Car rental report

Project purpose:

Our project is about car rental. The idea of selecting this subject comes from several weeks ago while we are planning a trip to New York City. We had browsed through a car rental website trying to rent a perfect car. After we finally decided which car we want to rent, the company told me that I must return the car back to the same place after use. I told him that I'm not going to use the car while I'm in New York City since the traffic problem there is crazy. So, I'm planning to return the car after I got to New York City and rent it later when I'm planning to drive back to Syracuse. The company told me that can't be done because they don't have places for me to return the car in New York City and their policy is every user must return the car to where they rent it. I canceled my car rental order from this company right away since I'm not going to be driving in New York City. If I need to rent a car for 7 days but just drive it for 2 days, that sounds like a bad trade for me. After that, a business idea pops out. What if I can build a car rental company that allows my user to rent a car and return the car at different locations, that will be really attractive for certain customers like me. That's basically what our database is about.

Conceptual data model:

The left chart we can see our 7 entities clearly and their attributes on the right. We had also put the description for each attribute for better understanding. The right chart has shown the relationship between each entity. Base on this conceptual data model, we can build our E-R diagram and logical data model much easier.

Entities and Attributes

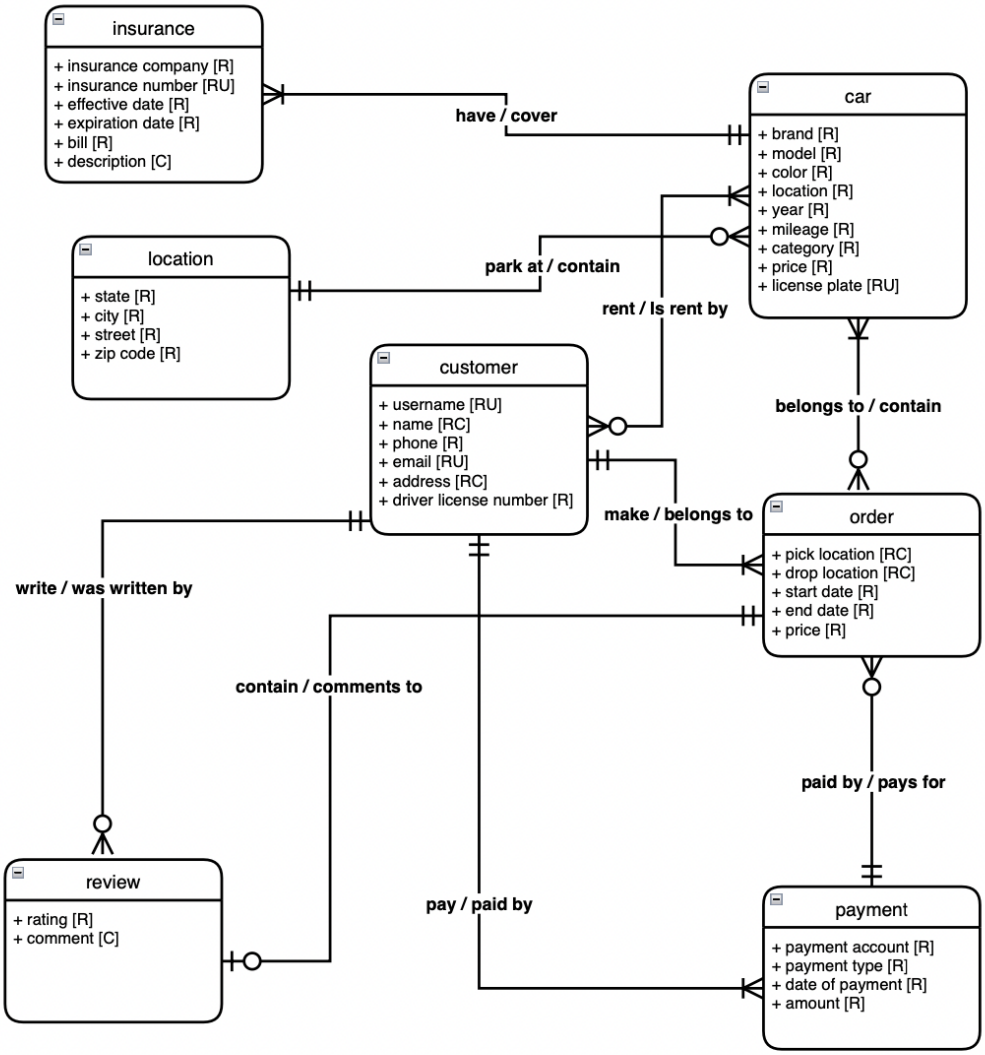
Entity	Attribute	Props	Descripion
customer	username	RU	customer's username
-	name	RC	customer's name
-	phone	M	customer's phone
-	email	RU	customer's email
-	address	RC	customer's address
	driving license number	R	customer's driving license number
Car	brand	R	car's brand
-	model	R	car's model
	color	R	car's color
-	location	R	car's current location
-	year	R	car's production year
-	mileage	R	the mileage of the car
-	category	R	the category of car
	price	R	the price of car per day
	license plates	RU	Car's license plate
Order	pick_location	RC	the city of pick-up the car
	drop_location	RC	the city of drop the car
	start_date	R	the start date of rental
	end_date	R	the end date of rental
	price	R	the price of order
insurance	insurance company	R	insurance company
	insurance number	RU	insurance id number
	effective date	R	the start date of insurance
	expiration date	R	the expiration date of insurance
	bill	R	the price of insurance
	description	C	description about insurance
payment	payment account	R	the account of payment
	payment type	R	the payment type
	Date of payment	R	payment date
	Amount	R	total payment
review	rating	R	the rating from customers(0-10)
	comment	C	the comments from customers
location	state	R	the state of parking spot
	city	R	the city of parking spot
	street	R	the address of parking spot
	zip code	R	the zip code of parking spot

