

Software Engineering Final Project Report: Car Price Estimator Web Application

Juan Cisneros

Angel Meraz Payan

Luis Perez

December 8, 2022

Abstract

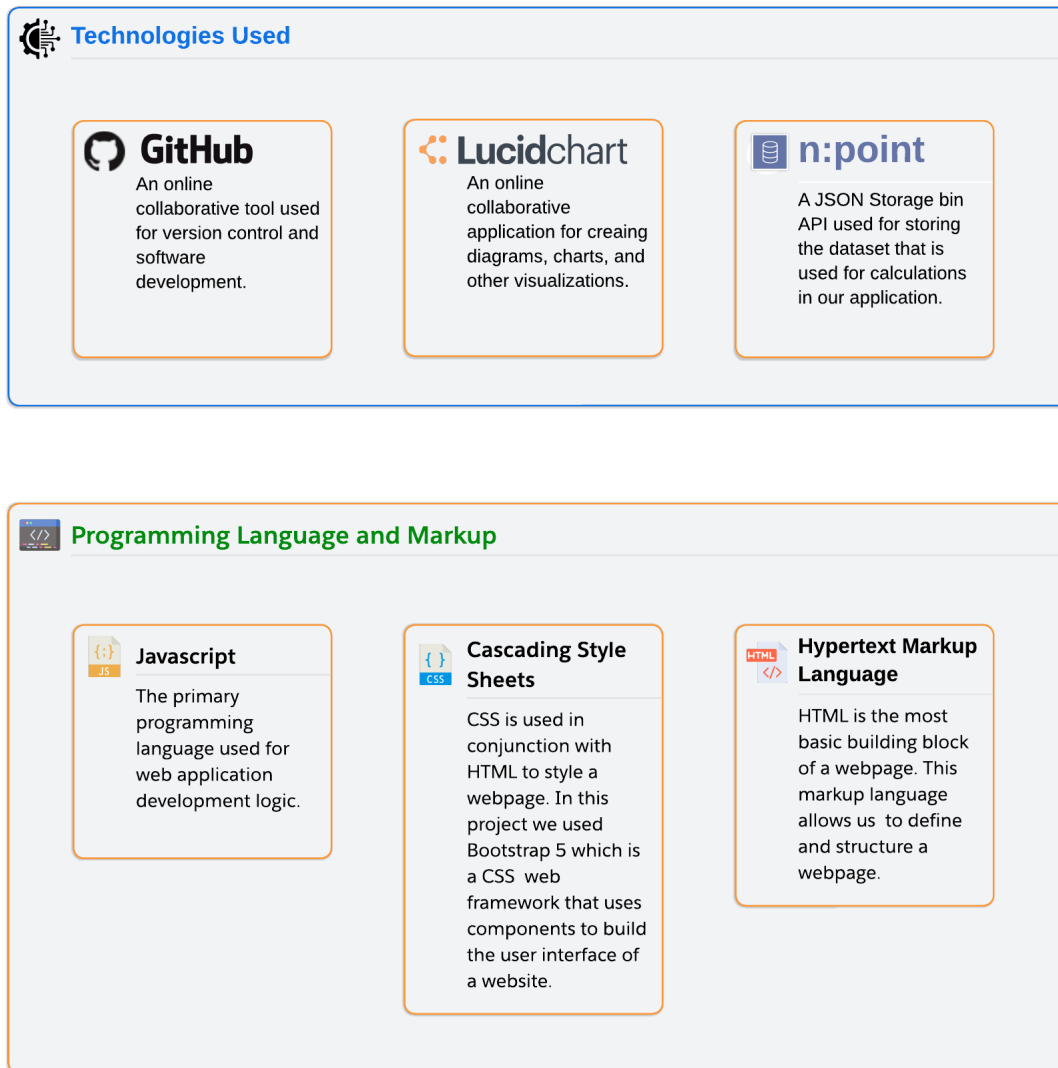
In this final project for software engineering, we utilize various technologies and use our acquired knowledge to design, implement and deploy a web application. The application is created using JavaScript, HTML, and CSS, with a JSON storage bin acting as our database. In this paper, we describe the process and work put in by the team to complete the project from the design phase to deployment.

Keywords: Software Engineering; Web Application;.

1 Introduction

Software engineering is the process of creating software systems, software engineering has a set of processes on how to develop software. We begin with an idea for a product that will serve a customer or provide certain functionality to benefit or improve a process in a company or organization. Our idea was to create an application of some sort that would estimate the prices of a vehicle based on certain criteria which we would obtain as input from a user. We also discussed making our project open-source since some of the tools and technologies we would be using are also open-source, we decided to use the MIT License for our project [1]. This application should follow a logical and smooth flow because it will work depending on provided options that we obtain from the user. In order for this website to be fully functional we must implement a user interface that is uniform and will assist the user in choosing the options or suggestions that they desire. The results shall be displayed on the web page and it will have condensed information and the different attributes that made the price of the vehicle result in whatever the outcome was.

Figure 1: Technologies, and Languages



We now discuss the technologies we used for our development process. Figure 1 showcases the technologies and tools we utilized for this project. We decided to use GitHub for development and source control, this tool was an essential piece in keeping our program well-documented and made it easier for us to make contributions to the project. GitHub also has a feature called GitHub pages that allows us to host static websites, our application could be constructed as a static website that runs on the client side. Another crucial tool is Lucidchart which is an online collaborative web application that allowed us to create diagrams and other visualizations. This tool allowed us to create many of the

diagrams used for designing our application and also accompanying the visual aids for our report and presentation. The final but also highly important technology we used was the free and open-source JSON storage bin API n:point[2]. This tool essentially acted as our data storage, this storage stores a JSON array that contains the data set that will be used for calculating prices in our application. The n:point API works in conjunction with JavaScript FETCH API [3] to retrieve the JSON array from the online hosted storage bin that contains our data. Again since our application is meant to run on the web browser the languages and markup we would be using to construct our website are the commonly used technologies of JavaScript, HTML, and CSS. For JavaScript, we would be using the core "Vanilla" JavaScript [4] version meaning without any external libraries such as JQuery or JavaScript frameworks such as React Native, instead, we used built-in tools and libraries such as the FETCH API[?]. In terms of HTML [5] and CSS [6], we decided to use Bootstrap 5 which is a free and open-source HTML and CSS framework that consists of reusable components[7]. This framework provides pre-built reusable components which makes developing the UI (user interface) much faster with properly documented code. Bootstrap 5 is also used for building responsive web pages that automatically conform all content of a web page to the size of the device accessing the web page. This is an important feature of this framework since it means we do not have to create a separate website specifically for mobile devices, allowing us to save development time.

