

<https://github.com/mumtazalee/Online-Book-Store>

# Table of Contents

[Table of Contents 2](#_Toc196526751)

[Introduction 2](#_Toc196526752)

[Overview of the Website from the User’s Perspective 3](#_Toc196526753)

[Discussion of How You Tested Your Website 7](#_Toc196526754)

[Functional Testing 7](#_Toc196526755)

[Testing for Browser Compatibility 8](#_Toc196526756)

[Testing for Mobile & Responsiveness 8](#_Toc196526757)

[Testing for Usability and Accessibility 9](#_Toc196526758)

[Bug Tracking and Fixes 9](#_Toc196526759)

[User Testing and Feedback 10](#_Toc196526760)

[Tools Used 10](#_Toc196526761)

[JavaScript Report for "WORLD OF BOOKS" 11](#_Toc196526762)

[Book Data Structure 11](#_Toc196526763)

[Search Functionality 12](#_Toc196526764)

[Display Results 13](#_Toc196526765)

[User Login System 14](#_Toc196526766)

[Wishlist Feature 15](#_Toc196526767)

[DOM Interactions & Event Handling 16](#_Toc196526768)

[Conclusion 17](#_Toc196526769)

[References 17](#_Toc196526770)

# Introduction

The "World of Books" web application project aimed to make an interesting and lively online store. My main goal was to use HTML, CSS, and JavaScript to create and start a front-end website for a store that works perfectly. Key features include a wishlist, an easy-to-use login process, the ability to connect with a user's biography through local storage (Mozilla, 2019), and the ability to look for books using filters the user sets. The project also tried to put web development theory into practice while making sure that the design was flexible (W3Schools,2024), that it was easy to use, and that users had a good time.

**The main aims of the project were:**

* To build a clean and engaging HTML interface for users.
* To make an HTML system that is both easy to use and interesting.
* By using JavaScript to make the website more interactive and respond to what users type in.
* To make a login system for local files that shows up on the main page and greets guests.
* To let you look for books by price, author, genre, language, and so on.
* To use local files so that a request can be updated on the fly and data can be saved between sessions.

# **Overview of the Website from the User’s Perspective**

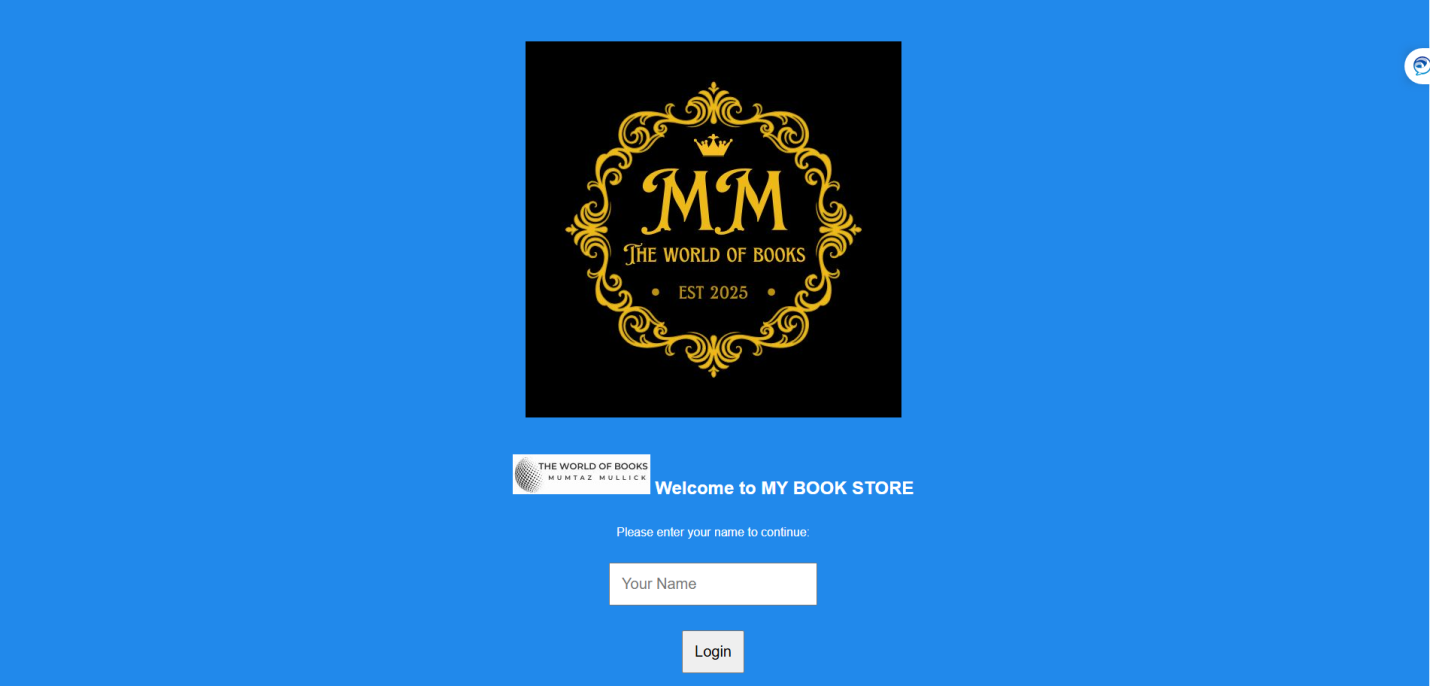
For users, the "World of Books" website starts with a simple page where they can log in. People put their name here and are taken to the home page. This is more personal because it shows a welcome word.

