Report 3

Course:

CSCI 441 VA Software Engineering

Project Title:

A Web Based Application for Vehicle Sales, Purchase and Inventory Management

URL of Project Website:

https://web-app-for-vehicle-sales-dev.herokuapp.com/

Submission Date:

04/12/2021

Team-Members:

Brady Neumann
Fatih Gurbuz
Taylor Cook
Seth Whitaker

Breakdown of Contributions

All team members contributed equally. As such, each member has typed their own name below.

Taylor Cook
Brady Neumann
Fatih Gurbuz
Seth Whitaker

Glossary of Terms

React: React.js is a front-end framework used to develop and design components.

Node: Node.js is a back-end runtime environment used for developing server-side and networking applications.

Express: Express.js is a web application framework based in node.js.

Heroku: A hosting service used to deploy and manage the web application.

Table of Contents

Summary of Changes

Added Glossary of Terms
Combined Decomposition of Sub-problems and Business Goals
Complete rework of System Architecture

Customer Problem Statement

Car dealerships have a continually growing problem trying to keep up with today's technology. Nowadays, "70% of shoppers expected to find the ability to configure a payment on a dealership website, and 83% indicated that online buying technology would help them narrow down their vehicle choice and determine what is affordable" (V12, no date). Furthermore, with the prevalence of new car-buying websites such as Carvana, car shoppers do not need to set foot on a car lot in order to research, compare, and ultimately buy a car.

From the perspective of the consumer, one of the most prevalent problems with any type of shopping is being able to compare products. This problem exists not only for vehicles but for all electronic-styled purchases. The wide array of choices available, combined with limited availability for comparison, especially between brands, leaves the consumer feeling overwhelmed. Vehicles are one of the most problematic when it comes to this overwhelming feeling. Choices seem endless, comparing vehicles is challenging due to large amounts of variables, and the costs are immensely high. These types of issues are what has allowed online-only car selling to boom in the last couple of years.

Every dealership is encountering these sorts of problems in one way or another; some may have already solved the problem, but others are struggling. This is where a product needs to fill the gap. Dealers need a solid online presence where customers feel that they can shop without buyer's remorse. Whether this comes in the form of searching for vehicles then test driving them at the dealership, or purchasing completely online, the customer needs to be able to dictate their own buying experience.

The customer should be able to sort through vehicles based on multiple different criteria (price, brand, consumer rating, make, model, color, etc.). This ability for comparison will allow the customer to feel confident in the car they are choosing. This, coupled with a very easy to navigate site, makes the car-buying experience a one-stop-shop.

The dealers themselves face data problems. Keeping track of sales data as well as inventory management is a challenging task. If done incorrectly, a misplaced file, or mistyped keystroke can lead to wildly inaccurate information. The application needs to keep track of all of this data for the dealer. Historic sales data should be kept safe for reference whenever the dealer wants to access it, and only admin login credentials should be able to edit these data points as needed. Employee logins should allow use of the data without the ability to create, edit, or delete, and customers or users who have not logged in will not be able to access anything. Inventory management needs to be in a similar state. Admins can add, delete, or edit prices or availability, while employees and customers will only be able to see the data. Another major feature needs to be the ability for the admin to edit the data through an interface rather than requiring any coding. Thus, someone with no technical knowledge at all will still be able to oversee the site. In short, the benefit of all of this is automating a series of tasks so that it removes the human element as much as possible. The security should allow for the database to be protected from anyone other than authorized users, and historic sales data will be kept automatically.

Another issue facing dealers is the cost of the website. Websites are often expensive and getting up off the ground usually has large one-time costs associated with it. The ability to scale is key, as it is incredibly important that should the site need to grow, it does not need to be remade. Upkeep requirements should be low, as no one on staff will know how to fix problems that arise.

Instead, everything should be able to be managed by someone with no technical ability interacting with various interfaces. In the future, additional features may need to be added.

Security will also need to be one of the primary concerns. If unauthorized people are able to change prices or listings, it could cost enough money to put a dealership out of business. Similarly, a normal employee should not be able to manipulate data as this is both a security and error-proofing requirement. An admin, or multiple admins, should be able to log into the site and make changes, but no one else should have access to any sensitive information. No price changes, inventory adjustments, or any editing should be able to be performed by anyone other than the approved admins. Furthermore, some data should only be available to employees and not to customers. Customers should not be able to see historic sales data in any form, nor should they be able to see inventory amounts as they could use either of these to gain a leg up in price negotiations. Finally, it is incredibly important that malicious attackers be unable to take the website down within reason. A dealer will not have the money for security like a larger company would, but an individual who is looking to take down the site for fun or for profit should be unable to.

Many sites scare customers away by having either too much information or not letting them access the content until they sign up. After the customer does sign up, sites often just use the customer's information to target them with ads, emails, and incessant car-buying information. The web application needs to serve the customer a pleasant experience without shoving ads down their throat or trying to get them to sign up instantly. The site should allow a customer to browse as they see fit and allow them to sign up and sign in for saved searches and preferences. Customers need to find it easy to navigate the pages and information, and if they do wish to purchase something, the process should be as simple as possible.

Car information should be shown in a simple layout on some sort of virtual card, that when clicked, takes the customer to a more detailed layout of that car's information. The "cards" should contain brief descriptions, key information such as price, and an image of the vehicle. The detailed information should have everything a customer might want to know about a car and possibly a link to the manufacturer's website. In addition to cars, different car parts and accessories should be available for browsing and purchasing. All of these parts and accessories should be modeled in a similar way to the cars but in a different section of the site.

Customers should be able to add multiple cars and parts to a cart, which will be saved in case the customer leaves the page. Once the customer has added at least one item to the cart, they should be able to click a checkout button, which will take them to a checkout page. This checkout page should provide payment options, financing estimates and dealership location for vehicles, and forms to fill out necessary information. The page should provide a submit button that the customer can click when they have finished filling out the information. This page should also verify that all necessary information is filled out. If not, the page should highlight missing information, and prompt the customer to complete it. If all information is complete, the application should display a confirmation page with a receipt.

Business Goals

- Customer Needs
    - Simplicity
        - Comparisons
            - Browsing by brands/color/model/etc.
                - Sort options
                    - Database with all related info
            - Search Bar
                - Being able to query everything available at once
                    - Database is searchable through this search bar
                - Interface for search bar
                    - Front end work (Possibly ReactJS)
            - Comparison tool
                - Front end can show multiple options side by side
        - Online-Only
            - Ability to purchase and research without employee interaction
                - Email form if questions are needed
                - Straightforward options that anyone could understand
        - Navigation
            - User sign in allows for saved queries
                - Ability to save previous searches
                    - Only if the user has a sign in
            - Visually appealing and easy to use
                - Design work
                - Simple layout
                - Navigation bar is constant throughout website
    - Low Pressure
        - Independence
            - No Required Sign Up
                - Browsing without sign in
            - Fully functional from start to finish
                - Cart
                - Checkout
                - Purchase Options
                    - Options available on each car/part

        - No invasive ads
            - If ads are included, they are sidebar only and do not error out
            - No auto playing ads or sounds
        - No data-mining/lead generation
- Dealership Needs
    - Security
        - Protection from malicious activities
            - Verification of data
                - Filtering all user input taken into site
            - Password hashing system
                - Secured database of user logins
        - Separation of permissions
            - Ability for front end to show different options depending on user-type
                - Back end needs to serve up info depending on login
            - Customer
                - Not able to see historic sales data
                - Not able to see inventory amounts
            - Employee
                - Not able to edit/create/delete historic sales data
                - Not able to edit/create/delete inventory amounts
            - Admin
                - Ability to edit/create/delete historic sales data without coding
                - Ability to edit/create/delete inventory amounts without coding
                - Interface specifically for admins
    - Cost
        - Possible future expansion
            - Allow scalability
                - Well used comments throughout
                - Ability to scale database (MySQL Community Edition)
        - Low cost
            - Work on needs first, then wants
                - Completion of base site, then add in extras
        - As small of database as possible
            - Fits within MySQL Community

Enumerated Functional Requirements

Identifier	Priority	Requirement
REQ-1	1	The application should allow the user to sort through vehicles based on multiple different criteria (price, brand, consumer rating, make, model, color, etc.).
REQ-2	5	The application should keep track of inventory and sales history, with sales history data being tracked automatically.
REQ-3	4	The application should provide 3 levels of authentication: Customer, Employee, and Admin.
REQ-4	3	The application should allow Admins to view, create, edit, and delete all data.
REQ-5	2	The application should allow Employees to view inventory and sales history data without the ability to create, edit, or delete it.
REQ-6	2	The application should allow Customers to view inventory data, without the ability to create, edit, or delete it, however, they should not be allowed to view inventory numbers.
REQ-7	4	The application should provide a graphical interface for Admins to view, create, edit, and delete data without the need for any coding.
REQ-8	3	The application should display inventory information in a simple layout on a virtual card. These cards should contain brief descriptions, key information, and an image of the vehicle.
REQ-9	4	The application should display a page with detailed information when the corresponding card is clicked that contains all the information a customer might want to know.
REQ-10	3	The application should allow Customers to add multiple cars and parts to a cart. The cart should be saved to allow Customers to leave the page and come back. A checkout button should be displayed after at least one item has been added to the cart, which will take the customer to a checkout page.
REQ-11	1	The checkout page should provide financing estimates, dealership location selector, payment options, and forms for necessary information.
REQ-12	2	The checkout page should verify all necessary information is filled out, highlighting all missing information on submitting. If all necessary information is filled out on submitting, a confirmation page with a receipt should be loaded.

Enumerated Nonfunctional Requirements

Identifier	Priority	Requirement
REQ-13	2	The user interface must be user-friendly and easy to use.
REQ-14	1	The user interface of the system must be simple enough so that any user can use it with a minimum training.
REQ-15	5	In case of failure, the system must be easy to recover and must suffer minimum loss of important data.
REQ-16	5	The application should be always readily available for concurrent use (by multiple users)
REQ-17	4	The application should be designed and work as a distributed system (running on multiple machines)
REQ-18	2	The application should have reusable design, including modularity in coding for future updates.
REQ-19	5	The application should have reliable features that enable data to be transmitted securely through the system.
REQ-20	4	The application should save errors faced by users and report them to admin.
REQ-21	3	The application should have a well defined theme that will be used throughout the whole website.
REQ-22	1	Each page in the website should be loaded relatively quickly.
REQ-23	4	The application should have a clear distinction between customer, employee, and admin account and make this distinction clear to users.
REQ-24	1	In case of a maintenance routine, the customers should be warning at least a day prior.
REQ-25	5	The application should be available at any time of the day

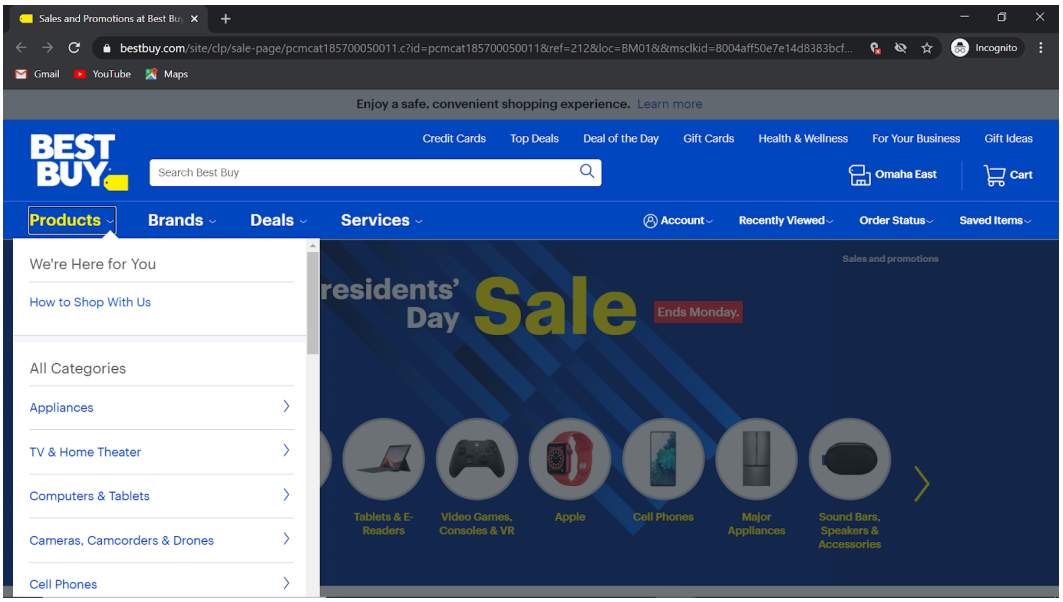
User Interface Requirements

I. Easy to navigate

A. **Description:** The user should not have any issues figuring out how to navigate the web application. Users should be able to find anything that they are looking for with navigation options such as a navigation menu, filter and sort options, and a search bar.

B. **Priority:** 2

C. **Graphic:**



(Includes many drop down navigation menus, sub-navigation menus, and a search bar for an easy to navigate experience.)
Source: https://www.bestbuy.com

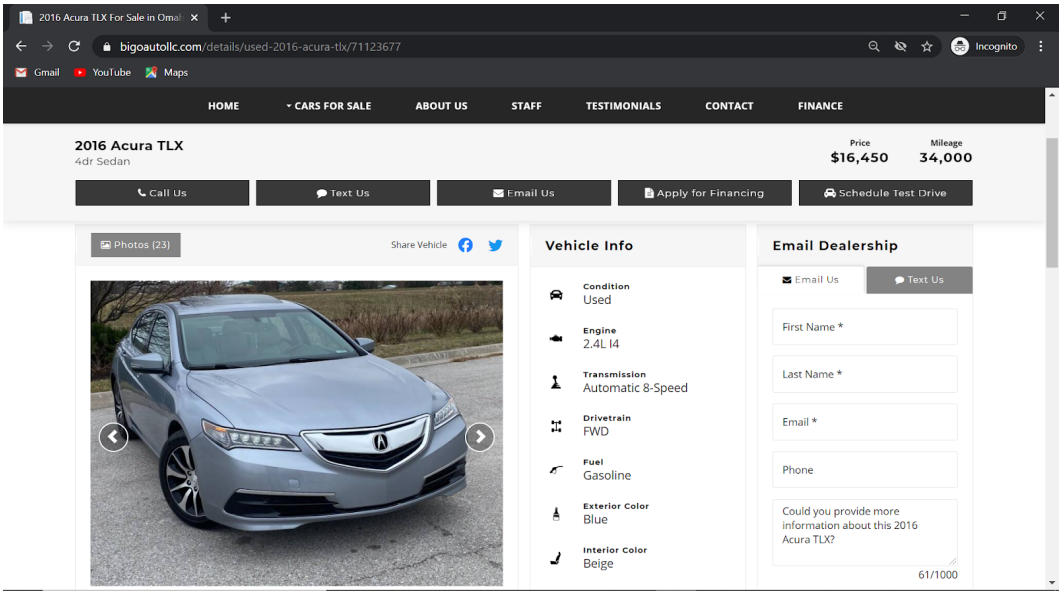
II. Informative and concise

A. **Description:** The user interface should include everything that a user will need to know about a car that they are planning to buy. This information should be displayed in an easy to read format that will not overwhelm the user with

paragraphs of text per product.

B. **Priority:** 3

C. **Graphic:**



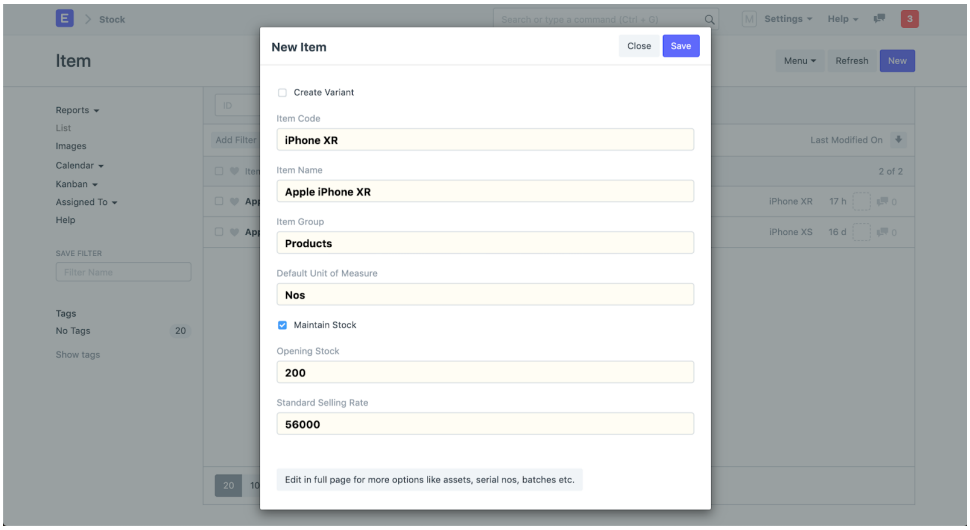
(Provides customers with price and mileage at the top, images on the left side, and concise information about the vehicle in the middle that is easy to process.)
Source: https://www.bigoautollc.com/details/used-2016-acura-tlx/71123677

III. Customizable

A. **Description:** The user interface should provide administrators with the ability to customize it without any coding knowledge. This may include adding, modifying, or deleting products.

B. **Priority:** 4

C. **Graphic:**



(Being able to add a new product easily without having to code it in.)
Source: [How to configure a Product Page]

IV. Scalable

A. **Description:** The user interface should be usable on devices with bigger screens and smaller devices such as a phone.

B. **Priority:** 1

C. **Graphic:**

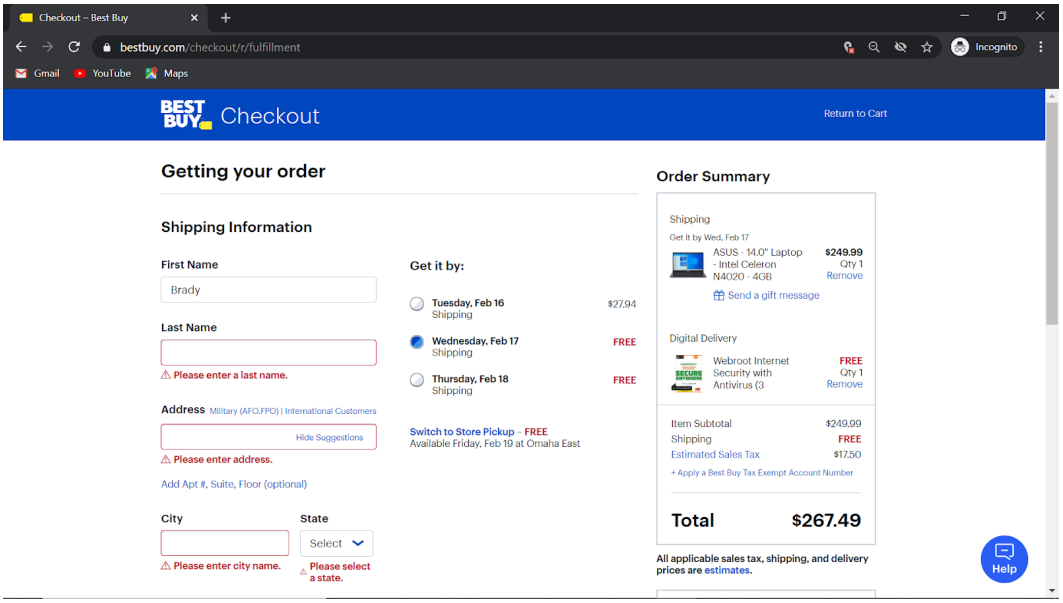
(Web application being usable for desktop and mobile devices.)
Source: [mock-up-desktop-and-mobile]

V. Feedback

A. **Description:** The user interface should provide responsive feedback to the user when they interact with it. This may vary from reminding the user to enter any information in a form that they may have forgotten, to notifying them when they have added something to their cart.