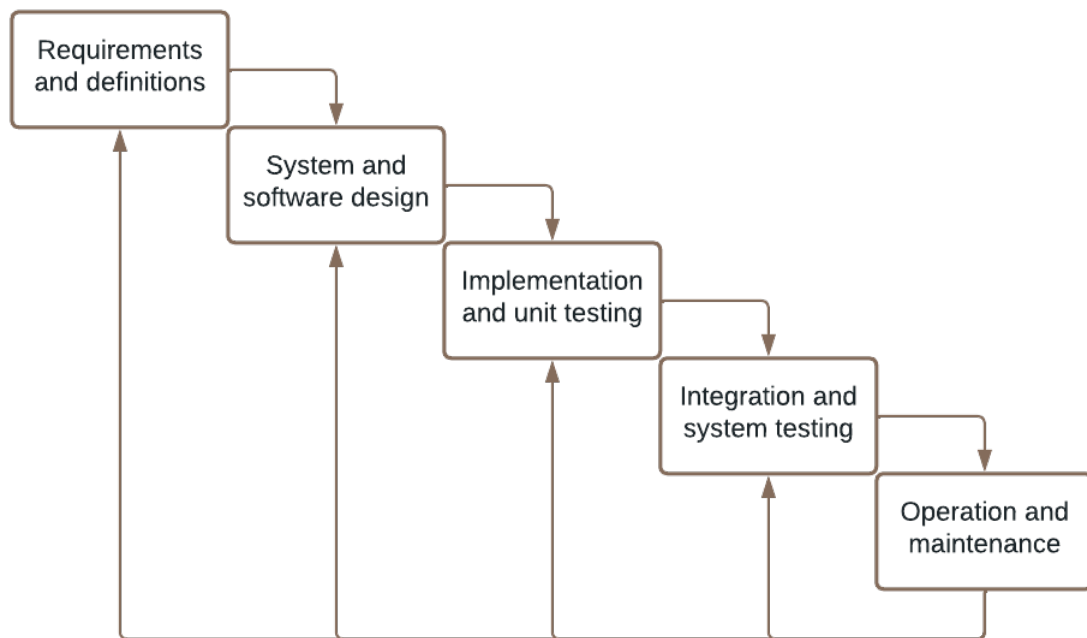
2 Methodology

2.1 Requirements and definitions and System and Software design

The first key step in the development process is deciding the development methodology to use for developing our web application. As a group, we discussed the various approaches to the development process of our application, and what stood out most was the waterfall methodology [8]. The waterfall method is comprised of the following steps:

first, we create system requirements and documentation, then we design the system (this includes creating flowcharts and diagrams, and brainstorming technologies to use), next we implement the system based on the requirements and design patterns, finally we test the application and perform reevaluations to ensure the product is ready to be released, finally we deliver the finished application (this step means we deploy the application so that it is accessible by the user), and finally we maintain the project after delivery to fix errors or to add extra functionality. As we can see in Figure 2 we start with system requirements and definitions. We note that our system requirements were updated twice so we will only discuss our most recent updated requirements.

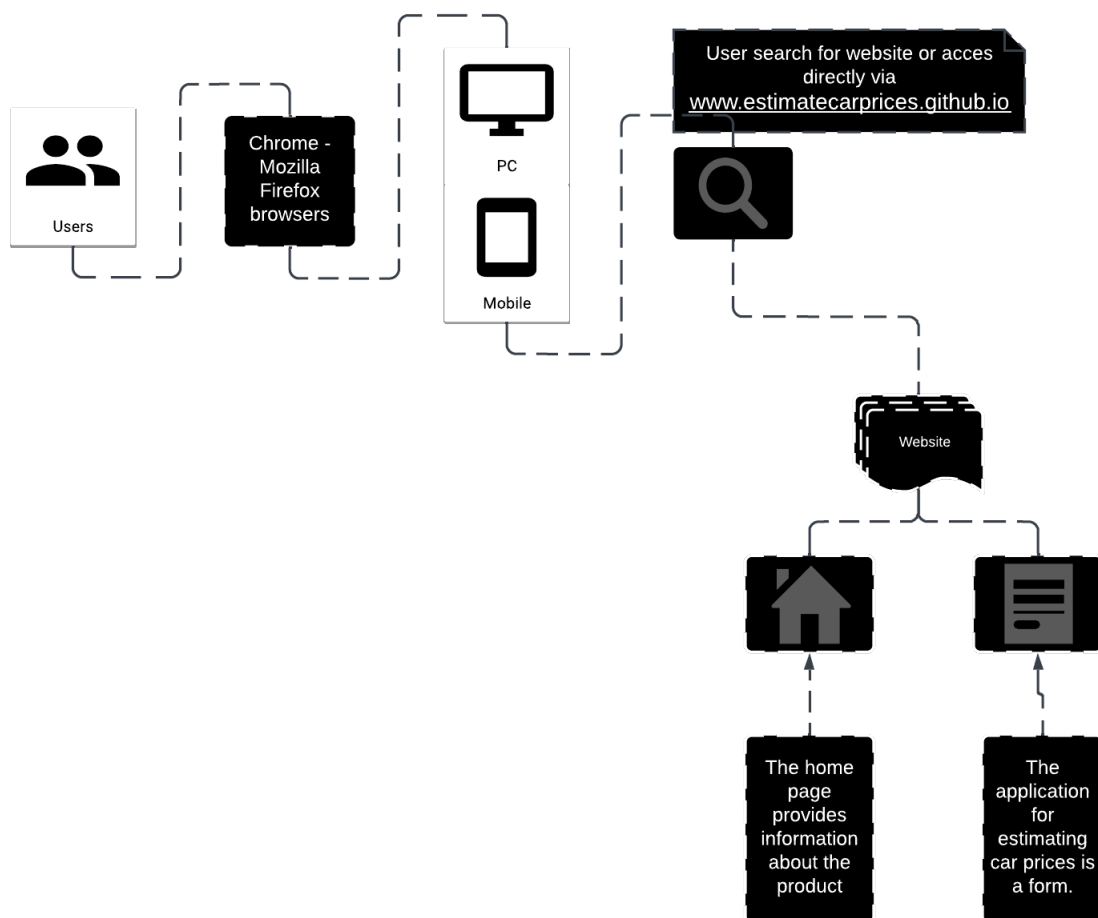
Figure 2: Waterfall methodology for software development



Our system requirements were created by us but will simulate a client doing business with our team, in essence, the client wants us to develop a web application based on these requirements (Reference Appendix A for these requirements). The first section of requirements basically states what we are building, first, it must be a website, this requires the use of a web browser, and as previously stated this will work on any desktop and mobile