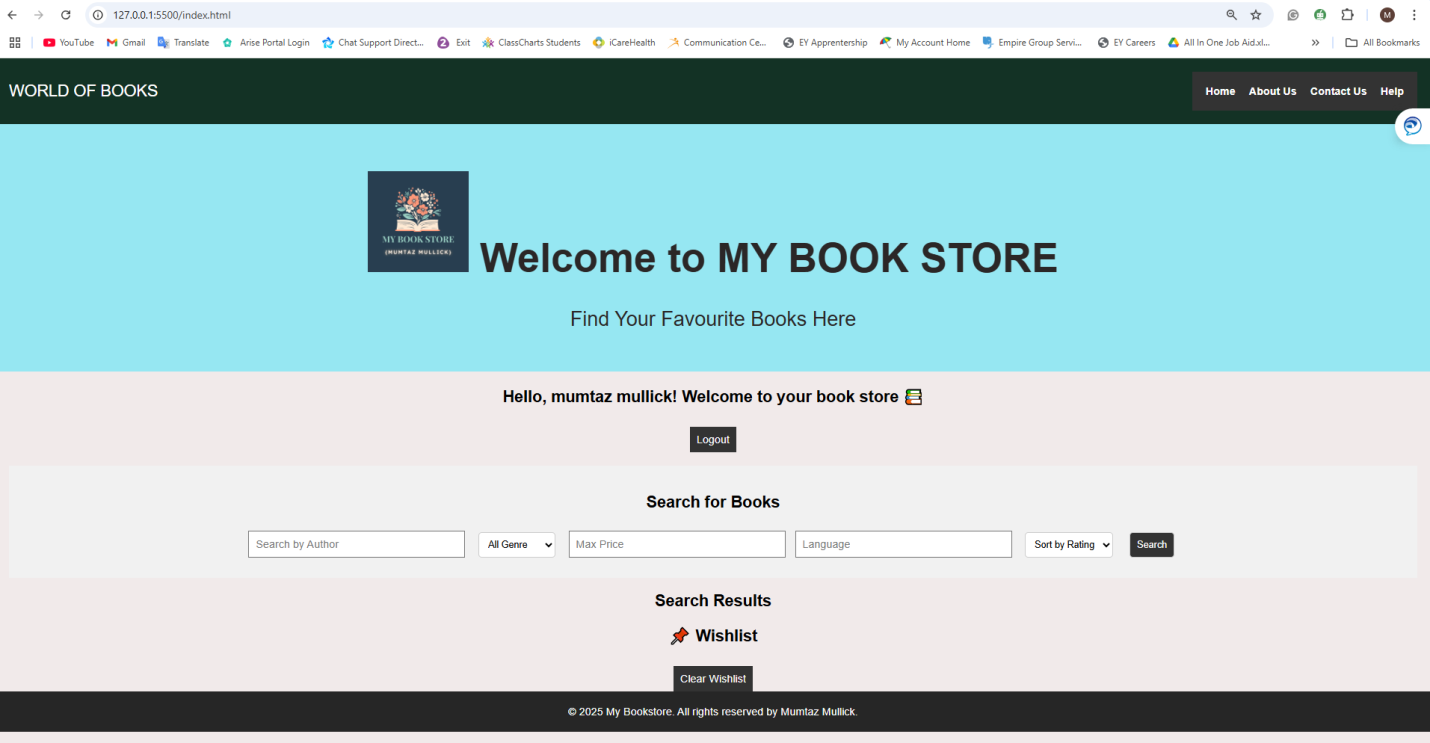
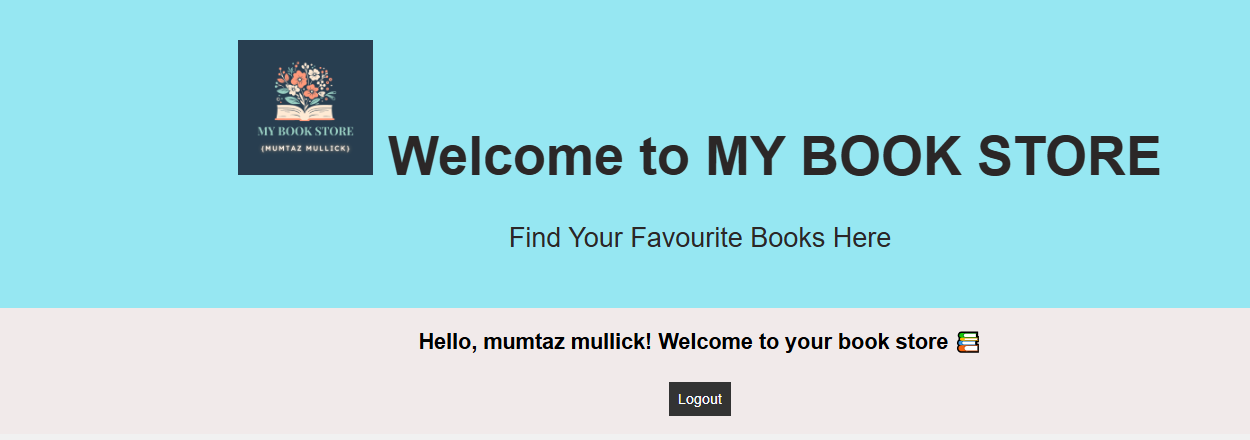
When people go to the site, they see a big area called "hero" that has the bookstore's name and slogan in bold. The main menu bar below has links to Home, Books, About Us, and Contact Us, making it easy for the user to find their way around (Duckett, 2014).  
  
