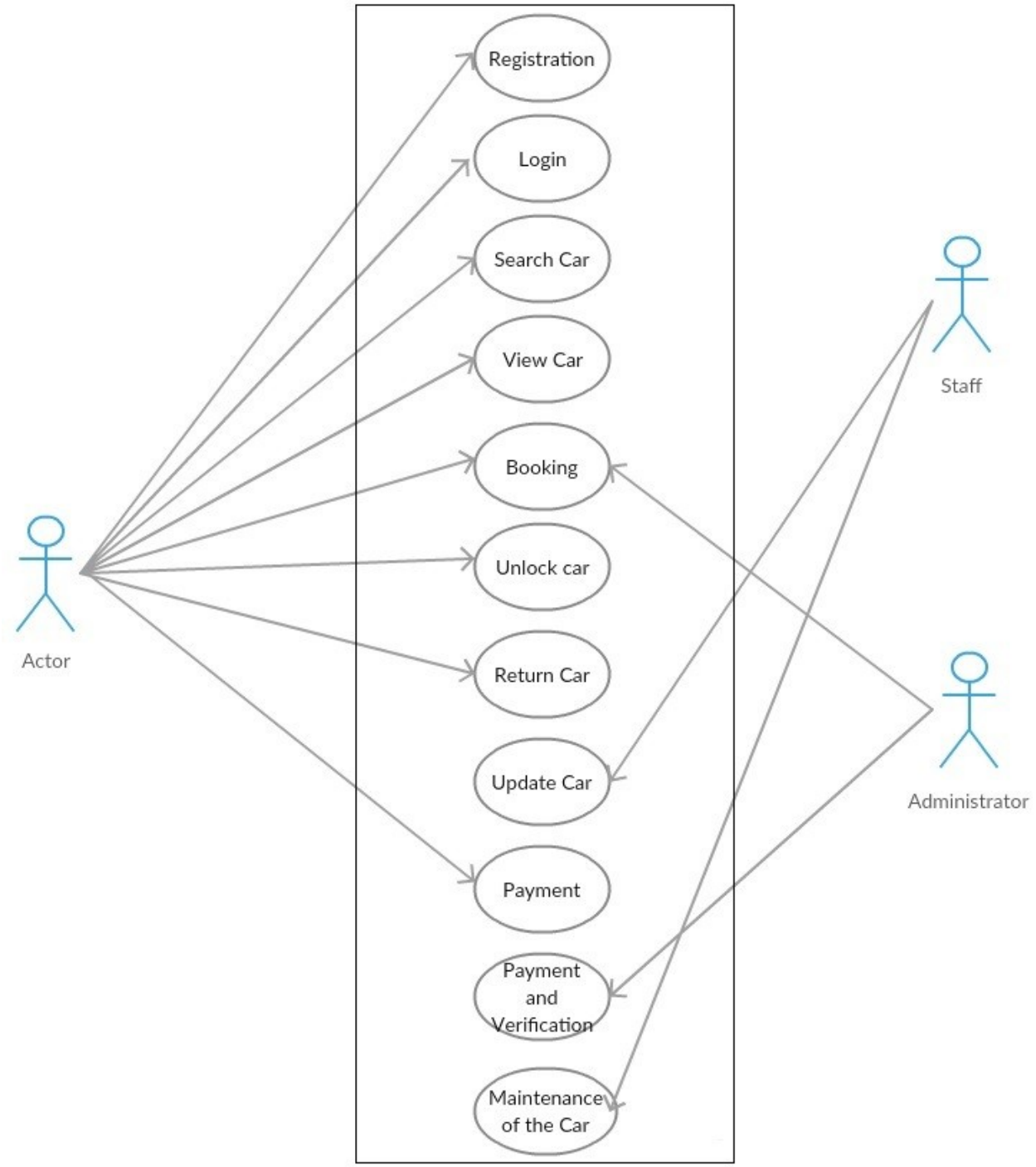
Car rental system

As per the system, a person can book a car according to their needs, e.g., Hatchback, Sedan, SUV. From the location of the person the system will locate the nearest car. Once the person has booked the car, the staff available to that location will provide the user keys to the car. Payment can be done prior to the booking with debit card or credit card.

