# CAR RENTAL MANAGEMENT - PROJECT REPORT FOR CS 5200

Group Name: SelvakumaranHKandambethR Haritha Selvakumaran NUID: 002727950 Rohith Chandra Kandambeth NUID: 002784518 21 April 2023

# Motivation And Problem Statement:

According to observations, some small businesses already have a vehicle rental system that is not a web-based system application. This restriction allows them to keep customer information, but it also prevents them from making their services more accessible to the general public online; instead, they use posters to promote their services to passer-by. These businesses can solve these issues by converting to the web-based version of the system.

# Top Level Description:

A car rental is a vehicle that may be rented for a price and utilied for a specific length of time. Getting a rental automobile makes it easier for people to travel around when they don't have access to their own vehicle or don't own one at all. A person who needs transportation must call a rental car company and sign a contract. This method improves client retention while also making car and employee management more straightforward. The goal of the suggested online car rental system is for users to easily access the company's website to look for and reserve their preferred cars over the Internet from anywhere in the world at any time. As a result, the business is able to increase customer happiness while also increasing efficiency by giving customers better services. Customers will be able to reserve their vehicles from anywhere in the world due to the Car Rental System. Consumers provide information to this application by filling in their personal information. When a consumer creates an account on the website, he or she can reserve a car. The proposed system is an online system that is fully integrated. It effectively and efficiently automates manual procedures. Customers are aided by this automated method, which allows them to fill in the specifics according to their needs. It contains information on the sort of car they want to hire as well as the location. The goal of this system is to create a website where customers can book their automobiles and request services from anywhere in the world.

# Entities:

The designed system composes of the following entities.

* Customer
* Vehicle
* Vehicle Category
* Location
* Booking
* Membership Packages
* Maintenance
* Insurance
* Employee

# SQL vs NO SQL

Criteria used for selecting SQL over NO SQL database:

* Data is highly structured and that structure doesn’t change frequently.
* High Volume media files are not stored.
* Require a high degree of data integrity and security.
* Doesn’t require the scale-out capabilities that NoSQL offers.
* Does not need flexibility of a dynamic schema or want more choice over the data model.
* Schema does not grow rapidly.
* Hierarchical dependency among the columns is absent.
* Column dependencies are flat.

# Technical description

* **Front End Framework**: HTML, CSS, and Bootstrap.
* **Data Base**: MySQL Server.
* **Back End Framework**: JavaScript, PHP.
* **Operating Systems**: macOS, Windows