B. **Priority:** 2

C. **Graphic:**



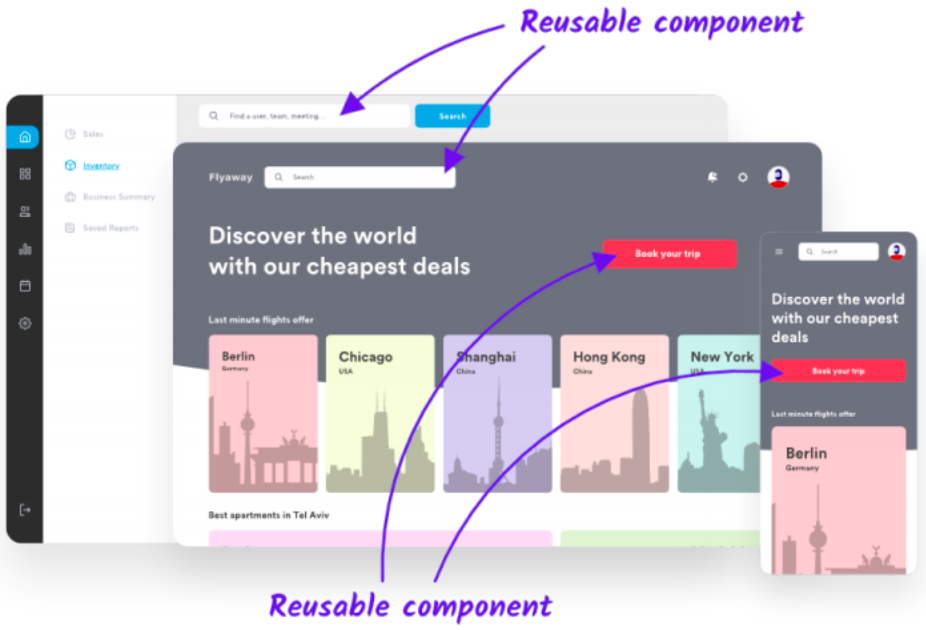
(The website highlighted the form entries I still need to fill out and printed a message below each one before it will let me check out.)
Source: https://www.bestbuy.com

VI. Consistent

A. **Description:** The user interface should be consistent across all of the different pages. This may include the general layout, fonts used, buttons, and icons.

B. **Priority:** 2

C. **Graphic:**



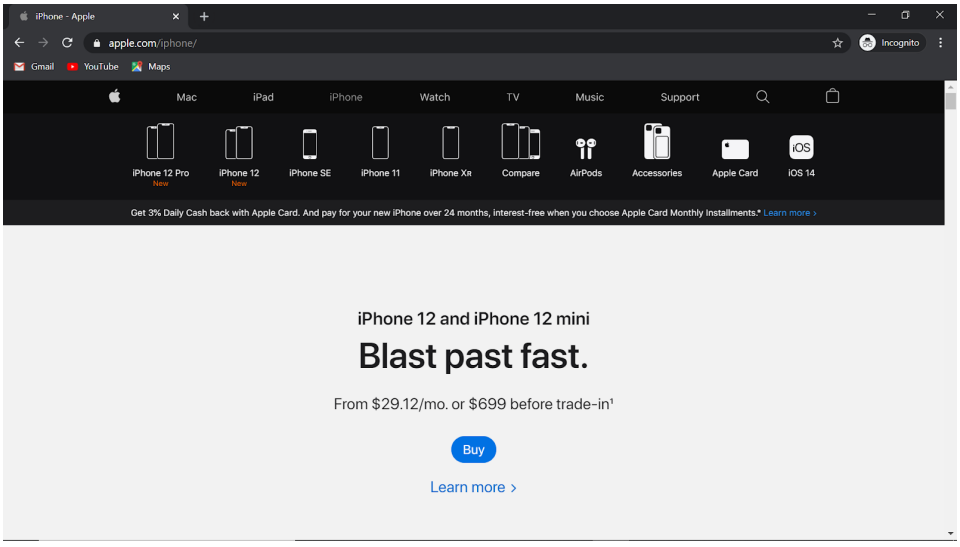
(Shows reusing certain content to provide a consistent look across the web application.)
Source: https://blog.bitsrc.io/5-ways-to-improve-ui-consistency-99013bf20417

VII. Attractive

A. **Description:** The user interface should be pleasant to look at and eye catching to the user.

B. **Priority:** 4

C. **Graphic:**



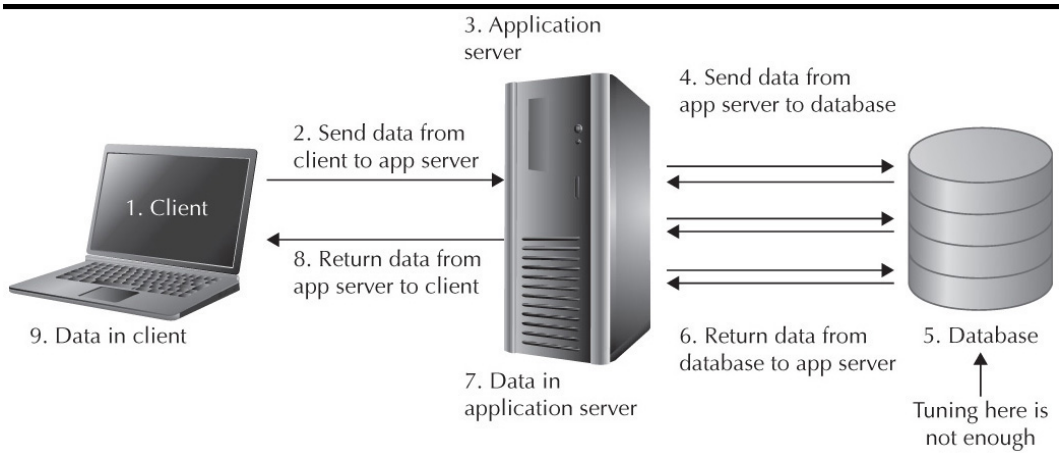
(Deciding to use small images of the products in the navigation menu is an example of grabbing more attention from the user rather than simple text.)
Source: https://www.apple.com/iphone/

VIII. Efficient

A. **Description:** The user interface should process data quickly and not have the user waiting for it to respond.

B. **Priority:** 3

C. **Graphic:**



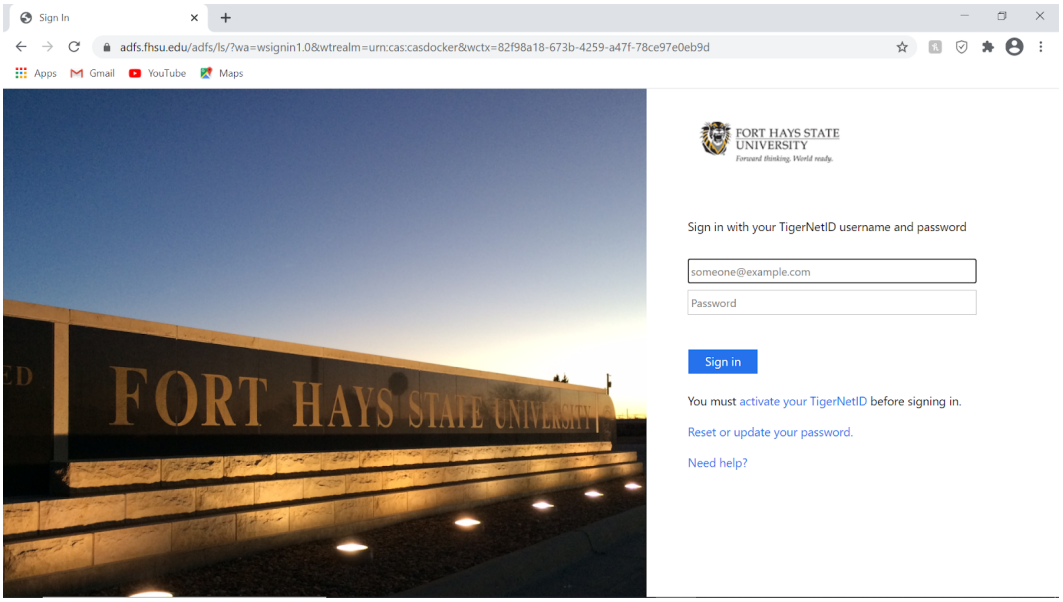
(Cycle for data processing in web application.)
Source: https://logicalread.com/application-flow-oracle-mc08/#.YCfXl2hKjb0

IX. Secure

A. **Description:** The user interface should ensure that data is secure and not allow a data breach to occur. This can include preventing a customer from editing the products by having an administrator login system, and protecting the financial information of the customer.

B. **Priority:** 4

C. **Graphic:**



(It is demanded that I login before I am able to access in order to prevent someone else from accessing my data.)
Source: https://blackboard.fhsu.edu/

Use Cases

Stakeholders

The stakeholders who hold an interest in this web application consist of customers, dealer employees, and dealer administrators.

Actors and Goals

Actor	Actor's Goal	Type	Use Case Name
Customer	Sort by category	Initiating	Sort (UC-1)
	Search via search bar	Initiating	Search (UC-2)
	Compare options	Initiating	Compare (UC-3)
	Sign up	Initiating	SignUp (UC-4)
	View information about vehicles/parts	Initiating	ViewCarInfo (UC-5)
	Email contact form to dealer	Initiating	EmailDealer (UC-6)
	Add item to cart	Initiating	AddItem (UC-7)
	Remove item from cart	Initiating	RemoveItem (UC-8)
	Checkout	Initiating	Checkout (UC-9)
Employee	Read sales data	Initiating	CrudSalesData (UC-10)
	Read inventory data	Initiating	CrudInventoryData (UC-11)

Admin	Edit/create/delete sales data	Initiating	CrudSalesData (UC-10)
	Edit/create/delete inventory data	Initiating	CrudInventoryData (UC-11)
	Add employee	Initiating	AddEmployee (UC-12)
	Remove employee	Initiating	RemoveEmployee (UC-13)
Application	Route email form from customer to dealer email	Participating	EmailDealer (UC-6)
	Record/store sales data	Participating	Checkout (UC-9)
	Record/store inventory data	Participating	CrudInventoryData (UC-11)
	Display content	Participating	All Use Cases
	Error generation/tracking	Participating	All Use Cases
	Filter user data entered	Participating	UC-4, 6, 7, 8, 9, 10, 11, 12, 13
	Hash passwords	Participating	UC-4, 12
	Storage of user login data	Participating	UC-4, 12
	Authenticate user	Participating	Authenticate (UC-14)

Casual Descriptions

Use Case Name	Sort (UC-1)
Actors	Customer, Application
Description	Customers can come to the site looking for a vehicle/part, and they would like to filter by the specific categories that interest them (e.g. brand, make, model, color, price, etc.). The application then filters by the customers selected criteria, then displays results.

Use Case Name	Search (UC-2)
Actors	Customer, Application
Description	Customers can come to the site looking for a vehicle/part, and they would like to search by certain criteria. A search bar allows them to enter any keyword they like, and the application filters and displays the results.

Use Case Name	Compare (UC-3)
Actors	Customer, Application
Description	Customers can come to the site looking for a vehicle/part, and they would like to compare different options. The customer should be able to select vehicles/parts to be compared side by side. The application then displays the content requested.

Use Case Name	SignUp (UC-4)
Actors	Customer, Application
Description	Customers can sign up to the application. The application will prompt the customer for an email/username and password. Upon receiving the data from the customer, the application filters and checks the data against already stored user login data. If the email/username is new, the application creates a new set of information in the user login datatable after hashing the password entered. If the email/username already exists, the application outputs to the customer that there is already a login with that information.

Use Case Name	ViewCarInfo (UC-5)
Actors	Customer, Application
Description	Customers will be able to see information about any vehicle/part they would like. They can go about this by viewing the “cards” available while sorting or viewing products. The customer can then click one of these cards to view more detailed information about the product. The application will serve up whatever is sorted/clicked on/available for the customer.

Use Case Name	EmailDealer (UC-6)
Actors	Customer, Application
Description	Customers who wish to email a content form to the dealer employees (this can be a question, request for contact, etc.), can fill out the form on the application and click submit to send it to the dealer. The application then verifies that this information is filled out correctly, and sends an email to the dealer.

Use Case Name	AddItem (UC-7)
Actors	Customer, Application
Description	Customers can add any available vehicle/part to their cart. The application will then store the item inside of the customer’s cart until they either checkout or delete the item from their cart.

Use Case Name	RemoveItem (UC-8)
Actors	Customer, Application
Description	Customers can delete any vehicle/part from their cart. The application will then remove the item from their cart.

Use Case Name	Checkout (UC-9)
Actors	Customer, Application
Description	Customers who wish to checkout with the items currently in their cart can enter their payment information and the application will process the payment. The application will then edit the sales data and inventory data accordingly. The application will then email the customer a receipt.

Use Case Name	CrudSalesData (UC-10)
Actors	Admin, Employee, Application
Description	An admin can choose to create, read, update, or delete sales data. An employee can only read sales data. The application will allow a specific interface for these options by employees/admin so that no coding is required. Any information that will be created or updated will be filtered to ensure the data fields are correct before submission. The application will then update the database according to changes made.

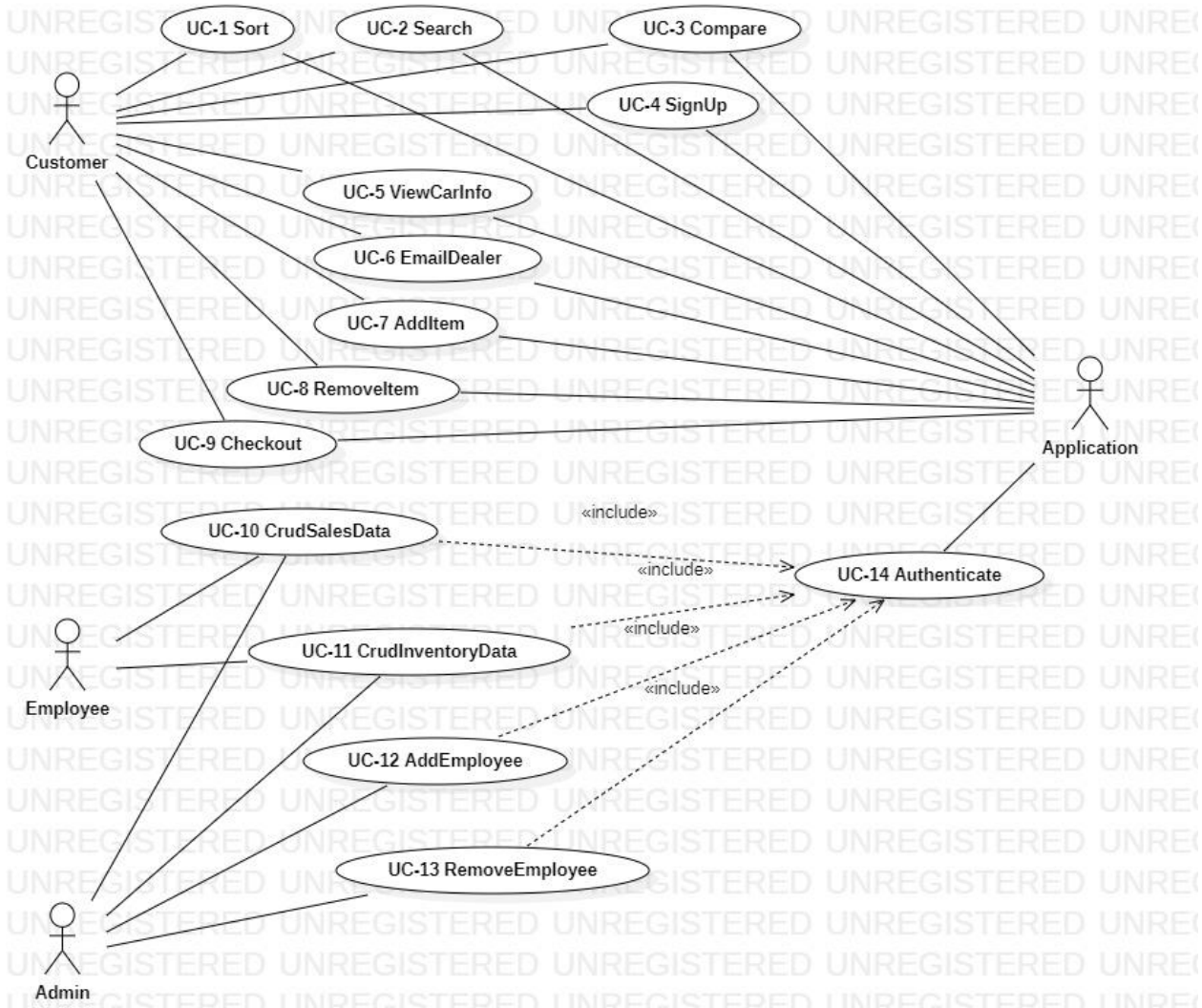
Use Case Name	CrudInventoryData (UC-11)
Actors	Admin, Employee, Application
Description	An admin can choose to create, read, update, or delete inventory data. An employee can only read inventory data. The application will allow a specific interface for these options by employees/admin so that no coding is required. Any information that will be created or updated will be filtered to ensure the data fields are correct before submission. The application will then update the database according to changes made.

Use Case Name	AddEmployee (UC-12)
Actors	Admin, Application
Description	An admin can choose to add an employee login to the application. The application will ensure that the username/email does not already exist in the user login database, then store the new employee login data to the database. This includes hashing the password before storage.

Use Case Name	RemoveEmployee (UC-13)
Actors	Admin, Application
Description	An admin can choose to remove an employee login to the application. The application will ensure that the username/email already exists within the database, then allow the admin to delete the login data. The application will then make the corresponding changes within the database.

Use Case Name	Authenticate (UC-14)
Actors	Application, Customer, Employee, Admin
Description	The application will authenticate any user signing into the application from the user login database. Upon login, the application will check the entered data against stored data to ensure that the correct person is logging in.

Use Case Diagram



Traceability Matrix

Req	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14
REQ1	1	x	x			x									
REQ2	5									x	x	x			
REQ3	4				x		x						x	x	x
REQ4	3										x	x			
REQ5	2										x	x			
REQ6	2	x	x			x									
REQ7	4										x	x			
REQ8	3	x	x			x		x	x						
REQ9	4					x		x	x						
REQ10	3			x				x	x	x					
REQ11	1									x					
REQ12	2									x					
Max PW		3	3	3	4	4	4	4	4	5	5	5	4	4	4
Total PW		6	6	3	4	10	4	10	10	11	14	14	4	4	4

Fully-Dressed Description

Use Case UC-5: **ViewCarInfo**

Related Requirements: REQ-1, REQ-6, REQ-8, REQ-9, REQ-13, REQ-14

Initiating Actor: Customer

Actor's Goal: View information about vehicles/parts

Participating Actors: Application

Preconditions: The customer is viewing the application on their browser

Postconditions: The application has served the information in whatever form the customer requested

Flow of Events for Main Success Scenario:

→ 1. Customer selects option to view products (sort, search, card click, etc.)

← 2. The application takes input from user, and displays information accordingly

Use Case UC-9: **Checkout**

Related Requirements: REQ-2, REQ-10, REQ-11, REQ-12

Initiating Actor: Customer

Actor's Goal: Checkout and arrange payment for products in cart

Participating Actors: Application

Preconditions:
- The customer has items in their cart
- The customer has clicked the checkout button

Postconditions: The customer has purchased their items, and has been sent a receipt

Flow of Events for Main Success Scenario:

→ 1. Customer clicks checkout button

← 2. Application displays the checkout page and form

→ 3. Customer fills out form (address, credit card info, shipping, etc.)

← 4. Application confirms information

← 4a. Application updates databases

← 4b. Application emails receipt to customer

← 4c. Application displays success to customer

Flow of Events for Extensions (Alternate Scenarios):

→ 1. Customer clicks checkout button

← 2. Application displays the checkout page and form

→ 3. Customer fills out form (address, credit card info, shipping, etc.)

