

The background features a dark blue-grey central area. On the left, a solid lime green shape extends from the top to the bottom. On the right, a series of overlapping, semi-transparent green triangles and polygons create a dynamic, layered effect, ranging from light lime green to dark forest green.

CAPSTONE PROJECT

BOOK CONNECT

BY AVIWE KOLI

# USER EXPERIENCE TO ACHIEVE:

- ▶ As a user, I want to view a list of book previews, by title and author, so that I can discover new books to read.
- ▶ As a user, I want an image associated with all book previews so that I can recognize a book by the cover even if I forgot the name.
- ▶ As a user, I want to have the option of reading a summary of the book so that I can decide whether I want to read it.
- ▶ As a user, I want to have the option of seeing the date that a book was published so that I can determine how easy it is to obtain second-hand.
- ▶ As a user, I want to find books based on specific text phrases so that I don't need to remember the entire title of a book.
- ▶ As a user, I want to filter books by author so that I can find books to read by authors that I enjoy.
- ▶ As a user, I want to filter books by genre so that I can find books to read in genres that I enjoy.
- ▶ As a user, I want to toggle between dark and light modes so that I can use the app comfortably at night.

# PROJECT PROPS

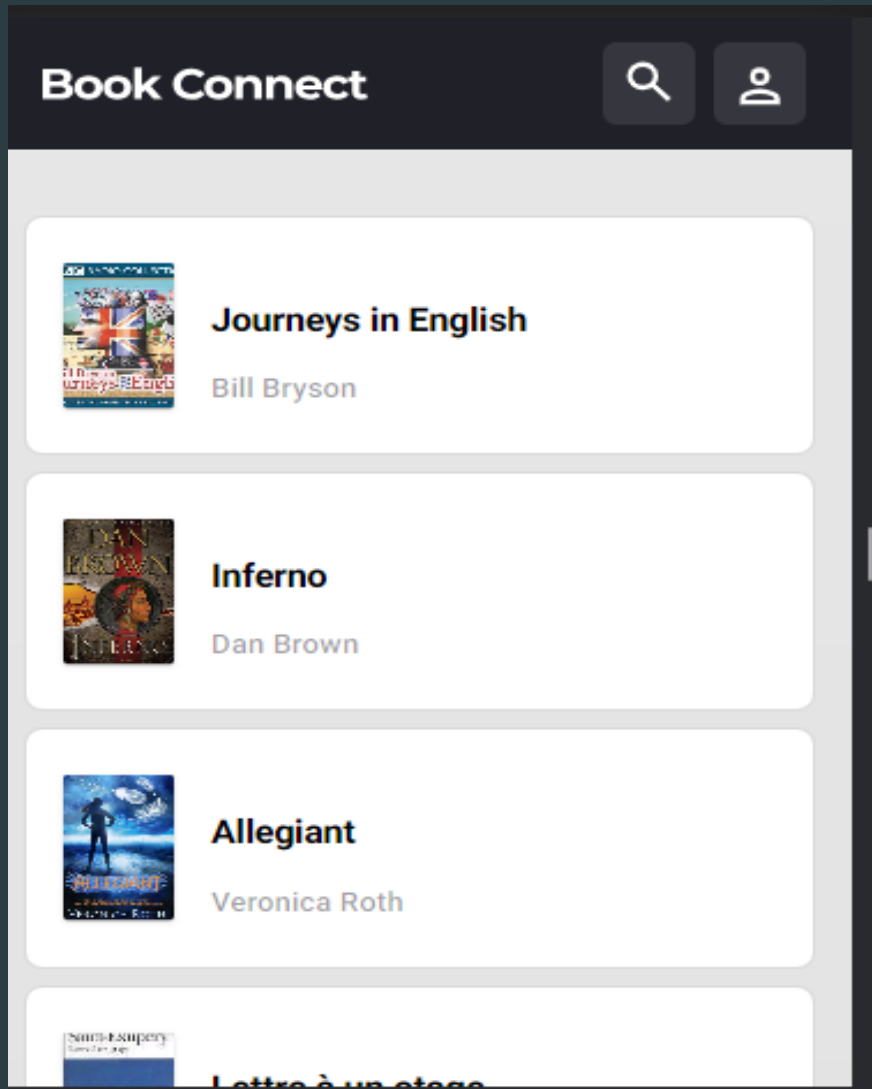
```
const html = {
  header: {
    search: document.querySelector('[data-header-search]'),
    settings: document.querySelector('[data-header-settings]') ,
  },
  main: {
    mainContainer: document.querySelector('.list'),
    list: document.querySelector('[data-list-items]'),
    message: document.querySelector('[data-list-message]'),
    button: document.querySelector('[data-list-button]'),
  },
  active: {
    overlay: document.querySelector('[data-list-active]'),
    close: document.querySelector('[data-list-close]'),
    image: document.querySelector('[data-list-image]'),
    imageBlur: document.querySelector('[data-list-blur]'),
    title: document.querySelector('[data-list-title]'),
    subtitle: document.querySelector('[data-list-subtitle]'),
    summary: document.querySelector('[data-list-description]'),
  },
  search: {
    overlay: document.querySelector('[data-search-overlay]'),
    form: document.querySelector('[data-search-form]'),
    title: document.querySelector('[data-search-title]'),
    genres: document.querySelector('[data-search-genres]'),
    authors: document.querySelector('[data-search-authors]'),
    cancel: document.querySelector('[data-search-cancel]'),
    save: document.querySelector('[form="search"]')
  },
  settings: {
```

```
    settings: {
      overlay: document.querySelector('[data-settings-overlay]'),
      form: document.querySelector('[data-settings-form]'),
      cancel: document.querySelector('[data-settings-cancel]'),
      save: document.querySelector('[form="settings"]'),
      theme: document.querySelector('[data-settings-theme]'),
    },
  }
}
```

```
import {BOOKS_PER_PAGE, authors, genres, books} from './data.js';

/**
 * CONTANTS USED IN THE APP
 */
let page = 1;
const css = {
  day: ['255, 255, 255', '10, 10, 20'],
  night: ['10, 10, 20', '255, 255, 255'],
}
/**
```