Relationships

Relationship	Entity	Rule	Min	Max	Entity
customer-car	**customer**	rent	1	M	**car**
	car	is rent by	0	M	**customer**
	-				-
customer-order	**customer**	make	1	M	**order**
	order	belong to	1	1	**customer**
	-				-
customer-payment	**customer**	pay	1	M	**payment**
	payment	paid by	1	1	**customer**
	-				-
customer-review	**customer**	wirte	0	M	**review**
	review	was written by	1	1	**customer**
					-
car-order	**car**	belongs to	0	M	**order**
	order	conatin	1	M	**car**
					-
car-insurance	car	have	1	M	**insurance**
	insurance	cover	1	1	**car**
					-
order-payment	order	paid by	1	1	**payment**
	payment	pays for	0	M	**order**
					-
order-review	order	contain	0	1	**review**
	review	comments to	1	1	**order**
					-
car- location	car	park at	1	1	**location**
	location	contain	0	M	**car**
					-
					-
					-
					-
					-
					-

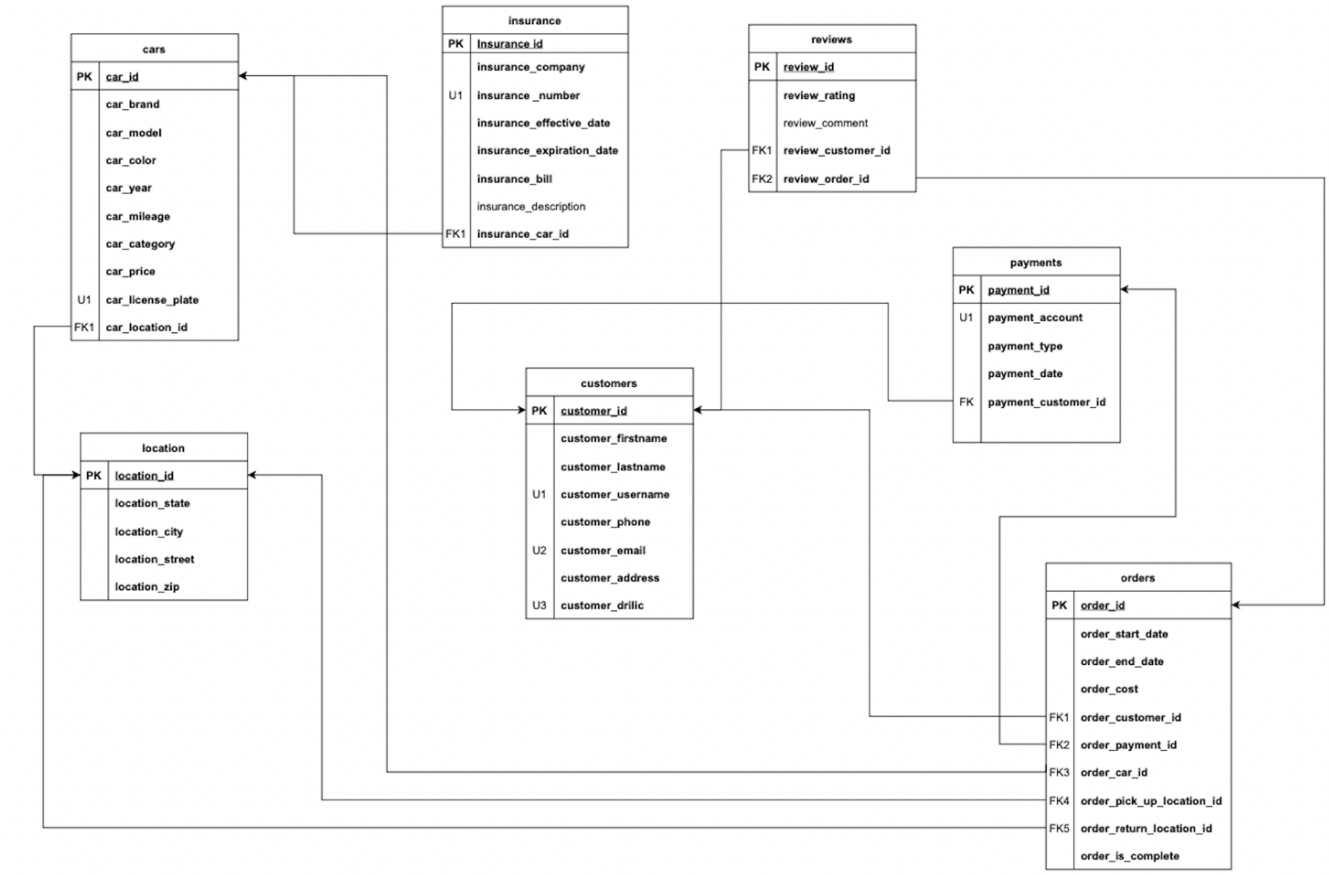
Entity relationship diagram:

The E-R diagram is a type of flowchart that shows how entities related to each other within a system. We can tell the basic idea about the whole database from just looking at this diagram. The 7 entities are mark up and their attributes are included. The line between different boxes shows the relationship between different entities.



Logical data model:

A logical data model is a type of data model that describes data elements in detail and is used to develop visual understandings of data entities, attributes, keys, and relationships. So in this model, we had mark out the Primary Key for each entity. We had also marked all the foreign key and the unique key if it's needed. After the logical data model is done, we are now official ready for some coding!



SQL up/down script:

```
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_car_location_id')
alter table cars  drop CONSTRAINT fk_car_location_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_order_customer_id')
alter table orders drop CONSTRAINT fk_order_customer_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_order_payment_id ')
alter table orders drop CONSTRAINT fk_order_payment_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_order_car_id')
alter table orders drop CONSTRAINT fk_order_car_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_order_pickup_location_id')
alter table orders drop CONSTRAINT fk_order_pickup_location_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_order_return_location_id')
alter table orders drop CONSTRAINT fk_order_return_location_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_insurance_car_id')
alter table insurances drop CONSTRAINT fk_insurance_car_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_payment_customer_id')
alter table payments drop CONSTRAINT fk_payment_customer_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_review_customer_id')
alter table reviews drop CONSTRAINT fk_review_customer_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where CONSTRAINT_NAME='fk_review_order_id')
alter table reviews drop CONSTRAINT fk_review_order_id


drop table if exists customers
GO
drop table if exists cars
```

```
GO
drop table if exists locations
Go
drop table if exists orders
GO
drop table if exists insurances
GO
drop table if EXISTS reviews
GO
drop table if EXISTS payments
GO
```

