Report 3 – Final Full Report: Vehicle Sales

CSCI441: Software Engineering

Group: Team E

November 23, 2025

Repository URL: https://github.com/wyatt4543/vehicle_sales

Team E Group Members

Moussa Mballo

Wyatt McDonnell

Mamadou Oumar Ndiaye

William Steele

Colin Showalter

Individual Contributions Breakdown

Report 1 – Part 1

Wyatt worked on the problem statement, decomposition into sub-problems, and a glossary of terms.

Moussa worked on the Goals, Requirements, and Analysis - (System Requirements and Analysis).

Report 1 – Part 2

Mamadou worked on Stakeholders & Actors and Goals.

William worked on Use Cases & System Sequence Diagrams

Colin Showalter worked on Preliminary Design & User Effort Information with some assistance from Wyatt

Report 1 – Part 3

Wyatt worked on the entirety of 5. System Architecture and System Design, and Mamadou provided a few updates to the phrasing

Report 2 – Part 1

Wyatt worked on Conceptual Model, Merging the Contributions from Individual Team Members, Project Coordination and Progress Report, and Breakdown of Responsibilities

Mamadou worked on System Operation Contracts

Colin worked on Data Model and Persistent Data Storage

Report 2 – part 2

Wyatt updated the Data Model and Persistent Data Storage and Plan of Work sections to state the relational DBMS being used

Moussa worked on providing Class Diagram and Interface Specification

Colin worked on Interaction Diagrams with some help from Wyatt

Report 2 – Full Report

Wyatt updated the Class Diagram and Interface Specifications to correct the class diagram and associations with domain concepts

Wyatt updated the Plan of Work section to state the software that will be used for integration testing.

Colin worked on the User Interface Design and Implementation section

Wyatt worked on the Test Designs and the Algorithm and Data Structures sections

Report 3 – Final Full Report

Anything not mentioned below is unchanged, and the sections that were unchanged shows what will be completed in demo 2:

Colin added a description of the design pattern we used into the Interaction Diagrams section and wrote out the History of Work section.

Wyatt created the Summary of Changes section, added a Design Patterns section to the Class Diagram and Interface Specification, modified UC10: Manage Vehicle Inventory in section 4a. Preliminary Design, added more detail to the Test Designs section, and added a future work section.

We are planning on having the people assigned to each report alternate between team members, but all team members have contributed equally.

Table of Contents

Summary of Changes

- The delete button from the vehicle inventory webpage in UC10: Manage Vehicle Inventory has been removed. This was so that vehicles cannot mistakenly be deleted, as there is no way to currently add the vehicles back through the website.
- Added verification of identity so that normal users could not access administrative webpages.
- Changed from using Jenkins to Pytest as the website is hosted using Python. Pytest being a Python library allows testing to occur easier because both the website and testing are coded specifically for Python.

Work Assignment

William and Colin have been assigned to front-end development. Moussa and Mamadou have been assigned to back-end development. Wyatt has been assigned to front-end development and back-end functionality.

Individual Student Competencies

Moussa Mballo: Senior CS student at FHSU, advanced programmer in python, java, and databases. My main interest is in cloud computing, and virtual environments.

Wyatt McDonnell (Team Leader): Skills in programming include Python, C++, C#, JavaScript, Java, HTML/CSS, and Lua. Proficient at learning new programming languages within a week. Technical writing skills include documentation, communicating technical concepts through diagrams, and summarizing ideas in text form. My main interest is in game development.

Mamadou Oumar Ndiaye: Senior undergraduate in Computer Science, skills in programming with diverse languages like Java, Python, HTML/CSS and JavaScript. My main interest is in data security and privacy

William Steele: Current CS student at FHSU. I have used several programming languages such as C++, Java, HTML and CSS but haven't had much experience with them outside of classwork. My main interest is game development.

Colin Showalter: Junior FHSU CS major. Experience in C++, Python, Java, JS, and HTML. I am mainly interested in Software Engineering and Game Development

1. Customer Problem Statement

1a. Problem Statement

In many cities throughout the United States, you need a vehicle to be able to travel safely to various locations. Those who cannot successfully walk to their destination are usually forced to purchase a vehicle. Even people outside of cities may require vehicles. Those who live in small towns may want a vehicle for traveling to various destinations for groceries. The most prominent option for any customer is either a private dealer or a car dealership. Generally, these two options are a hassle for people.

At a car dealership or when buying from a private dealer there may be pressure on the buyer caused by their presence. The private dealer may not always give an up-front price. Along with this, the condition of the vehicle may not always be fully provided. There are customers who may need to travel a great distance to a car dealership if they live in a small town. Even if a potential customer does not live in a small town, they may not want to travel to purchase a vehicle.

These inconveniences show that a web application for vehicle sales would provide a more convenient service for purchasing a vehicle. The website would display as a catalogue full of vehicles that shows its image, name, price, and stock. Along with this, the customer can get their vehicle delivered or for pick-up. Accounts will allow any information provided at checkout to be changed.

To better understand the struggles of those who are purchasing vehicles, some potential customer stories are provided below:

Molly

Molly is the manager of a local restaurant. She lives in a small town of 500 people. This restaurant is visited by many of the locals for almost every meal. Every so often she needs to buy ingredients to allow how restaurant to keep running. She takes a van which can contain those ingredients to a nearby city every week. After years of use, her car had finally broken down beyond repair. Molly is now a customer in the market for a new vehicle.

Molly's van being unusable is an issue. Normally, she would be able to drive great distances. The reason it is an issue is because the nearest car dealership is 30 miles away. Along with this, every used vehicle in her town is being sold at a higher price than if the car was new. The used vehicles are outside of Molly's budget, so she would like an alternative option.

Molly would really like a way to have a vehicle delivered to her. Her current situation raises the issue of needing a new way to have ingredients delivered to her. This would likely be costly for Molly's business, and the added management issues would take up more of her time. The increased amount of time spent on her business would not give her much time to purchase a new vehicle. In the past, Molly has used computers while running her business, but she has had difficulties with navigating unintuitive applications. These applications were made to help calculate the cost of meals, but the labels for each button used were often confusing. Eventually,

she decided that she would do the work manually due to how unintuitive the applications were. Also, Molly has often made mistakes while entering information. These mistakes are often not able to be corrected, and this causes Molly to feel frustrated.

John

John is a businessman who needs to budget his money wisely. Even though he would be able to spend money outside of his budget, he would likely have to tighten certain aspects of it. John keeps a record of every purchase he makes, and he does this to make budgeting easier for himself. John lives in the middle of a city, and he finds it very convenient to drive anywhere he needs to go. John usually travels a few miles for groceries every week, and on the weekends, he loves to go to the bar or an arcade. John has not purchased a new vehicle in years, so he believes that his car should be replaced soon.

John does not trust private sellers, so he decides to go to a car dealership. John has a general budget in mind for which type of vehicle he wants to buy. As soon as he gets to the car dealership, he is greeted by someone. He tells them what his general budget is, and that he is looking for a minivan. Even though he explicitly stated he was on a budget, they still showed him around to all of the expensive vehicles in the store. The seller at that dealership was pressuring John to make a purchase every time they went up to a new vehicle. This pressure caused John to purchase a vehicle outside of his budget. He loved how functional his new vehicle was, but he was unable to have as much fun on the weekends due to spending money outside of his budget. John would have rather stayed within his budget.

John likes picking up cars because he does not trust other people with anything he purchases. He would have rather been within his budget and not have the presence of another person to influence him to spend money outside of his budget. In the past, he has bought expensive things due to the influence of other people. It would be convenient for him to have a way to purchase a vehicle with minimal human interaction. John has had no issue with navigating computers in the past, but he has found certain websites to have an inconvenient interface occasionally. He works with computers while at his job, and he mainly uses excel due to his job requiring him to work with numbers. John would like purchase details to be communicated with him in as many ways as possible. Having purchase details communicated to him allows him to budget more accurately. This accuracy comes from having back-ups for if he does something like losing a physical copy of purchase records.

Mark

Mark is a college student with a part-time job as a custodian. He receives an allowance from his parents, and his parents are paying for his education. As a college student, he mainly focuses on his schoolwork to not disappoint his parents. His part-time job leaves him with only his weekends to spend his free time. He lives on-campus, so there are businesses like grocery stores and fast-food places throughout his city. While many of these businesses are within walking distance, it is much safer for him to use a vehicle to go to any of these locations. For this reason, he would like to purchase a vehicle.

He decides to look for a used car on Craigslist. The reason for this is because he wants to purchase something within his budget. He finds someone that is selling a used car listed with various parts repaired including the transmission. After messaging the seller, they both come to

an agreement on where to meet. Mark takes the time to walk all the way to the seller's house about 5 miles from campus. While there the seller tries to make him pay more than the initial price listed on the seller's page. Mark attempts to dismiss the seller's price hiking, but he fails to do so and pays for the car at a higher price than was listed. He drives the vehicle back to his campus's parking lot, and he parks the vehicle normally. The next day, he attempts to start his car and fails to do so. He checks under his car's hood and realizes that the transmission had been incorrectly replaced. Mark is forced to replace his transmission and spends more money than if he had purchased a new car.

Mark likes having things delivered, but this situation has made him trust people less. For this reason, he would rather pick up any future vehicles. This was not the first time Mark had been scammed. He was once scammed out of his money while trying to buy an online currency for his favorite video game, but that had not left a lasting impact on his opinion of other people. It would be convenient for him to have a price that does not change as well as a guarantee of the quality of a vehicle. He would like to pick up future vehicles to look at their quality in person. Mark has had access to computers since he was a small child, and he has had access to a phone since he was a teenager. Whether he was on a website or using an application, a horrible user interface always caused him to delete that application or close out of a website. Mark would like to have his data secure as he does not want to have any future problems with any person.

1b. Decomposition into Sub-problems

Individual Customers Like Molly:

1. Customers want to have a car delivered due to the distance to a car dealership being inconvenient to travel to.
2. Customers want a convenient user interface to navigate, so they will not waste their time while purchasing a vehicle.
3. Customers want an up-front price, so they can budget for a vehicle appropriately.
4. Customers would like an account system to be able to correct any mistakes they might have made when entering any information.

Individual Customers Like John:

1. Customers want to have a way to pick-up a vehicle due to the possibility of the vehicle being damaged
2. Customers want a convenient user interface to navigate, so they will not waste their time while purchasing a vehicle.
3. Customers want a way to have minimal interaction with another person
4. Customers would like multiple sources for purchase details for budgeting purposes

Individual Customers Like Mark:

1. Customers want to have a way to pick-up a vehicle due to the possibility of the vehicle not being of the quality stated
2. Customers want a convenient user interface to navigate, so they will not waste their time while purchasing a vehicle.
3. Customers want an up-front price, so they can budget for a vehicle appropriately.
4. Customers want a way to have minimal interaction with another person
5. Customers want a secure system, so none of their information is provided to someone they do not trust

1c. Glossary of Terms

Catalog: The digital listing of all available vehicles.

Stock: Number of units available for each vehicle model.

Pick-up Code: A unique identifier provided to customers who choose in-store collection, used for verification.

Delivery Option: The method where the purchased vehicle is delivered to the customer's provided address.

Verification: The process of confirming customer identity and validating payment before finalizing a purchase.

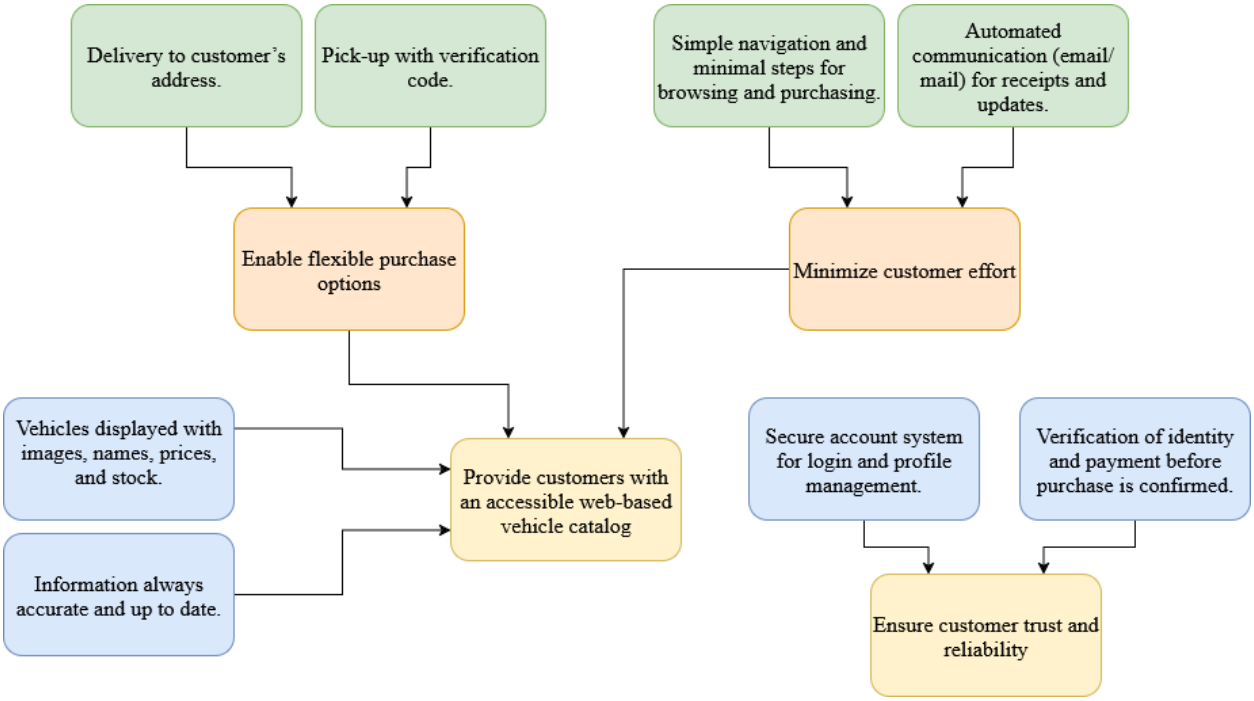
Account System: A feature allowing users to register, log in, and manage their personal details.