The Search Section is where most of the features are. This is where users can sort books by author name, subject, price range, or language. When you click "Search," JavaScript changes the "Search Results" area so that it shows the right books without having to reload the page (Mozilla, 2020).

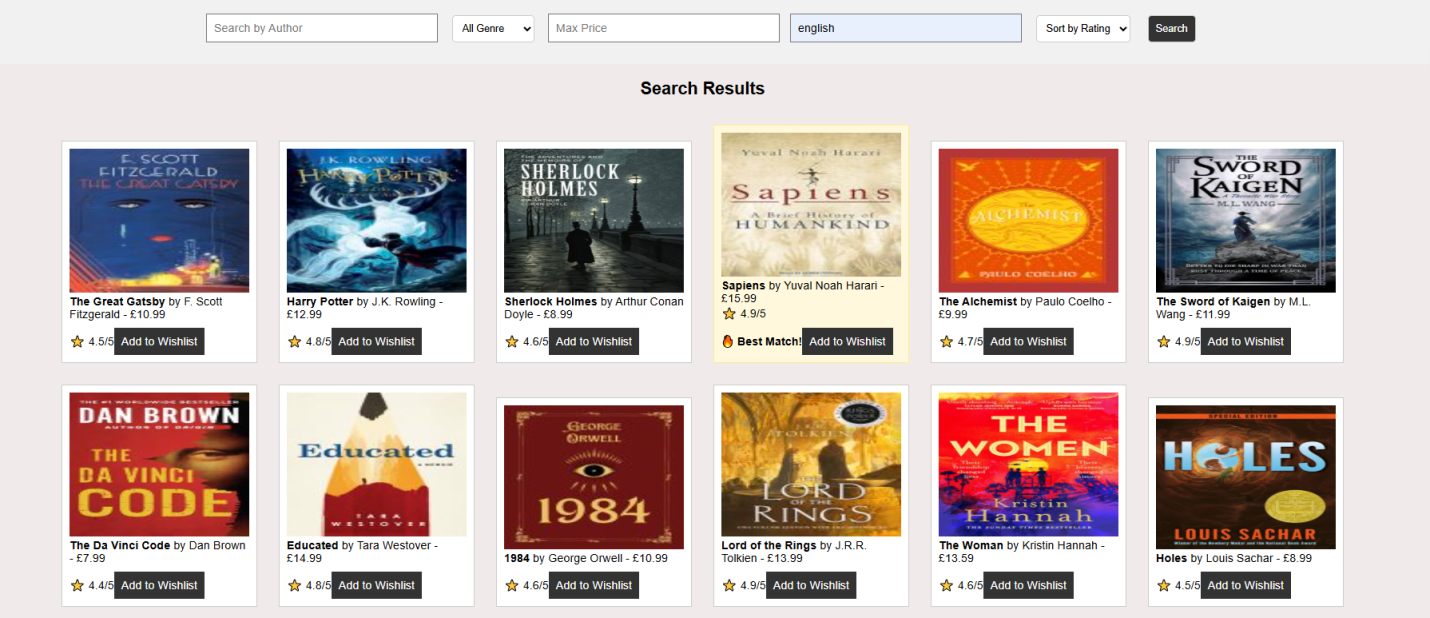
People can put books they like on the Wishlist. Book lovers can see this list of books and remove books from their wishlist at any time. The list is saved in the browser's local storage, so it stays there even after refreshing (Mozilla, 2019).

