Project team 8 – Car Rental System

Team Members:
Alisha Gujarathi
Hima Bindu Sigili
Nipun Gupta
Reddy Vishnuvardhan Reddy

PROJECT PROPOSAL

Content, Scope and Objectives

Content:

In order to facilitate users to rent a car as and when required we would like to design a car rental system. It is more feasible than owning a car and maintaining it. Car rental system serves people who don't own a car, who are out of town or owners of damaged cars who are awaiting repair or insurance compensation.

Scope:

1. Providing car catalog to the Users so that they can choose the best option based on their concern.
2. Admin can manage the catalog by adding or removing the cars based on their availability and allows only authorized Users (Driver License) to rent a vehicle
3. Admin must make sure that the rented cars must have valid insurance.
4. Admin must allow user to provide feedback at the end of every ride.

Objective:

1. To reduce the effort of booking a car in a conventional procedure.
2. To ease the search process of a customer who is in need of a car.
3. To provide services to the customers in order to achieve the best customer satisfaction.

Sprint 0

PROJECT ENVIRONMENT

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. Frequently used operations of phpMyAdmin include managing databases,

tables, columns, relations, indexes, users, permissions etc. which can be performed via the user interface, while having the ability to directly execute any SQL statement.
phpMyAdmin Version information: 4.0.10deb1. Database Server details are as follows:

- Server: 127.0.0.1 via TCP/IP
- Server type: MySQL
- Server version: 5.5.57-0ubuntu0.14.04.1 - (Ubuntu)
- Protocol version: 10
- Database client version: libmysql - mysqlnd 5.0.11-dev

HIGH LEVEL REQUIREMENTS

Initial user roles

User Roles	**Description**
Customer	Customer rents a car from the Car Rental system
Admin	Admin manages the car rental system
Guest	Browses the available list of cars from the Car Rental system

Initial user story descriptions

Story ID	**Story description**
US1	As a Guest, I want to browse the list of available cars in the Car Rental system
US2	As a Guest, I want to sign up in Car Rental system so that I can rent a car
US3	As a Customer, I want to rent a car by logging in the Rental system
US4	As an Admin, I want to add a new car to the list of available cars
US5	As an Admin, I want to change the status of rented cars
US6	As a Customer, I want to cancel the booked car in Car Rental system

Note: Customer here means a person who has an account in Car Rental System.

HIGH LEVEL CONCEPTUAL DESIGN

Entities:

Admin
Car
Customer

Relationships:

Customer rents a Car
Customer returns a Car
Admin adds a Car
Admin removes a Car

Sprint 1

REQUIREMENTS

Story ID	Story description
US4	As an Admin, I want to add a new car to the Car catalog
US1	As a Guest, I want to browse the list of available cars in the Car Rental system
US2	As a Guest, I want to sign up in Car Rental system so that I can rent a car
US3	As a Customer, I want to rent a car by logging in the Rental system
US5	As an Admin, I want to change the status of rented cars

Story refinement, with notes:

1. As an admin, I want to add new cars to the catalog of available cars.

   Notes:
   - User must be logged in as the admin to use this feature.
   - Catalog contains car make, model, manufacturing year, car color, seating capacity and car mileage
   - Only one car is added at a time.
   - There's no separate entity called Catalog. Catalog is simply a collection of cars.

Additional Note:Considered login feature so as to store the username and password of admin in the database

   Updated stories:
   a. As an admin, I want to log in to the system so that I can access features specific to my role.
   b. As an admin, I want to log out to the system.
   c. As an admin, I want to add a car to the Car Rental System's catalog with status as Available.

2. As an admin, I want to change status of cars in the catalog of cars in Car Rental System.

Updated stories:

a. As an admin, I want to change the status of the car as Not available as and when the customer rents the car.
b. As an admin, I want to change the status of the car as Available if returned by the customer.

3. As a Customer, I want to sign in into the Car Rental System.

Updated stories:

a. As a customer, I want to login into the car rental system to rent the car
b. As a customer, I want to rent a car.

Note:

- Considered login feature so as to store the username and password of the customer in the database and creating an Account (membership) for the particular customer.
- A guest becomes a customer after signing up.

Stories to be considered:

1. As an Admin, I want to log in to the system so that I can access features specific to my role.
2. As an Admin, I want to log out to the system.
3. As an Admin, I want to add a car to the store’s catalog.
4. As an Admin, I want to change the status of the cars in the catalog.
5. As a Customer, I want to log in to the Rental system so that I can rent a car of my choice.
6. As a Customer, I want to rent the car.