Create table:

```
create table customers(
customer_id int identity not null,
customer_firstname varchar(20) not null,
customer_lastname varchar(20) not null,
customer_username varchar(30) not null,
customer_phone varchar(10) not null,
customer_email varchar(50) NOT null,
customer_address varchar(50) not null,
customer_drilic varchar(20) not null,
constraint pk_customers_customer_id PRIMARY KEY(customer_id),
CONSTRAINT u_customer_email UNIQUE(customer_email),
CONSTRAINT u_customer_drilic UNIQUE(customer_drilic),
constraint u_customer_username UNIQUE(customer_username)
)
GO

create table locations(
location_id int identity not null,
location_state VARCHAR(50) not null,
location_city VARCHAR(50) not null,
location_street VARCHAR(50) not null,
location_zip int not null,
constraint pk_locations_location_id PRIMARY KEY(location_id)
)
GO

create table cars(
car_id int identity not null,
car_brand varchar(50) not null,
car_model varchar(50) not null,
car_color varchar(50) not null,
```

```
car_year varchar(50) not null,
car_mileage int not null,
car_category VARCHAR(50) not null,
car_price money not null,
car_license_plate VARCHAR(50) not null,
car_location_id INT not null
constraint pk_cars_car_id PRIMARY KEY(car_id),
CONSTRAINT u_car_license_plate UNIQUE(car_license_plate)
)
alter table cars add
CONSTRAINT fk_car_location_id FOREIGN key(car_location_id) REFERENCES
locations(location_id)
GO

create table payments(
payment_id int identity not null,
payment_account varchar(50) not null,
payment_type varchar(50) not null,
payment_date datetime not null,
payment_customer_id int not NULL
constraint pk_payment_id PRIMARY KEY(payment_id),
constraint u_payment_account unique(payment_account)
)
alter table payments add
CONSTRAINT fk_payment_customer_id FOREIGN key(payment_customer_id) REFERENCES
customers(customer_id)
GO

create table orders(
order_id int identity not null,
order_start_date datetime not null,
order_end_date datetime not null,
order_cost money null,
order_is_complete int null,
order_customer_id int null,
order_payment_id int not null,
order_car_id int not null,
order_pickup_location_id int not null,
order_return_location_id int not null
constraint pk_orders_order_id PRIMARY KEY(order_id),
constraint ck_orders_start_end_date check(order_start_date < order_end_date)
)
alter table orders add
CONSTRAINT fk_order_customer_id FOREIGN key(order_customer_id) REFERENCES
customers(customer_id),
```

```
CONSTRAINT fk_order_payment_id FOREIGN key(order_payment_id) REFERENCES
payments(payment_id),
CONSTRAINT fk_order_car_id FOREIGN key(order_car_id) REFERENCES cars(car_id),
CONSTRAINT fk_order_pickup_location_id FOREIGN key(order_pickup_location_id)
REFERENCES locations(location_id),
CONSTRAINT fk_order_return_location_id FOREIGN key(order_return_location_id)
REFERENCES locations(location_id)
GO

create table insurances(
insurance_id int identity not null,
insurance_company varchar(50) not null,
insurance_number varchar(50) not null,
insurance_effective_date datetime not null,
insurance_expiration_date datetime not null,
insurance_bill money null,
insurance_description varchar(150) null,
insurance_car_id int not null,
constraint pk_insurances_insurance_id PRIMARY KEY(insurance_id),
CONSTRAINT u_insurance_number UNIQUE(insurance_number)
)
alter table insurances add
CONSTRAINT fk_insurance_car_id FOREIGN key(insurance_car_id) REFERENCES
cars(car_id)
GO

create table reviews(
review_id int identity not null,
review_rating int not null,
review_comment varchar(500) null,
review_customer_id int not null,
review_order_id int not null,
constraint pk_reviews_review_id PRIMARY KEY(review_id),
CONSTRAINT u_review_order_id UNIQUE(review_order_id),
constraint ck_review_rating check(review_rating>=0 and review_rating<=10)
)
alter table reviews add
CONSTRAINT fk_review_customer_id FOREIGN key(review_customer_id) REFERENCES
customers(customer_id),
CONSTRAINT fk_review_order_id FOREIGN key(review_order_id) REFERENCES
orders(order_id)
GO
```