**Storyboard Summary**:

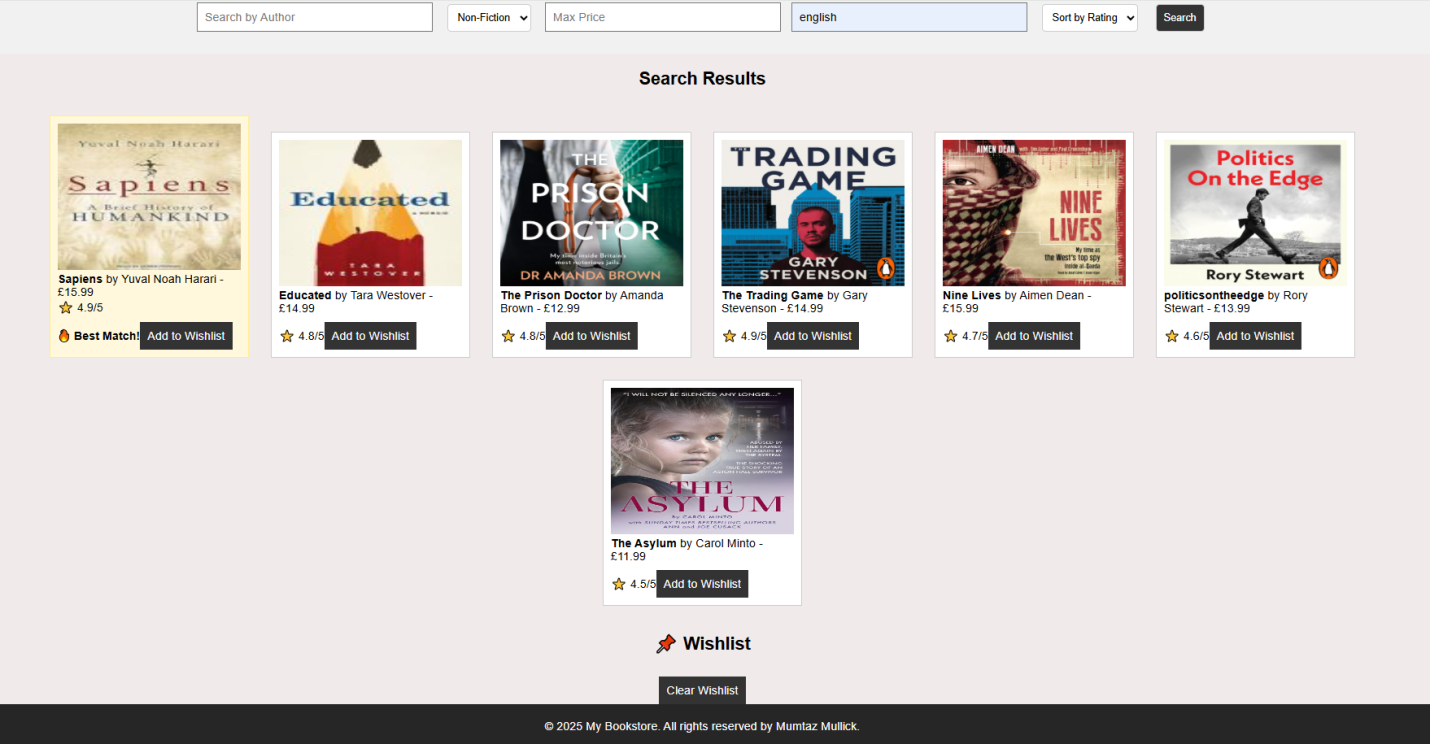
1. **Login Page** → User enters the name → Send to the homepage.



1. **Homepage** 
2. **The user gets the personal welcome and a part called “Hero”** → 
3. **Search Books** → Form to filter for the search.



1. **View Results** → Results change all the time



1. **Wishlist** → Add books, view list, clear list.



# **Discussion of How You Tested Your Website**

Making sure that the "World of Books" website worked well, was reliable, and was easy to use took a lot of testing over the course of its creation. I used different devices and websites to do different kinds of testing, such as functional, usability, and human testing (W3C, 2018). The goal was to find and fix bugs, make sure that data was handled correctly, and make the user experience better.

## **Functional Testing**

To make sure every functionality on the website operated as intended, functional testing was done:

**Testing the Login**:

To confirm that the system appropriately saves the users name in localStorage, both valid and incorrect names were entered. It has been verified that empty input stops page redirection and generates an alert message.

**Search Form**:

Author, Genre, Price, and Language were tested individually and in combination in the search form.

Invalid or nonexistent data was entered to verify that the results section appropriately handled empty states.

**Dynamic Results**:

Verified that the results changed without requiring a page reload.

To prevent repeated listings, I made sure that previous results were removed before showing new ones.