2. Goals, Requirements, and Analysis

2a. Business Goals

The main business goal is to create a transparent, convenient, and pressure-free online platform for purchasing vehicles. The platform is intended to reduce the stress and inconvenience associated with buying from dealerships or private sellers.

Goal Hierarchy:



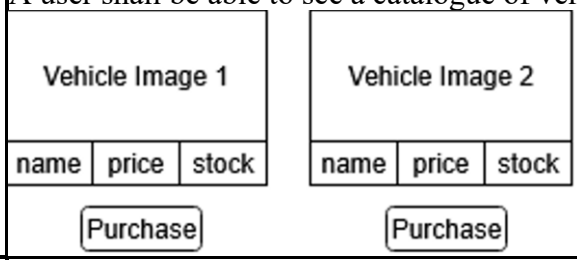
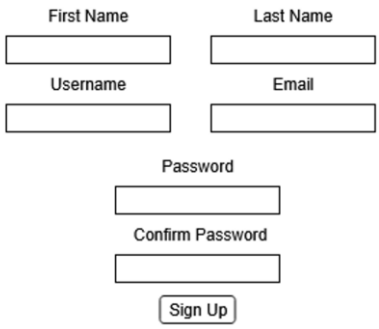
2b. Enumerated Functional Requirements

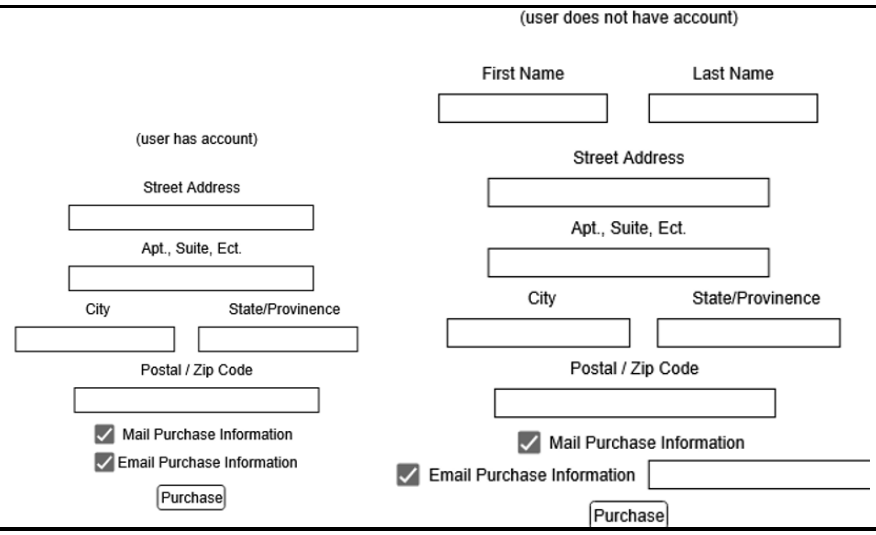
ID	PW	Requirement	Justification
REQ1	5	The website shall display vehicles in a catalog with image, name, price, and stock.	Customers require visibility of all available options before making decisions.
REQ2	5	The website shall display up-to-date information for each vehicle's price and stock.	Prevents misinformation and ensures fair pricing.
REQ3	5	The website shall allow customers to create an account and log in.	Accounts are necessary for managing personal and purchase details.
REQ4	5	The website shall store and allow updates to user-provided purchase information.	Users may need to change address, payment, or contact details.
REQ5	5	The website shall verify customer identity and payment before completing a purchase.	Protects against fraud and ensures legitimate transactions.
REQ6	3	The website shall provide customers with a unique pick-up code if they choose in-store collection.	Ensures secure and efficient pick-up process.
REQ7	3	The website shall provide delivery option by collecting and verifying customer address.	Adds flexibility for users in smaller towns or with limited mobility.
REQ8	3	The website shall send confirmation and purchase details via email/mail.	Keeps customers informed of transaction details.
REQ9	1	The website shall allow filtering/searching vehicles by price, model, or availability.	Improves customer experience but is not core functionality.
REQ10	3	The website shall allow Administrators to see a report of information on what customers have bought.	Allows system managers to see the number of vehicles purchased, their name, and their price.

2c. Enumerated Non-Functional Requirements

ID	Category	PW	Requirement
REQ11	Performance	5	The website shall load catalog pages within 3 seconds under normal load.
REQ12	Reliability	5	The system shall not crash more than once a month during normal operation.
REQ13	Security	5	Customer data and transactions must be encrypted using SSL/TLS.
REQ14	Usability	3	A new user shall be able to complete a purchase in fewer than 7 clicks from the homepage.
REQ15	Scalability	3	The system shall support at least 500 concurrent users.
REQ16	Maintainability	3	Code shall be modular so schema updates can be applied within 2 hours.
REQ17	Portability	1	The website shall be accessible on desktop, tablet, and mobile browsers without layout issues.

2d. User Interface Requirements

ID	PW	Requirement
REQ18	5	A user shall be able to see a catalogue of vehicles 
REQ19	5	The main webpage shall contain a sign in and sign out button 
REQ20	5	A user shall be able to sign up/edit information for an account 
REQ21	5	A user shall be able to sign into an account 
REQ22	5	A user shall be able to enter delivery and/or mailing information

		
REQ23	3	A user shall be able to logout 

3. Functional Requirement Specification and Use Cases

This section identifies the different stakeholders that will take part in the project, the actors involved and their goals but also outlines the use cases for the online vehicle purchasing website based on the requirements and user stories.

3a. Stakeholders

Stakeholders are individuals or groups with an interest in the success of our project. This includes:

- **Customers:** Individuals like Molly, John and Mark who purchase vehicles and use the website to do so.
- **Business Owner/Management Team:** The group responsible for the website's operations, sales, and inventory management.
- **Web Development Team:** The individuals responsible for building and maintaining the website.
- **Financial Institutions/Payment Providers:** Organizations that process online transactions.
- **Delivery and Logistics Partner:** The company responsible for delivering vehicles to customers.

3b. Actors and Goals

Actors are the different roles that interact directly with the system. They are as followed with their goals:

- **Customer (Initiating Actor)**
    - **Searches, views and orders from the website**
    - **Chooses a way to receive their product (delivery or pickup)**
    - **Manages their account information**
- **Administrator (Initiating Actor)**

    - **Manages customer orders**
    - **Manages the product inventory (add, remove, update the stock)**
    - **Checks sales report**
- **Database (Participating Actor)**
    - **Keeps all the information related to the vehicles, the users, transactions, etc...**
- **Payment Gateway (Participating Actor)**
    - **Processes various payments in a secure manner**
- **Mailing Service (Participating Actor)**
    - **Sends order confirmation messages to the customers (email, imessage, etc...)**

3c. Use Cases

i. Use Cases

UC1: Browse Catalogue

- Description: Allows the customer to view vehicles from the catalog along with their details (image, name, price, stock)
- Derived from requirements REQ1 and REQ2

UC2: Create account and log-in

- Description: Allows a user to create or to log into an account in the system for personal details to be used.
- Derived from requirements REQ3

UC3: Manage Purchase Information

- Description: Allows the customer to update their information such as address, contact information, and payment information.
- Derived from requirements REQ4

UC4: Verify and Complete Purchase

- Description: Allows the customer to finalize a purchase and make a secure purchase through the payment gateway.
- Derived from requirements REQ5

UC5: In-Store Collection

- Description: Allows the customer to collect a purchased car at the store using a unique pick-up code.
- Derived from requirements REQ6

UC6: Home Delivery

- Description: Allows the customer to have a purchased car delivered to a chosen address.
- Derived from requirements REQ7

UC7: Send Purchase Confirmation

- Description: Allows the system to send confirmation details to the customer from the mailing service.
- Derived from requirements REQ8

UC8: Search/Filter Vehicles

- Description: Allows the customer to filter catalog for specific criteria (price, model, etc.)
- Derived from requirements REQ9

UC9: Manage User Information (Admin)

- Description: Allows the Administrator to view or update a customer's profile information
- Derived from REQ4

UC10: Manage Vehicle Inventory (Admin)

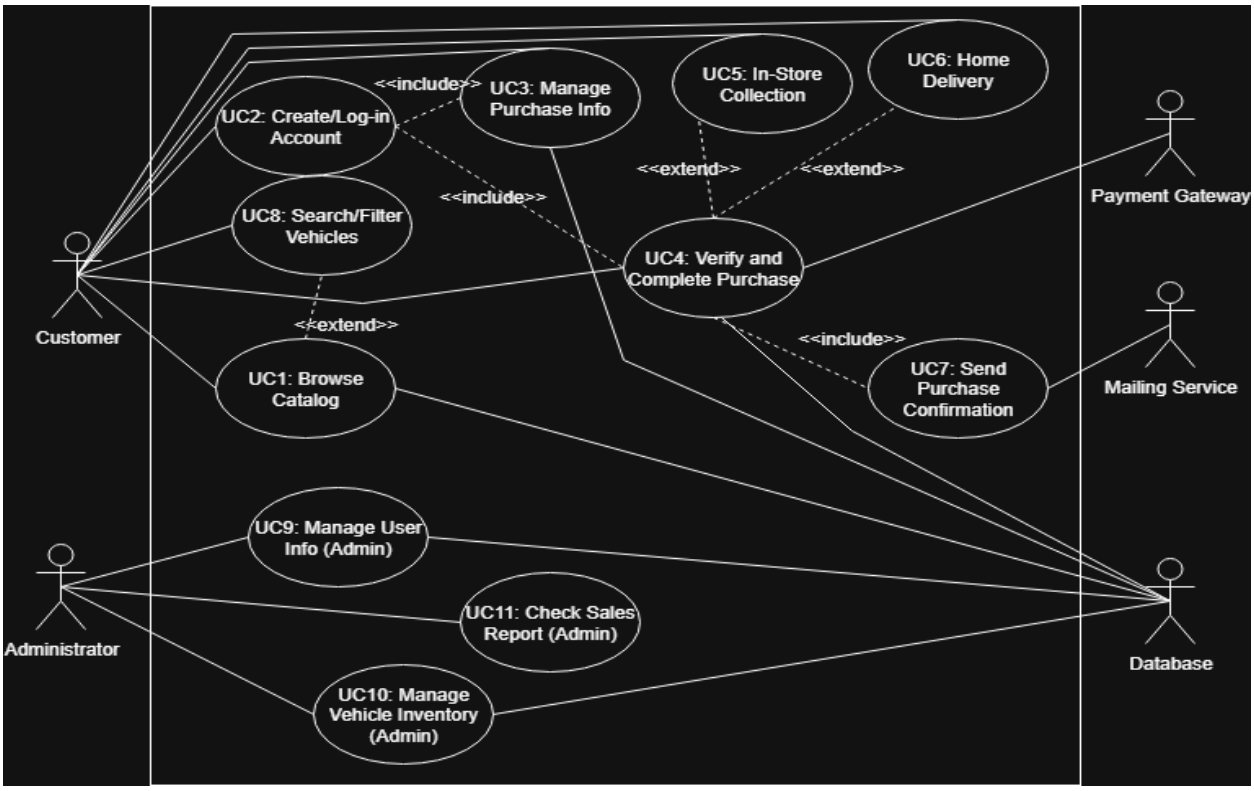
- Description: Allows the Administrator to add or remove vehicles to the catalog as well as update details about vehicles.