Insert data:

```
insert into customers(customer_firstname, customer_lastname, customer_username,
customer_phone, customer_email, customer_address, customer_drilic) VALUES
('Yaping','Wang','ashley74747','6803560103','ywa380@syr.edu','150 Hnery street, Syracuse,
NY','475647564'),
('Zhiyu','Lin','jerry27364','3156409303','zlinhh19@syr.edu','919 e genesse street, Syracuse,
NY','746754983'),
('Kevin','Hsu','kk808','3155678990','zkuang34@syr.edu','300 University Avenue, Syracuse,
NY','122156780'),
('Hans','Duan','hd7233','2608987334','hans25@gmail.com','234 marry street, Brooklyn,
NY','383847365'),
('Ben','Wang','ben0810','3428847564','ben0810@gmail.com','345 real street, Brooklyn,
NY','399876283'),
('Eric','Tien','eric0000','4532887364','ericyaa@gmail.com','333 wass street, Brooklyn,
NY','485938475'),
('David','Lin','david888','9368734637','david666@gmail.com','666 hey street, Brooklyn,
NY','334564578')
GO
select * from customers
GO

insert into locations(location_state,location_city,location_street,location_zip) VALUES
('NY','Syracuse','150 henry street','13210'),
('NY','Brooklyn','374 tree street','14203'),
('NY','Syracuse','767 layy street','14545'),
('NY','Queens','232 hype street','13234'),
('LA','Irvine','888 jump street','17878'),
('LA','Irvine','676 papa street','17465'),
('NY','Brooklyn','554 wee street','16454'),
('NY','Syracuse','787 mama street','13667'),
('NY','Queens','165 what street','13255'),
('LA','Irvine','165 lowkey street','17666')
GO
select * from locations
GO

insert into
cars(car_brand,car_model,car_color,car_year,car_mileage,car_category,car_price,car_license
_plate,car_location_id) VALUES
('Toyota','corolla','red','2015','5000','SUV','100','2345SU','1'),
('Toyota','sienna','black','2018','15030','standard','80','8766MP','1'),
('Toyota','corolla','black','2019','12635','compact','110','7763AA','3'),
('Toyota','camry','white','2015','8373','standard','110','7766EE','2'),
('Toyota','camry','grey','2016','8767','standard','100','7657AH','2'),
('Audi','A5','grey','2017','10787','coupe','180','1651BA','1'),
```

```
('Audi','A5','black','2019','10219','coupe','190','6652CH','1'),
('Ford','mustang','black','2017','13343','coupe','140','7767MM','4'),
('Ford','mustang','white','2019','9890','coupe','160','5556AM','5'),
('Kia','rio','white','2019','9890','economy','95','6676CM','5'),
('Kia','rio','white','2018','13420','standard','80','5554YU','2'),
('Honda','odyssey','black','2018','14532','minivan','150','7788UU','3'),
('Honda','odyssey','black','2018','12230','minivan','165','6363HE','2'),
('Honda','odyssey','white','2016','16670','minivan','130','6119QP','2'),
('Honda','odyssey','white','2016','14930','minivan','140','9079WW','4')
GO
select * from cars
GO

insert into payments(payment_account,payment_type,payment_date,payment_customer_id)
VALUES
('4047890767899980','credit card','2022/01/06','5'),
('4510747387512563','credit card','2022/03/11','2'),
('6184766215561894','credit card','2022/03/09','3'),
('3806412935594473','credit card','2022/03/03','4'),
('8536210303723188','debit card','2022/03/22','5'),
('4500846096740233','debit card','2022/04/01','6'),
('4940633840876030','debit card','2022/04/13','7'),
('5858064301120099','debit card','2022/04/15','1'),
('9311467986236172','debit card','2022/02/18','1'),
('6638024357201131','debit card','2022/03/25','3')
GO
select * from payments
GO

insert into orders(order_start_date,
order_end_date,order_cost,order_is_complete,order_customer_id,order_payment_id,order_car
_id,order_pickup_location_id,order_return_location_id) VALUES
('2022/01/06','2022/01/09','325','1','5','1','1','1','3'),
('2022/03/11','2022/03/13','160','1','2','2','2','2','3'),
('2022/03/09','2022/03/10','110','1','3','3','3','5','4'),
('2022/03/03','2022/03/05','220','1','4','4','4','3','3'),
('2022/03/22','2022/03/24','200','1','5','5','5','2','1'),
('2022/04/01','2022/04/03','360','1','6','6','6','3','5'),
('2022/04/13','2022/04/14','190','1','7','7','7','5','4'),
('2022/04/15','2022/04/17','280','1','1','8','8','2','2'),
('2022/02/18','2022/02/20','320','1','3','9','9','4','1'),
('2022/03/25','2022/03/27','190','1','1','10','10','3','5')
GO
```

```
select * from orders
GO

insert into
insurances(insurance_company,insurance_number,insurance_effective_date,insurance_expirati
on_date,insurance_bill,insurance_description,insurance_car_id) VALUES
('Allstate', '12345678910','2021/9/1','2022/9/30','790','null','1'),
('Allstate', '3808115257','2021/9/1','2022/9/30','990','null','2'),
('Allstate', '3563727780','2021/9/1','2022/9/30','990','null','3'),
('Allstate', '3436939330','2021/9/1','2022/9/30','990','null','4'),
('Allstate', '8155412201','2021/9/1','2022/9/30','990','null','5'),
('Allstate', '7669697680','2021/9/1','2022/9/30','2000','null','6'),
('Allstate', '1351660973','2021/9/1','2022/9/30','2000','null','7'),
('Allstate', '0880998988','2021/9/1','2022/9/30','1290','null','8'),
('Allstate', '9917504925','2021/9/1','2022/9/30','1290','null','9'),
('Allstate', '1677189927','2021/9/1','2022/9/30','1290','null','10'),
('Allstate', '1780336094','2021/9/1','2022/9/30','1290','null','11'),
('Allstate', '7571425976','2021/9/1','2022/9/30','1390','null','12'),
('Allstate', '3841513015','2021/9/1','2022/9/30','1390','null','13'),
('Allstate', '8516031393','2021/9/1','2022/9/30','1390','null','14'),
('Allstate', '3537629343','2021/9/1','2022/9/30','1390','null','15')
GO
select * from insurances
GO

insert into reviews(review_rating,review_comment,review_customer_id,review_order_id)
VALUES
('10','very clean,pretty good experience','5','1'),
('9','clean, great car condition','2','2'),
('9','awesome smell on the car','3','3'),
('8','Everything is fine and clean','4','4'),
('10','Perfect car and services, can't ask for more','5','5'),
('8','Great car but the price are a bit high','6','6'),
('10','the car is pretty new and the services they provide are perfect','7','7'),
('9','Great car, good experience','1','8'),
('10','The car is really new, totally worth the price','7','9'),
('10','Great car, great people! Highly recommend!','7','10')
GO
select * from reviews
GO
```