CONCEPTUAL DESIGN

Entity: **Admin**
Attributes:
- username
- password

Entity: **Car**
Attributes:
- car_id
- car_details [composite]
    - model
    - make
    - color
    - manufacturing_year
    - mileage
    - Seating_capacity
- status
- car_type
- price_per_day

Entity: **Customer**
Attributes:
- license_number
- email_id
- mobile _number
- name [composite]
    - last_name
    - middle_name
    - first_name
- Date_of_birth
- age
- address [composite, multi-valued]
    - address_line_1
    - address_line_2
    - city
    - state

Zip_code

Entity: **Account**
Attributes:
- <u>username</u>
- password

Entity: **CardDetails** (Weak Entity)
Attributes:
- <u>card_number</u>
- name_on_card
- date_of_expiration
- billing_address

Relationship: **Customer** rents a **Car**
Cardinality: Many to Many
Participation:
- Customer has partial participation
- Car has total participation

Relationship: **Customer** has **Account**
Cardinality: One to One
Participation:
- Customer has total participation
- Account has total participation

Relationship: **Account** has **CardDetails**
Cardinality: One to Many
Participation:
- Account has partial participation
- CardDetails has total participation

Relationship: **Admin** adds a **Car**
Cardinality: One to Many
Participation:
- Admin has partial participation
- Car has total participation

LOGICAL DESIGN

Table: **Admin**
Columns:
username
Password

Justification: username is unique for everyone

Table: **Customer**
Columns:
license_number
email_id
mobile _number
first_name
middle_name
last_name
date_of_birth
Residential_address[foreign key; references **address_id** of **Address**]

Justification: license_number is unique for everyone

Table: **Car**
Columns:
car_id
model
make
color
manufacturing_year
mileage
seating_capacity
status
car_type
price_per_day
username[foreign key; references **username** of **Admin**]

license_number[foreign key; references **license_number** of **Customer**]

Justification: car_id of each car will be unique.

Note: car_id is number plate of the car.

Table: **Account**
Columns:
<u>username</u>
password
license_number[foreign key; references license_number of Customer]

Justification: username is unique for everyone

Table: **CardDetails**
Columns:
<u>username</u>
<u>card_number</u>
name_on_card
date_of_expiration
billing_address[foreign key; references **address_id** of **Address**]

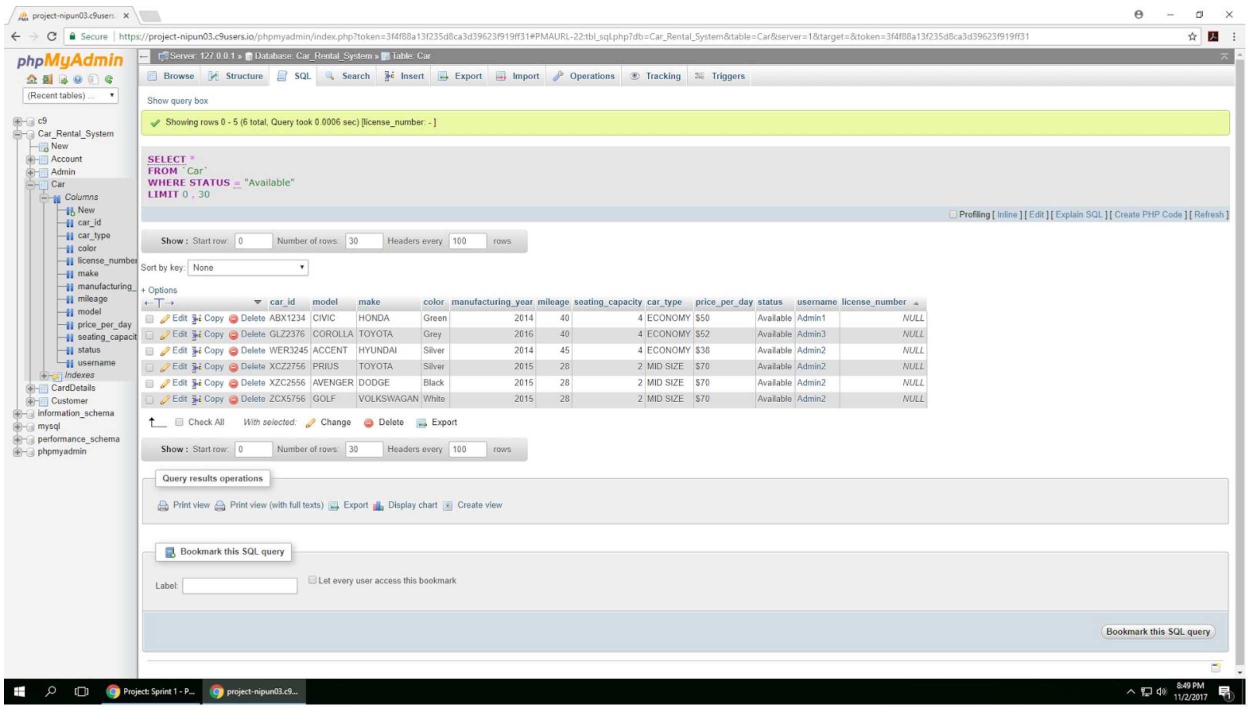
Justification: <u>username</u> and the descriptive attribute(<u>card_number</u>) *is the primary because CardDetails is the weak entity of Account so the primary key of account is a part of the primary key of CardDetails*

Table: **Address**
Columns:
<u>address_id</u>
address_line_1
address_line_2
city
state
Zip_code
Justification: <u>address_id</u> is uniquely assigned by the system

Writing key SQL queries:

1. Query for the details of all the available cars:

SELECT * FROM `Car` WHERE status="Available"



Server: 127.0.0.1 » Database: Car_Rental_System » Table: Car

Showing rows 0 - 5 (6 total, Query took 0.0006 sec) [license_number: -]

```
SELECT *
FROM `Car`
WHERE STATUS = "Available"
LIMIT 0 , 30
```

