Project 5 - Fetch

Start Assignment

Due Oct 17 by 11:59pm **Points** 20 **Submitting** a website url

Due: Sunday, October 17
Points: 20 points
Deliverables: Upload [Name]BookstoreFetch.war to **cs5244.cs.vt.edu:8080/ArchiveUpload/ (http://cs5244.cs.vt.edu:8080/ArchiveUpload/)**
Resources: **CorsFilter.java (https://drive.google.com/file/d/11PqIl8djz376mwvDY20t1z0KhUYHYu03/view?usp=sharing)**

Overview

In this project, you will combine Project 3 (Vue) with Project 4 (Rest). The result will be a home page and category page that dynamically update depending on the selected category. To do this, we are going to use the Fetch API built into JS6 to retrieve data from the REST API created in Project 5.

Setup your project

Create a new (no-plugin) Gradle project in IntelliJ called [Name]BookstoreFetch. Create two modules:

- server - Gradle with Java and web plugins
- client - Vue with Linter and Router (as in P3)

Once you've done this, **I recommend quitting IntelliJ** before copying the following files into your new project.

From your Vue project (P3) copy the following into the **client** side:

- src folder
- vue.config.js

From your REST project (P4) copy the following into the **server** side;

- build.gradle
- src folder

Set up a Tomcat run configuration for the server (with application context [Name]BookstoreFetch) and **run the Tomcat configuration**. The REST API should work as in Project 4.

Run the client using "npm run serve". The Home and Category pages should look and behave just as they did in Project 3. Look at the address bar. Does it still have "[Name]BookstoreVue" in the path? If so,

modify the vue.config.js file to change that to "[Name]BookstoreFetch".

Fetch Categories for your CategoryNav component

Follow along with these slides to complete this portion of the assignment:

- **Bookstore Fetch slides (https://drive.google.com/file/d/1x-DbqldMvCTFyjoD1-xLiTSsm7KOmP_G/view?usp=sharing)**

Fetch Categories as needed

In addition to CategoryNav, fetch the categories for HeaderCategoryDropdown. If you have categories on your front page, also fetch the categories for HomeCategoryList. Use the ApiService that has already been created.

Fetch Books for your Category component

In your ApiService, create a method called "fetchSelectedCategoryBooks". It should take the category name as a parameter. Use this method in your Category view — similar to how we used ApiService.fetchCategories in the CategoryNav component — to display the book boxes for the selected category. This will involve passing the fetched books as properties to the CategoryBookList and then CategoryBookListItem components.

To load the book images, create a custom JavaScript method that takes a the book title from the database, and returns the image file name. It should look something like this:

```
bookImageFileName: function(book) {
  let name = book.title.toLowerCase()
  name = name.replace(/ /g, '-')
  return `${name}.jpg`
}
```

Modify this code to work with your book-image filenames. For example, my book-image files were "gif" files. Also, I had books with punctuation in the title, so I replaced all punctuation with an empty string.

Finally, to enable the navigation between categories to reload the books on the category page, in App.vue add :key="$route.fullPath" to the router view tag:

```
<router-view id="router-view" :key="$route.fullPath"></router-view>
```

This tells Vue to reload each page when the path changes.

Now navigating to different categories should load the books for each category.

Style and Interface Requirements

Remember that requirements of project are cumulative, so all relevant requirements from Projects 1–4 still apply.

Fix Past Problems

For Project 5, please fix any problems that an instructor or grader has mentioned in their comments to previous project (especially Project 3, your Vue project). The grades and comments for Project 3 may not be out until the second week of this project, but this requirement still holds. If you are still working on issues when your submission is due, let us know in Canvas when you submit Project 5. In general, we try not to take off points twice for the same problem if you did not have time to fix it. For example, if an instructor or grader took off points for something in Project 2, then you did not have time to fix it when you submitted Project 3, so we should not take off points for the same problem. However, for Project 5, you *will* have time to fix these problems, so we will take off points again.

Implement a transition

- Implement a transition somewhere in your project. If the transition is not obvious (for example, sliding dropdown menus or add-to-cart button effects are typically obvious), please let us know what it is in Canvas when you submit Project 5
- If you implement a hamburger menu similar to the one in the hamburger-demo, please do add or change something about the menu that distinguishes it from the one in the demo. For example, maybe the hamburger icon changes to an "X", or maybe the sub-menu comes out from the side.

Buttons

- The CTA button (call-to-action) on the welcome page should take the user to your default category page. Your default category will typically be your first one (with ID 1001). This also means that your CTA button should be named appropriately. For example, it should be called "Shop Now" or "Shop for Books" or something like that. It should not be called "Sign up Now" or "Create an Account" or "Buy this Book Now".
- The cursor should always change to a pointer (hand) when you hover over any button (including icon buttons) or link. The CTA, primary, and secondary buttons should all have a style change (for example, a darker background or bold text) when the user hovers over them.
- Note that when you hover over an element and its style changes, the position of surrounding elements should not change. For example, if you hover over a button and the text changes to bold, the button should be large enough so that the size of the button does not change. If you intend the size of the button to change, there should be enough space around the button so that elements around it do not move.

Padding in icon buttons and cart-count

- Make sure your icon buttons have padding around the icon. The icon should not touch the edge of the button. There should be at least a few pixels of padding around it.
- The same thing applies to the cart-count, whether you put your cart-count in a little circle or inside of your shopping cart. Make sure the count is not to close to edge of the shape that surrounds it.

- If you put your cart count inside (or beside) your cart, make sure it is centered vertically or horizontally (as appropriate).

Book Boxes

- I've seen a few book boxes (in Project 3) that do not wrap until the boxes get very thin. Please set a min-width on your book box that keeps both the book image and the text (book info) easily readable.
  - The book image should not shrink. It should always keep its original height and width (and therefore it will always maintain its correct aspect ratio).
  - The text of the book title should not wrap so much that there is only one word on a line.
- Book boxes should wrap within the space of the browser width that we will be looking at, which is from about 1000 pixels to 1400 pixels. When we expand or shrink the browser between those two widths, we should see the book wrapping occur. For example, we should not have to shrink the browser to 600 pixels before we see evidence of wrapping.

No Book Boxes on Welcome Page

- Do not put book boxes on your welcome page. You can put book images on your welcome page, and even include the title, but do not put the same kind of book boxes that you have on your category page on your welcome page. Users should never be able to mistake your welcome page for your category page.
- Do not put add-to-cart buttons (or read-now buttons) on your welcome page. The user's attention should go to the call-to-action button, and the add-to-cart buttons are distractions from this goal.

Miscellaneous

- Make sure that your header and footer extend the width of the page (up to the max width of the body). For example, do not use a footer that remains at the bottom left (or button right) of the page.
- Make sure your overlays are flush with bottom of image that they overlay. I've seen a couple pages where there are a few pixels of image underneath the transparent overlay.