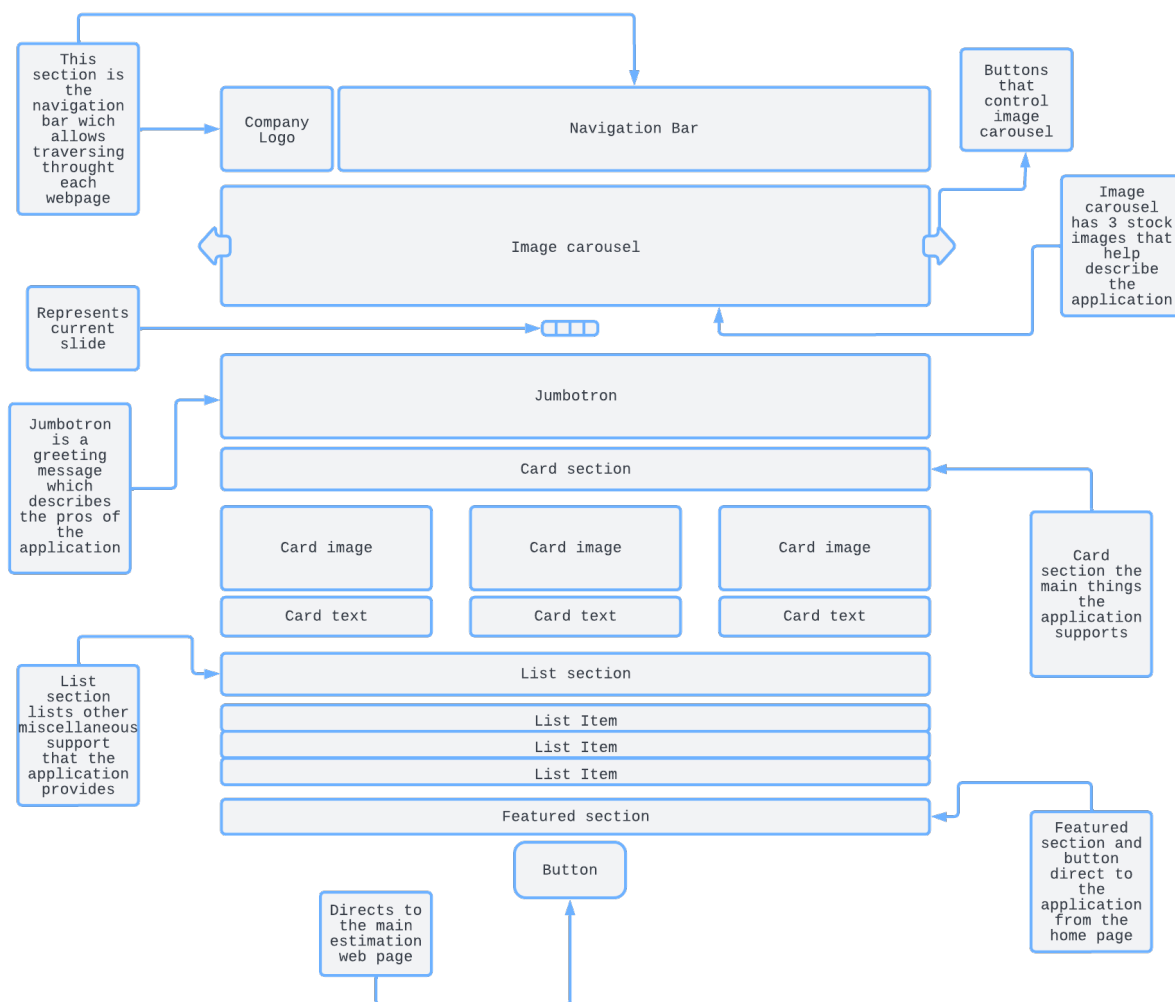
device of any operating system that supports the internet connection and web browsing. This section also states some behavior that the application should have such as providing navigation and how the user experience should act. The next section describes how the home page should function the main purpose of this web page is to inform the user about the application. The next section goes more into detail on how the application that estimates the price of a vehicle should function. After we discussed these requirements we were ready to begin system and software design, in this stage we created flow charts and diagrams of how the web application would be implemented. Figure 3 visualizes the description of the first set of requirements for how the application will be accessed by a user.

Figure 3: Accessing Application Diagram



As described by Figure 3 the user will access a web browser using either a mobile device or a Laptop or a Desktop PC, the user will then directly access the website using the URL shown or via a search engine query. Accessing the website will start out with the user seeing the home page and at any point, the user can access the price estimating application via the navigation bar. The next figure describes the structure of what the home web page or the landing page will look like.

Figure 4: Homepage or Landing Page Structure



As we can see from Figure 4 the structure of the homepage follows the provided requirements that it must contain a navigation bar, followed by an image carousel then a large section containing information about what is supported by the application. It must also

end with a section that contains a button that directs to the estimation web page. It is a convention for the navigation bar to be at the top of a web page since the browser upon loading a web page starts from the very top, so this will allow fast navigation through the website. Again following the principle that the contents of a web page are loaded from the top to bottom we want to catch the user's eye by providing visuals, in this case, we added stock images in a carousel that briefly introduces the user to the application. The jumbotron section provides an actual welcome message to the user, while the card section provides visual imagery of the support provided by the application and the list section reinforces other support provided. The button in the featured section is placed at the bottom of the page to draw in the user to click on the link with leads to the estimation web page. Each section is placed in that specific order to make the web page easy to follow and continuous. A note on the web page structure, for each section such as navigation, image carousel, jumbotron, and so on, we utilized Bootstrap 5 components to build the web page.