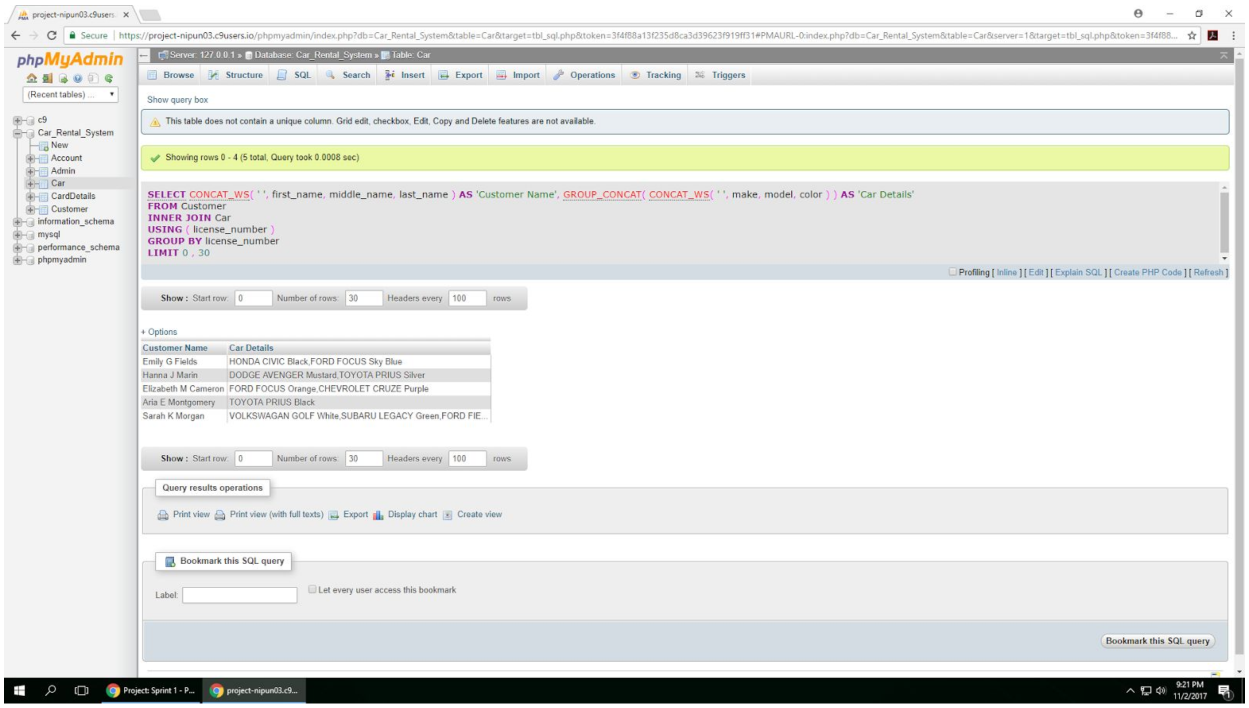
Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

car_id	model	make	color	manufacturing_year	mileage	seating_capacity	car_type	price_per_day	status	username	license_number
ABX1234	CIVIC	HONDA	Green	2014	40	4	ECONOMY	$50	Available	Admin1	NULL
GLZ2376	COROLLA	TOYOTA	Grey	2016	40	4	ECONOMY	$52	Available	Admin3	NULL
WER3245	ACCENT	HYUNDAI	Silver	2014	45	4	ECONOMY	$38	Available	Admin2	NULL
XCZ2756	PRIUS	TOYOTA	Silver	2015	28	2	MID SIZE	$70	Available	Admin2	NULL
XZC2556	AVENGER	DODGE	Black	2015	28	2	MID SIZE	$70	Available	Admin2	NULL
ZCX5756	GOLF	VOLKSWAGAN	White	2015	28	2	MID SIZE	$70	Available	Admin2	NULL

2. Query for showing all the cars(comma separated) rented by the customers:

SELECT CONCAT_WS(' ' , first_name , middle_name, last_name) AS 'Customer Name', group_concat(concat_ws(' ',make,model,color)) as 'Car Details'
FROM
Customer
INNER JOIN Car using (license_number)
group by license_number;