User story 1:

I can sign in or join in the platform so I can rent a car through this application.

```
drop procedure if exists p_upsert_customer
GO
create procedure p_upsert_customer(
   @customer_firstname varchar(50),
   @customer_lastname varchar(50),
   @customer_username VARCHAR(50),
   @customer_phone varchar(50),
   @customer_email varchar(50),
   @customer_address varchar(50),
   @customer_drilic varchar(50)
)as
begin
   if exists(select*from customers
               where customer_email=@customer_email)
   begin update customers set customer_firstname=@customer_firstname,
                     customer_lastname=@customer_lastname,
                     customer_username=@customer_username,
                     customer_phone=@customer_phone,
                     customer_email=@customer_email,
                     customer_address=@customer_address,
                     customer_drilic=@customer_drilic
            where customer_email=@customer_email
   end
   else BEGIN
   declare @id int
   set @id = (select max(customer_id)+1 from customers)
   insert into customers(customer_firstname, customer_lastname, customer_username,
customer_phone, customer_email, customer_address, customer_drilic)
      values(@customer_firstname, @customer_lastname,@customer_username,
@customer_phone, @customer_email, @customer_address, @customer_drilic)
   END
END
GO

select * from customers
exec p_upsert_customer  @customer_firstname='joey', @customer_lastname='Chou',
@customer_username='joey778', @customer_email='joey01@syr.edu',
@customer_phone='3189135340' , @customer_address='301 University Avenue, Syracuse, NY',
@customer_drilic='896654379'
select * from customers
```

User story 2:

I can check the avilibility of the car and I can see how much I need to pay.

```
drop PROCEDURE if exists p_check_avail
go
create procedure [dbo].[p_check_avail](
@start datetime,@end datetime)
as begin
select distinct car_id from cars left join orders ON cars.car_id=orders.order_car_id
where orders.order_end_date > @start and orders.order_is_complete = 0
union
select distinct car_id from cars left join orders ON cars.car_id=orders.order_car_id
where orders.order_is_complete = 1
end

drop PROCEDURE if exists p_calcul
go
create procedure [dbo].[p_calcul](
@start datetime,@end datetime,@car_id int)
as begin
select car_price*(datediff(DAY,@start,@end)) from cars where car_id=@car_id
end
```

User story 3:

I can rate for the car I rented after I finished my rental and I can write the comments for the car.

```
drop procedure if exists p_ratings
GO
create procedure p_ratings(
  @order int,
  @rating int,
  @comment varchar(500)
) as BEGIN
begin TRY
 begin transaction
  insert into reviews(review_order_id, review_rating, review_comment) VALUES
  (@order, @rating, @comment)
commit
end TRY
begin CATCH
print 'ROLLBACK'
```

```
ROLLBACK
;
THROW
end catch
END
```

Manager story 1:

As a manager I can insert cars' information and update the car's detail.

```
drop PROCEDURE if exists p_.update_car
go
create procedure [dbo].[p_update_car](@car_id int,
@car_brand varchar(20),
@car_model varchar(20),
@car_color varchar(20),
@car_year varchar(20),
@car_mileage int,
@car_category varchar(20),
@car_price money,
@car_license_plate varchar(50),
@car_location_id int)
as begin
if ISNULL(@car_brand,'00') <>'00'
begin
update cars set car_brand = @car_brand where car_id = @car_id
END
if ISNULL(@car_model,'00') <>'00'
begin
update cars set car_model = @car_model where car_id = @car_id
END
if ISNULL(@car_color,'00') <>'00'
begin
update cars set car_color = @car_color where car_id = @car_id
END
if ISNULL(@car_year,'00') <>'00'
begin
update cars set car_year = @car_year where car_id = @car_id
END
if ISNULL(@car_mileage,'00') <>'00'
begin
update cars set car_mileage = @car_mileage where car_id = @car_id
END
if ISNULL(@car_category,'00') <>'00'
begin
```

```
update cars set car_category = @car_category where car_id = @car_id
END
if ISNULL(@car_price,'00') <>'00'
begin
update cars set car_price = @car_price where car_id = @car_id
END
if ISNULL(@car_license_plate,'00') <>'00'
begin
update cars set car_license_plate = @car_license_plate where car_id = @car_id
END
if ISNULL(@car_location_id,'00') <>'00'
begin
update cars set car_location_id = @car_location_id where car_id = @car_id
END

end

drop PROCEDURE if exists p_insert_car
go
create procedure [dbo].[p_insert_car](
@car_brand varchar(20),
@car_model varchar(20),
@car_color varchar(20),
@car_year varchar(20),
@car_mileage int,
@car_category varchar(20),
@car_price money,
@car_license_plate varchar(50),
@car_location_id int)
as begin
insert
cars(car_brand,car_model,car_color,car_year,car_mileage,car_category,car_price,car_license
_plate,car_location_id) values (@car_brand,
@car_model,@car_color,@car_year,@car_mileage,@car_category,@car_price,@car_license
_plate,@car_location_id)

End
```