Figure 5: Estimation Web-page Architecture Flow Chart

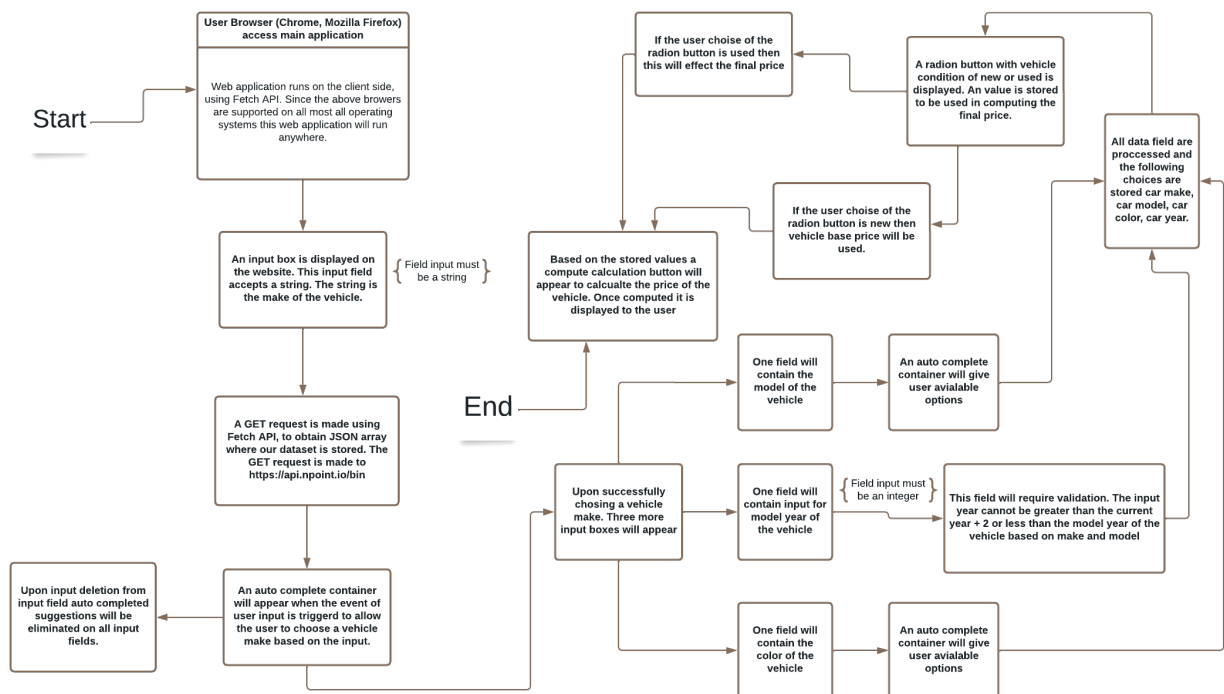


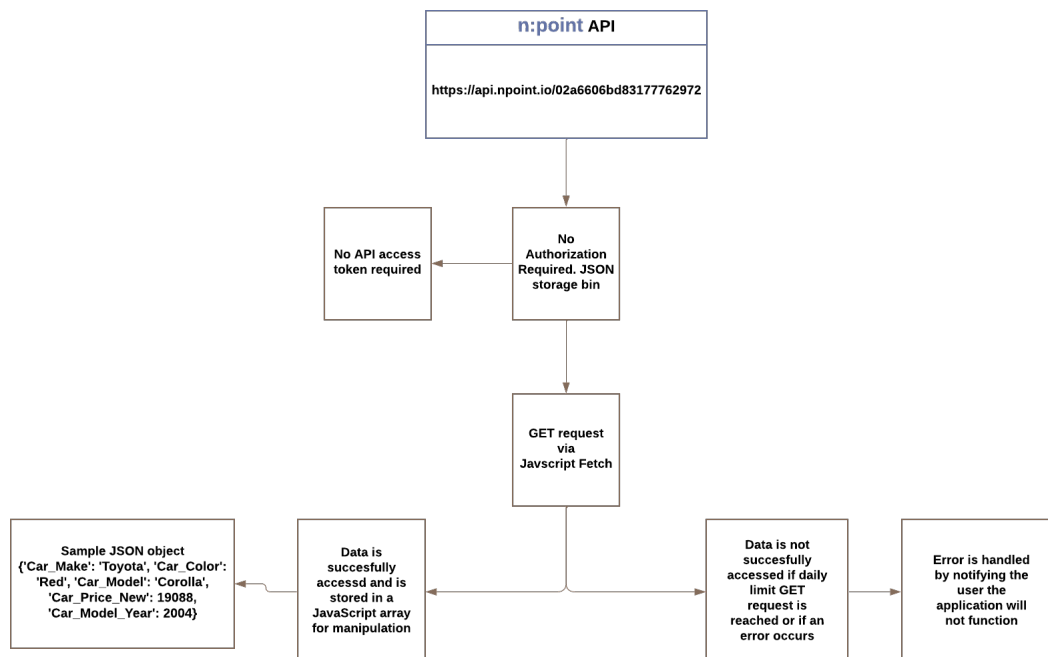
Figure 5 describes the process flow of the architecture for the estimation web page, while Figure 6 is the flow sequence of what the actual web page will look like and how it will function. The application begins by prompting the user with a form that receives input. This form takes in a string input and makes a GET request using FETCH API to the JSON storage bin to retrieve the data. Then using JavaScript built-in libraries we will match the user input with what is stored in the JSON array data that was fetched to display options, these options are the make of a vehicle more specifically the car make the user is searching for. In essence, this feature is similar to a search menu that provides a user with auto-completed suggestions, so that way there is no need to handle incorrect or irrelevant input by the user in common string input forms. Each option or suggestion based on the input is a clickable button that will store the value of the selected option or suggestion, after a user clicks a certain button all other options or suggestions will be erased until the user decides to change the input on the form. This basic principle is applied to the rest of the forms that take in the input of the vehicle model, and color. For reference on how the process works see Figure 6. The section that takes in the year of a vehicle is also a form much like the first, in this case, this form accepts an integer, and we must handle incorrect or irreverent input. The radio button seen in Figure 6 is a type of button that has only one of two states, new or used, this button gets toggled by the user and is used for calculating the estimated price of a vehicle. We note that our design was made to follow procedural programming paradigm along with JavaScript's event-driven programming. This is because after the user triggers one event another event will be set in motion, and there is a logical sequence or procedure for displaying the application to the user. So overall our architecture design was made to have this logical sequence so that a user finds the application intuitive and easy to use.

Figure 6: Estimation Web-page Mock-up Flow Chart



Below, Figure 7 gives an overview of how the n:point API works in conjunction with JavaScript's FETCH API.

Figure 7: n:point API diagram



To retrieve the data we first use JavaScript FETCH API to perform a GET request to n:point API where our JSON storage bin is located, since this is a storage bin we do not

require an API token. API tokens are commonly used to authenticate users, to ensure that a user making a request actually has access to the requested data which is only offered to paying users [9], in this instance, our data is publicly available. However, there are some restrictions given by n:point there can be no more than 100 requests/min per IP address or 600 requests/min per JSON bin [2]. Because of this case, we must ensure that if we have reached these limitations we must handle any issues or errors derived from these limitations by notifying users that our service is down, it is highly improbable that these limits will ever be surpassed. In the case that we successfully retrieve the JSON array of objects then our application can proceed. As we can see from Figure 7 an example of one of the JSON objects that is retrieved contains the make, color, model, base price, and model year. This is an essential process that allows us to calculate a vehicle's estimated price based on user input.

2.2 Implementation and unit testing

As we previously mentioned in our introduction we would be using JavaScript, CSS, and HTML to build our application, we discuss our implementation and testing process. We start off with creating the first web page which is the home page, for reference on the HTML code written for this web page see Appendix B. The home page HTML follows the basic HTML structure that every web page must have, it starts with the type of document in this case html, this is followed by a head tag that contains various metadata including the title of the web page, the favicon which is the website icon displayed by the browser, the file link to the style sheets and the meta tag that allows the web page to be responsive (mobile, desktop ready). In the body tag, we follow the structure proposed in figure 4 using Bootstrap 5 components. At the very bottom is where we provide the source of the JavaScript files we will be using, these files are delivered by a CDN and are necessary since they are a part of the Bootstrap framework. Appendix C contains the HTML for the estimation page. Again it follows the same structure as the home page until we reach

the body tag. In the body tag, there is a form tag that is the main container where the user will input their choices. Below each auto-complete supported input tag there is a div tag that is used to display the auto-completed suggestions. Appendix D consists of the style that we added to both web pages, these styles are plainly for visuals used in the UI and are not necessarily important to discuss since they do not contribute to the overall functionality of the web page. Appendix E is comprised of all our client-side logic that will be executed within the user's web browser. This JavaScript program consists of adding the auto-complete functionality, selecting elements from the HTML file for DOM manipulation, taking in user input for manipulation, retrieving the JSON array using FETCH API, and using all the input fields to calculate the estimated price of a vehicle.