This table does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

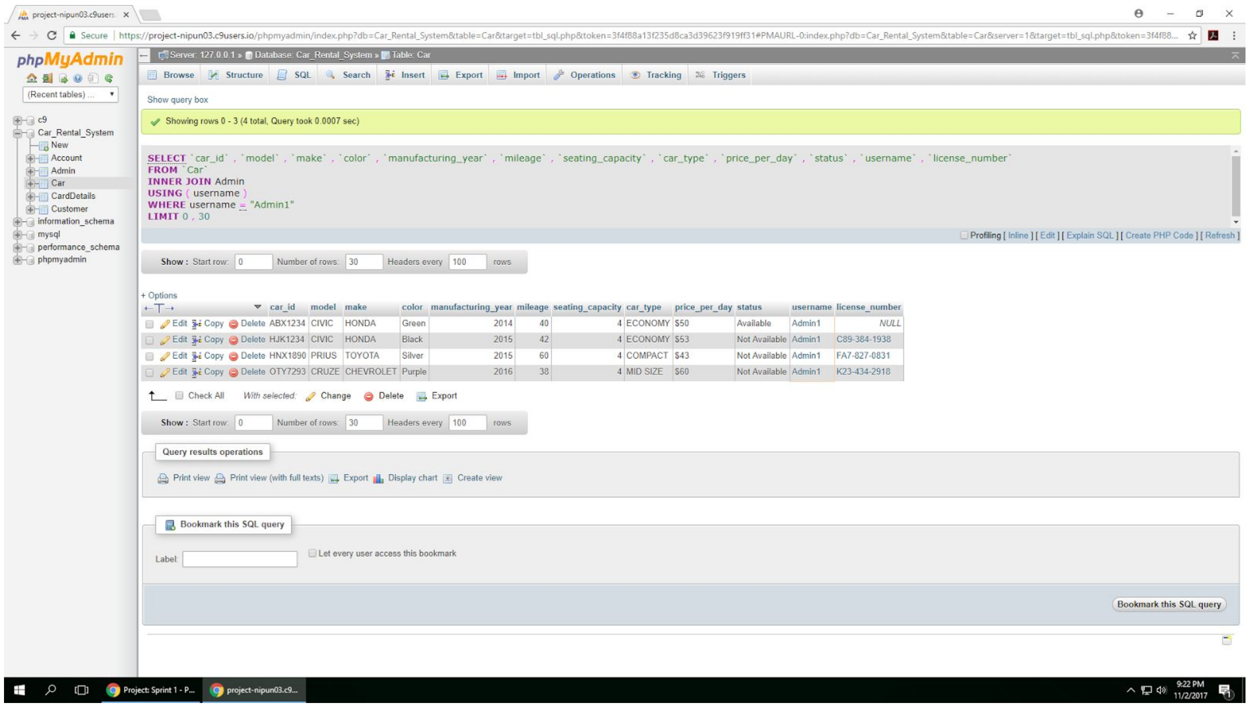
Showing rows 0 - 4 (5 total, Query took 0.0008 sec)

```
SELECT CONCAT_WS( ' ', first_name, middle_name, last_name ) AS 'Customer Name', GROUP_CONCAT( CONCAT_WS( ' ', make, model, color ) ) AS 'Car Details'
FROM Customer
INNER JOIN Car
USING ( license_number )
GROUP BY license_number
LIMIT 0 , 30
```

Customer Name	Car Details
Emily G Fields	HONDA CIVIC Black,FORD FOCUS Sky Blue
Hanna J Marin	DODGE AVENGER Mustard,TOYOTA PRIUS Silver
Elizabeth M Cameron	FORD FOCUS Orange,CHEVROLET CRUZE Purple
Aria E Montgomery	TOYOTA PRIUS Black
Sarah K Morgan	VOLKSWAGAN GOLF White,SUBARU LEGACY Green,FORD FIE...

3. Query for showing details of all cars that are added by Admin1:

SELECT \`car_id\`, \`model\`, \`make\`, \`color\`, \`manufacturing_year\`, \`mileage\`, \`seating_capacity\`, \`car_type\`, \`price_per_day\`, \`status\`, \`username\`, \`license_number\` FROM \`Car\` INNER JOIN Admin using (username) WHERE username="Admin1";



Showing rows 0 - 3 (4 total, Query took 0.0007 sec)

```
SELECT `car_id` , `model` , `make` , `color` , `manufacturing_year` , `mileage` , `seating_capacity` , `car_type` , `price_per_day` , `status` , `username` , `license_number`
FROM `Car`
INNER JOIN Admin
USING ( username )
WHERE username = "Admin1"
LIMIT 0 , 30
```

car_id	model	make	color	manufacturing_year	mileage	seating_capacity	car_type	price_per_day	status	username	license_number
ABX1234	CIVIC	HONDA	Green	2014	40	4	ECONOMY	$50	Available	Admin1	NULL
HJK1234	CIVIC	HONDA	Black	2015	42	4	ECONOMY	$53	Not Available	Admin1	C89-384-1938
HNX1890	PRIUS	TOYOTA	Silver	2015	60	4	COMPACT	$43	Not Available	Admin1	FA7-827-0831
OTY7293	CRUZE	CHEVROLET	Purple	2016	38	4	MID SIZE	$60	Not Available	Admin1	K23-434-2918

Sprint 2

REQUIREMENTS

Story ID	Story description
US4	As an Admin, I want to add a new car to the Car catalog
US1	As a Guest, I want to browse the list of available cars in the Car Rental system
US2	As a Guest, I want to sign up in Car Rental system so that I can rent a car
US3	As a Customer, I want to reserve a car by logging in the Rental system
US7	As a Customer, I want to choose my pickup and drop-off location
US6	As a Customer, I want to cancel the booked car in Car Rental system
US5	As an Admin, I want to change the status of rented cars
US8	As a Customer, I want to return the rented car
US9	As a Customer, I want to make a payment for rented car

Story refinement, with notes:

1. As a Customer, I want to reserve a car from the Car Rental System.

Updated stories:

   a. As a Customer, I want to reserve a car by logging in the Rental system.
   b. As a customer, I want to cancel the booked car
   c. As a customer, I want to return the rented car
   d. As a customer, I want to make payment for the rented car

2. As a Customer, I want to choose the location

   Update Stories:

    a. As a customer, I want to choose pickup location of car
    b. As a customer, I want to choose drop-off location of car

3. As a Customer, I want to make a payment for the car I rented from the
4. Car rental system

Stories to be considered:

1. As a Customer, I want to reserve a car by logging in the Rental system.
2. As a Customer, I want to cancel the booked car in Car Rental system.
3. As a Customer, I want to return the rented car.
4. As a Customer, I want to choose my pickup and drop-off location.
5. As a Customer, I want to make a payment for rented car.