- Derived from REQ2

UC11: Check Sales Report (Admin)

- Description: Allows the Administrator to see a report of information on what customers have bought
- Derived from REQ10

ii. Use Case Diagram



iii. Traceability Matrix

REQ	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC 11
REQ1	5	X										
REQ2	5	X									X	
REQ3	5		X									
REQ4	5			X						X		
REQ5	5				X							
REQ6	3					X						

REQ7	3					X					
REQ8	3						X				
REQ9	1							X			
REQ103											X
Total Weight	10	5	5	5	3	3	3	1	5	5	3

iv. Fully Dressed Description

Use Case UC1: Browse Catalog

Name:	UC1: Browse Catalog
Actors:	Customer (Initiating), Database. (Participating)
Precondition:	Customer accessed the system. Data exists in the system
Postcondition:	The customer can view the vehicle catalog
Main Flow Steps	1. The customer starts browsing 2. System requests information from database 3. Database returns the data about the vehicles and their information 4. System displays the information to the customer 5. Customer view and explores the information
Alternate Flow Steps	If there are no vehicles in stock the system will show a message to the customer telling them that there are no cars in the database. If the database fails to connect, then the system will show the user a message telling them it was unable to load and to try again later

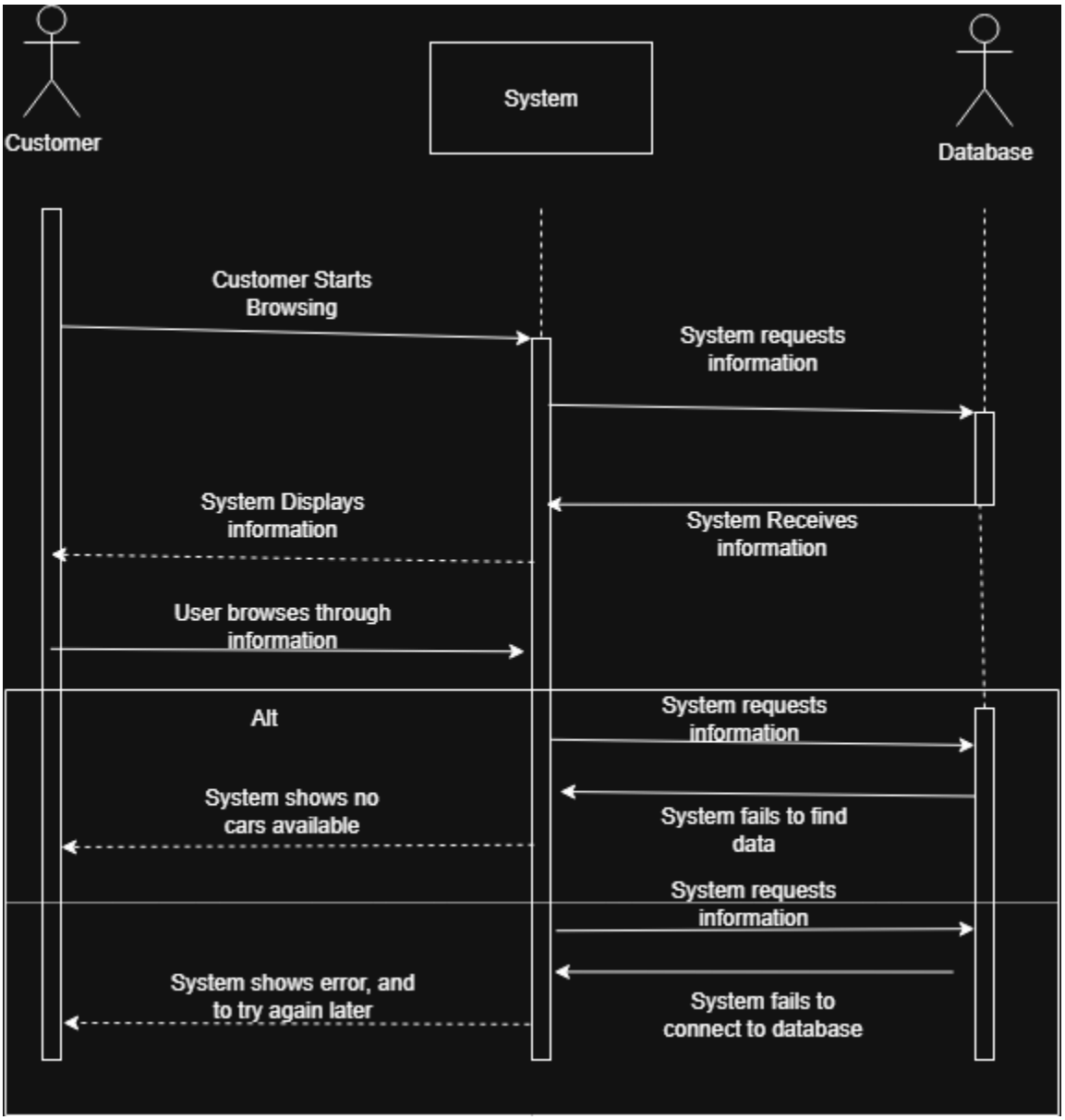
Use Case UC4: Verify and Complete Purchase

Name:	UC4: Verify and Complete Purchase
Actors:	Customer (Initiating), Payment Gateway (Participating), Database (Participating)
Preconditions:	Customer is logged in, Customer is trying to buy a vehicle
Postconditions:	Payment is verified
Main Flow Steps:	1. Customer starts the checkout 2. System retrieves customer information from database 3. System asks customer to review details 4. Customer confirms details 5. Details sent to Payment gateway 6. Payment gate way verifies information 7. System continues and triggers UC7

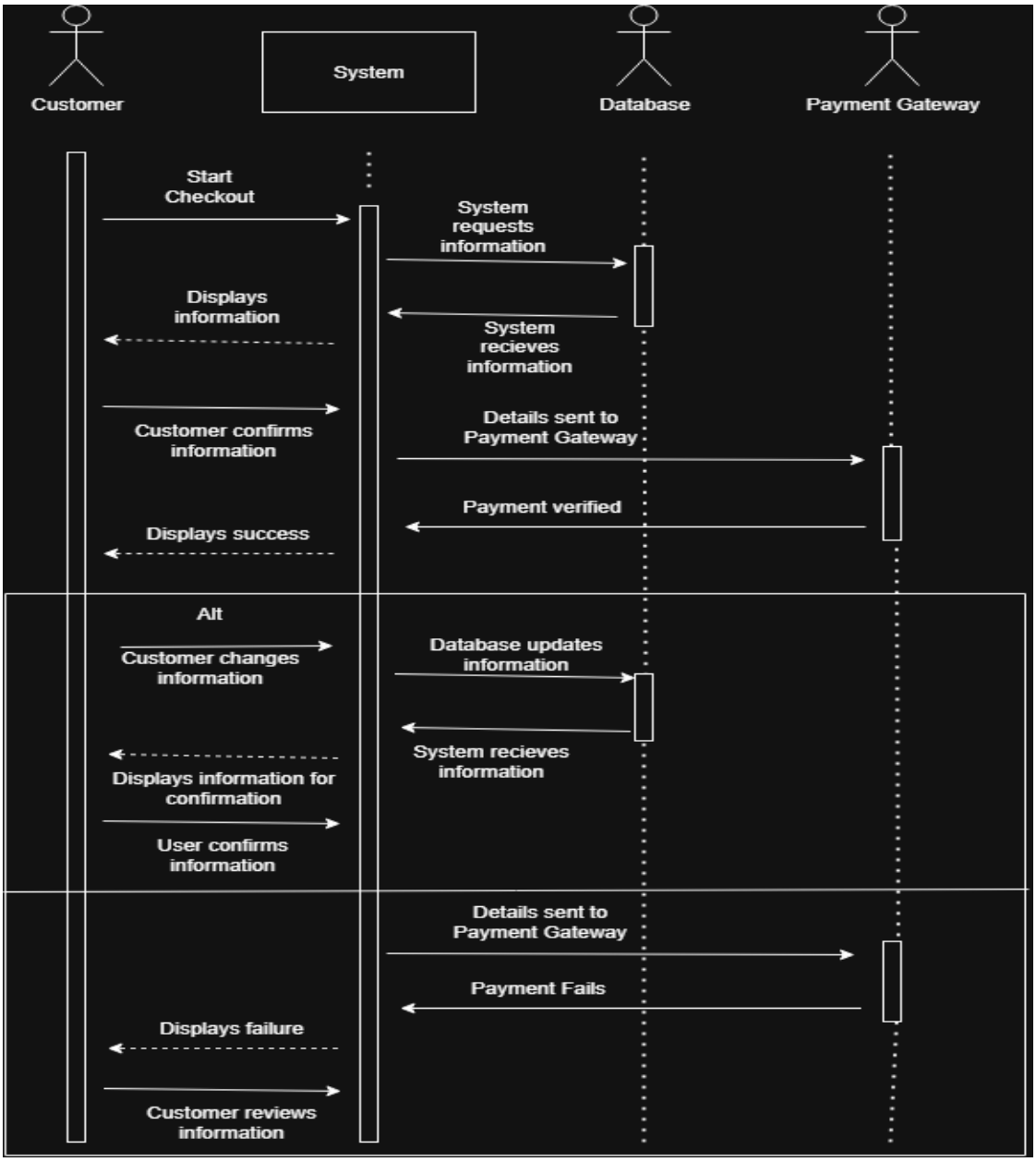
Alternate Flow Steps: Customer changes the information during the review part of the process. If the payment fails, the system will attempt to get the customer to try again or use a different payment option.

3d. System Sequence Diagrams

UC1 Browse Catalog



UC4 Verify and Complete Purchase

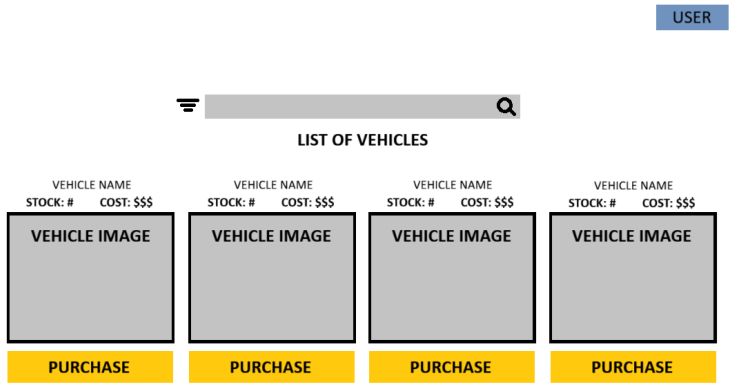

Customer
System
Database
Payment Gateway
Start Checkout
System requests information
Displays information
System recieves information
Customer confirms information
Details sent to Payment Gateway
Payment verified
Displays success
Alt
Customer changes information
Database updates information
System recieves information
Displays information for confirmation
User confirms information
Details sent to Payment Gateway
Payment Fails
Displays failure
Customer reviews information

4. User Interface Specification

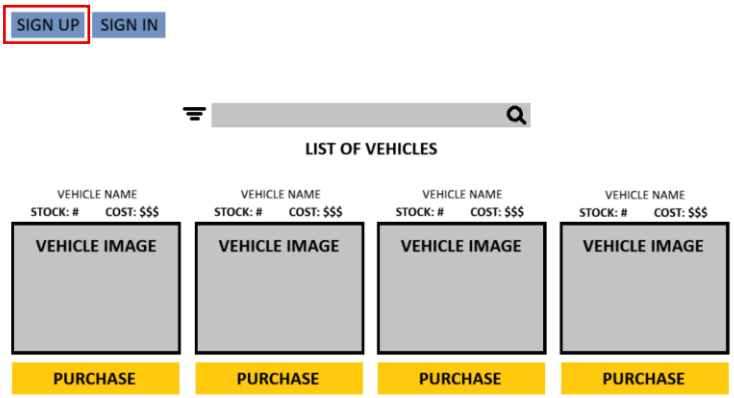
The purpose of this section is to give further details as to how the user will interact with the website to complete whatever task they are doing. Step-by-step descriptions of how the user inputs information and how the results will be displayed will be in this section.

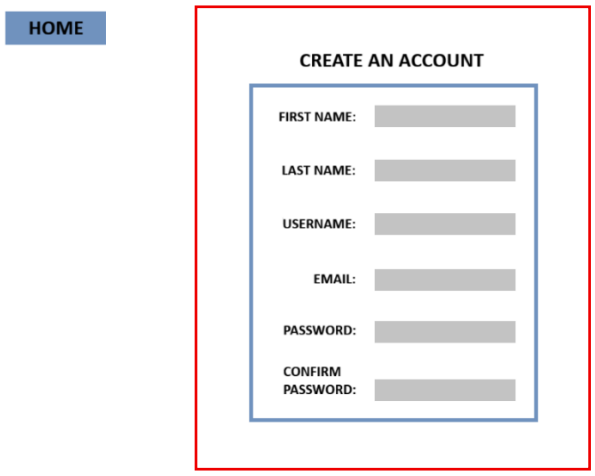
4a. Preliminary Design

UC1: Browse Catalogue

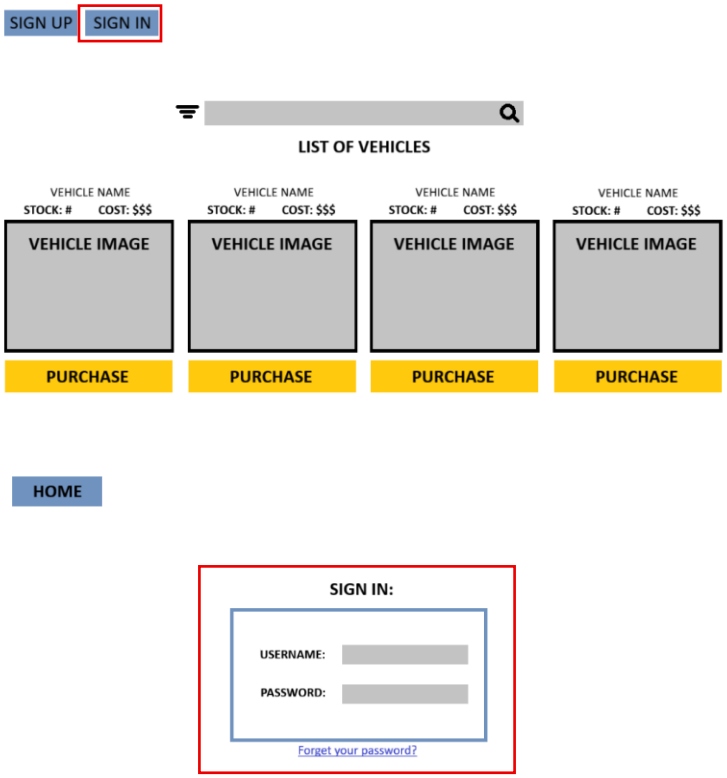


UC2: Create Account and Log in

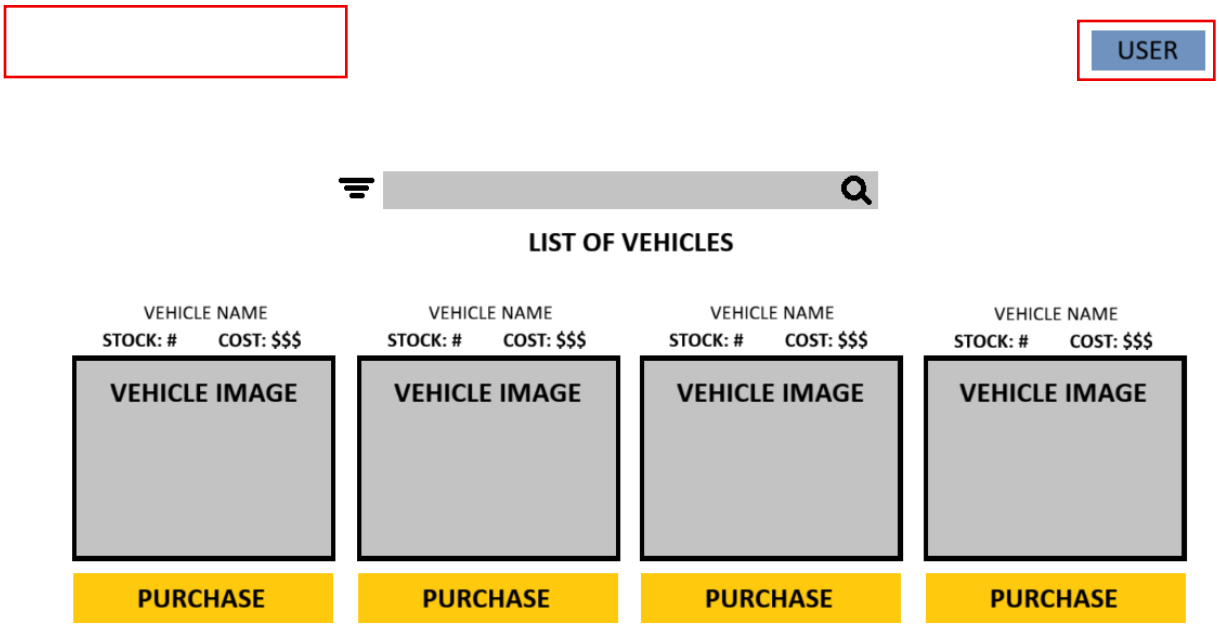




The user will enter their first name, last name, desired username, email, and desired password into the required fields to create their account.

