**Software Design Document**

**Green Team**

**Towson University, COSC412**

**Date: 04/23/2020**

**Table of Contents**

1. **Introduction**
   1. **Purpose**
   2. **Scope**
   3. **Definitions and Acronyms**
2. **References**
3. **Decomposition description**
   1. **Module decomposition**
      1. **Module 1 description**
      2. **Module 2 description**
   2. **Data decomposition**
      1. **Entity Relationship Diagram**
   3. **Graphical Representation**
      1. **Use case**
      2. **Sequence Diagram**
      3. **Class Diagram**
      4. **Activity Diagram**
4. **Dependency description**
   1. **Intermodule dependencies**
   2. **Interprocess dependencies**
   3. **Traceability Table**
5. **Interface description**
   1. **User Interface**
      1. **Menu screens**
      2. **Input and Output screens**
   2. **GUI Transition Diagram**
   3. **User manual**
6. **Introduction**
   1. **Purpose**

This Software Design Description (SDD) describes the detailed structure of the components of the Morisette web app and the precise implementation details required to satisfy the requirements as specificted in the Software Requirements Specification (SRS).The intended purpose of this software requirements specification is to provide the stakeholders and the developers with a proper outline of the design and needs of the software that is required to be built. In this document, information regarding the flow of the development process and other accompanying information that provides a clear and informative perspective of the product agreed upon by all stakeholders will be thoroughly detailed. Additionally, this document contains unambiguous requirements for the development team handling this project to ensure the agreed upon SRS and the final product are one-to-one.

* 1. **Scope**

Morisettee is a web application that offers customers a place to buy new vehicles as well as give customers the opportunity to trade in their used vehicles for a cash value which will be added to their account and applied to their purchase as a credit. The application allows customers to make a purchase using debit or credit card.  In addition, registered customers can apply for financing options on the site by filling out a form with their financial history. Once the application is signed and submitted customers are guaranteed that they will receive a fair down payment and monthly payment estimate within a 24-hr period if they qualify. Otherwise, customers will receive a decline notice with an explanation. Once down payment is received customers will get a notification with their delivery or pick up instruction and status.

* 1. **Definitions and Acronyms**

|  |  |
| --- | --- |
| MangoDB | A nosql database program used for storing, finding, and changing information |
| collection | A grouping of **MongoDB** documents |
| Class | An extensible program code for creating objects, providing initial values for state and implementations of behavior. |
| Object | Refers to a particular instance of a class. |
| Parameter | Variables used to pass information between functions or procedures |
| Method | A named section of a program that performs a specific task. |
| Table | A collection of related data held in a **table** format within a database |
| Cash | Payment through cash or credit card |
| Parameter | Variables used to pass information between functions or procedures |
| Method | A named section of a program that performs a specific task. |
| Table | A collection of related data held in a **table** format within a database |
| AD | Activity Diagram |
| MS | Menu Screen |
| IOS | Input/Output Screen |

1. **References**

Izzo, Miles, et al. *Software Design Document (SDD) Event Driven DIS PDU Logger (EDDIS System)*. 2002, *Software Design Document (SDD) Event Driven DIS PDU Logger (EDDIS System)*, portal.unimap.edu.my/portal/page/portal30/Lecture%20Notes/KEJURUTERAAN\_KOMPUTER/Semester%202%20Sidang%20Akademik%2020112012/EKT420%20Software%20Engineering/Example%20of%20Software%20Design%20Document(SDD)/EDDISS.pdf.

McElrath, Rex. *XML Legal Document Utility Software Design Document*. 2007, *XML Legal Document Utility Software Design Document*, robotics.ee.uwa.edu.au/courses/design/examples/example\_design.pdf.

Pressman, Roger S., and Bruce R. Maxim. *Software Engineering: a Practitioner's Approach*. McGraw-Hill Education, 2020.

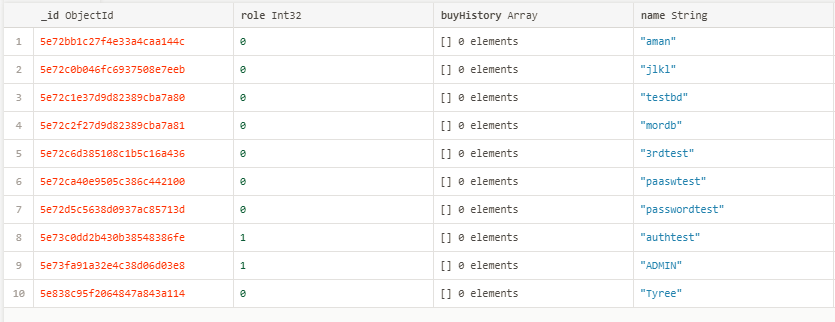
Reaves, Michael J. *Software Design Description Version 1.1*. 2003, *Software Design Description Version 1.1*, www.slideshare.net/peny\_mg/sdd-software-des-sample.

1. **Decomposition description**
   1. **Database decomposition**

Morissette uses a no-SQL external database through MangoDB. The database will hold all the data related to our web application. There are multiple related Collections used to hold data in our cluster each identified with a uniquely generated ID. Currently we have 6 tables(collections). Those are User, Vehicle, Finance, Trade-In, Order, Cart.

* + 1. **User Tables**

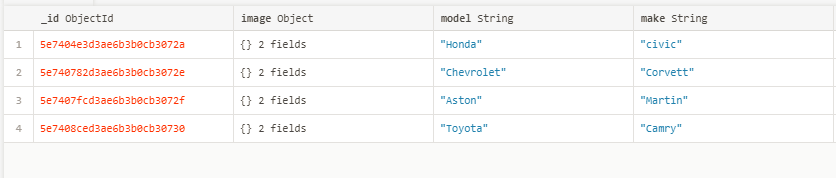
This table stores all information related to all our Customers and Admins. This   includes a unique id number for each user’s account, Name, Buy History, Email, Password and their respective role (Admin, Customer, Trader) and their password.

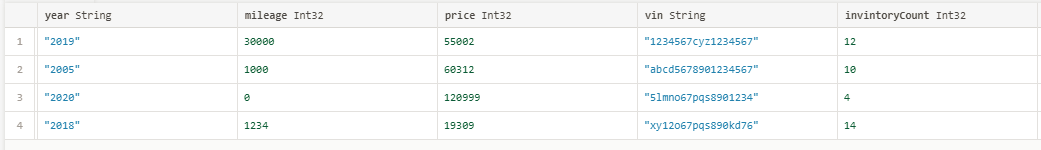




* + 1. **Vehicle Table**

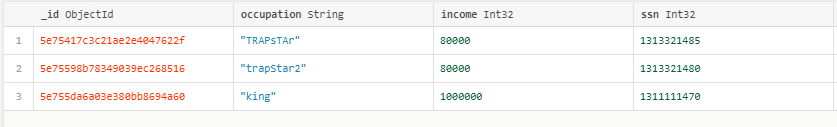
This table stores all information related to all the vehicles being sold. This includes a unique id number for each Vehicle, an image, Vehicle model, make, year as well as the Price and inventory count for each car.



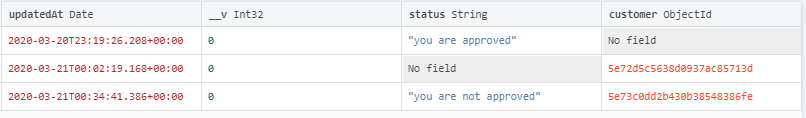


* + 1. **Finance Table**

This table stores the financial information of a customer who applies for vehicle financing through the Moressitee. All our customers' applications with their respective information will be saved here. This includes Customers unique id from the **User Table**, occupation, income, social security number, date-of-birth, credit score description and employers name. Once admin reviews the lone status of the application is marked as (Approved or Denied).

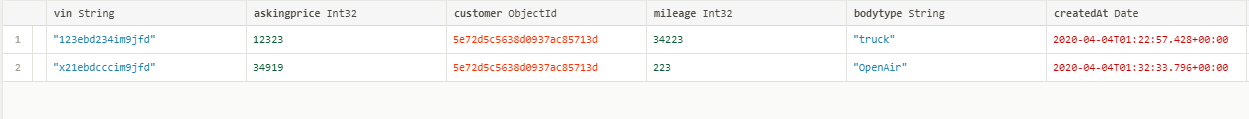
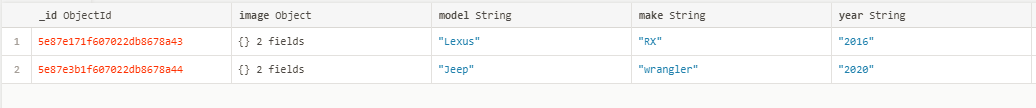




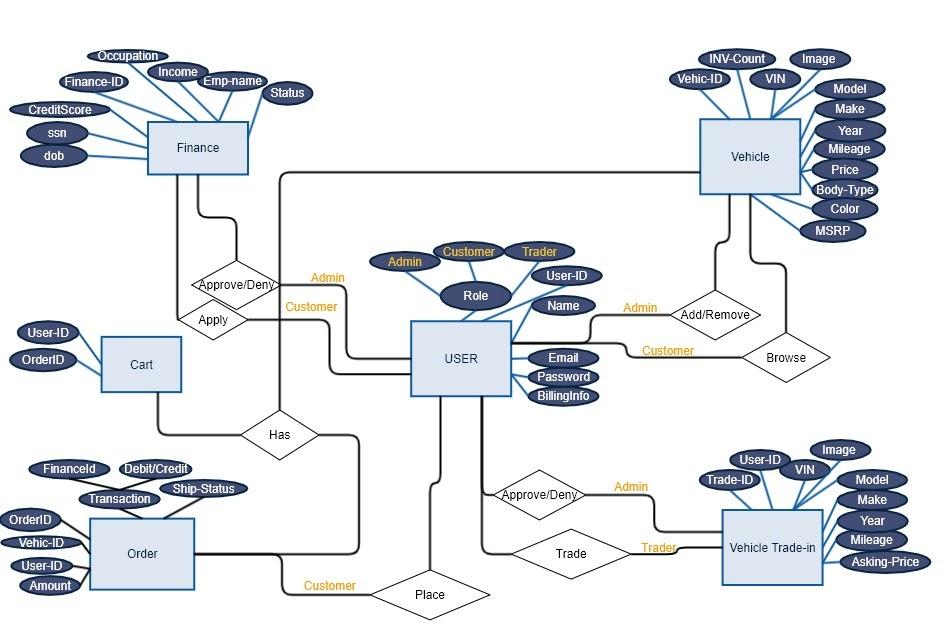


* + 1. **Vehicle Trade-In Table**

This table stores the vehicle information of a customer who submitted their vehicle to be traded in Morissette. All our customers' vehicle information\will be saved here. This includes Customers unique id from the **User Table**, vehicle image, model, mileage, year, body type and the asking price. Based on the information submitted here admin may grant or deny their asking price and give customers a credit on their purchase.

* 1. **Data decomposition**
     1. **Entity Relationship Diagram**



* 1. **Graphical Representation**
     1. **Use case**

Use Case <uc\_000\_001>

|  |
| --- |
| **Description:** |
| Create an account |
| **Actors:** |
| Guest Customers |
| **Priority:** |
| High |
| **Pre-condition:** |
| * User is on our website |
| **Flow of Events:** |
| 1. User clicks register 2. User enters identifying personal details    1. Username    2. Email    3. Password    4. First and Last Name    5. Phone number (?) 3. Send confirmation email 4. User clicks confirmation email 5. Redirect user to their user homepage (?) |
| **Alternative Flow (if it exists):** |
| 1. User clicks sign in 2. Website directs user to register page 3. Register through social media login 4. Redirect user to their user homepage (?) |

Use Case <uc\_000\_002>

|  |
| --- |
| **Description:** |
| Login to website |
| **Actors:** |
| Registered Customer  Admins |
| **Priority:** |
| High |
| **Pre-condition:** |
| The user has already registered an account |
| **Flow of Events:** |
| 1. User clicks log-in button 2. User enters identifying personal details    1. Users enters a username    2. User enters a unique password associated with their account 3. Redirect user to their user homepage |
| **Alternative Flow (if it exists):** |
| 1. User enters wrong username    1. User will be asked to re-enter username 2. User enters wrong password    1. User will be asked to re-enter password |

Use Case <uc\_000\_003>

|  |
| --- |
| **Description:** |
| Browse vehicle listings |
| **Actors:** |
| Guest Customers  Registered Customers  Admins |
| **Priority:** |
| High |
| **Pre-condition:** |
| * User is on our website |
| **Flow of Events:** |
| 1. User arrives at website 2. User enters parameters of related vehicles    1. Make    2. Model    3. Engine Type    4. Zip Code 3. Website displays related listings 4. User can click on a vehicle and view information about it 5. User can move to next pages (if they exist) |
| **Alternative Flow (if it exists):** |
| 1. If user enters invalid zip code, site redisplays page stating that the user has entered an invalid zip code |

Use Case <uc\_000\_004>

|  |
| --- |
| **Description:** |
| View customer profile |
| **Actors:** |
| Admins |
| **Priority:** |
| High-Medium |
| **Pre-condition:** |
| * Customer has signed up and has an active profile |
| **Flow of Events:** |
| 1. Admin or employee enter in customer credentials in a search 2. Admin or employee is shown a list of potential matches 3. Admin or employee clicks on a customer 4. Admin or employee is brought to that customer’s profile and shown general information regarding customer (name, address, phone, etc.) |
| **Alternative Flow (if it exists):** |
| 1. If no customer exists with the entered in credentials, the search returns ‘no results found’ |

Use Case <uc\_000\_005>

|  |
| --- |
| **Description:** |
| Upload vehicles to website |
| **Actors:** |
| Admins |
| **Priority:** |
| High |
| **Pre-condition:** |
| * Admin or employee is actively working for company * Admin or employee is logged into their account on the website |
| **Flow of Events:** |
| 1. Admin or employee chooses ‘Upload Vehicle’ option in ‘Tools’ tab 2. Admin or employee are prompted to fill in vehicle credentials    1. Make    2. Model    3. Engine Type    4. Zip Code    5. Comments 3. Admin or employee click the upload pictures button to add pictures of vehicle 4. Admin/employee choose desired pictures to upload to the car profile 5. Admin/employee click ‘Upload Vehicle’ button 6. Admin/employee is prompted ‘Are you sure you want to upload vehicle?’ 7. Admin/employee clicks ‘Yes’ 8. Admin/employee is redirected to the now existing car profile page |
| **Alternative Flow (if it exists):** |
| 1. If admin/employee clicks ‘No’ in response to ‘Are you sure you want to upload vehicle?’, the page will do nothing, and the dialog box will disappear 2. If admin/employee does not want to upload the vehicle, they can choose the option (next to ‘Upload Vehicle’) to ‘Cancel Profile’ 3. If ‘Cancel Profile’ is selected, admin/employee is prompted ‘Are you sure you want to cancel?’ 4. If admin/employee selects ‘Yes’, the entered information will be discarded and they will be redirected to their dashboard 5. If admin/employee selects ‘No’, the dialog box will disappear and nothing will happen |

Use Case <uc\_000\_006>

|  |
| --- |
| **Description:** |
| Modify credentials |
| **Actors:** |
| Registered Customer |
| **Priority:** |
| Medium |
| **Pre-condition:** |
| * User is on our website * User is on their account settings page |
| **Flow of Events:** |
| * User logs into website * User clicks on account settings button   + User clicks on respective input boxes such as username, password, address, credit card information, & insurance information |
| **Alternative Flow (if it exists):** |
| * User fails to login and clicks the “forgot password?” link   + Users are prompted to input their email address along with their address to ensure they are the person they say they are.   + If the information matches our records, the user is sent a temporary password generated from our service to their email address. Otherwise, we send an alert message saying the records don’t match. * Once the user has signed in, they will follow the normal flow of events to reset their information. |

Use Case <uc\_000\_007>

|  |
| --- |
| **Description:** |
| Purchase Vehicle |
| **Actors:** |
| Registered Customer  Guest Customer |
| **Priority:** |
| High |
| **Pre-condition:** |
| * User is on our website browsing car listings |
| **Flow of Events:** |
| * User selects car from our inventory   + User selects any upgrades they want on their car   + User inputs required information such as address, credit card information, and insurance information in their respective input fields. * User clicks “purchase” with a confirmation screen of their recent purchase.home page |
| **Alternative Flow (if it exists):**   * User selects car from our inventory   + User selects any upgrades they want on their car   + User inputs required information such as address, credit card information and insurance information in their respective input fields * User clicks “purchase with financing” with a confirmation screen of their recent purchase. * User is redirected to a financing page with space to enter in more information   + User will be asked about their credit score   + User will be asked what financing option they prefer * User clicks “accept financing option” * Confirmation screen of their pending purchase is displayed to user |
| N/A |

Use Case <uc\_000\_008>

|  |
| --- |
| **Description:** |
| Search for existing vehicles based on specific criteria |
| **Actors:** |
| Registered Customer  Guest Customer |
| **Priority:** |
| Low |
| **Pre-condition:** |
| * User clicks on search bar and inputs the name of the car they desire * User clicks buttons that represent car color, style (Sedan, SUV, Truck), and brand * User inputs the price range * User inputs the year range |
| **Flow of Events:** |
| * User is logged in   + User inputs the car name into the search bar   + User views list of available cars based on their search query * User views cars based on their input |
| **Alternative Flow (if it exists):** |
| * User is logged in   + User clicks buttons that describe the car they want   + User types into input fields to describe year and price ranges for their car * User views cars based on their input |

Use Case <uc\_000\_010>

|  |
| --- |
| **Description:** |
| Remove vehicles that are unavailable |
| **Actors:** |
| Admins |
| **Priority:** |
| High |
| **Pre-condition:** |
| * Admin is actively working for company |
| **Flow of Events:** |
| * Once admin in their appropriate dashboard * Admin clicks on search bar and inputs the vehicleID * Admin chooses the vehicle needs to be deleted * Admin click the delete vehicle button * Admin is prompted ‘Are you sure you want to delete the vehicle?’ * Admin clicks ‘Yes’ * Vehicle is deleted and Admin is redirected to their respective dashboard |
| **Alternative Flow (if it exists):** |
| * Admin inputs incorrect vehicleID   + Admin is prompted to re-enter the vehicle * Admin clicks no when prompted ‘Are you sure you want to delete the vehicle?’   + Admin is redirected to their respective dashboard |

Use Case <uc\_000\_011>

|  |
| --- |
| **Description:** |
| Accept or deny trade-ins |
| **Actors:** |
| Admins |
| **Priority:** |
| High |
| **Pre-condition:** |
| * Admin is actively working for company * Admin is on our website |
| **Flow of Events:** |
| * Admin is in their appropriate dashboard * Admin select the option View Pending Trade-Ins * Admin browse customers who have requested trade-ins   + CustomerID   + VehicleID   + This page will show the valuation and information on the car being traded * Admin click the Accept order button   + Admin is prompted ‘Are you sure you want to Accept this trade-in?’   + Admin clicks ‘Yes’   + Trade-in is approved and Admin is redirected to their respective dashboard * Admin clicks the Deny order Button   + Admin is prompted ‘Are you sure you want to Deny this trade-in?’   + Admin clicks ‘Yes’   + Trade-in is cancelled and Admin is redirected to their respective dashboard |
| **Alternative Flow (if it exists):** |
| * Admin inputs incorrect CustomerID or VehicleID   + Admin is prompted to re-enter the information * Admin is prompted ‘Are you sure you want to Accept this trade-in?’ * Admin clicks ‘No’   + Admin is redirected to their respective dashboard |

Use Case <uc\_000\_012>

|  |
| --- |
| **Description:** |
| Provide financing options for customers |
| **Actors:** |
| Admins |
| **Priority:** |
| High |
| **Pre-condition:** |
| * Admin is actively working for company |
| **Flow of Events:** |
| * Admin is in their appropriate dashboard * Admin chooses Add Finance Company option in ‘Tools’ tab   + Admin or employee are prompted to fill out the necessary information     - CompanyID     - Company Name     - LoanID     - Loan Amount     - Payment Forms   + Admin clicks Submit Policy Button * Admin/employee is redirected to the now existing car profile page |
| **Alternative Flow (if it exists):** |
| * Admin inputs incorrect information   + Admin is prompted to re-enter the information |

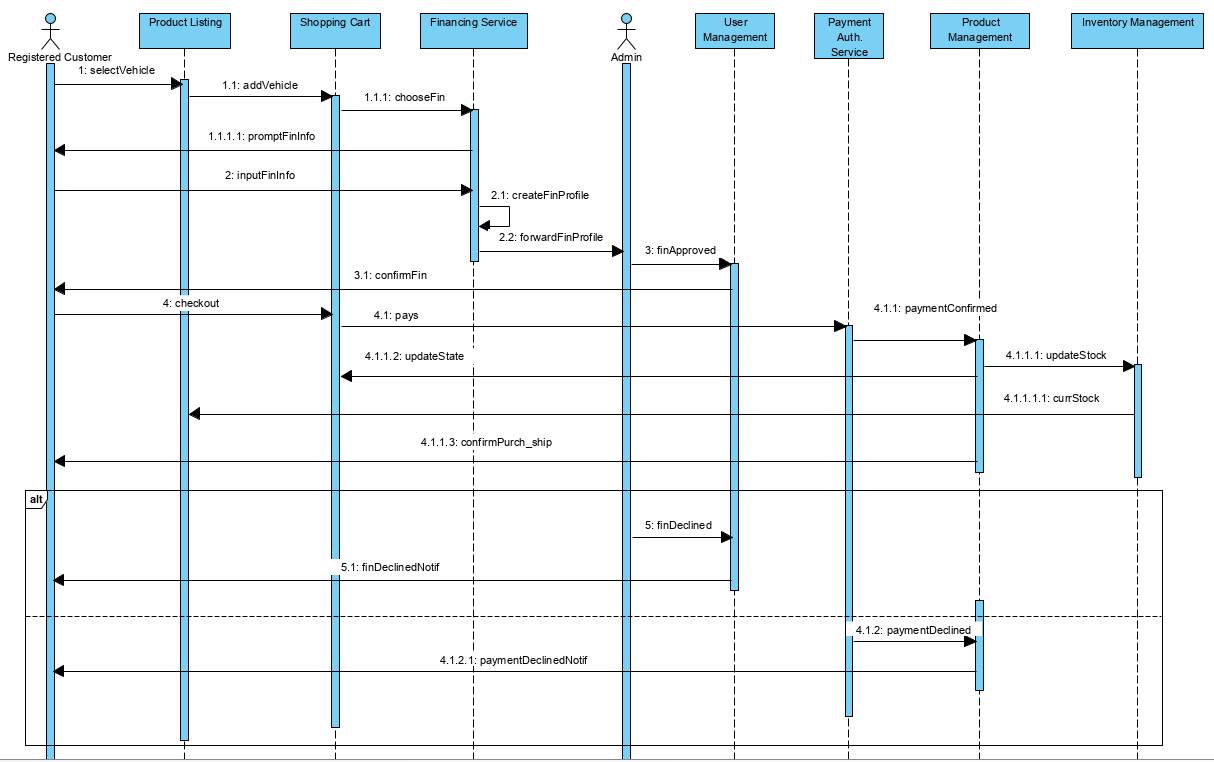
Use Case <uc\_000\_013>

|  |
| --- |
| **Description:** |
| Request vehicle status and location |
| **Actors:** |
| Admins |
| **Priority:** |
| Low/Medium |
| **Pre-condition:** |
| * Admin or employee is actively working for company * Admin or employee is logged into their account on the website |
| **Flow of Events:** |
| 1. Admin or employee chooses ‘Check Vehicle Status’ option in ‘Tools’ tab 2. Admin or employee are prompted to fill in identification information/number of the vehicle 3. Admin or employee is redirected to tracker page showing:    1. Current shipping status       1. Shipped       2. Delivering       3. Preparing for Delivery       4. In Storehouse    2. Current location       1. GPS exact location   and/or   * + 1. Last handled location   1. General Information about vehicle      1. Identification Number      2. When it was ordered (If applicable)      3. Picture      4. Link to vehicle’s page in general inventory |
| **Alternative Flow (if it exists):** |
| 1. If admin/employee enters invalid identification info/number, they will be informed that there is “no such car” and met with a prompt to try again |

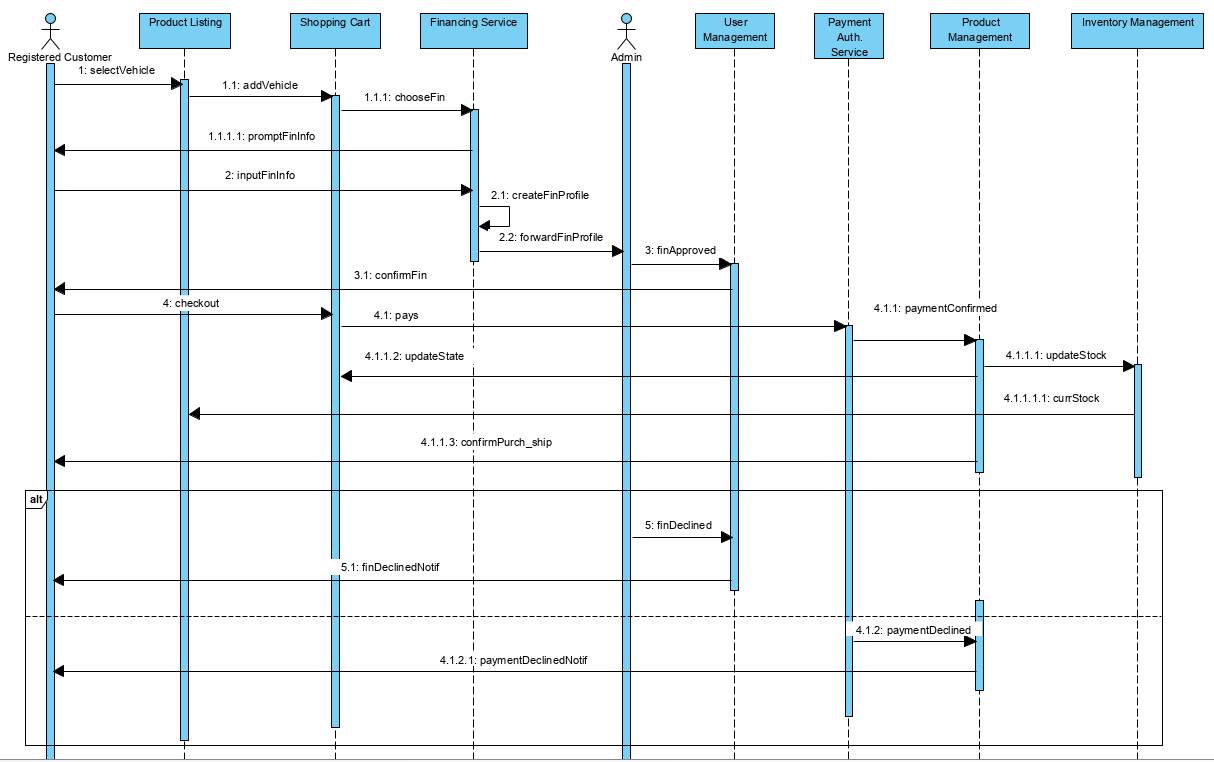
Use Case <uc\_000\_014>

|  |
| --- |
| **Description:** |
| Trade in Vehicle |
| **Actors:** |
| Registered Customers |
| **Priority:** |
| High |
| **Pre-condition:** |
| * Customer must have an account created |
| **Flow of Events:** |
| * User logs into their account * User chooses Trade-in car option from their appropriate dashboard * User chooses Add a vehicle Description option in ‘Tools’ tab   + User is prompted to fill out the necessary information     - Vehicle VIN number     - Vehicle information     - Vehicle Picture     - Vehicle condition     - Vehicle Value   + User click Submit Button * User is redirected to their dashboard |
| **Alternative Flow (if it exists):** |

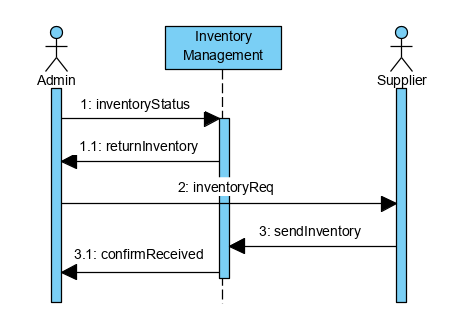
* + 1. **Sequence Diagram**
       1. **Financing a Vehicle**



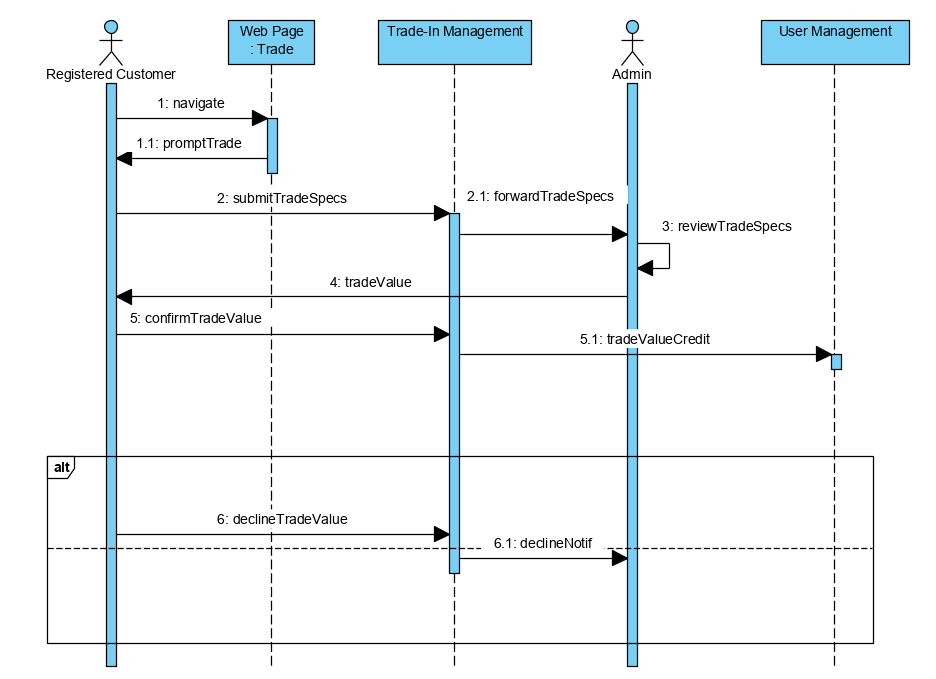
* + - 1. **Buying a Vehicle**



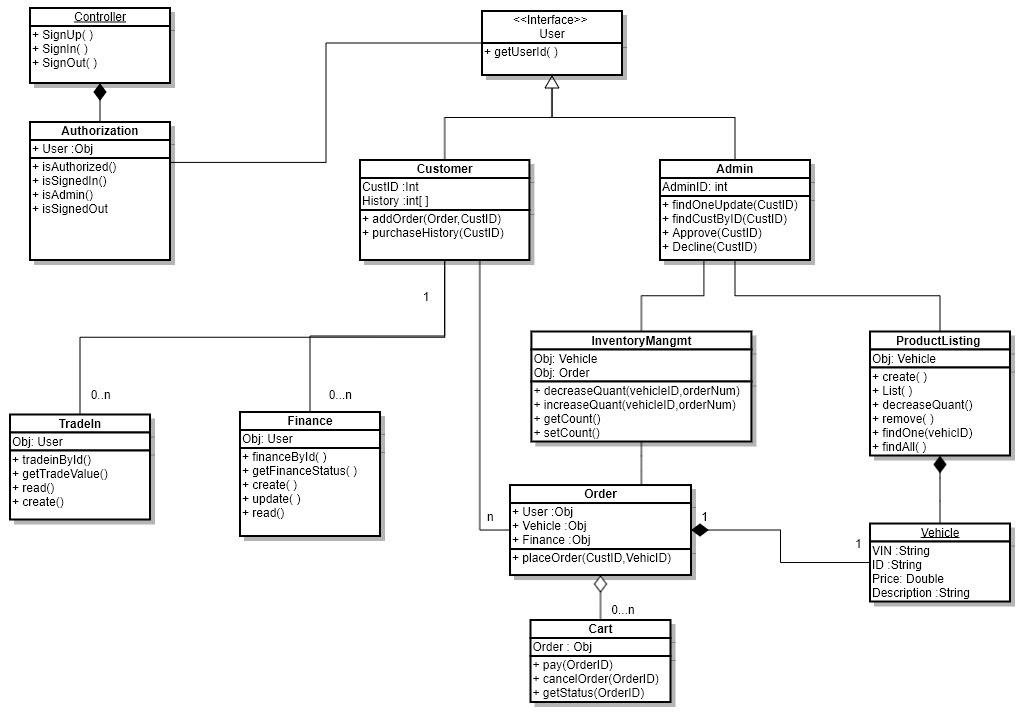
* + - 1. **Restocking Inventory**



* + - 1. **Trading Vehicle**



* + 1. **Class Diagram**



**Identification:**

Controller

**Type:**

Class

**Purpose:**

The Controller is responsible for supporting the logic and validation of users signing up,in or out of Morissette web application.