# UML Diagram:

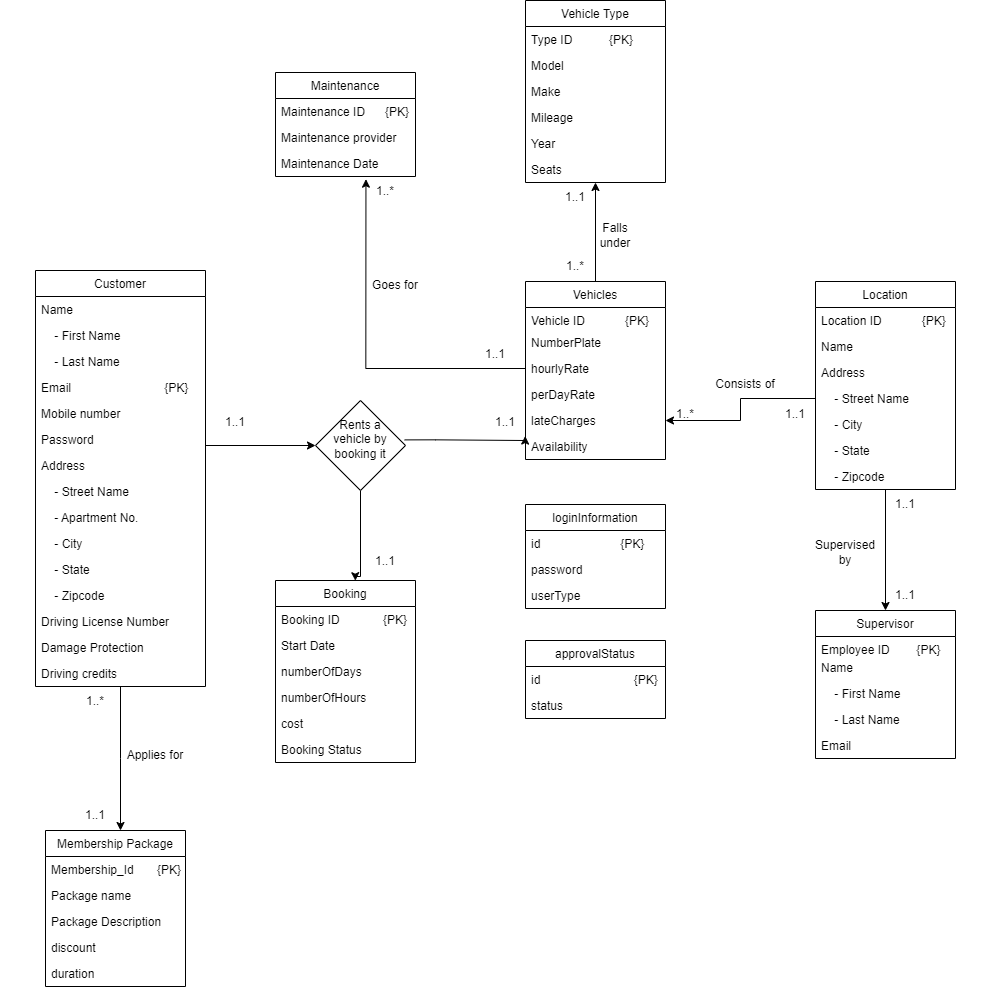


Fig (i): UML diagram for the **CAR RENTAL MANAGEMENT** System

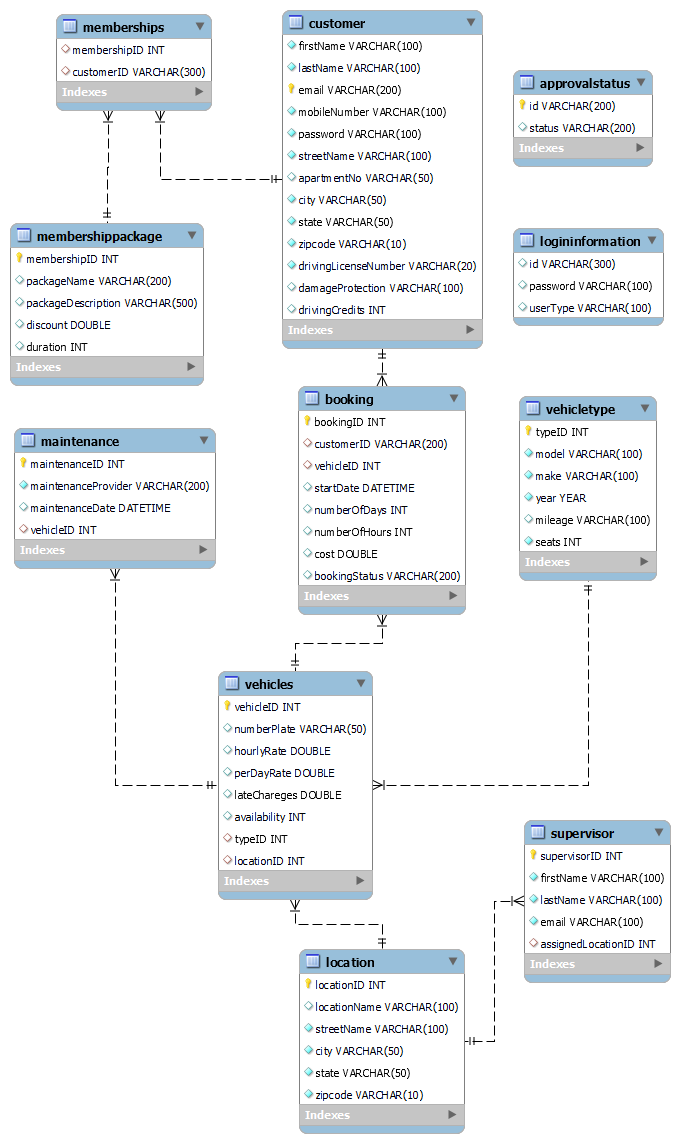


