INFO3067 Week 2 Class 1

Rev 1.0

Categories/MenuItems

Last class we had our CategoryController/CategoryDAO returning a JSON array to the Vue client. Today, we'll bring our MenuItem domain class into the mix and allow the end user to select a category and subsequently view all the menu items that belong to it

1. Create a new **MenuItemDAO** class on the server that retrieves menu items based on a category Id. Add the following code:

```
using ExercisesAPI.DAL.DomainClasses;
using Microsoft.EntityFrameworkCore;

namespace ExercisesAPI.DAL.DAO
{
    public class MenuItemDAO
    {
        private readonly AppDbContext _db;
        public MenuItemDAO(AppDbContext ctx)
        {
            _db = ctx;
        }
        public async Task<List<MenuItem>> GetAllByCategory(int id)
        {
            return await _db.MenuItems!.Where(item => item.Category!.Id == id).ToListAsync();
        }
    }
}
```

2. We'll also need a **MenuItemControlle**r that will return the desired menu item JSON back to our Vue client. Add the following code for it:

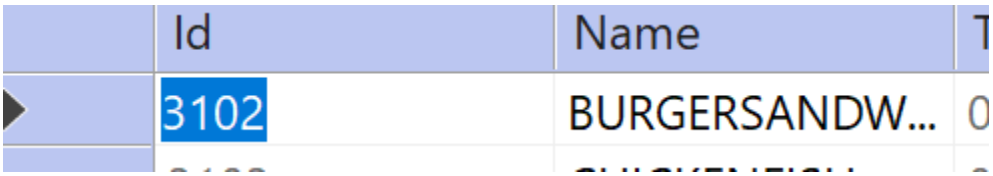
```
using ExercisesAPI.DAL;
using ExercisesAPI.DAL.DAO;
using ExercisesAPI.DAL.DomainClasses;
using Microsoft.AspNetCore.Mvc;

namespace ExercisesAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class MenuItemController : ControllerBase
    {
        readonly AppDbContext _db;
        public MenuItemController(AppDbContext context)
        {
            _db = context;
        }
        [HttpGet]
        [Route("{catid}")]
        public async Task<ActionResult<List<MenuItem>>> Index(int catid)
```

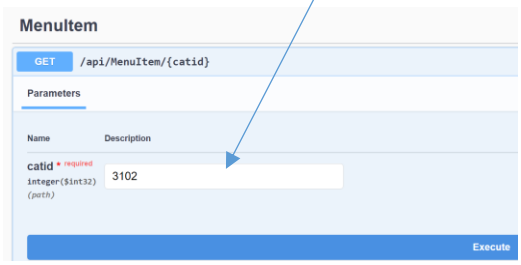
```
        {
            MenuItemDAO dao = new(_db);
            List<MenuItem> itemsForCategory = await dao.GetAllByCategory(catid);
            return itemsForCategory;
        }
    }
}
```

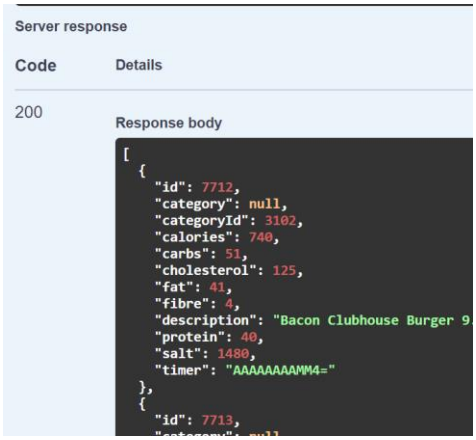
Notice how the Route attribute is expecting a parameter (catid) and how we're using it as an argument for our Index method.

3. To test this out you'll need a valid categoryid from your categories table. Open the categories table in the SQL Server Object Explorer and obtain a valid category Id, the example uses **3102** but yours will be different:



4. Then using that category id, start your exercises project and test it out in Swagger, load the category id from your setup:





Now, let's enhance the CategoryList component by allowing the user to choose a category and display all the menu items for it.

