Restaurant Website

# A bunch of food on a table Description automatically generatedFront Welcome page

This is the opening page. It has one button saying get started. This button is linked to a app.get() function that responds with the home page being rendered. The restaurant object is passed in containing all the restaurants in the data base. These restaurants were received using the findAll() function. This page cannot be scrolled.

# A picture containing indoor, building, window, front Description automatically generatedHome Page

Here the home page is shown. There is a nav bar at the top which is ‘sticky’ which stays at the top of the view port even while scrolling. The restaurants are rendered in cards with the title and the menu titles listed out at the bottom of each card. The cards are responsive to the viewport width using flexbox to space correctly.

Text

Description automatically generatedText

Description automatically generated

Within each card element there is a div for the details of the restaurant. Here I loop through the menus that are assigned to ‘this’ object which is the restaurant the card is for. I find the titles of each menu and display them in a row at the bottom of the card.

Here is the home.handlebars file. The entire page uses a div element which contains all the elements. This is styled using the contents class. The top nav bar has four elements, one of which is active, which sets its CSS styling slightly different to the others. In the div below I look through each object in the restaurant object past through which provides me with the keys and values of each restaurant. I then make a hyperlink tag (<a>) and give it an href to a link containing the id of the restaurant. Within this hyperlink tag I create the div that creates each card and has the details inside. Each card has styling that keeps it slightly grey and dull, and only when hovered over does it show its full colors. It also grows slightly and can be clicked on to show the menus of the restaurant. Scrolling down reveals more cards.

# Graphical user interface Description automatically generatedRestaurant page

Here the menus are displayed using the app.get() function however as can be seen in the screenshot above, the restaurant object being passed in is now just the restaurant we have selected. Also passed through is the menu objects associated with this restaurant, this are displayed using the same {{each}} loop as seen above. There is also another menu rendered with an add Menu button allow for more menus to be added. Above in the nav bar, two buttons are shown. One deletes the entire restaurant using the .destroy() function. The other allows for the restaurant name and image to be edited. This is also where you navigate to editing each menus items. The buttons also change color when hovered over with a transition period of 2s as shown below

Text

Description automatically generated

A picture containing diagram

Description automatically generatedA picture containing text

Description automatically generated

To the left you can see the page to add a menu to the restaurant. It provides a field for the menu name and two item option fields. If more want to be added the add item button will run some javascript creating the fields, that way only the amount of fields needed will be created. The submit button runs a app.post() function that adds the data to the db and then redirects you to the restaurant page.

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Here the addFields() function is shown. The idea is that the js replicated the structure of the html below. It creates a div element and attaches the same class as the one hard coded have of “editItem-container”. It then creates the list element and inside of it appends the label element and the input element that has been created. It does this twice so that the price and item name can be edited.

Below is code written in the server file. When the submit button is clicked it runs this post function.

First the restaurant is found by the id past in by using req.params.id . Then create a new menu object with the .create() function passing in the body which contains the item names in an array, the item prices in an array and the menu title.

Then I loop through the arrays and create item objects using one item name and one item price, and then add it to the menu using addItem().

Diagram

Description automatically generatedHere the editing of menus can be seen that already exist where editing is easily done and submitted.