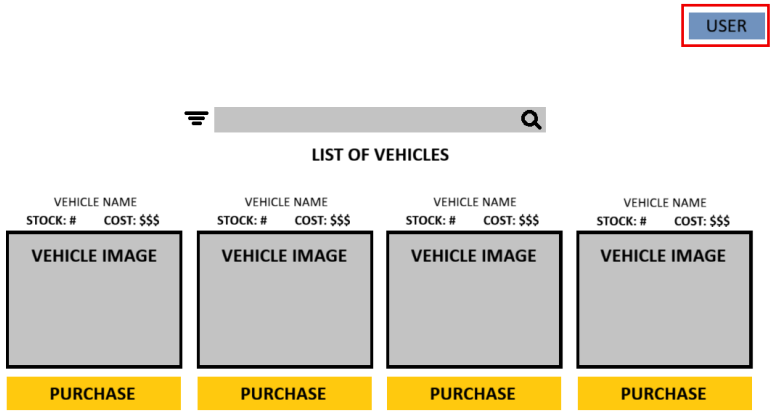


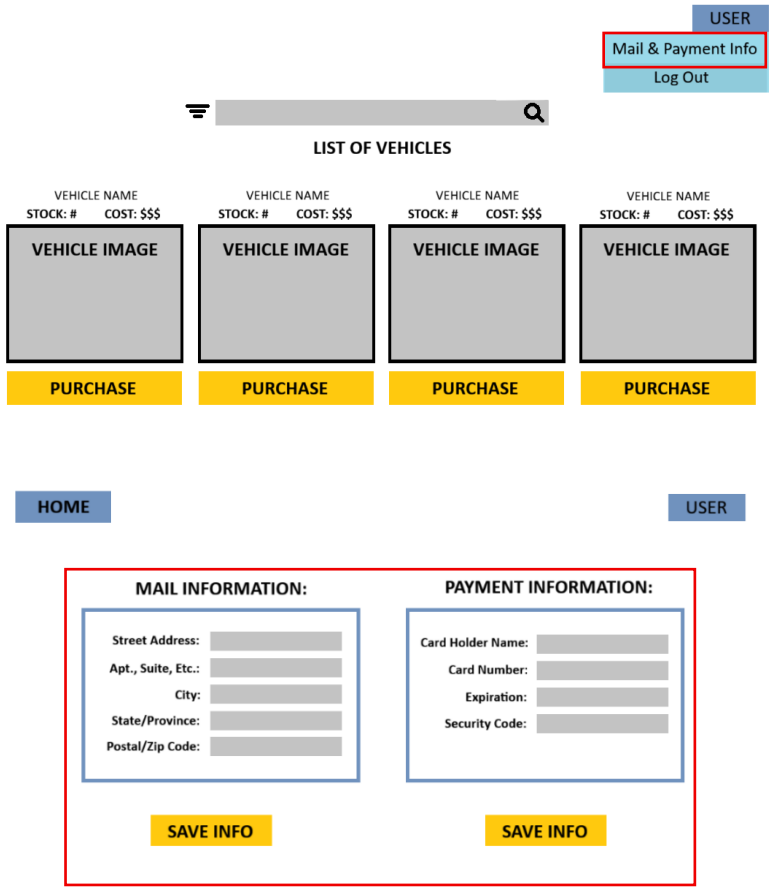
The user will enter their created username and password into the required fields.



After creating an account or signing into the website, the main page will be displayed slightly differently. The “Sign Up” and “Sign in” buttons will no longer be visible, instead showing a “User” button in the upper right-hand corner of the display.

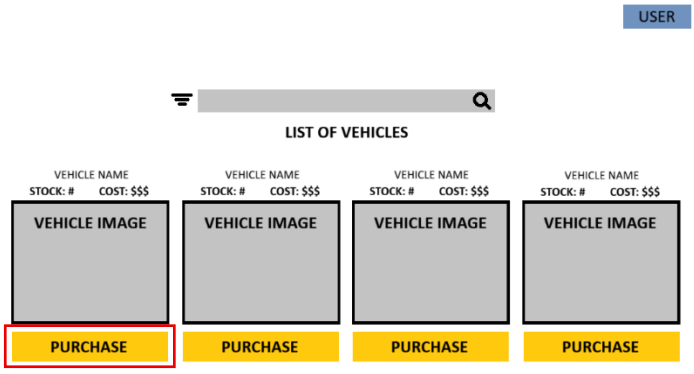
UC3: Manage Purchase Information

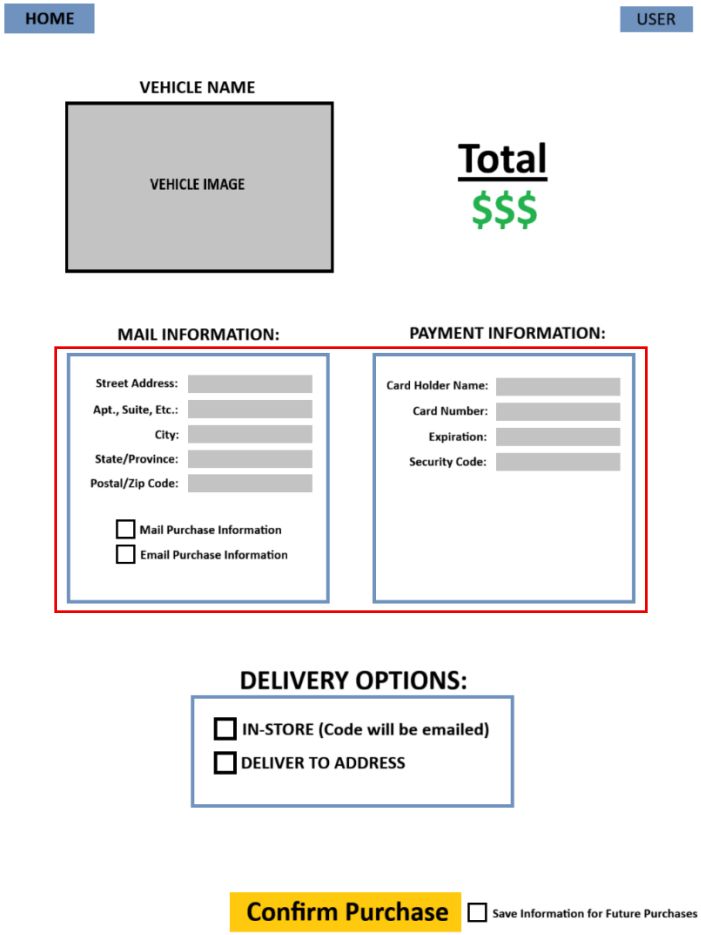




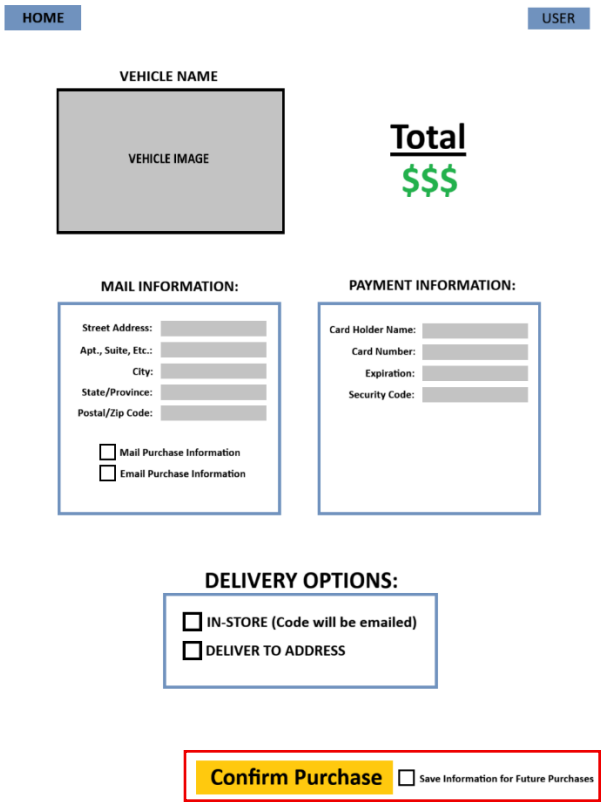
The user will enter their desired information into the selected fields to change their mail/payment information. Then they can click on the “Save info” button to save their changes.

UC4: Verify and Complete Purchase





The user will enter their desired information into the selected fields, so their mail information and payment information is available for the purchase to process.

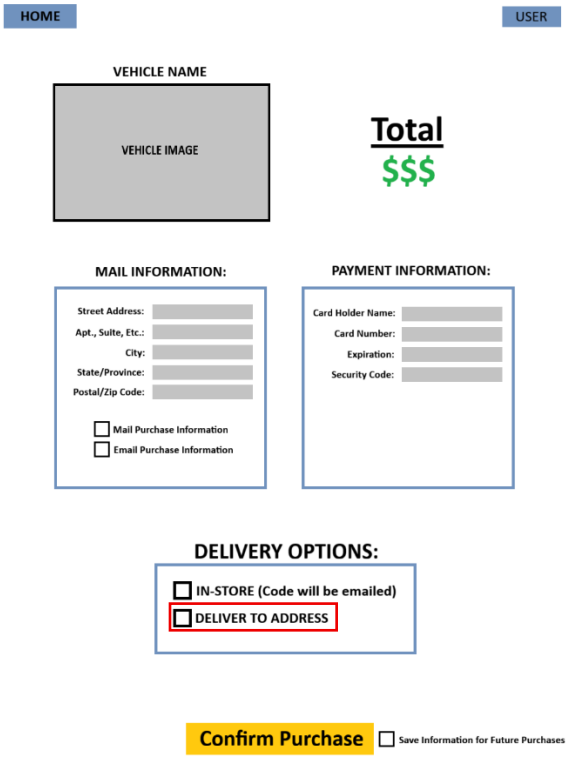


The user will click on the “Confirm Purchase” button to submit the above information to the website for processing. The user will also have the option to “save information for future purchases” buy checking the box with the same name.

UC5: In-Store Collection

The user can click on the box for “In-Store” to mark the purchase for in-store pickup.

UC6: Home Delivery

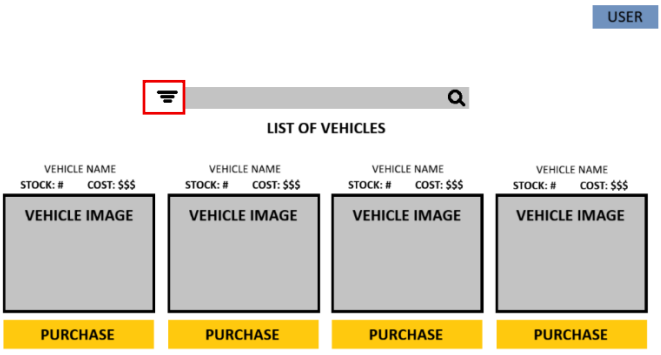


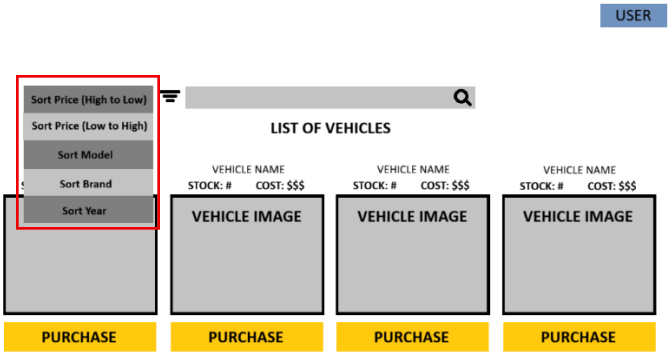
The user can click on the box labeled “Deliver to Address” to mark the purchase for delivery to the entered mailing address above.

UC7: Send Purchase Confirmation

The user can click on the box labeled “Mail Purchase Information” to mark the transaction so that they will be mailed information related to the transaction. The user can also choose to click on the box labeled “Email Purchase Information” to have them be sent an email of the purchase information for the transaction. The user can choose both, either or, or neither.