Fig (ii): Reverse Engineered UML diagram for **CAR RENTAL MANAGEMENT** System

# Activity Diagrams:

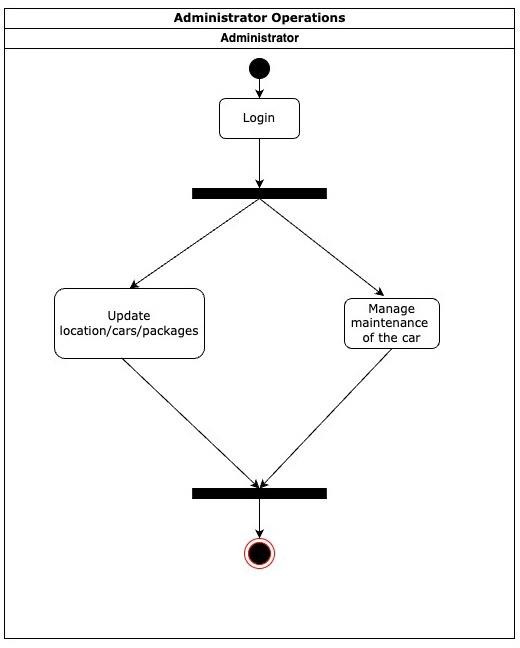
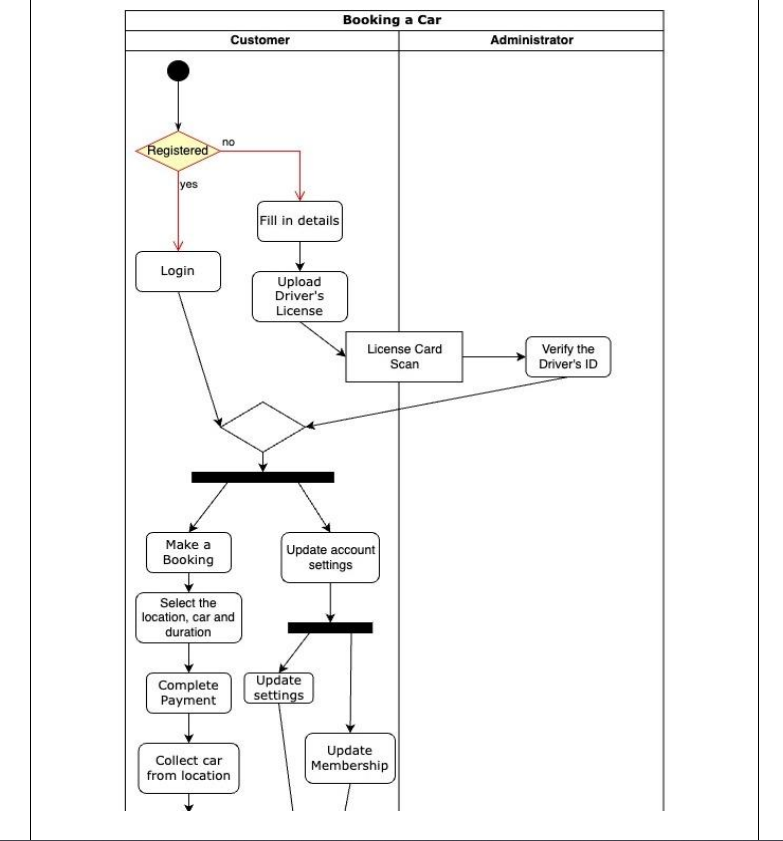


