**Online Automotive Dealership System**

**Abstract**

This online car dealership system aims to provide a platform for customers to buy, sell, and test drive cars. The system is designed to make the process of car management and order fulfillment more efficient for both customers and the dealership. The system's main features include car management, order management, and test drive management. The system also includes a login and dashboard module for admin use.

To develop this system, HTML and CSS were used to design the page layout, while JavaScript was used for validation tasks and animations. Python was used for implementing the business logic, and MySQL database was used as the database for the project. The system was developed over the Django Framework.

The purpose of this project is to create a user-friendly and efficient online platform that streamlines the car buying and selling process. As of now, the development of the app is complete, and the app is ready for deployment.

**Introduction**

Having a platform where cars, trucks, and other vehicles may be sold online is essential in today's intensely competitive automotive dealer market. An automotive dealers management system was created to manage the store and sell automobiles online. An automobile store's staff can use this application to manage customers' orders and make it easier for customers to place their own. By categories, the vehicles in the shop are arranged. Each car is assigned a category, name, cost, and description. The staff can log in and manage the category details, product details, customer details, and order details along with the information about order date and time, bill amount, payment status, and delivery status. The following customer information will be maintained: Cust-Id, Name, Delivery Address, Phone number, and Email. The customer must register and log in to access the product page with the automobiles listed. The main point of developing this system is to help Dealers to manage their business and help customers with online ordering.

**Features**

* Login – This module is used for admin login.
* Logout Functionality
* Dashboard – Admin dashboard related to all Car, Car details.
* Car Management Module
  + Adding New Car Details
  + Edit the Exiting Car Details
  + View all the details of the Car
  + Listing of all Car
* Order Management Module
  + Adding New Order Details
  + Edit the Exiting Order Details
  + View all the details of the Order
* Test Drive Management Module
  + Adding New test drive Details
  + Edit the Exiting test drive Details
  + View all the details of the test drives

**Technologies**

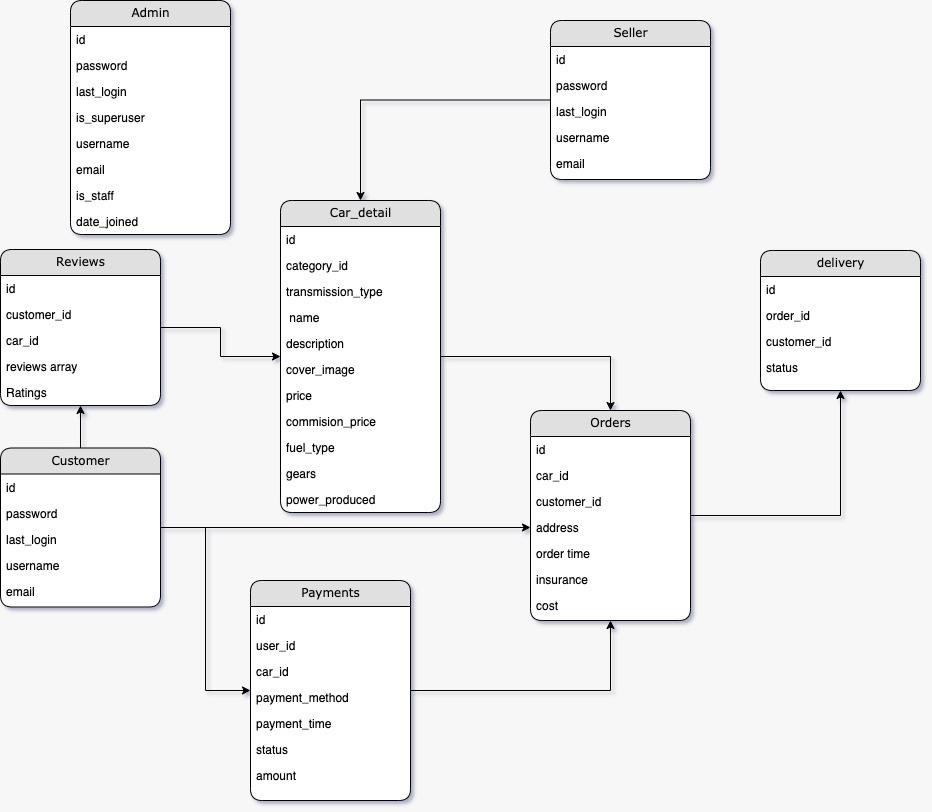
* HTML: The page layout has been designed in HTML
* CSS: CSS has been used for all the designing part
* JavaScript: All the validation task and animations has been developed by JavaScript
* Python: All the business logic has been implemented in Python
* RDBMS: MySQL has used as a database for this project
* Django: Project has been developed over the Django Framework

**Database Description**

Having a platform where cars, trucks, and other vehicles may be sold online is essential in today's intensely competitive automotive dealer market. An automotive dealers management system was created to manage the store and sell automobiles online. An automobile store's staff can use this application to manage customers' orders and make it easier for customers to place their own. By categories, the vehicles in the shop are arranged.

For each car, we must keep information like category, cost, pic, power produced, transmission type, and fuel type. About the order, we store the address for delivery of the vehicle along with the date and status of the order. Forever customer registers with us we store their first name, last name, email, and mobile number with the login credentials. Customers can also book a test drive before they order a vehicle. For each test drive booking, we keep a record of the test drive date. There is no payment gateway since the car dealer can process only check based system.

**Database Model**



**Purpose of the Database**

The purpose of the database in this automotive dealer management system is to store and manage essential data related to vehicles, customers, orders, and test drives. It serves as the backbone of the system, facilitating data retrieval, storage, and manipulation. The database is needed to:

* Store vehicle details: This includes information such as category, name, cost, and description of each vehicle, which is crucial for listing and managing the inventory of the automotive dealer.
* Store customer information: It should maintain customer data, including Cust-Id, Name, Delivery Address, Phone number, and Email, enabling the system to associate customers with their orders and contact them for updates or promotions.
* Store order details: The database should record order information, including order date and time, bill amount, payment status, and delivery status. This allows for order tracking and financial management.
* Store test drive details: It should also keep track of test drive information, such as the date, time, and details of the test drive. This is important for managing and scheduling test drives for potential customers.

**Users and Information Needs**

Dealer: They need access to all modules of the system for managing the inventory, orders, and test drives. They require the ability to add, edit, and view car details, order details, and test drive details. They also need access to the admin dashboard for an overview of the system's performance.

Customers: They need access to the product page with a list of automobiles for sale. They must register and log in to place orders. Customers also want to view their order history and, if applicable, schedule test drives.

**Problem**

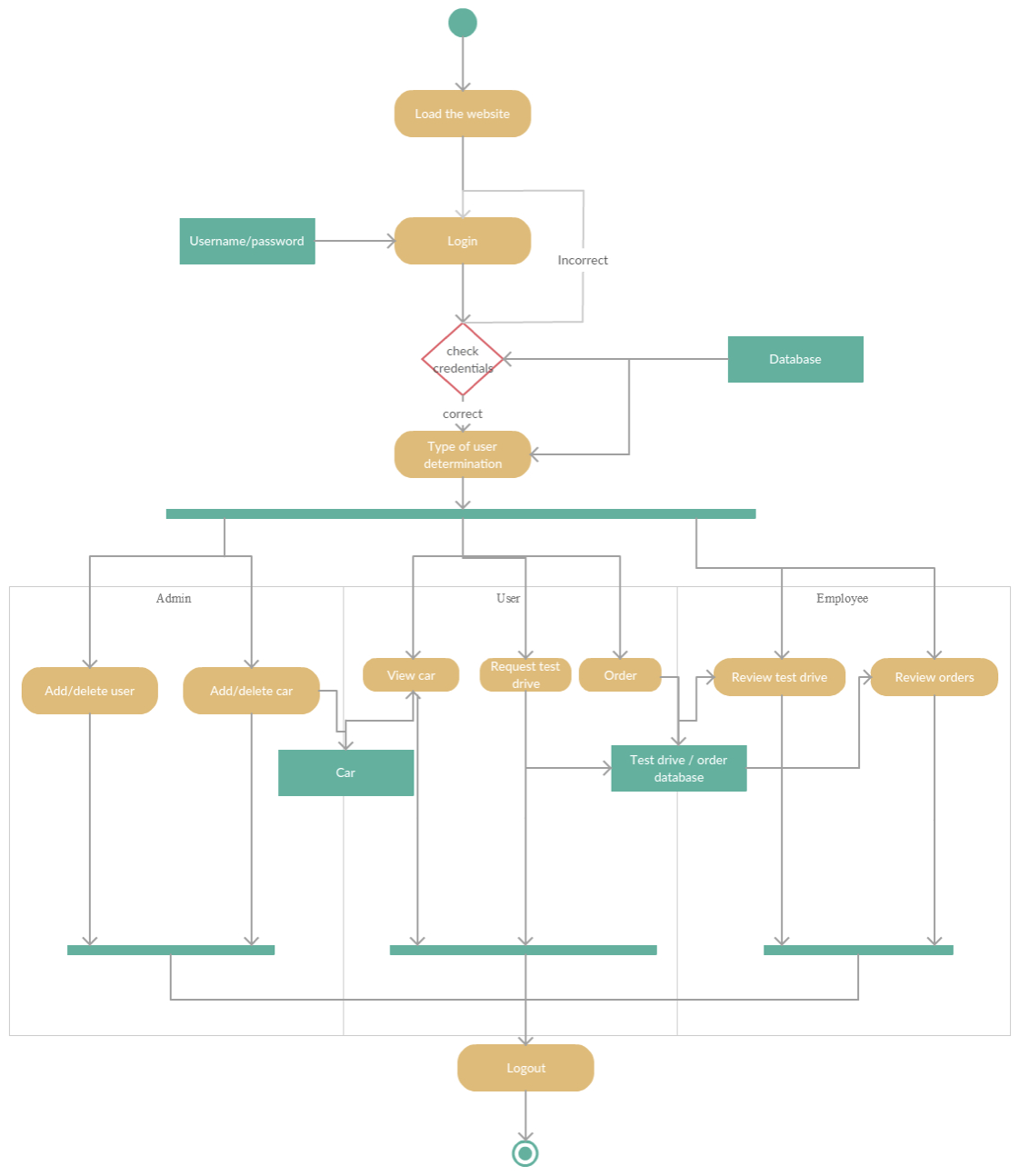
1. Efficiently manage and organize vehicle inventory.
2. Streamline the ordering process for customers.
3. Enable administrators to track and manage orders, including payment and delivery status.
4. Facilitate scheduling and tracking of test drives.
5. Provide a user-friendly interface for both customers and administrators.

**Input Data Available to the Database**

* Vehicle details: Category, name, cost, description.
* Customer information: Cust-Id, Name, Delivery Address, Phone number, Email.
* Order details: Order date and time, bill amount, payment status, delivery status.
* Test drive details: Date, time, and related information for test drive scheduling.