CONCEPTUAL DESIGN

Entity: Location
Attributes:
location_id
email_id
phone_number
address

Entity: Reservation
Attributes:
reservation_id
start_date
end_date
actual_end_date
pentaly_amount
rental_amount
status
license_number
car_id
pickup_location_id
drop_location_id

Entity: Payment
Attributes:
- payment_id
- Card_number
- username
- amount_paid

Relationship: **Location** has **Car**
Cardinality: One to Many
Participation:
- Location has partial participation
- Car has total participation

Relationship: **Card_details** makes **Payment**
Cardinality: One to Many
Participation:
- Card_details has partial participation
- Payment has total participation

Relationship: **Reservation** have **Payment**
Cardinality: One to One
Participation:
- Reservation has total participation
- Payment has total participation

LOGICAL DESIGN

Table: **Location**
Columns:
- location_id
- email_id
- phone_number
- address[foreign key; references **address_id** of **Address**]

Justification: location_id of each location will be uniquely generated by system.

Table: **Payment**
Columns:

<u>payment_id</u>
username[foreign key; references **username** of **CardDetails**]
card_number[foreign key; references **card_number** of **CardDetails**]
amount_paid
reservation_id[foreign key; references **reservation_id** of **Reservation**]

Justification: payment_id will be uniquely generated by system for each payment.

Table: **Reservation**
Columns:

<u>reservation_id</u>
start_date
end_date
actual_end_date
rental_amount
penalty_amount
status
license_number[foreign key; references **license_number** of **Account**]
car_id [foreign key; references **car_id** of **Car]**
pickup_location_id[foreign key; references **location_id** of **Location**]
drop_location_id [foreign key; references **location_id** of **Location]**

Justification: reservation_id will be uniquely generated by system.

VIEWS AND STORED PROGRAMS

Views:

1.Create view EntireDetails as

Select concat_ws(' ' , first_name, last_name) as fullname,cd.card_number,cd.username,c.license_number,r.car_id,cr.car_type,price_per_day,reservation_id,start_date,end_date,actual_end_date,penalty_amount,rental_amount,penalty_amount+rental_amount as amounttobepaid,pickup_location_id,drop_location_id from Customer c inner join Account a on c.license_number=a.license_number inner join CardDetails cd on a.username=cd.username inner join Reservation r on r.license_number=c.license_number inner join Car cr on r.car_id=cr.car_id inner join CarType ct on cr.car_type=ct.car_type inner join Location l on cr.location_id=l.location_id

View: Entire Details
Goal: This view contains details about the customer such as Full name(First name, last name), card number, username, license number and car details such as car id, car type, price per day, reservation details such as reservation id, start date, end date, actual end, date, penalty amount, rental amount, amount to be paid and details about the location such as pick up location and drop off location.
The purpose for creating this view is that all the relevant information is one place i.e rental details including location details and the amount to be paid. Users can view the amount they have to pay including the penalty amount if any and the admin can see entire details of customers including the cars they rented.

2.Create view Rentalinfo as
Select reservation_id,r.license_number, concat_ws(',' , first_name, middle_name,last_name) as FullName, car_type, a.city as 'Pickup Location',ad.city as 'Drop Location', rental_amount
FROM
Reservation r
inner join Car c using(car_id)
inner join Customer using(license_number)
inner join Location l on l.location_id=c.location_id
inner join Location lc on lc.location_id=r.pickup_location_id
inner join Address a on a.address_id=l.address
inner join Address ad on ad.address_id=lc.address;

View: Rentalinfo

Goal: This view contains rental information that is license number, name of the user, pickup and dropoff location and rental amount.
The purpose for this view is that various information from different table is clubbed together so that it becomes easier for Admin to view the details in one place.

Stored Procedure:

```
create procedure GetRentalAmount(in p_license_number varchar(50), out
p_rental_amount int(30), in p_start date, in p_end date, in p_price int (30))
begin
set p_rental_amount =datediff(p_end,p_start)*p_price;
end
```

Stored procedure: Get Rental Amount
Parameters: IN license number varchar, OUT rental amount int, IN start date date, IN end date date and IN price int
Goal: This stored procedure will calculate the rental amount based on the date difference between start date and end date, and price of type of car selected(i.e. Economy has price per day of $45, Mid Size has price per day of $55etc.)