5. Replace the <q-select you currently have with the following markup in the CatalogList component:

```
<q-select
  class="q-mt-lg q-ml-lg"
  v-if="state.categories.length > 0"
  style="width: 50vw; margin-bottom: 4vh; background-color: #fff"
  :option-value="'id'"
  :option-label="'name'"
```

```
      v-model="state.selectedCategoryId"
      :options="state.categories"
      label="Select a Category"
      emit-value
      map-options
      @update:model-value="loadMenuitems()"
    />
```

6. Then add this <div and <q-scroll-area after the <q-select:

```
    <div
      class="text-h6 text-bold text-center text-primary"
      v-if="state.menuitems.length > 0"
    >
      {{ state.selectedCategory.name }} ITEMS
    </div>
    <q-scroll-area style="height: 55vh">
      <q-card class="q-ma-md">
        <q-list separator>
          <q-item
            clickable
            v-for="item in state.menuitems"
            :key="item.id"
          >
            <q-item-section class="text-left">
              {{ item.description }}
            </q-item-section>
          </q-item>
        </q-list>
      </q-card>
    </q-scroll-area>
```

Note that we're creating a scrolled list with a single column using the menu item's description property.

7. Next, add the following **method** that acts as an event handler for the drop down's change event and adds the menu item data to the state variable, notice how it uses the selected category id in the URL:

```
const loadMenuitems = async () => {
  try {
    state.selectedCategory = state.categories.find(
      (category) => category.id === state.selectedCategoryId
    );
    state.status = `finding menuitems for category ${state.selectedCategory}...`;
    state.menuitems = await fetcher(
    `Menuitem/${state.selectedCategory.id}`
    );
    state.status = `loaded ${state.menuitems.length} menu items for
${state.selectedCategory.name}`;
  } catch (err) {
    console.log(err);
    state.status = `Error has occured: ${err.message}`;
  }
};
```

8. Add a couple of state array variable to contain the menu items (bolded code) and selected category, and an Id field:

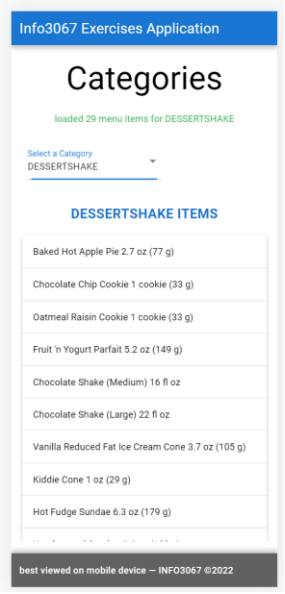
```
    let state = reactive({
```

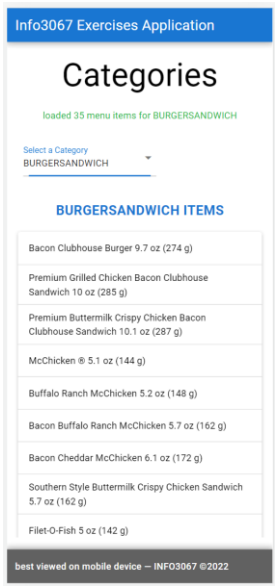
```
  status: "",
  categories: [],
  menuitems: [],
  selectedCategory: {},
  selectedCategoryId: "",
});
```

9. Because this method is triggered from the template, we need to include it in the return statement:

```
return {
  state,
  loadMenuitems,
};
```

10. Save all changes and select a couple of categories now, notice how the region for the items is contained in that <q-scroll-area tag:





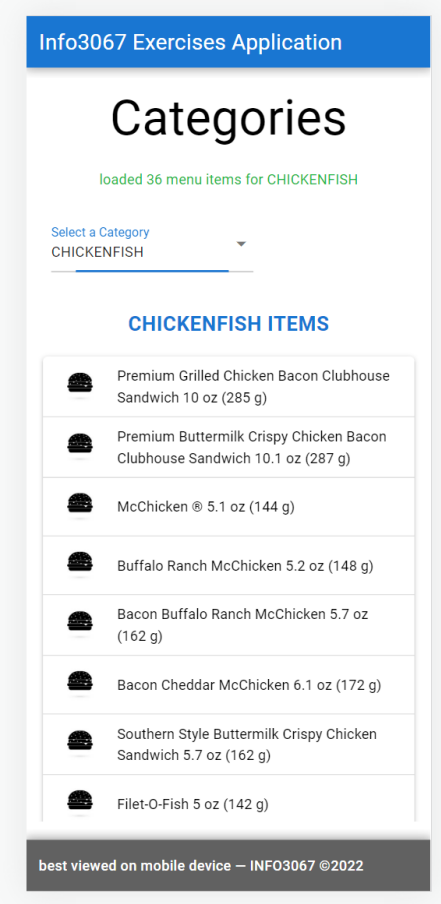
Converting the MenuItem List to a Catalogue Format

As shown in the code above the menu items are displayed as list items. To add a bit of styling replace the <q-item tag contents with this markup:

```
<q-item v-for="item in state.menuitems" :key="item.id">
  <q-item-section avatar>
    <q-avatar>
      <img :src="`/burger.jpg`" />
    </q-avatar>
  </q-item-section>
  <q-item-section class="text-left">
    {{ item.description }}
  </q-item-section>
</q-item>
```

We see that the markup is using a graphic in the <img tag. You can download the **burger.jpg** from the content for week 2 on FOL and place the file in the **public** folder for it to render.

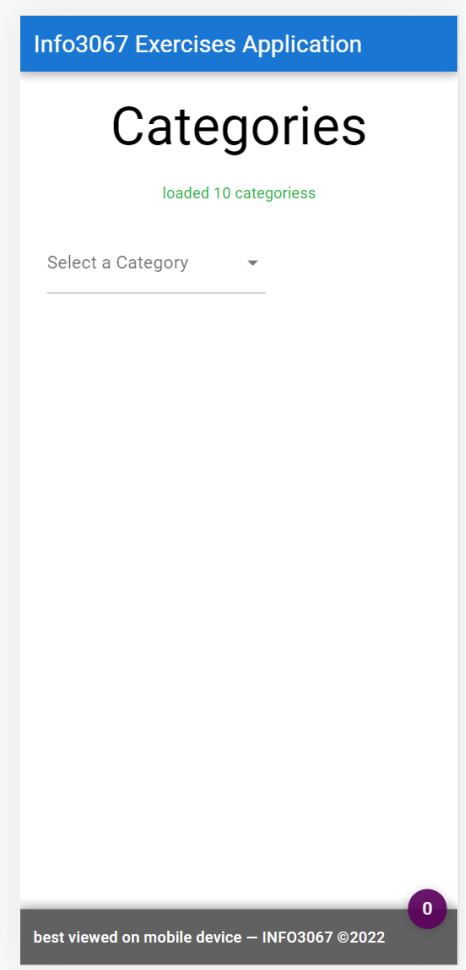
Then rerun the application, choose a category which would produce a page that looks something like in a mobile footprint:

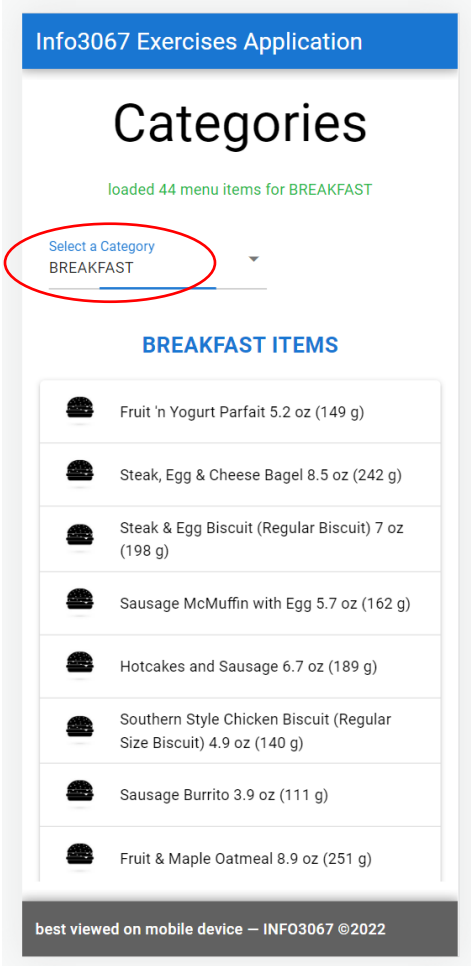


REMEMBER TO PLACE ALL SCREENSHOTS IN A SINGLE WORD DOCUMENT

LAB 3 – Part A (1%)

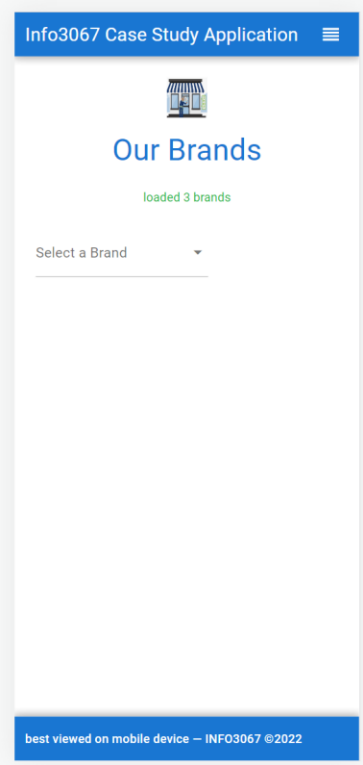
- Add 2 screenshots in **a mobile footprint** to a word document (make sure your **name is visible in the footer**):
    1. Categories page before selecting a category
    2. Categories page after selecting **Breakfast** category

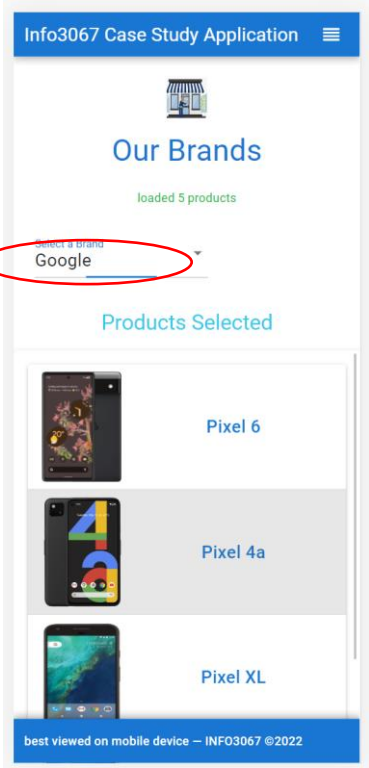




Lab 3 Part B (1%) - Case Study Updates

Proceed now with your brand/product catalogue so that it functions like the category/menu item functionality does. Allow the user to select products by brand then display all individual products as catalogue items. An example follows for phone products (but remember nobody gets to sell phones). The user selects the Google brand and just the phones of the Google brand are displayed:





You'll need to add your images to a folder created in public (I used **img**) and have a separate image per product (unlike the exercise where a single graphic was used). Notice there is a field in the products table called GraphicName (maps **as graphicName** in the JSON), so store the name of the image in the database, not the image itself. You can reference it like so:

```
<q-item-section avatar>
   <q-avatar style="height: 125px; width: 90px" square>
     <img :src="`/img/${product.graphicName}`" />
   </q-avatar>
 </q-item-section>
```

Add 2 screen shots to the Word document again, **both in mobile format**:

1. A screen shot of the catalogue before selecting a product and with the select expanded out with all brands (a minimum of **3 brands)** for a particular product line.
2. As screen shot of the catalogue showing products for a particular brand (minimum **5 products**).

Lab 3 Part C (.5%) - Lab 3 Theory Quiz on FOL - 10 questions

Review Questions

1. What new DAO was added this class?
2. What did the parameter for the only method in the DAO from Q1 represent ?
3. What is returned to controller from the method represented in Q2?
4. What URL would you use to retrieve the return value from Q3 in a browser?
5. What does the last part of the URL from Q4 represent?
6. How would this <img tag from the exercises project differ in the casestudy project::

```
<img :src="`/burger.jpg`" />
```

7. What syntax could you use to incorporate conditional rendering in the markup?
8. How did we use conditional rendering in both projects this class?
9. What is the minimum number of brands you need to account for in the case study?
10. What is the minimum number of products per brand do you need to account for in the case study?
11. T/F – you store individual graphics in the database?