

# Hands-on lab: Part 1: Front-end Enhancement



## Estimated time needed: 45 minutes

In this lab, you'll enhance the front-end of your application by adjusting logic and addressing styling-related elements.

## Objectives

In this lab, you will:

- Convert the "States" dropdown into a searchable textbox.
- Change the color scheme of the application.
- Improve the look and feel of the application.

## Prerequisites

1. Before moving forward, make sure you have finished all the previous labs and created the Car Dealership website. Also make sure that the project is runnable and the functionality and output are as expected.
2. Sentiment Analyzer service deployed on Code Engine should be deployed and accessible. Please refer to the section Deploy sentiment analysis on Code Engine as a microservice in the lab [Create Django Proxy Services Of Backend APIs](#) for the same.
3. Backend service with Express-MongoDB should be running on one of the terminals. Please refer to the lab [Implement API endpoints using Express-Mongo](#) for the same.
4. The front-end of the application should have been built. Please refer to the section Build the client-side and configure it in the lab [User Management](#) for the same.
5. The Django (main application) server should be running. Please refer to the section Environment setup in the lab [Build CarModel and CarMake Django Models](#) for the same.

## Convert the 'States' dropdown on the 'Dealerships' page into a searchable textbox:

To transform the "States" dropdown on the "Dealerships" page into a searchable textbox, you will be working on the Dealers.jsx file:

1. Access the frontend/src/components/Dealers/Dealers.jsx file.
2. You will notice that the code for the dealership dropdown is presented within the `<select>` dropdown element:

```
<select name="state" id="state" onChange={(e) => filterDealers(e.target.value)}>
  .....
</select>
```

3. Substitute the existing `<select>` element with an `<input>` field incorporating the following attributes:

- Enables users to search for states by typing into the text box.
- Filters the displayed dealers based on the entered search query, matching the state.
- Resets the displayed dealers to the original list when the input field loses focus.

```
<input type="text" placeholder="Search states..." onChange={handleInputChange} onBlur={handleLostFocus} value={searchQuery} />
```

**Observe the functions within the input element. Now, let's create them.**

1. Create a new function named `handleInputChange` to manage input changes and filter the dealers based on the entered state query.

```
const handleInputChange = (event) => {  
  const query = event.target.value;  
  setSearchQuery(query);  
  const filtered = originalDealers.filter(dealer =>  
    dealer.state.toLowerCase().includes(query.toLowerCase()))  
  );  
  setDealersList(filtered);  
};
```

- This function is triggered each time the value inside the input field changes.
- It retrieves the current value typed into the input field by the user, storing it as a query in the `setSearchQuery` variable for filtering dealerships.
- A new array is generated, containing only the dealerships whose state matches the entered query.
- Both the query and the dealer's state are made case-insensitive by converting them to lowercase, ensuring a more user-friendly search experience.
- The function then renders the dealerships that match the entered state query.

2. In summary, the `handleInputChange` function dynamically filters the list of dealerships based on the user's input in the search field. It updates the displayed `dealersList` in real-time, providing a responsive search functionality for the user.

3. Create the `handleLostFocus` function to ensure that when the user leaves the search input empty and clicks or tabs away, the list of dealerships resets to the original list.

```
const handleLostFocus = () => {  
  if (!searchQuery) {  
    setDealersList(originalDealers);  
  }  
}
```

- This function verifies whether the `searchQuery` state is empty upon execution.
- If the `searchQuery` is indeed empty, it reverts the list of dealerships to the original state before any search was initiated.
- The `handleLostFocus` function is invoked when the user clicks outside the input field or tabs away from it, triggered by the `onBlur` event.

4. Add this code along with the other previously defined state variables:

```
const [searchQuery, setSearchQuery] = useState('');
```

- This utilizes the `useState` hook to create a state variable named `searchQuery` and a corresponding function `setSearchQuery` to update its value.

- This `searchQuery` state variable holds the value entered by the user in the search input field in real-time. As the user types into the search field, this state variable is updated by calling `setSearchQuery`, and it triggers the re-rendering of the component, reflecting any changes made to the search query.

5. Initialize and set the state variables for managing the original list of dealers.

- Add the below code where the `dealersList`, `searchQuery`, and `states` variables and their setter functions are initialized:

```
const [originalDealers, setOriginalDealers] = useState([]);
```

In this code, the state variable `originalDealers` and its setter function `setOriginalDealers` are used for keeping track of and updating this list of original dealers.

- Place the below code inside the `getDealers` function where the other states (`setStates` and `setDealersList`) are updated.

```
setOriginalDealers(all_dealers);
```

This line sets the state variable `originalDealers` with the array of all dealers fetched from the API which will be used to store the original list of dealers before any filtering.

6. Ensure that all the changes are saved.

7. Run the below commands to build the client side of your app:

```
cd /home/project/xrwvm-fullstack_developer_capstone/server/frontend
npm run build
```

8. Test the app output by accessing the Dealership details page.

9. Observe the appearance of a search-box for dealership searches, replacing the previous dropdown.

Dealerships				
Home About Us Contact Us				
Dealer Name	City	Address	Zip	<input type="text" value="Search states..."/>
<a href="#">Holdlamis Car Dealership</a>	El Paso	3 Nova Court	88563	Texas
<a href="#">Temp Car Dealership</a>	Minneapolis	6337 Butternut Crossing	55402	Minnesota
<a href="#">Sub-Ex Car Dealership</a>	Birmingham	9477 Twin Pines Center	35285	Alabama
<a href="#">Solarbreeze Car Dealership</a>	Dallas	85800 Hazelcrest Circle	75241	Texas

10. Search for a state by entering the search query, for example, `texas`, as indicated below:

Dealer Name	City	Address	Zip	<input type="text" value="texas"/>
<a href="#">Holdlamis Car Dealership</a>	El Paso	3 Nova Court	88563	Texas
<a href="#">Solarbreeze Car Dealership</a>	Dallas	85800 Hazelcrest Circle	75241	Texas
<a href="#">Job Car Dealership</a>	Dallas	253 Hanson Junction	75216	Texas
<a href="#">Tempsoft Car Dealership</a>	San Antonio	5057 Pankratz Hill	78225	Texas
<a href="#">Treeflex Car Dealership</a>	El Paso	0 Rieder Trail	79994	Texas
<a href="#">Namfix Car Dealership</a>	San Antonio	95321 Superior Hill	78245	Texas
<a href="#">Onela Car Dealership</a>	Houston	5423 Spaight Road	77218	Texas

## Change the color scheme of the application:

In this section, you will learn the procedure for changing the color scheme of the app in the following areas:

- Aspects related to the app home page, which include:
  - A. Changing the background color of the Navbar.
  - B. Modifying the background color of the `View Dealerships` button.
- Aspects related to the Dealership Review pages, which include:
  - A. Adjusting background colors related to the Review panel.
  - B. Customizing the on-hover border connected with Review icons

**Aspects related to the app Home page**

### A. Changing the background color of the Navbar

1. Open the file `frontend/static/Home.html`.
2. You'll see the following code snippet, which imparts the current dark turquoise background to the Navbar within the application:

```
<nav class="navbar navbar-expand-lg navbar-light" style={{backgroundColor:"darkturquoise",height:"1in"}}>
```

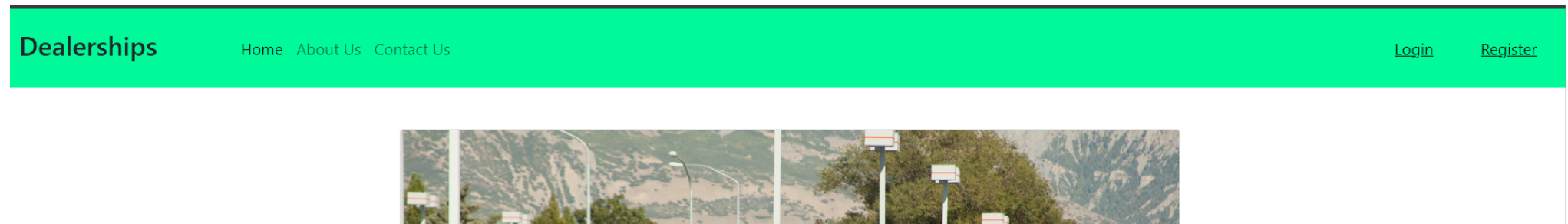
3. Substitute this with a color of your preference (for example, `mediumspringgreen`) to modify the background color of the header.

```
<nav class="navbar navbar-expand-lg navbar-light" style="background-color:mediumspringgreen; height: 1in;">
```

4. Save the changes.
5. Run the provided command to build the client side and display the above changes:

```
npm run build
```

6. Refresh the application page.
- Note:** Ensure that you perform steps 4, 5, and 6 every time you make a modification to view the updated app output.
7. Observe the changed background for the Navbar, as depicted here:



### B. Modifying the background color of the View Dealerships button

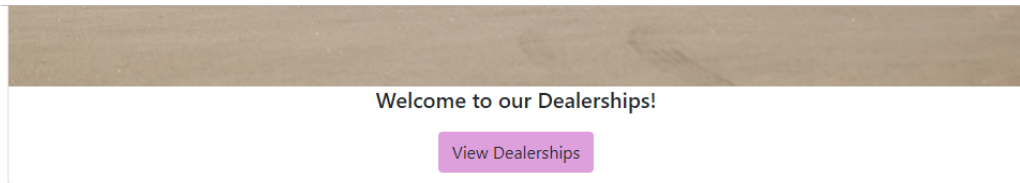
1. The following code represents the background colour of the View Dealerships button on the application's home page:

```
<a href="/dealers" class="btn" style="background-color: aqua;margin:10px">View Dealerships</a>
```

2. Adjust this to your preferable colour (for example, plum, a shade of purple).

```
<a href="/dealers" class="btn" style="background-color: plum; margin:10px">View Dealerships</a>
```

3. Observe the updated colour scheme of the View Dealerships button, as depicted below.



**Note:** You have the flexibility to improve the color scheme by selecting and applying colors of your choice to the Navbar and View Dealerships button.

#### Aspects related to the Dealerships and Reviews pages

The styles in the `Dealers.css` file are tailored for the Dealerships and Reviews panel in the application.

##### A. Adjusting background colors related to the Review panel

1. The `review_panel` class contains the code for styling the Dealership review panel.
2. Notice that it exhibits a solid grey border.

```
border: solid grey;
```

3. Adjust the code to give it a solid purple border:

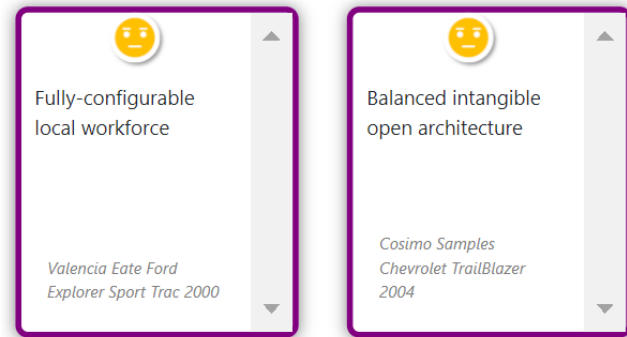
```
border: solid purple;
```

4. You'll notice that the Review panel has acquired a solid purple border, resulting in a changed appearance.

# Alphazap Car Dealership



Washington,108 Memorial Pass, Zip - 20005, District of Columbia



## B. Customizing the on-hover border connected with Review icons





1. The `.review_icon:hover` class applies styling to the Review icon when a user hovers over it.
2. Notice that it takes on a Solid light gray background upon hovering.

```
border: solid lightgray;
```

3. Adjust the color to a black hue with a thinner border (e.g., 2 pixels) for a distinct yet slim background appearance during hover.

```
border: 2px solid #000000;
```

4. Observe the modified icon with a thin black border, displaying a distinct presence when hovered over.

88563	Texas	
55402	Minnesota	
35285	Alabama	
75241	Texas	

**Note:** You can enhance the color scheme by choosing colors that resonate with your preferences for both the Review panel's border and the on-hover border of Review icons.

## Altering the look and feel of the application

In this section, you will learn the steps to refine the visual aspects of the Dealer Review panel in the application.

This includes:

- Adjusting the font size
- Ensuring proper font alignment for an improved overall appearance

### Refining the visual aspects of the Dealer Review panel in the application

#### Adjusting the font size

1. The `.reviewer` class defines the styling for User reviews.
2. The current font size is set to small.
3. Adjust the font size to a specific value (for example, 18 pixels) to enlarge the Review text for improved readability.

```
font-size: 18px;
```

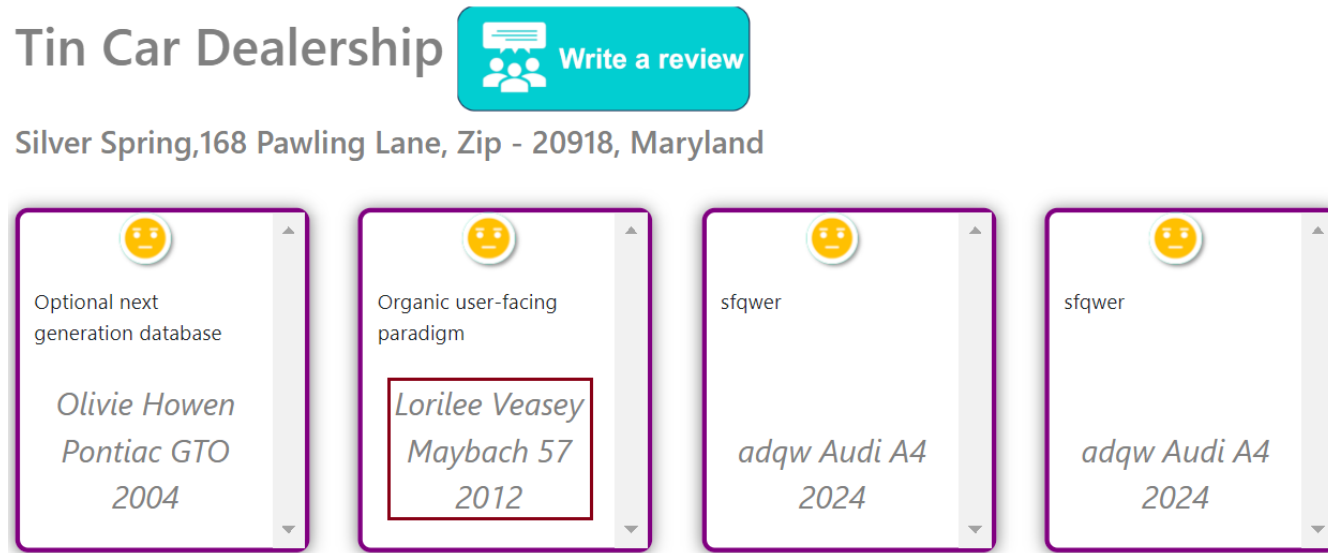
#### Ensuring proper font alignment for an improved overall appearance

1. The `text-align` property is not specified.
2. Thus, the browser's default text alignment (typically left-align) is applied, resulting in left-aligned Review text.
3. Center the text to ensure the User reviews appear at the center of the Review pane.

```
text-align: center;
```



4. Once the aforementioned two styling adjustments are implemented, the updated Review panel will present as follows:



The Review panel's look and feel have been enhanced as outlined above.

## Conclusion

Congratulations! You have completed this lab.

In this lab, you have:

- Successfully transformed the "States" dropdown on the "Dealerships" page into a searchable textbox, providing users with the ability to filter dealerships by entering search strings.
- Improved the color scheme of both the Navbar and Dealerships button on the app's home page, as well as the colors of the Review panel and review icons on the "Dealership Review" page.
- Fine-tuned the visual elements of the Dealer Review panel, adjusting aspects like font size and font alignment.

The implementation of these procedures has brought about an enhancement in the application's interface, color scheme, and aesthetic appeal.

## Author(s)

K Sundararajan

© IBM Corporation. All rights reserved.