Figure 8: Algorithm diagram

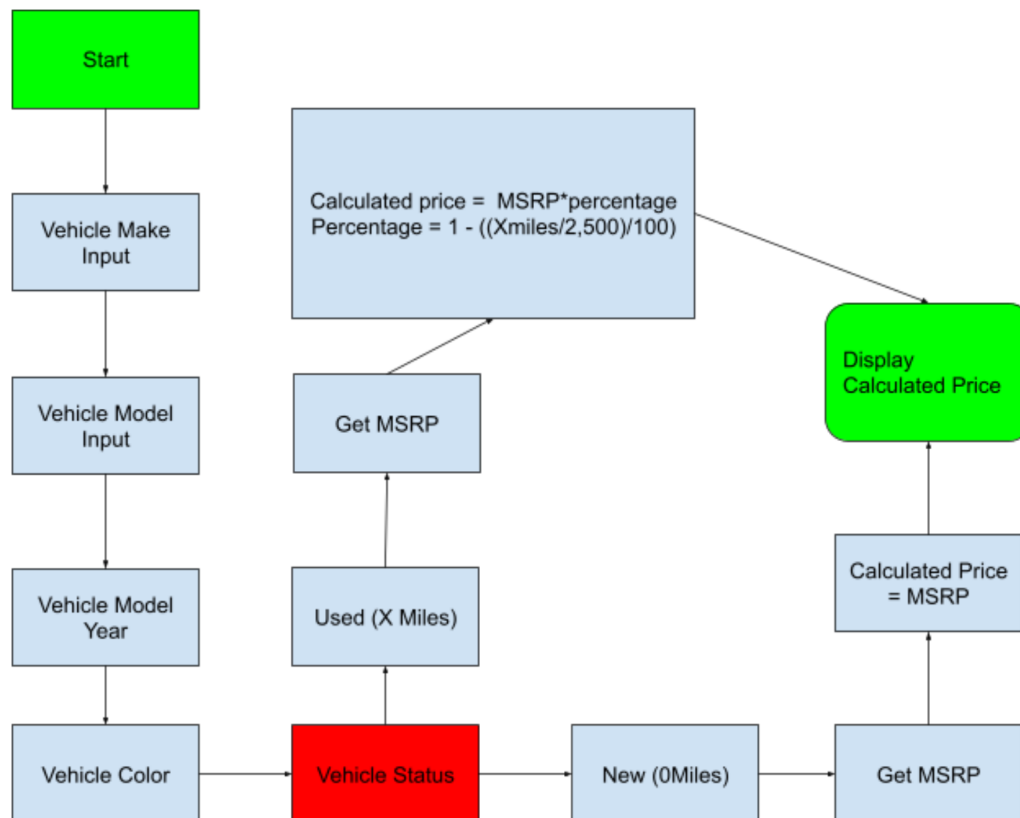


Figure 8 describes the algorithm for computing the final price based on all input from


the user.

3 Discussions and Conclusions

3.1 Integration and System testing

As we have previously mentioned we decided to host our website using GitHub web pages[10], this was our integration and deployment. GitHub pages are used for hosting a static website, since our website does not require a server we can use this tool for deployment, we can view our website using the following URL <https://lp2399.github.io/estimatecarprices.github.io/>. We now discuss our testing methods to ensure the application behaved properly. We used two cars to test out the vehicle price estimation. The first vehicle we estimated was a Red Ford Mustang from 2020, one was a new vehicle with 0 miles on the dash and the other was also a Red Ford Mustang from 2020 but this vehicle had 99,000 miles on the dash. Without calculations working we had the used car at an estimated price of \$18,119 whereas the new one would be \$29,998. The second vehicle we tested was a 2015 Green Dodge Challenger the new one with 0 miles was at an estimated price of \$22,839 whereas the used one with 69,000 miles on the dash was estimated to be worth \$16,535. Below is a set of figures from 9-12 from our unit testing the application behaved as expected.

Figure 9: New Mustang



Estimate Vehicle Prices Now

Follow the step by step procces for your free vehicle pricing estimate

Our algorithm is the perfect tool for auto-dealers and customers in search of vehicles

Enter Vehicle Make

Ford

Enter Vehicle Model

Mustang

Enter Vehicle Color

Red

Select Vehicle Mileage


Select model year 1990-2023

☒ New ☐ Used

Calculate Price

29998

Figure 10: New Challenger



Estimate Vehicle Prices Now

Follow the step by step proces for your free vehicle pricing estimate

Our algorithm is the perfect tool for auto-dealers and customers in search of vehicles

Enter Vehicle Make

Dodge

Enter Vehicle Model

Challenger

Enter Vehicle Color

Green

Select Vehicle Mileage 0


Select model year 1990-2023

☐ New ☒ Used

Calculate Price

22839

Figure 11: Used Mustang



Estimate Vehicle Prices Now

Follow the step by step procces for your free vehicle pricing estimate

Our algorithm is the perfect tool for auto-dealers and customers in search of vehicles

Enter Vehicle Make

Ford

Enter Vehicle Model

Mustang

Enter Vehicle Color

Red

Select Vehicle Mileage 99000


Select model year 1990-2023

☒ New ☐ Used

Calculate Price

18119

Figure 12: Used Challenger



Estimate Vehicle Prices Now

Follow the step by step procces for your free vehicle pricing estimate

Our algorithm is the perfect tool for auto-dealers and customers in search of vehicles

Enter Vehicle Make

Dodge

Enter Vehicle Model

Challenger

Enter Vehicle Color

Green

Select Vehicle Mileage 69000

Select model year 1990-2023

☒ New ☐ Used

Calculate Price

16535

3.2 Operation and Maintance

Figure 13: Gantt Chart



We now discuss our work and contributions, Figure 13 demonstrates a Gantt chart, which shows the contributions each team member made. As we can see from this chart most of our work was put toward the design and implementation phases. Week 1 is where we first began the project designs and requirements, week 14 is the final week where everything was tested and completed. Each team member contributed our meeting days were Monday, and Wednesday from 12:15 PM to 1:00 PM, including the time provided during class, and Saturday from 12:00 PM to 1:00 PM. Most of our work was again done using GitHub version control, we also used messaging apps to keep ourselves on track. In the operation and maintenance section, we updated various functionality such as adding a hash table containing color keys, and prices as values. As far as maintenance goes we plan on making some additions that would have more support such as displaying images

of the vehicles based on the make and model with different colors. Overall this final project allowed us to utilize various technologies and our acquired knowledge to design, implement and deploy a web application. The technologies we used of JavaScript, HTML, and CSS, with a JSON storage bin acting as our database, allowed us to have a responsive website that conformed with the requirements and is accessible from mobile and desktop devices.

References

- [1] “The mit license — open source initiative.” The MIT License.
- [2] JSON storage bins that won’t break your app.
- [3] “Using the fetch api - web apis: Mdn.” Using the fetch API - web apis: MDN.
- [4] What is javascript? - learn web development: MDN.
- [5] Hypertext Markup Language: MDN.
- [6] What is CSS? - learn web development: MDN.
- [7] J. Thornton and M. Otto. Introduction.
- [8] A. Waseem, Nov 2022. Waterfall methodology – ultimate guide.
- [9] API tokens and Authentication.
- [10] Websites for you and your projects.

4 Appendices

A System Requirements

Overall System Requirements

1. The system shall be comprised of two interactive web pages that should work on Chrome and Mozilla Firefox browsers for desktop and mobile.
2. The system shall provide a home page as one of the two interactive web pages that contain an overview of the application's functionality.
4. The system shall provide a link from within the home page to the car prices estimation application page.
5. The system shall provide a link from the car prices estimation application page to the home page.
6. The system shall be hosted using GitHub pages with the domain name `www.estimatecarprices.github.io`.
7. After the user deletes from an input field all input fields and displayed data below must be removed.
8. The user must be informed of any incorrect or unrelated input on any given input field.
9. The application should be minimalist and intuitive to use.

System Functional Requirements of the Home page.

1. The system shall upon loading display stock images that include captions describing the application and there should be a visible navigation bar.

2. The system shall describe the car makes that are supported by the application.
3. The system shall provide a link at the bottom of the website that directs to the car price estimate application.

System Functional Requirements of the car prices estimation application page.

1. The system shall upon loading provide an input box to enter vehicle make.
2. The system shall provide the user with different vehicle models depending on the vehicle make.
3. The system shall provide input fields for vehicle model year, available vehicle colors, and vehicle condition (New, Used).
4. The system should display a menu of options regarding vehicle mileage if the user selects a used vehicle.
5. The system shall then retrieve, the base price of a vehicle based on the model, year, and make. If the user selected a used vehicle the system shall accommodate the price which subtracts from the vehicle base price.
6. The system shall provide a button to calculate the final price of the vehicle based on the base price plus the color, vehicle condition, and year.
7. The system shall display the calculated price of the vehicle based on the user's input upon the user's click of a button.