Stored Function:

```
create function GetPenaltyAmount(p_license_number VARCHAR(50),
p_actualend date, p_end date, p_price int(30)) returns int(30) deterministic
begin
set p_rental_amount =datediff(p_end,p_start)*p_price;
end
```

Stored Function: Get Penalty Amount
Parameters:license number varchar,actual end date date,end date date and price int(Stored Function has only IN as parameter type
Goal: This stored function will calculate the penalty amount based on the date difference between end date and actual end date(i.e. If the user returns the car later than the end date selected) and multiplying the price of the car type by 0.5. It will return penalty amount int.

Triggers:

```
1.CREATE TRIGGER `after_reservation_insert` AFTER INSERT ON `Reservation`
 FOR EACH ROW
BEGIN
UPDATE Car set status='Not Available' where car_id=new.car_id;
END
```

Trigger: after insert on Reservation
Goal: This trigger will be invoked when a user reserves a car and updates the car status to not available so that no other user can book the same car.

```
2.CREATE TRIGGER `before_reservation_insert` BEFORE INSERT ON `Reservation`
 FOR EACH ROW
BEGIN
declare location int(30);
select location_id into location from Car where car_id=new.car_id;
set new.pickup_location_id= location;
END
```

Trigger: before insert On Reservation
Goal: This trigger will be invoked when the user reserves the car, the pickup location will be set to the location of the car.

```
3.CREATE TRIGGER before_reservation_update
    BEFORE UPDATE ON Reservation
    FOR EACH ROW
BEGIN
declare p_username varchar(50);
declare p_cardno bigint(30);
declare price int(30);
if old.actual_end_date is null && new.actual_end_date is not null &&
old.status="Booked" then
set new.status = "Returned";
UPDATE Car set status='Available',location_id=new.drop_location_id where
car_id=new.car_id;
select price_per_day into price from Car inner join CarType using (car_type)
where car_id=new.car_id;
set
new.rental_amount=DATEDIFF(new.actual_end_date,new.start_date)*price;
if new.actual_end_date >= new.end_date then
```

```
set new.penalty_amount=
DATEDIFF(new.actual_end_date,new.end_date)*price*0.5;
else
set new.penalty_amount=0;
end if;
select username, card_number into p_username, p_cardno from Account
inner join CardDetails using (username) where license_number=
new.license_number;
insert into Payment(username,card_number,reservation_id,amount_paid )
values(p_username,p_cardno,new.reservation_id,new.rental_amount+new.p
enalty_amount);
end if;
if new.status='Cancelled' then
UPDATE Car set status='Available',location_id=new.pickup_location_id
where car_id=new.car_id;
end if;
END
```

Trigger: before update on Reservation
Goal: This trigger will be invoked when user returns car and it will calculate penalty amount, rental amount, set status to returned and insert payment details into payment table
Explanation:

- When a user returns a car and the actual_end_date is updated then the trigger should calculate the **rental amount(** date difference between the actual end date and start date multiplied price of the car per day.),**penalty amount** if any (if the actual end date is greater than the tentative end date calculate as date diffand set, otherwise set to 0)
- Set status as returned
- Insert the username and card number details into the payment table

Events:

```
1. CREATE EVENT IF NOT EXISTS `Count_cars_event`
ON SCHEDULE EVERY 20 DAY_HOUR
DO
SELECT status, COUNT(status)
FROM Car
```

```
GROUP BY status;
```

Event: Count cars event(Recurring)

Goal: This event will tell the number of cars available/ not available in the car rental system that is how many cars have been rented(not available) and how many cars can be rented(available)

```
2. CREATE EVENT IF NOT EXISTS `Daily_Income`
ON SCHEDULE EVERY 20 DAY_HOUR
DO
select sum(amount_paid)
from Payment
inner join Reservation using(reservation_id)
where curdate()=actual_end_date
```

Event: Daily Income(Recurring)

Goal: This event will calculate income of car rental system at the end of the each day.

Sprint 3:

REQUIREMENTS

Story ID	**Story description**
US4	As an Admin, I want to add a new car to the Car catalog
US1	As a Guest, I want to browse the list of available cars in the Car Rental system
US2	As a Guest, I want to sign up in Car Rental system so that I can rent a car
US3	As a Customer, I want to reserve a car by logging in the Rental system
US7	As a Customer, I want to choose my pickup and drop-off location
US6	As a Customer, I want to cancel the booked car in Car Rental system
US5	As an Admin, I want to change the status of rented cars

US8	As a Customer, I want to return the rented car
US9	As a Customer, I want to make a payment for rented car
US10	As a Customer, I want to buy insurance for my car
US11	As an Admin, I want to give discount to eligible customers
US12	As an Admin, I want to provide additional feature(driver) for the physically challenged people

Story refinement, with notes:

1.As an Admin, I want to give discount to the eligible customers

Updated stories:

a. As an Admin, I want to give discount of 10% to the first time users
b. As an Admin, I want to give one time discount of 25% to the user on his 20th ride from Car Rental system
c. As an Admin, I want to give one time discount of 15% to the user on his 10th ride from Car Rental system

CONCEPTUAL DESIGN

Entity: **Admin**

Attributes:

- <u>username</u>
- password

Entity: **Car**

Attributes:

- <u>car_id</u>
- car_details [composite]
  - model
  - make
  - color
  - manufacturing_year
  - mileage
  - Seating_capacity
- status

car_type
price_per_day

Entity: **Customer**
Attributes:
- <u>license_number</u>
- email_id
- mobile _number
- name [composite]
  - last_name
  - middle_name
  - first_name
- Date_of_birth
- age

