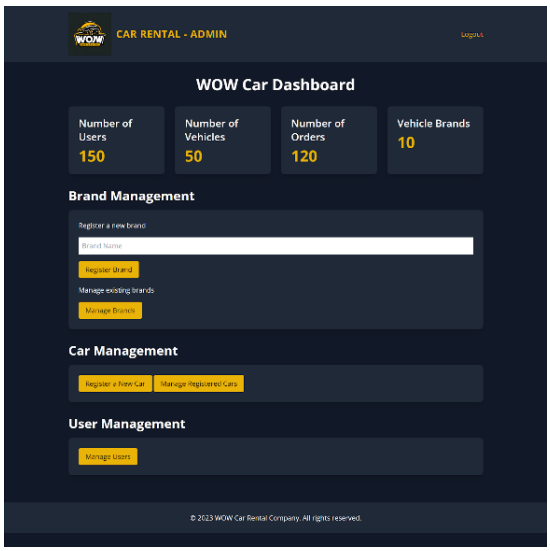
- **Admin Login/Logout**

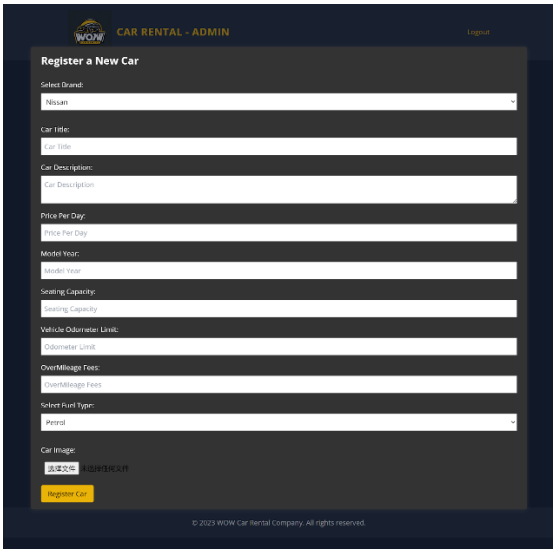
This is the Login/logout interface for administrators. Users can login to their account or register for a new account.



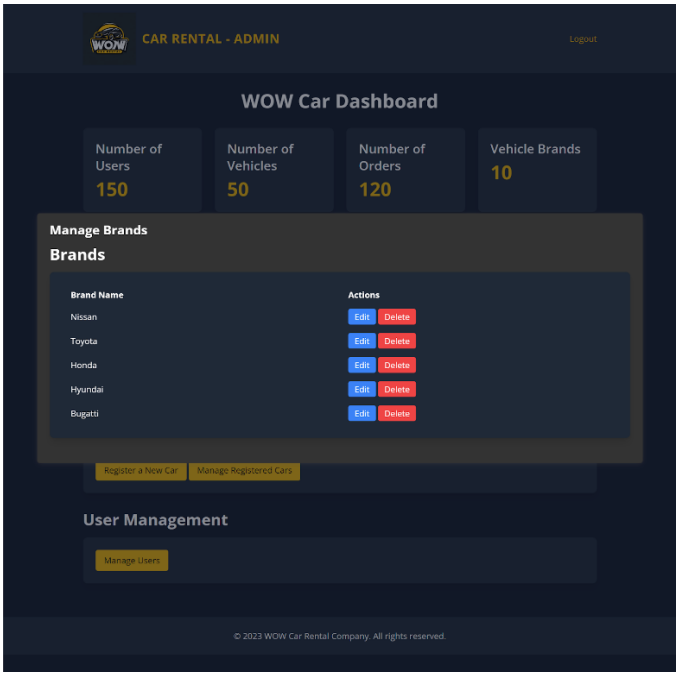
1.2.2 Administrator Interface

The administrator interface is used by administrators to manage the application. At the top of the dashboard of the administrator interface, there is a summary of the application's data, including numbers of users, vehicles, orders, and vehicle brands.





Below the summary table are several functional sections: brand management, car management and user management. All these three parts can realize corresponding SQL operations in the database: CREATE, READ, UPDATE and DELETE.



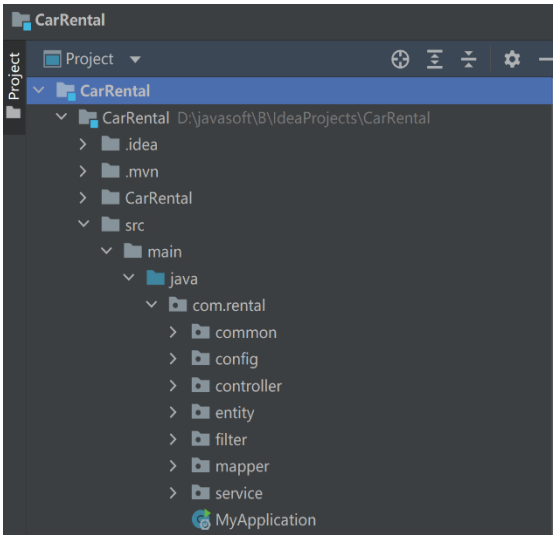
By clicking on the profile button, administrators can change their password. Administrators can verify their identity by entering the original password, then enter the new password twice. If the passwords they input twice are the same, the reset operation is successful after clicking the button.