Fig (ii) : Activity diagram for Administrator operations



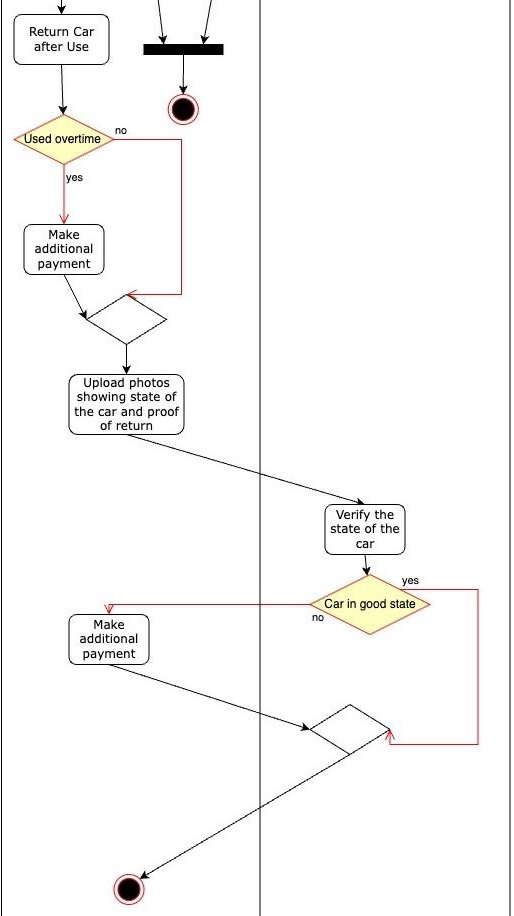


Fig (iii) : Activity diagram for Booking car

# External Libraries Required:

# Bootstrap

# <https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css>

# <https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js>

# SweetAlert2

# <https://cdn.jsdelivr.net/npm/sweetalert2@10>

# jQuery

# <https://code.jquery.com/jquery-3.5.1.slim.min.js>

# Popper.js

# <https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js>

# Logical Design

# Tables Created for the Project –

# customer

# firstName

# lastName

# email – (Primary Key)

# mobileNumber

# password

# streetName

# apartmentNo

# city

# state

# zipcode

# drivingLicenseNumber

# damageProtection

# drivingCredits

# location

# locationID – (Primary Key)

# locationName

# streetName

# city

# state

# zipcode

# loginInformation

# id

# password

# userType

# vehicleType

# typeID – (Primary Key)

# model

# make

# year

# mileage

# seats

# vehicles

# vehicleID – (Primary Key)

# numberPlate

# hourlyRate

# perDayRate

# lateCharges

# availability

# typeID – (Foreign Key)

# locationID – (Foreign Key)

# maintenance

# maintenanceID – (Primary Key)

# maintenanceProvider

# maintenanceDate

# vehicleID – (Foreign Key)

# membershipPackage

# membershipID – (Primary Key)

# packageName

# packageDescription

# discount

# duration

# memberships

# membershipID – (Foreign Key)

# customerID – (Foreign Key)

# supervisor

# supervisorID – (Primary Key)

# firstName

# lastName

# email

# assignedLocationID – (Foreign Key)

# booking

# bookingID – (Primary Key)

# customerID – (Foreign Key)

# vehicleID – (Foreign Key)

# startDate

# numberOfDays

# numberOfHours

# cost

# bookingStatus

# approvalStatus

# id – (Primary Key)

# status

# Foreign Keys

# The vehicle table has 2 foreign keys – One is typeID to associate the vehicle to the vehicle type, and second is the locationID to know from which location the car can be picked up.

# Maintenance table has 1 foreign key – VehicleID is used to relate which vehicle is given for maintenance.

# Memberships table has 2 foreign keys – One is membershipID and the other is customerID to match which customer has what kind of membership.

# Supervisor table has 1 foreign key – assignedLocationID is used to identify the location to which the supervisor is assigned.

# 

# Booking table has 2 foreign keys – One is customerID and other is vehicleID to associate which user has booked which car in a particular booking.

# All the foreign keys have ON UPDATE RESTRICT AND ON DELETE RESTRICT constraint.

# Requirements to build the project - README

# Download xampp

# Upload the SQL dump to MySQL workbench and run the SQL file

# Put the project folder that contains the php files into htdocs folder of xampp

# Update the credentials in dbconnection.php file

# Go to Chrome browser and type localhost/ the folder project folder that contains the php files/ login.php (Eg - localhost/dbms/login.php)

# The php files contain all the frontend code and the backend code used to make modifications to the database.

# To modify or check php files you can use PHPStorm or Visual Studio.

# Technical Flow of the Project

# Firstly, type <http://localhost/dbms/registration.php> to open the registration page. The following page will appear –

# 

# 

# After clicking on Register, the user is redirected to a page that shows waiting for approval. The admin then checks whether driving license number is right or wrong by clicking on approve.