**Wishlist Functionality**:

To ensure that duplicates were avoided, many books were added to the wishlist. I used localStorage to validate that the wishlist remained active after refreshing the page. The "Clear Wishlist" button was tested to make sure it cleared the localStorage data and the on-page list.

## **Testing for Browser Compatibility**

The browsers like Google Chrome, Firefox and Edge were used to test the website to guarantee consistent behaviour across various environments (Duckett, 2014).

## **Testing for Mobile & Re**s**ponsiveness**

Prioritising responsive design helped guarantee that the bookshop will work on a variety of devices like Smart Phone, Tables and Desktops (W3Schools, 2024).

## **Testi**n**g for Usability and Accessibility**

**Navigation Test:** Every link in the navigation bar was examined to make sure it scrolled smoothly and, if necessary, redirected. Users found the material they required with ease because to the page structure.

**Form Field Testing:** Users were guided with placeholder text. There was enough contrast and padding in the input fields to make them readable.

**Colour Contrast:** To make sure the text had enough contrast with the backgrounds, it was tested using Chrome's Lighthouse feature (W3C, 2018).

## **Bug Tracking and Fixes**

Some bugs and issues and how they were resolved:

|  |  |  |
| --- | --- | --- |
| **Bug/Issue** | **How It Was Detected** | **Fix Applied** |
| Wishlist data cleared after refresh | Testing localStorage behaviour | Used JSON.parse() and getItem() during DOM Content Loaded to load wishlist |
| Login input allowed empty redirect | Manual testing | Added conditional logic with .trim() to ensure only valid names are saved |
| Search results not updating properly | During combined field searches | Added logic to clear old results before appending new ones |
| Responsive layout broke at 360px width | Mobile simulator testing | Added media queries to adjust font sizes and input widths |

## **User Testing and Feedback**

I asked two classmates to test the site without giving them any guidance. Observations included:

One classmate tried searching without clicking the button, prompting me to add form.submit functionality.

User testing revealed areas where user-centered design improvements were needed, such as adding visual feedback to buttons and submitting forms on "Enter" keypress.

## **Tools Used**

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| **Chrome DevTools** | Debugging, responsiveness, console error tracking |
| **Firefox Inspector** | CSS padding/margin tweaks |
| **W3C Validator** | HTML validation and standards compliance |
| **Lighthouse (Chrome)** | Performance, accessibility, SEO suggestions |
| **VS Code** | Primary code editors with syntax highlighting and live preview |

# **JavaScript Report for "WORLD OF BOOKS"**

This report describes how the script works and how it is organised and used in the "WORLD OF BOOKS" dynamic webpage as a .js file. The JavaScript file's job is to make the website interactive and changeable, so you can do things like look for books, log in as a user, and make your own wish list. The script uses the Document Object Model (DOM) to talk to the HTML parts on the page and local Storage to store user info.

## **Book Data Structure**

At the beginning of the script, a books array is defined. Each book is stored as an object containing key details: title, author, genre, price, language, rating, and an image path. This array acts as the database for all search and display operations throughout the site (Duckett, 2014).



## **Search Functionality**

The searchBooks() function is triggered when a user submits the search form. It filters the books array based on user input in the following fields:

* Genre
* Author
* Maximum price
* Language

All filters are case-insensitive, and partial matches are allowed for author and language. The function returns all matching books. Additionally, it finds the best-rated book in the results and highlights it as the "Best Match".



## Display Results

The displayResults() function is responsible for visually showing the filtered results in the HTML. It creates a new div element for each book and populates it with the book's image, title, author, price, and rating. If the book is the best match, a special label is added (Mozilla, 2019).



## **User Login System**

The script allows a user to log in by entering a username, which is then stored in the browser's localStorage. Once logged in, a welcome message is displayed. Logging out removes the username from storage and hides the welcome message.

The relevant functions include:

* displayUser() – Shows or hides the welcome message based on login status.
* logout() – Clears the stored username and updates the interface.