1.3 Backend Design

The WOW CAR RENTAL backend program is a Java Spring Boot and MyBatis application that provides the functionality for the car rental web application.

1.3.1 Architecture

- Controller: The controller layer is responsible for handling HTTP requests from the web application. It prepares the response based on the service's output and sends it back to the web application.
- Service: The service layer implements the business logic of the application.
- Entity: This layer Represents the data models of the application, mapping to database tables and defining the structure of data objects.
- Mapper: The mapper layer Uses MyBatis to simplify data access operations. It helps to translate data between objects and databases.



1.3.2 Security

The WOW CAR RENTAL backend program takes security seriously. The following security measures are implemented.

- Password encryption: Passwords are encrypted before they are stored in the database.

This backend program prioritizes user security by utilizing the SHA-3 hashing algorithm for password encryption.

```
public class BCSha3Utils {

    // SHA3-224
    public static String sha3224(byte[] bytes) {
        Digest digest = new SHA3Digest(224);
        digest.update(bytes, 0, bytes.length);
        byte[] rsData = new byte[digest.getDigestSize()];
        digest.doFinal(rsData, 0);
        return Hex.toHexString(rsData);
    }

    // SHA3-256
    public static String sha3256(byte[] bytes) {
        Digest digest = new SHA3Digest(256);
        digest.update(bytes, 0, bytes.length);
        byte[] rsData = new byte[digest.getDigestSize()];
        digest.doFinal(rsData, 0);
        return Hex.toHexString(rsData);
    }

    // SHA3-384
    public static String sha3384(byte[] bytes) {
        Digest digest = new SHA3Digest(384);
        digest.update(bytes, 0, bytes.length);
        byte[] rsData = new byte[digest.getDigestSize()];
        digest.doFinal(rsData, 0);
        return Hex.toHexString(rsData);
    }
```

```
    // SHA3-512
    public static String sha3512(byte[] bytes) {
        Digest digest = new SHA3Digest(512);
        digest.update(bytes, 0, bytes.length);
        byte[] rsData = new byte[digest.getDigestSize()];
        digest.doFinal(rsData, 0);
        return Hex.toHexString(rsData);
    }

    // SHAKE-128
    public static String shake128(byte[] bytes) {
        Digest digest = new SHAKEDigest(128);
        digest.update(bytes, 0, bytes.length);
        byte[] rsData = new byte[digest.getDigestSize()];
        digest.doFinal(rsData, 0);
        return Hex.toHexString(rsData);
    }

    // SHAKE-256
    public static String shake256(byte[] bytes) {
        Digest digest = new SHAKEDigest(256);
        digest.update(bytes, 0, bytes.length);
        byte[] rsData = new byte[digest.getDigestSize()];
        digest.doFinal(rsData, 0);
        return Hex.toHexString(rsData);
    }
```

SHA-3, also known as Keccak, is a cryptographic hash function employed for data integrity verification and secure password storage. We choose this algorithm because it has high collision resistance and security level.

- SQL injection prevention: The pre-prepared statement method is used to prevent SQL injection attacks.

The pre-prepared statement method allows the program to specify the SQL statement to be executed before the statement is actually executed. This helps to prevent attackers from injecting malicious SQL code into the statement.

```
String sql="SELECT * FROM users WHERE username = ? AND password = ?";
        try(PreparedStatement preparedStatement=connection.prepareStatement(sql)){
        preparedStatement.setString(1,username);
        preparedStatement.setString(2,password);
        ResultSet resultSet=preparedStatement.executeQuery();
        // ...
        }
```

2. Reflections

Reflecting on the development of the WOW CAR RENTAL project, incorporating Flask and MySQL for the web application was a key learning experience. Here are some detailed insights and lessons learned from the project.

2.1 Flask as a Versatile Framework

Utilizing Flask, a lightweight web framework, was a strategic decision that paid off. Its simplicity and flexibility allowed for rapid development and easy integration with MySQL. The framework's ability to handle requests, manage sessions, and render templates dynamically played a crucial role in the project's success.

2.2 Database Integration and Security

Implementing MySQL with Flask showcased the importance of robust database design and secure integration. The use of password hashing with “werkzeug.security” for user authentication was a critical security measure. However, this also highlighted the need for constant vigilance against potential security vulnerabilities, especially in web-based applications dealing with sensitive user data.

2.3 Error Handling and Debugging

The use of try-except blocks for error handling in database interactions was crucial. Running the application in debug mode aided in identifying and resolving issues promptly. This approach was effective for immediate error tracking but also underscored the need for a more sophisticated error logging and handling system for larger-scale applications.

What Could Be Improved?

