COMP5347 Assignment 1

Client-side Development

The primary goal of this assignment is to demonstrate that you have the basic understanding of the core client-side technologies (HTML, CSS, and JavaScript) and you are able to create HTML pages with desirable style and certain interactive features.

A2ZBooks has developed a digital transformation strategy part of which is the development of a web store for selling books. They ask you, as a front-end web developer, to develop the main web page with the key functionalities/features that were defined in the first iteration. They provided you with small *start file set*(see Resources below), which contains a complete CSS file and a skeleton HTML file. Your task is to develop the main web page of A2ZBooks online book store that implements the following requirements without using any third-party CSS frameworks or JavaScript libraries.

Main Requirements

The following functionalities must be implemented in your Web page:

* The main page should display the main information of each book including a thumbnail picture, title, author, year, star rating, price, publisher and category (see the enclosed Figures under Resources as an example). The star rating position can be located in either in line with the title text or in a separate column of the page (e.g. "Ratings" column after the "Title" column).
* The page must load the book items list from the provided json file (inside the start file set, see Resources section). Details on how to read the json files will be described later in this assignment description. Copying the content of json file into your javascript file is **not** allowed.
* End users can search books based on their title. The rows that match the search term will be highlighted with a coloured background of your choice. This search feature does not affect the list of shown items, only changing the background colour of the rows that match the search terms (can be word(s) or letter(s)).
* End users should be able to filter books through their categories (only books with the selected category are shown in the list). The list of category works like a drop-down menu and should cover an extra category that is currently not in the bookstore for boundary test use. Meanwhile, end users should always have the option to return to the default status (display all books no matter which category they belong to).
* The Search and Filter functions should work together and combine the result of each other.
* End users should be able to select books and add them to the “Shopping Cart” through check boxes and a button “Add to cart”.  Users can only select one book at a time through checkbox.
* A quantity input box will be prompted when the users select a book item and click "Add to cart" button. The page can either show a popup or a text-box asking about the quantity of the selected item. The book selection must be cleared after the book(s) is added to cart.
* End users can clear the shopping cart through the “Reset the cart” button. Users should be prompted with a message box to confirm their desired action (“Reset the cart” or “Cancel”).
* The “Shopping Cart” should always show the correct number of books in the cart based on the end user actions (note: based on the total quantity of added books).
* The page provides a checkbox (at the top of the page) to enable "Dark Mode" where the themes of the page are changed to use grey colours or other dark colours. This dark mode will affect both the webpage background colour, texts, highlight colour and any other relevant colours.

Design and Implementation Requirements

The online book store data and required thumb images are provided in the enclosed start file set. The layout does not have to be the same as the example in Figure 1. For instance, you may modify the given CSS file to have a slightly different colour scheme, border and text styles.

You **must** consider the behaviour of boundary cases and design suitable UI changes to **notify/alert** the end users of inappropriate inputs or outputs. Sample boundary cases include, but are **not restricted** to the followings:

* Search term does not appear in any title
* Users select the category that does not contain any book

Make sure that your HTML, CSS rules and JavaScript code are kept in separate files. Inline style or JavaScript code is **not** accepted.

Make sure that your HTML and CSS files are “clean” and easy to maintain. They should not contain any unnecessary automatically generated content such as scattered <span> element with pieces of styles.

Use of jQuery or other JavaScript libraries is **not** allowed in this assignment. Use of Bootstrap or other CSS framework/library is **not** allowed either.

Deliverables and Submission Guidelines

Submit a zip folder containing the HTML, CSS and JavaScript files through the provided link. Please do not include the image files in your submission. *Students will be permitted to use the image files, including any images they need to add, during their demo.*

The zip file should include Unikey in the file name. For example, *asabcd1234\_Assignment1.zip* where abcd1234 is your Unikey.

*Important Note: A software system will be used to detect software code similarity and allegations will be reported to the University’s central unit.*

Resources

[Start file set](https://canvas.sydney.edu.au/courses/21515/files/9665821/download?wrap=1) (HTML skeleton and CSS files)

[Figures](https://canvas.sydney.edu.au/courses/21515/files/9665822/download?wrap=1) (Requirements and Design of A2ZBooks web page)

Note: you may use any other cart or star image available online for your website.

Assignment Demo

Each student will have to demo their submitted assignment to their tutor during week 4's lab. Demos include showing that all requirements are implemented and working correctly. Also, every student should be able to explain and answer code-related questions as part of their demo.

A Tutorial to Load a JSON File

Before being able to read the json file, you will need to setup a simple HTTP server first. You can either use PHP or Node.js to run the simple HTTP server. Please select one of these options:

* **Using PHP**: For Mac OS or linux users, your operating system usually has PHP installed so you do not have to install it. For windows users, you will need to install PHP since it is not available by default. After having PHP installed, you first open your command line or terminal and navigate to your project directory by using "cd" command. Afterwards, you run this command: "*php -S localhost:8000*" and it should say "*Listening on http://localhost:8000*". By running this command you can open your project from your browser using this address: "*http://localhost:8000*".
* **Using Node.js**: You will be using it for upcoming tutorials and you can use it to work on your second assignment, so installing Nodejs can serve other purposes. They provide a UI installer so it is easy to install, you can go to this link: [https://nodejs.org/en/download/ (Links to an external site.)](https://nodejs.org/en/download/). After installing Node.js, open your terminal (mac/linux) or command prompt (windows) and run this command: "*npm install http-server -g*" (put sudo before npm if you are mac/linux users). If http-server installation is complete, go to your project directory by using "*cd*" command and run command "*http-server*". It will provide you url(s) to access your project from your browser, you usually can access your project from this address: "*http://127.0.0.1:8080*".

Having one of above options ready, you have to insert this function into your javascript file:

function getJsonObject(path, success, error) {  
 var xhr = new XMLHttpRequest();  
 xhr.onreadystatechange = function() {  
 if (xhr.readyState === XMLHttpRequest.DONE) {  
 if (xhr.status === 200) {  
 if (success) success(JSON.parse(xhr.responseText));  
 } else {  
 if (error) error(xhr);  
 }  
 }  
 };  
 xhr.open("GET", path, true);  
 xhr.send();  
}

Finally, to read the json file, you can call the function by something like this inside your *window.onload* method:

bookList = []; // book list container  
getJsonObject('data.json',  
 function(data) {  
 bookList = data; // store the book list into bookList  
 console.log(bookList); // print it into console (developer tools)  
 console.log(bookList[0]); // print the first book object into console   
 // here you can call methods to laod or refresh the page   
 // loadBooks() or refreshPage()  
 },  
 function(xhr) { console.error(xhr); }  
);

When you have added these lines of code into your javascript file, you can open your browser's developer tool and check the console output. Make sure you hard reload/refresh the page after adding these lines (windows/linux: "*Ctrl+F5*" and mac: "*⌘+⇧+R*")

Marking Guide

* Demonstrate that the static features of the bookstore Web page are implemented correctly and as per the requirements
  + All HTML elements and structure as per the main requirements
  + Applying similar CSS styles as per the main requirements
* Demonstrate that the dynamic features of the bookstore Web page are implemented correctly and as per the requirements
  + Search function
  + Filter function
  + Search and filter functions together
  + Add to Cart function (including quantity function)
  + "Dark mode" function
  + Error handling (inappropriate inputs or outputs)
* Code / implementation requirements
  + code quality (clean and maintainable)
  + demonstrate understanding and ability to explain and discuss your web page implementation