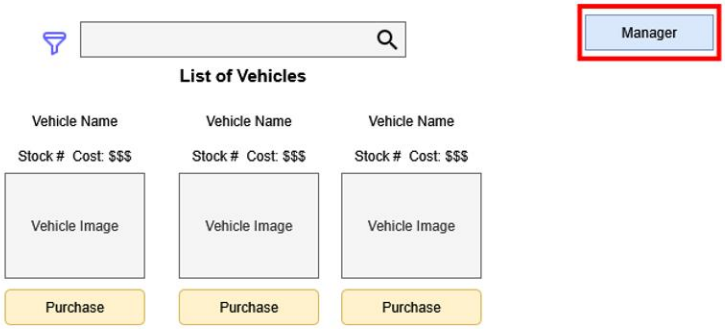
UC8: Search/Filter Vehicles

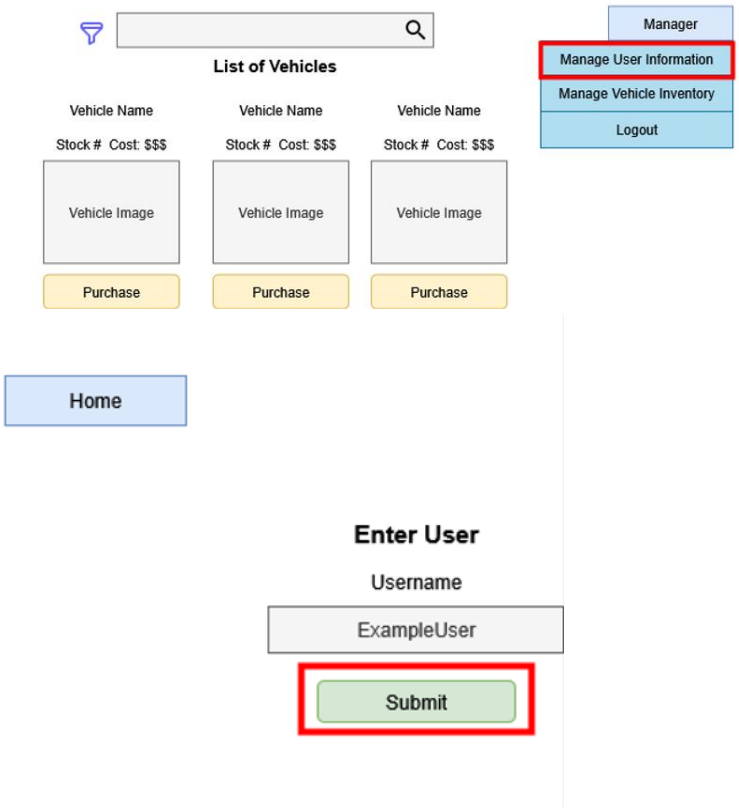




The user can use the drop-down menu to sort the list of vehicles by price, model, brand, year, etc.

UC9: Manage User Information (Admin)

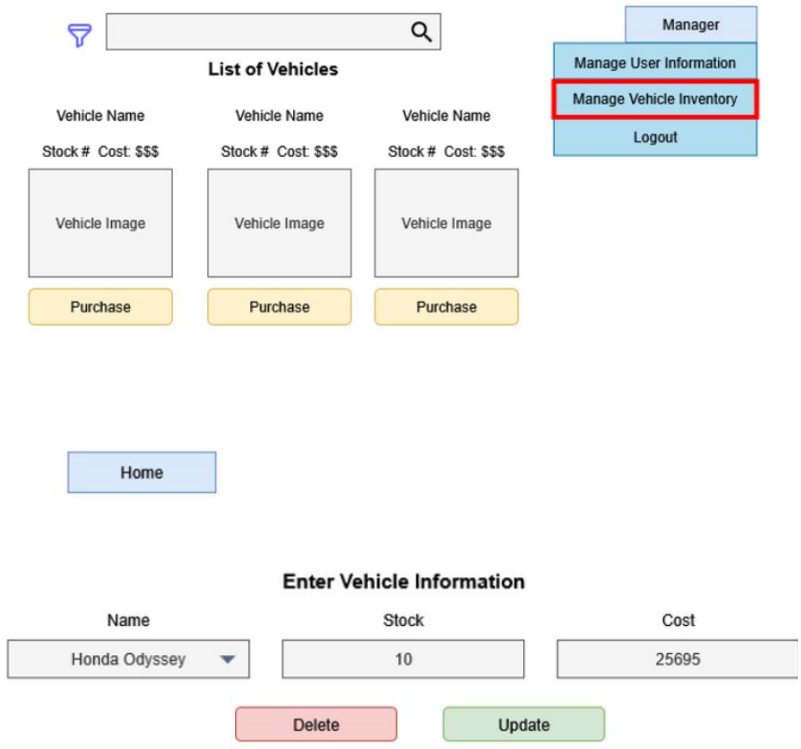




The manager will search for a user based upon their username and press submit

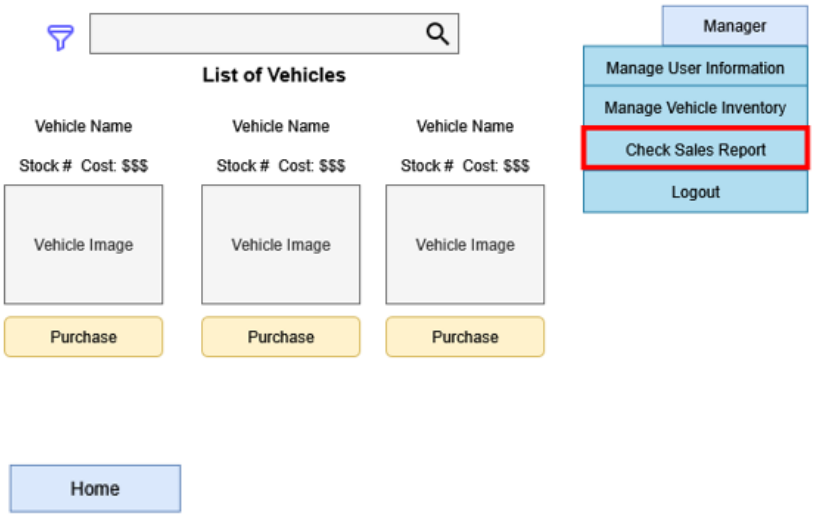
Then, they will update the user's information based on what's requested to be changed

UC10: Manage Vehicle Inventory (Admin)



The manager will enter any new information, and it will change the information based on the name of the vehicle entered. The vehicles can be found through a dropdown, and the dropdown will update based upon what is entered. The manager can also click on the delete button to delete the vehicle. The delete button will need to be pressed a second time after a pop-up warning occurs. The delete button no longer exists.

UC11: Check Sales Report (Admin)



Home

Vehicle	Price	Quantity	Date
Honda Odyssey	26435	5	9/20/2025
Ford F-150	25435	5	9/21/2025
Toyota Prius	23465	5	9/21/2025

Display the most recent purchase information

4b. User Effort Estimation

1. Enter sign up details: The sequence below shows the actions required to sign up as John Smith

   Navigation: 2 clicks

   1.1. Click on the sign-up button on the home page
   1.2. Click the sign-up button after entering information on the sign-up page

   Data Entry: 49 key presses and 8 clicks

   1.3. Click on the entry box and type the first name John
   1.4. Click on the entry box and type the last name Smith
   1.5. Click on the entry box and type the username JSmith
   1.6. Click on the entry box and type the email address JSmith@gmail.com
   1.7. Click on the entry box and type password 1Qar5q@r!
   1.8. Click on the entry box and retype the password 1Qar5q@r!
2. Enter sign in details (only if user is coming back to the website): The sequence below shows the actions required to sign in as John Smith

   Navigation: 2 clicks

   2.1. Click on the sign-in button on the home page
   2.2. Click the sign-in button after entering information on the sign-in page

   Data Entry: 15 key presses and 2 clicks

   2.3. Click on the entry box and type the username JSmith
   2.4. Click on the entry box and type the password 1Qar5q@r!
3. Purchase a vehicle: The sequence below shows the actions required to have a vehicle delivered to 700 College Dr, Hays, KS 67601

   Navigation: 2 clicks

   3.1. Click the purchase button below one of the vehicles on the homepage
   3.2. Click the confirm purchase button on the vehicle purchase page

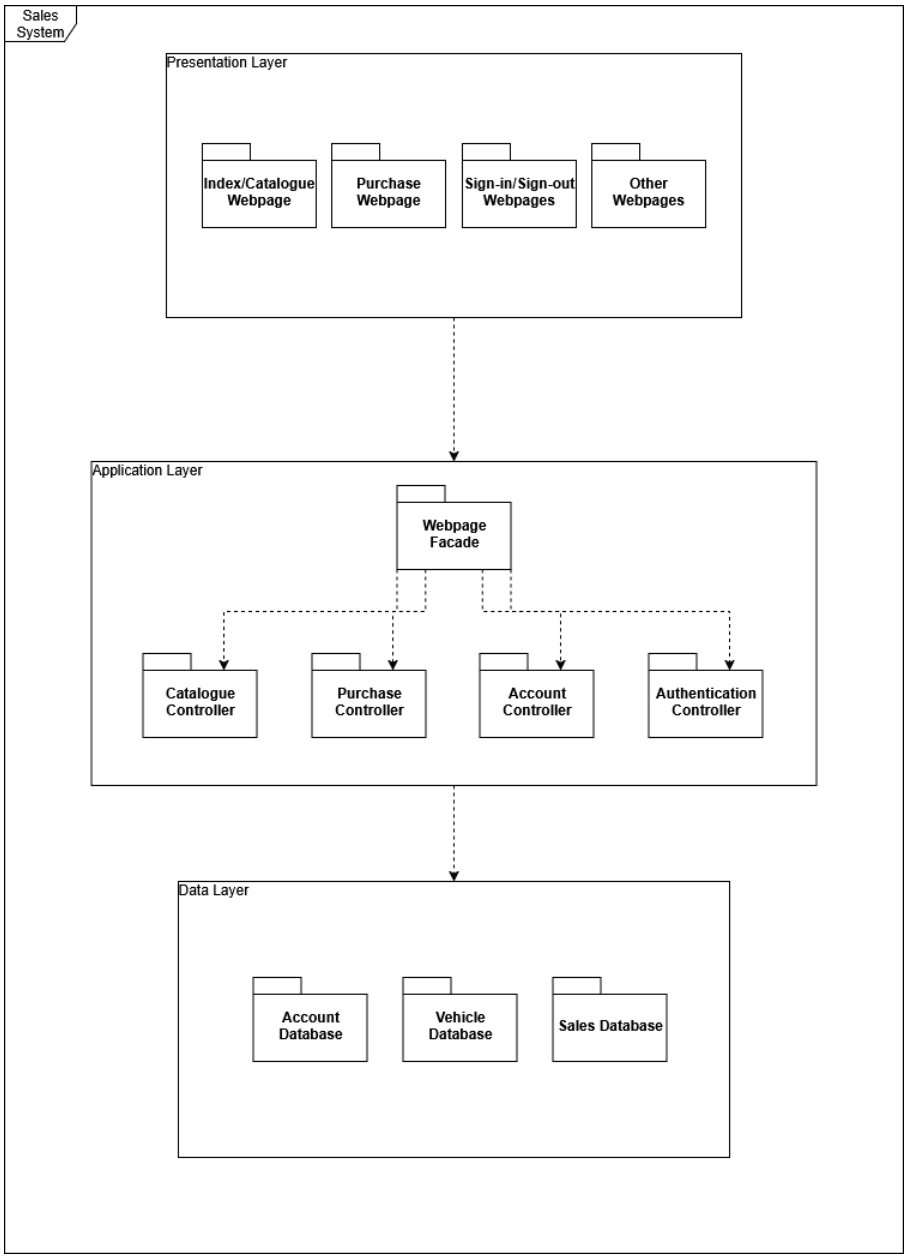
   Data Entry: 58 key presses and 13 clicks

   3.3. Click on the entry box and type the street address 700 College Dr
   3.4. Click on the entry box and type the city Hays
   3.5. Click on the entry box and type the state KS
   3.6. Click on the entry box and type the zip code 67601
   3.7. Click on the entry box and type the card holder name John Smith
   3.8. Click on the entry box and type the card number 5588 9756 5070 3832
   3.9. Click on the entry box and type the expiration date 8/27

3.10. Click the entry box and type the security code 999
3.11. Click on the mail & email purchase information checkboxes
3.12. Click on the delivery option

5. System Architecture and System Design

5a. Identifying Subsystems



This system follows the multi-tiered design pattern, and it consists of a presentation, application, and data subsystems. The subsystems can be described as follows:

Presentation: This subsystem displays the information on a webpage. On each webpage there are various buttons and entry boxes the user can interact with to input data or move between each page.

Application: This subsystem does the computations. Any user input is processed in this subsystem, as soon as a user submits information the correct database is updated accordingly. There is user verification and purchase verification done here, and there are various requests to databases for the verification of the provided information.

Data: This subsystem stores information. Any information provided through the application subsystem is stored here. There are three separate databases for all of the different types of data stored.

5b. Architecture Styles

The multi-tiered architecture style was chosen for this project. The separation of the different layers makes code corrections easier in the long run. The presentation layer has webpages that use code for various tasks including sales reports, user sign-in/sign up, information updates, and purchases. The application layer runs code, and it updates databases based on the information provided. The data layer has databases which store information related to vehicles, users, and sales.

5c. Mapping Subsystems to Hardware

Various Controllers: This subsystem runs on the server, and every computation is done by the server.

Various Webpages: This subsystem runs on the various user's clients, and the webpages are displayed on the users' computers.

Various Databases: This subsystem is stored on the main server computer, and the information is updated based on the information from the various controllers.