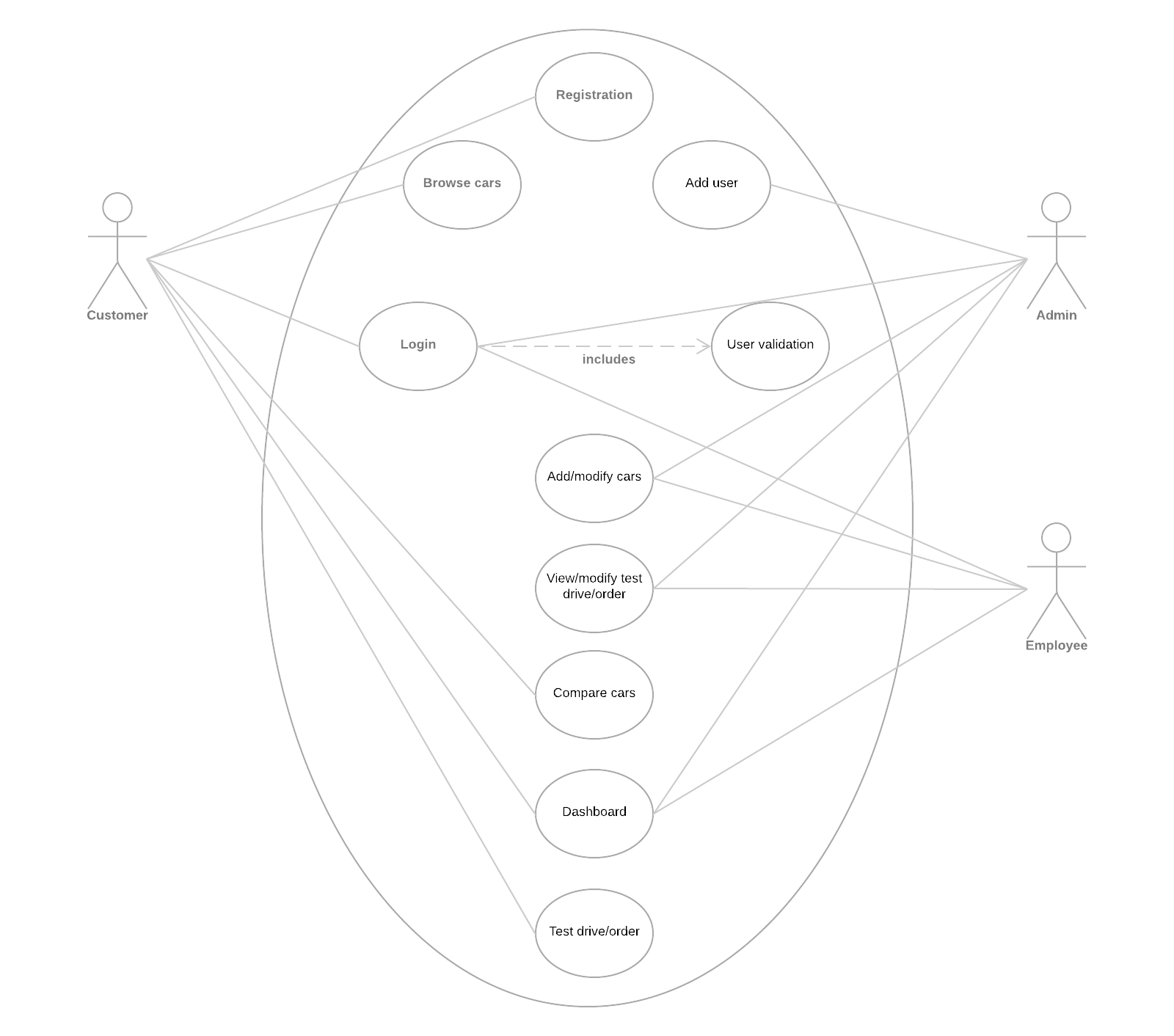
**Activity Diagram**

Activity diagrams can be useful in this online car dealership system to visually represent the flow of activities that occur within the system.

****

Activity diagrams can help to illustrate the steps involved in different processes within the system, such as adding a new car to the inventory, editing an existing car's details, or placing an order. By providing a visual representation of the steps involved in a process, activity diagrams can help to identify potential inefficiencies or bottlenecks in the system. In addition to helping with system design and optimization, activity diagrams can also be useful for communicating the system's functionality to stakeholders, such as developers, managers, or customers. Activity diagrams can provide a clear and concise overview of the system's processes, which can help stakeholders to understand how the system works and how it can be improved.

**Use Case Diagram**

****

A use case diagram can be useful in this online car dealership system to provide a high-level view of the system's functionality and how it interacts with its users. Use case diagrams can help to identify the different types of users who interact with the system, such as customers, dealers, or administrators, and the different use cases or scenarios in which they interact with the system.

For example, a use case diagram could include use cases such as "place an order," "view car details," or "schedule a test drive." These use cases can help to define the different functionalities of the system and how they relate to each other.

**AWS Deployment**

1. **A**mazon EC2 (Elastic Compute Cloud):
   1. EC2 instances will host our Django application.
2. Amazon RDS (Relational Database Service):
   1. RDS is used to host our database.
3. Amazon VPC (Virtual Private Cloud):
   1. Set up a Virtual Private Cloud to isolate our resources and provide network-level security.
4. Amazon Route 53:
   1. Use Route 53 for domain registration and DNS management. It can route incoming traffic to your EC2 instances.
5. AWS Identity and Access Management (IAM):
   1. Create IAM roles and policies to control access and permissions for AWS resources.
6. Elastic IP address
   1. Amazon EC2 instance will have a public IP address

**Implementation**

**Home Screen**

**<add screen from your system>**

This is the landing page of the application where user can register an account or login here.