**Attributes (fields):**

User Object

**Attribute Description**:

The user objects contain all the attributes of the user entity such as ID,Name,BuyHistory ,Email,Password, Role(Admin,Customer,Trader), and  password.

**Processing**

**Methods**

  SignUp()

**Method Description**

This method is responsible for handling the process for customers and admins account creation. This method receives all necessary information from the create account GUI and stores the information into the user table in the database.

**Methods**

  SignIn()

**Method Description**

This method is responsible for allowing customers and admins to login into their respective account. This method receives email addresses and passwords from SignIn GUI and verifies if the user exists in the database.

**Methods**

  SignOut()

**Method Description**

This method is responsible for allowing customers and admins to logout of their respective account.This method uses SignOut Button found in the Account  sidebar, which will give the user the option to logout of their account, then they will be logged out and taken to the homepage.

**Identification:**

Authorization

**Type:**

Class

**Purpose:**

The Authorization class is responsible for validation of user’s privileges throughout the system.

**Attributes (fields):**

User Object

**Attribute Description**:

ID, Name, BuyHistory, Email, Password, Role (Admin, Customer, Trader), and password.

**Processing**

**Methods**

  IsAuthorized()

**Method Description**

This method uses the role attribute of a specific user to ensure they can only access their own account information and prevent them from accessing web pages which they do not have access to.

**Methods**

  IsAdmin()

**Method Description**

This method is responsible for verifying the user in session has admin privilege by checking the role attribute of a specific user.

**Methods**

  IsSignedIn()

**Method Description**

This method is responsible for verifying the user in session is already signed in.

**Methods**

  IsSignedOut()

**Method Description**

This method is responsible for verifying the user in session is already signed out.

**Identification:**

User

**Type:**

Interface

**Purpose:**

The User interface is responsible for specifying the user object behaviors.

**Attributes (fields):**

**Attribute Description**:

**Processing**

**Methods**

  getUserID()

**Method Description**

This method is responsible for retrieving the user id from the database used for identification purposes of a user.

**Identification:**

Customer

**Type:**

Class

**Purpose:**

The Customer class is responsible for supporting the functionalities and features of the customers in the Morissette web Application.

**Attributes (fields):**

CustomerID, History, Order

**Attribute Description**:

CustomerID is used to store the unique identification number of each customer.

History is an array used to store all the current and previous Orders placed by the user.

Order is an object which stores all information for a specific object.

**Processing**

**Methods**

  addOrder()

**Method Description**

This method is responsible for adding a specific order using customers id and order id.

**Methods**

  PurchaseHistory()

**Method Description**

This method is responsible for adding the current and previous Orders placed by the user into the History array.

**Identification:**

Admin

**Type:**

Class

**Purpose:**

The Admin class is responsible for supporting all the activities and operations of the system administrator.

**Attributes (fields):**

AdminID

**Attribute Description**:

The AdminID is used to store the unique identification number of each Administrator.

**Processing**

**Methods**

  FindOneUpdate(CustID )

**Method Description**

This method receives Customers id as a parameter and updates any information regarding that customer’s in the system such as Payment-Information,Order, and Account-Information.

**Methods**

  FindCustByID(CustID)

**Method Description**

This method receives Customers id as a parameter and returns all information regarding that customer throughout the system.

**Methods**

  Approve(CustID)

**Method Description**

This method receives Customers id as a parameter and is used for approving customers vehicle trade-in requests or financing applications for a vehicle purchase.

**Methods**

  Decline(CustID)

**Method Description**

This method receives Customers id as a parameter and is used for declining customers vehicle trade-in requests or financing applications for a vehicle purchase along with an explanation for denial.

**Identification:**

Inventory Management

**Type:**

Class

**Purpose:**

This class is responsible for managing all the administrative functions of the admin regarding product inventory.

**Attributes (fields):**

Vehicle Object , Order Object

**Attribute Description**:

The Vehicle object is used by the inventory Management class for accessing any information regarding a vehicle.

**Attribute Description**:

The Order object is used by the inventory Management class for accessing all the information of customers' order.

**Processing**

**Methods**

  decreaseQuant(VehicleID,orderNum)

**Method Description**

This method receives VehicleIDid and customers order number as a parameter and decreases the total quantity of vehicles in the inventory.

**Methods**

  IncreaseQuant(VehicleID,orderNum)

**Method Description**

This method receives VehicleIDid and customers order number as a parameter and increases the total quantity of vehicles in the inventory whenever a customer is not able to proceed with the order or their purchase was denied by the admin.

**Methods**

  getCount( )

**Method Description**

This method returns the total vehicle quantity in the inventory.

**Methods**

  setCount( )

**Method Description**

This method updates the total vehicle quantity when a new vehicle is added in the inventory.

**Identification:**

Product Listing

**Type:**

Class

**Purpose:**

This class is responsible for managing all activities regarding the listing of products.

**Attributes (fields):**

Vehicle Object

**Attribute Description**:

The Vehicle object is used by the Product Listing class for accessing any information regarding a vehicle.

**Processing**

**Methods**

  create( )

**Method Description**

This method is a new product listing.

**Methods**

  List( )

**Method Description**

This method lists the new product that was created.

**Methods**

  remove( )

**Method Description**

This method is used for removing a product that was listed..

**Methods**

  findOne(vehicelD )

**Method Description**

This method finds one vehicle by its id and returns its information.

**Methods**

  findAll( )

**Method Description**

This method finds and returns all vehicles and all of their information.

**Identification:**

Order

**Type:**

Class

**Purpose:**

This class is responsible for managing all activities regarding a customer's order.

**Attributes (fields):**

Vehicle Object

**Attribute Description**:

The Vehicle object is used by the order class for accessing any information regarding a vehicle.

**Attributes (fields):**

User Object

**Attribute Description**:

The user object is used by the order class for accessing any information regarding a user.

**Attributes (fields):**

Finance Object

**Attribute Description**:

The finance object is used by the order class for accessing any information regarding a customer's financial history.

**Processing**

**Methods**

  PlaceOrder( )

**Method Description**

This method receives customers id and vehicle id as a parameter and places a new order, if a customer has been approved for financing the method will be able to access the payment information through the finance object.

**Identification:**

Cart

**Type:**

Class

**Purpose:**

This class is responsible for managing all activities regarding a customer's order.

**Attributes (fields):**

Order Object

**Attribute Description**:

The Order object is used by the Cart class for accessing any information regarding a customer's order.

**Processing**

**Methods**

  Pay( )

**Method Description**

This method receives customers' order id as a parameter and provides all the necessary functions for making a payment .

**Methods**

  cancelOrder( )

**Method Description**

This method receives customers' order id as a parameter and provides all the necessary functions for canceling an order.

**Methods**

  getStatusr( )

**Method Description**

This method receives customers' order id as a parameter and provides all the necessary information such as shipping and handling, payment approval status.

**Identification:**

Finance

**Type:**

Class

**Purpose:**

This class is responsible for managing all activities regarding a customer's financial application.

**Attributes (fields):**

User Object

**Attribute Description**:

The user object is used by the order class for accessing any information regarding a user.

**Processing**

**Methods**

  FinaceByID( )

**Method Description**

This method will allow customers to submit a financial application using their customer id .

**Methods**

  getFinaceStatus( )

**Method Description**

This method will return the status of  the customer's financial application,using this method the customer can find out whether they got approved or denied for a loan.

**Methods**

  Create( )

**Method Description**

This method will allow customers' to create a new financing application.

**Methods**

  update( )

**Method Description**

This method will allow customers' to to update their financing application with new information. a new financing application.

**Identification:**

Trade in

**Type:**

Class

**Purpose:**

This class is responsible for managing all activities regarding a customer's vehicle trade in application.

**Attributes (fields):**

User Object

**Attribute Description**:

The user object is used by the trade in class for accessing any information regarding a user.

**Processing**

**Methods**

  FinaceByID( )

**Method Description**

This method will verify if the customer existes in the database.

**Methods**

  getTradeinValue( )

**Method Description**

This method will return the status of the customer's vehicle trade in application, using this method the customer can find out the valued price for their vehicle.

**Methods**

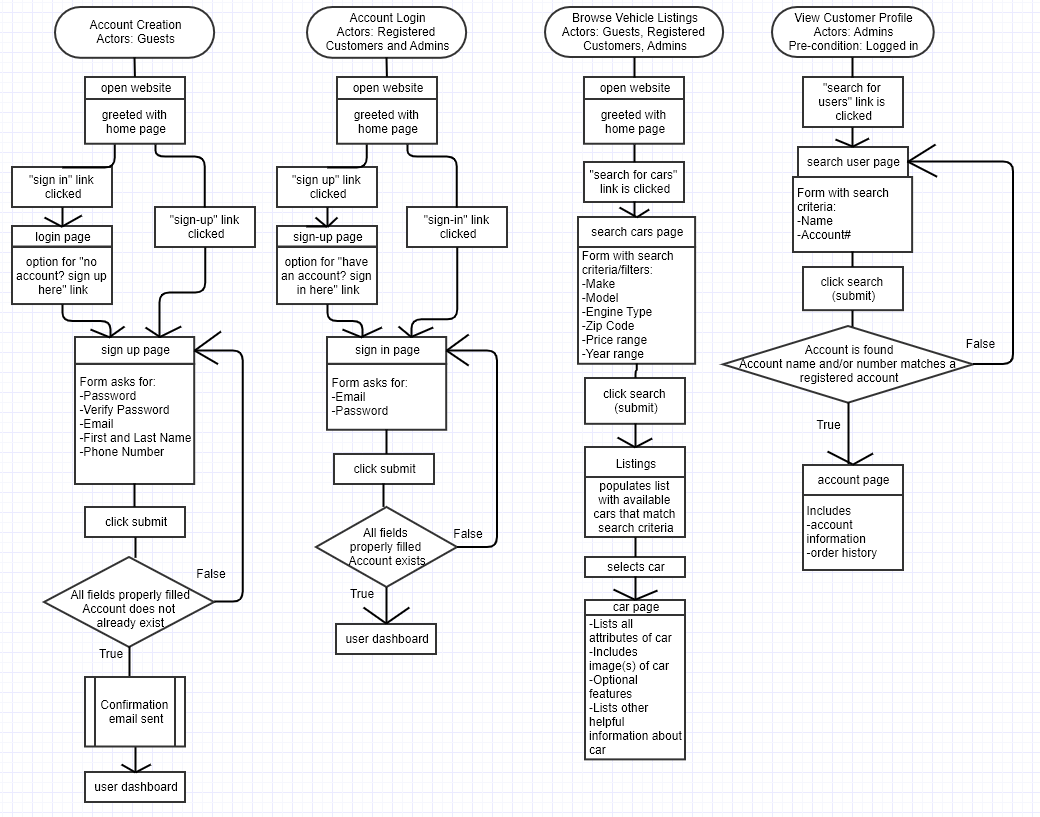
  Create()

**Method Description**

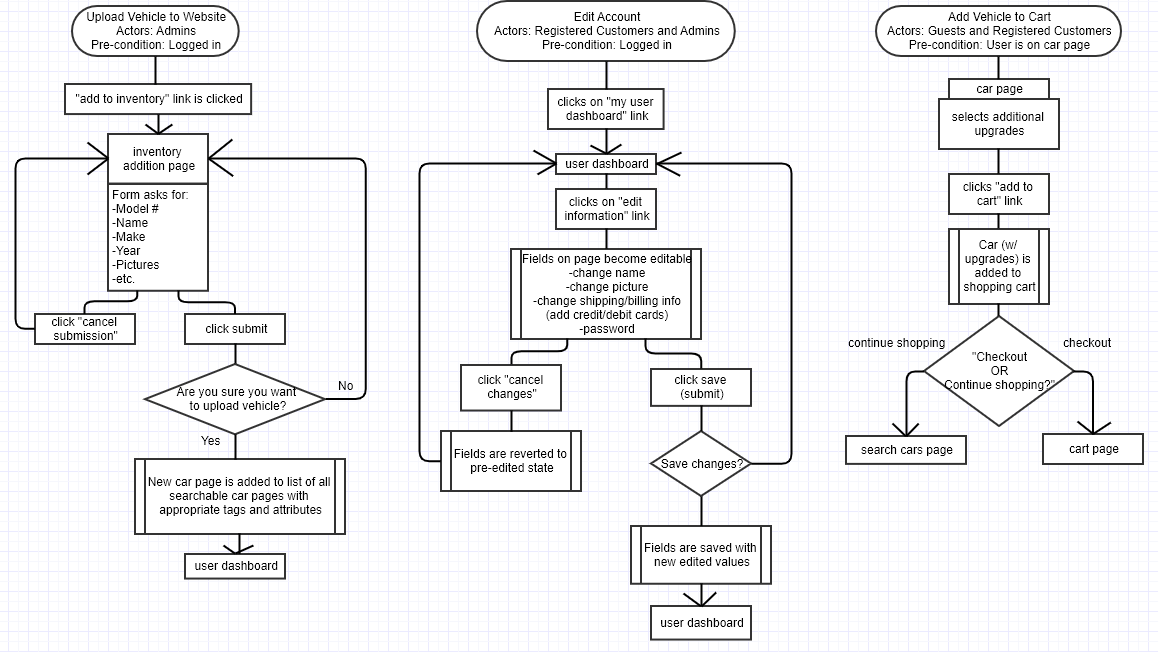
This method will allow customers to create a new trade in application.

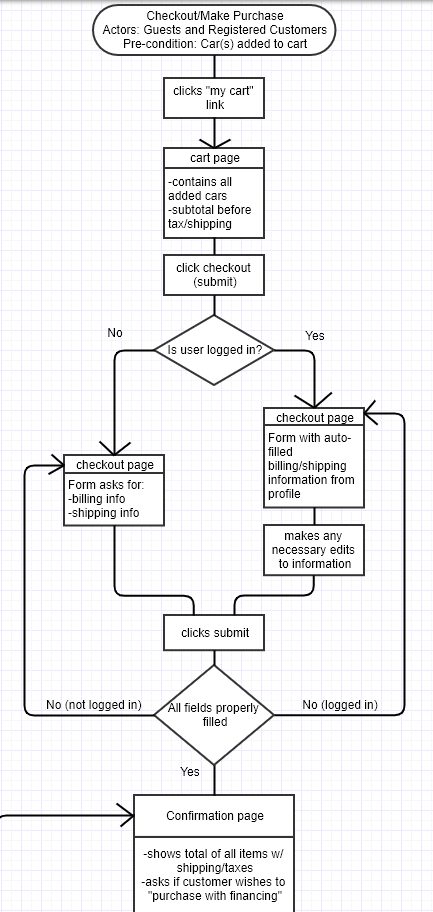
* + 1. **Activity Diagram**

AD\_001 AD\_002 AD\_004 AD\_010

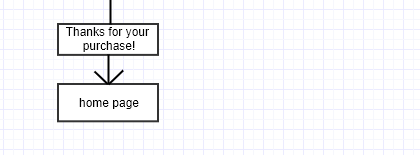
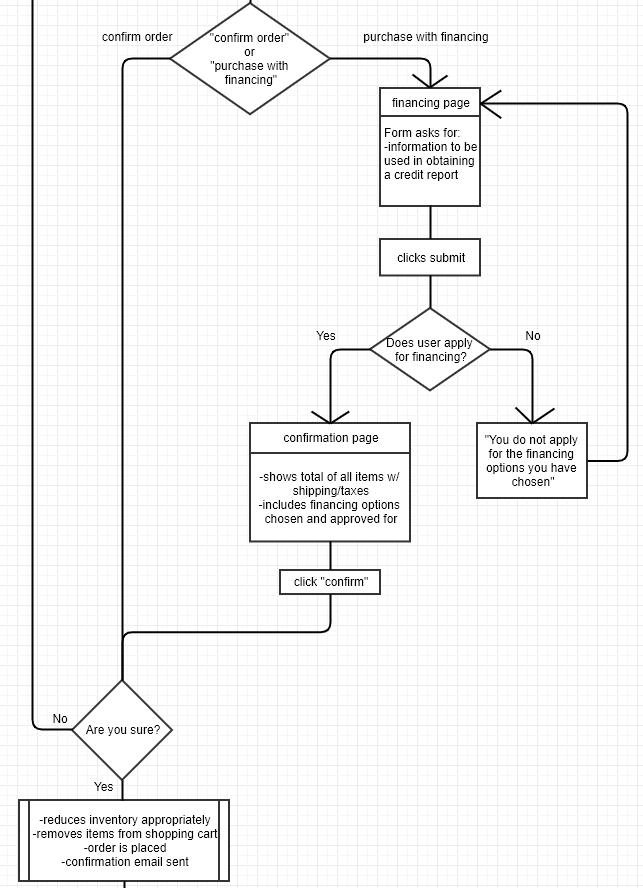


AD\_012 AD\_003 AD\_005

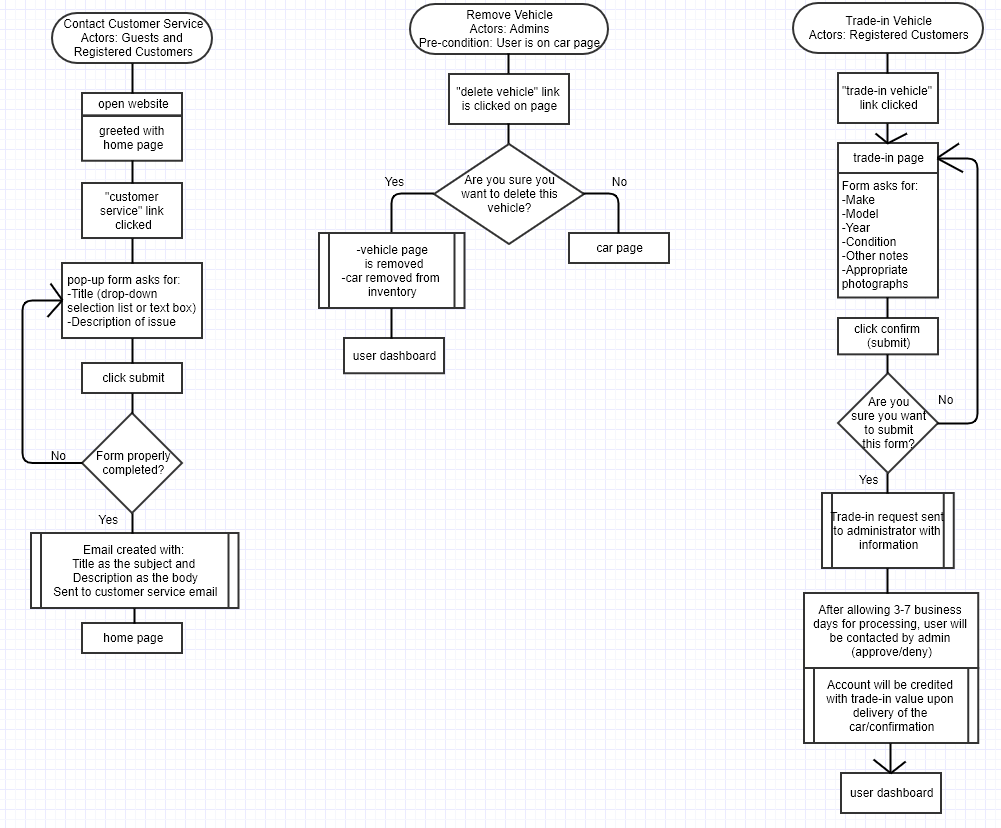


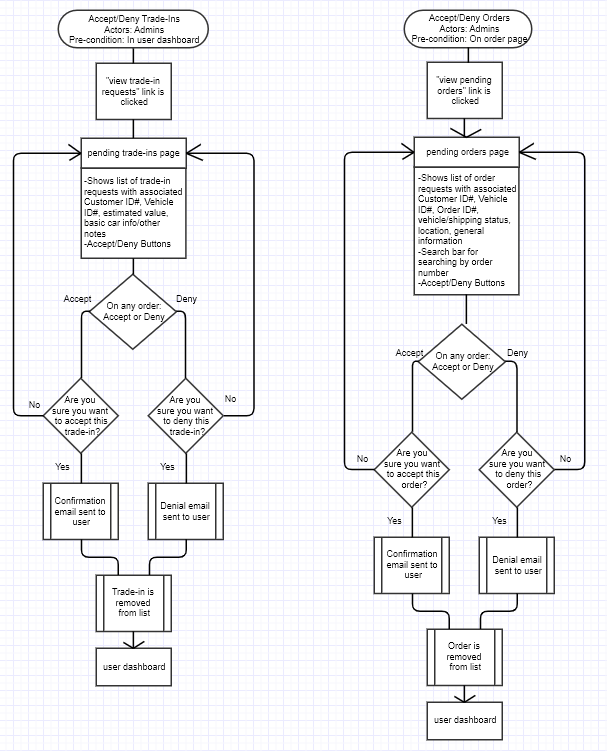


AD\_006



AD\_008 AD\_009 AD\_007



AD\_013 AD\_011

1. **Dependency description**
   1. **Intermodule dependencies**

**Vehicles Module:** keeps tracks of all vehicles in stock and adjusts their quantity accordingly.

**CustomerProfile Module:** collects and stores register customer information after initial sign-up.

**Trade-In Module:** stores the vehicle’s information a registered customer has entered.

**Finance Module:** stores registered customers financial information.

**RequestInventory Module:** communicates with suppliers to restock inventory

**ProcessPayment Module:** processes customer’s payment

**Login Module:** processes customer’s request to log in to view their dashboard

**Filter Module:** filters vehicle listing to customer’s preferences

**Shipping Module:** sends vehicle to customer and tracks vehicle status

**UpdateShoppingCart Module:** updates customer’s shopping cart accordingly

**UpdateInventory Module:** updates inventory when necessary

**FinanceApplicationProcessed Module:** sends customer finance approval or denial

* 1. **Interprocess dependencies**

**Product Listing:** pulls vehicles (and their stock) from Inventory Management and behind-the-scenes database to display available products. Product Listing is updated secondarily from Product Management after Product Management updates Inventory Management.

**Shopping Cart:** holds vehicles customer adds to cart from the Product Listing pages. The Shopping Cart syncs added vehicles and quantities from Product Listing. The Shopping Cart Process acts as a gateway for the customer to access the Payment Authorization Service to purchase a vehicle. The Shopping Cart Process updates the Shopping Cart webpage to “no items in cart (or something similar)” when the Product Management Process receives confirmation from the Payment Authorization Process.

**Payment Authorization Service:** processes customer payment information. The customer is directed to this process from the Shopping Cart. If the Payment Authorization Service Process accepts customer payment, this triggers the Product Management Process to update the Inventory Management Process, Shopping Cart, and customer confirmation notification. If the Payment Authorization Service Process declines customer payment, this triggers a decline notification that the customer will receive from the Product Management Process.

**Product Management:** controls how the system interacts together and notifies customers regarding product/payment. When a purchase is made, the Product Management Process tells Inventory Management to update its stock of that vehicle, updates the state of the Shopping Cart back to a neutral state, and sends out confirmation of purchase to the customer. When inventory is low, the Product Management Process sends a request for specified vehicles to suppliers, and once the vehicles are received, the Inventory Management Process populates the Product Management Process with the updated stock.

**Inventory Management:** works in sync with the Product Management Process. Product Management updates the Inventory Management when necessary. For example, when vehicles are purchased, Product Management sends a request for the vehicle to Inventory Management and updates the stock in Inventory Management.

**Trade-In Management:** processes customer’s request to trade-in their vehicle and verifies all required fields have been filled out and forwards customer’s vehicle information to Admin in charge of reviewing vehicle trade-in.

**User Management:** holds information regarding registered customers such as, name, address, vehicle trade-in information (if applicable).

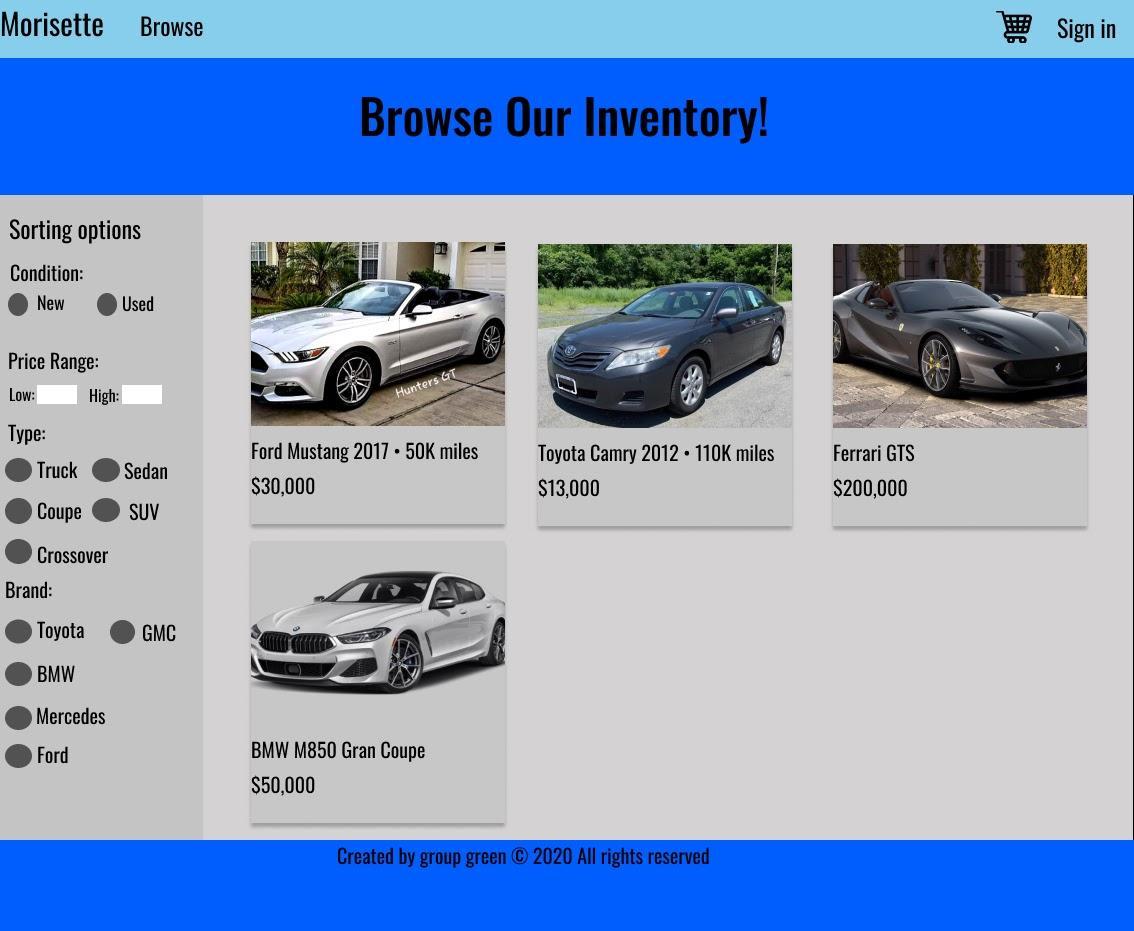
**Financing Service:** processes information input by Registered Customer to create a Financing Profile for the Registered Customer that is then forwarded on to the Admin for approval or denial. This Financing Profile is then attached to the Registered Customer’s account in the User Management Process. The Financing Service Process is triggered when the Registered Customer chooses to finance from the Shopping Cart page.

