Storefront App - Upgrade

Authoring 4 - Upgrade Assignment

Assignment Description:

You will be building a digital storefront for a client of your choice. Use Node on the back end and Vue on the front end to achieve this. Your storefront needs to display product images and descriptions as well as videos that demonstrate their use. A user will be able to select / add a product and add it to a shopping cart.

You will need three product categories and five products per category. As an example, consider a storefront for DeWalt, a popular tool manufacturer. DeWalt makes hand tools, power tools, marking and measuring tools, cutting tools etc. Pick three categories and find five products you'd like to showcase for each.

Find product shots and video resources for each tool and decide how you'll show them in your UI (Home Depot, Lowes, DeWalt, etc would be great resources and have UI examples, etc).

Show video on a click, streamed from your back end. Add the product to a shopping cart on a click.

A user should be able to filter products based on category and select any product - and a product quantity - in order to add it to a shopping cart. Your shopping cart UI should update based on whether or not it's empty.

You will need to select or create three product categories with a minimum of five products for each - **15 unique products** in total.

The content for the digital storefront must be retrieved from a database.

Shopping Cart Functionality
============================

The shopping cart's functionality must include the ability to add and remove products, as well as change product quantity.

Your shopping cart should total the prices of all of the products added and give a total price after tax.

The shopping cart should persist between page loads - you will need to store selected products as you see fit, either on the front end using the LocalStorage API or on the back end using SQL and a database.

Dev Stack Requirements
==========================
Use the NEVM stack (Node, Express, Vue, MySql) as your framework. You can re-use some of your work from the winter term, but you'll need to update the database to reflect the different content as well as build in the pieces you'll need to make your shopping cart functional.

You will build the back end separately from the front end - you'll end up with 2 standalone mini-projects that communicate with AJAX. Data retrieval must be done with Express routing and SQL - you **MAY NOT** use any PHP.

Product images, descriptions and videos should be stored server-side and loaded via routes using AJAX.

The video sources must be streamed via a route attached to the video element's source attribute - there shouldn't be any video player logic on the client side, with the exception of any custom UI and related JavaScript code (pause, play, etc) and the logic required to change the video player's route.

Requirements Per Role

Designer

Design the look and feel of a mobile-first web application based on the information above. Creatively, this project is wide open - push yourself to explore the latest trends in mobile-first design. Consider the end user; focus on engaging UX and UI.

Think about interactive elements, including what might be required in terms of Single Page Applications (SPA), hybrid multipage applications, and data transactions (preloaders, microtransactions, flash messaging etc).

Include a Design Reference document (this should be a Google Doc) that includes as much detail as appropriate - branding concerns, creative direction, etc.

Use the SASS library to develop the CSS modules and files for layout / presentation.

Developer

Create the proper project structure including a Github repo / readme. You will need to modify any existing databases to reflect the project requirements. Be sure to test / implement the API endpoints to retrieve and populate content in your application.

Refer to the project design to determine the Vue components and additional functionality you'll need to create. Use the Vue router to control what appears on the page and when.

All data will be loaded via AJAX on initial load and as required during the app's runtime. Create the required mechanisms to accomplish this (the Fetch API / Vue.js components and client-side routing).

You will also need to implement routing with Node and Express on the back end for your data retrieval and updates. Consider implementing some server-side rendering where appropriate.

You should research and implement how to include shopping cart functionality. However, **ALL CODE MUST BE YOUR OWN**.

Implement your functionality on individual branches and merge to the master branch as you go. Think (and develop) in sprints - one piece at a time.

The digital resources could get quite large, so consider an alternative to Github for storing them. You can submit a dropbox link or a compressed media archive with your submission if you need to.

Submission

Homework must be submitted by midnight of May 31st. Include the following:

Github repo: master branch with any other dev branches you feel you need

Name the dev branches appropriately per feature IE des.tvr.icons, des.tvr.player, dev.tvr.login, dev.tvr.commenting

Design document (Google Drive doc)

Dev notes (Google Drive doc)

Submit the repo link via FOL dropbox

Please follow correct folder and file structure as outlined in class

Additional Information:

Missed tests/exams will not be rescheduled without some valid evidence of some important event over which the student has no control (e.g., Court appearance, death in the family). Missed tests or exams, therefore, can receive a zero. The students are advised to notify the professor prior to missing the test.

Students are expected to hand in all assignments to the course instructor on the due date, and all assignments must be submitted in the format specified by the instructor (e.g., on FOL, in printed form, on a specific lab computer, etc.); assignments will not be accepted in any format other than that specified.

Late assignments will not be accepted, nor will make up test or assignments be permitted, without some valid evidence of some important event over which the student has no control (e.g., documented illness, death in the family). Missed tests or assignments, therefore, will receive a mark of zero. Late assignments and make-up tests will only be permitted following the submission of adequate documentation acceptable to the instructor (e.g., a doctor's note). Students are advised to notify the instructor prior to missing an assignment due date or a scheduled test.

Immediately upon return from an illness/absence in which a test or assignment has been missed, the student is responsible for contacting the course instructor to discuss the problem. The instructor will make arrangements for any student deemed eligible. The alternative test/assignment will be of equal value to the one missed with no grade penalty. The timeline and due dates will be determined by the course instructor.

At mid-term, any unsatisfactory results will be reported to the student.

This course may be revised by the professor with suitable notification to the students. Students are responsible for making arrangements to pick up missed handouts, assignments and course announcements from classmates.

Plagiarism (e.g., failure to acknowledge sources used, submitting another student's work under your name, or producing work for another student to submit) is a serious academic offense that shall result in appropriate penalties, to be determined at the discretion of the course professor in consultation with the chairperson of the Communication Arts division. The penalties shall range from failure of an assignment to possible failure of the course. Students shall not make the assumption that any provision will be made by the professor to permit the student to rewrite or redo failed assignments.