5d. Connectors and Network Protocols

HTTPS: This protocol is what the web application will run on. It is used for communicating between the client and server

API Connection: The google mail API will be used for sending automated messages through email. This includes the purchase controller for sending billing information, and the account controller for changes to account information. This will be programmed through Python.

5e. Global Control Flow

Execution orderness: The system uses event-driven architecture. Our website waits for certain events to happen. For example, the user John Smith enters his details to sign in. John could also have picked to purchase a vehicle first, and then he would have entered details for his purchase. Any event could happen in any order, one before the other or one after the other.

Time dependency: The system is an event-response type of system. It does not have to wait for anything to happen, and the user can spend as much time as they want on each page. The processes do not time out.

5f. Hardware Requirements

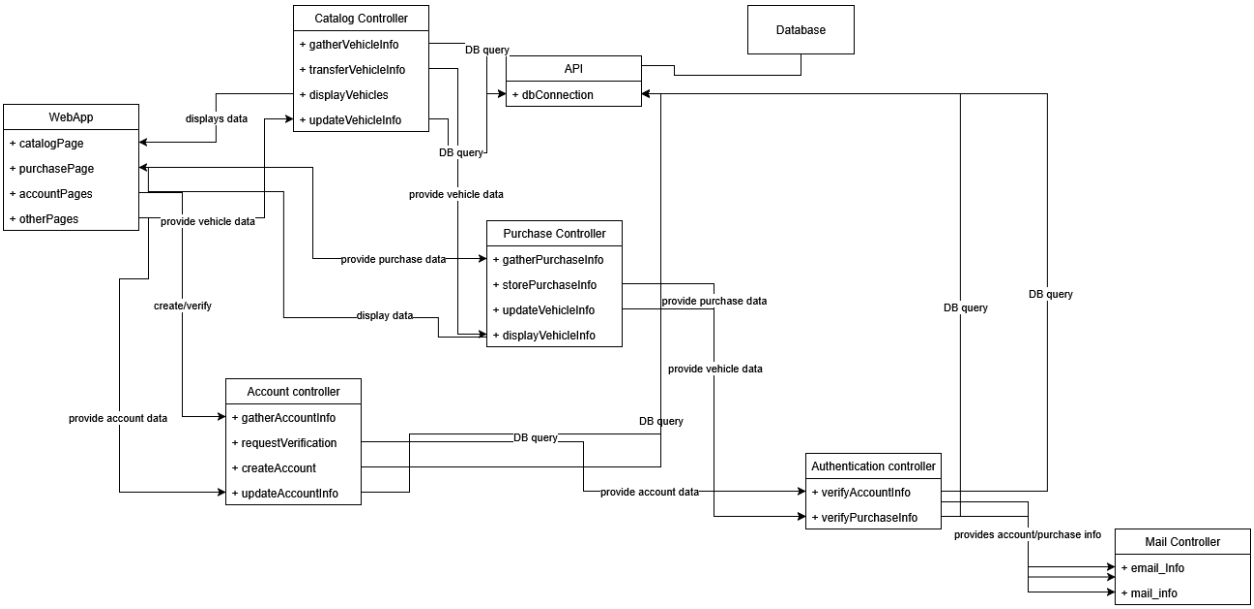
Screen Display: The webpages on our website can display on most laptops, computers, and mobile devices. The layout will change depending on each device and may not work on hardware that cannot run windows 7.

Communication Network: While the website may display with a slow internet connection, it is recommended to have at least an average internet connection (42.86Mbps).

Database Server: Our server will have a database to store all information related to the vehicles and user accounts.

6. Analysis and Domain Modeling

6a. Conceptual Model



i. Concept Definitions

Responsibility Description	Type	Concept Name
Display the various webpages related to purchasing a vehicle, updating database information, or making an account. Also, allow user interaction.	D	WebApp
Provide vehicle information to display on the webpage and allow an administrator to update vehicle information.	D	Catalog Controller
Allow the user to sign in / sign up for an account. Allow an administrator to update account information.	D	Account Controller
Allows the user to purchase a vehicle and save their information.	D	Purchase Controller
Authenticate any logins and vehicle purchases	D	Authentication Controller
Email/mail the user information related to their account and purchases	D	Mail Controller
Securely fetch and store data to the database	D/K	API

ii. Association Definitions

Concept Pair	Association Description	Association Name
WebApp ↔ Catalog Controller	The WebApp display the vehicles in a catalog	displays data
WebApp ↔ Catalog Controller	The WebApp updates vehicle information through the Catalog Controller	provide vehicle data
WebApp ↔ Purchase Controller	The WebApp displays the vehicle being purchased	display data
WebApp ↔ Purchase Controller	The WebApp provides purchase information input by the user to the Purchase Controller	provide purchase data
WebApp ↔ Account Controller	The WebApp provides account information from the sign in / sign up page to the Account Controller	create/verify
WebApp ↔ Account Controller	The WebApp updates account information through the Account Controller	provide account data
Catalog Controller ↔ API	The Catalog Controller sends updates through an API to vehicle information or gathers vehicle information for the WebApp	DB query
Catalog Controller ↔ Purchase Controller	The Catalog Controller sends the information of the vehicle being purchased to the Purchase Controller	provide vehicle data
Account Controller ↔ Authentication Controller	The Account Controller provides sign in data to verify through the Authentication Controller	provide account data
Account Controller ↔ API	The Account Controller provides information on a new account or update an existing account's information through the API	DB query
Purchase Controller ↔ Authentication Controller	The Purchase Controller provides purchase information to be verified through the Authentication Controller	provide purchase data
Purchase Controller ↔ Authentication Controller	The Purchase Controller provides updated vehicle information to be stored through the Authentication Controller after the purchase is verified	provide vehicle data
Authentication Controller ↔ API	The Authentication Controller provides information related to accounts or vehicles to be stored through the API after being verified	DB query
Authentication Controller ↔ Mail Controller	The Authentication Controller provides information related to a verified	provides account/purchase info

purchase or login to the Mail Controller
that emails/mails users

iii. Attribute Definitions

Concept	Attributes	Attribute Description
WebApp	catalogPage	User interface displaying a selection of vehicles
	purchasePage	User interface displaying a specifically chosen vehicle
	accountPages	User interface for signing up/signing in
	otherPages	User interface for changing vehicle/account information
Catalog Controller	gatherVehicleInfo	Gather information on all of the vehicles
	transferVehicleInfo	Give information on a specific vehicle to the Purchase Controller
	displayVehicles	Send information on all of the vehicles to the WebApp
	updateVehicleInfo	Update a vehicle's information in the database
Account Controller	gatherAccountInfo	Take the user's input for the sign in/sign up web pages
	requestVerfication	When the user is signing in request verification of the data provided
	createAccount	Store a new user in the database
	updateAccountInfo	Modify an account's information and store the updated information in the database
Purchase Controller	gatherPurchaseInfo	Take the user's input for credit card information
	storePurchaseInfo	At the user's request, store the credit card information in the database, but verify it first
	updateVehicleInfo	Wait for the purchase information to be verified and update the purchased vehicle's information
	displayVehicleInfo	Display the chosen vehicle's information on the web page
Authentication Controller	verifyAccountInfo	Verify account information through double checking information in the database
	verifyPurchaseInfo	Verify purchase information through double checking information in the database
Mail Controller	email_Info	Email information related to purchases and account updates
	mail_Info	Mail information related to purchases
API	dbConnection	Handle all database requests

iv. Traceability matrix for Domain Concepts

Use Case	PW	Domain Concepts						
		WebApp	Catalog Controller	Account Controller	Purchase Controller	Authentication Controller	Mail Controller	API
UC1	**10**	**x**	**x**					**x**
UC2	**5**			**x**		**x**		**x**
UC3	**5**			**x**				**x**
UC4	**5**					**x**		**x**
UC5	**3**				**x**			
UC6	**3**				**x**			
UC7	**3**						**x**	
UC8	**1**	**x**	**x**					
UC9	**5**			**x**		**x**		**x**
UC10	**5**		**x**			**x**		**x**
UC11	**3**	**x**						**x**

6b. System Operation Contracts

UC1 – Browse Catalog

Operation: retrieveCatalog()

- **Preconditions:**
  - Customer has accessed the system.
  - Database connection is available.
- **Parameters:** None.
- **Return Values:** List of available vehicles (with attributes: make, model, year, price, status).
- **Postconditions:** Vehicle catalog is displayed to the customer.
- **Error Conditions:**
  - No cars available → return empty catalog + "No cars available" message.
  - Database connection fails → return error message prompting retry.
- **Atomicity:** Not required (read-only operation).

UC4 – Verify and Complete Purchase

Operation 1: startCheckout()

- **Preconditions:**
  - Customer is logged in.
  - Customer has selected a vehicle to purchase.
- **Parameters:** Vehicle ID, Customer Session ID.
- **Return Values:** Checkout Session ID.
- **Postconditions:** Checkout process is initialized.
- **Error Conditions:**
  - Vehicle no longer available → system aborts checkout.
- **Atomicity:** Not required.

Operation 2: retrieveCustomerInfo()

- **Preconditions:**
  - Checkout session is active.
- **Parameters:** Customer ID.
- **Return Values:** Customer profile (name, billing address, contact info).
- **Postconditions:** System retrieves and displays customer data for confirmation.
- **Error Conditions:**
  - Database unavailable or corrupted customer data → operation fails.
- **Atomicity:** Not required.

Operation 3: confirmCheckoutDetails()

- **Preconditions:**
  - Customer profile and order details are successfully retrieved.
- **Parameters:** Confirmation flag, updated details (if customer edits info).
- **Return Values:** Boolean (confirmation accepted or rejected).
- **Postconditions:** Order details are locked for payment.
- **Error Conditions:**
  - Missing or invalid confirmation input → order not locked.
- **Atomicity:** Not required.

Operation 4: processPayment()

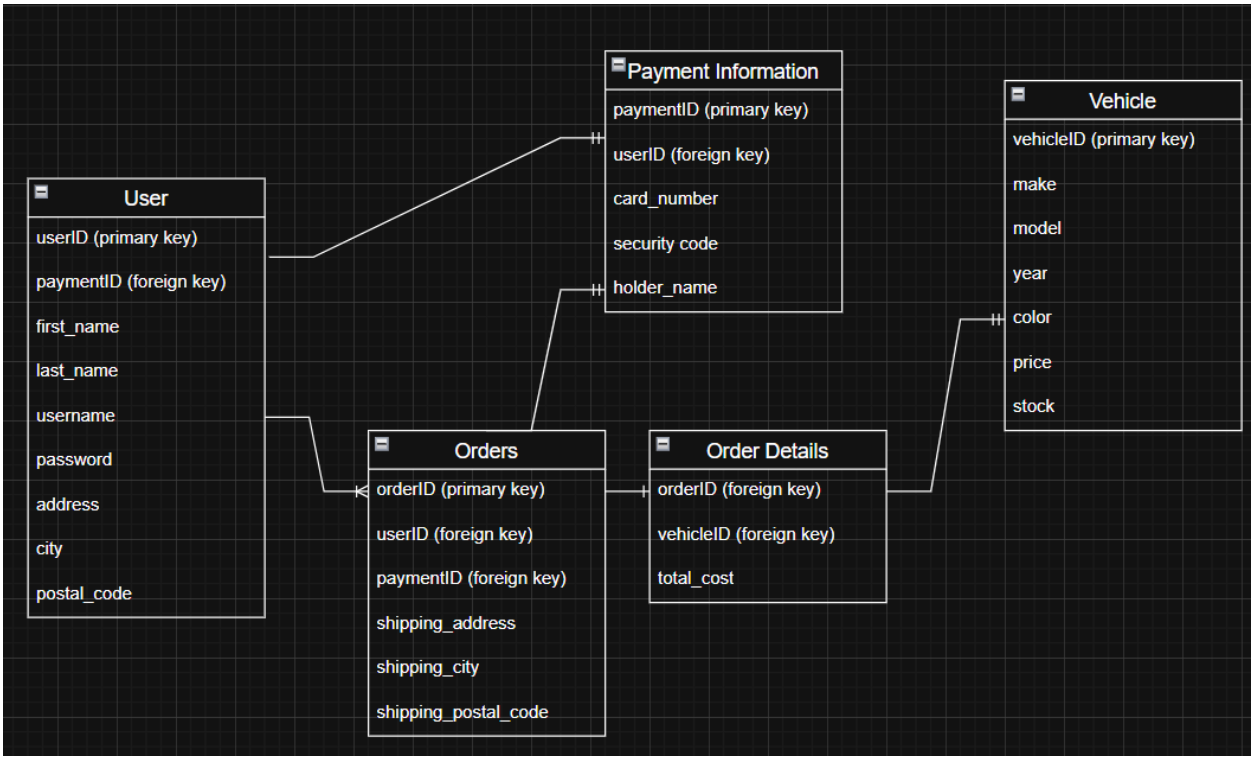
- **Preconditions:**
  - Checkout details are confirmed.
- **Parameters:** Payment method, Amount, Order ID.
- **Return Values:** Payment status (Success/Failure), Transaction ID.
- **Postconditions:**
  - If success → funds reserved and recorded.
  - If failure → system prompts retry or alternate payment option.
- **Error Conditions:**
  - Insufficient funds, invalid payment method, gateway timeout.
- **Atomicity:** Required — payment must be all-or-nothing. If any step fails, no funds are deducted, and order is not processed.

Operation 5: finalizePurchase()

- **Preconditions:**
    - Payment was successfully processed.
- **Parameters:** Order ID, Payment Transaction ID.
- **Return Values:** Order Confirmation ID, Pickup Code.
- **Postconditions:**
    - Order is saved to database.
    - Confirmation and pickup code are issued to customers.
- **Error Conditions:**
    - Database update fails → rollback, order not confirmed.
- **Atomicity:** Required — both order recording and pickup code generation must succeed, or system reverts transaction.