* 1. **Traceability Table**

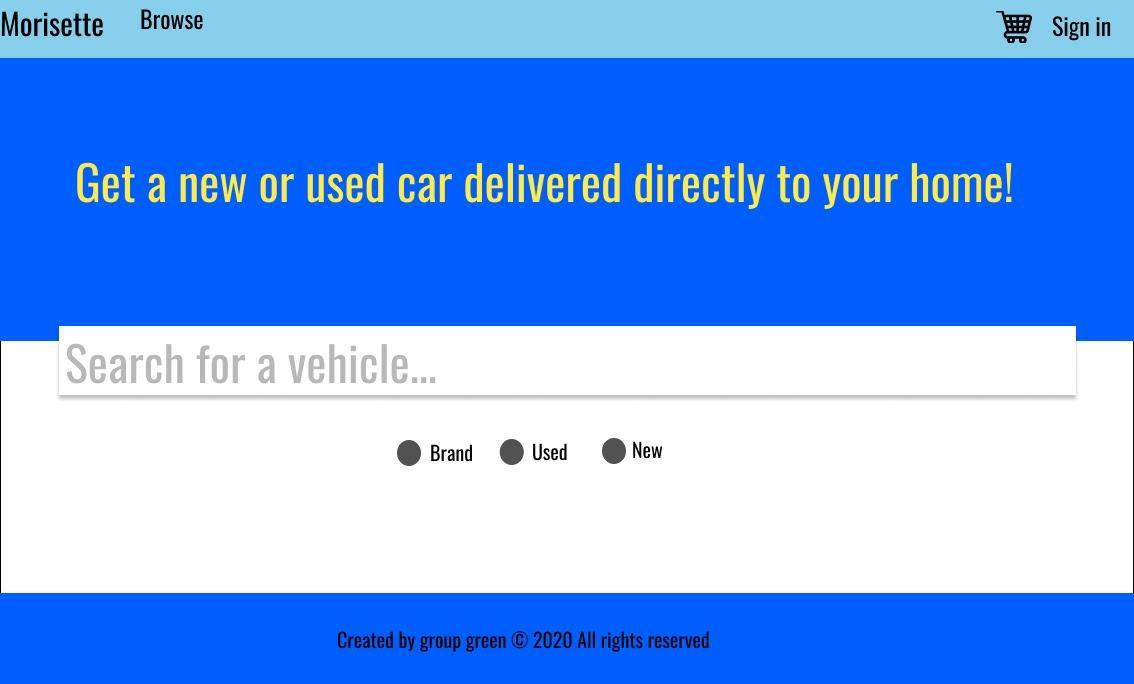
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User story** | **Use cases** | **ActivityDiagram** | **GUI(s)** | **User Manual Section** |
| As a customer I should be able to create an account | Create an account  <uc\_000\_001> | AD\_001 | MS\_008 | UM\_001 |
| As a customer I should be able to log into my account | Login to website  <uc\_000\_002> | AD\_002 | MS\_007 | UM\_002 |
| As a customer I should be able to edit my account details | Modify Credentials  <uc\_000\_006> | AD\_003 | MS\_103 | UM\_003 |
| As a customer I should be able to browse existing vehicles in the company’s inventory | Browse Vehicle Listings  <uc\_000\_003> | AD\_004 | MS\_107 | UM\_004 |
| As a customer I should be able to make a purchase of a vehicle | Purchase a Vehicle  <uc\_000\_007> | AD\_005/AD\_006 | MS\_004 | UM\_006 |
| As a customer I should be able to trade in my vehicle and receive store credit | Trade-in Vehicle  <uc\_000\_014> | AD\_007 | MS\_109 | UM\_007 |
| As a customer I should be able to purchase a vehicle with financing | Purchase a Vehicle  <uc\_000\_007> | AD\_006 | MS\_101 | UM\_005 |
| As a customer I should be able to search for vehicles by keyword | Search for existing vehicles based on specific criteria  <uc\_000\_008> | AD\_004 | MS\_002 |  |
| As an admin I should be able to add or remove vehicles that are unavailable | Remove vehicles that are unavailable  <uc\_000\_010> | AD\_009/AD\_012 | MS\_214/215 |  |
| As an admin I should be able to view customer information (ID, insurance, address) | View customer profile  <uc\_000\_004> | AD\_010 | MS\_205 |  |
| As an admin I should be able to login to my account and be directed to the proper dashboard |  | AD\_002 | MS\_200 |  |
| As an admin I should be able to view a list of all vehicles in the system |  | AD\_004 | MS\_107 |  |
| As an admin I should be able to request vehicle status and location | Request vehicle status and location  <uc\_000\_013> | AD\_011 | MS\_201 |  |
| As an admin I should be able to provide financing options to customers | Provide financing options for customers  <uc\_000\_012> | AD\_006 | N/A |  |
| As an admin I should be able to view customer contact information |  |  | MS\_205 |  |
| As an admin I should be able to set a price for a vehicle |  | AD\_012 | IOS\_101 |  |
| As an admin I should be able to approve or deny trade-ins | Approve or deny trade-ins  <uc\_000\_011> | AD\_013 | MS\_212 |  |