The billing will be counted according to the time from start to end, and the kilometers the customer has driven the car. The system can constantly locate the car via GPS. Once the customer leaves the car to the dropping point and ends the reservation, the system will show that car is vacant to other users.


Registration
Login
Search Car
View Car
Booking
Unlock car
Return Car
Update Car
Payment
Payment and Verification
Maintenance of the Car
Actor
Staff
Administrator

Detailed description of three use cases:

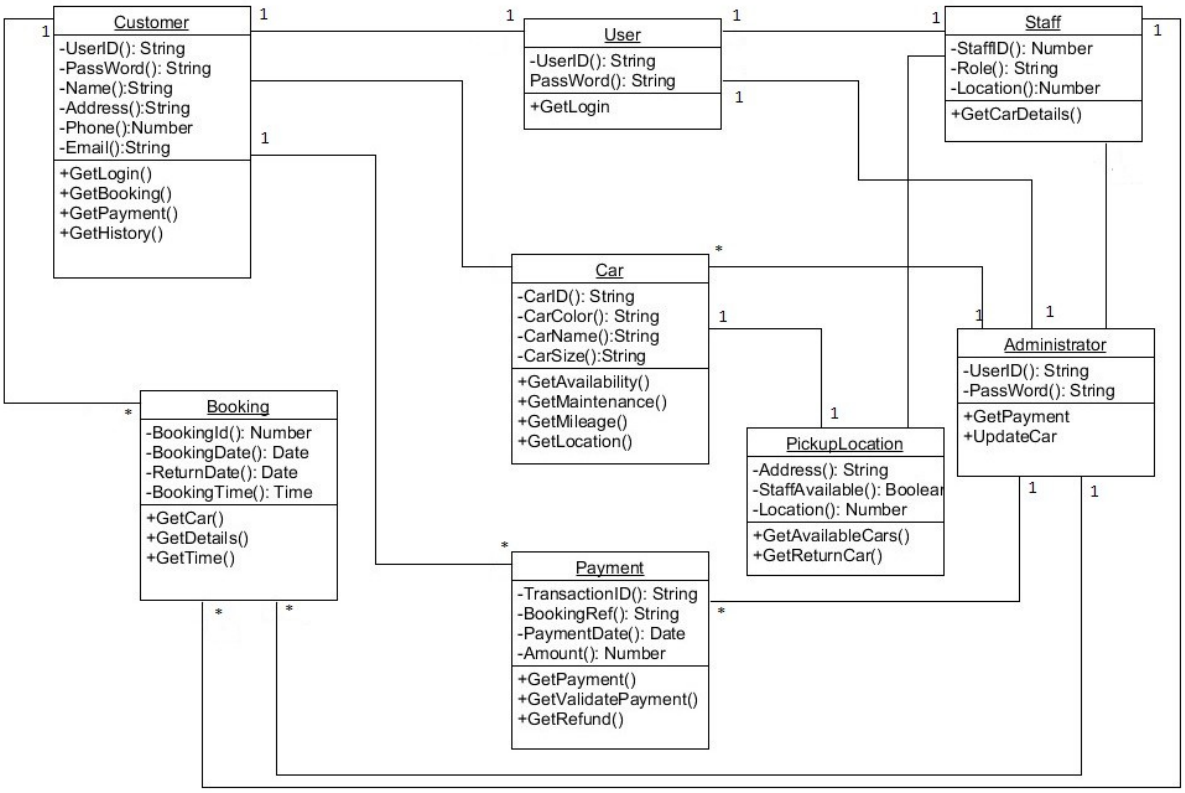
Use Case Name	**Booking**
Participating Actors	Customer
Flow of Events	1. Once the user has become a member and signed into his/her account he/she can choose a car from the dropdown list available in the site according to the preference. 2. Customer has to mention the date and time of when the car is needed. 3. Customer should choose a location to pick up the car from, according to his/her reachability to the location.
Entry Condition	User should have registered to become a member of the system.
Exit Condition	Customer should have the pickup location details of the car he/she booked.
Quality Requirements	The car that the customer chose will be marked as booked and other customers can't book the same car.
Priority	High

Use Case Name	**Unlocking car**
Participating Actors	Customer
Flow of Events	1. Customer needs to go to the pickup location of the car. 2. Customer can use the app in his/her smartphone to identify the car by triggering the horn on the car. 3. Customer will have to unlock the car using his/her smartphone.
Entry Condition	Customer should have booked a car from the system using his account.
Exit Condition	Customer should have gained access to his/her car.
Quality Requirements	Customer should be able to unlock the car from his/her smartphone.
Priority	High

Use Case Name	**Return Car**
Participating Actors	Customer and Staff
Flow of Events	1. Customer parks the car in one of the designated parking locations. 2. The Staff will see to it that the car is returned in proper condition and acknowledge the return of the car by the customer.
Entry Condition	When the customer is ready to return the car.
Exit Condition	The car is returned and is added to the list of available cars in the system.
Quality Requirements	Revoke the unlocking access of the car from the customer.
Priority	High

Registration	The user will provide his/her details. E.g. Name, username, Password, Address, Contact Details, Payment Details (if a customer).
Login	Once the user is registered, as per the user login authentication, users will be given specific rights to the system.
Search Car	Anyone can search for the car on the website, even the user is registered or not. If the user is registered as a customer, he/she can book the car. But if the user is registered as admin, they can make changes in the system. Like, add a new car, remove an old car, or put a car under maintenance for temporary time.
View Car	Customer can view the car as per his requirement for the trip.
Update Car	If the car is recently serviced or anything was changed in the car. It will be updated to the system
Booking	If the user is registered, he/she can book the car as per requirements. For book a car, the user should have a valid license and credit/debit card.
Return Car	Once the customer has finished his/her trip, will return the car to any location as per his/her convenient.
Unlocking Car	The customer will be able to unlock the car with his smart phone, access given to the customer.
Check the car	The staff will check the car for any possible dents, and the kilometers ran.
Payment	The customer will pay the payable amount for the rented car.
Payment verification and acknowledgement	Adminstrator will verify and acknowledge the payment
Maintenance of the car	After certain period of time or kilometers, each and every car will undergo the servicing and regular repairing.

Class Diagram



Entity Object:

- Customer
- Booking
- Staff
- Car
- Administrator
- Payment

Boundary Objects:

- Return the car
- Book the car

Control Objects:

- Payment verification
- Update the car

Detailed Use Case Descriptions:

Use case name	**Registration**
Participating Actors	Customer
Flow of events	1. Customer activates the registration session by clicking on "Sign Up" link. 2. Fills all the required details in the registration form a. First Name b. Last Name c. Postal Address d. Contact Number e. Email address f. Driving License Details 3. Create Username and Password 4. Review the details 5. Submit
Entry Condition	A new Customer visits the website
Exit Condition	Customer will have a unique username and password to log on to the system.
Quality Requirements	All the details will be verified by the system. User will receive and verification email and has to acknowledge to activate the account.
Priority	High

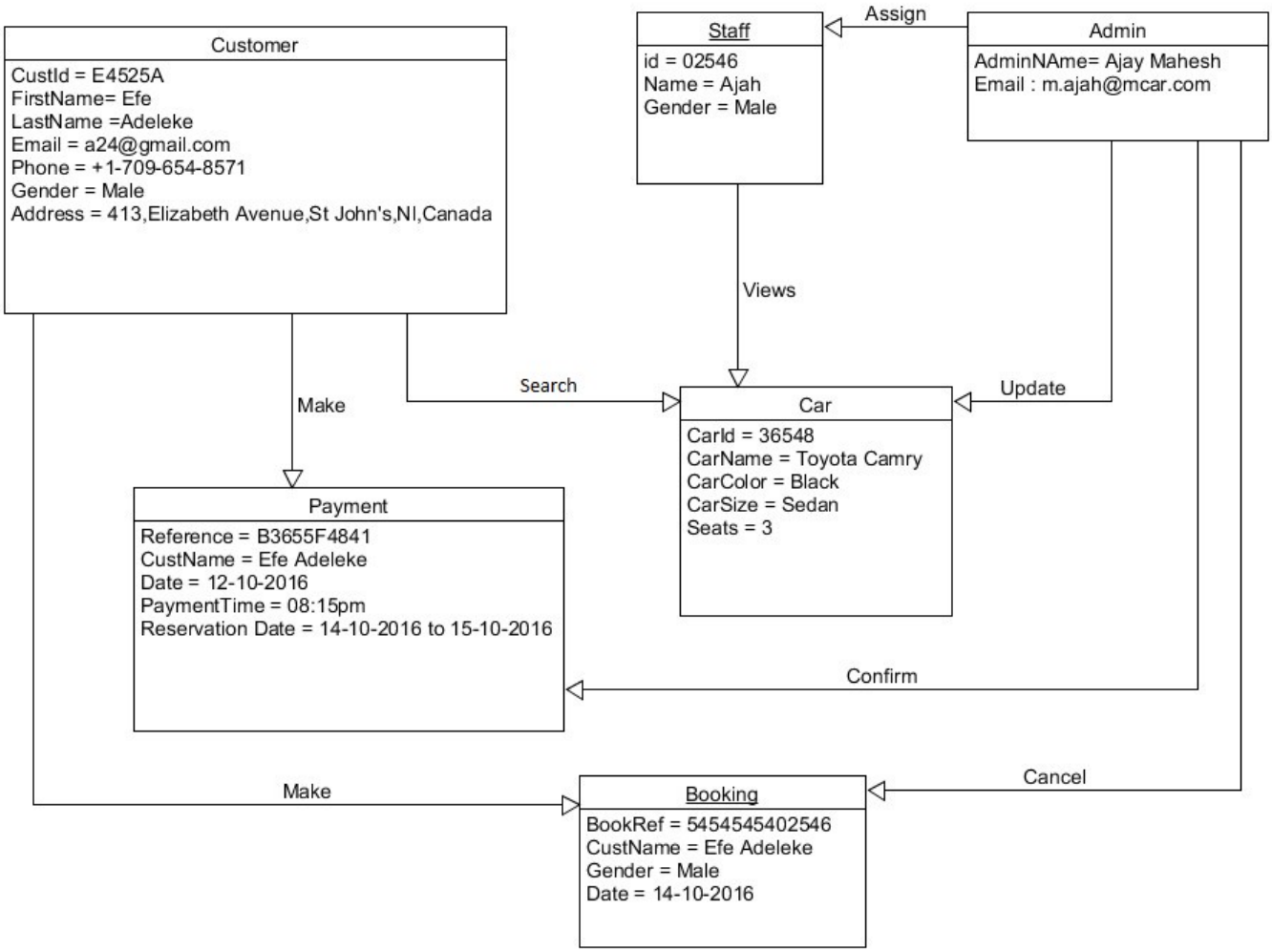
Use case name	Generate Bill
Participating Actors	Staff
Flow of events	1. Customer returns the car at the location. 2. The staff will inspect for any possible damage/dent and will note down the distance travelled. 3. The staff will enter the details in the system and confirms the return. 4. The system will calculate the amount as per distance and time. 5. A PDF bill will be generated and will be sent to the customer's email ID. 6. The due amount will be reflected in the customer account.
Entry Condition	The customer returns the car at the location.
Exit Condition	The bill is generated for the customer.
Quality Requirements	Itemized bill will be generated and amount will be reflected in customer's account when he returns the car.
Priority	High

Design Goals:

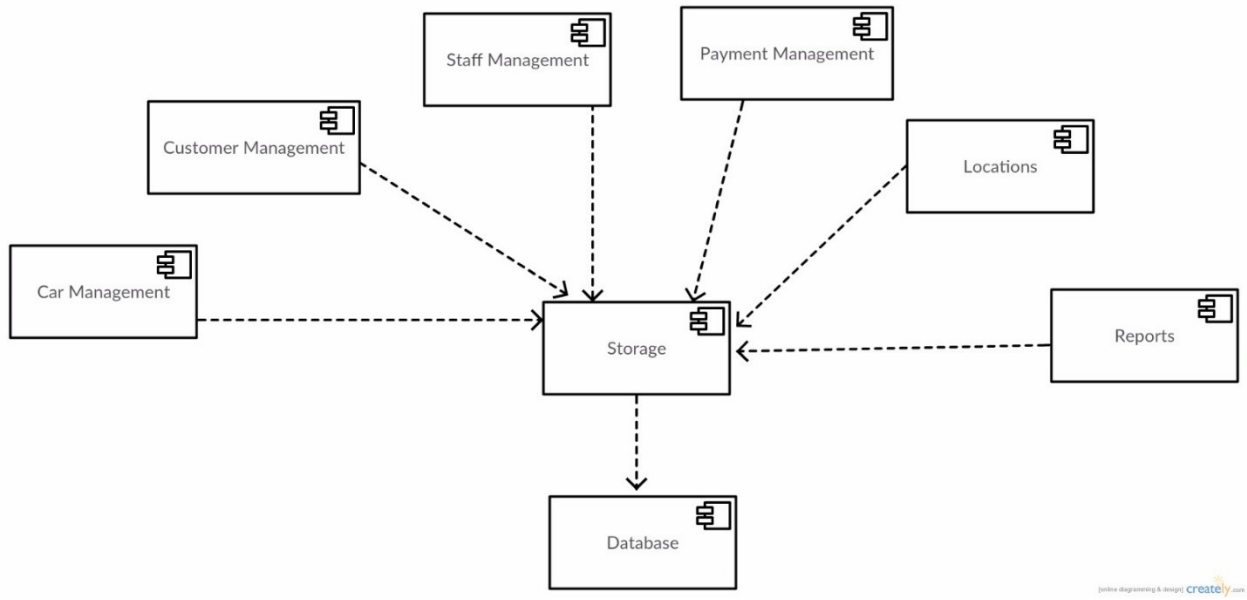
Reliability	The common design goal for the developer and the customer should be that the system be reliable and should work as expected.
Usability	It should be user-friendly so that even a customer with not much basic web browsing skills can use the system with ease.
Availability	As of now, our system is hosted on a 3rd party Linux server. The downtime of the server will be minimum or almost never. For maintenance, we will be using load balancer for HTML and slave server for database. During downtime, the customer will not be able to make any changes in his current booking or new booking. They will be restricted to view the booking.
Portability	The system created is on HTML, java script and node.js, hence it can be adapted to any browsers, operating systems and platforms.
Performance	The system is designed with node.js to serve our webpages and the database is MongoDB. Both the technologies are compatible with each other, hence the connectivity and the

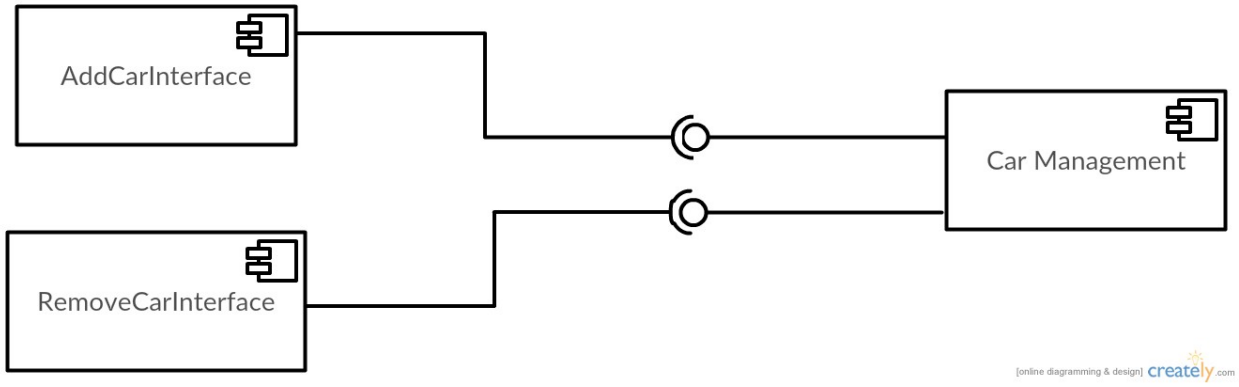
	data exchange rate will be done faster. The data should be stored and in a properly designed database. It should not take more than 2 seconds to reflect the event. The system should perform in a fast manner in terms of responsiveness and stability. The system will be tested thoroughly for defects/bugs before it is deployed.
Scalability	As the numbers of the users grows the system should be able to handle the increase in data as the architecture to design the system is MVC. The HTML pages will directly get connected to the database which will handle the traffic well.
Security	Security testing will be done to make sure the system is not vulnerable to any attacks. The customer information and credentials saved in the database should be protected at all times so that it can't be accessed by any external entities.