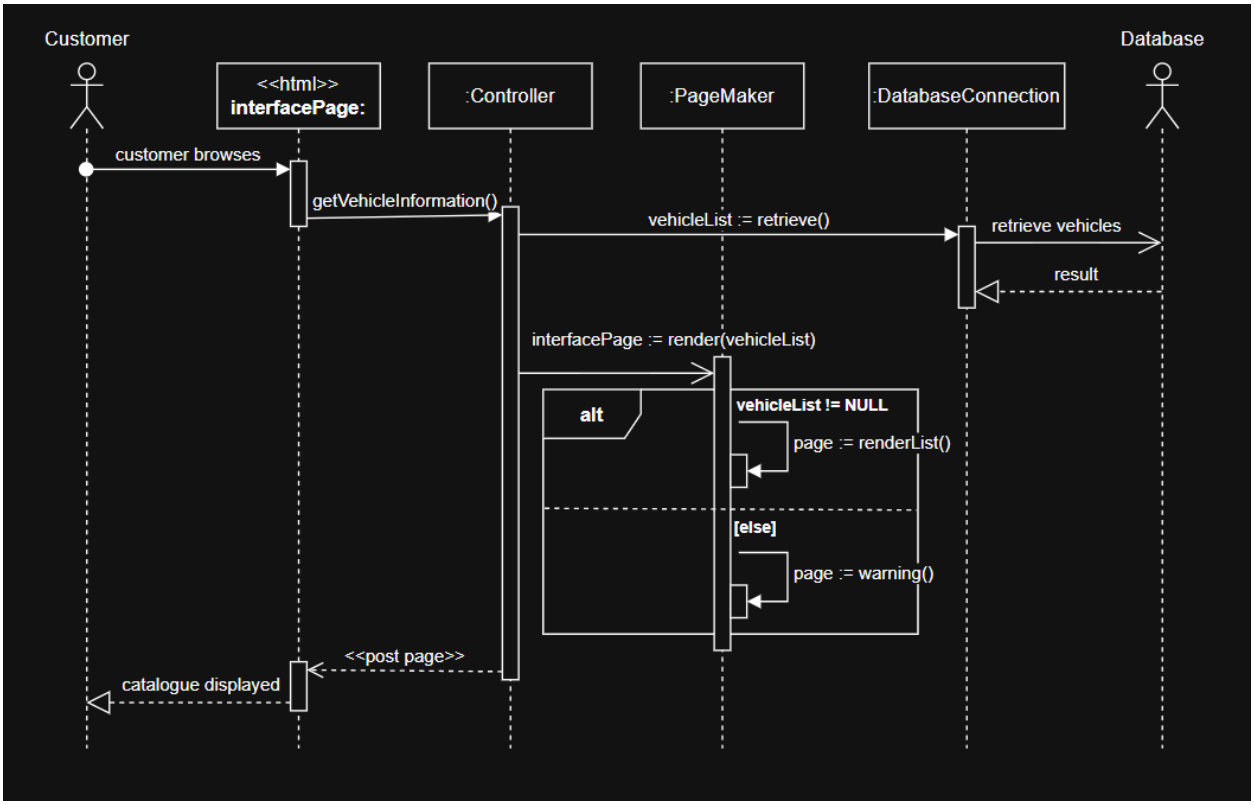
6c. Data Model and Persistent Data Storage

MySQL Database: The webpage will have to store information throughout multiple instances of the webpage running. For the user: login, order, mail, and payment information. For the webpage, vehicle information. The best storage management strategy for this would be a MySQL database.



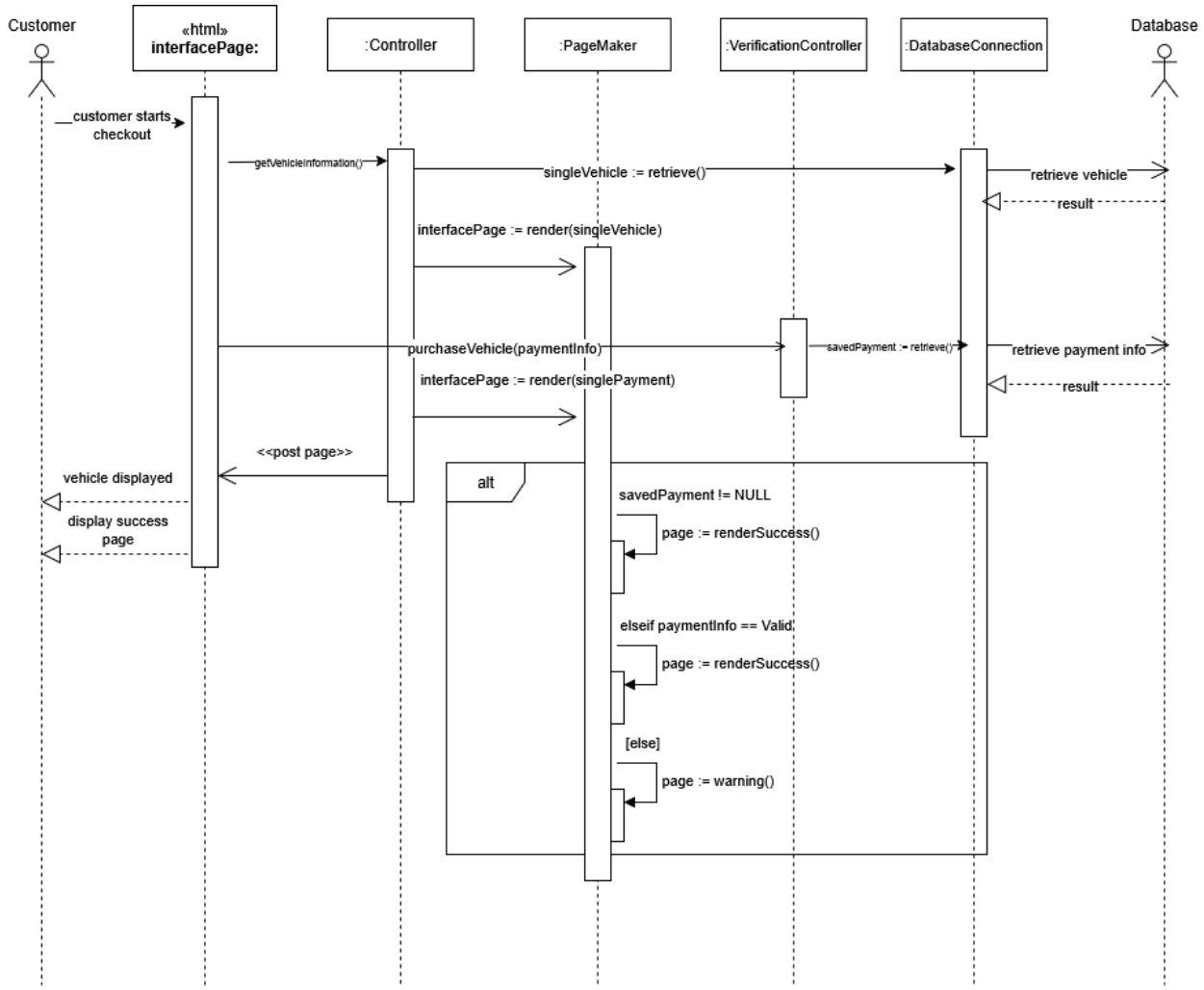
7. Interaction Diagrams

UC1: Browse Catalogue



This diagram makes use of the High Cohesion principle by spreading out the workload across all objects, making sure they're not overloaded DatabaseConnection, PageMaker, Controller, and interfacePage all have about two main tasks, balancing the workload.

UC4: Verify and Complete Purchase



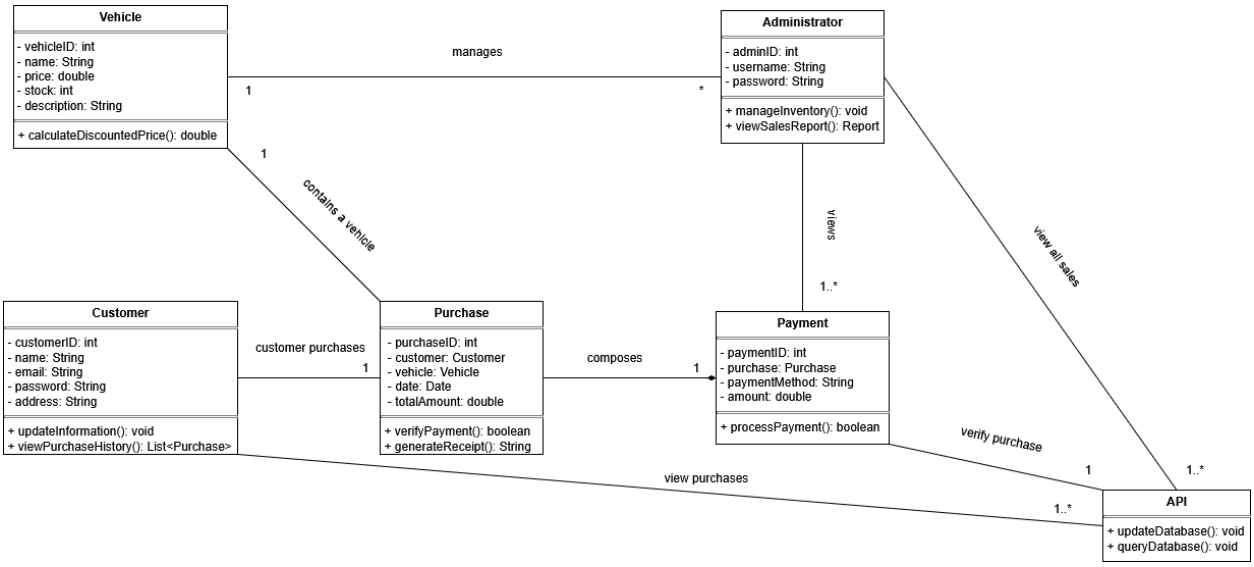
This diagram uses the High Cohesion principle by spreading the workload across all objects, so as to not overwhelm them. There are at most two tasks a singular object has to perform.

Design Patterns:

The design pattern that best fits our implementation is the Proxy pattern, more specifically, the Protection Proxy pattern. Using this pattern if helpful, because we need to restrict access to certain aspects of the website from users who do not have the right credentials. Managing stock and orders, for example, should only be available someone with admin credentials. We also don't want a customer to have the potential of accessing another customer's order history, personal data, etc.

8. Class Diagram and Interface Specification

8a. Class Diagrams



Class: API

Operations:

- updateDatabase(): void — Updates information in the MySQL database upon request.

- queryDatabase(): void — Provides information from the MySQL database from anywhere requested.

Class: Vehicle

Attributes:

- vehicleID: int — Unique identifier for each vehicle.

- name: String — The name/model of the vehicle.

- price: double — The listed price of the vehicle.

- stock: int — The quantity of the vehicle available in inventory.

- description: String — A short description of the vehicle features.

Operations:

- calculateDiscountedPrice(): double — Calculates the vehicle's price after discounts (if applicable).

Meaning: Represents individual vehicles listed in the catalog that customers can view and purchase.

Class: Customer

Attributes:

- customerID: int — Unique identifier for the customer.

- name: String — Customer's full name.

- email: String — Customer's email address.

- password: String — Encrypted password for authentication.

- address: String — Customer's delivery or mailing address.

Operations:

- updateInformation(): void — Updates the customer's account details.

- viewPurchaseHistory(): List<Purchase> — Displays past transactions for the customer.

Meaning: Represents the system's users who purchase vehicles and manage their accounts.

Class: Purchase

Attributes:

- purchaseID: int — Unique identifier for the purchase.

- customer: Customer — Reference to the customer who made the purchase.

- vehicle: Vehicle — Reference to the vehicle purchased.

- date: Date — Date of purchase.

- totalAmount: double — Total cost of the transaction.

Operations:

- verifyPayment(): boolean — Confirms payment validity.

- generateReceipt(): String — Generates purchase confirmation.

Meaning: Represents a transaction between a customer and the system involving one or more vehicles.

Class: Payment

Attributes:

- paymentID: int — Unique identifier for each payment.

- purchase: Purchase — The related purchase.

- paymentMethod: String — Method used (credit card, PayPal, etc.).

- amount: double — Payment amount.

Operations:

- processPayment(): boolean — Validates and processes the transaction.

Meaning: Handles transaction details related to vehicle purchases.

Class: Administrator

Attributes:

- adminID: int — Unique identifier for the administrator.

- username: String — Administrator's login username.

- password: String — Encrypted password.

Operations:

- manageInventory(): void — Adds, removes, or updates vehicles.

- viewSalesReport(): Report — Displays a summary of all sales.

Meaning: Represents the website administrators responsible for managing data and sales.

8b. Data Types and Operation Signatures

Class	Operation	Return Type	Description
API	updateDatabase()	void	Updates information in the MySQL database upon request.
API	queryDatabase()	void	Provides information from the MySQL database from anywhere requested.
Vehicle	calculateDiscountedPrice()	double	Returns the vehicle price after applying discounts.
Customer	updateInformation()	void	Updates the customer's personal details.
Customer	viewPurchaseHistory()	List<Purchase>	Lists all previous purchases.
Purchase	verifyPayment()	boolean	Checks if payment is valid.
Purchase	generateReceipt()	String	Produces purchase confirmation text.
Payment	processPayment()	boolean	Processes the customer's payment.
Administrator	manageInventory()	void	Updates inventory details.

Administrator	viewSalesReport()	Report	Generates a sales report for management.