← 4. Application denies information, informs customer of incorrect data, and sends them back to checkout page and form

→	5.		Customer fills out form again (4-5 can loop)
←	6.		Application confirms information
←		6a.	Application updates databases
←		6b.	Application emails receipt to customer
←		6c.	Application displays success to customer

Use Case UC-10, 11:	**CrudSalesData, CrudInventoryData**
Related Requirements:	REQ-2, REQ-4, REQ-5, REQ-7
Initiating Actor:	Admin, Employee
Actor's Goal:	Create, read, update, delete sales data
Participating Actors:	Application
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	The application has either displayed the information requested, or updated the databases as requested

Flow of Events for Main Success Scenario:

→	1.		Admin requests change to information (or view in case of employee)
←	2.		Application checks data, verifies it successfully
←		2a.	Application updates databases
←		2b.	Application outputs success to user

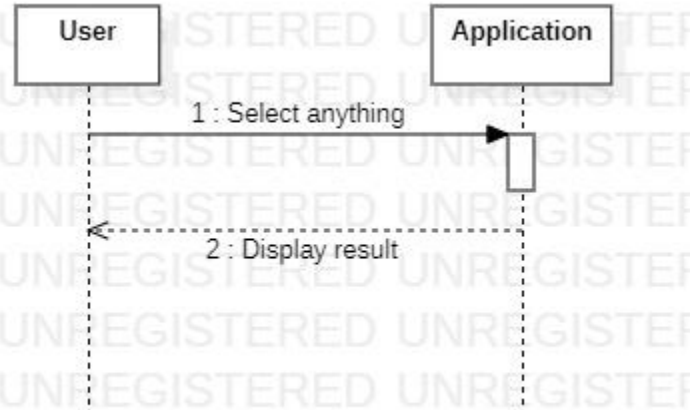
Flow of Events for Extensions (Alternate Scenarios):

→	1.		User request change to information
←	2.		Application checks data, does not verify

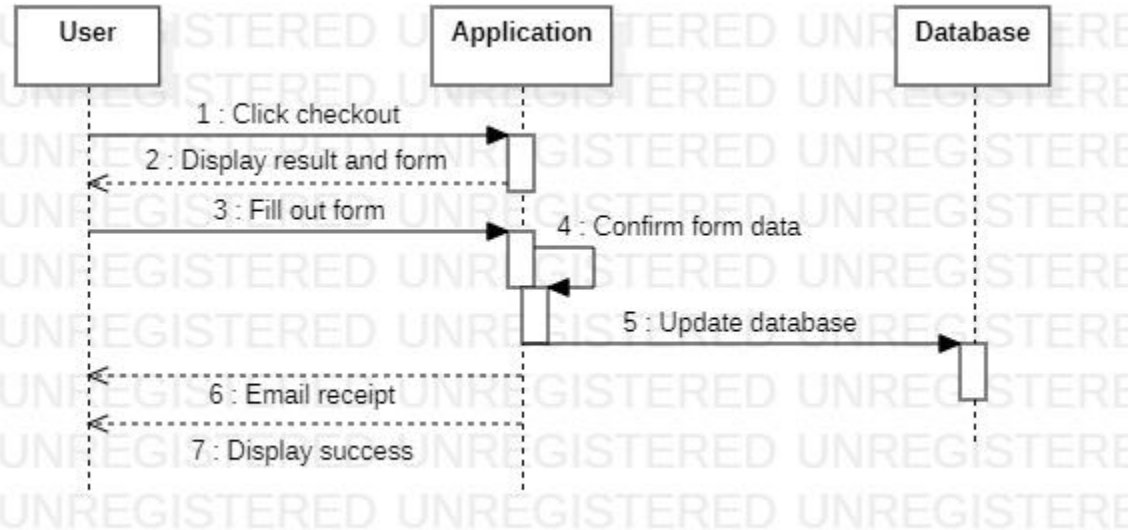
←	2a.	Application sends error message to user

System Sequence Diagrams

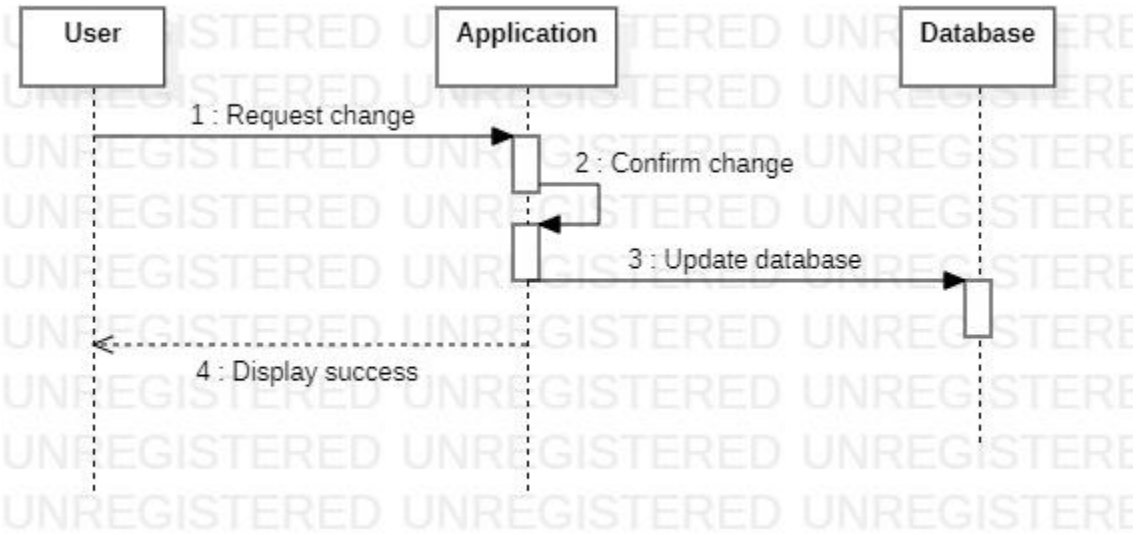
ViewCarInfo (UC-5)



Checkout (UC-9)



CrudSalesData, CrudInventoryData (UC-10, UC-11)



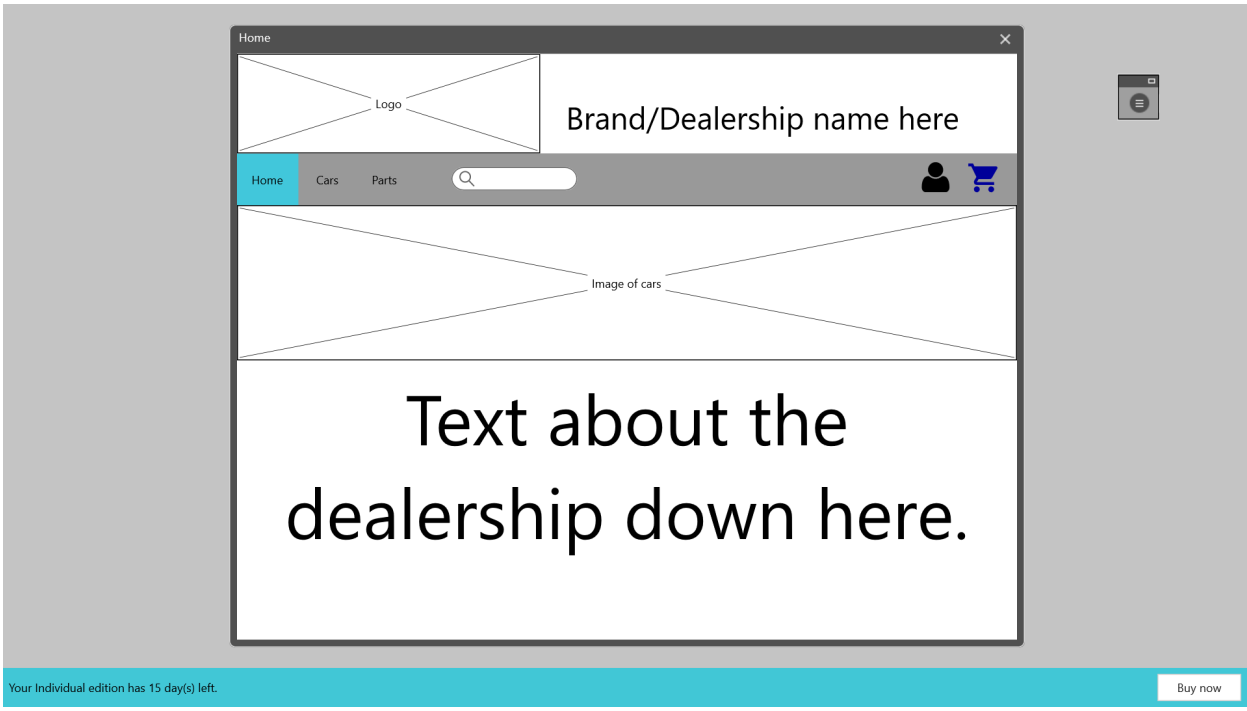
User Interface Specification

The mockups in the pictures are a blueprint of the features that should be present in our web application and are not what the final product will look like. All mockups in this section of the report were created using the application MockPlus.

I. Preliminary design

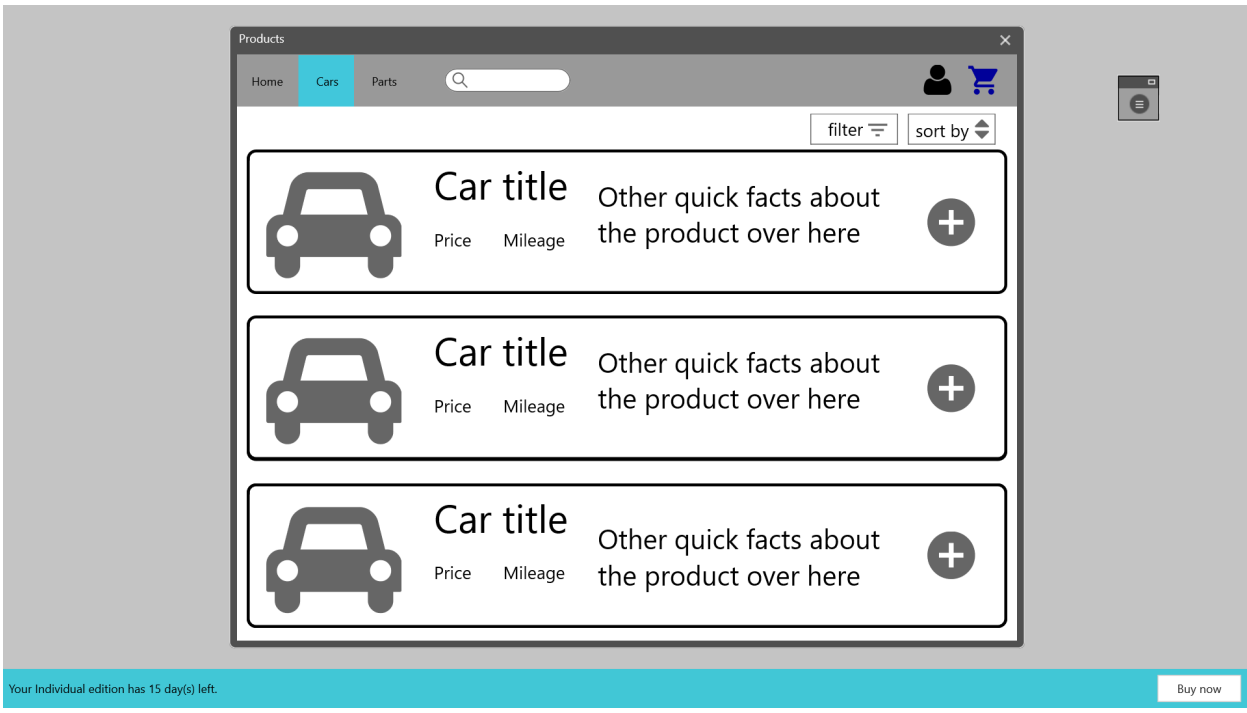
A. UC-5: ViewCarInfo

Once the user enters the application, they will be greeted with the home page.

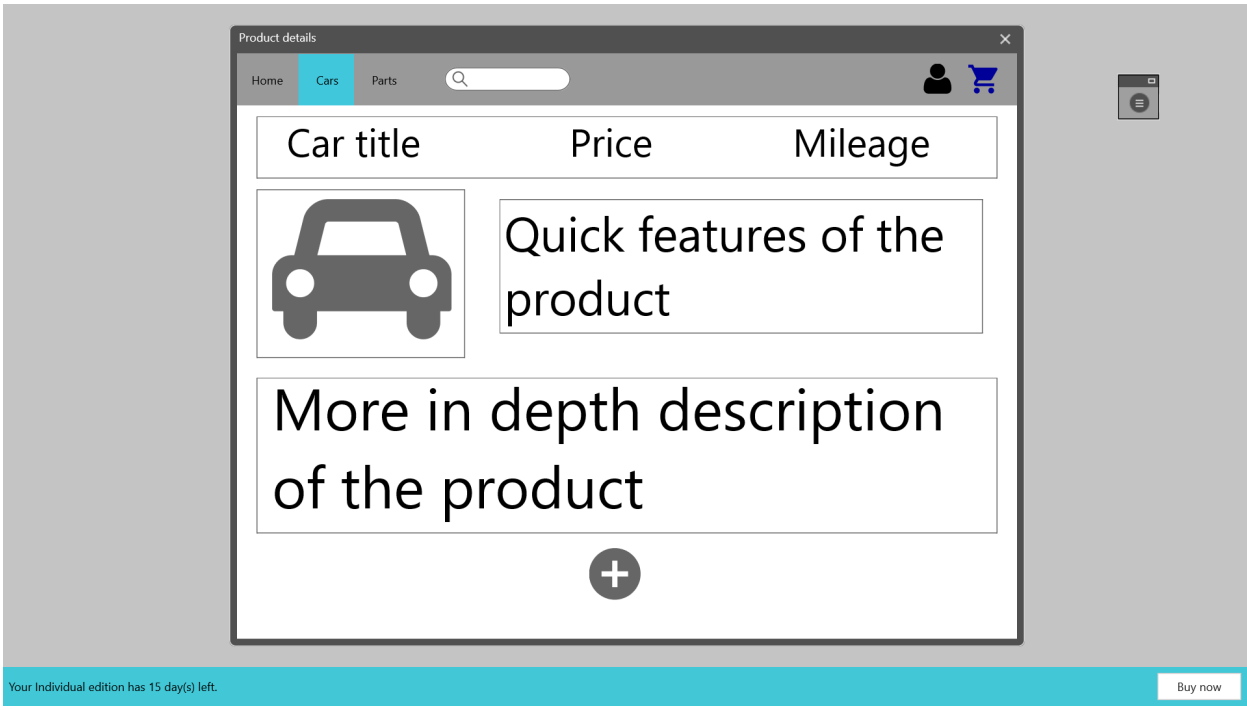


To view the cars that we have available, they can click on the ‘Cars’ tab at the top of the page. If they know exactly what they are looking for, then they can use the search bar that is also located at the top of the page to search for their desired car.

Once the user clicks on the ‘Cars’ tab, they will be taken to a new page with all of the available cars.



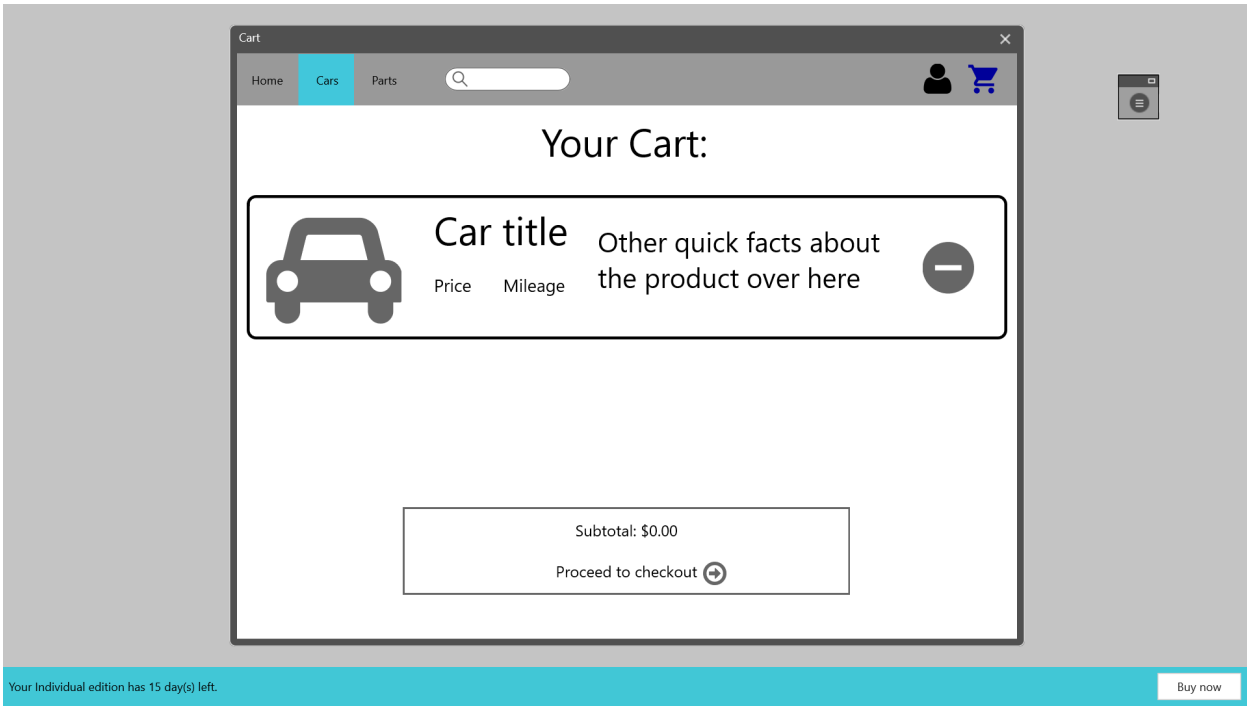
All of the cars we have available will be able to be seen in a set order. The user can change the order that the vehicles are displayed by clicking on the sort by button. They can choose to sort by price, mileage, and alphabetical, all in ascending and descending order. If the customer wants to limit the number of cars they see, they can use the filter button above the products. They can choose to filter products depending on make, model, price, or mileage. Once they have found a product they are interested in, they can add it to their cart by clicking the plus button on the card, or click on the card itself to get a more detailed view of the product.



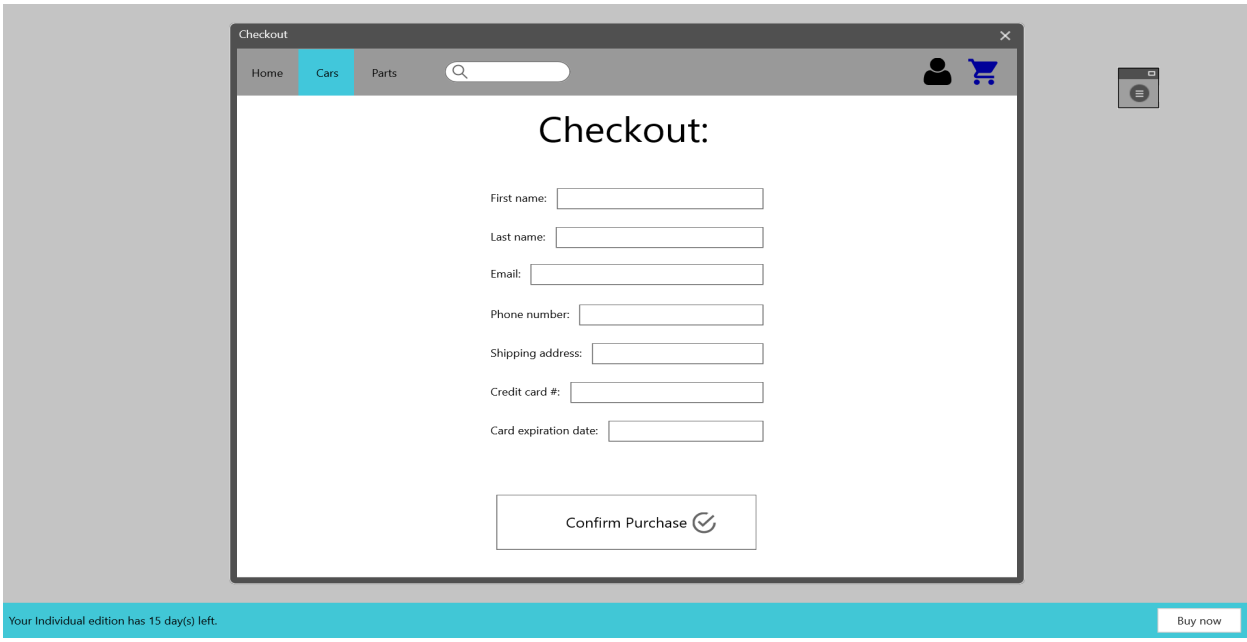
This page will display all the information that a user needs to know about a particular vehicle that they are interested in. If they decide that they want to purchase the car, they can add it to their cart by clicking the button at the bottom of the page.