Entity: **Account**
Attributes:
- <u>username</u>
- password

Entity: **CreditCard** (Weak Entity)
Attributes:
- <u>card_number</u>
- name_on_card
- Date_of_expiration

Entity: **Address**
Attributes:
- address_line_1
- address_line_2
- city
- state
- zip_code

Entity: **Location**
Attributes:
- <u>location_id</u>
- phone_number

Entity: **Offers**
Attributes:
- <u>promo_code</u>
- description
- promo_type
- percentage
- discounted_amount
- status

Entity: **Additional_Driver**
Attributes:
- <u>license_number</u>
- name

Relationship: **Customer** has **Account**
Cardinality: One to One
Participation:
- Customer has total participation
- Account has total participation

Relationship: **Account** has **CreditCard**
Cardinality: One to Many
Participation:
- Account has total participation
- CreditCard has total participation

Relationship: **Admin** adds a **Car**
Cardinality: One to Many
Participation:
- Admin has partial participation
- Car has total participation

Relationship: **Customer** has **Address**
Cardinality: One to One
Participation:
- Customer has total participation
- Address has total participation

Relationship: **Customer** rents a **Car**
Cardinality: Many to Many
Participation:
- Customer has partial participation
- Car has partial participation

Relationship: **Location** has **Car**
Cardinality: One to Many
Participation:
- Location has partial participation
- Car has total participation

Relationship: **Location** has **Address**
Cardinality: One to One
Participation:
- Location has total participation
- Address has total participation

Relationship: **Customer** pays for **Car**
Cardinality: Many to Many
Participation:
- Customer has total participation
- Car has total participation

Relationship: **Customer** insures **Car**
Cardinality: Many to Many
Participation:
- Customer has partial participation
- Car has partial participation

Relationship: **Customer** gets **Offers**
Cardinality: Many to Many
Participation:
- Customer has total participation
- Offers has total participation

LOGICAL DESIGN

Table: **Admin**
Columns:
<u>username</u>
Password

Justification: username is unique for everyone

Highest normalization level: 4NF

Indexes:
Index 1: clustered
Columns: username
Justification: primary key should be indexed for lookups for table joins

Table: **Customer**
Columns:
<u>license_number</u>
email_id
mobile_number
first_name
middle_name
last_name
Date_of_birth
Residential_address[foreign key; references **address_id** of **Address**]

Justification: <u>license_number</u> is unique for everyone and is natural primary key

Highest normalization level: 4NF
Indexes:
Index 1: clustered
Columns: license_number
Justification: primary key should be indexed for lookups for table joins

Index 1: non-clustered
Columns: first_name,last_name
Justification: A search index would allow for quick searches for admin using customers first and last name.

Table: **Car**
Columns:

Car_id
model_id[foreign key; references **model_id** of **CarModel**]
status
car_type[foreign key; references **car_type** of **Car_Type**]
username[foreign key; references **username** of **Admin**]
location_id[foreign key; references **location_id** of **Location**]

Justification: car_id of each car will be unique.

Note: car_id is number plate of the car.

Highest normalization level: 4NF

Indexes:

Index 1: clustered
Columns: car_id
Justification: primary key should be indexed for lookups for table joins

Index 2: non-clustered
Columns: status
Justification: A search index would allow for quick searches for customers to see all the available cars .

Table: **CarModel**
Columns:

model_id
model
make
color
manufacturing_year

mileage
seating_capacity

Justification: model_id of each car will be unique.

Highest normalization level: 4NF

Indexes:

Index 1: clustered
Columns: model_id
Justification: primary key should be indexed for lookups for table joins

Table: **Car_Type**
Columns:
car_type
price_per_day

Justification: car_type will be unique and a natural primary key.

Highest normalization level: 4NF

Indexes:

Index 1:clustered
Columns: car_type
Justification: primary key should be indexed for lookups for table joins

Table: **Account**
Columns:
username
password
license_number[foreign key; references license_number of Customer]

Justification: username is unique for everyone

Highest normalization level: 4NF

Indexes:

Index 1:clustered
Columns: username

Justification: primary key should be indexed for lookups for table joins

Table: **CreditCard**
Columns:
username
Card_number
name_on_card
date_of_expiration
billing_address[foreign key; references **address_id** of **Address**]

Justification: username and the descriptive attribute(card_number) *is the primary because CreditCard is the weak entity of Account so the primary key of account is a part of the primary key of CreditCard*

Highest normalization level: 4NF
Indexes:
Index 1: clustered
Columns: username, card_number
Justification: primary key should be indexed for lookups for table joins

Table: **Address**
Columns:
address_id
address_line_1
address_line_2
city
state
zip_code
Justification: address_id is uniquely assigned by the system

Highest normalization level: 4NF

Indexes:
Index 1:clustered
Columns: address_id
Justification: primary key should be indexed for lookups for table joins

Index 2:non-clustered
Columns: city

Justification: A search index would allow for quick searches for customers to see all the pickup and drop off locations in a particular city.

Table: **Location**
Columns:
location_id
phone_number
address[foreign key; references **address_id** of **Address**]
Justification: location_id of each location will be uniquely generated by system.

