Ideas and Objectives:

As per the requirements of the portfolio assignment, a web application was designed and formulated to implement a dynamic frontend as well as communication with a backend service. Based on my personal level of experience and skills with web development, I devised a simple web app solution for my project, with integrated API communication. This application was named 'Booklog'.

The main concept of the Booklog application is based on a standard blog application. As an avid reader myself, I wanted to adapt this concept to cater to readers in general. As such, the motivation behind Booklog was to help readers maintain a list of books they are reading or want to read, so they can easily track their reading material. Another idea that shaped the application was a digital space for writing personal reviews for a book entered in the list. Overall, Booklog was created as an easy-to-use app for blogging books with their relevant details.

Implementation Process and Design:

Originally, the idea had been to create an application that would allow users to first search for a book and then add it to a reading list with all its details pre-entered, similar to adding products to a cart. To this end, I first created a skeleton website with simple HTML inputs and tested API fetch calls with the Open Library Search API. However, based on the volume of object parameters and properties received, I considered the simplification of my original idea and decided to change my concept to the Booklog application.

Once the architecture for the new idea had been designed, I started development of the application layout using HTML. This was done to get an abstract idea of what inputs would be required and where the different elements would be placed. I then tested this layout by making API fetch calls to the Open Library Covers API. This external API communication was integrated using the fetch JavaScript function. The response to each call was received in JSON format and from this response object, the source_url property was utilized to be set as the cover image. Once the API communication was finalized, I started to style the layout using CSS. For the styling, I opted for neutral brown tones to reflect a cozy, bookish aesthetic and divided the book entry template and containers into organized sections.

The next step was adding functionality to the button elements using JavaScript. The initial focus was placed on the 'Add book' button, and after a few trials, the button worked to display a new book entry on the window with the user-entered values. This was followed by the remove, edit, and update buttons. During this, I also implemented user alerts to be displayed in the case that a book title was missing, a duplicate entry was being created, or if the API call returned anything else except a valid image URL.

Next, I started researching data storage and retrieval using databases. I first considered using MongoDB but then decided to use the browser local storage since it would allow for easy data access and other related operations from within the JavaScript code. For each previously coded button

functionality, I created relevant JavaScript functions to reflect the working in local storage i.e., the add button added a new object to a books array, the remove button deleted the object, and the update button updated the values within the relevant object. The added entries were also coded to be displayed upon page reload. I finalized the development of the application with a book counter and made several tests to ensure the complete working.

Challenges faced:

My biggest challenge during the development of Booklog was that of working with local storage. Due to a previous frontend project, I did not have too much difficulty in creating the frontend for my application, but the novelty of working with data storage called for extensive research and help on the topic. The main obstacle was trying to access the relevant book object in the array from a corresponding button. However, once I got a grasp of the main concepts through some online tutorials and videos, I tried out different techniques for the various functionalities I had to implement in local storage and was able to find a working solution for each.

Besides this, I also had to face the obstacle of not being able to immediately work with what I had initially decided for my application purpose. As such, I had to reevaluate the technologies for my project based on my knowledge and skills and also had to factor in personal time constraints.

Result self-evaluation:

The final Booklog product fully implements the envisioned and documented concept. The application is both easy-to-use and purposeful for its defined use. Although finding the right ISBN to retrieve a cover image may occasionally take some tries, the image display via an ISBN input adds an efficient and unique blog feature that functions well in the application.

A critical point to make is that the current state of the application is limited in terms of its features. Further developments could implement a search area for the added entries, together with sorting and filtering functions. The ability to create different lists with custom titles and group together books based on genre, status, or rating can also be developed.

Lessons learned:

One of my main takeaways from working on this project is that the development of software for any application, regardless of size or complexity, will always require multiple iterations, various coding techniques, searching for solutions, many phases of testing with every change, and most of all, extensive debugging. Regarding technical knowledge, I have learned quite a bit about working with APIs and local storage as well as its limitations in terms of permanent storage of data. Last but not least, I have also learnt that a developer always needs to be prepared to face all obstacles and constraints with the utmost determination to see their work to the end.