B. UC-9: Checkout

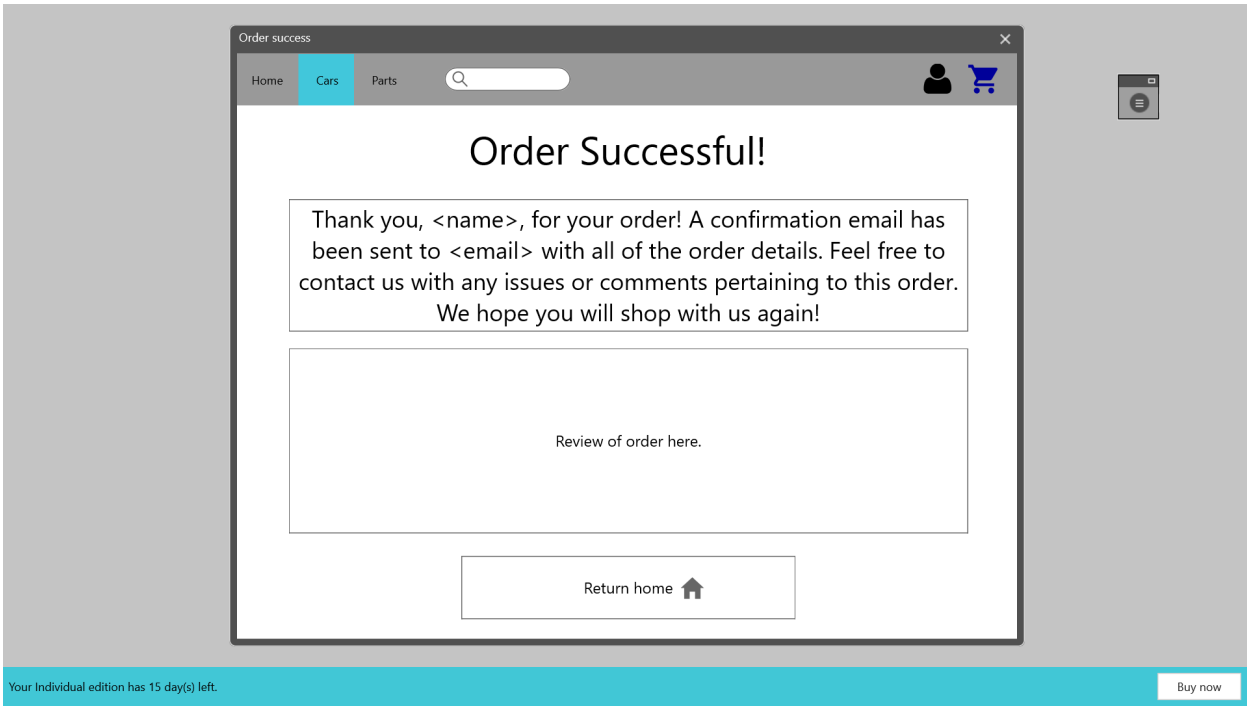
Once a user has one or more products in their cart, they are able to checkout and purchase their products. The first step to purchasing their items is to go to the cart page. They can go to the cart page by clicking the shopping cart button that is located on the top right corner of the navigation bar.



In their cart they can see the items that they are about to purchase, and the subtotal for all of the items. They can remove the items that they do not want using the button on the left side of their item. Once they have everything that they want to purchase in their cart, they can checkout by clicking the button below the subtotal.



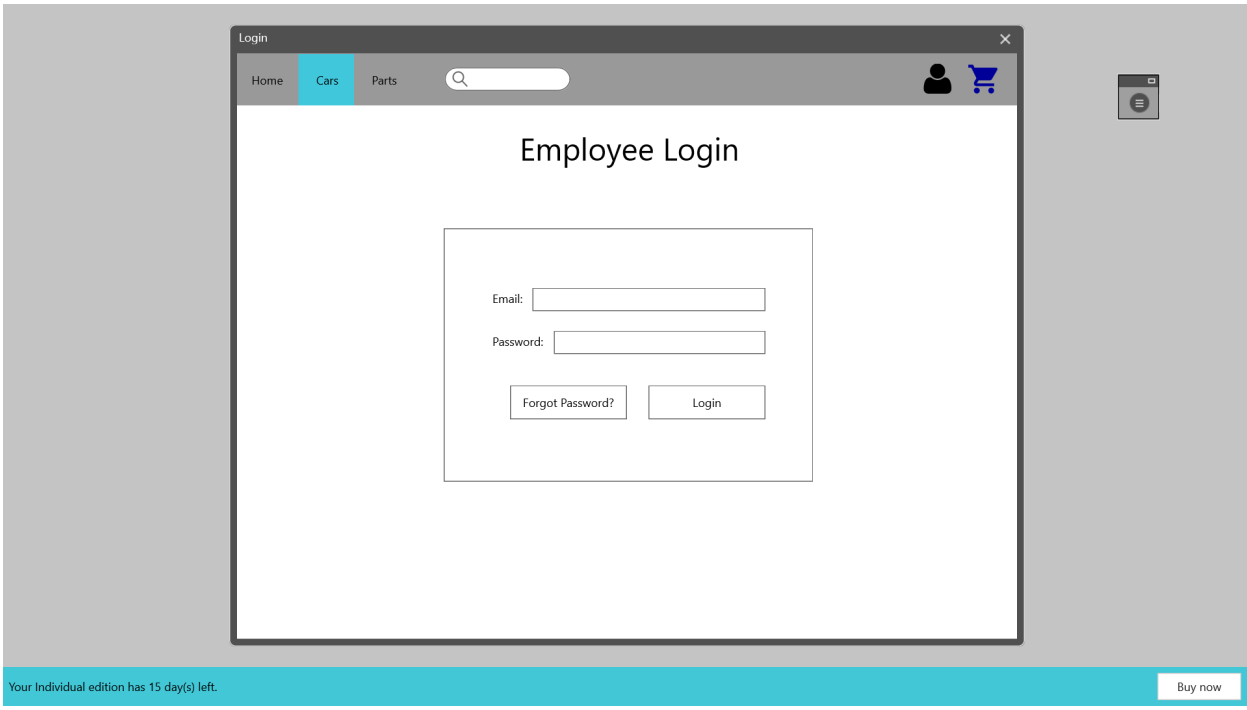
A form will appear that the customer must fill out in order to purchase their products. The information that the customer needs to fill out includes their first name, last name, email, phone number, address, and credit card information. Once they fill out every section, they can click the ‘Confirm Purchase’ button at the bottom of the page. If the user forgets to fill out a section or if they put incorrect information, then the application will return them to the form and tell them which section is wrong.



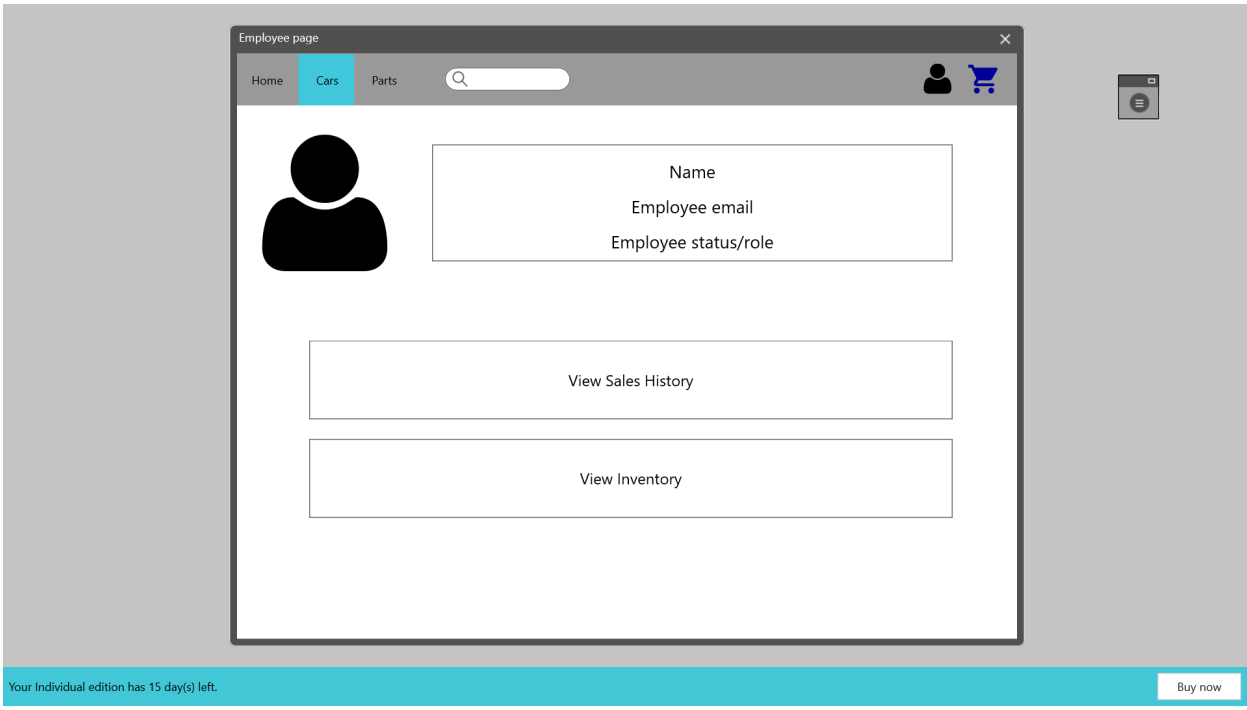
If the customer's order was successfully processed, they will be taken to a page that will thank them for their order. This page also tells them that a confirmation email has been sent to them. Below that will be a review of the customer's order. At the bottom of the page will be a button that a user can click to return to the home page.

C. UC-10, 11: CrudSalesData, CrudInventoryData

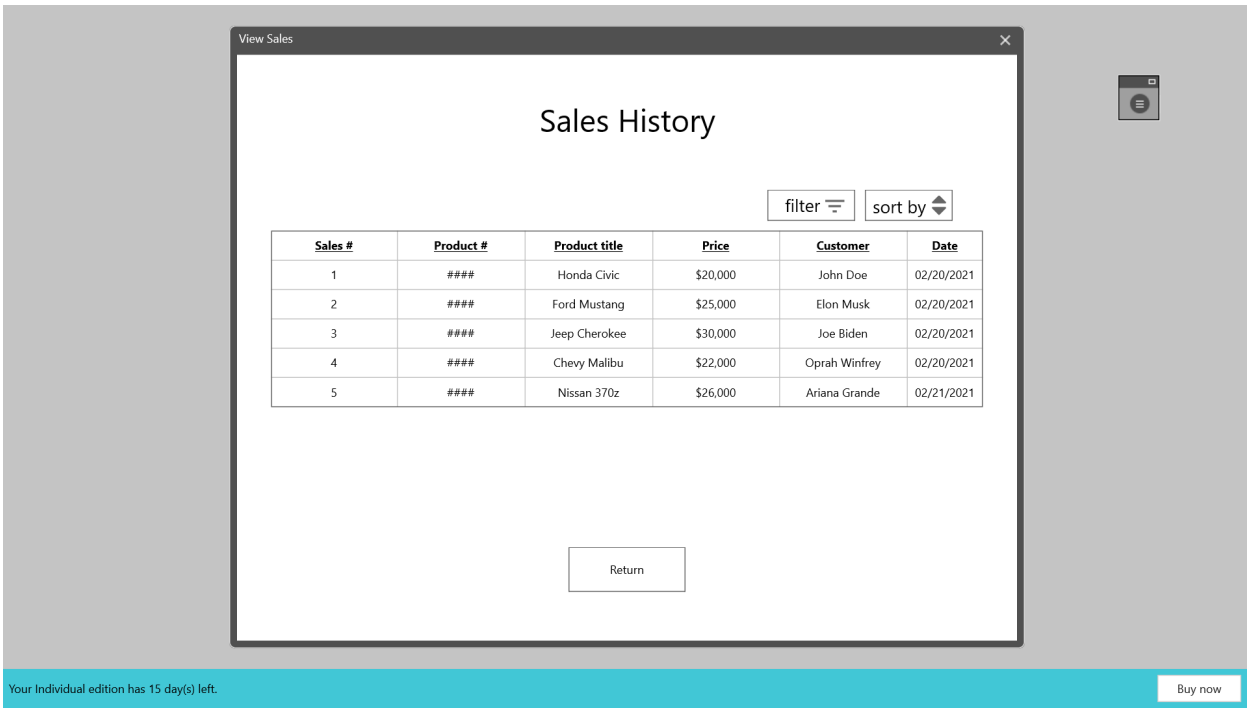
This web application will feature extra features for employees and administration that the customers will not be able to access. If an employee or a member of administration clicks on the login button that is on the navigation bar next to the shopping cart, they will be prompted to login.



The employee will be prompted to enter their email address and the password that are associated with their account. If they do not enter a valid email or password, the web application will tell them that their email or password is incorrect and they can try again. There is a forgot password button where they can have a reset confirmation sent to their email if they click the button. Once they have entered in the correct information, they can hit the login button to sign in. If the web application recognizes their information as an employee account, they will be taken to an employee page.



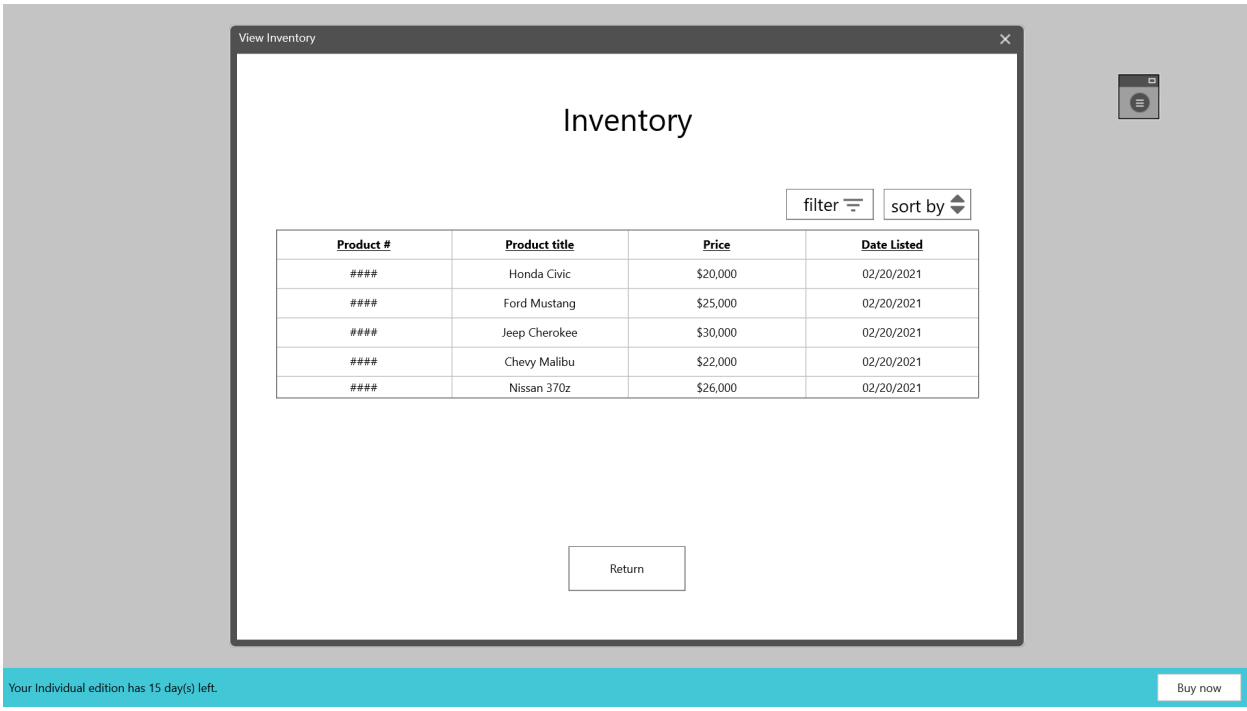
Within this employee page, there are two things that they can do that are not available to regular customers. The first thing that they can do is view sales history. This can be done by clicking on the 'View Sales History' button.



Sales #	Product #	Product title	Price	Customer	Date
1	####	Honda Civic	$20,000	John Doe	02/20/2021
2	####	Ford Mustang	$25,000	Elon Musk	02/20/2021
3	####	Jeep Cherokee	$30,000	Joe Biden	02/20/2021
4	####	Chevy Malibu	$22,000	Oprah Winfrey	02/20/2021
5	####	Nissan 370z	$26,000	Ariana Grande	02/21/2021

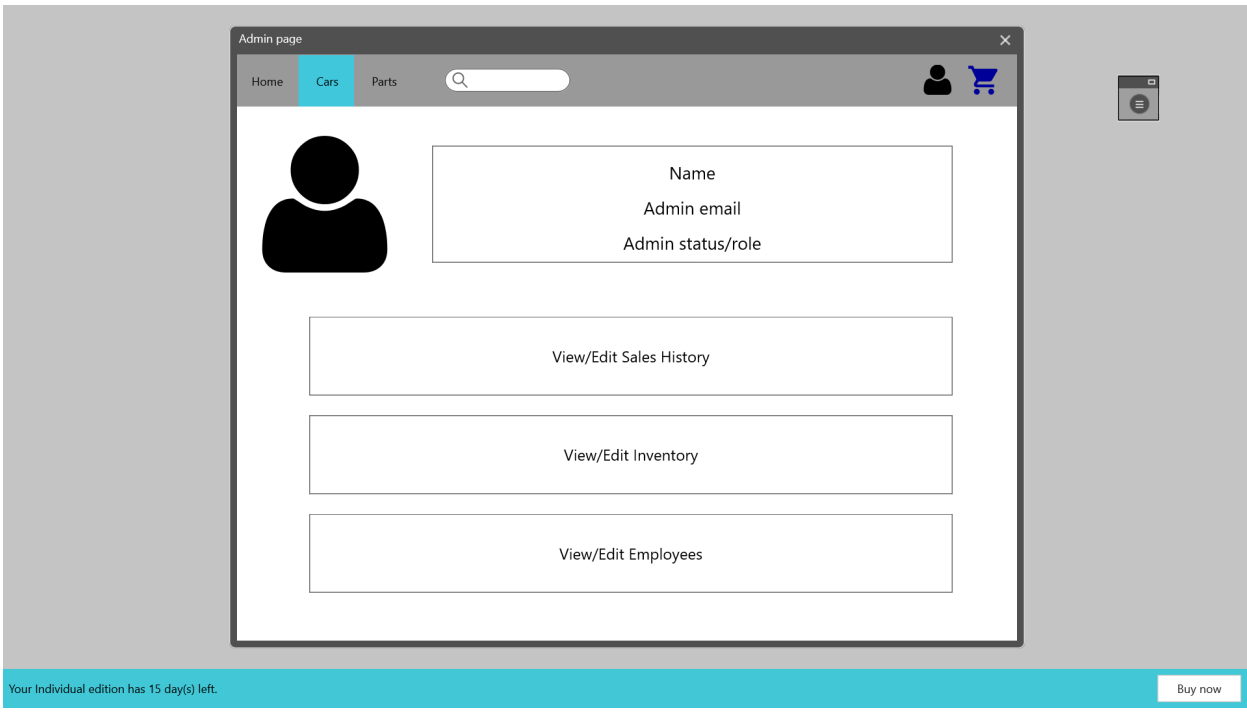
The employee will then see all of the previous sales that have been recorded to the database. This data will be in an easy to understand format. They will also have the ability to filter and sort the data as they please to find the specific sales record that they are looking for. Once they are done, they can click the button at the bottom of the page to return to their employee page.

An employee can also view all of the current inventory in the database by clicking on the ‘View Inventory’ button.

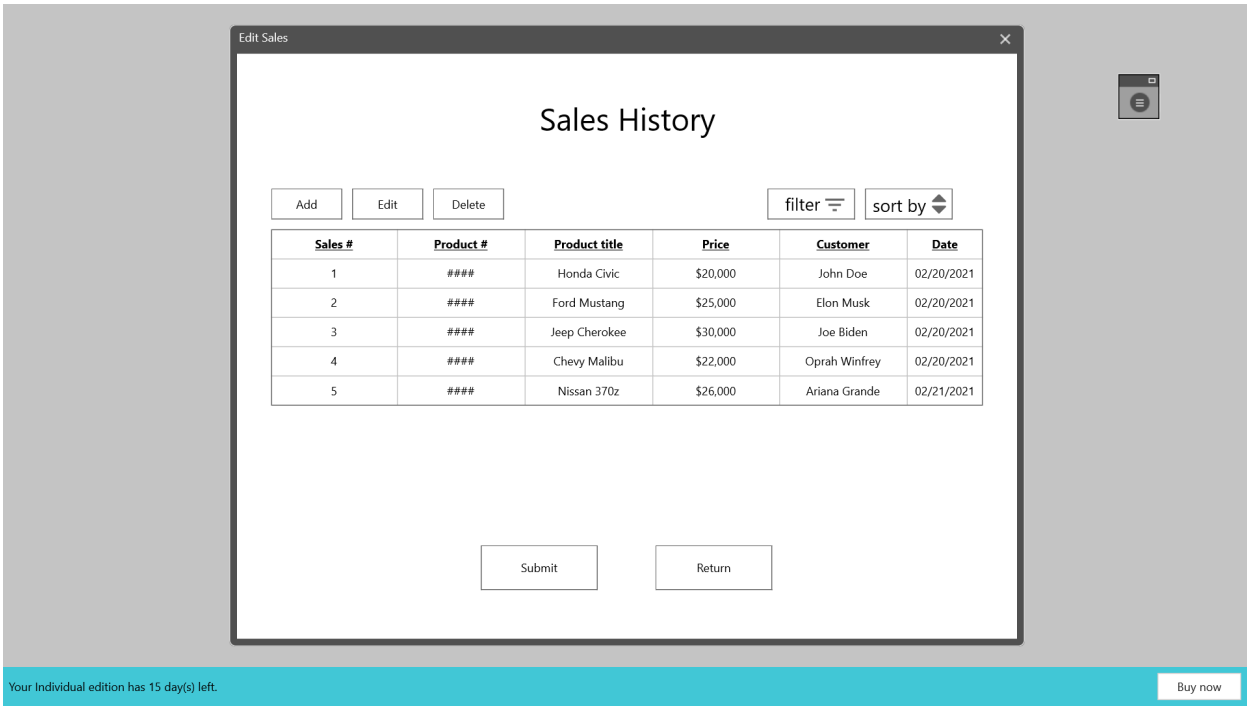


This will look similar to the view sales history, except that it will have accurate and updated inventory data rather than sales history. Once again, it will be in an easy to read format that the user will be able to filter and sort through. Once they are done, they can click the return button at the end of the page.