This simple login system does not require a password and is purely for personalisation and feature access, like the wishlist.



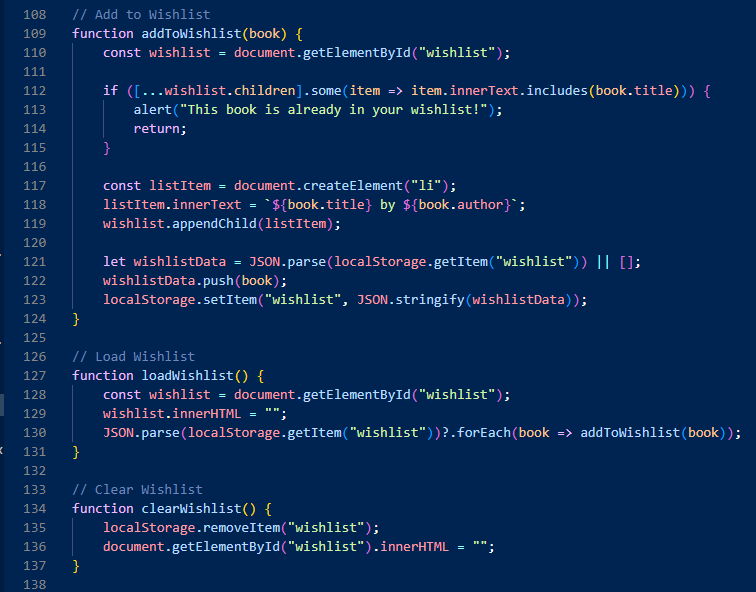
## **Wishlist Feature**

A user's wishlist is shown as a list on the page. Users can add books to it. There are two places where the chosen book is put when the "Add to Wishlist" button is clicked:

Important points:

* You can't put duplicate books on the wishlist.
* When the page loads, the load request() method gets any saved request data and adds it to the list again.

The "Clear Wishlist" button gets rid of the list from both the page and the files.

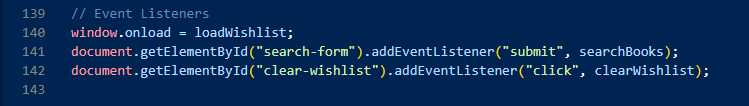


## **DOM Interactions & Event Handling**

Form entries and buttons are linked to a number of addEvent Listeners ():

* searchBooks () is called when the user sends the search form.
* As soon as the page loads, window.onload calls loadWishlist () automatically.
* The link to the "Clear Wishlist" button is in the clearWishlist () code.

These interactions make the experience smooth for the user and let changes happen in real time without having to restart the page.



# **Conclusion**

Overall, the script.js file is a key component of the "WORLD OF BOOKS" application. It adds important interactivity by enabling users to search for books, log in, and manage a wishlist. This project successfully demonstrated the power of JavaScript in creating a fully interactive, browser-based bookstore experience. Created a personalised login experience.

Through the process, I gained practical experience in working with the DOM, writing reusable JavaScript functions, and handling user data using localStorage. One of the main challenges was writing efficient and clean logic for dynamic rendering and filtering but solving those issues improved my understanding of core JavaScript concepts.

If I were to expand this project, I would add mobile responsiveness, backend integration, and user account creation using a real database.

# References

The Code and Report assessed by Mumtaz Mullick. I create a code in Visual Studio

Anon (n.d.) JavaScript Array filter() Method. [www.w3schools.com](http://www.w3schools.com).https://www.w3schools.com/jsref/jsref\_filter.asp.

Anon (n.d.) W3Schools online HTML editor. www.w3schools.com. https://www.w3schools.com/html/tryit.asp?filename=tryhtml\_basic.

Duckett, J. (2014) JavaScript and jQuery. John Wiley & Sons.

Mozilla (2019) Window.localStorage. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage.