Hausaufgabe 4
DOM Manipulation, Bootstrap und Fetch

Laden Sie Ihre Lösung für die Hausaufgabe in einer ZIP-Datei bis zum 12. Januar 2025 um 23:59 Uhr auf ISIS hoch. Die ZIP-Datei darf ausschließlich den Ordner `ha04` mit dessen Inhalt gemäß der Vorgabe enthalten (die Inhalte der Dateien müssen selbstverständlich gemäß der Aufgabenstellung bearbeitet werden). Es dürfen keine weiteren Dateien hinzugefügt oder entfernt und keine Dateien oder Ordner umbenannt werden. Verspätete Abgaben werden grundsätzlich nicht akzeptiert und mit 0 Punkten bewertet.

Vorbereitung

In dieser Hausaufgabe entwickeln Sie Schritt für Schritt eine Anwendung zur Buchsuche und Anzeige von Buchinformationen. Ziel ist es, das Grundgerüst der Anwendung sowie das Styling der Komponenten mit Bootstrap zu erstellen.

Ihnen wird dazu ein rudimentäres Gerüst bestehend aus einer HTML-Datei (`index.html`) und einer JavaScript-Datei (`app.js`) zur Verfügung gestellt. Entpacken Sie das ZIP-Archiv der Vorgabe und öffnen Sie den enthaltenen Projektordner `ha04` mit einer Entwicklungsumgebung Ihrer Wahl (z. B. Visual Studio Code oder IntelliJ). Bitte Vorgabe nicht verändern.

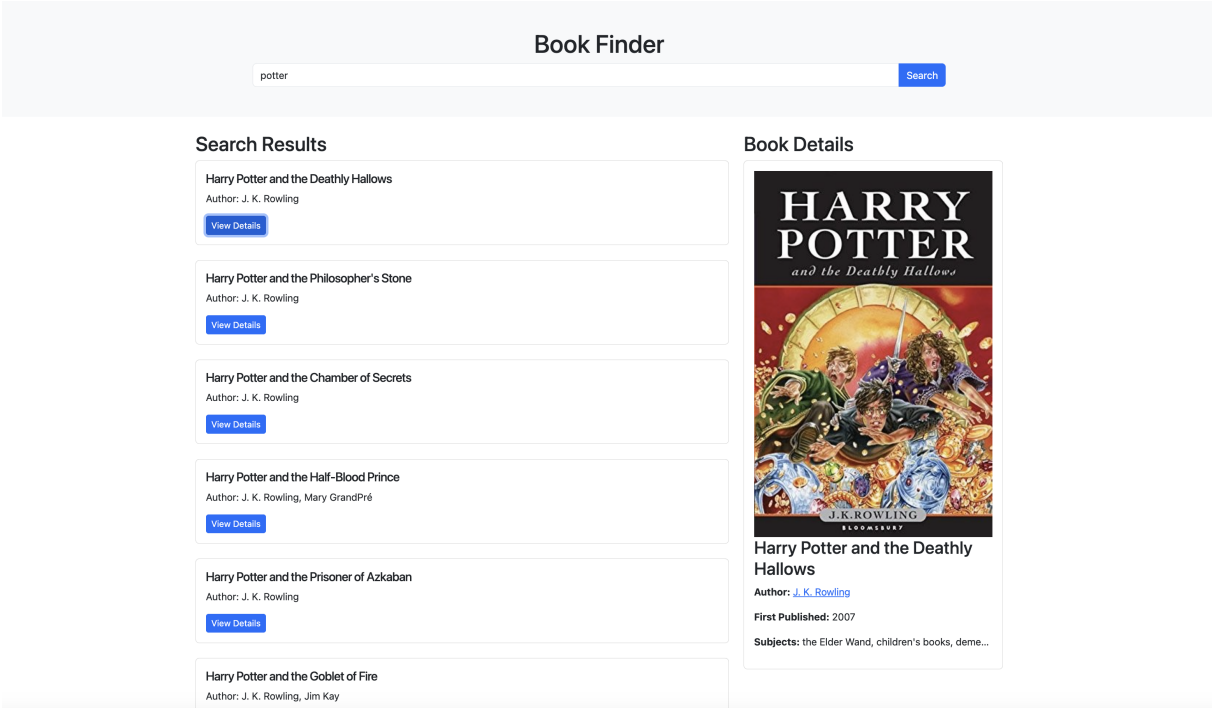

Abbildung 1: Screenshot der fertigen Anwendung.

Abbildung 1 zeigt einen Screenshot der fertigen Single-Page-Anwendung.

4.1 Bootstrap (2 Punkte)

In dieser Aufgabe erstellen wir das Grundgerüst mithilfe von Bootstrap, das wir für die DOM-Manipulation benötigen. Wie in der Abbildung 1 oben dargestellt, bauen Sie die Bereiche „Search Results“ und „Book Detail“ innerhalb des `<main>` Tags auf. Achten Sie dabei auf die Grid-Struktur von Bootstrap, die Sie in der Vorlesung kennengelernt haben. Wichtig ist, dass das Spaltenverhältnis 8 zu 4 beträgt.

1. Der Bereich „Search Results“ sollte ein Platzhalter-`<div>` mit der ID **resultsList** enthalten. Es muss eine Bootstrap-`row` sein.
2. Der Bereich „Book Detail“ muss ein Platzhalter-`<div>` mit der ID **bookDetails** enthalten. Verwenden Sie eine Bootstrap-Kartenkomponente (`card`) mit einem Padding von 3. Nutzen Sie hierfür die Bootstrap-Klasse `p-3`.

4.2 Ereignisse (4 Punkte)

In dieser Aufgabe fügen wir den Suchbutton aus dem Header Ereignisse hinzu. Implementieren Sie die folgenden zwei Ereignisse:

1. Sobald der `searchButton` per Mausklick ausgelöst wird, soll die Funktion `performSearch` aufgerufen werden (2 Punkte).
2. Die Funktion `performSearch` soll auch aufgerufen werden, wenn im Eingabefeld `searchInput` die Eingabetaste (Enter) ausgelöst wird (2 Punkte).

4.3 Fetch und DOM-Manipulation (14 Punkte)

In dieser Aufgabe suchen wir den eingegebenen Buchtitel über die OpenLibrary-API (`https://openlibrary.org/dev/docs/api/search`). Diese API liefert Antworten immer im JSON-Format. Die Ergebnisse werden genutzt, um den DOM zu manipulieren.

4.3.1 Eingabevalidierung (2 Punkte)

In der Funktion `performSearch` wird die Suchanfrage (`query`) validiert. Um die API zu entlasten, sollen leere Anfragen nicht zugelassen werden. Erweitern Sie die Funktion `validateQuery`, sodass bei leerer Anfrage ein Bootstrap-Alert ausgegeben wird. Dafür können Sie das HTML-Element mit der ID **emptyTextAlert** ein- und ausblenden.

4.3.2 Fetch API (2 Punkte)

In dieser Aufgabe rufen wir die API für Bücher mit dem gegebenen Suchtext (`query`) auf. Nutzen Sie die folgende URL innerhalb der `fetch`-Funktion: `https://openlibrary.org/search.json?title=${encodeURIComponent(query)}&fields=title,cover_i,author_name,first_publish_year,subject,author_key`. Erstellen Sie die `fetch`-Anfrage mit den Konstrukten `then`, `catch` und `finally`.

4.3.3 Fetch-Antwort (8 Punkte)

4.3.3.1 Leere Antwort (1 Punkt)

Falls die Antwort keine Bücher enthält, soll im `resultsList`-Element das folgende HTML ausgegeben werden:

```
<div class="col-12">
  <p class="text-center">No results found</p>
</div>
```

4.3.3.2 Bücher in der Antwort (4 Punkt)

Falls die Antwort Bücher enthält, erstellen Sie für jedes Buch ein HTML-Fragment. Dieses Fragment wird mit `document.createElement` erzeugt und ist in der Funktion `searchResultItem` vorgegeben. Achten Sie darauf, dass `book` ein Buch aus der Antwort darstellt.

4.3.3.3 Buchdetails (3 Punkt)

Erweitern Sie die Funktionalität, sodass bei einem Klick auf den **View Details**-Button die Details des entsprechenden Buchs im Bereich `bookDetails` angezeigt werden. Verwenden Sie das folgende HTML-Fragment für die Buchdetails:

```
<img class="responsive" src="https://covers.openlibrary.org/
   b/id/${book.cover_i}-L.jpg" alt="${book.title}"/>
<h3>${book.title}</h3>
<p><strong>Author:</strong> ${book.author_name?.join(', ')
   || 'Unknown Author'}</p>
<p><strong>First Published:</strong> ${book.
   first_publish_year || 'N/A'}</p>
<p class="d-inline-block text-truncate"><strong>Subjects:</
   strong> ${book.subject?.join(', ') || 'N/A'}</p>
```

4.3.4 Fetch Fehlerbehandlung (1 Punkt)

Implementieren Sie den `catch`-Block so, dass im Fehlerfall das folgende HTML im `resultsList`-Div ausgegeben wird:

```
<div class="col-12">
  <p class="text-center text-danger">Error fetching data</p>
</div>
```

4.3.5 Fetch Finalisierung (1 Punkt)

Im `finally`-Block blenden Sie den Spinner aus, indem Sie `toggleLoadingSpinner` mit den richtigen Parametern aufrufen.