If the email and password that was provided at the login screen is recognized as an being associated with an administrator account, they will be taken to a slightly different administrator page that has added features.

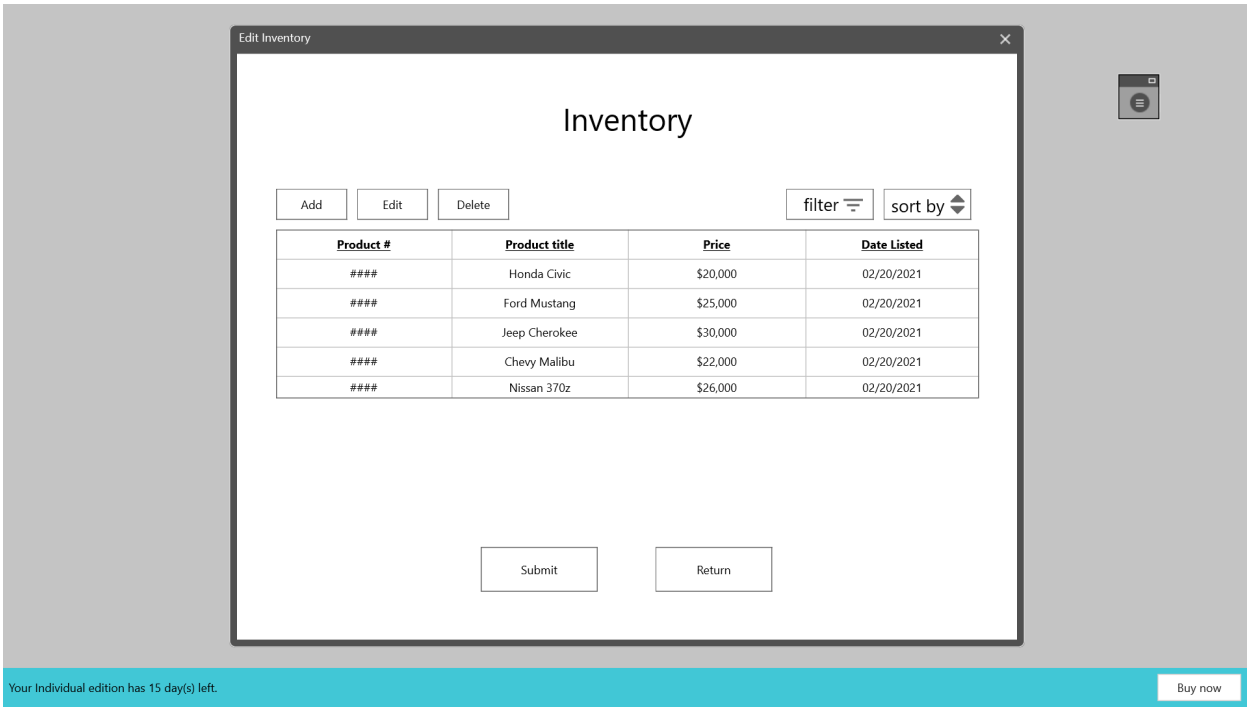


The first thing that an administrator is able to do is view and modify the sales history database by clicking on the top button that says ‘View/Edit Sales History’.



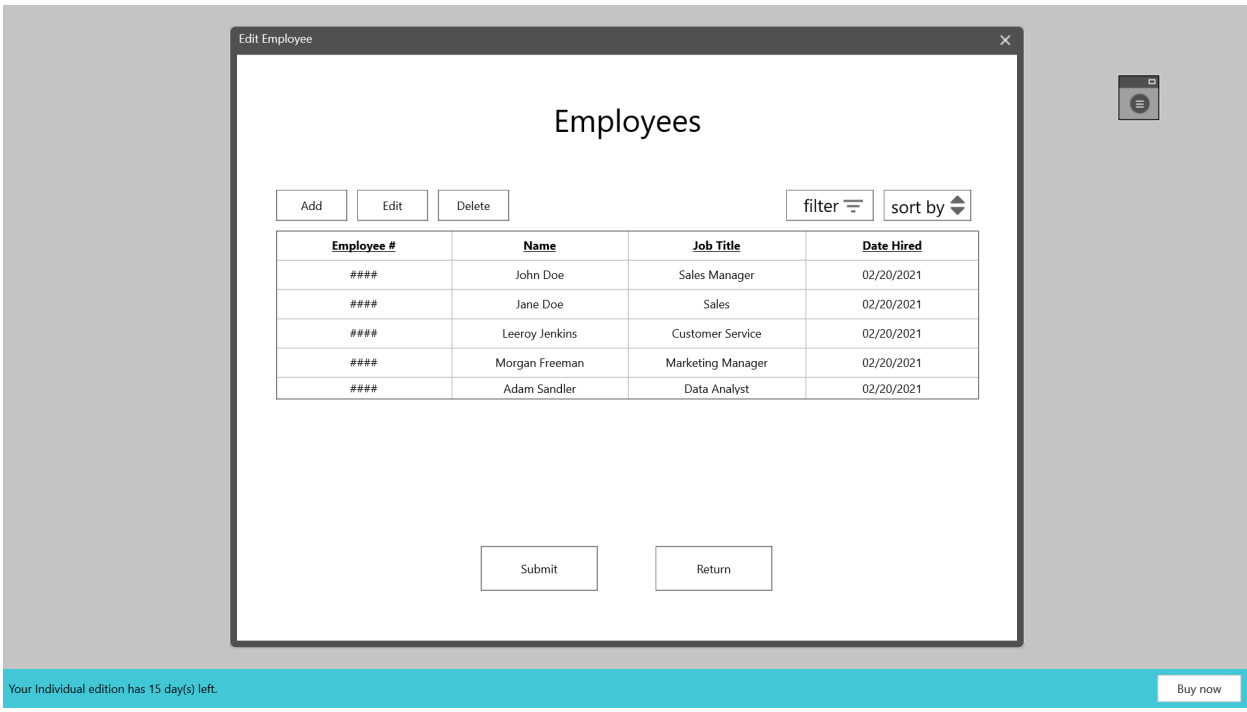
This will take them to the page that is similar to the sales history page of the employees, but the administrator will have the option to add, remove, or change the contents in it by clicking on the new buttons above the table. They will be able to do these tasks with no knowledge of coding or databases. Once they have modified the data to their liking, they can click the new ‘Submit’ button that is at the bottom of the page. The web application will verify these changes before they are submitted to the database. If the changes are able to get verified, the administrator will receive a message letting them know and the changes are added to the database. If the changes are not able to be verified, the database will not change and the administrator will be informed that their changes could not be submitted.

Back at the administrator page, the second button that they can click on is the ‘View/Edit Inventory’ button. This will take them to a page that allows them to view or modify the product inventory.



This is very similar to the view/edit sales history page except that it lists the products in the database instead. Once again, the administrator can add, change, or delete entries in this table by clicking on the corresponding buttons above the table. They can click the submit button at the bottom of the page and the web application will verify these changes. If they are approved, the changes are made to the database and the administrator is notified. If they are not approved, the database does not change and the administrator is notified that their changes could not be approved.

The third thing that an administrator is able to do on their page is view and edit the current employees at the dealership. This can be done by clicking on the last button on their administration page, ‘View/Edit Employees’.



They will then be taken to a page that lists all of the employees that are in our database. The administrator can view, add, edit, and delete employee information. They are also able to sort and filter the employee records to find a specific employee easier. Once they have modified the records, they can click the submit button at the bottom of the page to have their changes verified and receive a message about if the changes were approved or not.

II. User effort estimation

A. View car information

1. Navigation: total 2 clicks required, 4 optional clicks

a) Click ‘Cars’ tab
b) Click desired car card
c) Optional: Click ‘Filter’ tab
d) Optional: Click filter requirement
e) Optional: Click ‘Sort’ tab
f) Optional: click sort requirements

2. Data Entry: No data entry required

B. View sales history/inventory

1. Navigation: total 4 clicks required
    a) Click log in tab at the top of the page
    b) Click on email field
    c) Click ‘Login’ button
    d) Click ‘View Sales History’ or ‘View Inventory’ button

2. Data Entry: total 36 keystrokes required
    a) Enter email address (average 25 keystrokes)
    b) Press tab to move to password field
    c) Enter password (average 10 keystrokes)

C. Purchase an item

1. Navigation: total 6 clicks required
    a) Click ‘Cars’ or ‘Parts’ tab on navigation bar
    b) Click the plus button to add to cart
    c) Click the shopping cart button on the navigation bar
    d) Click the checkout button
    e) Click the ‘First name’ Field
    f) Click the confirm purchase button

2. Data Entry: total 110 keystrokes required
    a) Enter first name (average 6 keystrokes)
    b) Press tab to move to last name
    c) Enter last name (average 6 keystrokes)
    d) Press tab to move to email
    e) Enter email (average 25 keystrokes)
    f) Press tab to move to phone number
    g) Enter phone number (10 keystrokes)

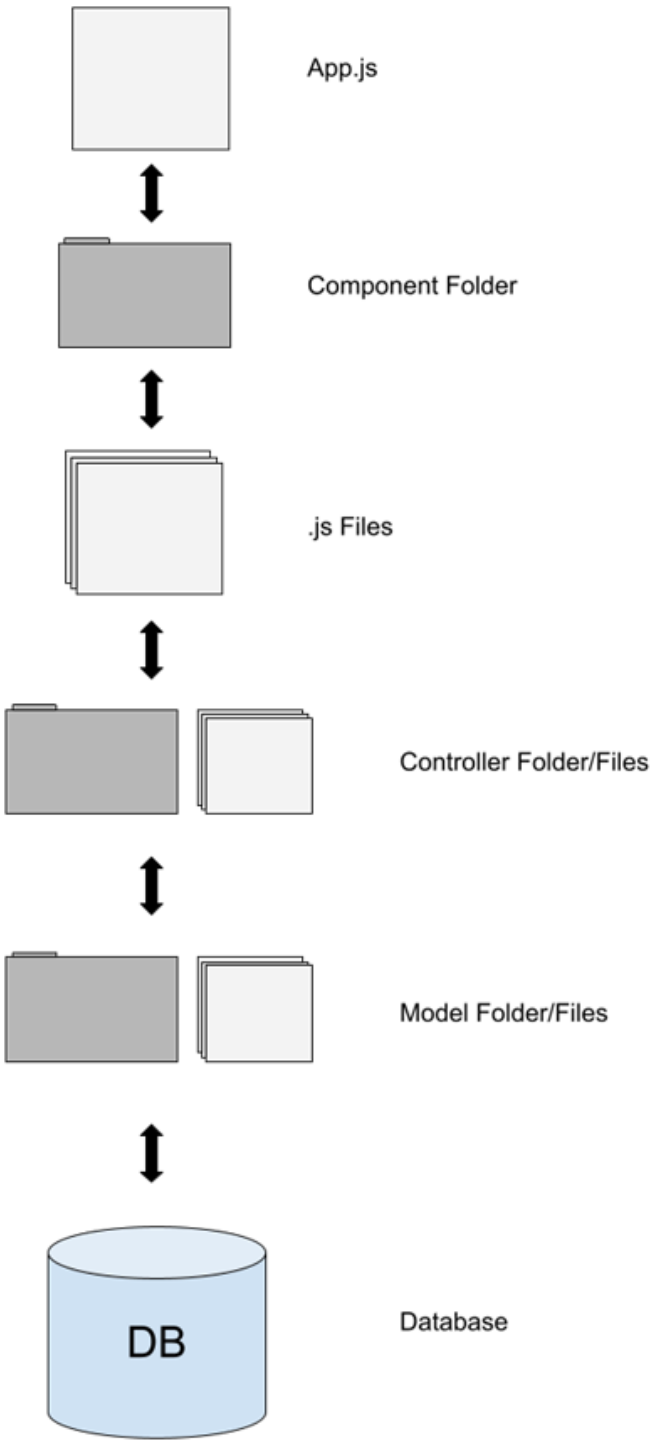
h) Press tab to move to shipping address
i) Enter shipping address (average 36 keystrokes)
j) Press tab to move to credit card #
k) Enter credit card #(16 keystrokes)
l) Press tab to move to card expiration date
m) Enter card expiration date (5 keystrokes)

D. Search for specific product

1. Navigation: total 1 click required
    a) Click the search bar in the navigation menu

2. Data entry: maximum of 41 keystrokes required
    a) Type the desired product name (40 keystrokes max)
    b) Hit enter to search

System Architecture



The image above depicts the structure of the web application from the perspective of how it is displayed to the user. Since our project implements React, everything is rendered through the App.js component. The App.js component pulls in other components through the file structure as needed in order to render them to the user. These other components are shown as the ".js files" within the depiction above. As each of these components needs to either get or post information from the database, they do so through the Controller folder and files. The Controller then sends the request to the Model, which in turn is connected to the database. This structuring is following the MVC web design pattern with the components acting as the view.

Project size (Use Case Points)

UAW

Actor	Description	Complexity (1-3)
Customer	Customers are interacting with the application through an online brower.	3
Employee	Employees are interacting with the application through an online brower.	3
Admin	Admin are interacting with the application through an online brower.	3
Database	Database is interacting with our application through sending/receiving data.	2

Total UAW = 11

UUCP

Use Case	Description	Complexity	Weight
Sort (UC-1)	Complex user interface. Four steps for the main success scenario. Two participating actors (Application, Database).	Average	10
Search (UC-2)	Complex user interface. Four steps for the main success scenario. Two participating actors (Application,	Average	10

	Database).		
Compare (UC-3)	Complex user interface. Five steps for the main success scenario. Two participating actors (Application, Database).	Complex	15
SignUp (UC-4)	Moderate user interface. Six steps for the main success scenario. Two participating actors (Application, Database).	Average	10
ViewCarInfo (UC-5)	Simple user interface. Four steps for the main success scenario. Two participating actors (Application, Database).	Simple	5
EmailDealer (UC-6)	Moderate user interface. Four steps for the main success scenario. One participating actor (Application).	Simple	5
AddItem (UC-7)	Complex user interface. Six steps for the main success scenario. Two participating actors (Application, Database).	Complex	15
RemoveItem (UC-8)	Complex user interface. Six steps for the main success scenario. Two participating actors (Application, Database).	Complex	15
Checkout (UC-9)	Complex user interface. Seven steps for the main success scenario. Two participating actors (Application, Database).	Complex	15
CrudSalesData (UC-10)	Complex user interface. Four steps for the main success scenario. Two participating actors (Application,	Complex	15

	Database).		
CrudInventoryData (UC-11)	Complex user interface. Four steps for the main success scenario. Two participating actors (Application, Database).	Complex	15
AddEmployee (UC-12)	Moderate user interface. Four steps for the main success scenario. Two participating actors (Application, Database).	Average	10
RemoveEmployee (UC-13)	Moderate user interface. Four steps for the main success scenario. Two participating actors (Application, Database).	Average	10
Authenticate (UC-14)	Simple user interface. Five steps for the main success scenario. Two participating actors (Application, Database).	Simple	5

Total UUCP = 155

TCP

Technical Factor	Description	Weight	Perceived Complexity	Calculation (W * PC)
T1	Distributed, Web application	2	3	6
T2	Users expect quick, but speed is not everything	1	2	1
T3	Users expect perfect efficiency	1	5	5
T4	Internal application workings are complicated	1	4	4
T5	Average reusability	1	3	3

T6	No install	0.5	0	0
T7	Ease of use is of primary importance	0.5	5	0
T8	Ability to keep database options open	2	1	2
T9	Additional features and updates are likely	1	4	4
T10	Concurrent use is required	1	1	1
T11	Security is challenging to implement, and very important	1	5	5
T12	No third party access	1	0	0
T13	No training needed	1	0	0

Technical Factor Total = 31

TCF = 0.6 + (0.01 * 31) = 0.91

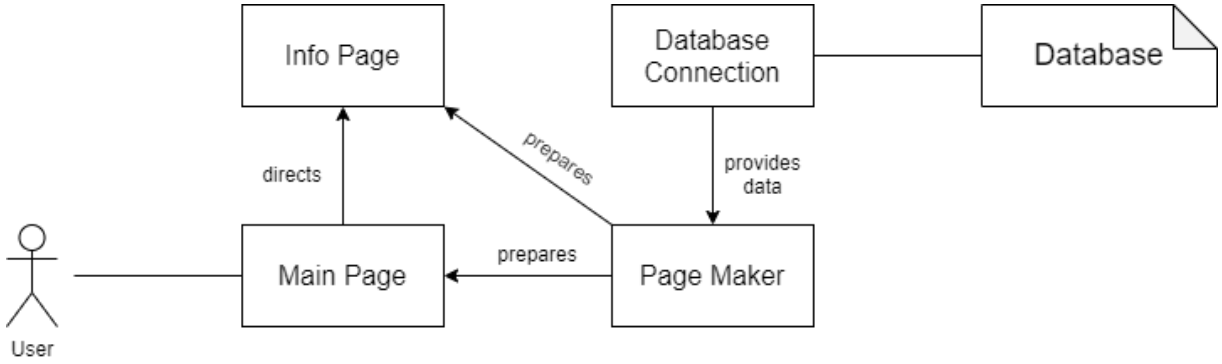
UCP = (11 + 155) * 0.91 = 151.06

Conceptual Model

Domain Analysis

Given the use cases ViewCarInfo, Checkout, CrudSalesData, and CrudInventoryData, it is inevitable that there will be a Main Page, Info Page, Checkout Page, Login Page and a Cart Page. Almost all of the information that will be displayed on these pages will be stored in the Database. Using a concept, namely Database Connection, the process of retrieving information from the database can be better visualized. An intermediary platform for making connections between the database and the front-end pages is also needed. A concept named Page Maker could be useful in depicting such a platform. For more advanced operations such as user login to the website or filling out forms, concepts such as Authenticator and Data Confirmer (for checking whether the input data fits to the requirements) are necessary. When a user purchases an item, a message, possibly an email, is needed for both the user and the vendor. For this purpose, concepts such as Actor Updater and Email Sender could be used, which will update the actors and send emails if necessary. Lastly and most importantly, making CRUD operations would require a concept, which could be called simply as CRUD Operator, to be active.

ViewCarInfo (UC-5)



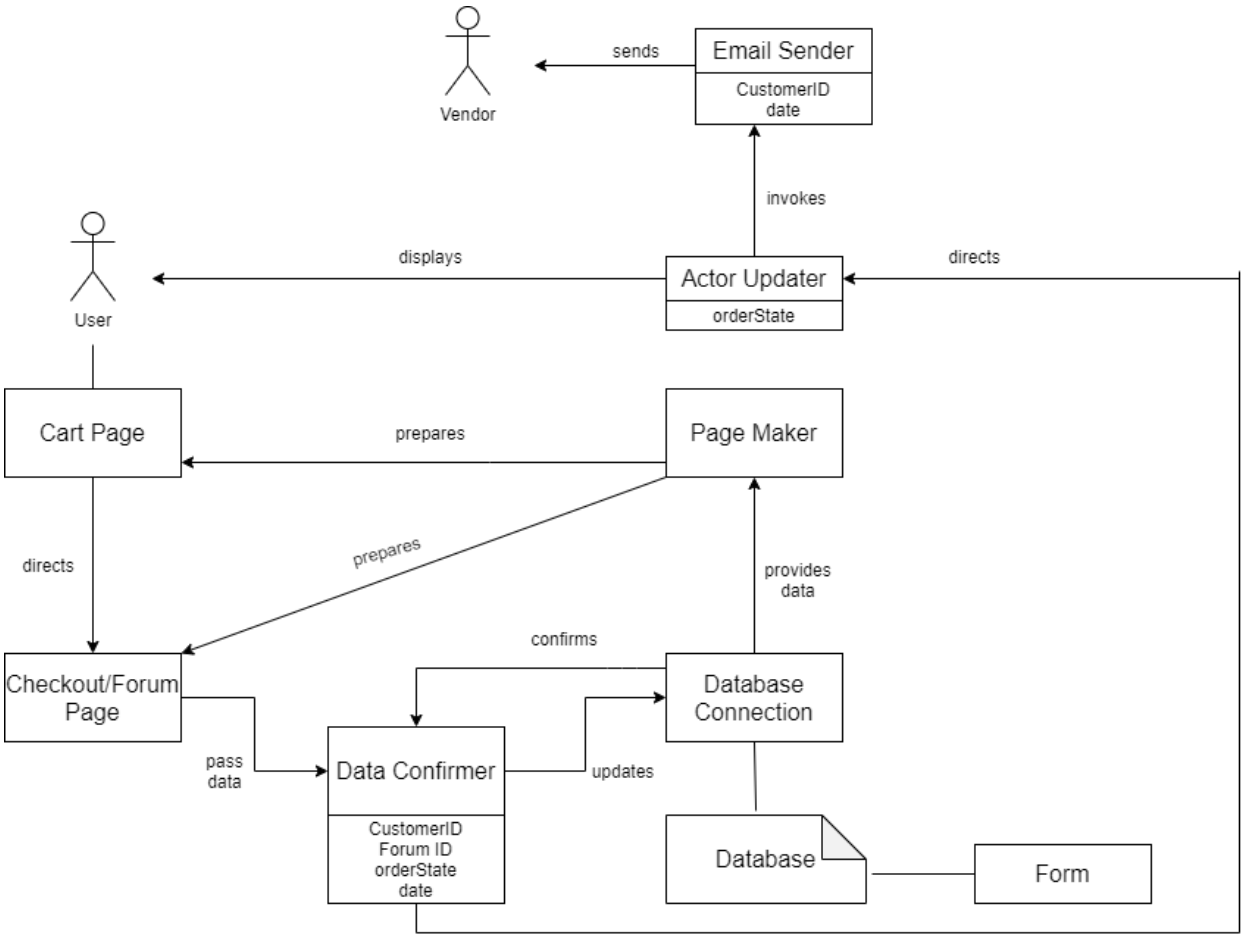
Concept Definitions

Responsibility Description	**Type**	**Concept Name**
Prepares a database query based on the specified cars.	D	Database Connection
The main HTML document that actor can use to interact with the application	K	Main Page
HTML document that shows information about the specified car in a conventional way.	K	Info Page
Render the retrieved records into an HTML document for sending to the actor's browser for display.	D	Page Maker

Association Definitions

Concept pair	**Association description**	**Association name**
Database Connection ↔ Page Maker	Database Connection passes the retrieved data to Page Maker to render them for display.	provides data
Page Maker ↔ Main Page	Page Maker prepares the associated page	prepares
Page Maker ↔ Info Page	Page Maker prepares the associated page	prepares
Main Page ↔ InfoPage	A click on a car card directs the actor from the Main Page to the Info Page of the particular car.	directs

Checkout (UC-9)



The user starts the checkout process on the Cart Page. Then, a button on the page directs the user to the Checkout/Forum page. On this page, the user is expected to fill out a forum, which will send data to the database after all the data is confirmed. After the confirmation, the user will get a message on the screen about the success of the purchasing process. Meanwhile, the vendor will get an email that his/her product has been purchased.

Concept Definitions

Responsibility Description	Type	Concept Name
Prepares a database query based on the specified request.	D	Database Connection
The main checkout page that actors can use to fill out the Form.	K	Checkout/Form Page
HTML document in which items selected by the actor is displayed	K	Cart Page
Render the retrieved records into an HTML document for sending to the actor's browser for display.	D	Page Maker
Send email to the Vendor about purchase with the required information.	D	Email Sender
Check the data on Form Page and confirm the given data if it fits the requirements.	D	Data Confirmer
Display a success page and update the user regularly on the purchase	D	User/Actor Updater
Container for the actor’s information that is necessary for a possible purchase.	K	Form

Association Definitions

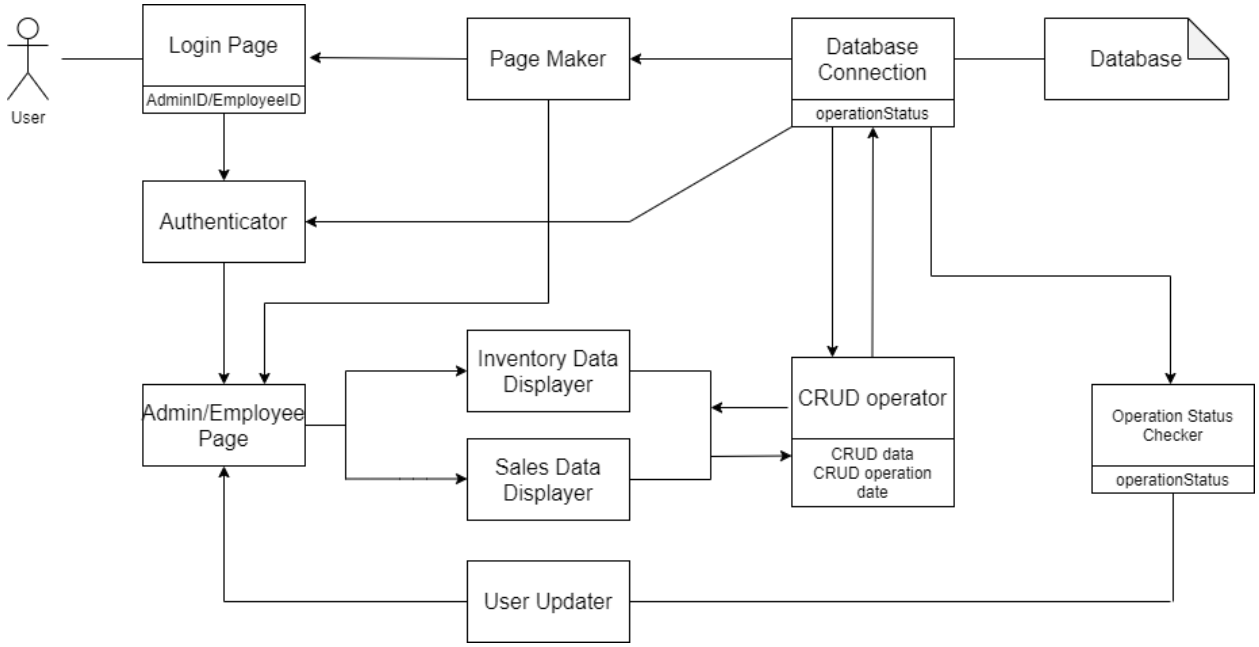
Concept pair	Association description	Association name
Database Connection ↔ Page Maker	Database Connection passes the retrieved data to Page Maker to render them for display.	provides data
Page Maker ↔ Checkout/Forum Page	Page Maker prepares the associated page	prepares
Page Maker ↔ Cart Page	Page Maker prepares the associated page	prepares
Cart Page ↔ Checkout/Forum Page	A click on an item in Cart Page directs the actor from Cart Page to the Checkout/Forum Page	directs

Checkout/Forum Page ↔ Data Confirmer	Checkout/Forum Page passes the data to the Data Confirmer	pass data
Data Confirmer ↔ Database Connection	Data Confirmer updates the Database Connection if data is confirmed	updates
Database Connection ↔ Data Confirmer	Database Connection confirms Data Confirmer on whether the updates have been successfully saved on the database.	confirms
Data Confirmer ↔ User/Actor Updater	Data Confirmer directs the actor from Checkout/Forum page to User/Actor Updater	directs
User/Actor Updater ↔ User	User/Actor Updater displays a success message/page to the actor	displays
User/Actor Updater ↔ Email Sender	User/Actor Updater activates Email Sender to make it perform its duty.	invokes
Email Sender ↔ Vendor	Email Sender sends an email to the Vendor, which includes related information about the purchase.	sends

Attribute Definitions

Concept	**Attribute**	**Attribute Description**
Data Confirmer	CustomerID	A specific ID that each Customer gets when the user account is created.
	FormID	A specific Id that each Form gets when a form is created.
	orderState	A value that describes the current state of an order
	date	The date in which order is given.
Actor Updater	orderState	The value that is changed after the order has been successfully given.
Email Sender	CustomerID	The same CustomerID that Data Confirmer contains. It is used for notifying the Vendor on the purchase.
	date	The date in which order is given.

CrudSalesData, CrudInventoryData (UC-10, UC-11)



The user will start the process by giving his/her credentials on Login Page. After it has been authenticated, the user will be directed to the Admin/Employee page, where he/she will have the privilege to access and update Inventory and Sales Data. When any update has been made to the database, the status of operation (the result of the query) is displayed to the user as a status message on the Admin/Employee page.

Concept Definitions

Responsibility Description	**Type**	**Concept Name**
Prepares a database query based on the specified cars.	D	Database Connection
The HTML document that will be used by the Admin/Employee to Login into the system	K	Login Page
Authenticates the user into the system by checking the given inputs and comparing them to the values storing the database.	D	Authenticator

The HTML document that is only used by Admin/Employee, that have the privileges of doing CRUD operations.	K	Admin/Employee Page
Displays Inventory Data by getting data from CRUD Operator and showing them to the actor	K	Inventory Data Displayer
Displays Sales Data by getting data from CRUD Operator and showing them to the actor	K	Sales Data Displayer
Creates SQL queries based on the request of the Admin/Employee Page. Transfers the results of the query to the Inventory and Sales Data Displayer	D	CRUD Operator
Checks the results of the operation and communicates them to the User Updater	D	Operation Status Checker
Render the retrieved records into an HTML document for sending to the actor's browser for display.	D	Page Maker
Display a success page and update the user regularly on the purchase	K	UserUpdater

Association Definitions

Concept pair	Association description	Association name
Database Connection ↔ Page Maker	Database Connection passes the retrieved data to Page Maker to render them for display.	provides data
Page Maker ↔ Login Page	Page Maker prepares the associated page	prepares
Page Maker ↔ Admin/Employee Page	Page Maker prepares the associated page	prepares
Login Page ↔ Authenticator	Login Page passes the data to the Authenticator	pass data
Database Connection ↔ Authenticator	Using the data from Login Page, Authenticator compares values stored in Database Connection and authenticates the user.	authenticates
Authenticator ↔ Admin/Employee Page	Authenticator directs user to the Admin/Employee Page after user credentials	directs

	have been verified	
Admin/Employee Page ↔ Inventory/Sale Data Displayer	Admin/Employee Page displays data from the database through several other concepts.	displays
Inventory/Sale Data Displayer ↔ CRUD Operator	Inventory/Sale Data Displayer connects to the CRUD Operator in order to get required values.	connects
CRUD Operator ↔ Database Connection	CRUD Operator makes queries to the Database Connection and gets the results	makes query
Database Connection ↔ Operation Status Checker	Database Connection informs the Operation Status about the results of the query	informs
Operation Status Checker ↔ User Updater	Operation Status Checker activates User Updater to make User Updater display message about the status of database	invokes
User Updater ↔ Admin/Employee Page	User Updater displays a message on the results of the query.	displays

Attribute Definitions

Concept	Attribute	Attribute Description
Login Page	AdminID/EmployeeID	A specific ID that each Admin/Employee gets when the user account is created.
CRUD Operator	CRUD data	Data that will be created, read, updated or deleted.
	CRUD operation	The operation that determines whether the data will be created, read, updated or deleted.
	date	The date in which operation was executed.
Operation Status Checker	operationStatus	A description about the status of the operation that was executed.

Traceability Matrix

USE CASE	PW	Database Connection	Main Page	Login Page	Info Page	Admin/Employee Page	Cart Page	Checkout/Forum Page	Email Sender	Authenticator	Page Maker	Inventory Data displayer	CRUD Operator	Operation Status Checker	Data Confirmer
UC1	3	X	X								X	X	X		
UC2	3	X	X								X	X	X		
UC3	3	X									X	X	X		
UC4	4	X		X					X	X	X				X
UC5	4	X	X		X						X	X	X		
UC6	4	X	X	X	X			X	X	X	X				X
UC7	4	X	X			X				X	X	X	X	X	X
UC8	4	X	X			X				X	X	X	X	X	X
UC9	5	X					X	X	X	X	X		X		X
UC10	5	X		X		X				X	X	X	X	X	
UC11	5	X		X		X				X	X	X	X	X	
UC12	4	X	X	X		X				X	X		X	X	
UC13	4	X	X	X		X				X	X		X	X	
UC14	4	X		X					X	X	X		X	X	X

System Operation Contracts

I. ViewCarInfo

A.

Operation:	displayResults(search, filter, sort)
Cross references:	ViewCarInfo
Preconditions:	● The customer is viewing the application on their browser
Postconditions:	● A search instance was created. ● Attributes of filter and sort have been modified.

B.

Operation:	carInfo()
Cross references:	ViewCarInfo
Preconditions:	● The customer is viewing the application on their browser
Postconditions:	● No notable post conditions. No instances or associations were formed or broken and no

	attributes have been changed.

II. Checkout

A.

Operation:	checkoutForm(customerID, formID)
Cross references:	Checkout
Preconditions:	• The customer has items in their cart • The customer has clicked the checkout button
Postconditions:	• A form was created and associated with the customer. • Basic attributes of the form were initialized.

B.

Operation:	submitForm(customerID, formID, date)
Cross references:	Checkout
Preconditions:	• The customer has items in their cart

	● The customer has clicked the checkout button
Postconditions:	● The form is placed into pending orders for the customer. ● The state of the order is changed upon approval. ● The order is associated with the customerID and the date.

III. CrudSalesData

A.

Operation:	readSalesData(sales:Sales)
Cross references:	CrudSalesData
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	● A connection is formed with the database and associated with sales data.

B.

Operation:	createSalesData(sales:Sales)
Cross references:	CrudSalesData
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	● A connection is formed with the database ● A new instance of sales is created. ● The new instance is associated with the database based on matching attributes

C.

Operation:	updateSalesData(sales:Sales)
Cross references:	CrudSalesData
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	● A connection is formed with the

	database. ● The attributes of a sales instance are modified. ● The modified sales instance is associated with the database based on matching attributes.

D.

Operation:	deleteSalesData(sales:Sales)
Cross references:	CrudSalesData
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	● A connection is formed with the database. ● An instance of sales is removed. ● The removed instance of sales is dissociated with the database.

IV. CrudInventoryData

A.

Operation:	readInventoryData(inventory:Inventory)
Cross references:	CrudInventoryData
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	● A connection is formed with the database and associated with inventory data.

B.

Operation:	createInventoryData(inventory:Inventory)
Cross references:	CrudInventoryData
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	● A connection is formed with the database. ● A new instance of inventory is created.

	• The new instance is associated with the database based on matching attributes.

C.

Operation:	updateInventoryData(inventory:Inventory)
Cross references:	CrudInventoryData
Preconditions:	• Admin/employee is logged in • Admin/employee is viewing application interface for CRUD changes
Postconditions:	• A connection is formed with the database. • The attributes of an inventory instance are modified. • The modified inventory instance is associated with the database based on matching attributes.

D.

Operation:	deleteInventoryData(inventory:Inventory)

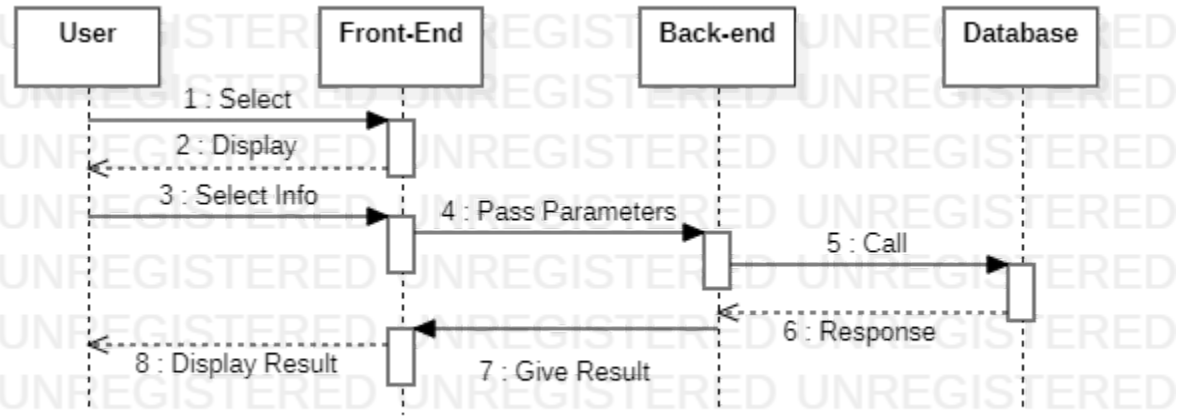
Cross references:	CrudInventoryData
Preconditions:	● Admin/employee is logged in ● Admin/employee is viewing application interface for CRUD changes
Postconditions:	● A connection is formed with the database. ● An instance of inventory is removed. ● The removed instance of inventory is dissociated with the database.

Data Model and Persistent Data Storage

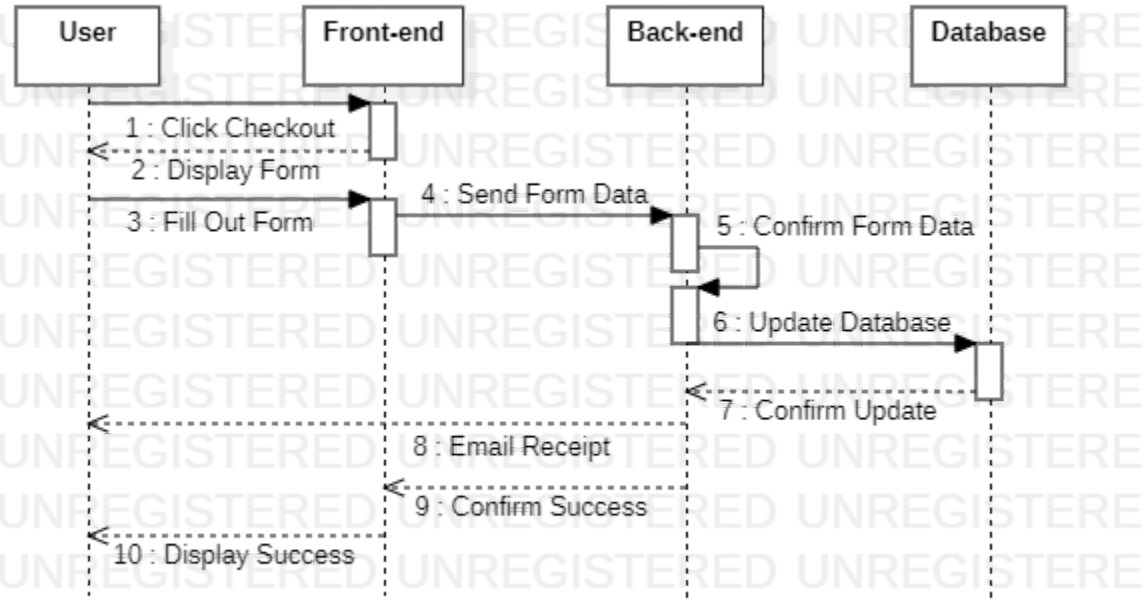
We will be using a relational database to store and reference data as well as deploying our web application using Heroku. The use of these will ensure that data can outlive a single execution of our web application. Objects that will be stored in the database include: sales, inventory, employees, customers, and cart.

Interaction Diagrams

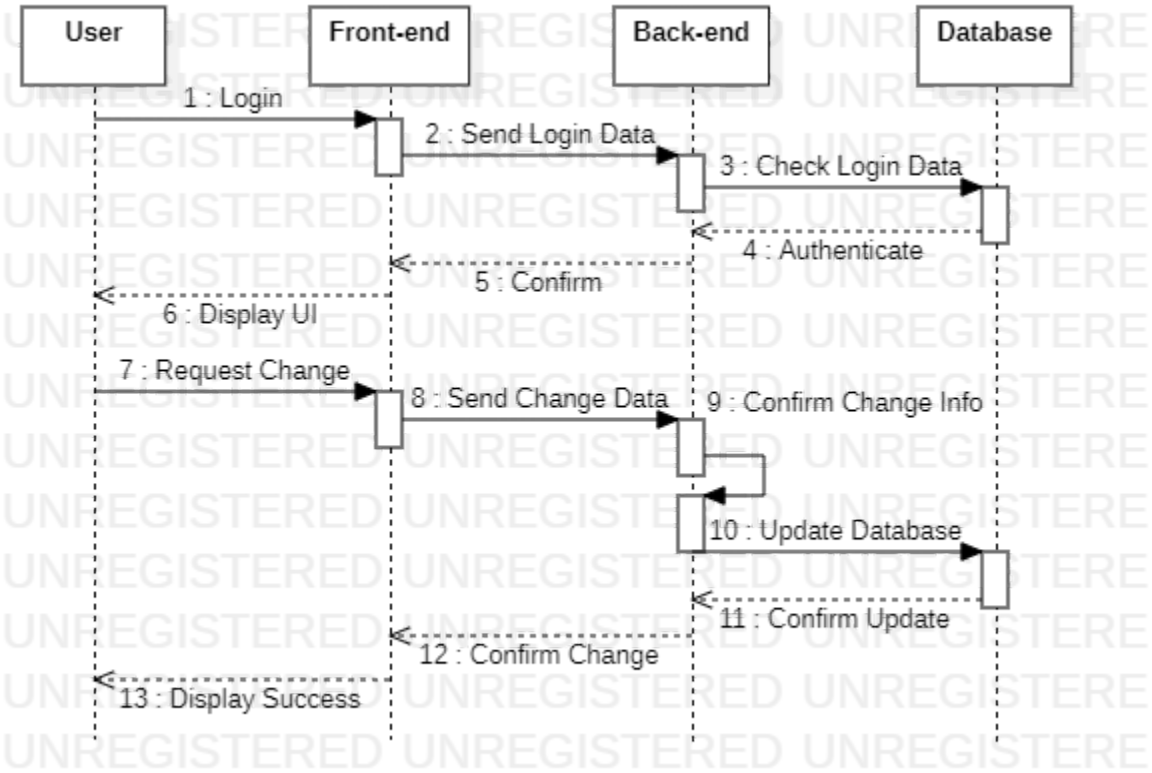
ViewCarInfo (UC-5)



Checkout (UC-9)

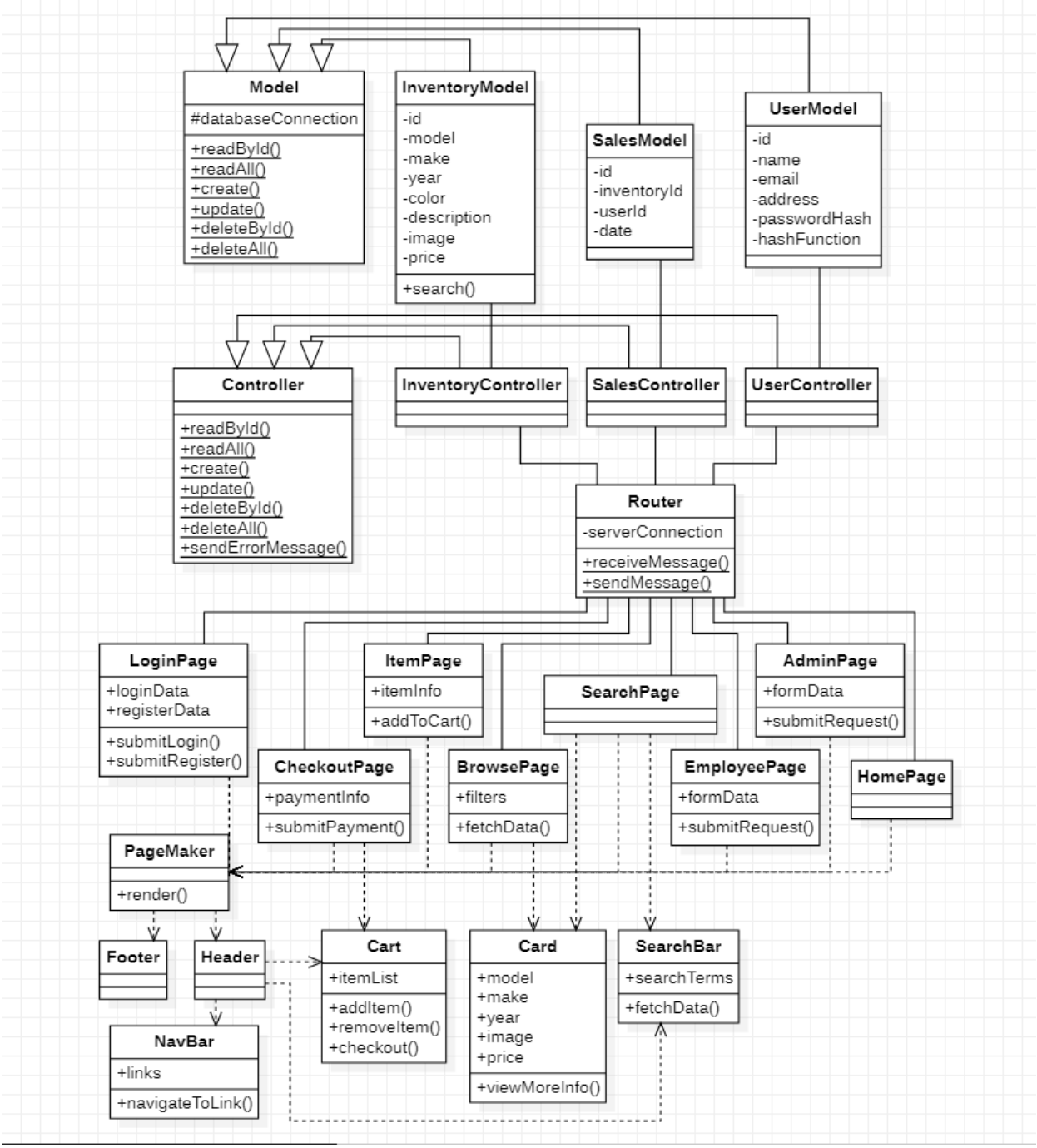


CrudSalesData, CrudInventoryData (UC-10, UC-11)



Responsibilities were given based on general guidelines for web application development. The user only interacts with the front-end, obscuring the moving gears of the back-end and database. This allows data entered by the user to be parsed correctly, and no malicious or accidental errors to be created.

Class Diagram



Interface Specifications

Data Types

Router: The router is the central hub that the website will use to connect with the server and to send and receive messages.

Inventory Controller: Mainly links to the central Controller and the Inventory Model.

Sales Controller: Mainly links to the central Controller and the Sales Model.

User Controller: Mainly links to the central Controller and the User Model.

Controller: Reads, creates, updates, deletes, or informs the user of an error using the Inventory, Sales, and User controllers.

Inventory Model: Will store the inventory of the car's features, such as the make and model, the price, descriptions of features, as well as a search function for the user to search the inventory.

Sales Model: Mainly tracks the sales and updates both the inventory and user when changes are made.

User Model: Stores user information such as username, password, email address, and any other information.

Model: Connects to a database and manages all information sent by the Inventory, Sales, and User models.

Login Page: Web page that displays login and registration information from users.

Item Page: Web page that displays information related to items, in this case vehicles.

Search Page: Web page that displays information related to user searches.

Admin Page: Web pages accessible only to site administrators showing various form data and features.

Checkout Page: Web page that displays checkout information after a user has placed an order.

Browse Page: Web page that allows users to browse the catalog.

Employee Page: Web pages accessible only to employees allowing them limited access to make necessary updates.

Home Page: Web page that is the first page a user will see when first accessing the website.

Page Maker: Renders all of the web pages using the provided information.

Footer: Static bottom information of the web page showing contact information and various links.

Header: Static top information of the web page showing links to parts of the website and necessary user information,

Nav Bar: Static navigation bar on the header of the page allowing the user easy access to the rest of the website.

Cart: Storage of user checkout information that displays on the header.

Card: Storing information and images related to vehicles.

Search Bar: Static bar on the web page allowing users to search the website.

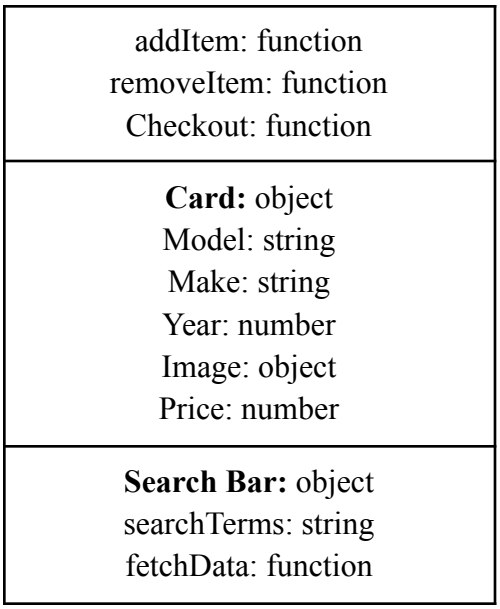
Operation Signatures

Router: object serverConnection: function receiveMessage: function sendMessage: function
Inventory Controller: object
Sales Controller: object
User Controller: object
Controller: object readByID: number

readAll: string create: string update: string deleteByID: number deleteAll: string sendErrorMessage: string
Inventory Model: object Id: number Model: string Make: string Year: number Color: string Description: string Image: object Price: number Search: function
Sales Model: object Id: number inventoryId: number userId: number Date: string
User Model: object Id: number Name: string Email: string Address: string passwordHash: string hashFunction: function
Model: object readByID: number readAll: string create: string update: string deleteByID: number deleteAll: string
Login Page: object

loginData: string registerData: string submitLogin: function submitRegister: function
Item Page: object itemInfo: string addToCart: function
Search Page: object
Admin Page: object formData: string submitRequest: function
Checkout Page: object paymentInfo: string submitPayment: function
Browse Page: object Filters: object fetchData: function
Employee Page: object formData: string submitRequest: function
Home Page: object
Page Maker: object Render: function
Footer: object
Header: object

Nav Bar: object Links: string navigateToLink: function
Cart: object itemList: string


addItem: function
removeItem: function
Checkout: function
Card: object
Model: string
Make: string
Year: number
Image: object
Price: number
Search Bar: object
searchTerms: string
fetchData: function

Traceability Matrix

Class	Database Connection	Main Page	Login Page	Info Page	Admin/Employee Page	Cart Page	Checkout/Form Page	Email Sender	Authenticator	Page Maker	Inventory Data Displayer	CRUD Operator	Operation Status Checker	Data Confirmer
PW	5	3	3	4	4	3	3	1	2	5	4	4	1	2
Model	X	X	X	X	X	X	X		X		X	X		
InventoryModel	X	X		X	X	X	X				X	X		
SalesModel	X			X	X		X					X		
UserModel	X		X		X	X	X		X			X		
Controller	X	X	X	X	X	X	X	X	X		X	X	X	X
InventoryController	X	X		X	X	X	X				X	X		X
SalesController	X				X		X					X		X
UserController	X		X		X	X	X	X	X			X		X
Router	X	X	X	X	X	X	X	X	X			X	X	
PageMaker		X	X	X	X	X	X			X	X			
LoginPage			X											
CheckoutPage							X							
ItemPage				X										
BrowsePage											X			
SearchPage											X			
EmployeePage					X							X		
AdminPage					X							X		
HomePage		X												
Footer										X				
Header										X				
NavBar										X				
Cart						X								
Card											X			
SearchBar										X				

Some of the classes evolved from the implementation of the MVC (Model, View, Controller) concept of web development. This type of design concept divides the file structure into three categories. The controller being the bridge between model (data) and view (front-end) and controlling the flow. Classes such as Model, InventoryModel, SalesModel, and UserModel are under the “Model” portion of the design. Similarly, any class with a name containing “controller” is part of the “Controller” portion (Router, and PageMaker class being the odd one out). This leaves all the rest of the classes in the “View” category, which is all user-facing.

The vast majority of the concepts were matched with a category of the MVC design class-structure in a simple manner. For example, the “Database Connection” and “CRUD Operator” concepts only interact with classes in the “Model” and “Controller” categories, as they only focus on behind-the-scenes work. Other classes were created simply to satisfy a singular concept. The “PageMaker” concept for example did not fall directly into any category, so a class was made to ensure the concept was included.

User Interface Design and Implementation

I. Modifications from Mock Ups

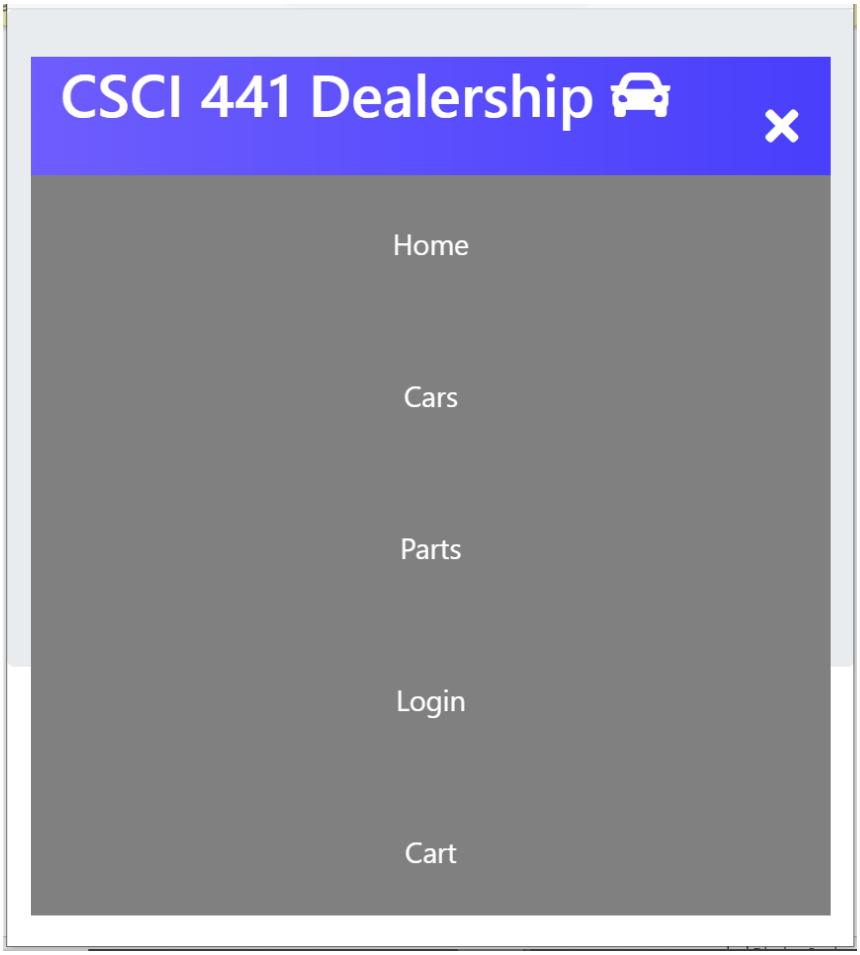
We still have a lot of implementation to do, but we will look at the difference between the mockups in report one with what we have implemented so far.

A. Navigation Menu

The Navigation menu has roughly the same functionality as the one in the mockup except for one thing, there is no search bar in our implementation. This will increase the user effort as they will not be able to search for a specific product directly from the navigation menu. We will most likely implement a search bar in the future to reduce the user effort and make it the same as the mockup.

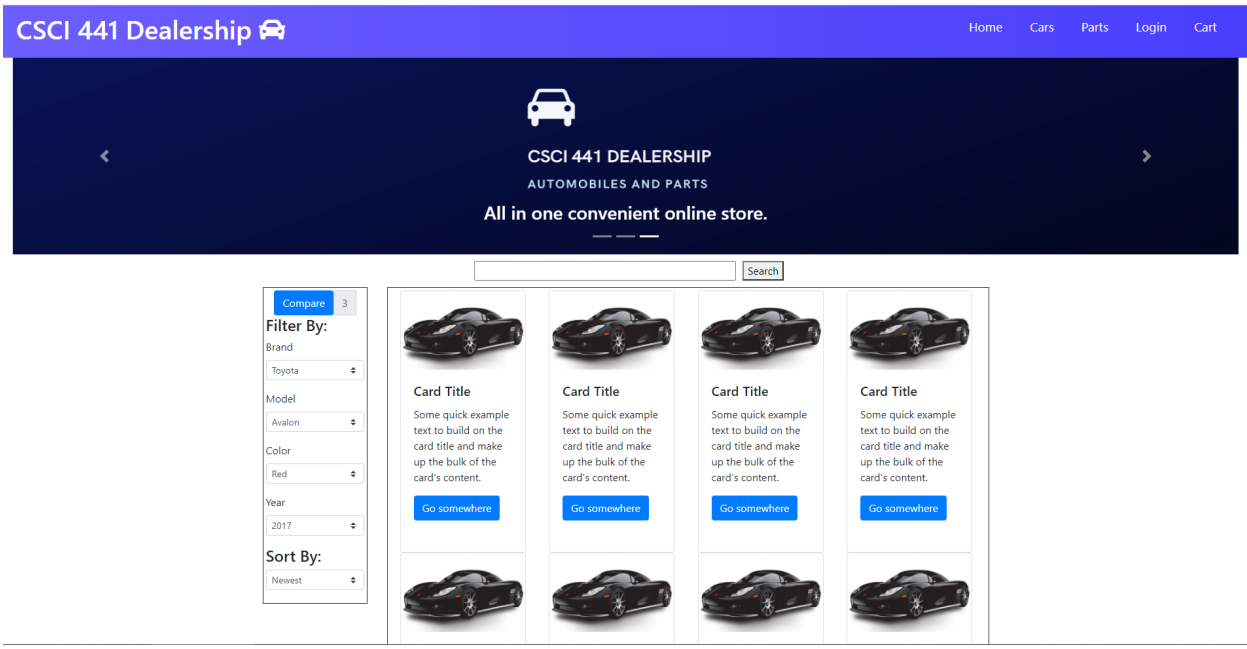
One improvement from the mockup is that there is styling for mobile devices. The navigation menu options are hidden for mobile devices until the user clicks the bars in the top right corner. From there, they have the same options that someone using a desktop has, just in an easier to navigate form factor. This does increase

the user effort by one click because they have to click the bars to see the page links. However, this feature increases the ease of use because it ensures that the users on mobile devices don't have to click tiny buttons from the navigation bar being shrunk down significantly.



B. Homepage

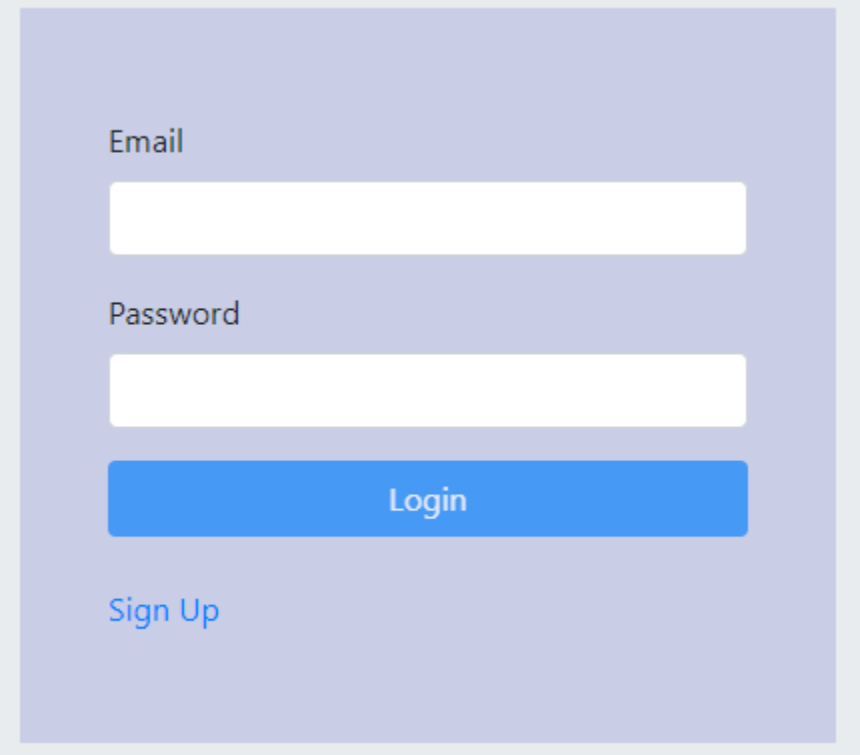
Our implementation of our homepage is a vast improvement from the one seen in the mock up. There are many different components to this homepage that will reduce the user effort. We have featured products that the user can click on to take them directly to the product page, rather than them having to go to a product listing page and then click on the product. There is a search bar at the top of the page so that a user can find any specific product that they want. There is a compare button that the user can use to easily compare two different products without having to go to each individual product page. Finally, there are filter and sort inputs that reduce the user's effort in finding a product that matches their criteria.

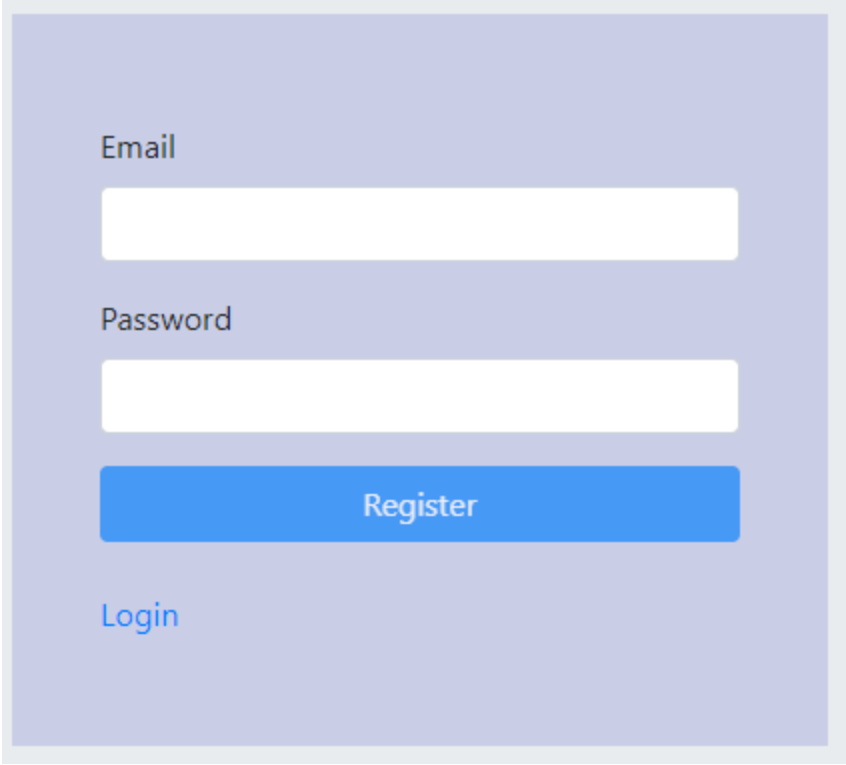


C. Login/Register

The Login page looks very familiar to the one that was shown in the mockup, a very simple form that asks for the email address and the password. There is one

difference though; there is a link below the login button that the user can click to create an account. After the user clicks the sign up link, they will be taken to another form where they can enter an email and password to create an account. From there, they can login with their newly created account.




Email
Password
Register
Login

Design of Tests

Test Cases

Test-case Identifier: TC-5 **Use Case Tested:** UC-5 **Pass/fail Criteria:** The test passes if the user clicks on a car info button on the homescreen and successfully gets an information page about the selected car. **Input Data:** the car id, obtained by the user click	
Test Procedure	**Expected Results**
Step 1. Click on a car info button on the car card	System directs the user to a new page which displays information about the selected car.

Test-case Identifier: TC-9 **Use Case Tested:** UC-9 **Pass/fail Criteria:** The test passes if the user clicks on the checkout button, fills out a form, and gets a success message. **Input Data:** the car id, form data given by the user	
Test Procedure	**Expected Results**
Step 1. Click on checkout button, fill out the form correctly Step 2. Click on the checkout button, fill out the form incorrectly	System displays a success message informing the user. System displays an error message, requesting the user to try again.

Test-case Identifier: TC-10, TC-11 **Use Case Tested:** UC-10, UC-11 **Pass/fail Criteria:** The tests passes if the operation requested by the user is successfully done **Input Data:** a CRUD operation	
Test Procedure	**Expected Results**
Step 1. Perform a valid CRUD operation on the system Step 2. Perform an invalid CRUD	System displays a success message, the result of the CRUD operation can be instantly viewed on the system. System displays a fail message. The user can try

operation	to perform a different CRUD operation.

Test Coverage

There are mainly four code coverage criteria, namely, equivalence testing, boundary testing, control-flow testing and state-based testing. Due to the relatively simple design and implementation of the program and most of the testing strategies being overtly sophisticated, (thus making the testing process unnecessarily long and nonsense) only some of the strategies could be used when testing the program.

There are several important points that require attention in the application. These points can be divided into two. Firstly, the program should be seen as a product that should have a functional design. In this perspective, one should consider the transition of a program from one state or page to another, general structure of a program, and the usability of the program. Secondly, the program could be seen from a “data” point of view. This aspect is where most of the software testing strategies and methods could be applied and tested.

Equivalence Testing

The partitioning of programs is not very diverse in our application. Since the program mostly deals with simple data and the feedback to the this data is either success or failure, partitioning the value space into one valid (successful) and one invalid (failure) equivalence class, as in boolean values of FALSE and TRUE, seems to be the only plausible solution, if an equivalence testing will be applied.

Boundary Testing

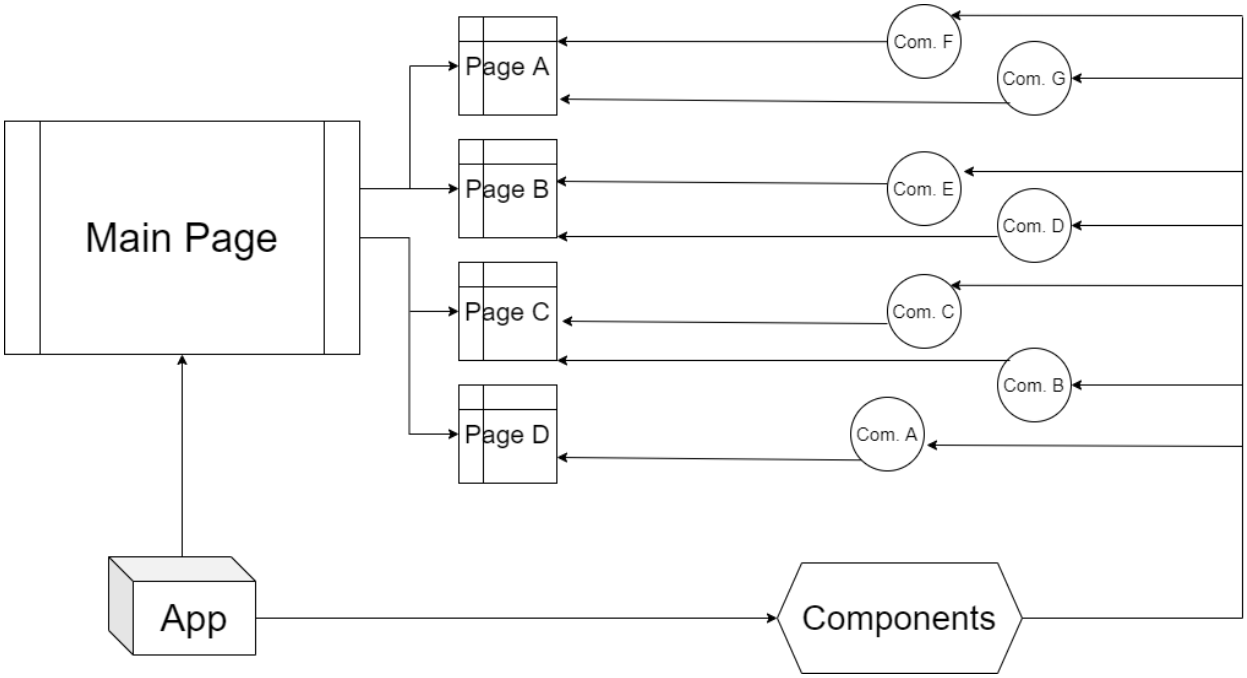
Boundary testing can be viewed as a part of equivalence testing in which boundaries, or “edges” are tested. Such boundaries in our programs could only be null values, or empty strings, which could easily be implemented as a part of equivalence testing.

Control Flow Testing

In order to make sure that the user does not face any errors, or software bugs while using the program, it is crucial to eliminate any possible situation that may lead to unexpected or unforeseen circumstances. Control flow testing allows a tester to reduce the number of bugs by minimizing the events that may create a problem for the user. Therefore, except the Path Coverage part of the control flow testing, which may not be applicable in all situations, all other coverages, such as Statement Coverage, Edge Coverage, and Condition Coverage could be applied.

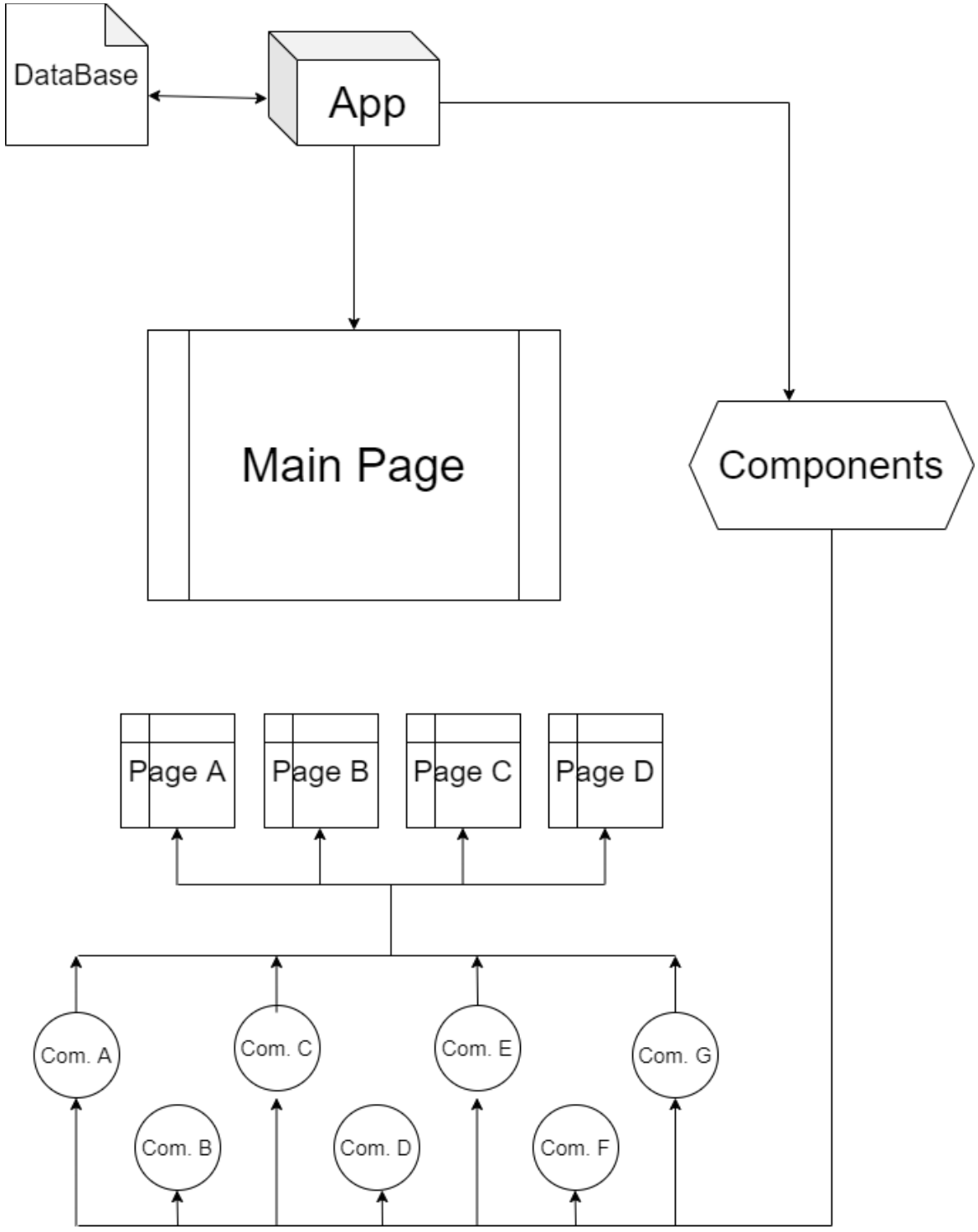
Integration Testing

Before mentioning about the Integration Testing, let’s have a look at the general conceptual structure of the application.



Conceptually, the main part of the program is a file named App, in which all the data communication is done. Main page is the second most important part of the program, which directs users to other pages. All the other pages are having a third place of importance and they contain components. Lastly, we have components, which are the smallest pieces of the program, also having the least level of importance.

However, from the point of view of a software engineer or a software tester, the program can be restructured (viewed) in a much better way, as such:

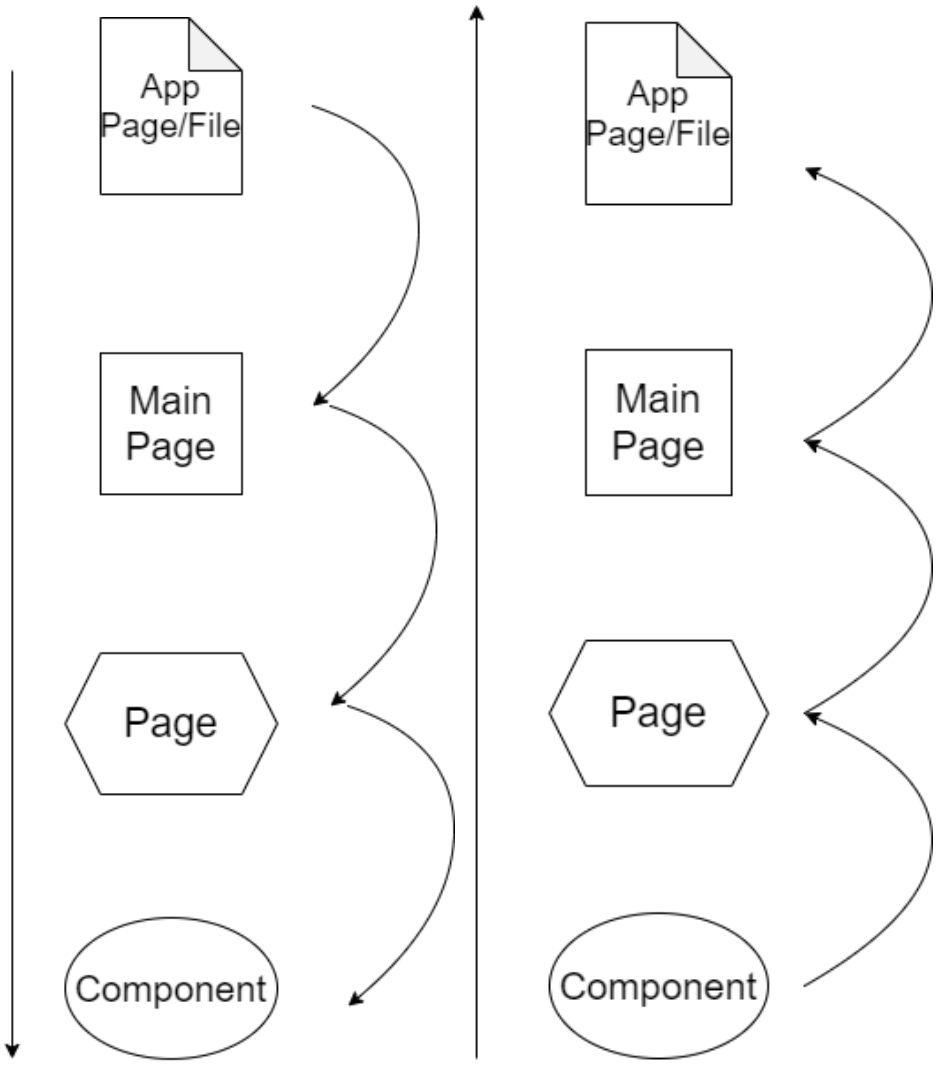


When we look from this perspective, it is easier to have an idea of which kind of testing strategies and methods could be used.

Horizontal Integration Testing

Horizontal Integration Testing includes different approaches, such as bottom-up integration, top-down integration and sandwich integration, which combines both bottom-up and top-down integration into one by having a middle level “sandwiched” by top and bottom. Sandwich integration may not be efficient for this program but bottom-up integration and top-down integration could easily and effectively be used.

Top-Down Integration **Bottom-up Integration**



Vertical Integration Testing

Having only 14 user cases, vertical integration testing strategies doesn't seem to be a very plausible testing solution for a project of such a range.

Security Testing

There are, in total, two user cases in which security testing could be done. The use case, SignUp (UC-4) and Authenticate(UC-14), are two critical components of the program in which security leaks could occur. The authentication part of the program should work flawless in order to prevent security issues and smoothness of the program. The program also includes three user accounts: Admin, Employee, and Customer, with each user account having access to different privileges. Any of the mistakes that are made while implementing these user accounts may lead to unacceptable circumstances, in which a Customer account can invade privacy of other customers, creating ethical problems, or access to employee accounts, changing the internal structure of the program in an irreversible way.

Manual Testing

An alternative to all different testing approaches could be using manual testing as a primary method of testing. Considering the low number of Use Cases (with 14) and the results that could be obtained (a boolean value of TRUE or FALSE), manual testing could be a probable and clearly shorter method of testing. The use cases could be tested by the developers and some of them even by the non-developers. The fact that the testing process does not require any coding is also a plus.

Project Management and History of Work

Merging the Contributions from Individual Team Members

The team has tackled portions of the reports in sections individually while sharing their progress on a shared google drive. The final product has been merged through attaching all of the portions together and smoothing out any inconsistencies/formatting differences. The team then goes through the report in a group call and ensures that everything is correct before it is finalized. No issues were encountered, everyone had their portions done in time so that no rushing was needed by anyone last minute.

Project Coordination and Progress Report

Thus far, everything is going to schedule as a group. The project is set to be completed in time for the final demo with full functionality per the report. SignUp, Search, Sort, ViewCarInfo, CrudInventoryData, and AddEmployee are all fully implemented use cases. The rest of the use cases will be completed by the final demo. No work planned for this semester will be put into the “future work” category..

The hosting, database connection, and react setup have all been completed. The navigation/header, home page, and login/sign-up components have been coded as well. API endpoints and database models are created. The admin interface has been completed and is under

work to fully connect to the backend. React router has added functionality to the web application. Inventory cards, sort, search, and view functions all are in place as well. Currently the parts and sales databases are under development. The compare functionality is underway as are the cart and checkout functions.

Key Accomplishments:

- The team learned a brand new framework in order to pursue a more in-depth project (React.js).
- A back-end was created via Node.js which also increased the difficulty.
- The team lost a member halfway through the semester, yet is still on course to finish the project on time.
- No one has missed the standing weekly meeting, or failed to complete any work expected of them.
- Team communication has been impressive, with no last minute rushing for any deliverable.
- The team has helped each other to complete challenging portions of the project.

Future Work:

All work for the project should be completed by the end of the class.

References

V12data.com. *90% of Car Shoppers Want an Buying Process Online | V12*. [online] Available at: https://v12data.com/blog/90-car-shoppers-prefer-dealership-where-they-can-start-buying-process-online/. [Accessed 8 Feb. 2021].

Tutorialspoint.com. (2019). *Software Requirements.* [online] Available at: https://www.tutorialspoint.com/software_engineering/software_requirements.[Accessed 10 Feb. 2021].

www.tutorialride.com. (n.d.). *User Interface*. [online] Available at: https://www.tutorialride.com/software-architecture-and-design/user-interface.htm#:~:text=There%20are%20eight%20characteristics%20considered%20while%20making%20a [Accessed 10 Feb. 2021].

www.bestbuy.com. (n.d.). *Best Buy | Official Online Store | Shop Now & Save*. [online] Available at: https://www.bestbuy.com. [Accessed 10 Feb. 2021].

www.bigoautollc.com. (n.d.). *Big O Auto* [online] Available at: https://www.bigoautollc.com/details/used-2016-acura-tlx/71123677. [Accessed 10 Feb. 2021].

docs.erpnext.com. (n.d.). *Product Page*. [online] Available at: https://docs.erpnext.com/docs/user/manual/en/website/product-page [Accessed 10 Feb. 2021].

ReviewPro. (n.d.). *mock-up-desktop-and-mobile*. [online] Available at: https://www.reviewpro.com/blog/reviewpro-continues-innovate-guest-experience-improvement/mock-up-desktop-and-mobile/ [Accessed 10 Feb. 2021].

Saring, J. (2019). *5 Ways to Improve UI Consistency*. [online] Medium. Available at: https://blog.bitsrc.io/5-ways-to-improve-ui-consistency-99013bf20417 [Accessed 10 Feb. 2021].

Apple. (2011). *iPhone*. [online] Available at: https://www.apple.com/iphone/. [Accessed 10 Feb. 2021].

SolarWinds. (2015). *Web Application Process Flow in Oracle Databases*. [online] Available at: https://logicalread.com/application-flow-oracle-mc08/#.YCfXl2hKjb0 [Accessed 10 Feb. 2021].

cas.fhsu.edu. (n.d.). *Login - CAS – Central Authentication Service*. [online] Available at: https://blackboard.fhsu.edu/ [Accessed 10 Feb. 2021].