B Home page HTML code

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8" />
5          <meta name="viewport" content="width=device-width, initial-scale=1
            , shrink-to-fit=no" />
6          <meta name="Luis Perez" content="" />
7          <link type="image/png" sizes="16x16" rel="icon" href="./
            icons8-car-sale-16.png" />
8          <meta http-equiv="X-UA-Compatible" content="ie=edge" />
9          <title>Home</title>
10         <link rel="stylesheet" href="style.css" />
11         <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/
            bootstrap.min.css" rel="stylesheet" integrity="
            sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65
            " crossorigin="anonymous" />
12     </head>
13     <body>
14         <nav class="site-header sticky-top py-1">
15             <div class="container d-flex flex-column flex-md-row
                justify-content-between" id="navbar">
16                 <a class="py-2" href="index.html">
17                     
18                 </a>
19                 <a class="py-2 d-none d-md-inline-block" href="
                    Estimate.html">Estimate</a>
20             </div>
21         </nav>
22
23         <div class="container-fluid">
24             <div class="carousel slide" data-bs-ride="carousel">
25                 <div class="carousel-indicators">


```

```

26      <button type="button" data-bs-target="#myCarousel"
        data-bs-slide-to="0" class="active" aria-label="
          Slide 1" aria-current="true"></button>
27      <button type="button" data-bs-target="#myCarousel"
        data-bs-slide-to="1" aria-label="Slide 2" class="">
        </button>
28      <button type="button" data-bs-target="#myCarousel"
        data-bs-slide-to="2" aria-label="Slide 3" class="">
        </button>
29    </div>
30    <div class="carousel-inner">
31      <div class="carousel-item active">
32        
37      <div class="container">
38        <div class="carousel-caption text-start">
39          <h1>Find your dream car</h1>
40          <p>Find a price estimate of your dream car
            based on your budject or preferences
            </p>
41        </div>
42      </div>
43    </div>
44    <div class="carousel-item">
45      <img class="d-block w-100 image img-fluid

```

```


46     <div class="container">
47         <div class="carousel-caption">
48             <h1>Our pricing estimates are used by
49                 dealers across the country</h1>
50             <p>Each pricing estimate is guranteed to
51                 be accurate</p>
52         </div>
53     </div>
54     <div class="carousel-item">
55         
60         <div class="container">
61             <div class="carousel-caption text-end">
62                 <h1>Our app works on any device so you can
63                     stay at home</h1>
64                 <p>No need to visit a dealership simply
65                     use our app</p>
66             </div>
67         </div>
68     </div>
69     <div class="container">
70         <div class="text-center">
71             <button class="btn btn-primary">Get the app</button>
72         </div>
73     </div>
74     <div class="text-center">
75         <button class="carousel-control-prev" type="button"
76             data-bs-target="#myCarousel" data-bs-slide="prev">
77             <span class="carousel-control-prev-icon" aria-hidden="
78                 true"></span>

```

```

65         <span class="visually-hidden">Previous</span>
66     </button>
67     <button class="carousel-control-next" type="button"
        data-bs-target="#myCarousel" data-bs-slide="next">
68         <span class="carousel-control-next-icon" aria-hidden="
            true"></span>
69         <span class="visually-hidden">Next</span>
70     </button>
71 </div>
72 </div>
73 <div class="container-fluid">
74     <div class="jumbotron">
75         <h1 class="display-4">Estimate Vehicle Prices</h1>
76         <p class="lead">We provide an estimate of a vehcile's
            price based on make, model, year, color, and condition
        </p>
77         <hr class="my-4" />
78         <p>Our algorithm is the perfect tool for auto-dealers and
            customers in search of vehicles</p>
79     </div>
80 </div>
81
82 <div class="container-fluid">
83     <div class="card text-center">
84         <div class="card-body">
85             <h5 class="card-title">We provide pricing estimate for
                the big three</h5>
86             <p class="card-text">Ford, GM, and Chrystler are the
                top USA vehicle manufactures</p>
87         </div>
88     </div>
89     <div class="card-group">
90         <div class="card">

```



```

91         
92     <div class="card-body">
93         <h5 class="card-title">Find pricing estimates for
           various Ford models</h5>
94     </div>
95 </div>
96 <div class="card">
97     
98     <div class="card-body">
99         <h5 class="card-title">Find pricing estimates for
           various General Motors models</h5>
100    </div>
101 </div>
102 <div class="card">
103     
104     <div class="card-body">
105         <h5 class="card-title">Find pricing estimates for
           various Chrystler models</h5>
106     </div>
107 </div>
108 </div>
109 <div class="card text-center">
110     <div class="card-body">

```

```

111         <h5 class="card-title">We offer pricing estimate for
           many other vehicle makes</h5>
112         <p class="card-text">Each make has many models to
           choose from</p>
113     </div>
114 </div>
115 </div>
116 <div class="container-fluid">
117     <ul class="list-group">
118         <li class="list-group-item">Hundai</li>
119         <li class="list-group-item">Toyota</li>
120         <li class="list-group-item">Honda</li>
121         <li class="list-group-item">Chevrolet</li>
122         <li class="list-group-item">Tesla</li>
123         <li class="list-group-item">Many more</li>
124     </ul>
125 </div>
126 <div class="container-fluid">
127     <div>
128         <div class="card text-center">
129             <div class="card-header">
130                 Featured
131             </div>
132             <div class="card-body">
133                 <h5 class="card-title">Use our algorithm to
                   estimate the cost of a vehicle</h5>
134                 <p class="card-text">Click the link below to use
                   our app</p>
135                 <a href="Estimate.html" class="btn btn-primary">
                   Estimate Now</a>
136             </div>
137         </div>
138     </div>

```

```

139         </div>
140     </body>
141     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/
        jquery.min.js"></script>
142     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/
        bootstrap.bundle.min.js"></script>
143 </html>

```

C Estimation page HTML code

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="UTF-8" />
5         <meta name="viewport" content="width=device-width, initial-scale=1
        , maximum-scale=1, user-scalable=no" />
6         <meta name="description" content="" />
7         <meta name="Luis Perez" content="" />
8         <link type="image/png" sizes="16x16" rel="icon" href="./
        icons8-car-sale-16.png" />
9         <meta http-equiv="X-UA-Compatible" content="ie=edge" />
10        <title>Estimate</title>
11        <link href="https://cdnjs.cloudflare.com/ajax/libs/
        bootstrap-datepicker/1.2.0/css/datepicker.min.css" rel="
        stylesheet">
12        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/
        bootstrap.min.css" rel="stylesheet" integrity="
        sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65
        " crossorigin="anonymous" />
13        <link rel="stylesheet" href="style.css" />
14
15    </head>

```

```

16 <body id="Estimate-body">
17     <nav class="site-header sticky-top py-1">
18         <div class="container d-flex flex-column flex-md-row
19             justify-content-between">
20             <a class="py-2" href="index.html"></a> <a class="py-2 d-none
23                 d-md-inline-block" href="index.html">Home</a>
24         </div>
25     </nav>
26     <div class="container-fluid">
27         <div class="jumbotron">
28             <h1 class="display-4">Estimate Vehicle Prices Now</h1>
29             <p class="lead">Follow the step by step procces for your
30                 free vehicle pricing estimate</p>
31             <hr class="my-4" />
32             <p>Our algorithm is the perfect tool for auto-dealers and
33                 customers in search of vehicles</p>
34         </div>
35     </div>
36     <div class="container-fluid">
37         <form autocomplete="off">
38             <div class="form-group col-xs-3"><label>Enter Vehicle
39                 Make</label> <input type="text" class="form-control
40                 input-sm" id="MakeInput" placeholder="Vehicle Make" />
41             </div>
42             <div id="AutoCompletedMake"></div>
43             <div class="form-group col-xs-3" id="VehicleModelSection">
44                 <label>Enter Vehicle Model</label> <input type="text"
45                 class="form-control" id="ModelInput" placeholder="
46                 Vehicle Model" /></div>
47             <div id="AutoCompletedModel"></div>
48             <div class="form-group col-xs-3" id="VehicleColorSection">

