CMPS 356 Enterprise Application Development - Spring 2019
Lab 8 – Client-side JavaScript

Objective

The objective of this lab is to practice implementing Web UI and calling Web API using Client-side JavaScript. Particularly DOM manipulation, events handling and using Fetch API.

Overview

This lab has two parts:

- **Part A**: Extend the Banking App to make the app functionality accessible via Web UI using Client-side JavaScript (1.5h).
- **Part B**: Extend the **BookStore** App to make the app functionality accessible via Web UI using Client-side JavaScript (1.5h).

Part A - Extend the Banking App to make the app functionality accessible via Web UI

1. Sync cmps356-content repo to get the Lab files.
2. Copy ***Lab8-ClientJS*** folder from cmps356-content repo to your repository.
3. Open ***Lab8- WebApi \BankingApp*** in WebStorm. Run **npm install** to install the packages.
4. Develop a Web UI to make the App functionality accessible to users:

Web Page	**Functionally**
index.html	First, the index page should have a main navigation menu providing 3 links: • <u>*Home*</u> link is the home page that allows getting accounts. • <u>*Add Account*</u> link allows adding an account. • <u>*Add Transaction*</u> link allows adding a transaction. - All the app pages should be accessible from the index.html page. - This page allows getting accounts by type. It provides a dropdown to select the account type: *Saving*, Current or All. When the page load or when a different account type is selected then the page should fetch the accounts from /api/accounts?acctType= - For each account with the balance=0, a *Delete* button is provided to enable deleting the account. Then this button is clicked the account should be deleted using /api/accounts/:id Web API. Also, the account row should be deleted from the html accounts table.
acct-form.html	Allows creating a new account by posting the account data to */api/accounts*
acct-trans.html	Allows selecting a particular account from the accounts dropdown then submitting a deposit or withdrawal transaction. The new transaction should be posted to /api/accounts/:id/trans. The page should display any error returned from the Web API (e.g., insufficient balance).

5. Test the Banking App Web UI and provide screenshots as evidence in the testing document.

Part B - Extend the Book Store to make the app functionality accessible via Web UI.

Deadline – Next week 1 hour before the Lab

1. Open ***Lab8- WebApi \BookStore*** in WebStorm. Run **npm install** to install the packages.
2. Design and implement a Web UI to make the App functionality accessible to users:

Important Note: The baseline solution will be posted on Wednesday at 1pm as the Lab B52 students will be submitting their assignment during that time. So, until the baseline solution is posted, you can use your own implementation.

Web Page	**Functionality**		
index.html	First, the index page should have a main navigation menu providing two links: - *Books* link is the home page that allows searching for books. - *Add Book* link allows adding a book. The index page should allow the user to enter the parameters to search for books by *category*, *name*, *isbn*, *author* or *pageCount*. The Book category and the Author parameter should be a **dropdown** filled with the data returned from /api/categories and /api/authors respectively. Upon submission of the search parameters, the app should fetch the books then display them in a table. Based on the selected search option the app should make use of the Web API developed in Lab 7 to fetch the books. You should use the Web API implementation in the provided base solution. Besides each book displayed in the table, the app should offer the following links: - details: this link should display the book details page. - update: this link should allow updating the book. - delete: this link should allow deleting the book. After the user confirmation, the app should delete the book using the Web API @ /api/books/:isbn		
	Reminder of books API from Lab 7:		
	Get	*/api/books/:isbn*	Gets a book by isbn.
	Get	/api/books?category=	Returns all the books for a particular category.

			If the **category** is not provided then you get all the books otherwise you should get the books that match that category. E.g. *category = Programming* should get all the programming books.
	Get	/api/books?name=	Get the books where the name matches the name parameter.
	Get	/api/books?pageCount=	Get the books with pages >= the pageCount parameter. E.g. if the pageCount=200 you should get all the books with pages >= 200.
	Get	/api/books?author=	Display all the books authored by that specific author.
book.html	This page displays the details of the selected book. The book details should be fetched using */api/books/:isbn*		
book-editor.html	Provides a form to allow adding a book to the book-catalog. The entered book should be posted to the Web API @ /api/books/ The form can also be used for updating an existing book. The updated book should be sent to the Web API @ /api/books/		
books-summary.html	Display a table that contains the author name and the number of books they have authored. The data for this page should be fetched using *Get /api/books/summary*		

Author Name	**Book Count**
James	2
Ali	4
Omar	7

Make sure the app main menu is accessible in all the app pages.

3. Test the **BookStore** Web UI and provide screenshots as evidence in the testing document.

After you complete the lab, fill in the ***Lab8-TestingDoc-Grading-Sheet.docx*** and save it in **Lab8-ClientJS** folder. Sync your repository to push your work to Github.