# 

# 3. After the admin accepts the license number, you can then login by going to login page. If password is incorrect then Incorrect password alert is given, or else the user is logged in and taken to select location page.

# 

# 

# 4. In the select location page, you can choose a location from the given options. You also have a navigation bar that contains accounts, memberships, bookings, and sign out option.

# 

# 5. When you click on the Accounts button, it shows the account information of the user.

# 

# 6. When clicked on Bookings, you can see the previous bookings of the user as well as booked or returned status which it fetches from the database.

# 

# 7. When clicked on memberships you can choose the membership package that you want, and based on that you get the offers on the total price.

# 

# 8. When you select a location, you can view all the available cars in that location and information of the car. You then see a book button to book the car.

# 

# 9. After choosing the car it takes you to confirm booking page, where you can calculate cost and then book the car.

# 

# 10. Now when you go back to Bookings tab you can check that the recent booking is updated.

# 11. Now when you login as admin the next page it takes you to is the admin home, where there are a lot of functions that the admin can implement.

# 

# 12. The admin when clicks on add location can add a new location and by using remove location can remove an existing location.

# 

# 

# 13. The admin can also add a maintenance of a car or else remove a car from maintenance once it’s done.

# 

# 

# 14. By clicking on add cars the admin can add a car to the rental system and by clicking on remove the admin can remove cars from the rental database.

# 

# 

# 15. The admin can add new supervisors, remove supervisors, and allocate new supervisors.

# 

# 

# 

# 16. The admin when clicks on return vehicle, can change the car status from booked to returned.

# 

# CRUD Operations

# We have used a total of 9 different operations which are as follows –

# READ for all bookings for a user – The bookings page displays all the bookings that a user has made.

# READ for all users – The credentials for a user login are read and validated against the database.

# CREATE for users – The user can register which creates a new user in the database.

# DELETE for supervisors – The remove supervisor button deletes the supervisor’s name from the database.

# DELETE for locations – The remove location button removes the location from the database.

# UPDATE for vehicles – The return vehicle option under admin, updates the status of vehicle from booked to returned.

# CREATE for cars – The add cars button lets you add new cars to the rental.

# DELETE for cars – The remove cars button lets you delete the cars from the database.

# Lessons Learned

# Were able to figure out tables that could imitate real world scenario and create entities for it. Were also able to understand dependencies between them.

# Were able to understand and implement CRUD operations.

# Could perform complex SQL queries and work on them.

# Were able to implement SQL procedures and triggers.

# ss

# Were able to connect front end to the database and vice versa.

# Learnt how to connect website or applications to the database.

# Implemented Integrity constraints within the tables.

# Implemented CAP theorem in the solution provided.

# Future work

# Planned uses of the database:

# Customer Analysis: The rental management database can be used to analyze customer behavior patterns, such as rental frequency, average rental duration, preferred car types, etc. This information can be used to develop targeted marketing strategies and improve customer satisfaction.

# Fleet Management: The database can be used to track the maintenance schedules of rental vehicles, as well as their usage history, fuel efficiency, and other relevant data. This can help rental companies optimize their fleet utilization, reduce costs, and improve vehicle performance.

# Potential areas for added functionality:

# Online Booking System: The rental management database can be integrated with an online booking system to allow customers to reserve vehicles online. This would improve the customer experience, reduce wait times, and increase rental revenue.

# Mobile Application: Developing a mobile application can help rental companies provide better customer service, offer promotions, and receive feedback from customers. The app can also be used to allow customers to extend their rental periods, make payments, and manage their rental history.

# Predictive Maintenance: The database can be used to develop a predictive maintenance system that uses vehicle usage data to anticipate maintenance needs and schedule service appointments proactively. This would help reduce downtime, increase vehicle longevity, and improve customer satisfaction.