```

```

    ><label>Enter Vehicle Color</label> <input type="text"
    class="form-control input-sm" placeholder="Vehicles
    Color" id="ColorInput" /></div>
37 <div id="AutoCompletedColor"></div>
38 <div class="form-group col-xs-3" ><label>Select Vehicle
    Mileage</label> <input id="SelectedMileage" type="range
    " value="0" min="0" step="1000" max="150000" oninput="
    this.nextElementSibling.value = this.value">
39 <output id="SelectedMileage">0</output></div>
40 <div class="form-group col-xs-3" ><label>Select model year
    1990-2023</label><input type="text" class="
    form-control" name="datepicker" id="datepicker" /></div>
    >
41 <div class="btn-group btn-group-toggle" data-toggle="
    buttons" id="NewUsedOptionButtonSection">
42 <label class="btn btn-secondary"><input type="radio"
    name="options" id="OptionNew" autocomplete="off" />
    New</label>
43 <label class="btn btn-secondary"><input type="radio"
    name="options" id="OptionUsed" autocomplete="off" /
    > Used</label>
44 </div>
45 <button type="button" class="btn btn-primary btn-sm" id="
    CalculatePriceButton" style="display: block;">Calculate
    Price</button>
46 <div class="alert alert-success" role="alert" id="
    DisplayCalculation" style="display: block;">
47 </div>
48 </form>
49 </div>
50 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/
    jquery.min.js"></script>
51 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/

```

```

        bootstrap.bundle.min.js"></script>
52     <script src="https://cdnjs.cloudflare.com/ajax/libs/
        bootstrap-datepicker/1.2.0/js/bootstrap-datepicker.min.js">
        </script>
53     <script src="script.js"></script>
54     <script>
55         $(document).ready(function() {
56             $("#datepicker").datepicker({
57                 format: "yyyy",
58                 viewMode: "years",
59                 minViewMode: "years",
60                 autoclose:true
61             });
62         })
63     </script>
64 </body>
65 </html>

```

D Styles added to both web pages code

```

1
2  ul {
3      list-style-type: none;
4  }
5  ul > li {
6      display: inline-block;
7  }
8  .center{
9      display: flex;
10     justify-content: center;
11     align-items: center;
12     text-align: center;

```

```

13 }
14 .site-header {
15     background-color: #2E282A;
16     -webkit-backdrop-filter: saturate(180%) blur(20px);
17     backdrop-filter: saturate(180%) blur(20px);
18     top: 0;
19 }
20 .pb-1, .py-1 {
21     padding-bottom: .25rem !important;
22 }
23 .pt-1, .py-1 {
24     padding-top: .25rem !important;
25 }
26 .sticky-top {
27     position: -webkit-sticky;
28     position: sticky;
29     top: 0;
30     z-index: 1020;
31 }
32 .py-2{
33     color: white;
34 }
35 .py-2:hover{
36     color: #CD5334;
37 }
38 .jumbotron{
39     background-color: #616272!important;
40     color: white !important;
41     background-color: #616272 !important;
42     border-bottom: 25px solid #2E282A;
43     border-top: transparent !important;
44     border-left: 25px solid #2E282A;
45     border-right: 25px solid #2E282A;

```

```

46     padding: 10px;
47 }
48 .list-group-item{
49     background-color: #4a434b !important;
50     color: white !important;
51 }
52 .carousel{
53     background-color: #616272 !important;
54     border-bottom: 25px solid #2E282A;
55     border-top: transparent !important;
56     border-left: 25px solid #2E282A;
57     border-right: 25px solid #2E282A;
58 }
59 .carousel-inner{
60     border: solid 40px #6d687d;
61 }
62 .card {
63     margin: 0 auto;
64     /* Added */
65     float: none;
66     /* Added */
67     margin-bottom: 10px;
68     /* Added */
69 }
70 .card{
71     background-color: #4a434b !important;
72     color: white !important;
73 }
74 a:link {
75     text-decoration: none;
76 }
77 a:visited {
78     text-decoration: none;

```



```

79  }
80  a:hover {
81      text-decoration: none;
82  }
83  a:active {
84      text-decoration: none;
85  }
86  #img-card-2{
87      max-width: 100%;
88      height: 50%;
89  }
90  .card-img-top{
91      height: 50%;
92      width: 50%;
93  }
94  .d-block{
95      opacity: .6;
96  }
97  .btn,.btn-primary, .btn-sm{
98      margin-left: 2px !important;
99      margin-right: 2px !important;
100     margin-top: 5px !important;
101     margin-bottom: 2px !important;
102     display: inline;
103 }
104 .form-control{
105     margin-bottom: 5px !important;
106 }
107 form{
108     margin-bottom: 5px !important;
109     color: white !important;
110 }
111 html{

```

```

112  -webkit-touch-callout: none;
113  /* iOS Safari */
114  -webkit-user-select: none;
115  /* Safari */
116  -khtml-user-select: none;
117  /* Konqueror HTML */
118  -moz-user-select: none;
119  /* Old versions of Firefox */
120  -ms-user-select: none;
121  /* Internet Explorer/Edge */
122  user-select: none;
123  /* Non-prefixed version, currently*/
124  background-color: #616272 !important;
125 }
126 .container-fluid,body{
127     background-color: #2e282a !important;
128 }
129 @media (max-width: 540px) {
130     .carousel-caption > h1{
131         font-size: 1vh !important;
132     }
133     .carousel-caption > p{
134         font-size: .5vh !important;
135     }
136 }
137 @media (max-width: 300px) {
138     .carousel-caption > h1{
139         font-size: .8vh !important;
140     }
141     .carousel-caption > p{
142         font-size: .4vh !important;
143     }
144 }

```

E JavaScript client-side logic code

```
1 let jsonFetchedData = [];
2 const url = 'https://api.npoint.io/02a6606bd83177762972';
3 let VehicleMakes = [];
4 let VehicleModels = [];
5 let VehicleBaseColors = [];
6 const MakeInput = document.getElementById('MakeInput');
7 const AutoCompleteMake = document.getElementById('AutoCompletedMake');
8 const ModelInput = document.getElementById('ModelInput');
9 const AutoCompleteModel = document.getElementById('AutoCompletedModel');
10 const VehicleMileageInput = document.getElementById('SelectedMileage');
11 const AutoCompletedColor = document.getElementById('AutoCompletedColor');
12 const ColorInput = document.getElementById('ColorInput');
13 const NewUsedOptionButtonSection = document.getElementById('
    NewUsedOptionButtonSection');
14 const CheckNewOption = document.getElementById('OptionNew');
15 const CheckUsedOption = document.getElementById('OptionUsed');
16 const CalculatePriceButton = document.getElementById('CalculatePriceButton
    ');
17 const displayCalculate = document.getElementById("DisplayCalculation");
18 const dateInput = document.getElementById('datepicker');
19 let SelectedMake = '';
20 let SelectedModel = '';
21 let SelectedYear;
22 let SelectedColor;
23 let ModelYear;
24 let isSeletectedConditonNew;
25 let BasePriceNew;
26
```

```

27 function GETRequestOnlineAPI(url, callback){
28     let jsonObject;
29     fetch(url)
30         .then(response => response.json())
31         .then(data => jsonObject = data)
32         .then(() => callback(jsonObject))
33 }
34
35 GETRequestOnlineAPI(url, getAPIData);
36
37
38 function getAPIData(Objects){
39     Objects.forEach((i) => {
40         jsonFetchedData.push(i);
41     });
42     VehicleMakes = uniqueVehicleMakeValues(jsonFetchedData);
43 }
44
45 function uniqueVehicleMakeValues(array) {
46     return [...new Set(array.map(item => item.Car_Make))];
47 }
48
49 function AutocompleteMatch(input, options) {
50     if (input == '') {
51         return [];
52     }
53     let pattern = new RegExp(input, 'gi')
54     return options.filter(function(option) {
55         if (option.match(pattern)) {
56             return option;
57         }
58     });
59 }

```

```

60
61 function AutoCompleteSelector(input,options,element) {
62     element.innerHTML = '';
63     let button = '';
64     let selected = AutocompleteMatch(input,options);
65     for (i=0; i<selected.length; i++) {
66         button += '<li >'+`<button type="button" class="btn btn-primary
        btn-sm " id="${selected[i]}" >` + selected[i] + '</button>'+`<
        /li>`;
67     }
68     element.innerHTML = `<ul id="${element.id}">` + button + '</ul>';
69 }
70
71 function DeleteNotSelectedElements(value,element){
72     let SelectedOption = value.innerHTML;
73     let liElements = document.getElementById(`${element.id}`).
        getElementsByTagName('li');
74     for (var i = 0, len = liElements.length; i < len; i++ ) {
75         try {
76             if(liElements[i].innerText!=SelectedOption){
77                 element = document.getElementById(`${liElements[i].
                    innerText}`);
78                 element.parentNode.removeChild(element);
79             }
80         } catch (error) {
81
82             return;
83         }
84     }
85 }
86
87 MakeInput.addEventListener('input', (event) =>{
88     AutoCompleteSelector(event.target.value, VehicleMakes, AutoCompleteMake)

```

```

    ;
89     let temp = document.getElementById(AutoCompleteMake.id).
        getElementsByTagName('button');
90     for (let i = 0; i < temp.length; i++) {
91         temp[i].addEventListener('click', (e) =>{
92             DeleteNotSelectedElements(e.target, AutoCompleteMake);
93             UpdateSelectedVehicleMake(e.target);
94             removeVehicleModelOptions();
95         })
96
97     }
98 });
99
100 function UpdateSelectedVehicleMake(val) {
101     SelectedMake = val.innerText;
102     DisplayVehicleModelOptions();
103     updateDisplayCalculation();
104 }
105
106 function DisplayVehicleModelOptions() {
107     document.getElementById("VehicleModelSection").style.display="block";
108     UpdateVehicleModels();
109     ModelInput.addEventListener('input', (event) =>{
110         AutoCompleteSelector(event.target.value, VehicleModels,
            AutoCompleteModel);
111         let temp = document.getElementById(AutoCompleteModel.id).
            getElementsByTagName('button');
112         for (let i = 0; i < temp.length; i++) {
113             temp[i].addEventListener('click', (e) =>{
114                 DeleteNotSelectedElements(e.target, AutoCompleteModel);
115                 UpdateSelectedVehicleModel(e.target);
116             })
117         }

```

```

118
119     });
120 }
121
122 function UpdateVehicleModels() {
123     VehicleModels = [];
124     VehicleBaseColors = [];
125     for (let i = 0; i < jsonFetchedData.length; i++) {
126         if (jsonFetchedData[i].Car_Make === SelectedMake) {
127             VehicleModels.push(jsonFetchedData[i].Car_Model);
128             VehicleBaseColors.push(jsonFetchedData[i].Car_Color);
129         };
130     }
131     VehicleModels = [...new Set(VehicleModels)];
132     VehicleBaseColors = ["White", "Black", "Gray", "Silver", "Blue", "Red",
        "Brown", "Green", "Orange", "Beige", "Purple", "Gold", "Yellow"];
133     updateDisplayCalculation();
134 }
135 function removeVehicleModelOptions() {
136     MakeInput.addEventListener('input', () => {
137         document.getElementById("VehicleModelSection").style.display = "none";
138     });
139     UpdateVehicleModels();
140 }
141
142 ColorInput.addEventListener('input', (event) => {
143     AutoCompleteSelector(event.target.value, VehicleBaseColors,
        AutoCompletedColor);
144     let temp = document.getElementById(AutoCompletedColor.id).
        getElementsByTagName('button');
145     for (let i = 0; i < temp.length; i++) {
146         temp[i].addEventListener('click', (e) => {
147             DeleteNotSelectedElements(e.target, AutoCompletedColor);

```

```

148         UpdateSelectedColor(e.target);
149     })
150 }
151 });
152
153 function UpdateSelectedVehicleModel(val) {
154     SelectedModel = val.innerText;
155     updateDisplayCalculation();
156 }
157
158 function UpdateBaseVehicleYear(val) {
159     ModelYear= val.innerText;
160 }
161 function UpdateSelectedColor(val) {
162     SelectedColor = val.innerText;
163 }
164
165
166 CheckNewOption.addEventListener('click', (event) => {
167     if(event.target.value == 'ON') {
168         isSeletectedConditonNew = true;
169     }
170 })
171
172 CheckUsedOption.addEventListener('click', (event) => {
173     if(event.target.value == 'ON') {
174         isSeletectedConditonNew = false;
175     }
176 })
177
178
179 CalculatePriceButton.addEventListener('click', (event) => {
180     CalculateFinalPrice();

```



```

181  })
182
183  function updateDisplayCalculation() {
184      displayCalculate.innerHTML = '';
185  }
186
187  function CalculateFinalPrice() {
188      displayCalculate.innerHTML = "";
189      let a = AutoCompleteMake.childNodes[0].innerText;
190      let b = AutoCompleteModel.childNodes[0].innerText;
191      let c = AutoCompletedColor.childNodes[0].innerText;
192      let colorsHashtable = new Map();
193      colorsHashtable.set("White", 0);
194      colorsHashtable.set("Black", 0);
195      colorsHashtable.set("Gray", 300);
196      colorsHashtable.set("Silver", 1000);
197      colorsHashtable.set("Blue", 1000);
198      colorsHashtable.set("Red", 300);
199      colorsHashtable.set("Brown", 600);
200      colorsHashtable.set("Green", 600);
201      colorsHashtable.set("Orange", 500);
202      colorsHashtable.set("Beige", 1000);
203      colorsHashtable.set("Purple", 100);
204      colorsHashtable.set("Gold", 2000);
205      colorsHashtable.set("Yellow", 200);
206
207      let temp = [];
208      for (let i = 0; i < jsonFetchedData.length; i++) {
209          let j = jsonFetchedData[i];
210          if(j.Car_Make==a && j.Car_Model==b) {
211              temp.push(j)
212          }
213      }

```