- Incorporating more advanced security features and error handling mechanisms.
- Enhancing the scalability and flexibility of the application for future expansion.
- Improving remote team coordination and project management methodologies.

Overall, the WOW CAR RENTAL project was a valuable learning experience, offering insights into web application development, database management, and the importance of security and user experience in software projects.

3. Sample Session

3.1 Demo

In this session, we will make a sample demonstration of the project.

Assume that we are a new user who has come to the website and the homepage is auto displayed. After registration and logging in with the information provided, we are in the logged in status. We will then be able to click the “car list” on the top navigation bar to enter the car list page that displays various sorts of cars. We can click “view details” from each card to get access to the detailed information about a particular car and will then be brought to the page for car details. On this page, we can book this car by choosing the appropriate date period and locations for pick-up and drop-off. After that, we are able to input the payment information including name on card, card number, and CVV as well as some personal information for billing. By clicking the button “Book now”, a prompt status for booking will be displayed and then we will be redirected to the “my bookings” page where we could find out the latest orders with invoice information attached. We are also capable of changing and updating the personal information in the profile page including the full name, email address, phone number, and address.

For the admin page, we can use the root password to log into the dashboard of the admin page. In this admin page, we have the number of users, vehicles, orders, and vehicle brands displayed. We can also implement some management on the car brand, car, or the user by editing or deleting the user, cars, and car brands.

3.2 Business Analysis

For business analysis with different SQLs using the project data, we will show the results as well as the business question that the query is answering.

- **Table Joins with at least 3 Tables in Join**

```
SELECT jc_customer.fname, jc_customer.lname, jc_service.days_no
FROM jc_customer
JOIN jc_service ON jc_customer.cust_id = jc_service.cust_id
JOIN jc_vehicle_class ON jc_service.service_id = jc_vehicle_class.service_id;
```

This query retrieves the first name, last name of customers, and the number of days of service for each rental along with the vehicle information. This query helps in understanding customer details and the duration of their vehicle rental. It is used in the “My book” page when retrieving user booking data and then integrating into the booking information.

fname	lname	days_no
David	Doe	2
Jane	Brown	1
Mary	Williams	2
Linda	Smith	6
Patricia	Brown	2
Jane	Garcia	3
William	Williams	5
David	Brown	3
Susan	Williams	2
William	Jones	7

- **Multi-row Subquery**

```
SELECT cust_id, fname, lname
FROM jc_customer
WHERE cust_id IN (SELECT cust_id FROM jc_service WHERE days_no > 5);
```

This query retrieves customer details for those who have rentals with a duration of more than 5 days, identifying customers who frequently rent vehicles for an extended period.

cust_id	fname	lname
4	Linda	Smith
10	William	Jones
16	Benjamin	Martinez

- **Correlated Subquery**

```
SELECT cust_id, fname, lname,
  (SELECT COUNT(*)
   FROM jc_service
   WHERE jc_service.cust_id = jc_customer.cust_id) AS rental_count
FROM jc_customer;
```

This query retrieves customer details along with the count of rentals they have made, helping to understand the rental activity of each customer.

cust_id	fname	lname	rental_count
1	David	Doe	1
2	Jane	Brown	1
3	Mary	Williams	1
4	Linda	Smith	1
5	Patricia	Brown	1
6	Jane	Garcia	1
7	William	Williams	1
8	David	Brown	1
9	Susan	Williams	1
10	William	Jones	1

- **SET Operator Query**

```
SELECT cust_id, fname, lname FROM jc_customer
WHERE cust_id IN (SELECT cust_id FROM jc_service WHERE days_no > 4)
INTERSECT
SELECT cust_id, fname, lname FROM jc_customer WHERE cust_type = 'C' ORDER BY cust_id;
```

This query retrieves corporate customer details for those who have rentals with a duration of more than 4 days, which could help identify companies that frequently rent vehicles for an extended period and make it easier to establish a long-term collaborative relationship.

cust_id	fname	lname
4	Linda	Smith
7	William	Williams
10	William	Jones

- **Query with in-line View or WITH Clause**

```
WITH nearest_customers AS (
    SELECT cust_id, fname, lname FROM jc_customer WHERE state IN ('NY', 'NJ')
)
SELECT * FROM nearest_customers;
```

This query retrieves details of customers residing in specific areas and it could be used to identify and analyze the nearest customers based on their geographic location.

cust_id	fname	lname
2	Jane	Brown
4	Linda	Smith
13	Olivia	Garcia
14	Ethan	Anderson

- **TOP-N/BOTTOM-N Query**

```
SELECT *FROM jc_service ORDER BY days_no DESC LIMIT 5;
```

This query retrieves the top 5 longest rental durations, identifying the top-performing rentals based on duration.

service_id	days_no	start_odo	end_odo	pickup_loc	dropoff_loc	discount_id	cust_id
10	7	28000	29000	4	1	10	10
16	7	18000	19000	10	1	16	16
4	6	16000	16700	7	8	4	4
7	5	22000	22700	3	5	7	7
12	5	50000	52000	2	8	12	12