Manager story 2:

As a manager, I can review the insurance history to make sure the car has effective insurance.

```
drop function if exists f_car_insurance
GO
create function f_car_insurance(@car_license_plate varchar(50))
```

```
returns TABLE AS
return(
   select * from cars
      join insurances on insurance_car_id=car_id
       WHERE car_license_plate=@car_license_plate)
GO

select * FROM f_car_insurance('2345SU')
```

Manager story 3:

As a manager, I can update the status of car when the order complete.

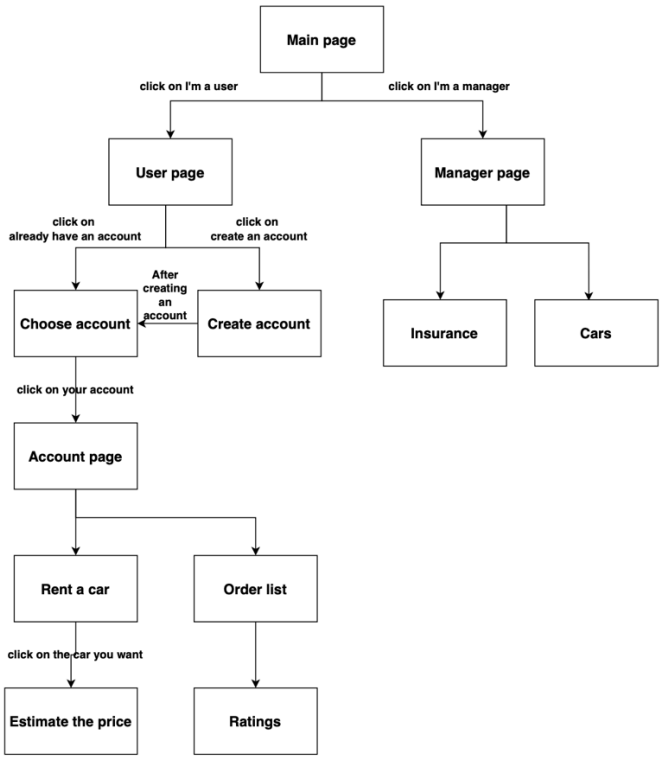
```
drop PROCEDURE if exists p_update_order_state
go
create procedure [dbo].[p_update_order_state](@order int)
as begin
   update orders set order_is_complete = 1 where order_id=@order
end
```

Power apps layout:

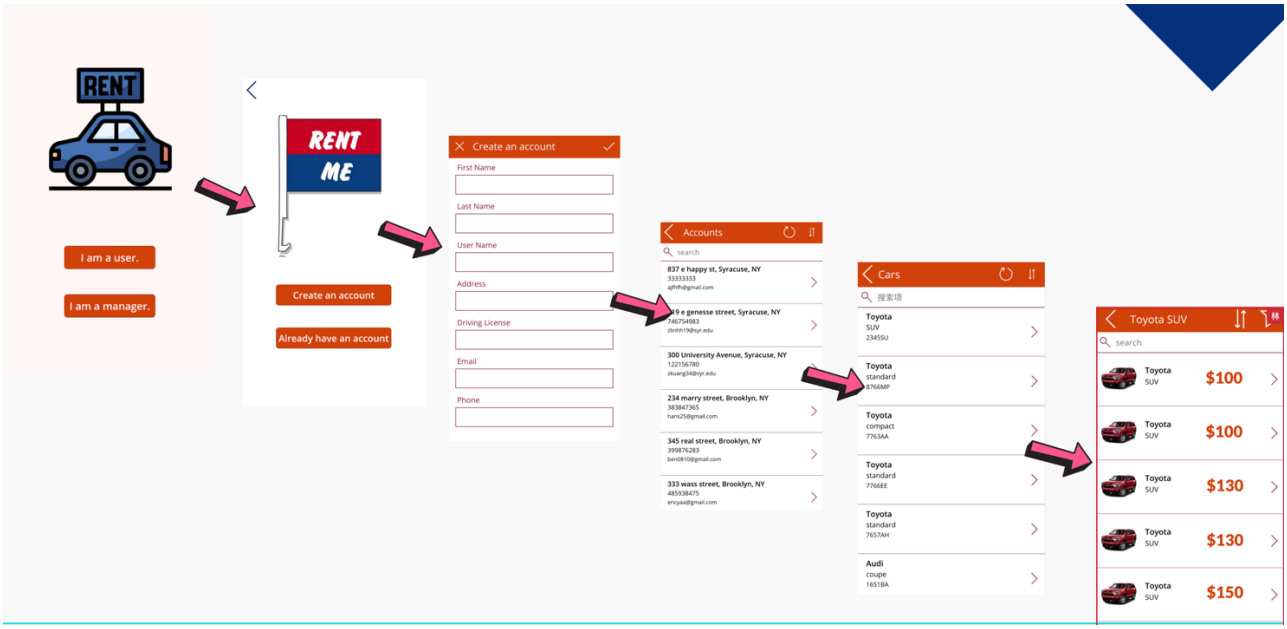
Our power apps is designed for our 2 stakeholders, users and managers. There will be two buttom to select on the main page, after you select your identity our app will direct you to different sections.

As a user, you will have to create an account to rent a car. After the user got an account and find their account on the account page they can start the car renting process. They can easily use the filter buttom on the top to find the car they want and the system will calculate the price for them. Based on different car models and the amount of days they want to rent the car from us, the final price will be different.

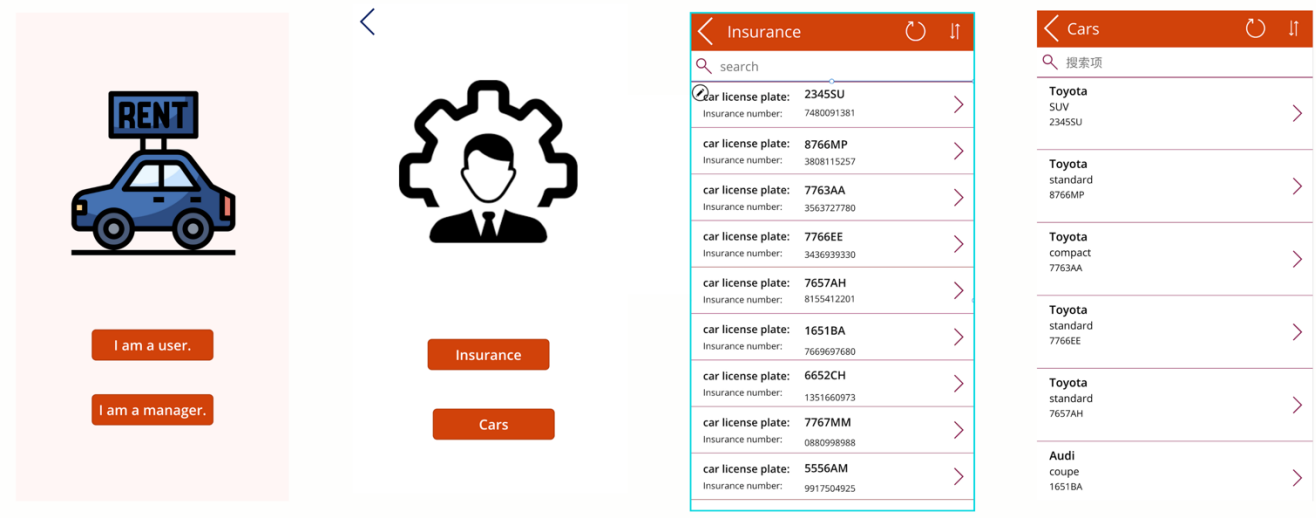
As a manager there will be only 2 bottom for you to select, insurance and cars. If you click on the insurance bottom, all the insurance detail for our cars will be shown on the interface. If you click on the cars bottom, all the cars that's available will be shown on the interface for a easy way to manage them.



User interface in our app:



Manager interface in our app:



Team log:

Task & Proportion	ZhiYu Lin	Yaping Wang	Completed time
Conceptual data model	60%	40%	April 16th
Logical data model	40%	60%	April 16th
Internal data model	50%	50%	April 20th
External data model	50%	50%	April 22th
Power apps	50%	50%	April 25th
Power point	40%	60%	April 26th
Report	60%	40%	April 30th