1. **Interface description**
   1. **User Interface**
      1. **Menu screens**

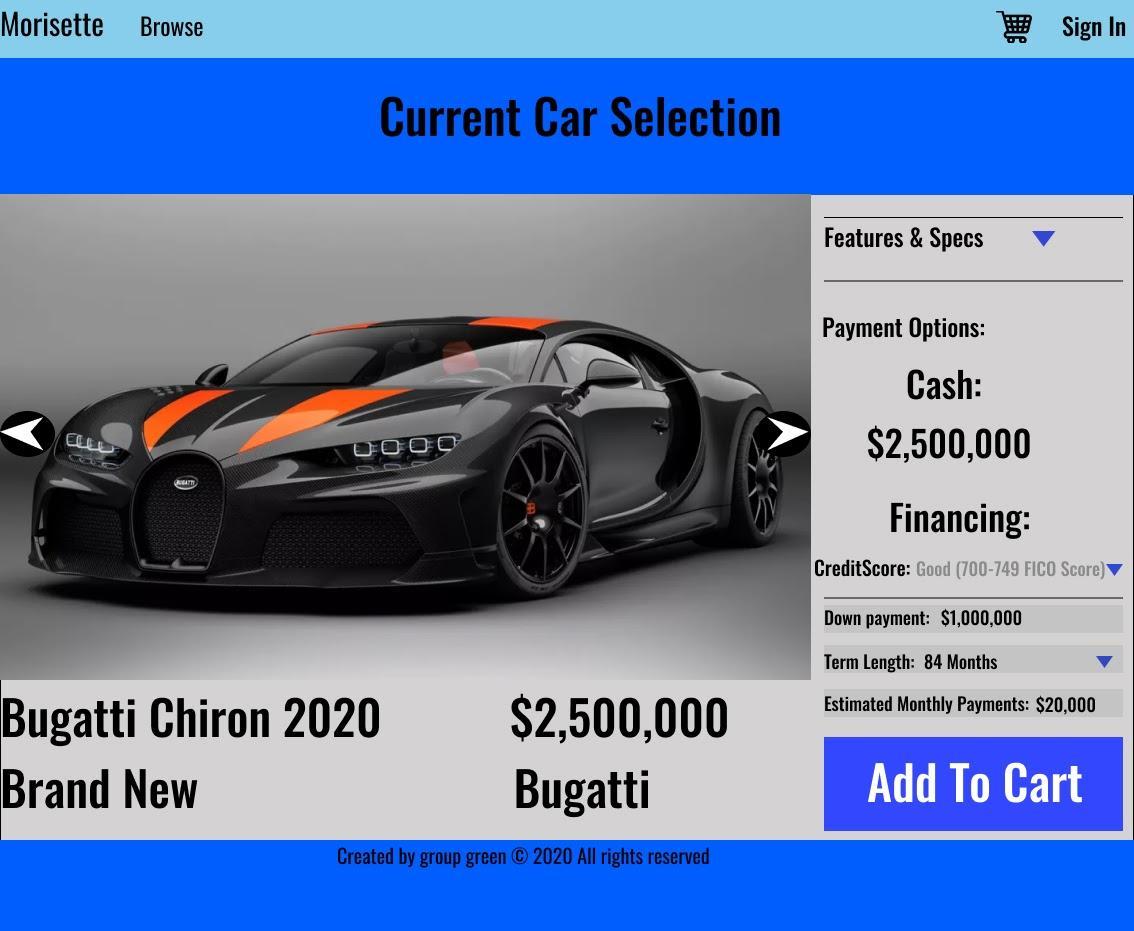
MS\_001



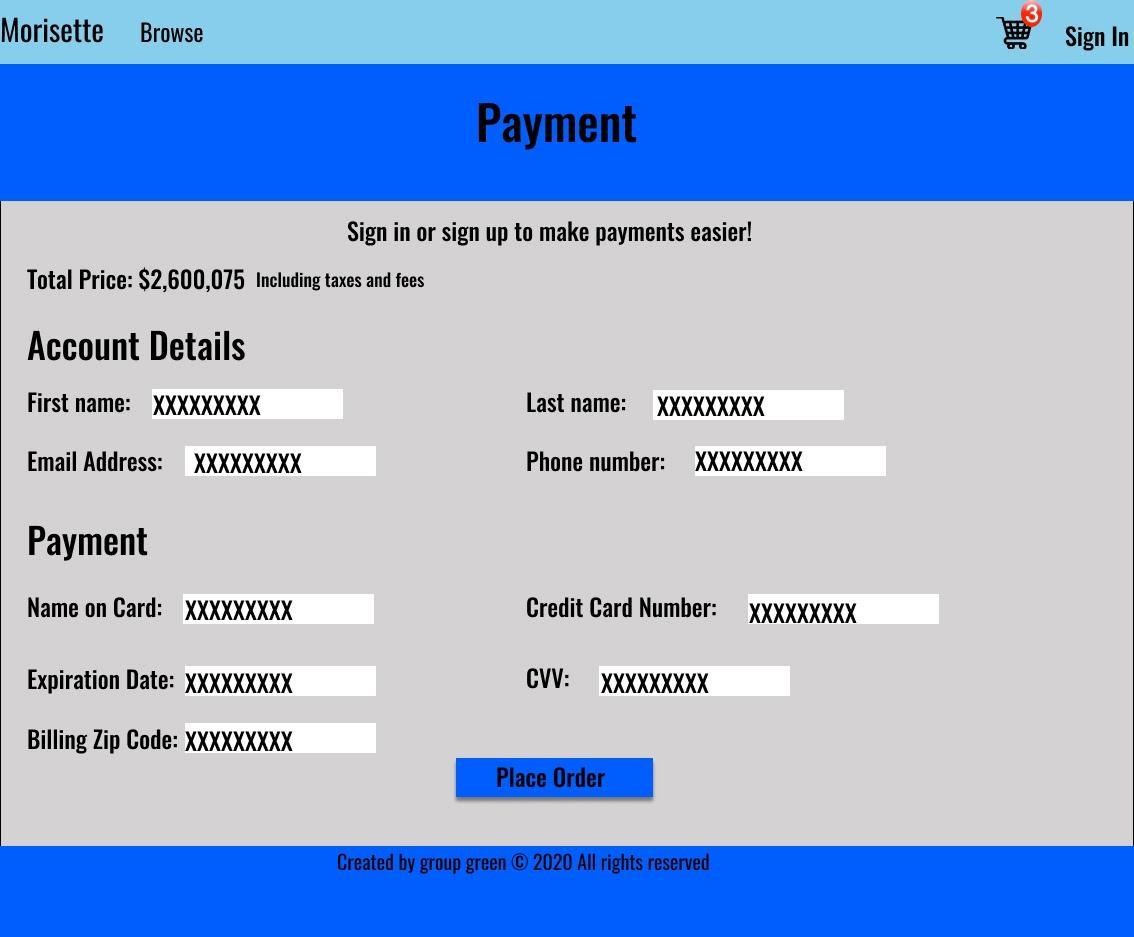
MS\_002



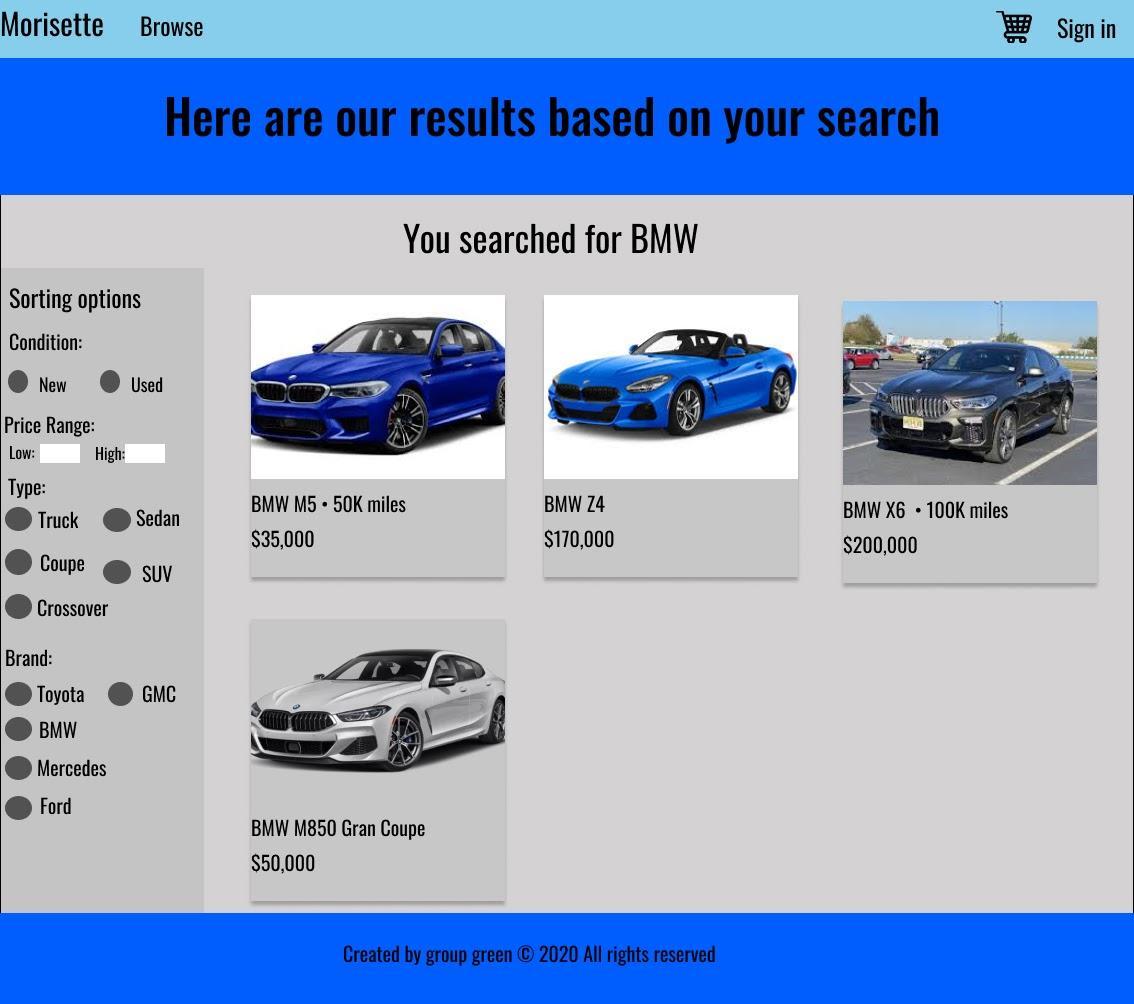
MS\_003



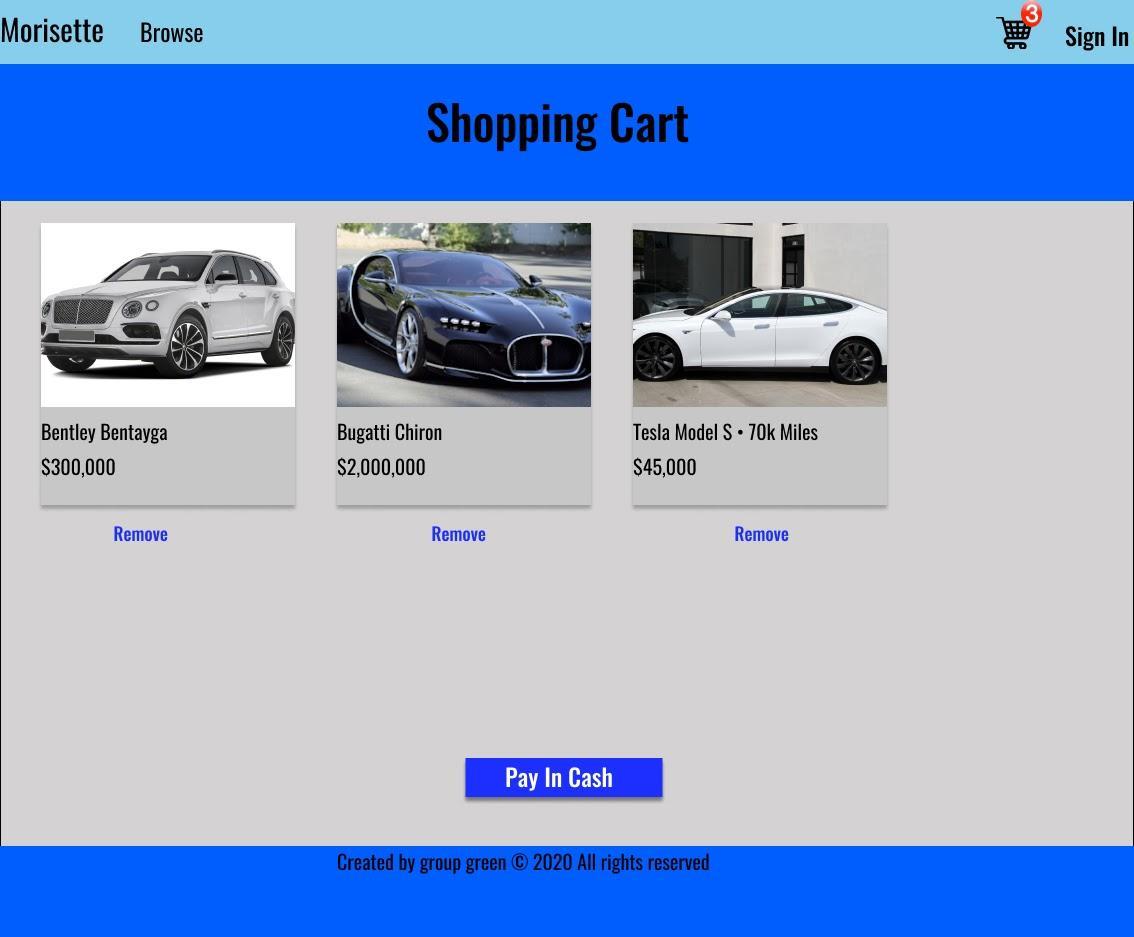
MS\_004



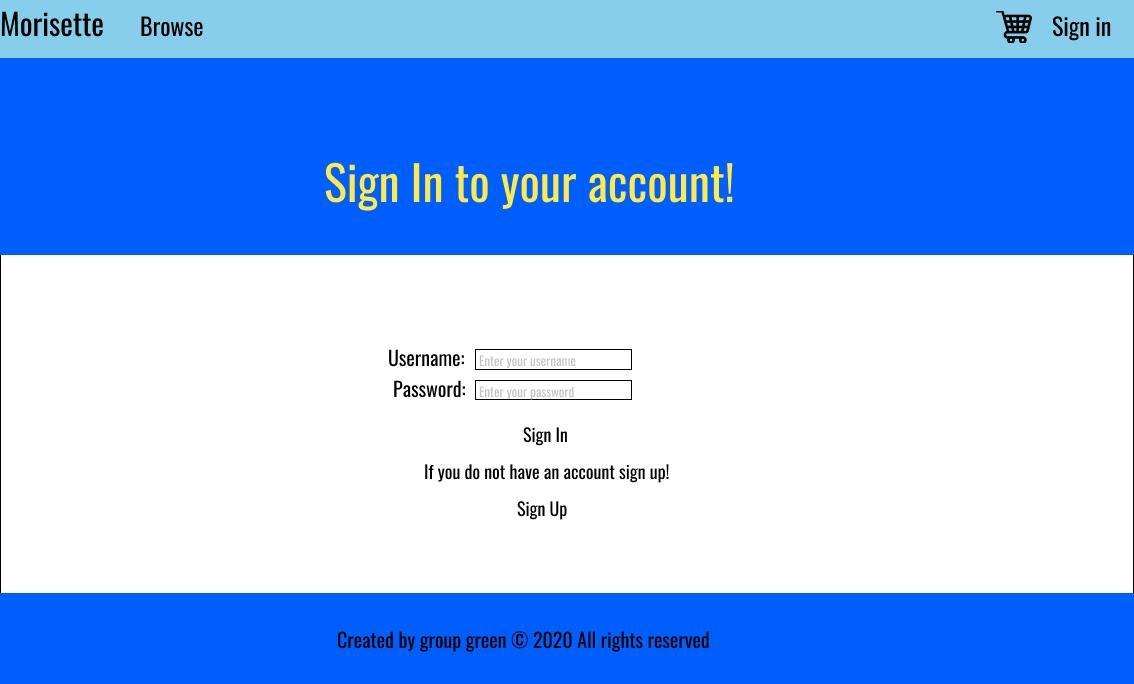
MS\_005



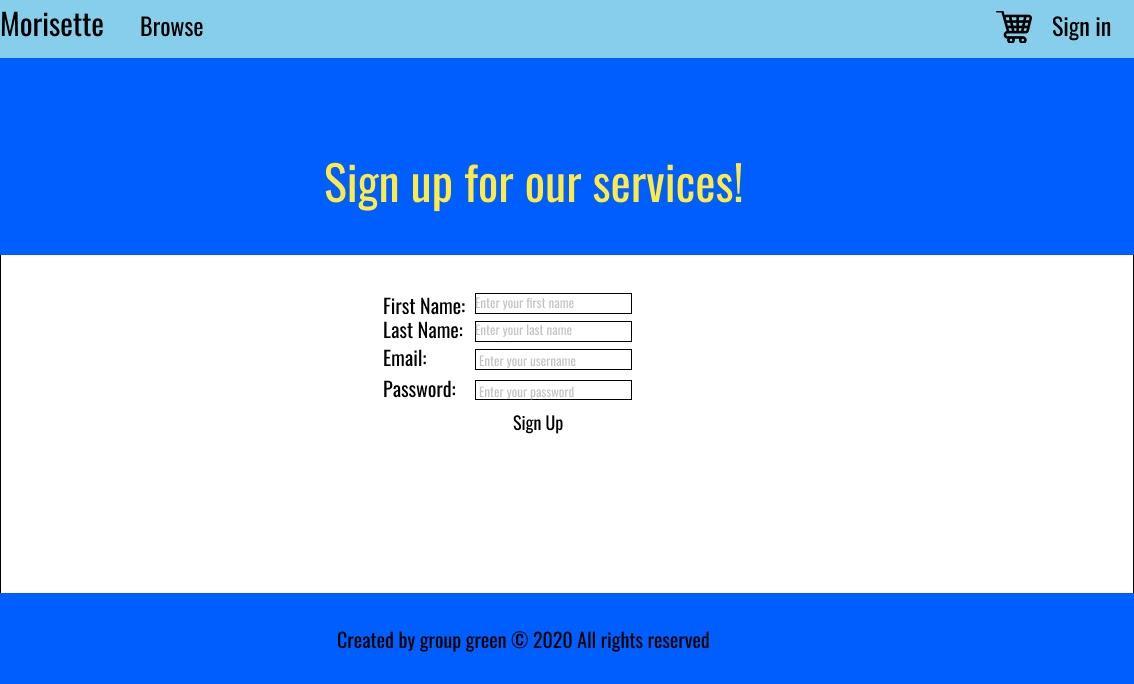
MS\_006



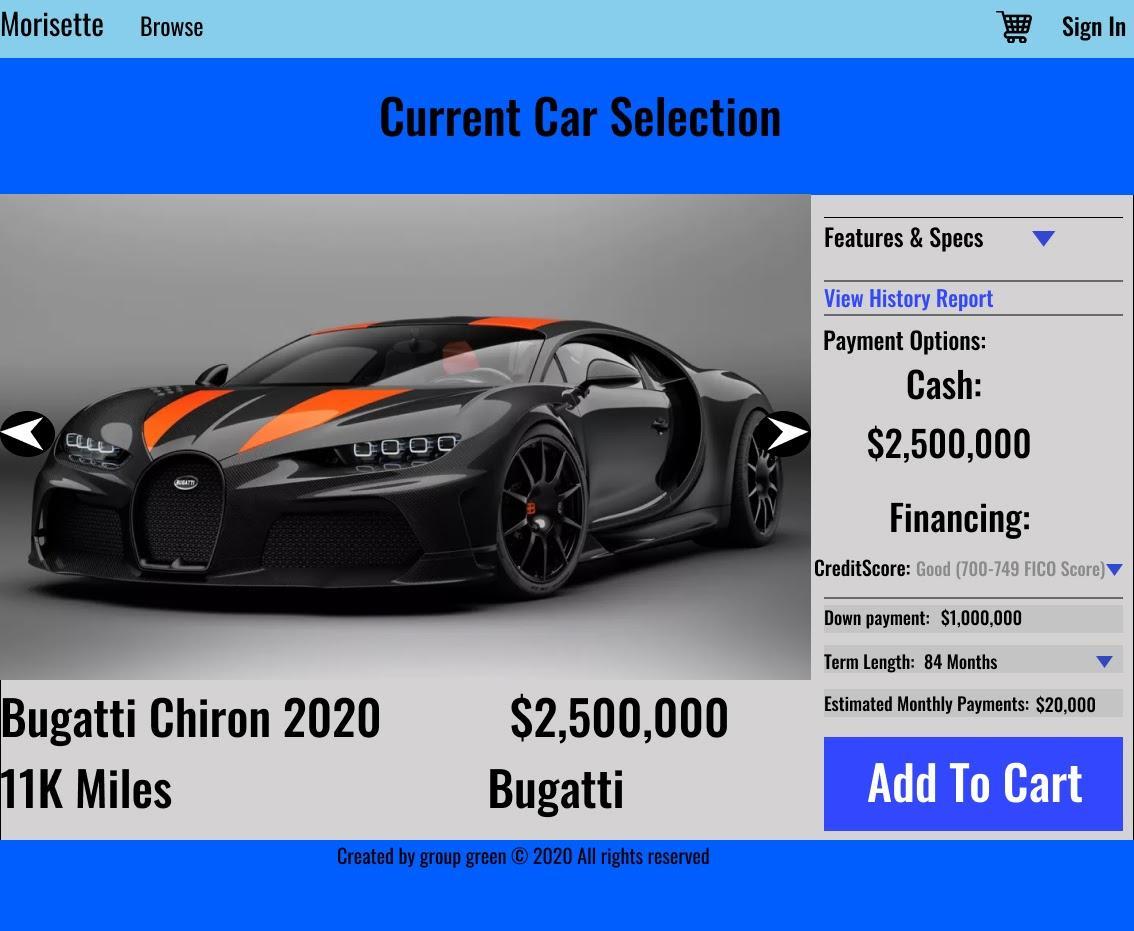
MS\_007



MS\_008

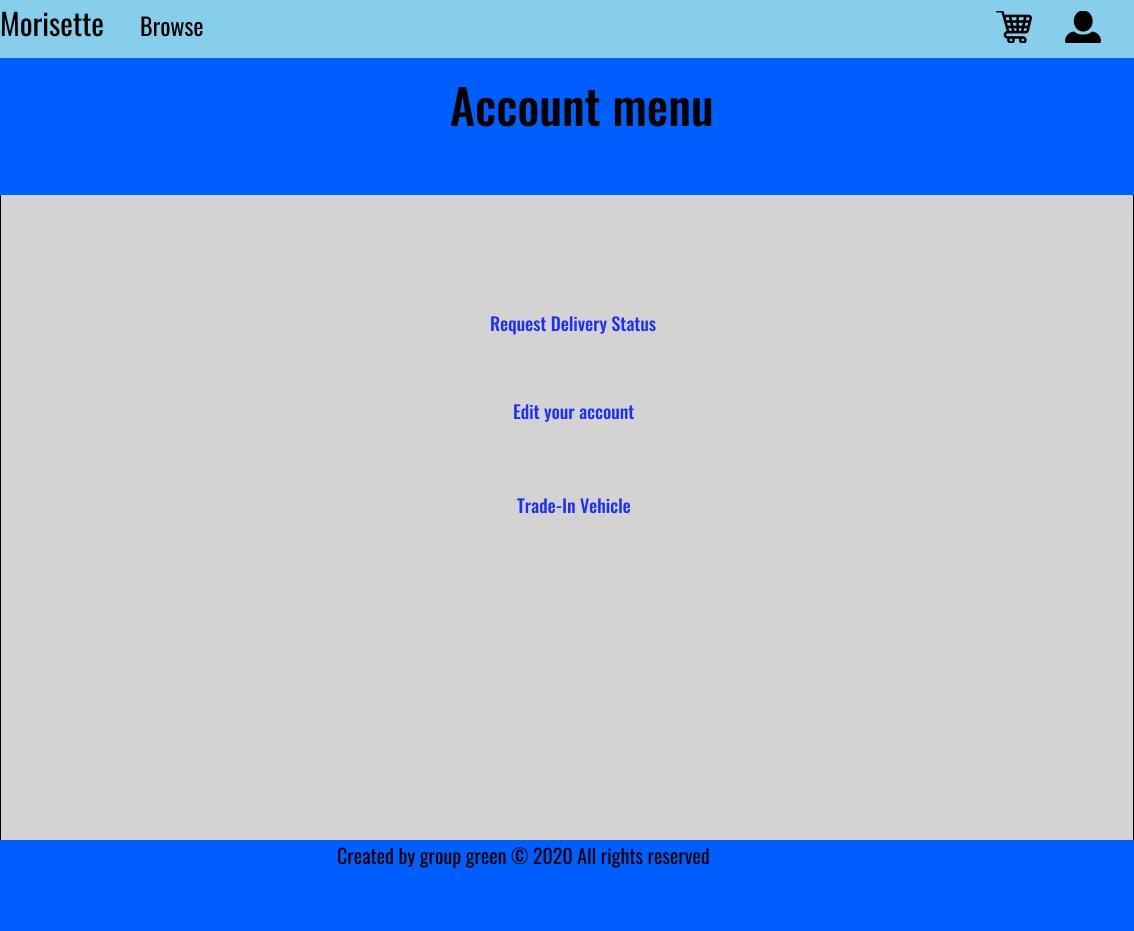


MS\_009

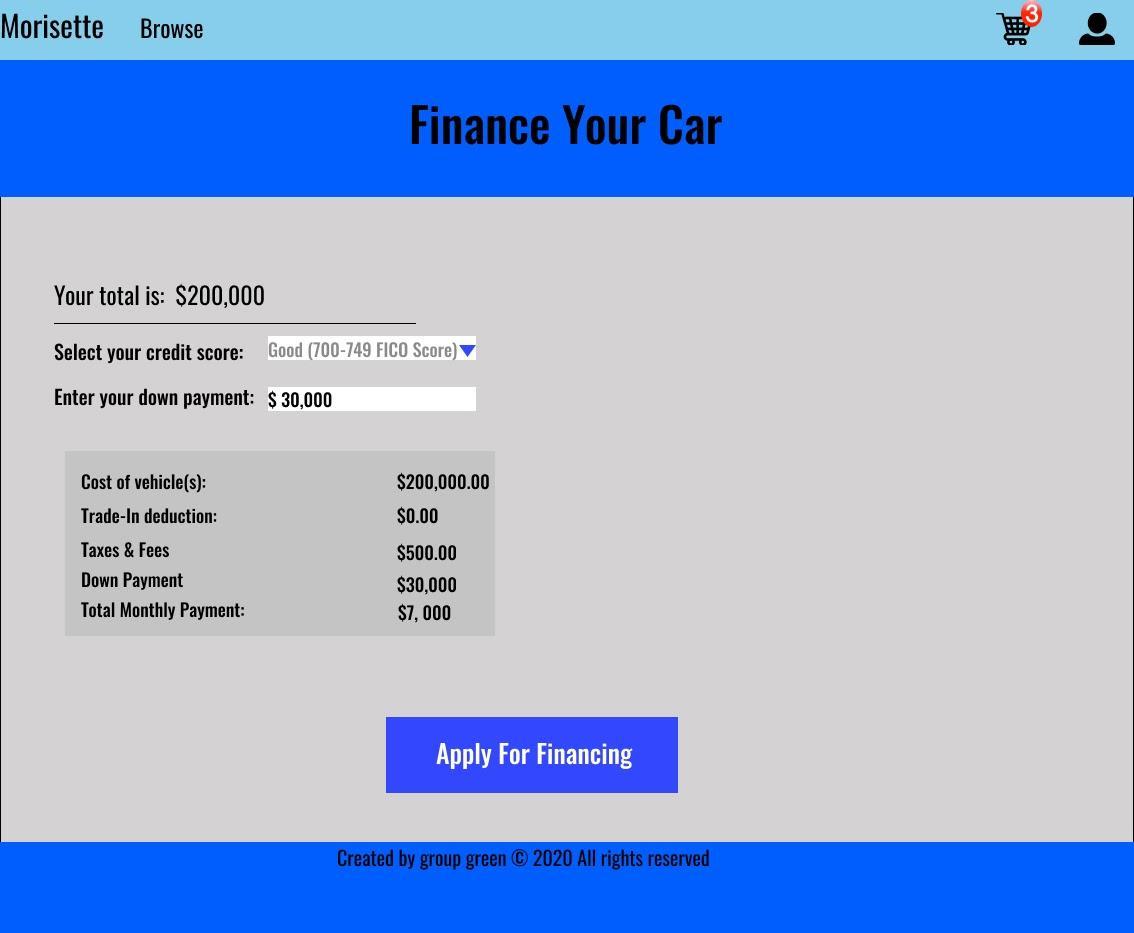


**5.2.1.2  Registered User Menu Screens**

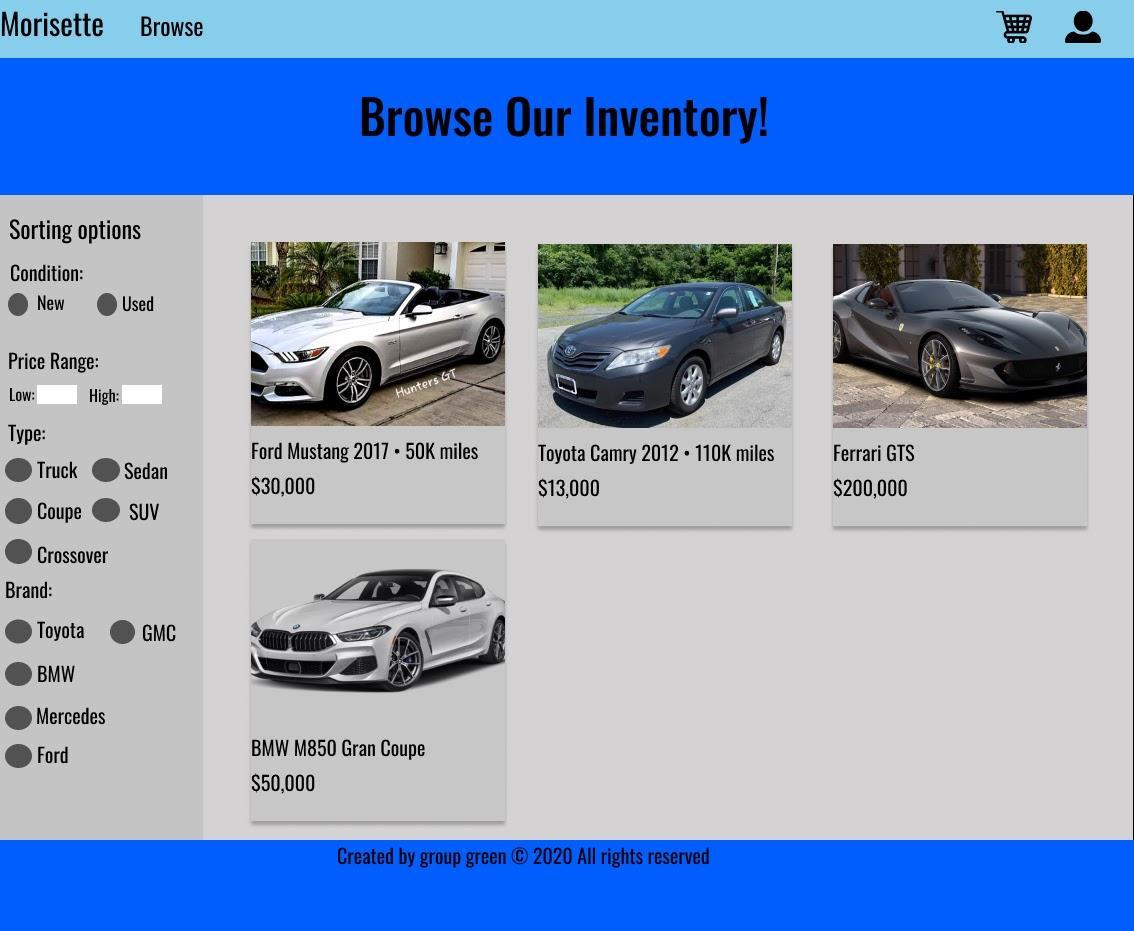
MS\_100



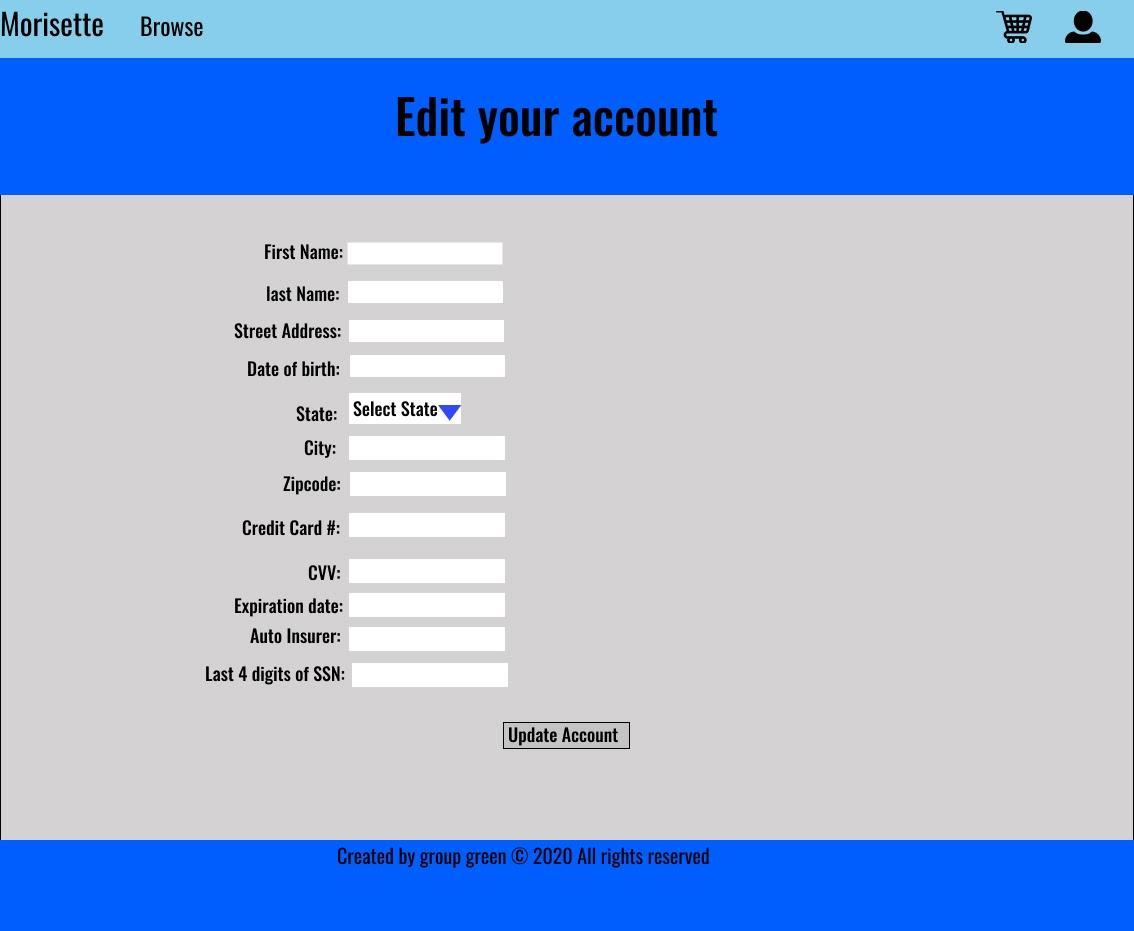
MS\_101



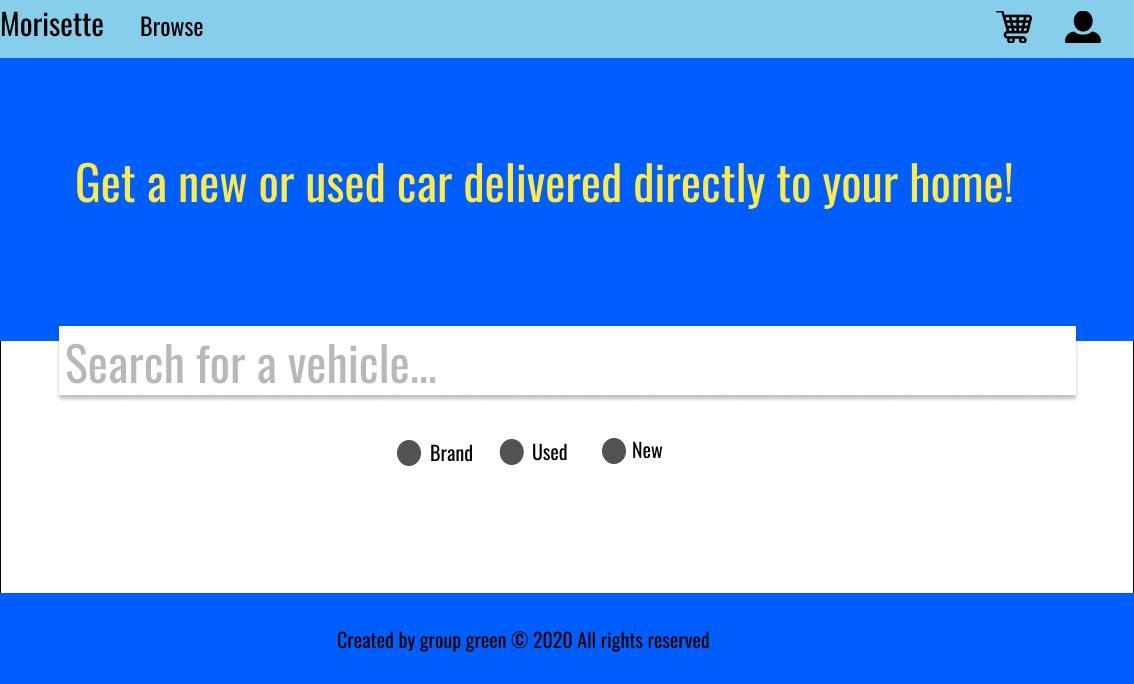
MS\_102



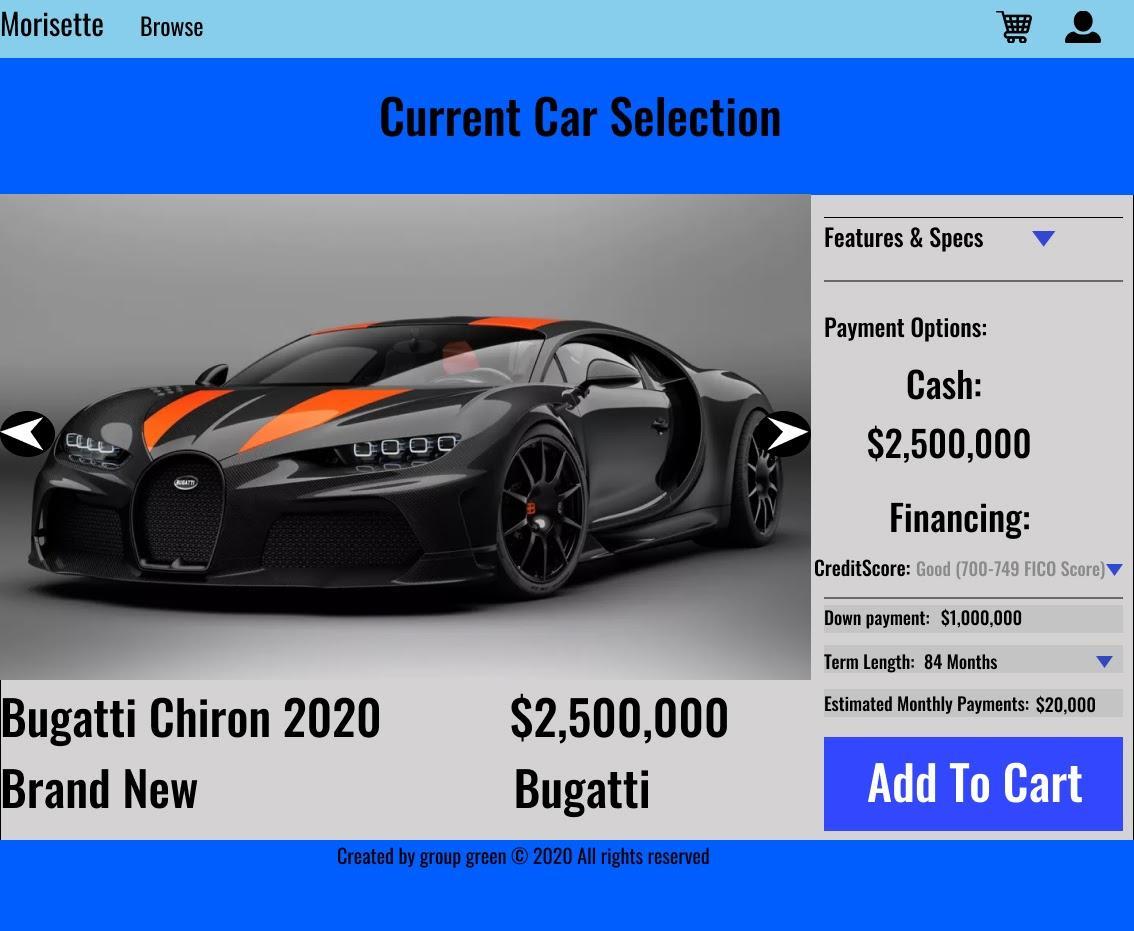
MS\_103



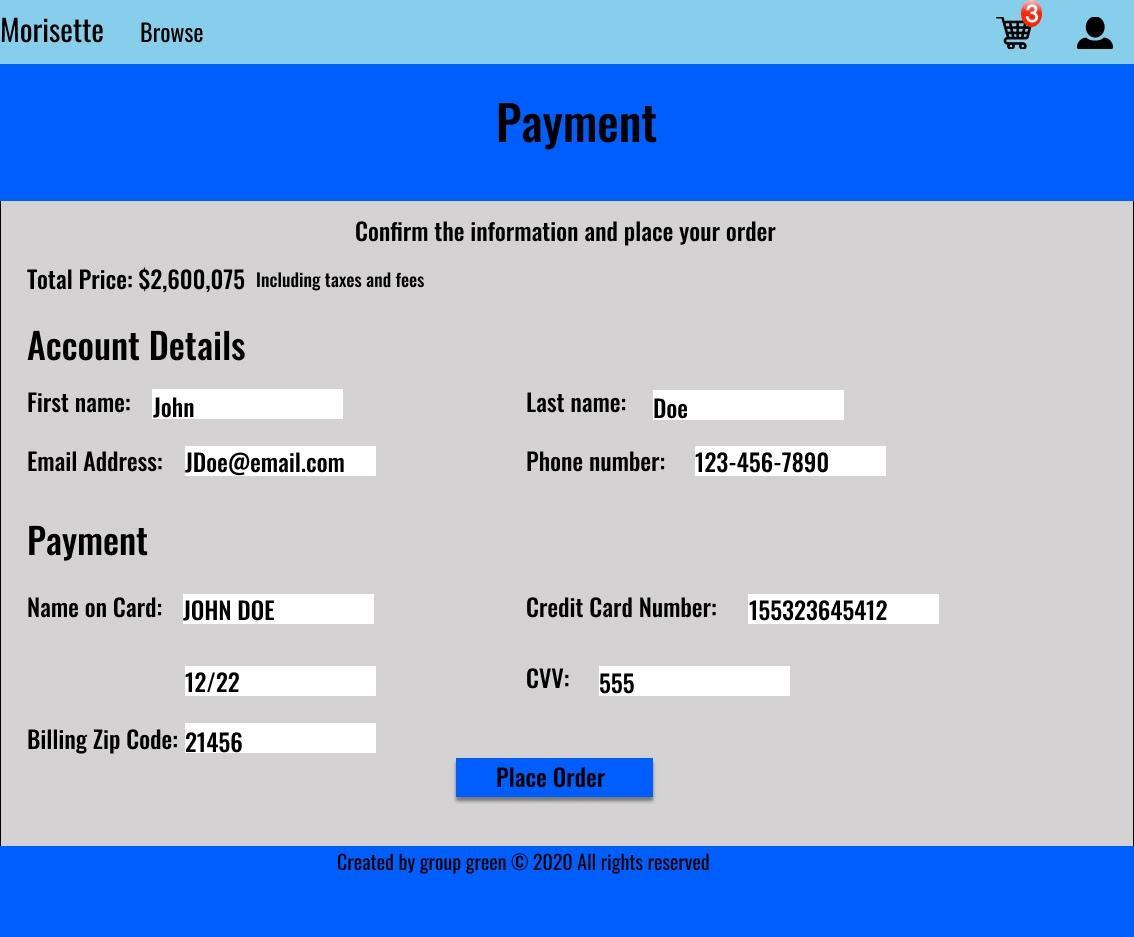
MS\_104



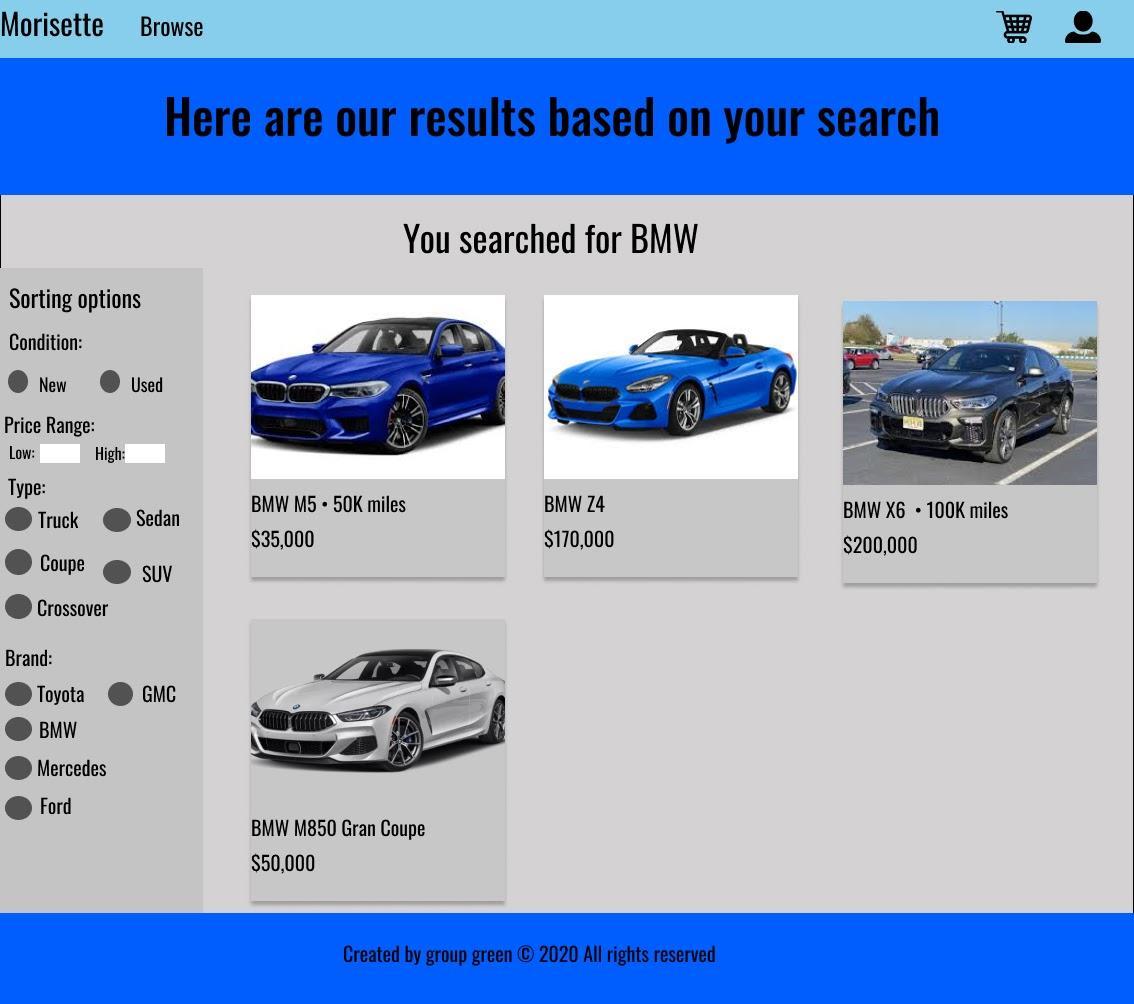
MS\_105



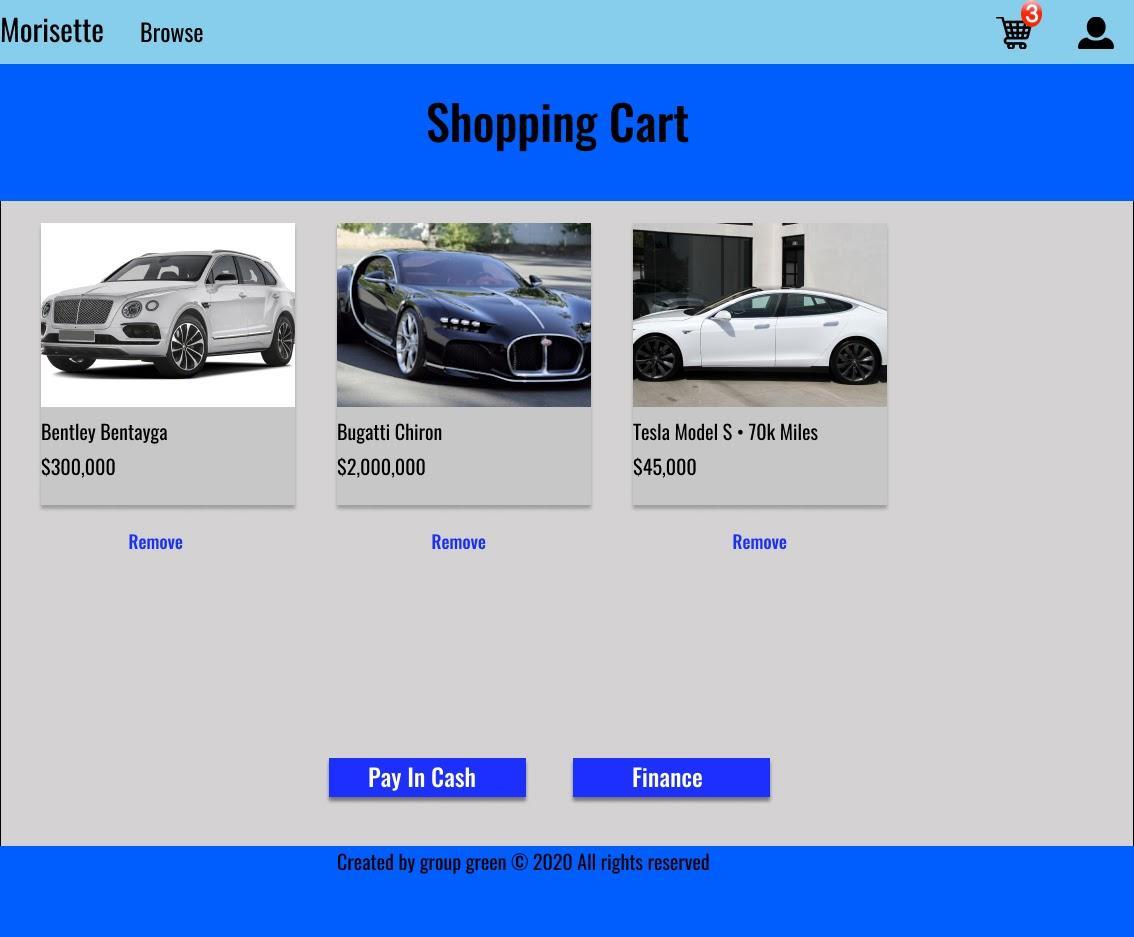
MS\_106



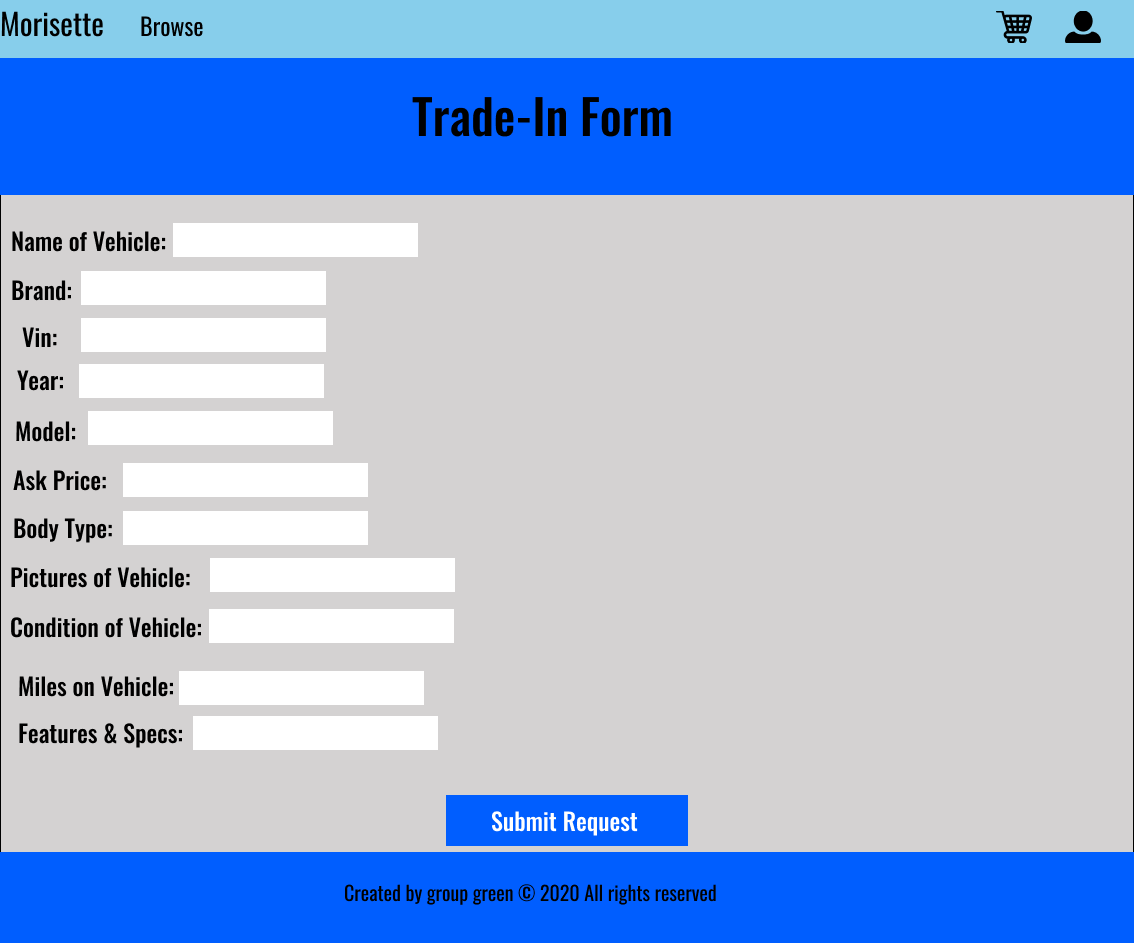
MS\_107



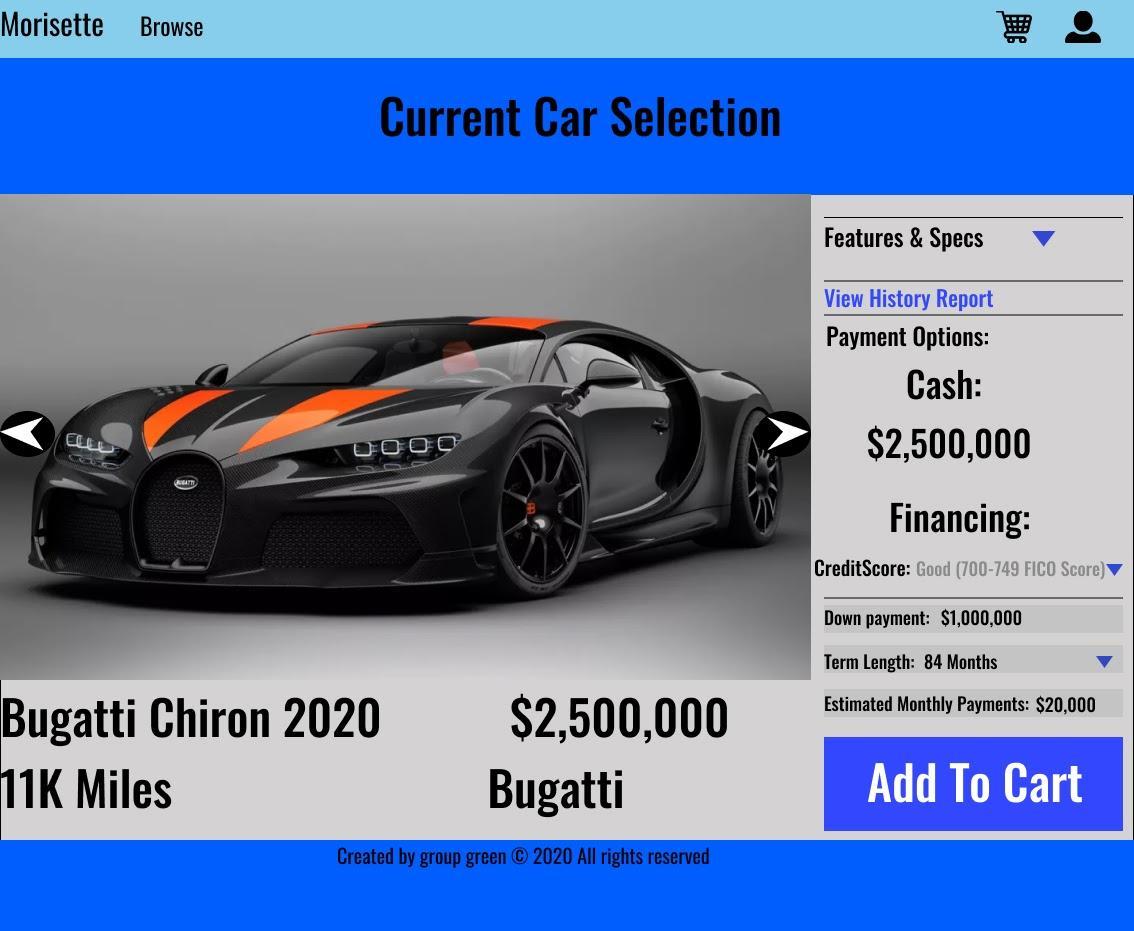
MS\_108



MS\_109

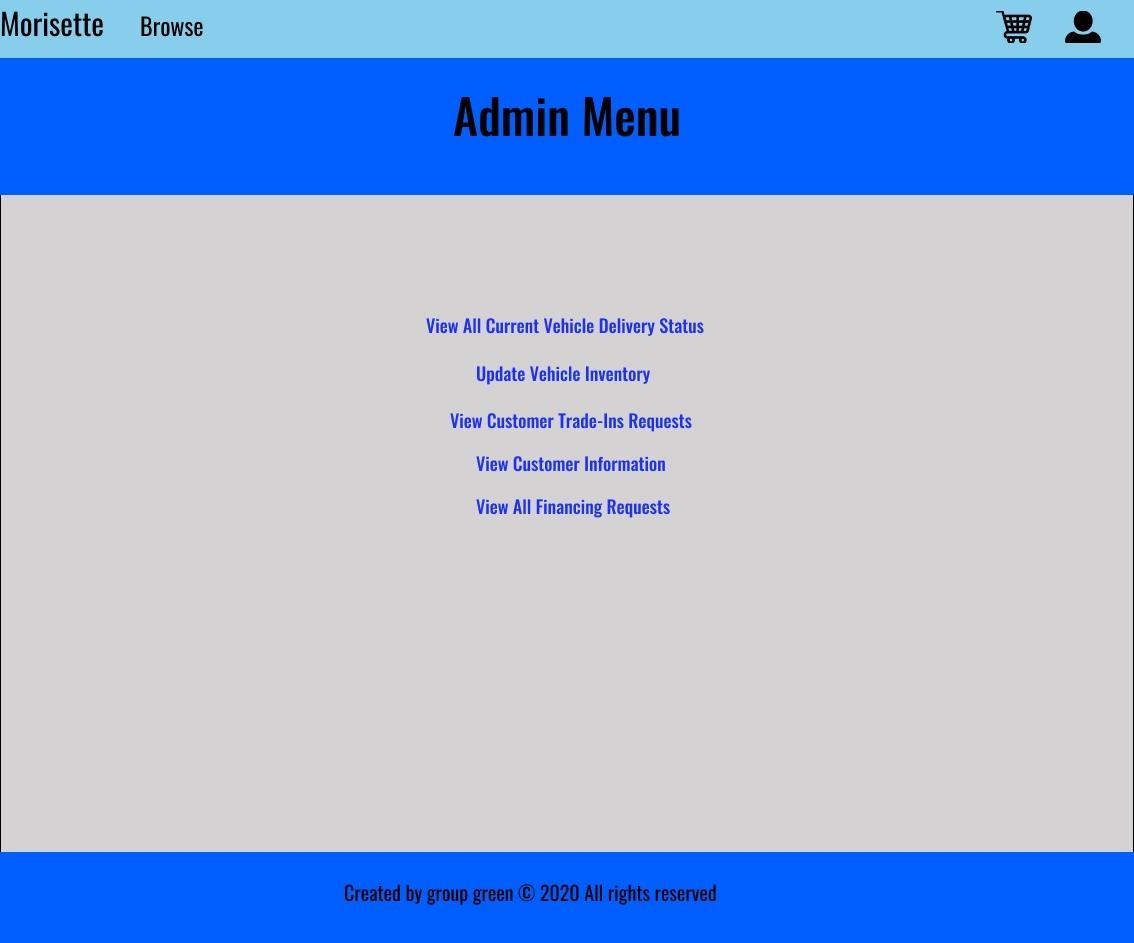


MS\_110

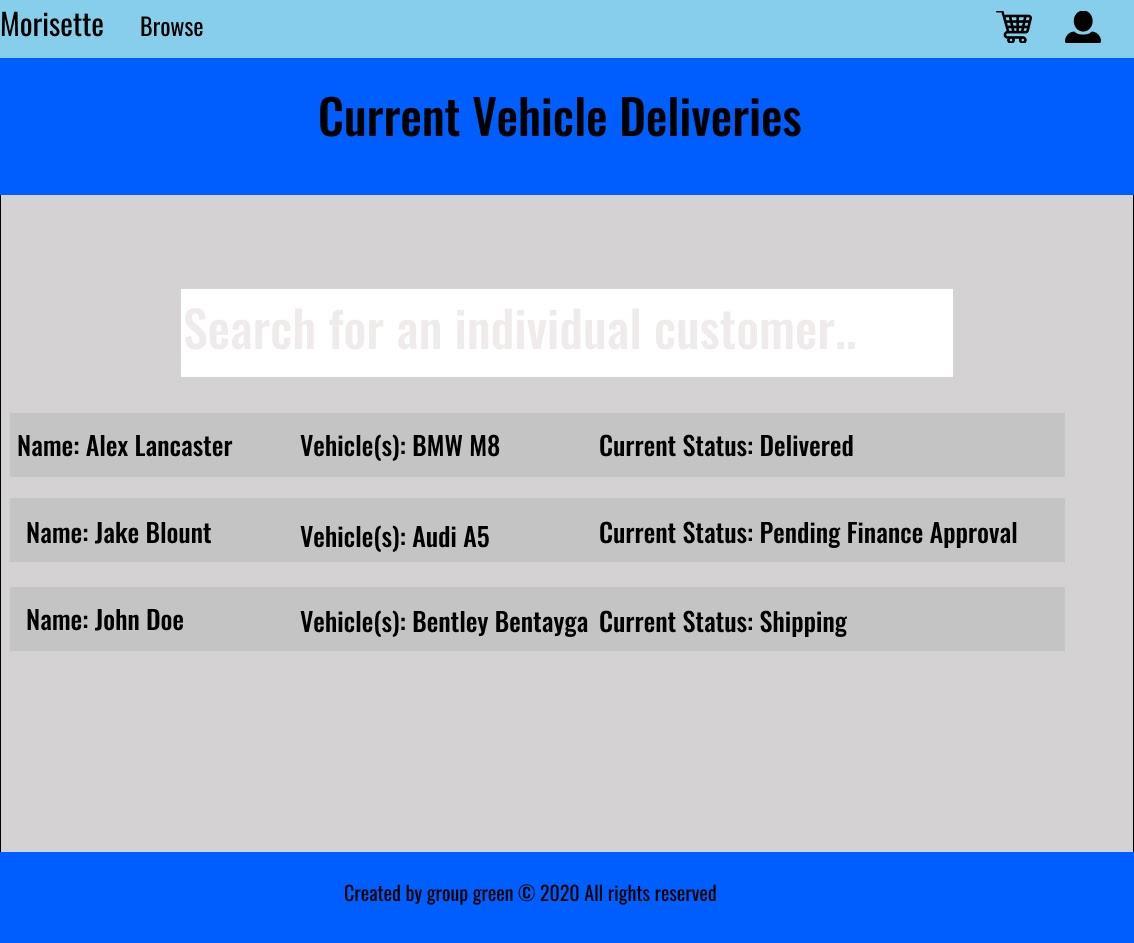


**5.2.1.3  Administrator Menu Screens**

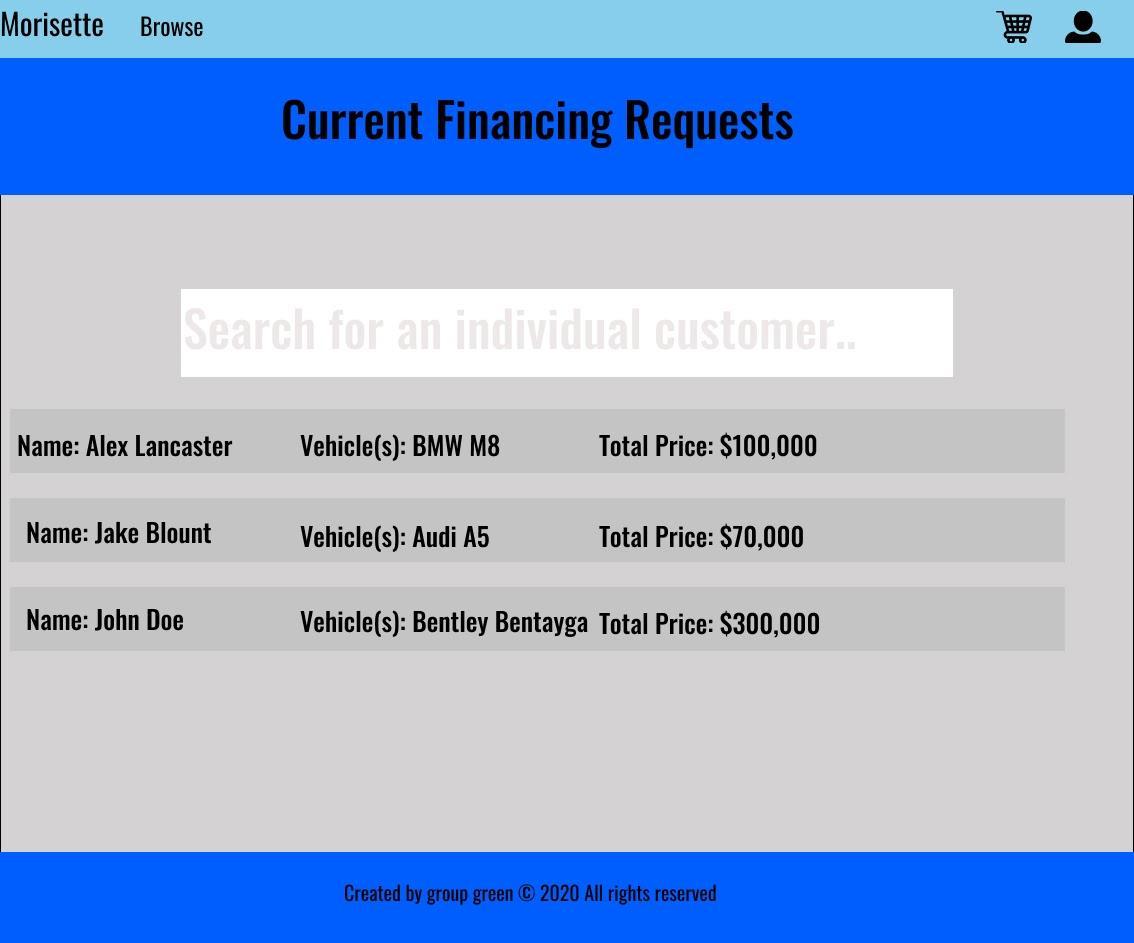
MS\_200



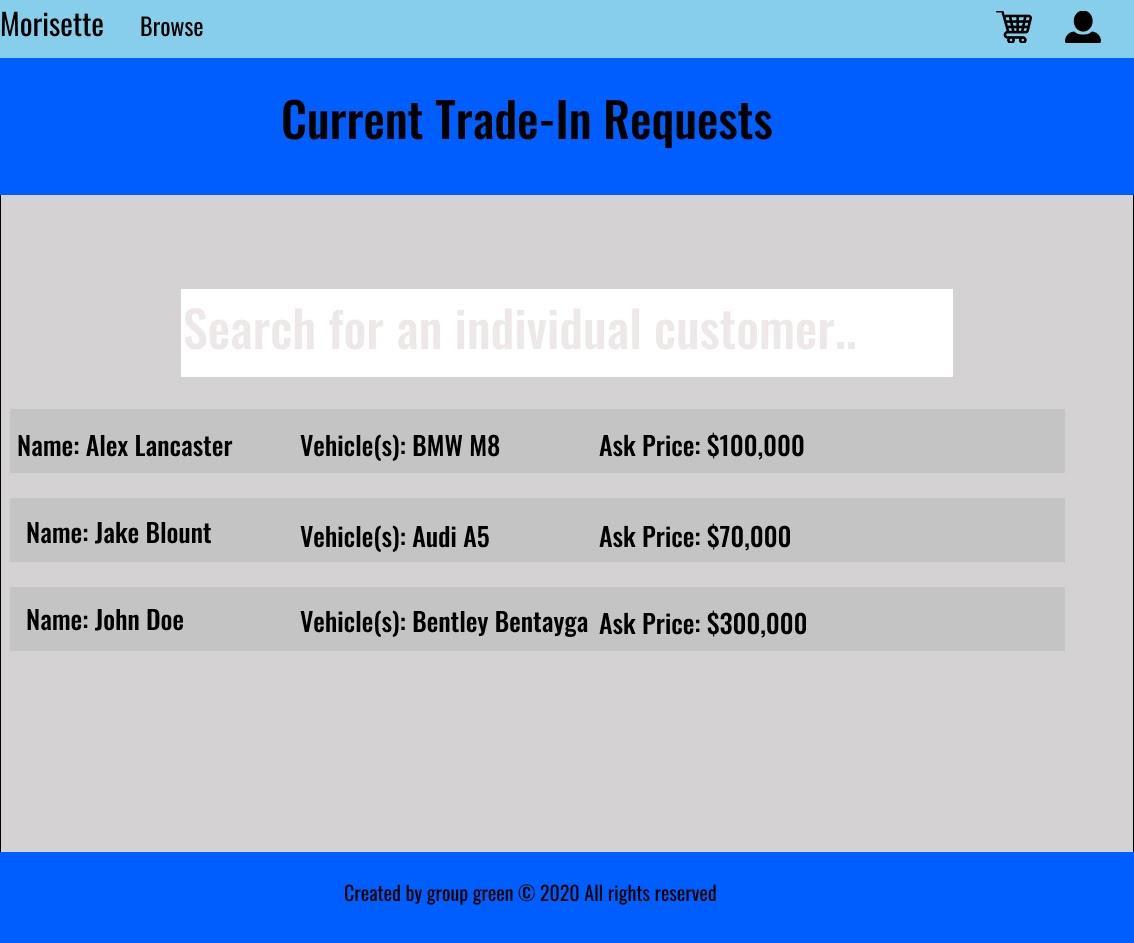
MS\_201



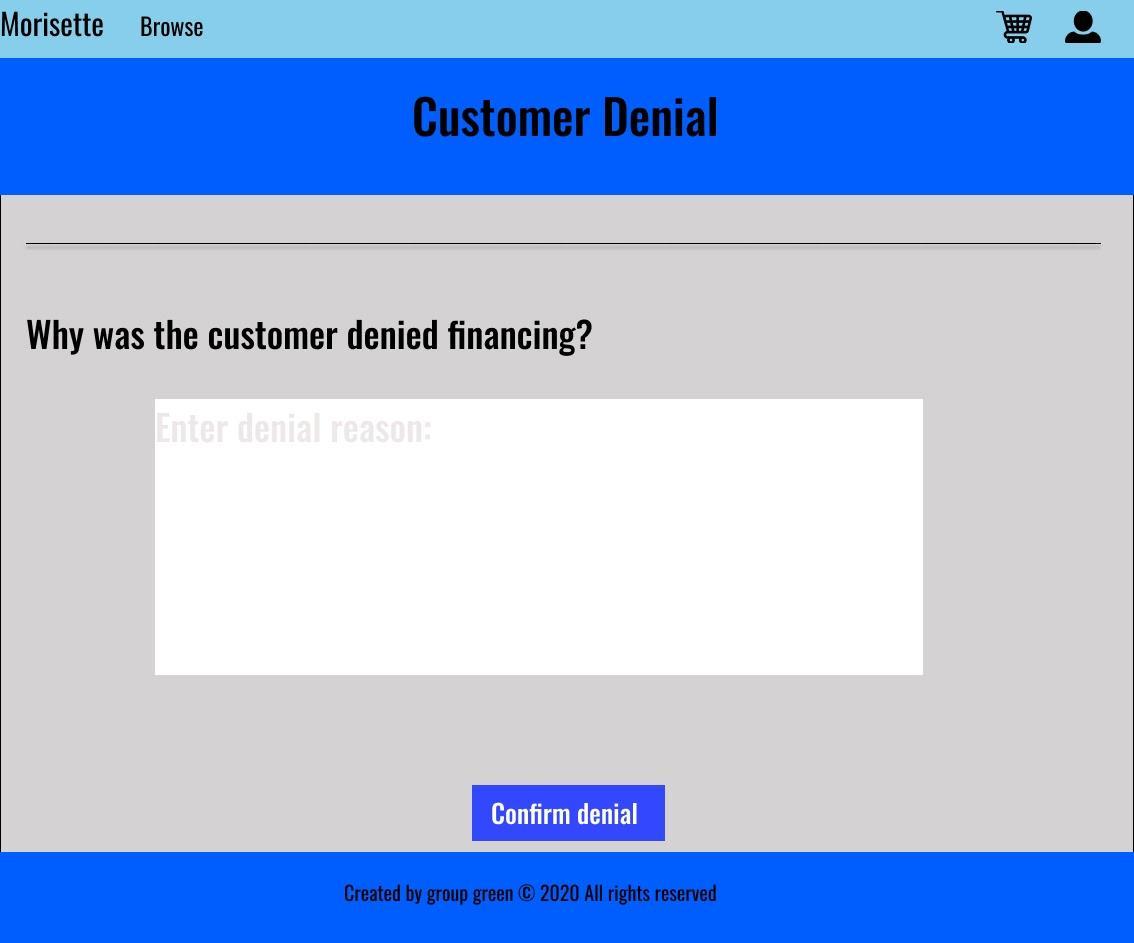
MS\_202



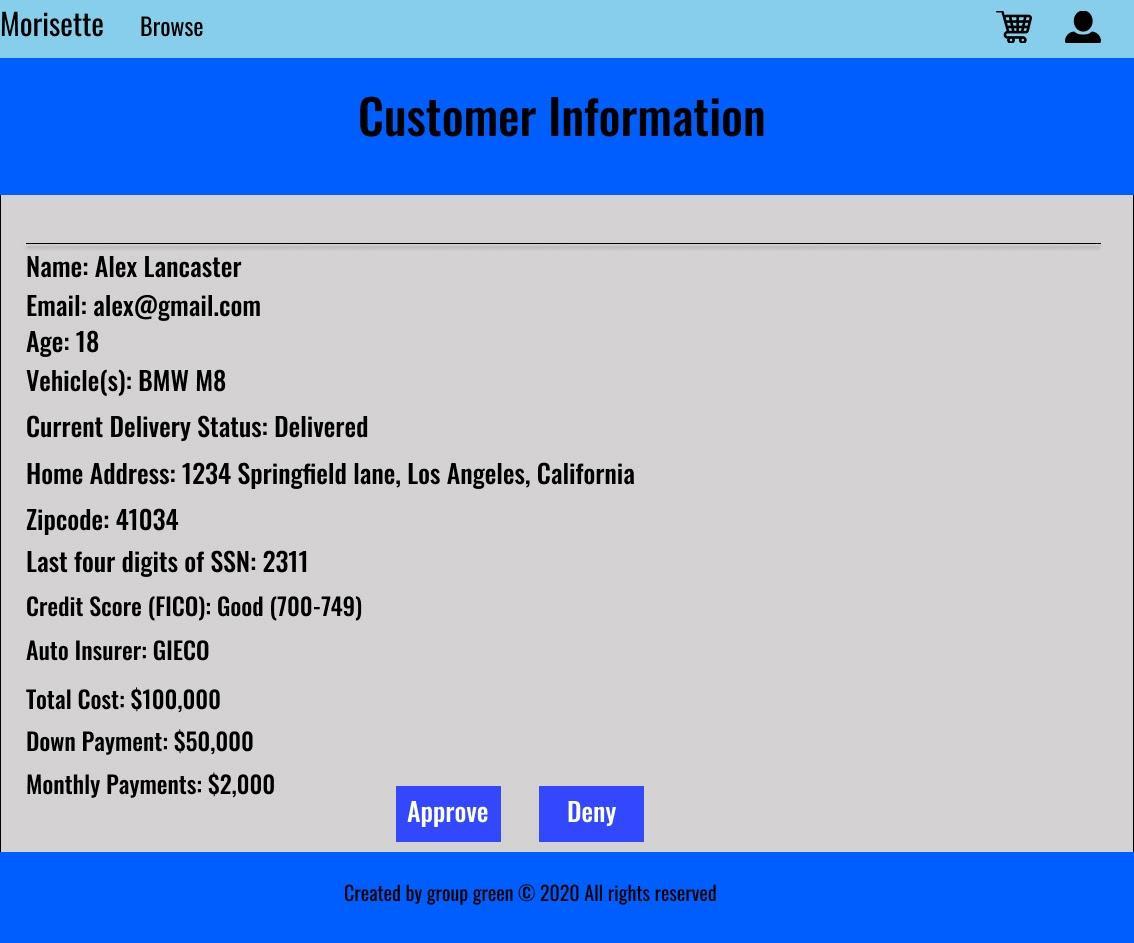
MS\_203



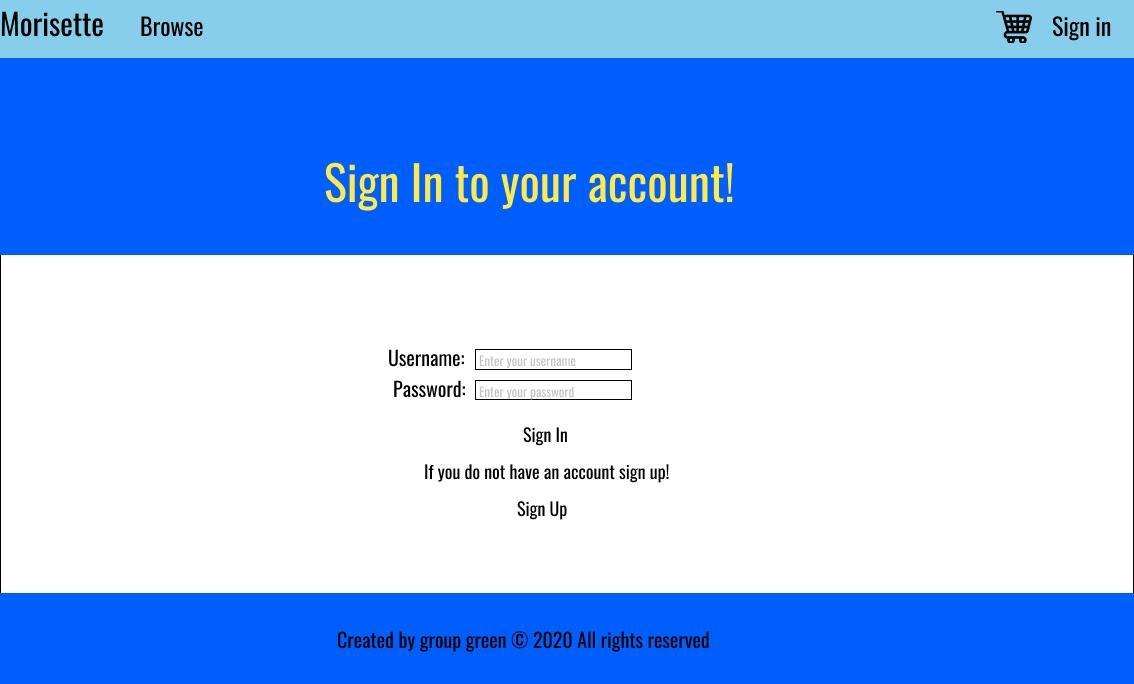
MS\_204



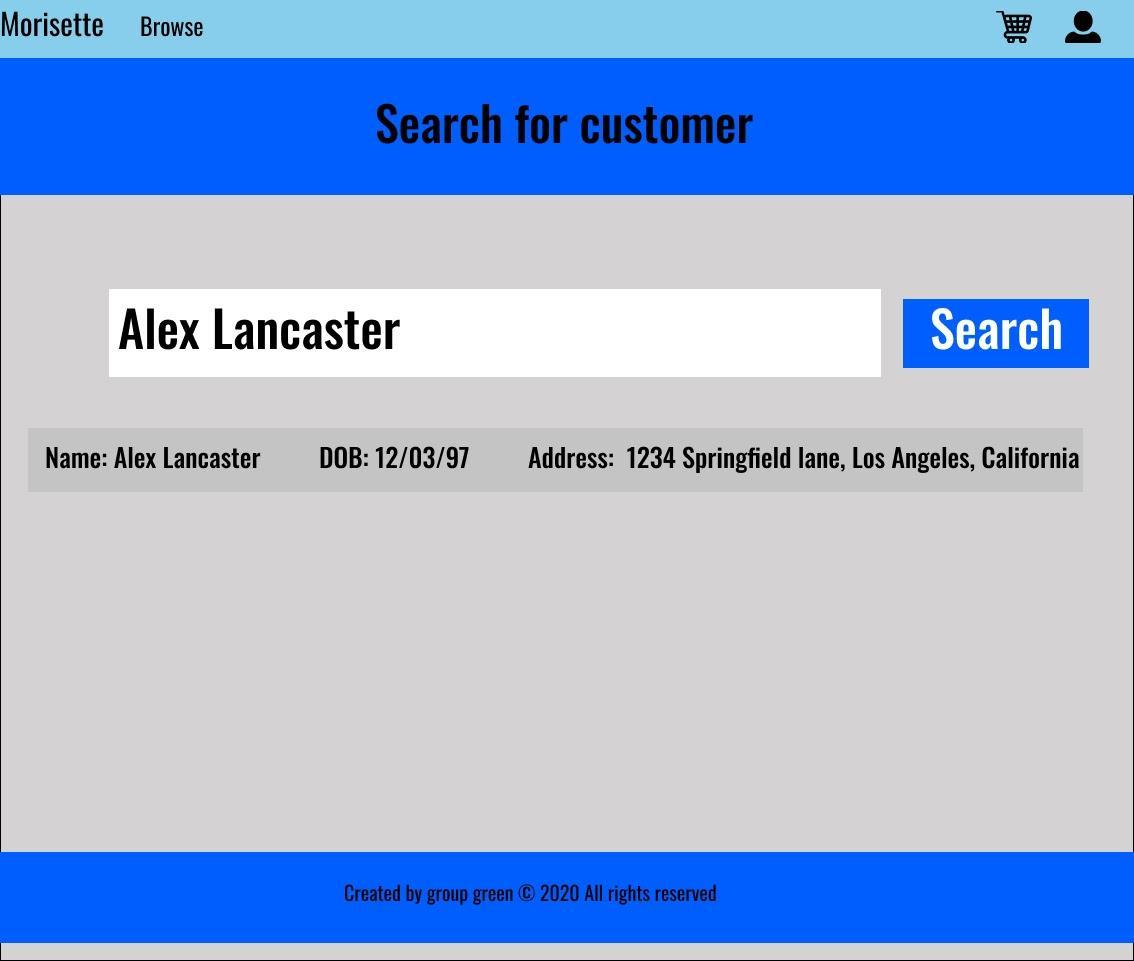
MS\_205



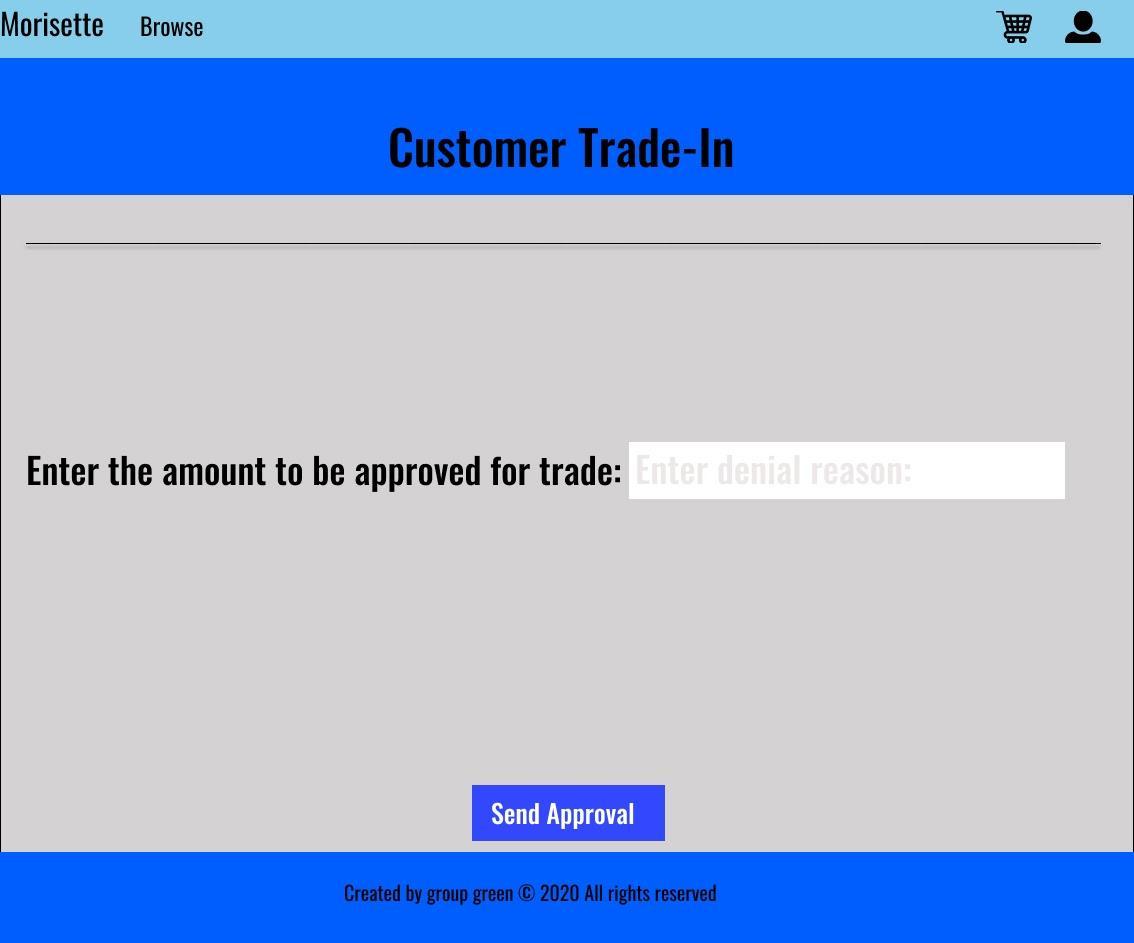
MS\_206



MS\_207



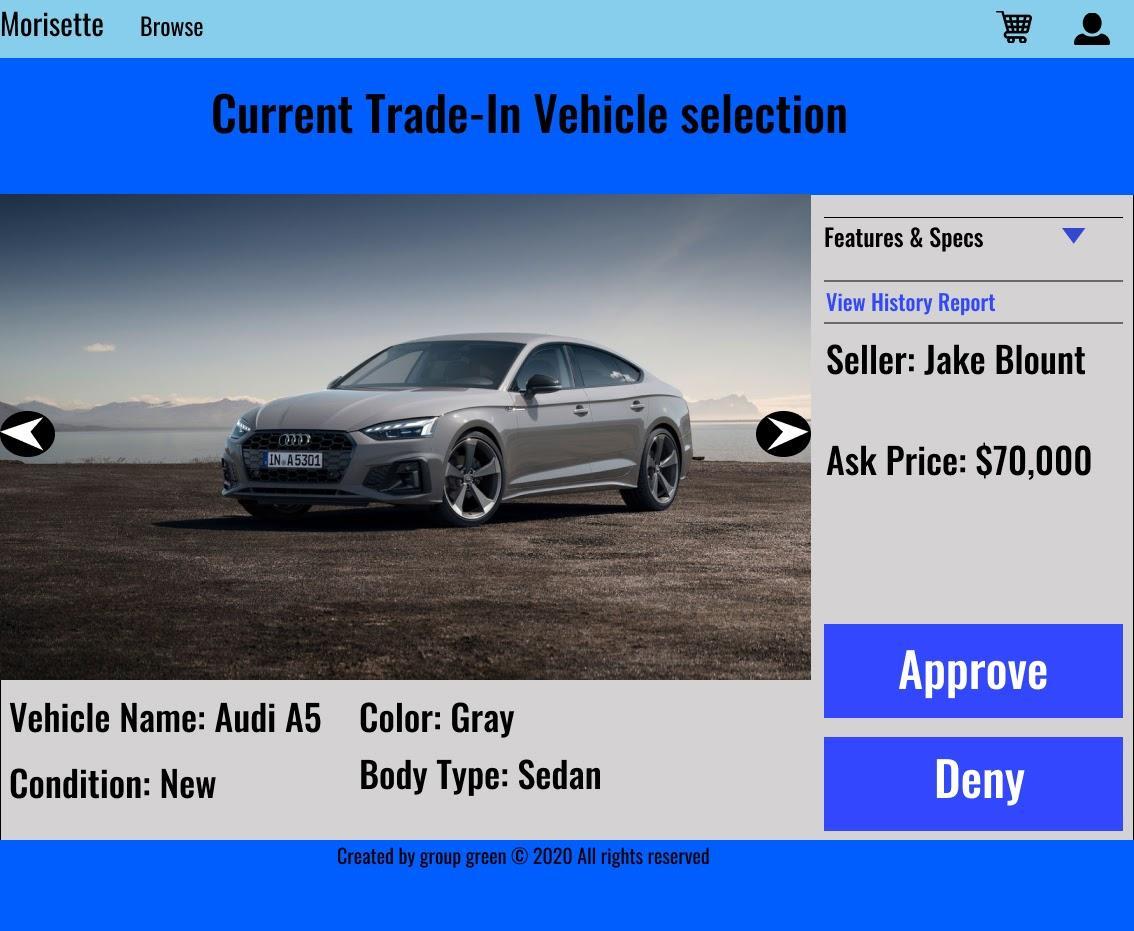
MS\_208



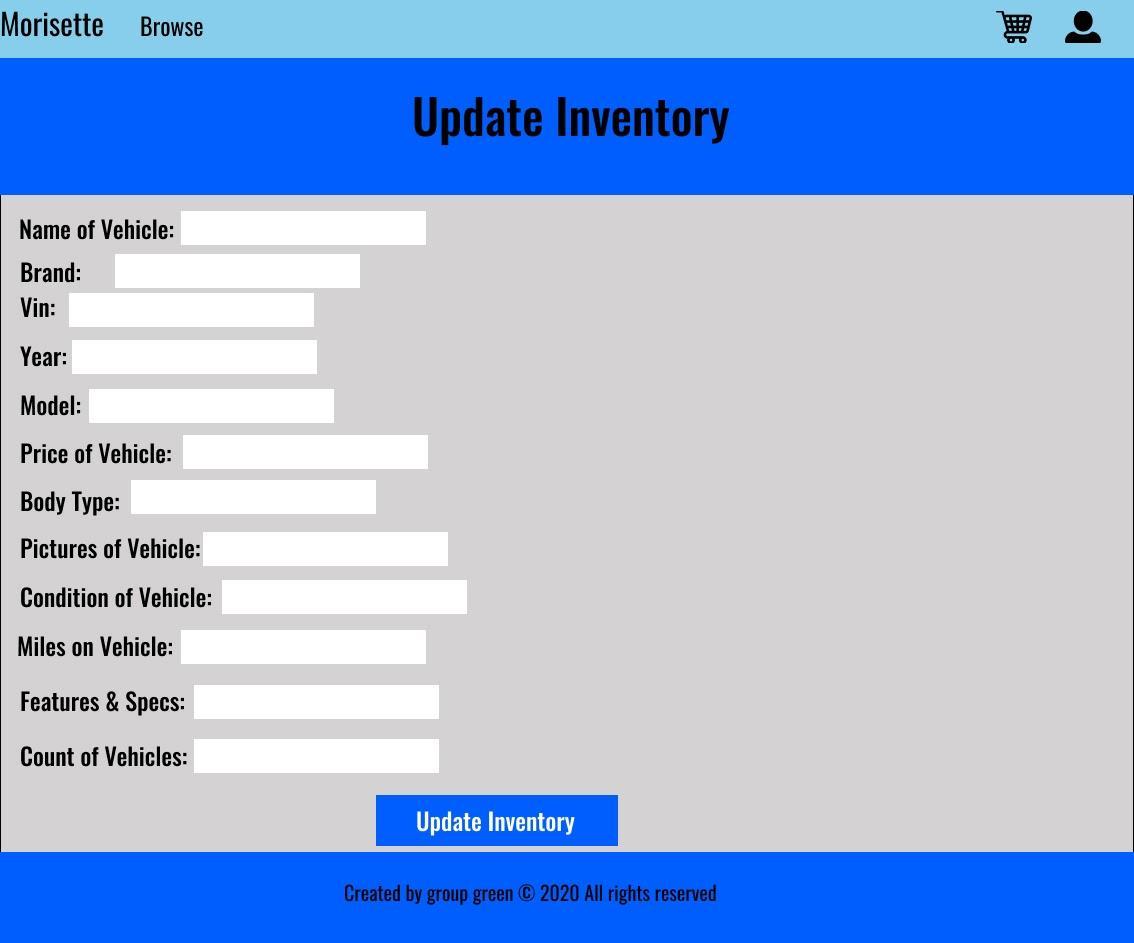
MS\_209



MS\_210



MS\_211

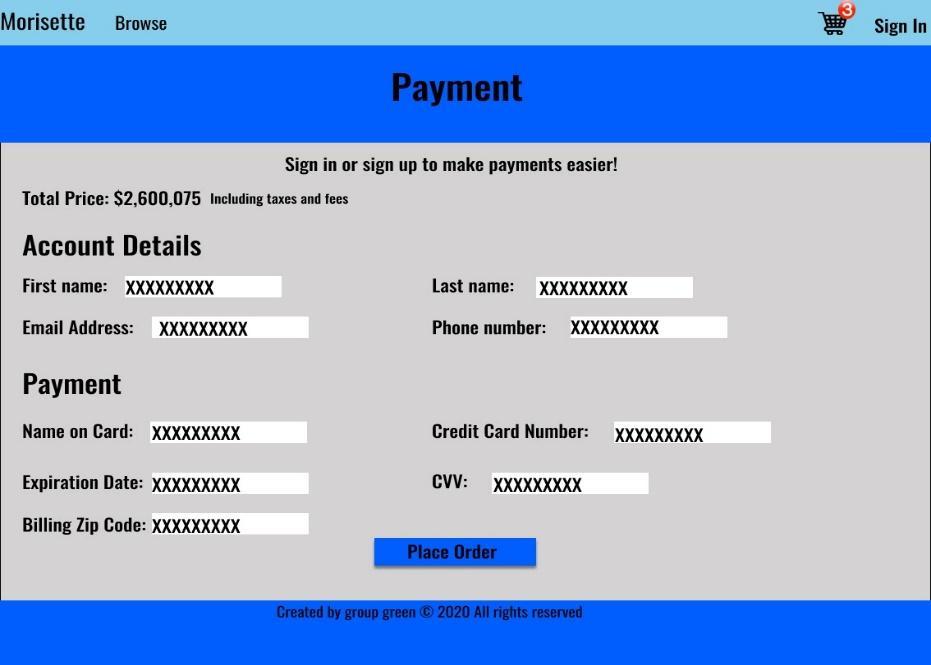


* + 1. **Input and Output screens**
       1. **Input/Output Guest Screens**

**Payment:**

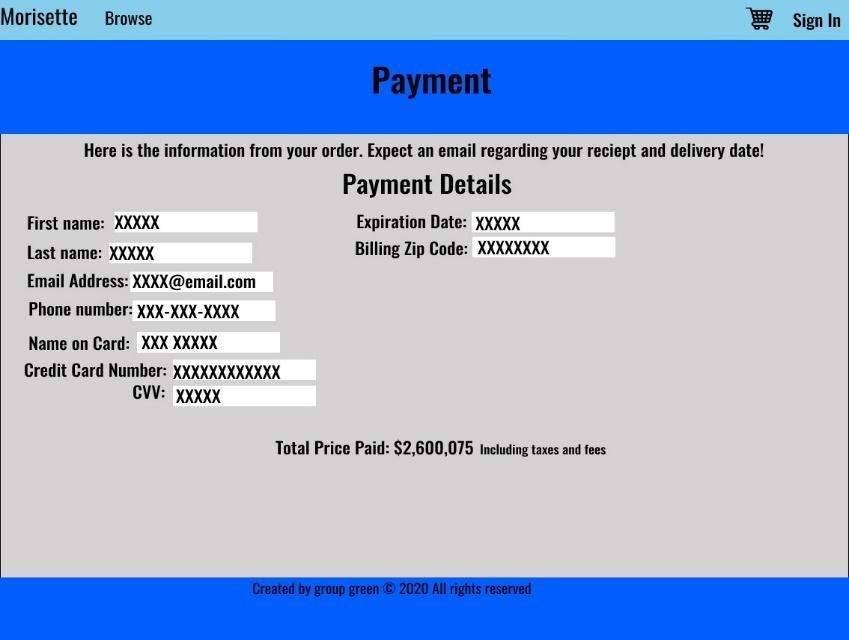
**Input:**

  IOS\_001

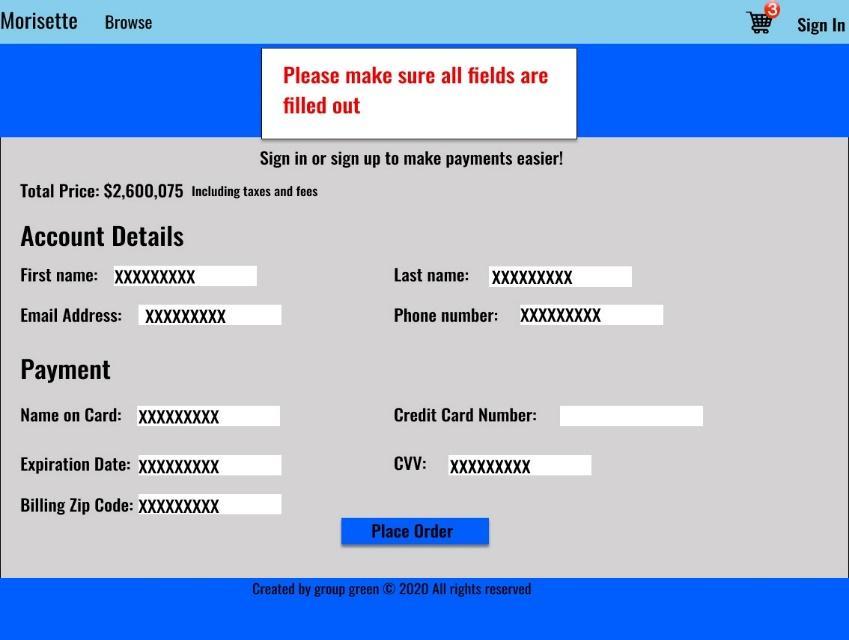


**Output:**

IOS\_002



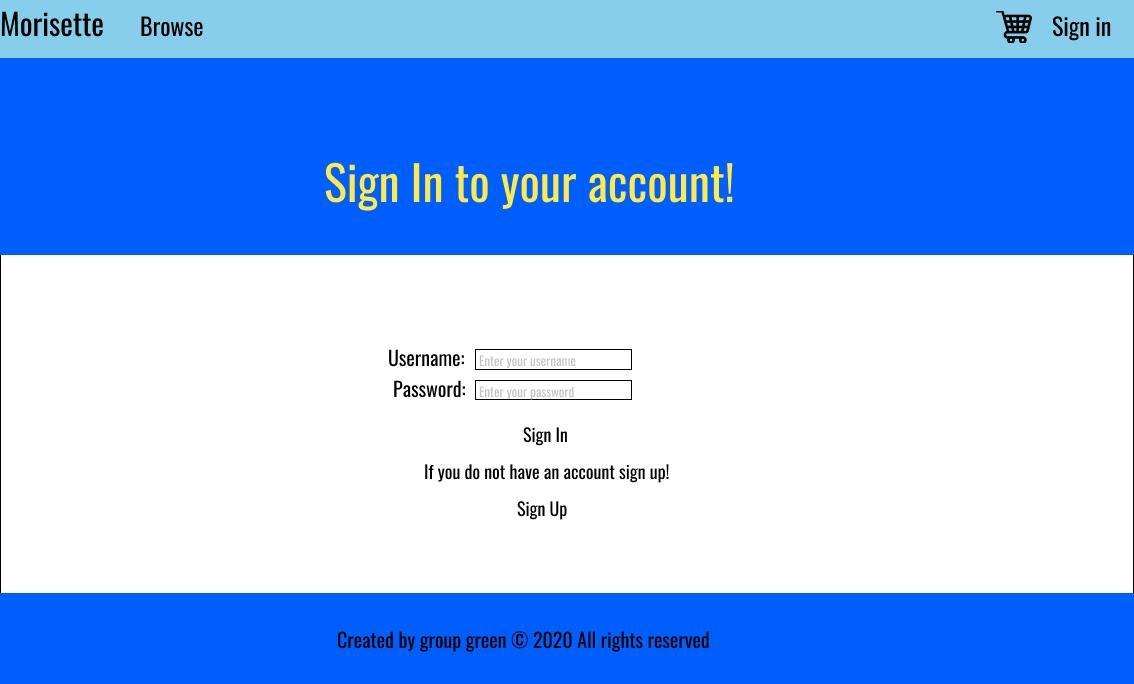
IOS\_003



**Sign In:**

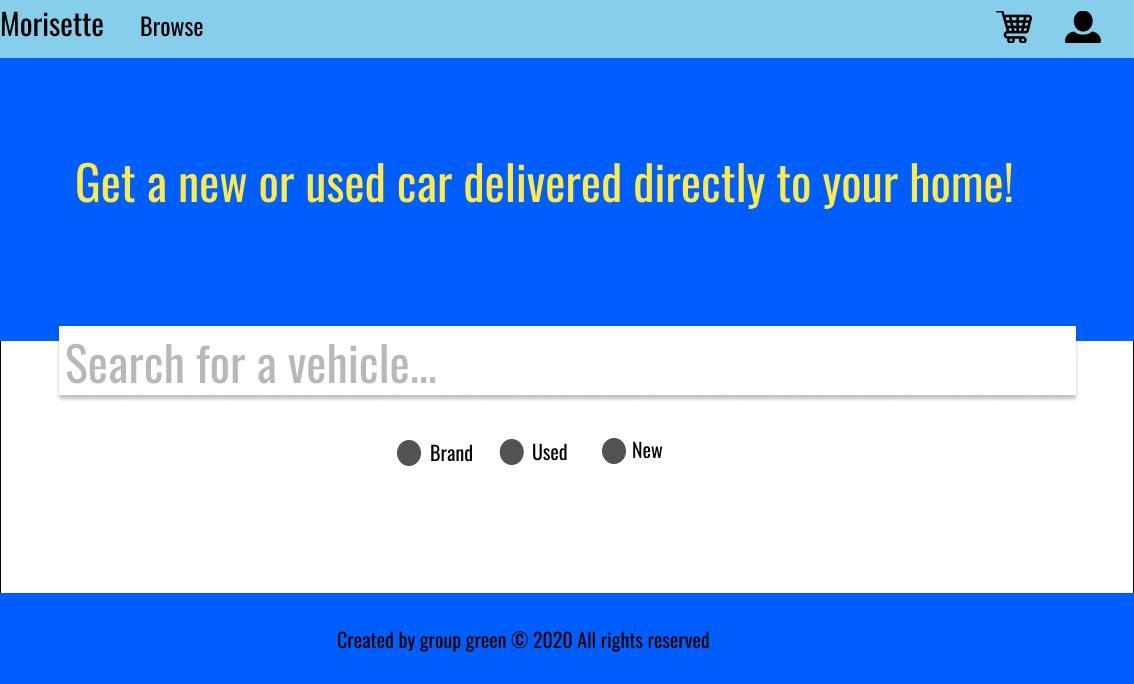
**Input:**

IOS\_004



**Output:**

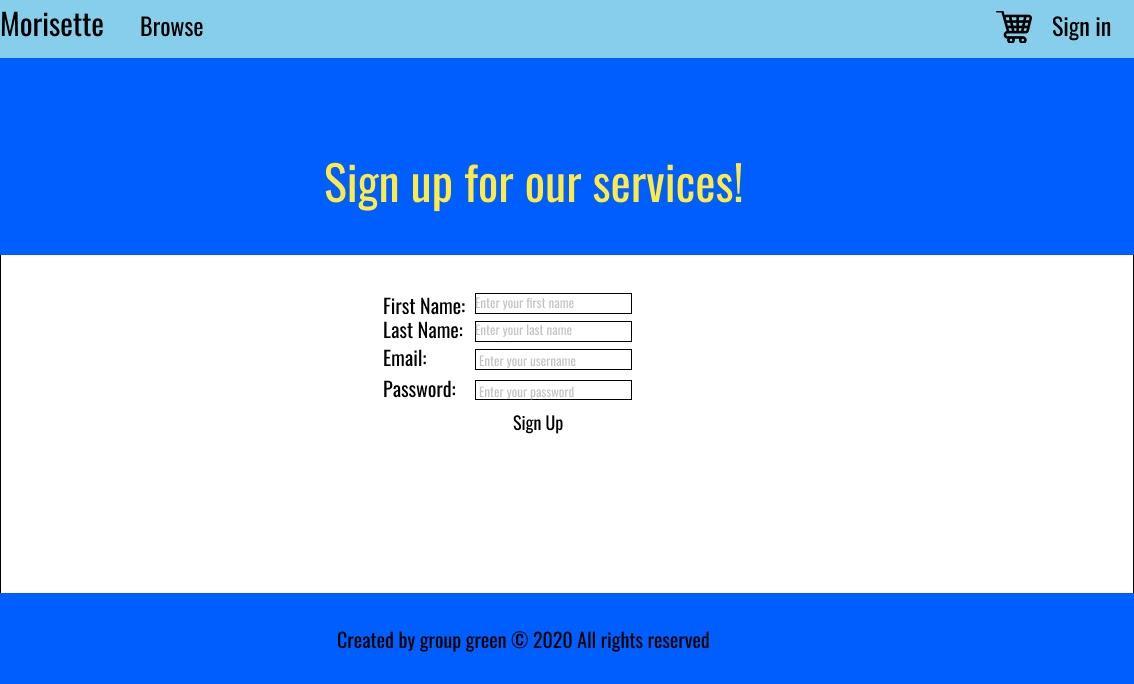
IOS\_005



**Sign Up:**

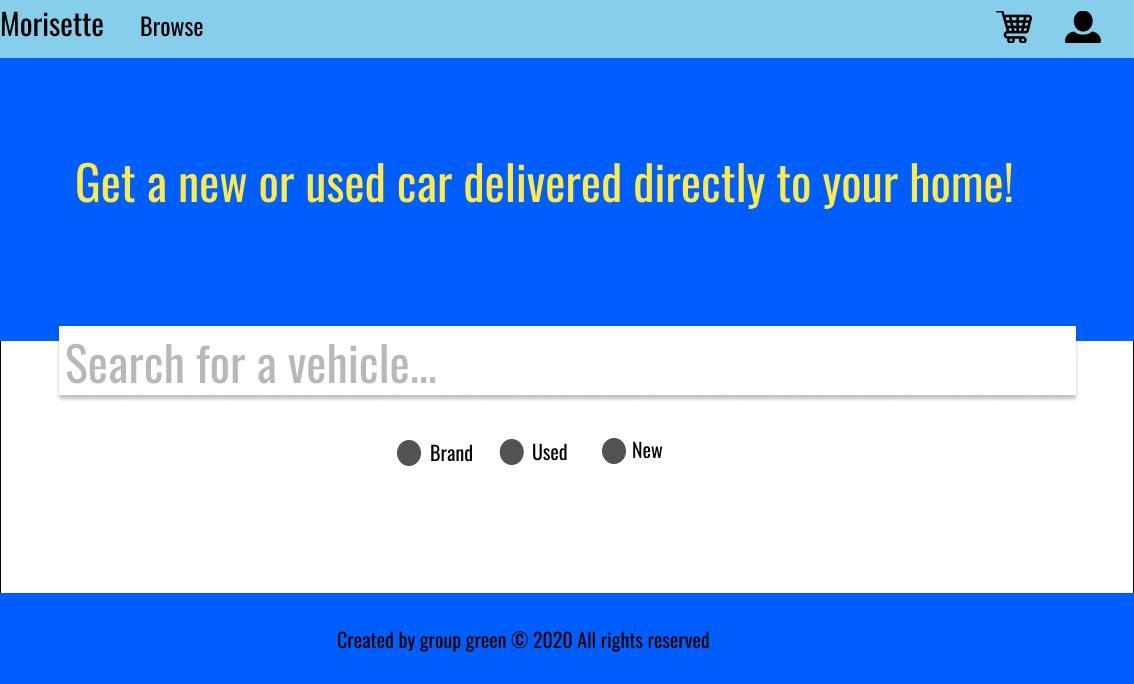
**Input:**

IOS\_004



**Output:**

IOS\_005

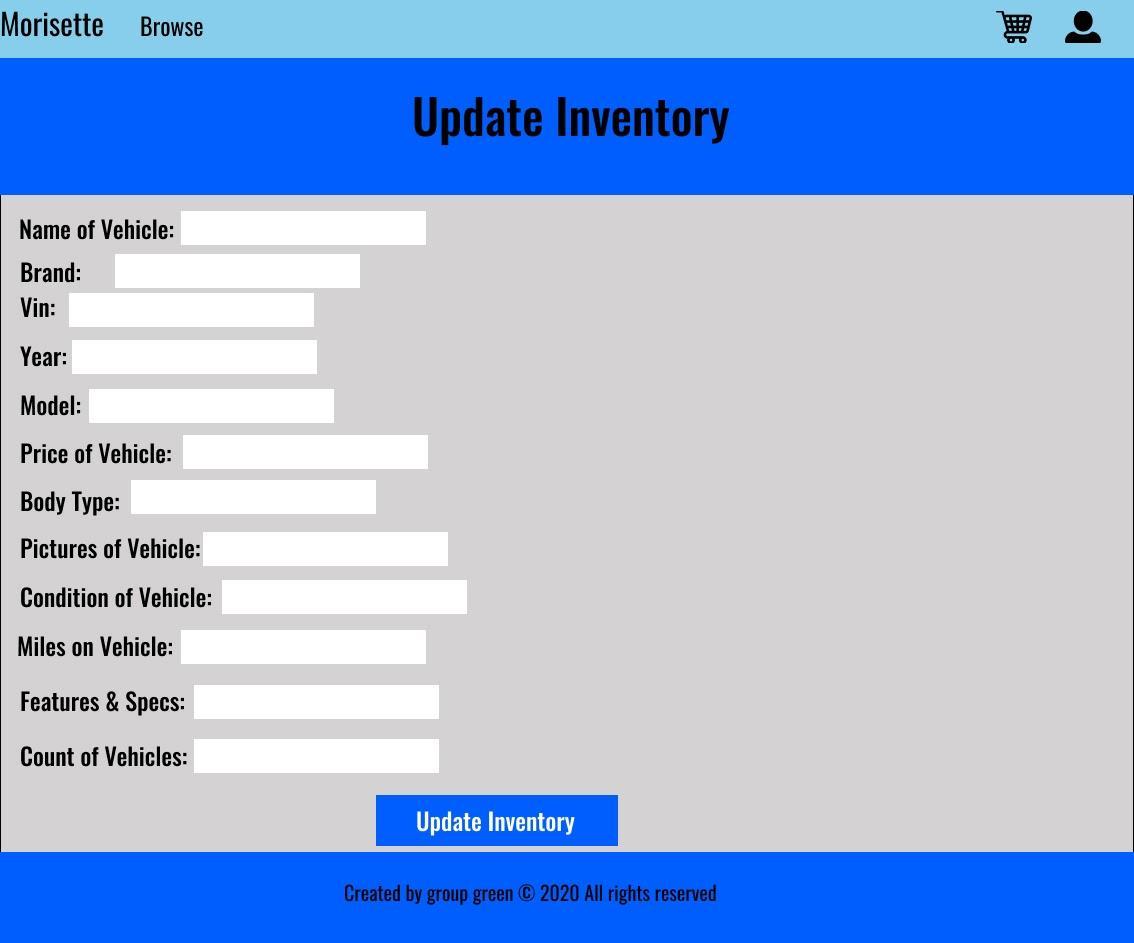


**5.1.2.2 Input/Output Administrator Screens**

**Update Vehicle Inventory:**

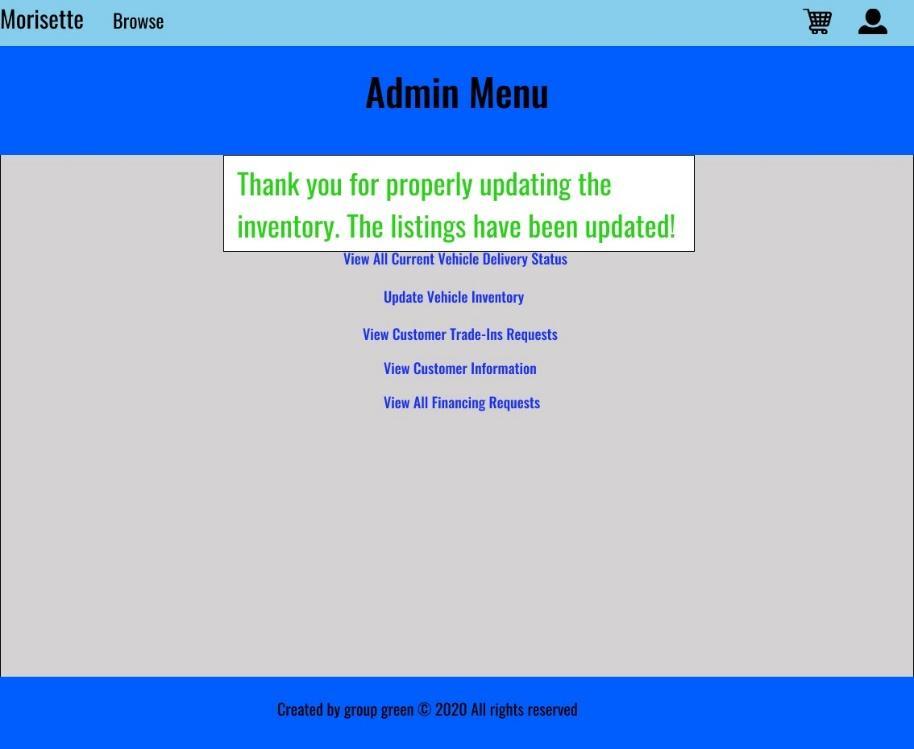
**Input:**

IOS\_101

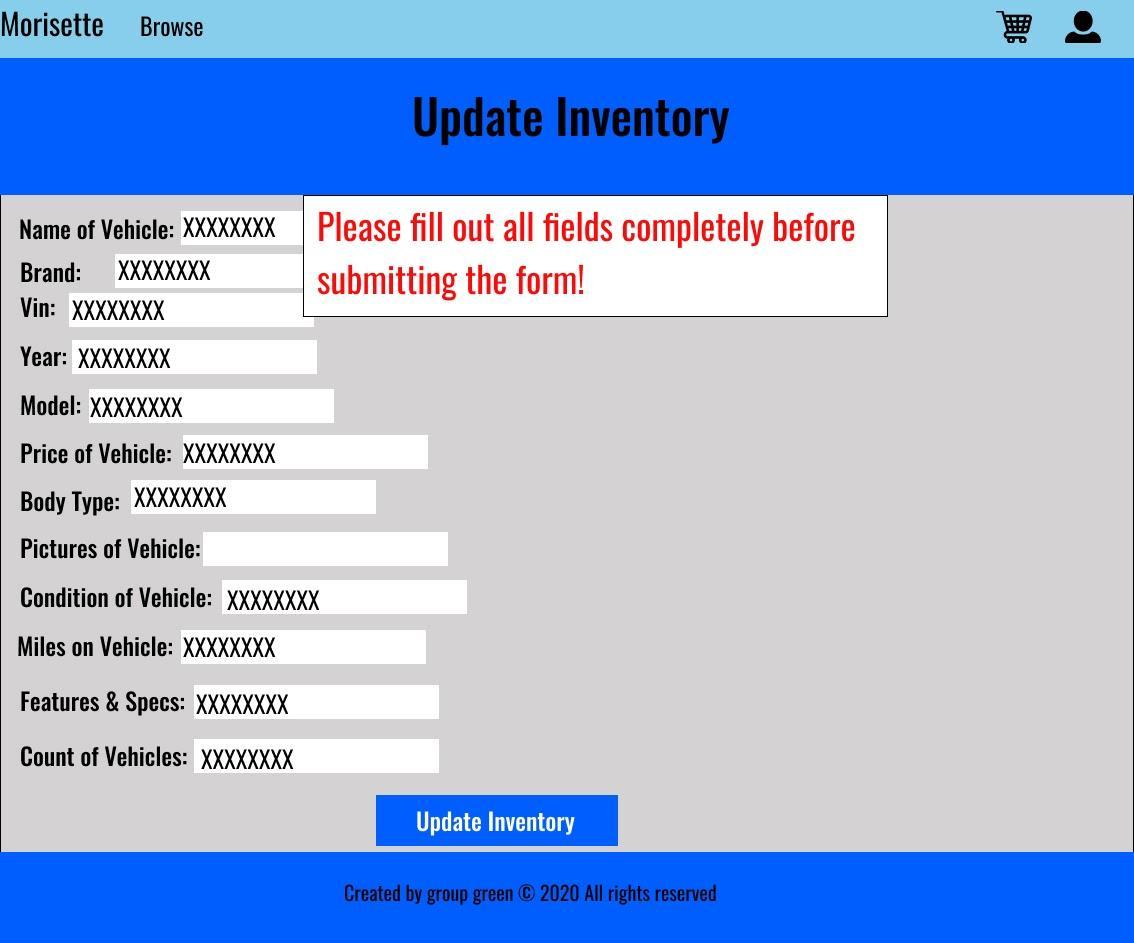


**Output:**

IOS\_102



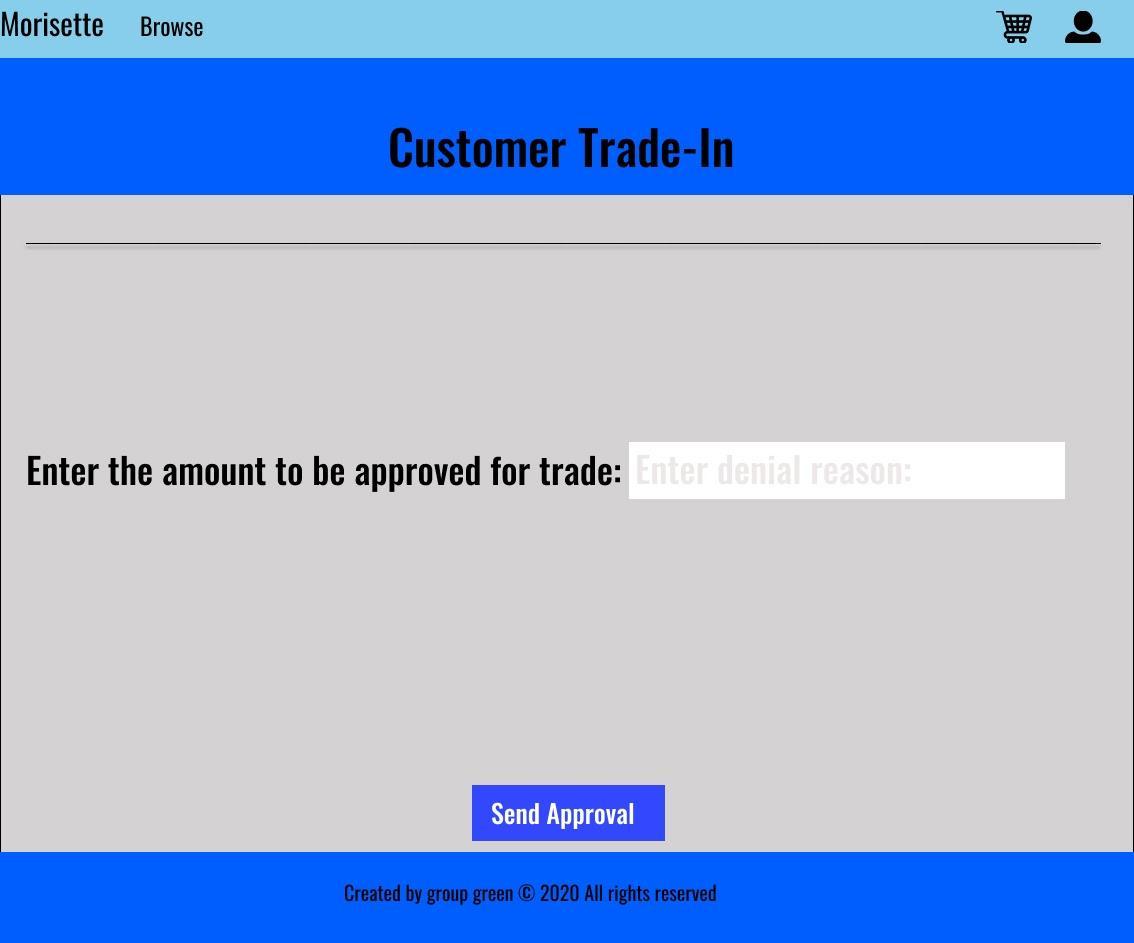
IOS\_103



**Trade-In Approved:**

**Input:**

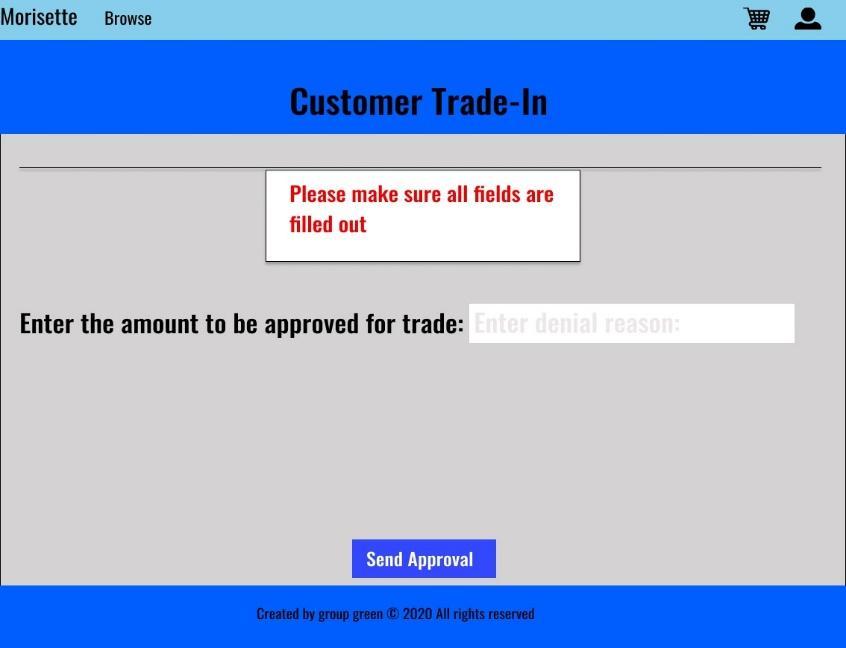
IOS\_104



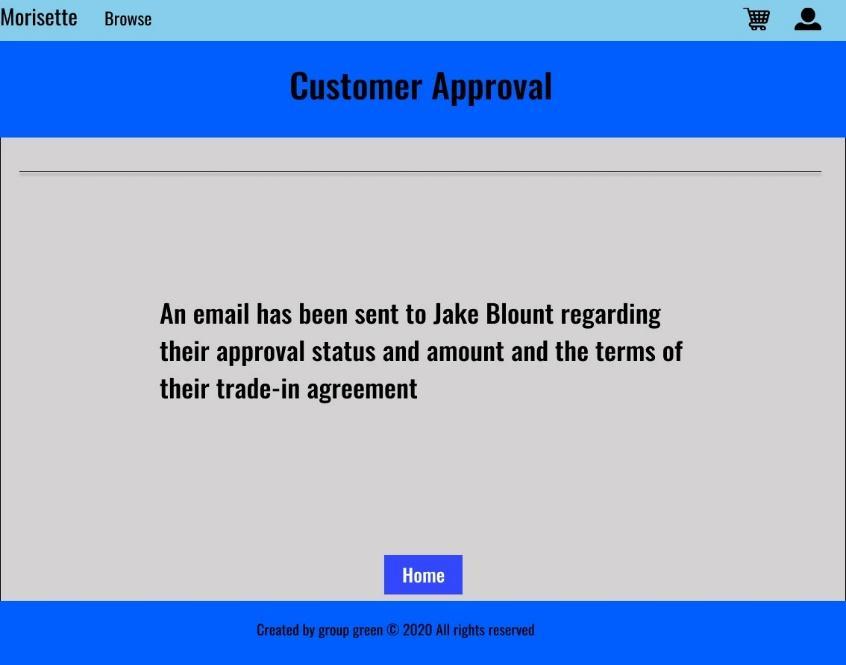
**Output: Output**

**Output:**

IOS\_105



IOS\_106



**Trade-In Denied**

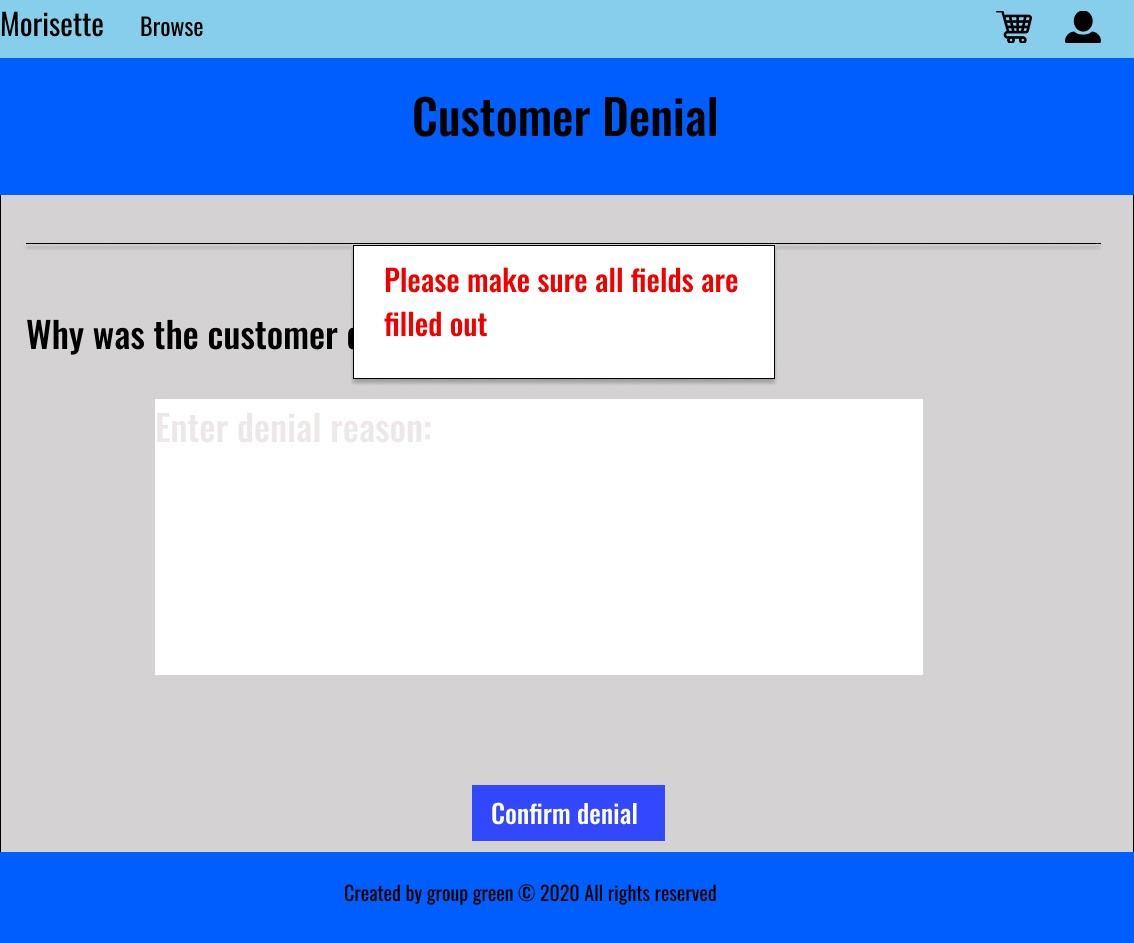
**Input:**

IOS\_106

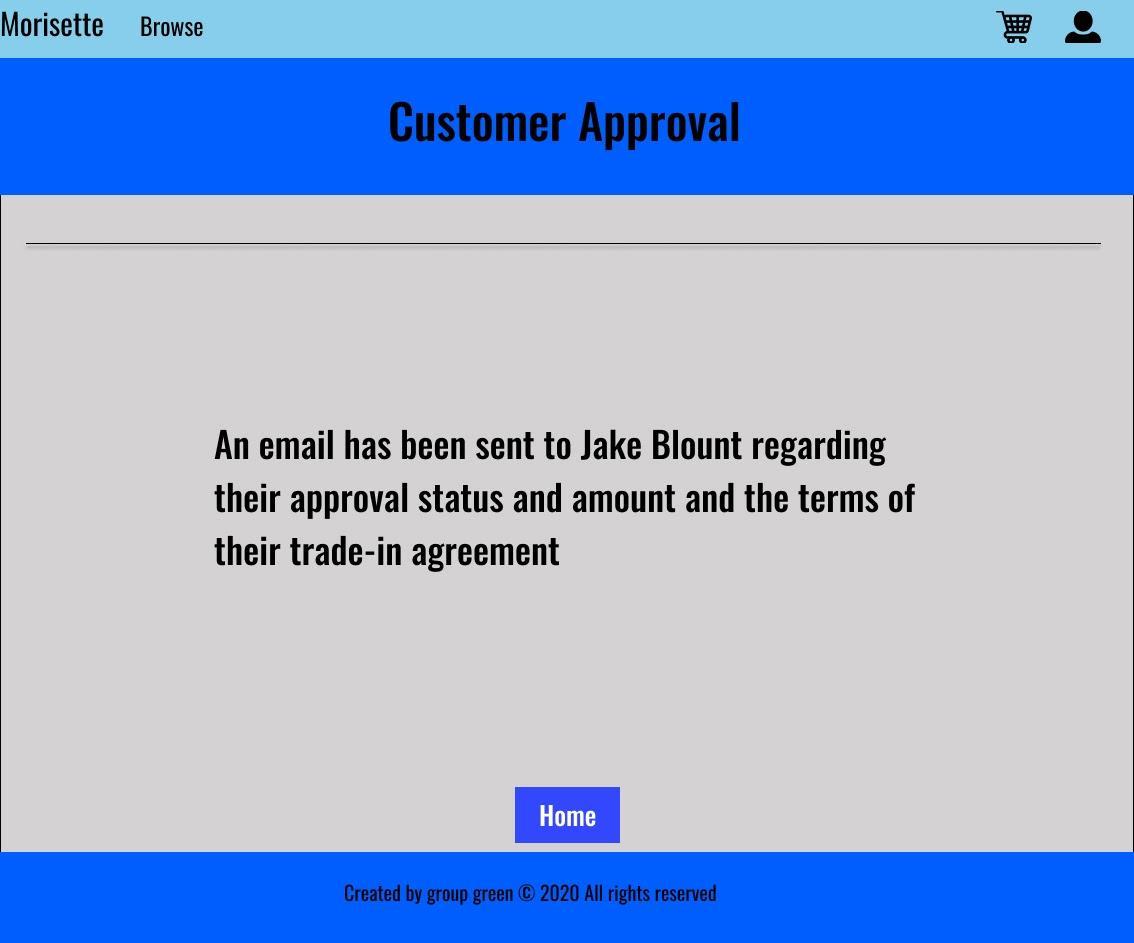


**Output:**

IOS\_107



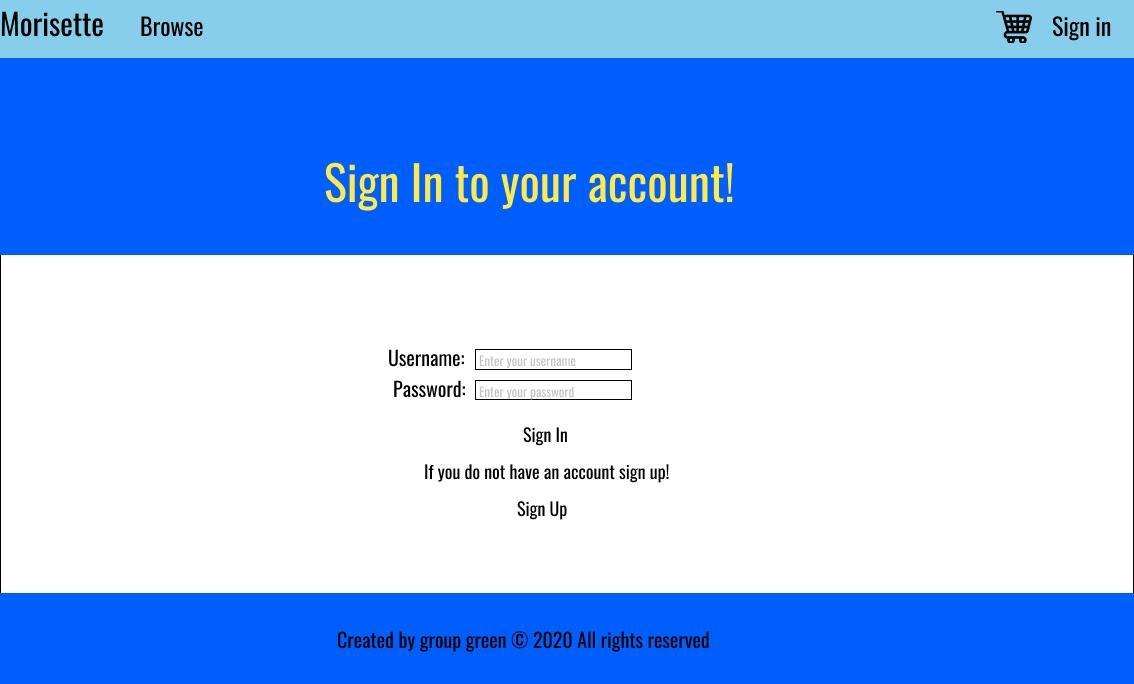
IOS\_108



**Signed In:**

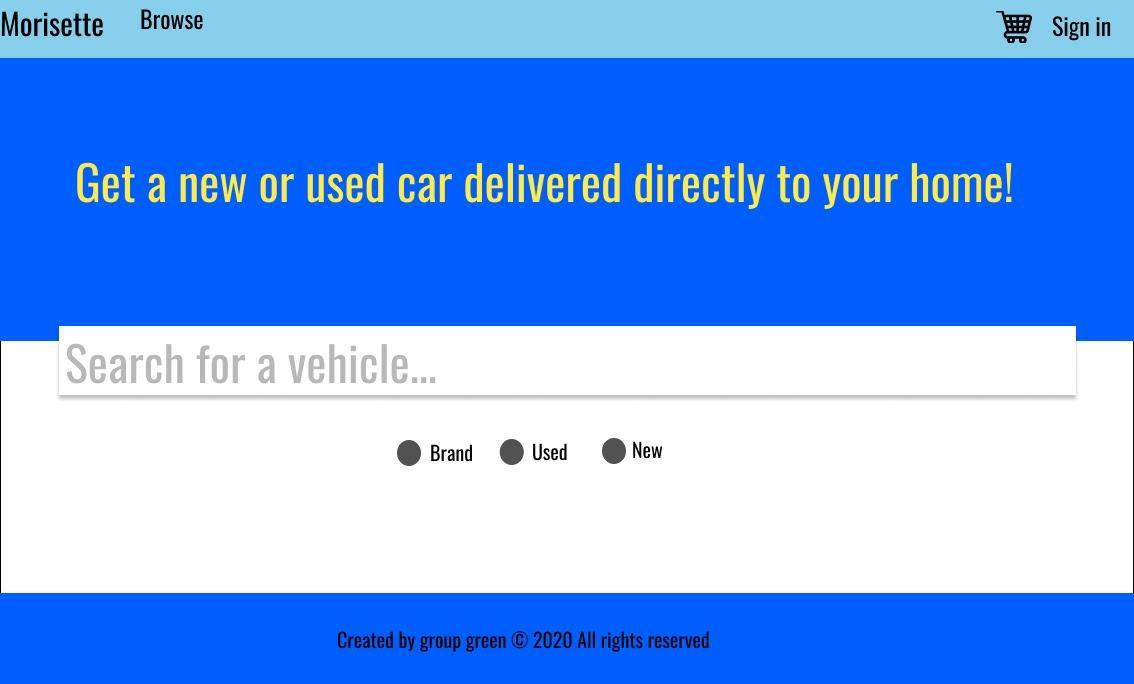
**Input:**

IOS\_109



**Output:**

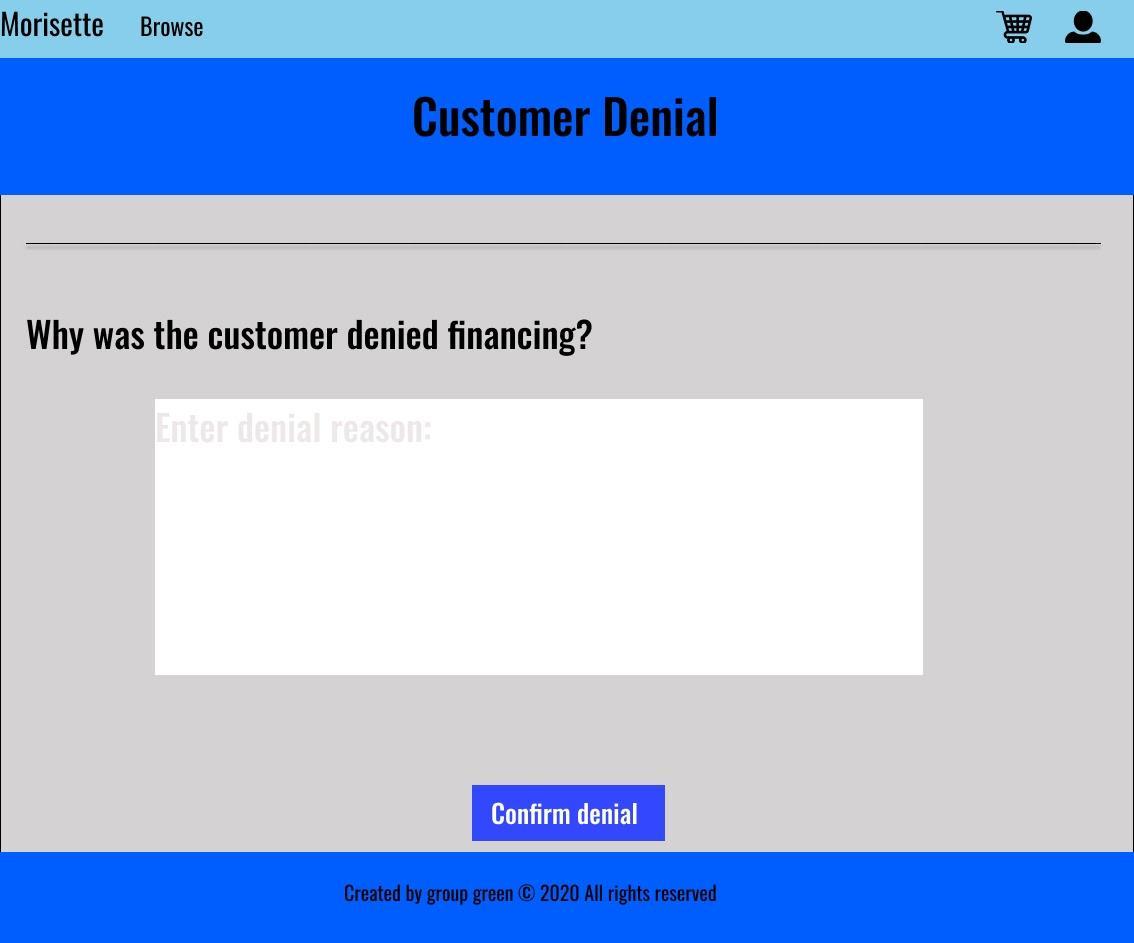
IOS\_110



**Customer Financing Denied:**

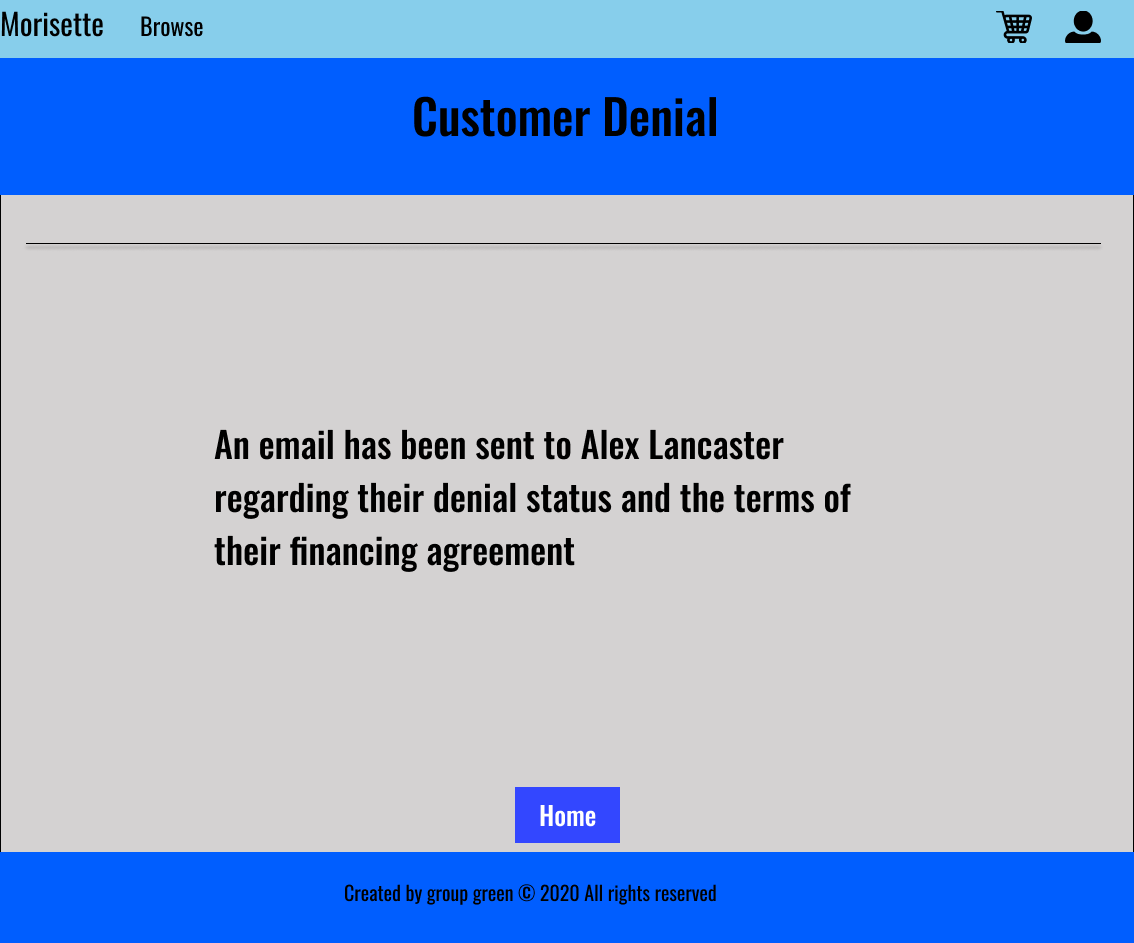
**Input:**

IOS\_111



**Output:**

IOS\_112



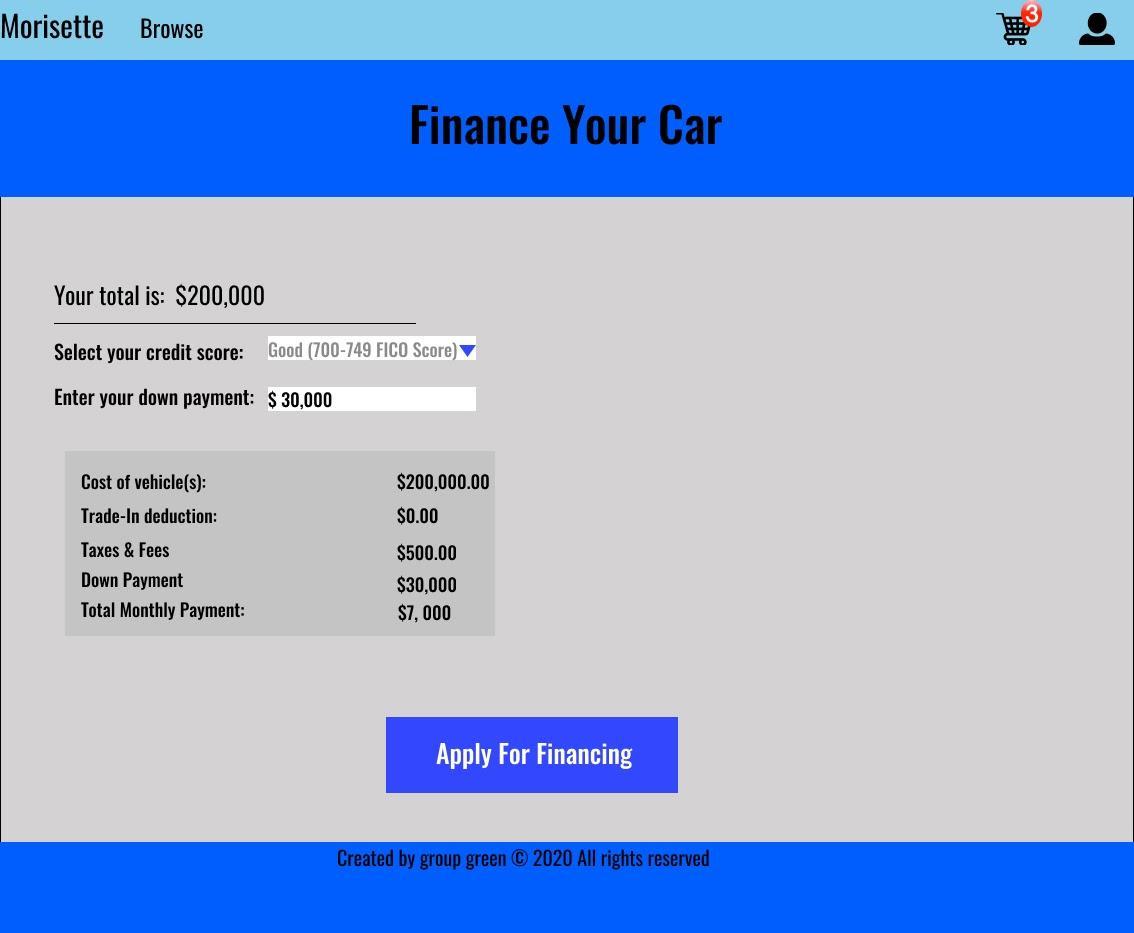
**Input/Output**

**Registered User Screens:**

**Apply for financing:**

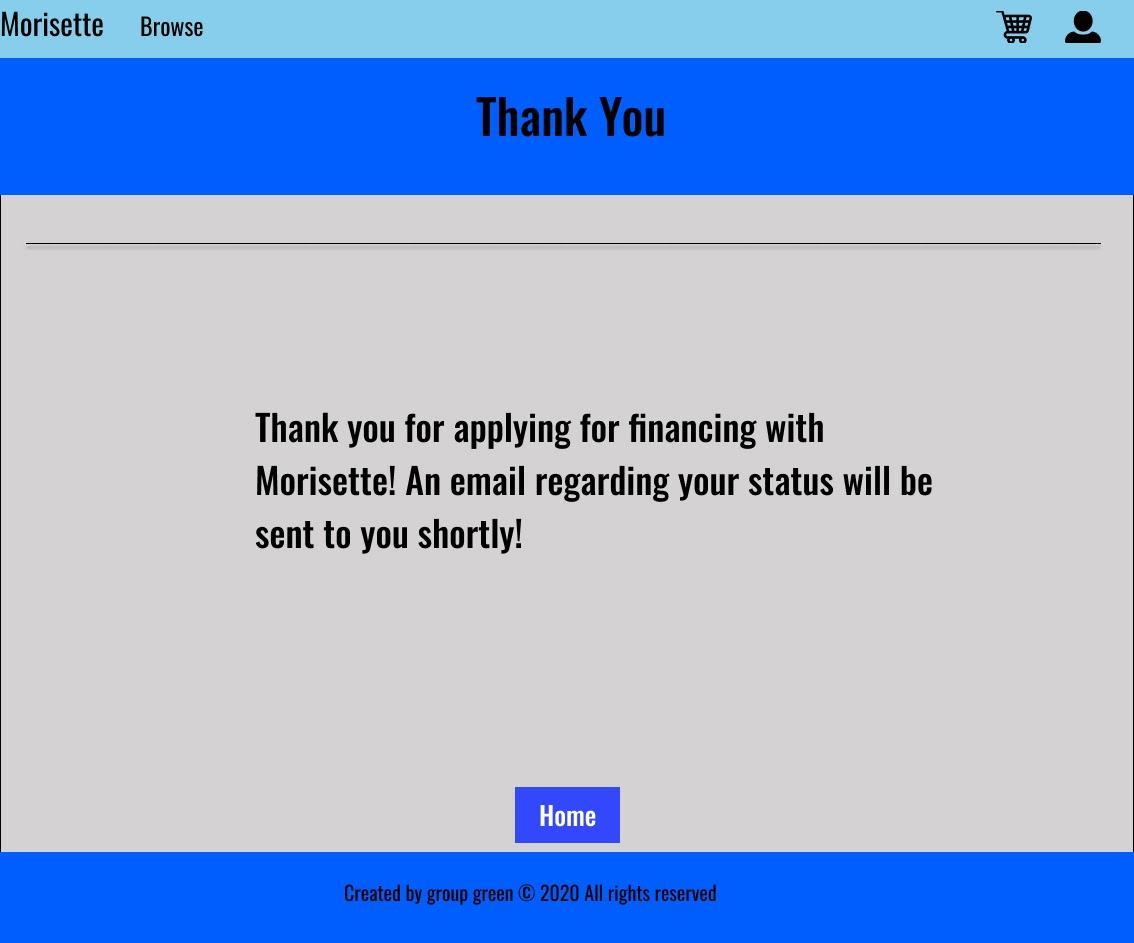
**Input:**

IOS\_200



**Output:**

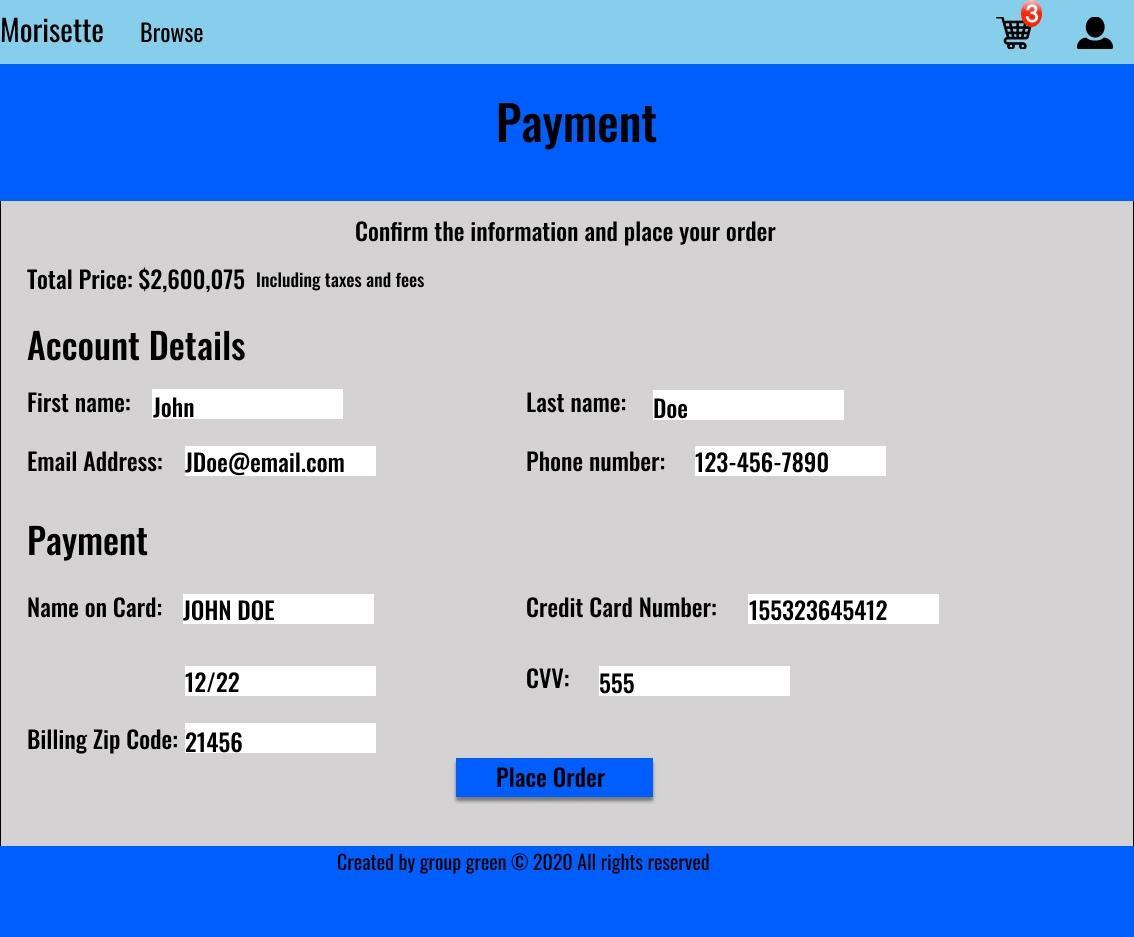
IOS\_201



**Payment:**

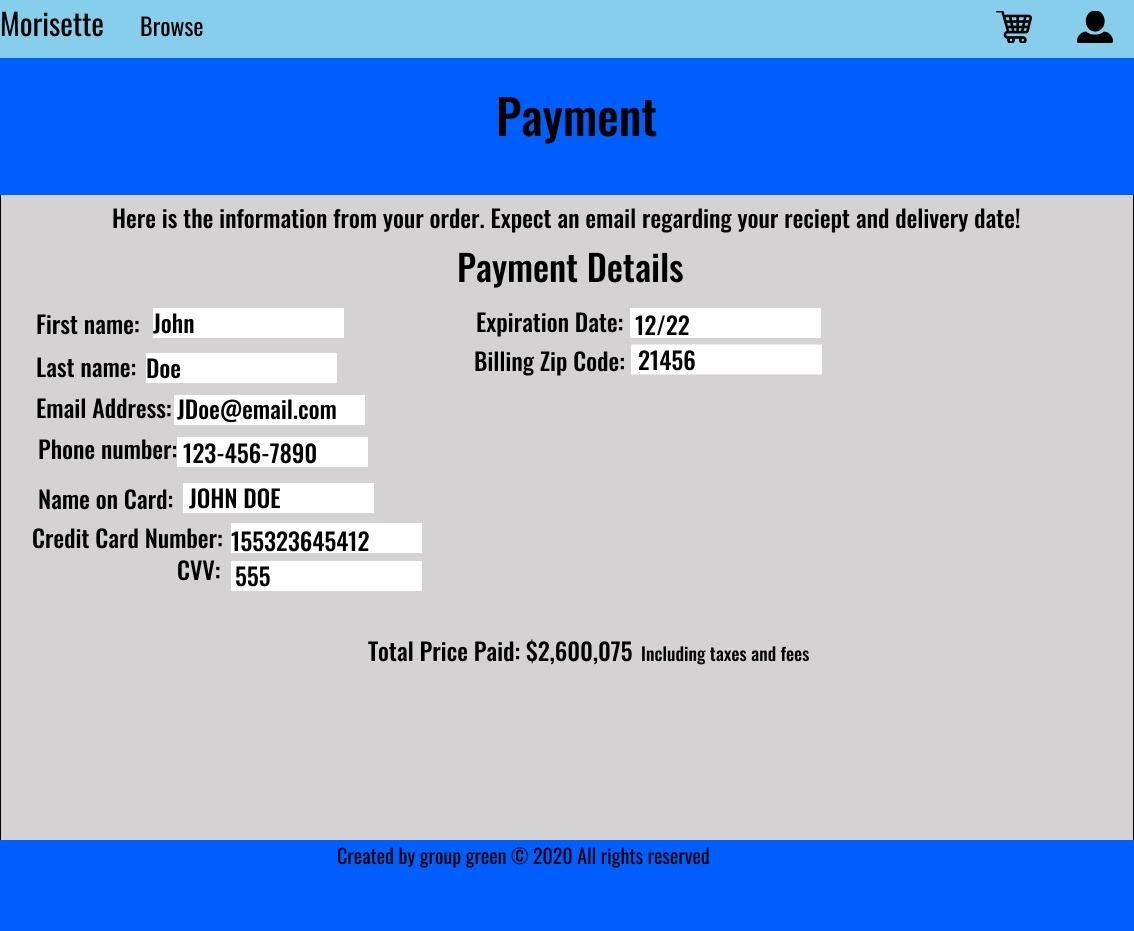
**Input:**

IOS\_202



**Output:**

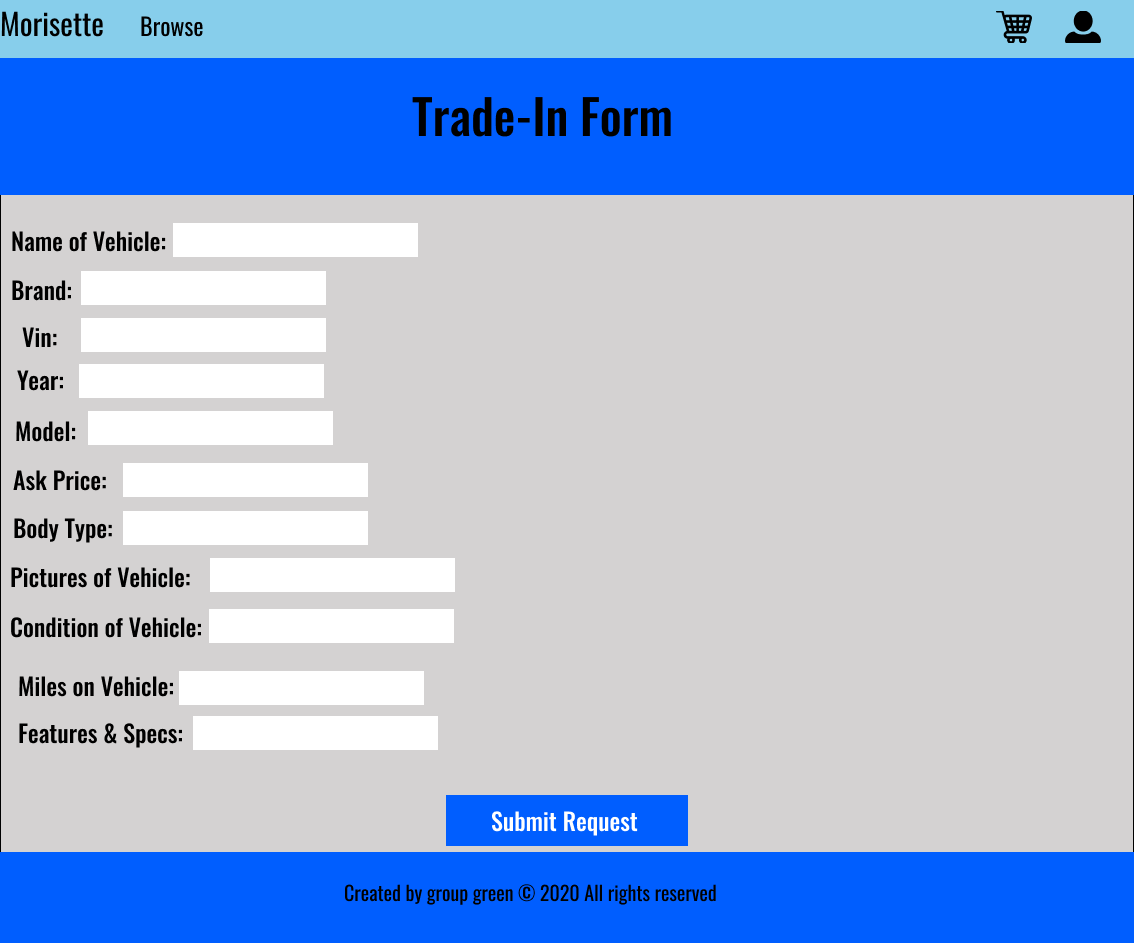
IOS\_203



**Trade-In Form:**

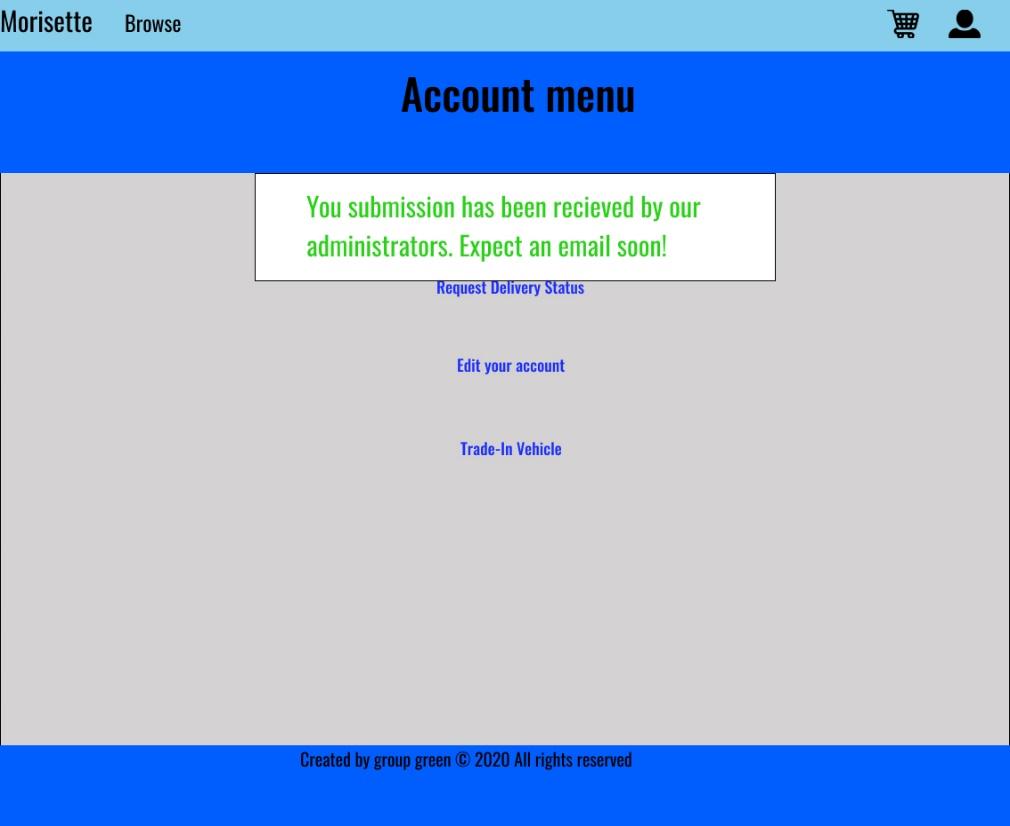
**Input:**

IOS\_204

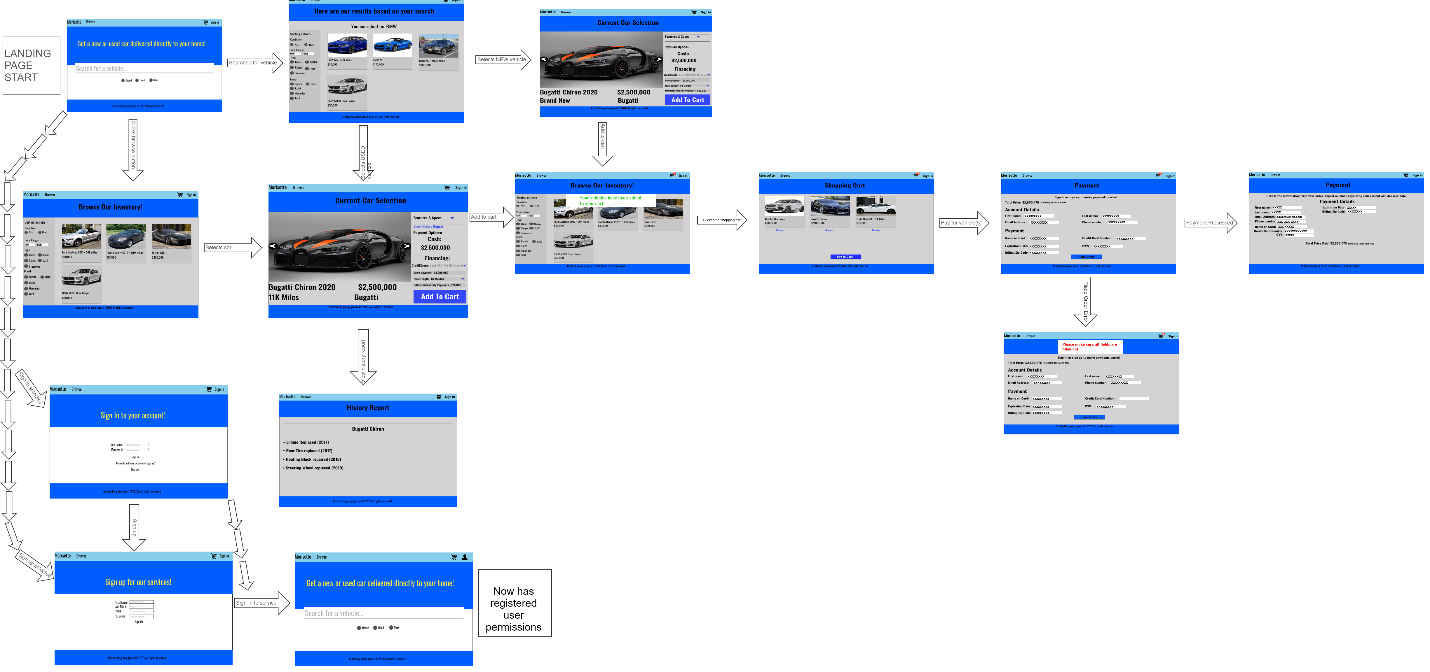


**Output:**

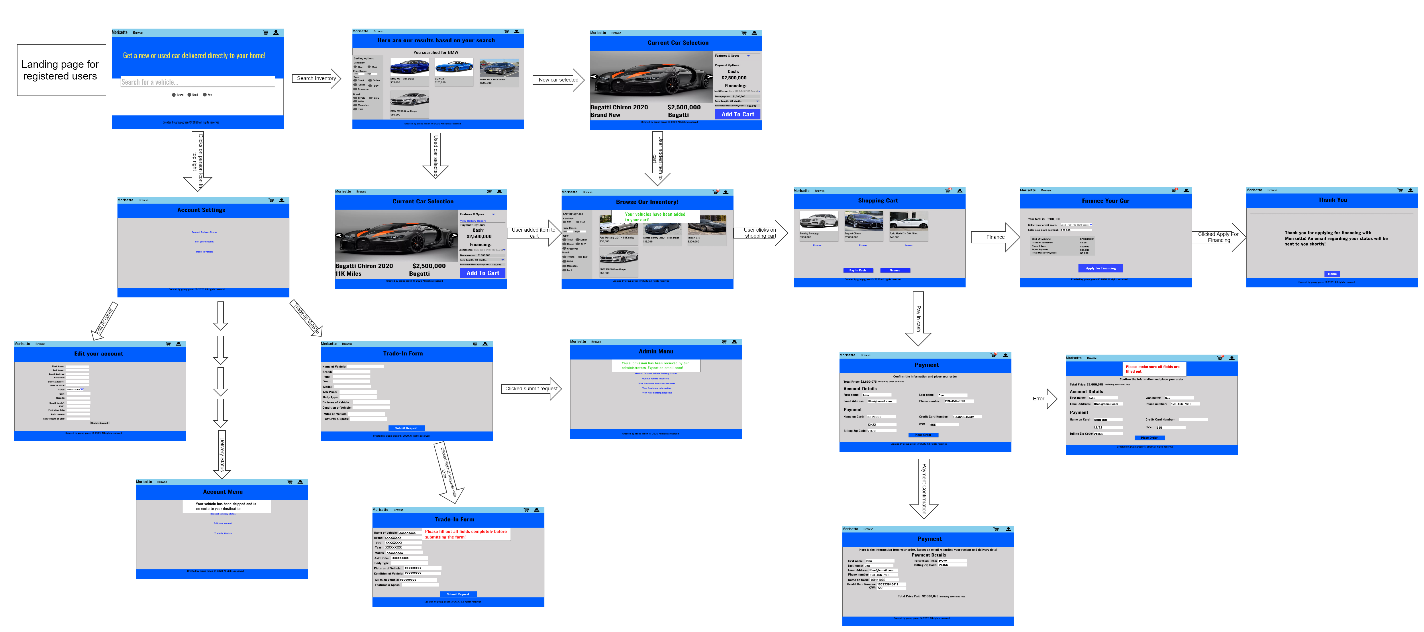
IOS\_205



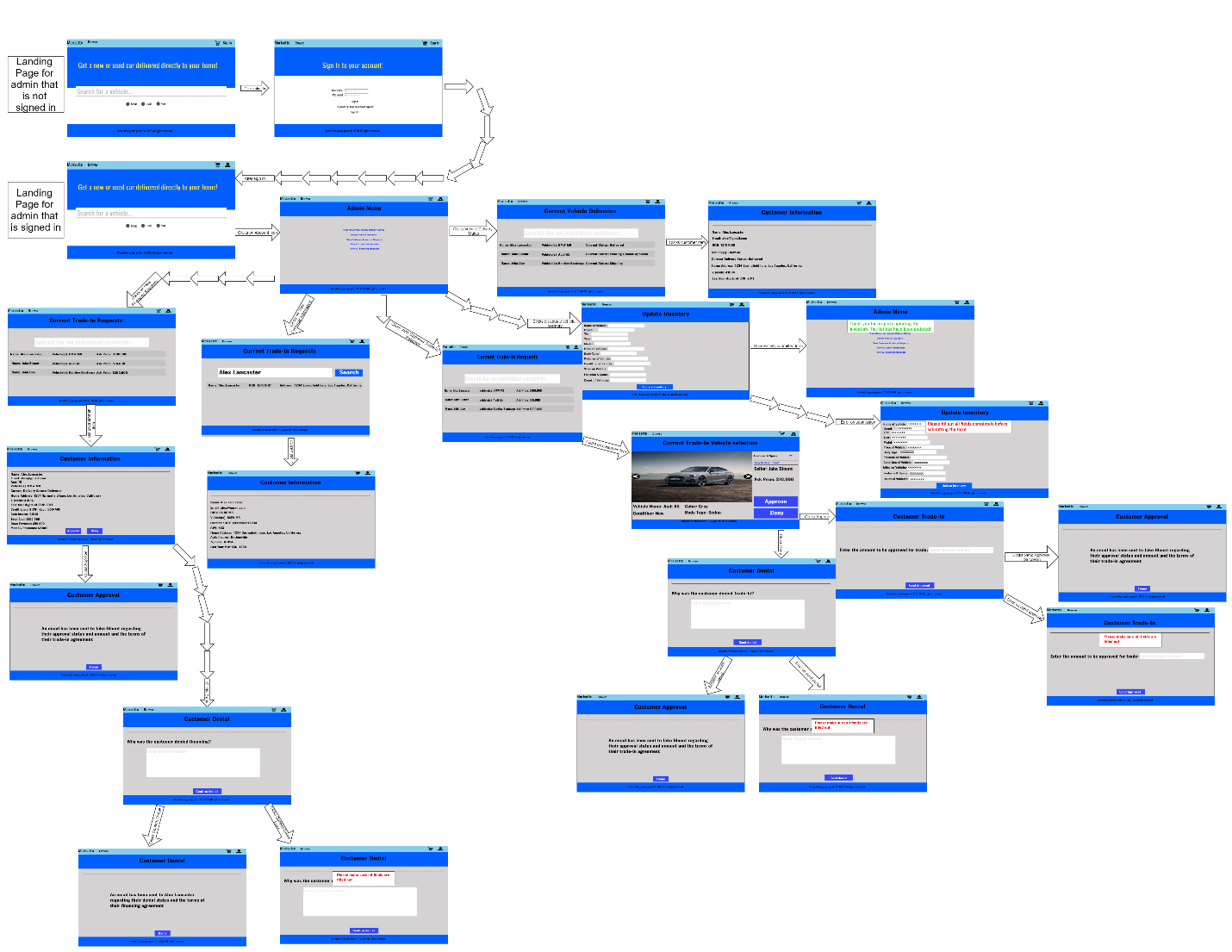
* 1. **GUI Transition Diagram**
     1. **Guest User Transition**

****

**5.2.2 Registered User Transition**

****

**5.2.3 Administrator Transition**

****

* 1. **User manual**

**UM\_001 [Register Account]**

1. Navigate to Morisette.com
2. Select “Sign In” from the navigation bar
3. Select the “Sign up” option
4. Enter in the necessary information
   1. First Name
   2. Last Name
   3. Email Address
   4. Password
5. A verification email will be sent to your email address
6. After verifying your account, you are now able to sign into your account

**UM\_002 [Login]**

1. Navigate to Morisette.com
2. Select “Sign In” from the navigation bar
3. Enter your login credentials
   1. Email
   2. Password
4. Select the “Sign In” option in the center of the screen

**UM\_003 [Edit account details]**

1. Navigate to Morisette.com
2. Select “Sign In” from the navigation bar
3. Enter your login credentials
   1. Email
   2. Password
4. Select the “Sign In” option in the center of the screen, you will be redirected to the Morisette homepage
5. Select the profile icon in the top right of the navigation bar
6. Select “Edit your Account”
7. This page will allow you to change the following:
   1. First Name
   2. Last Name
   3. Street Address
   4. Date of Birth
   5. State
   6. City
   7. Zip Code
   8. Credit Card Information (CVV, Credit Card Number, Expiration Date)
   9. Insurance information
   10. SSN details
8. When completed select the “Update Account” button

**UM\_004 [Browse Vehicle Listings]**

1. Navigate to Morisette.com
2. Select the “Browse” option from the navigation bar
3. On the initial browse page, the filter options on the left will help you find the right vehicle
   1. Filter by condition
   2. Filter by price range
   3. Filter by vehicle type
   4. Filter by Manufacturer
4. To see more details of a vehicle, select that vehicle to be taken to its sale’s page

**UM\_005 [Purchase a Vehicle] [Financing]**

1. Navigate to Morisette.com
2. Either browse or search for the vehicle you would like to purchase
3. Navigate to the sale’s page of the vehicle you would like to purchase
4. Select the “Add to Cart” option
5. Select the “Shopping Cart” icon from the navigation bar
6. Once in the Shopping Cart, select “Finance”
7. Enter the required credentials; these will determine your financing options
   1. Credit Score
   2. Down payment
8. Select “Apply for Financing”
9. An email will be sent to your email address either approving or denying you financing within 48 hours

**UM\_006 [Purchase a Vehicle] [Cash]**

1. Navigate to Morisette.com
2. Either browse or search for the vehicle you would like to purchase
3. Navigate to the sale’s page of the vehicle you would like to purchase
4. Select the “Add to Cart” option
5. Select the “Shopping Cart” icon from the navigation bar
6. Once in the Shopping Cart, select “Pay in Cash”
7. Enter the required credentials
   1. First Name
   2. Last Name
   3. Email Address
   4. Phone Number
   5. Name on the card
   6. Credit Card Number
   7. CVV
   8. Billing Address
   9. Expiration Date
8. Select “Place Order”
9. You will be redirected to a confirmation page to review your purchase and a receipt will be sent to your email address

**UM\_007 [Trade in Vehicle]**

1. Navigate to Morisette.com
2. Select “Sign In” from the navigation bar
3. Enter your login credentials
   1. Email
   2. Password
4. Select the “Sign In” option in the center of the screen, you will be redirected to the Morisette homepage
5. Select the profile icon in the top right of the navigation bar
6. Select the “Trade-In Vehicle” option
7. Enter in the required fields in the Trade-In Form
   1. Name of the Vehicle
   2. Model
   3. Manufacturer
   4. VIN
   5. Year
   6. Asking Price
   7. Body Type
   8. Condition of the Vehicle
   9. Miles on the vehicle
   10. Additional features and specs
8. Select “Upload Pictures” to upload any pictures of the vehicle you would like to include
9. Once all fields are completed, select “Submit Request”
10. Allow 3 to 7 business days for your trade-in processing after which you will be contacted by an administrator about the status of your trade-in
11. If your trade-in was approved your vehicle will be collected by a Morisette representative and your account will be credited the value of your trade-in

**UM\_008 [Search for Vehicles]**

1. Navigate to Morisette.com
2. On the main page enter your search query into the “Search” text box
   1. **Search by: Price, Color, Body Type, Make, Model**
3. Confirm your search by selecting the “Search” option