**Login Screen**

**<add screen from your system>**

This is the login screen for the registered user and also users can navigate to the signup screen to register themselves as new user.

**Register Screen**

**<add screen from your system>**

This is a signup screen for new users.

**Cars Screen**

**<add screen from your system>**

This is the car's screen where the user can see all the available cars for sale and also they can filter the cars based on the category and fuel.

**<add screen from your system>**

This screen is showing the details of the car selected in the previous screen. On this screen, users can book a test drive and order the same car.

**Compare cars**

**<add screen from your system>**

By clicking the compare nav item, the user can reach this page where the user can compare two different cars to understand the features.

**<add screen from your system>**

This is how the features are compared like this in a table where the user can also order or book a test drive from this screen itself.When the user submitted the form, the application says the success message on the screen to acknowledge the booking confirmation

**Screen**

**<add screen from your system>**

When a user clicks the book an order button, this dialog will pop up to get the data from the user to book an order for the car.

**<add screen from your system>**

When the user submitted the form, the application says the success message on the screen to acknowledge the booking confirmation.

**<add screen from your system>**

Once the user ordered a car, they can see the request and status of the request on the user profile page.

**Admin Login**

**<add screen from your system>**

Console to log in the Admin to manage the data in the application

**Adding Cars**

**<add screen from your system>**

This is the admin home screen where the admin can see the users registered, cars available, users booked for test drives and orders users made on the application with the information they entered

**<add screen from your system>**

The screen for adding and editing cars in this online car dealership system would typically include a form for the admin to input or edit the car details. The form may include fields for information such as the car's make, model, year, color, mileage, price, and any additional features or specifications.

To add a new car, the admin would fill out the form with the relevant details and submit it to the system. The system would then add the new car to the inventory.

**Editing Cars**

**<add screen from your system>**

To edit an existing car, the user would select the car from the list of existing cars, which may be displayed in a table or grid format, and then select the "edit" option. This would bring up the same form as used for adding a new car, but with the existing car's details pre-populated in the form fields. The admin could then make any necessary changes to the details and submit the updated form to the system, which would then update the car's details in the inventory.

**View/edit Orders**

**<add screen from your system>**

The screen where an admin can view and edit orders for a car ordered by a user on an application would typically include a variety of information and functionality to help the admin manage and track orders efficiently. The screen would likely display a list of all orders that have been placed, along with relevant details such as the order ID, customer name, car model, and order status.

**<add screen from your system>**

**View/edit Test drives**

**<add screen from your system>**

The screen where an admin can view and edit the test drive requests for a car requested from a user would typically include a table or list of all the test drive requests received by the dealership for a particular car. Each row in the table would represent a single test drive request and would display information such as the user's name, contact details, preferred test drive date and time

From this screen, the admin would be able to view and manage the test drive requests for the car. The admin could edit or cancel a test drive request if needed, by clicking on the appropriate action button next to each test drive request. The admin could also add new test drive requests for the car, by clicking on the "Add Test Drive Request" button.

**Conclusion**

In conclusion, this online car dealership system offers several advantages to both customers and the dealership. The system provides a user-friendly interface for customers to buy, sell, and test drive cars, which can increase sales and customer satisfaction. Additionally, the system's features, such as car management and order fulfillment, can help streamline dealership operations, leading to more efficient business processes.

However, there are also some potential disadvantages to consider. The use of a NoSQL database may not be suitable for large-scale operations, and additional testing may be required to ensure the system's scalability. Additionally, the system's reliance on JavaScript may lead to performance issues on older or less powerful devices.

For future work, the system could be expanded to include additional features, such as a recommendation engine to suggest cars based on customer preferences. The system could also be integrated with other third-party applications to enhance its functionality. Overall, this online car dealership system shows promise in improving the car buying and selling experience for both customers and dealerships and with further development and refinement, it could become an indispensable tool for the automotive industry.