Highest normalization level: 4NF

Indexes:
Index 1:clustered
Columns: location_id
Justification: primary key should be indexed for lookups for table joins

Table: **Payment**
Columns:
payment_id
username[foreign key; references **username** of **CreditCard**]
card_number[foreign key; references **card_number** of **CreditCard**]
amount_paid
reservation_id[foreign key; references **reservation_id** of **Reservation**]

Justification: payment_id will be uniquely generated by system for each payment.

Highest normalization level: 4NF

Indexes:
Index 1:clustered
Columns: payment_id
Justification: primary key should be indexed for lookups for table joins

Table: **Reservation**
Columns:
reservation_id
start_date
end_date
actual_end_date
insurance_type[foreign key; references **insurance_type** of **Insurance**]
promo_code[foreign key; references **promo_code** of **Offers**]
rental_amount
penalty_amount
final _amount
status
license_number[foreign key; references **license_number** of **Account**]
car_id [foreign key; references **car_id** of **Car]**
pickup_location_id[foreign key; references **location_id** of **Location**]
drop_location_id [foreign key; references **location_id** of **Location]**

Justification: reservation_id will be uniquely generated by system.

Highest normalization level: 4NF

Indexes:
Index 1: clustered
Columns: reservation_id
Justification: primary key should be indexed for lookups for table joins

Index 2: non- clustered
Columns: status
Justification: A search index would allow for quick searches for admin to check the status of the reservations (booked,cancelled,returned)

Table: **Insurance**
Columns:

insurance_type
bodily_coverage
medical_coverage
collision_coverage
insurance_price

Justification: insurance_type will be unique and a natural primary key.

Highest normalization level: 4NF

Indexes:

Index 1: clustered
Columns: insurance_type
Justification: primary key should be indexed for lookups for table joins

Table: **Offers**

Columns:

promo_code
description
promo_type
percentage
discounted_amount
status

Justification: promo_code will be unique and a natural primary key.

Highest normalization level: 4NF

Indexes:

Index 1: clustered
Columns: promo_code
Justification: primary key should be indexed for lookups for table joins

Table: **Additional_Driver**

Columns:

name
license_number
reservation_id[foreign key; references **reservation_id** of **Reservation]**

Justification: license_number will be unique and a natural primary key.

Highest normalization level: 4NF

Indexes:

Index 1: clustered

Columns: license_number, Reservation_id

Justification: primary key should be indexed for lookups for table joins

Updated Trigger:

```
CREATE TRIGGER before_reservation_update
    BEFORE UPDATE ON Reservation
    FOR EACH ROW
BEGIN
declare p_username varchar(50);
declare p_cardno bigint(30);
declare price int(30);
declare insurance int(30);
declare perpromoprice int(30);
declare dispromoprice int(30);
declare stat varchar(20);
select status into stat from Offers where promo_code=new.promo_code;
if old.actual_end_date is null && new.actual_end_date is not null &&
old.status="Booked" then
set new.status = "Returned";
UPDATE Car set status='Available',location_id=new.drop_location_id where
car_id=new.car_id;
select price_per_day into price from Car inner join CarType using (car_type) where
car_id=new.car_id;
set new.rental_amount=DATEDIFF(new.actual_end_date,new.start_date)*price;
if new.actual_end_date >= new.end_date then
set new.penalty_amount=
DATEDIFF(new.actual_end_date,new.end_date)*price*0.5;
else
set new.penalty_amount=0;
end if;
if new.insurance_type is NULL then
set insurance=0;
else
select insurance_price into insurance from Insurance where
insurance_type=new.insurance_type;
end if;
set new.final_amount=new.rental_amount+new.penalty_amount+insurance;
```

```
select discounted_amount into dispromoprice from Offers where
promo_code=new.promo_code;
select percentage into perpromoprice from Offers where
promo_code=new.promo_code;

if new.promo_code is not NULL && stat = "Available" then
if dispromoprice is null then
set
new.final_amount=((1-perpromoprice/100)*new.rental_amount)+new.penalty_am
ount+insurance;
else
set
new.final_amount=new.rental_amount+new.penalty_amount+insurance-dispromop
rice;
end if;
end if;

select username, card_number into p_username, p_cardno from Account inner join
CreditCard using (username) where license_number= new.license_number;
insert into Payment(username,card_number,reservation_id,amount_paid )
values(p_username,p_cardno,new.reservation_id,new.final_amount);
end if;

if new.status='Cancelled' then
UPDATE Car set status='Available',location_id=new.pickup_location_id where
car_id=new.car_id;
end if;
END
```

Trigger: before update on Reservation

Goal: This trigger will be invoked when user returns car and it will calculate penalty amount, rental amount, final amount along with the discount and insurance price and set status to returned and insert payment details into payment table

Explanation:

- When a user returns a car and the actual_end_date is updated then the trigger should calculate the **rental amount(** date difference between the actual end date and start date multiplied price of the car per day.),**penalty amount** if any (if the actual end date is greater than the tentative end date calculate as date diffand set, otherwise set to 0),**final amount** as the sum of rental amount, penalty amount, insurance price and offer price

- Set status as returned
- Insert the username and card number details into the payment table