Object Modeling:



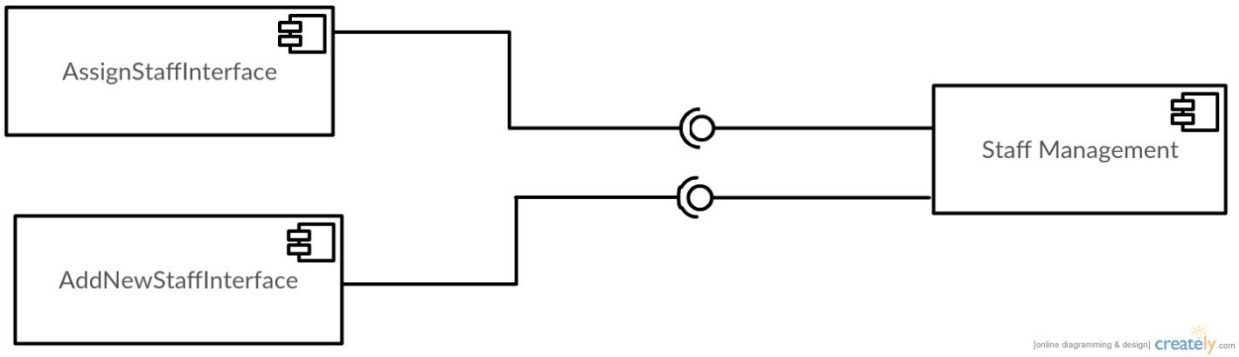
System Decomposition:



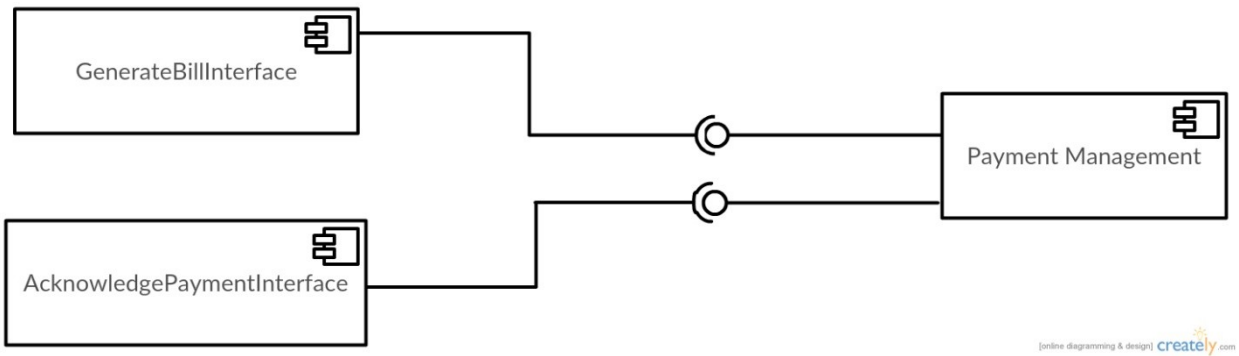


The car management system provides services to the other modules.