8c. Traceability matrix

Software Classes	Domain Concepts						
	WebApp	Catalog Controller	Account Controller	Purchase Controller	Authentication Controller	Mail Controller	API
Vehicle		**x**					
Customer	**x**		**x**				
Purchase				**x**		**x**	
Payment					**x**		
Administrator	**x**	**x**	**x**				
API							**x**

The corresponding domain concepts were ultimately implemented as non-class components where necessary (e.g., reporting functions integrated within the Administrator class rather than standalone).

Explanation of Evolution from Domain Concepts to Classes

Our team followed a modular object-oriented architecture, ensuring high cohesion and low coupling. Each domain concept from the analysis model was translated into one or more software classes, emphasizing responsibility separation and real-world object representation. Below is a detailed mapping and explanation for each concept.

Vehicle:

- Vehicle: Represents the catalog of cars, including name, price, stock, and description.
  This class originated directly from the "Vehicle" domain concept and supports additional system logic such as calculating discounts.

Customer:

- Customer: Represents system users who can browse vehicles, make purchases, and manage their profiles.
  The concept was kept as a single class, but attributes like password and address were added for account management.

Purchase:

- Purchase: Manages purchase details, such as the selected vehicle, date, and total amount.
- Payment: Created as a separate class to handle payment logic independently of the order structure, improving modularity.

Originally, “Purchase” included both logical and financial data; these were split for clarity and scalability.

Authentication Controller:

- Payment: Represents the implementation of the “Authentication Controller” concept, responsible for verifying and processing financial transactions securely.

Administrator:

- Administrator: Originated from the administrative control concept, handling inventory management and report generation.
  Combined management and analytics features were integrated into this class rather than making multiple administrative sub-classes.

Account Controller:

- Customer: Handles user account creation, authentication, and profile editing.
- Administrator: Represents administrative-level accounts with access to management tools.
  The “Account Controller” domain concept was split into these two roles to reflect user-type separation and access control.

Catalog Controller:

- Vehicle: The catalog controller concept was merged into the Vehicle class, as each vehicle instance represents an item in the catalog displayed to users.

WebApp:

- Administrator: The reporting functionality was embedded as viewSalesReport().
  This displays a report of vehicle sales and details related to the vehicle itself, and the time it was purchased.
- Customer: There is a functionality to view previous purchases viewPurchaseHistory().
  This displays a list of vehicles that were previously purchased by the Customer.

Purchase Controller:

- Purchase: This provides all of the information necessary for completing a purchase (like the purchase controller) and passes it onto the payment class for verification

Mail Controller:

- Purchase: This provides all of the information necessary for emailing the customer their receipt generated with the assistance of generateReceipt().

API:

- API: implemented as its own class for any interactions with the payment processes and account processes. These processes change information in the MySQL database when interacting with the API.

8d. Design Patterns

As mentioned in the Interaction Diagrams section, there was the Protection Proxy pattern used in the web application restricting users without the correct credentials. The classes Purchase and Customer have restricted webpages until the user signs in. The class Adminstrator has restricted webpages to with access only given to administrative accounts.

9. Algorithms and Data Structures

9a. Algorithms

The Luhn algorithm will be used for checking the validity of payment information. The steps for doing the algorithm are below:

1. Drop the check digit from the number (if it's already present). This leaves the payload.
2. Start with the payload digits. Moving from right to left, double every second digit, starting from the last digit. If doubling a digit results in a value > 9, subtract 9 from it (or sum its digits).
3. Sum all the resulting digits (including the ones that were not doubled).
4. The check digit is calculated by $(10 - (s \bmod 10)) \bmod 10$, where s is the sum from step 3. This is the smallest number (possibly zero) that must be added to s to make a multiple of 10. Other valid formulas giving the same value are $9 - ((s + 9) \, mod \, 10)$, $(10 - s) \bmod 10$, and $10\lceil s / 10 \rceil - s$. Note that the formula $(10 - s) \bmod 10$ will not work in all environments due to differences in how negative numbers are handled by the modulo operation.

9b. Data Structures

Arrays are used for flexibility when manipulating data related to vehicles or accounts.

10. User Interface Design and Implementation

No adjustments have been made to the initial design of the user interface for the webpage. It is currently in the process of being implemented and it is not yet fully complete. The initial design focused on “Ease-of-use”, avoiding excessive user navigation in any use cases. Most tasks on the webpage can be completed in less than 5 clicks.

11. Test Designs

11a. Test Cases

Test Case Identifier: TC-1

Use Case Tested: UC-1

Pass/Fail Criteria: This test succeeds if all of the vehicles are loaded in the catalog correctly. The test fails if the database fails to request the vehicles' information, or the vehicles fail to load correctly.

Input Data: database request

Test Procedure	Expected Results
The database of existing information for vehicles is requested.	The vehicles all load in the correct order on the webpage.
The database of existing information for vehicles is requested and collected in the wrong order.	The vehicles all load but in the incorrect order.
A database of non-existent information for vehicles is requested.	The vehicles fail to load.

Test Case Identifier: TC-2

Use Case Tested: UC-2

Pass/Fail Criteria: This test succeeds if an account is successfully created and the user logs in. This test fails if the account is not put into the database and the user is unable to login.

Input Data: form data

Test Procedure	Expected Results
The user registers an account, then they login to their account.	The user is able to successfully login.
The user registers an account with invalid characters, then they attempt to login to their account.	The user is warned about invalid characters. They ignore the warning and unsuccessfully attempt to login.

Test Case Identifier: TC-3

Use Case Tested: UC-3

Pass/Fail Criteria: This test succeeds if the user updates their purchase information. This test fails if the purchase information stays the same.

Input Data: form data

Test Procedure	Expected Results
The user provides valid address, contact, and payment information.	The user is able to use their updated information on the purchase screen.
The user provides invalid address, contact, and payment information. This information either has invalid characters or invalid formatting for payment or contact information.	The user is warned about their information being invalid.

Test Case Identifier: TC-4

Use Case Tested: UC-4, UC-5, UC-6, UC-7

Pass/Fail Criteria: This test succeeds if the user successfully purchases a vehicle and all of the information they requested is received. This test fails if the user doesn't receive request information or they are unable to purchase a vehicle.

Input Data: form data

Test Procedure	Expected Results
The user provides valid address, contact, and payment information. They request delivery and to have their receipt mailed and emailed information.	The user goes to the success screen. They receive their receipt in the mail and in their email.
The user provides valid address, contact, and payment information. They request delivery.	The user goes to the success screen.
The user provides valid address, contact, and payment information. They request to pick up their car and to have their receipt mailed and emailed information.	The user goes to the success screen and is given a pickup code. They receive their receipt in the mail and in their email.
The user provides valid address, contact, and payment information. They request to pick up their car.	The user goes to the success screen and is given a pickup code.
The user provides invalid address, contact, and payment information. This information either has invalid characters or invalid formatting for payment or contact information.	The user is warned about their information being invalid.

Test Case Identifier: TC-5

Use Case Tested: UC-8

Pass/Fail Criteria: This test succeeds if the user is able to successfully filter the cars based upon criteria. This test fails if some of the cars remain after being filtered out.

Input Data: filter data

Test Procedure	Expected Results
The user filters cars based on price, make, or model.	Vehicles are filtered out if they do not fit the criteria chosen in the filter.
The user incorrectly filters cars based on price, make, or model.	Vehicles remain even though they are meant to be filtered out if they do not fit the criteria chosen in the filter.

Test Case Identifier: TC-6

Use Case Tested: UC-9

Pass/Fail Criteria: This test succeeds if an admin is able to successfully update a user's information. This test fails if the admin is prevented from updating a user's information.

Input Data: form data

Test Procedure	Expected Results
The admin tries to update the information associated with a user's account.	The admin is able to update a user's information and is given a success screen.
The admin tries to update the information associated with a user's account, but they enter invalid characters.	The admin is warned about their information being invalid.

Test Case Identifier: TC-7

Use Case Tested: UC-10

Pass/Fail Criteria: This test succeeds if an admin is able to successfully update a vehicle's information. This test fails if the admin is prevented from updating a vehicle's information.

Input Data: form data

Test Procedure	Expected Results
The admin tries to update the information associated with a vehicle's price or stock.	The admin is able to update a vehicle's information and is given a success screen.
The admin tries to update the information associated with a vehicle's price or stock, but they enter invalid characters.	The admin is warned about their information being invalid.

Test Case Identifier: TC-8

Use Case Tested: UC-11

Pass/Fail Criteria: This test succeeds if an admin is able to successfully see all of the vehicles purchased. This test fails if none of the vehicles are displayed.

Input Data: database request	
Test Procedure	**Expected Results**
The database of existing information for purchases is requested.	The purchase information is loaded on the webpage.
A database of non-existent information for purchases is requested.	The purchase information fails to be loaded.

11b. Test Coverage

The test cases cover the most critical functions for each class, and some convenient functionality like filtering the vehicle catalog. These tests will allow for less errors to occur when a user uses the website. The unit tests will test what is described in the test cases above. The integration tests will ensure that the separate webpages all work together. Then, system testing will ensure that the non-functional and user interface requirements were tested or implemented correctly during previous testing, like integration testing. The code for this testing will be written in Python using the Pytest library.

11c. Integration Testing

Our integration testing will be performed with the approach of continuous integration testing. This will allow problems to be corrected as they appear in the code. The integration testing will test separate units that are related to each other. For example, this integration testing will ensure that a user can login to the website and purchase a vehicle. This checks two separate test cases that would have been tested separately during unit testing previously and instead test them together. Along with this specific example of integration testing, other related test cases will be tested together.

11d. System Testing

During system testing, there will be final checks for various device resolutions and a low number of interactions when purchasing a vehicle. The various device resolutions will be tested through checking the individual webpages using the responsive design mode found in the Firefox browser's inspector. The responsive design mode allows the developer to choose different resolutions corresponding to different devices. This will show how different devices will display the website. The low number of interactions will be tested by logging into an account and purchasing a website. The loading of the website will be tested by loading up the main webpage. The other non-functional requirements would have already been tested.

Project Management and History of Work

a. Merging the Contributions from Individual Team Members

Every week team members claim sections of the report. Every week the responsibility for working on the report alternates between two halves of the team members. The team leader, Wyatt, reviews the formatting of the report and information on the report and submits it when it is complete.

b. Project Coordination and Progress Report

The MySQL database has been created and implemented into the catalog. Our project can display a catalog of vehicles after correctly querying the database. A vehicle can be purchased. The login page is what is going to be handled next. The catalog and purchase page has been worked on by Wyatt.

c. History of Work

So far, we have implemented all the technical functions of the website. The user can sign/log in to the website and "purchase" their selected vehicle and manage their information. The admin can get access to stock and order information as well. All that is primarily left is CSS front-end implementation. Python is still being used to send emails to users, website hosting, and database querying. It is also still planned for the website to be locally hosted from the team leader's computer. In reference to the Gantt chart, the work plan was followed relatively well. There have been weeks where not much progress was made in comparison to others, but we are still on track for the website to be completely ready in time for Demo #2.

Key Accomplishments

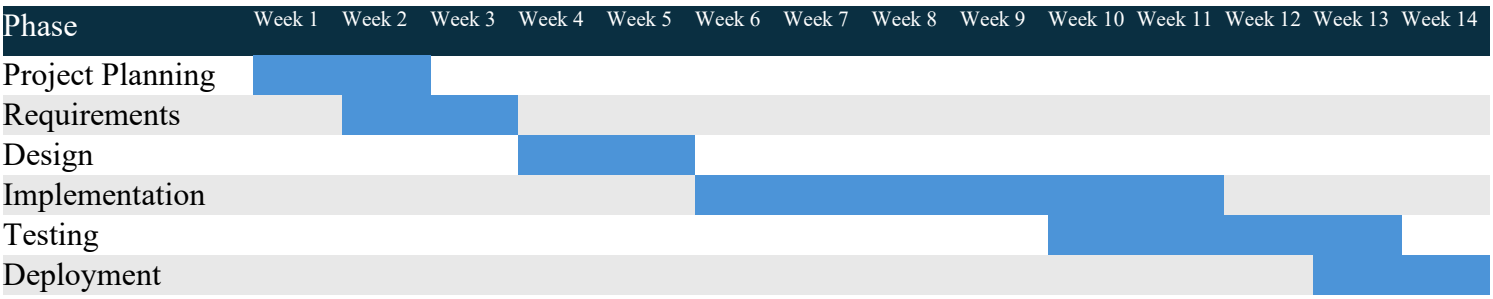
- Admin able to sign in and sign out of website
- Admin able to manage user information and vehicle inventory
- Admin able to check sales reports
- User able to sign in and sign out of website
- User able to browse website
- User able to search/filter through catalogue
- User able to input mail and payment information
- User able to view/modify their own mail and payment information
- User able to save information for future purchases
- CSS design on sign in and sign out pages
- CSS design on home page

Languages: HTML, CSS, JavaScript, Python, SQL

Platforms/Tools: MySQL, GitHub [1], Filess.io[3], Pytest[4]

Integrations: email [2]

Our initial work plan using the Gantt chart below:



d. Breakdown of Responsibilities

Work Item	Planned Team Member
Email/Mail	Wyatt
Catalog	Wyatt, Moussa, Mamadou
Purchase	Wyatt, Moussa, Mamadou
Login/Logout	Wyatt, Moussa, Mamadou
Security	Mamadou
Displays for various webpages	Wyatt, Colin, William

e. Future Work

All of the requirements specified in this report have been met. Features that could be implemented in the future include implementing actual charges to a person's debit card or other payment methods, automatic mailing of information to an actual address, and the ability to add new vehicles and delete them with images.

References

1. GitHub, version control and project planning. https://docs.github.com/en
2. Gmail, email provider. https://developers.google.com/gmail/api/guides/sending
3. Filess.io, host for databases. https://filess.io/
4. Pytest, software for testing. https://docs.pytest.org/en/stable/