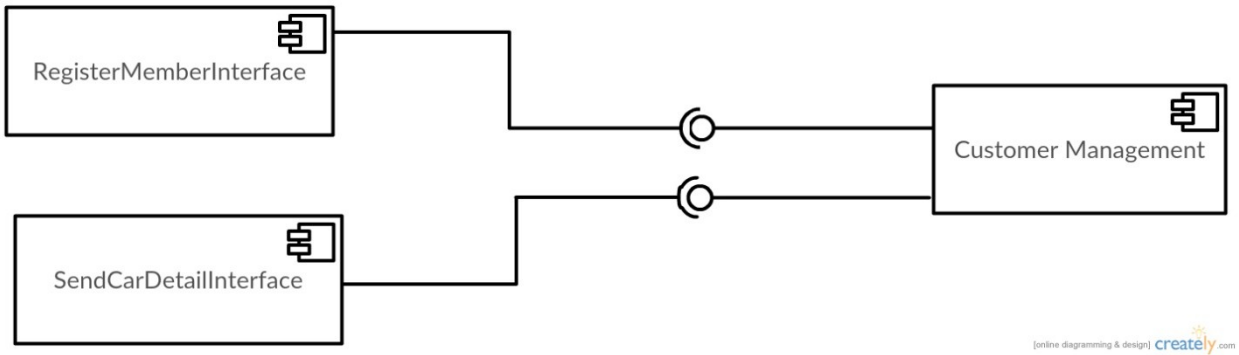
AddCarInterface, and RemoveCarInterface requires the services from Car Management to add and remove cars respectively.



AssignStaffInterface and AddNewStaffInterface requires services from Staff Management to maintain the staff.

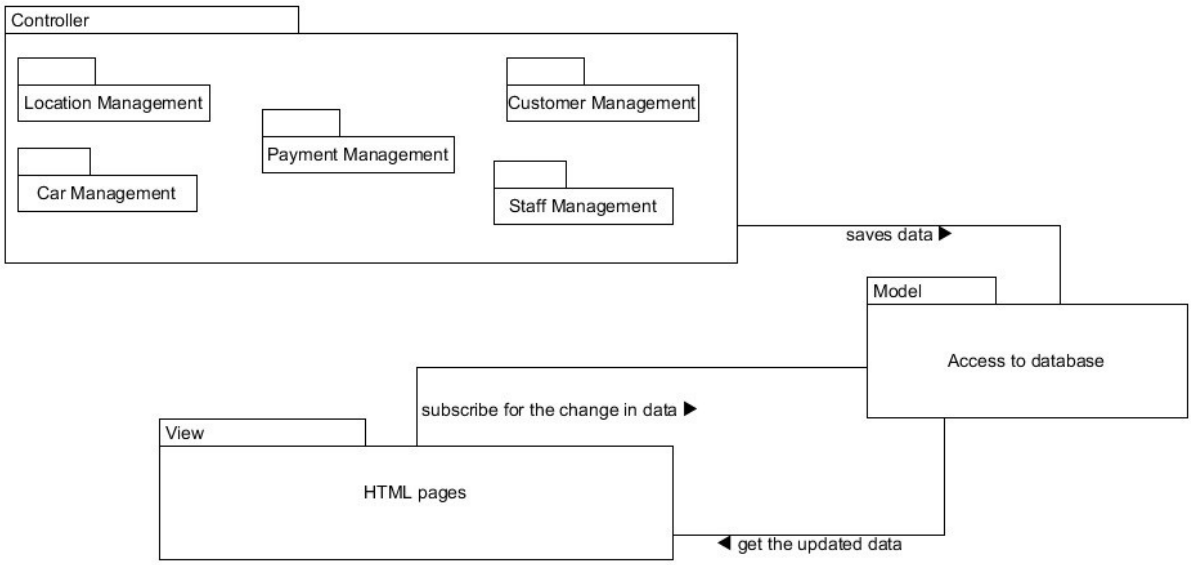


GenerateBillInterface and AcknowledgepaymentInterface requires the services from Payment Management to maintain the payments done by the customers.



RegisterMemberInterface and SendCarDetailInterface requires the services of Customer Management.

Logical Architecture: MVC



For our application, we are going to use "**Model View Controller**" system.

Separating control, data and presentation makes the application more maintainable. It is easier to make changes and it easier for the programmer to comprehend.

Model: handles the connection with the database

View: Entire HTML part will be a part of the view.

Controller: node.js

Implement a Prototype:

We have created a working prototype of the system to book a car. Please refer the following link for the prototype.

https://github.com/masb80