## Initial Rendering of the page:



```
const renderPreview = (booksToShow) => {
  const {main: {list, button}} = html;
  const fragment = document.createDocumentFragment();

  for (const book of booksToShow){
    const image = book.image;
    const title = book.title;
    const author = book.author;
    const id = book.id;

    const bookList = document.createElement('button');
    bookList.classList = 'preview';
    bookList.setAttribute('id', `${id}`);

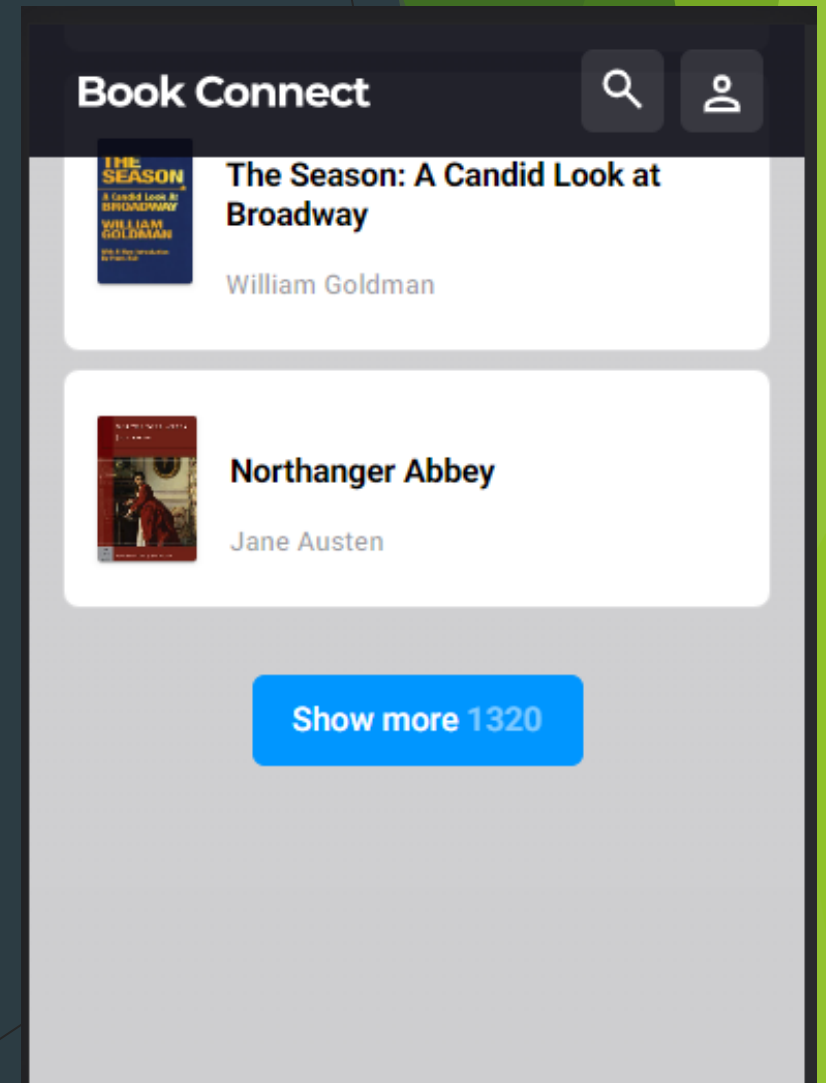
    bookList.innerHTML = `
      <img class = "preview__image"
        src = ${image}
      />
      <div class="preview__info">
        <h3 class="preview_title">${title}</h3>
        <div class="preview__author">${author}</div>
      </div>`;

    fragment.appendChild(bookList);
    list.appendChild(fragment);
  }
  return list;
}
```

# Show More Button:

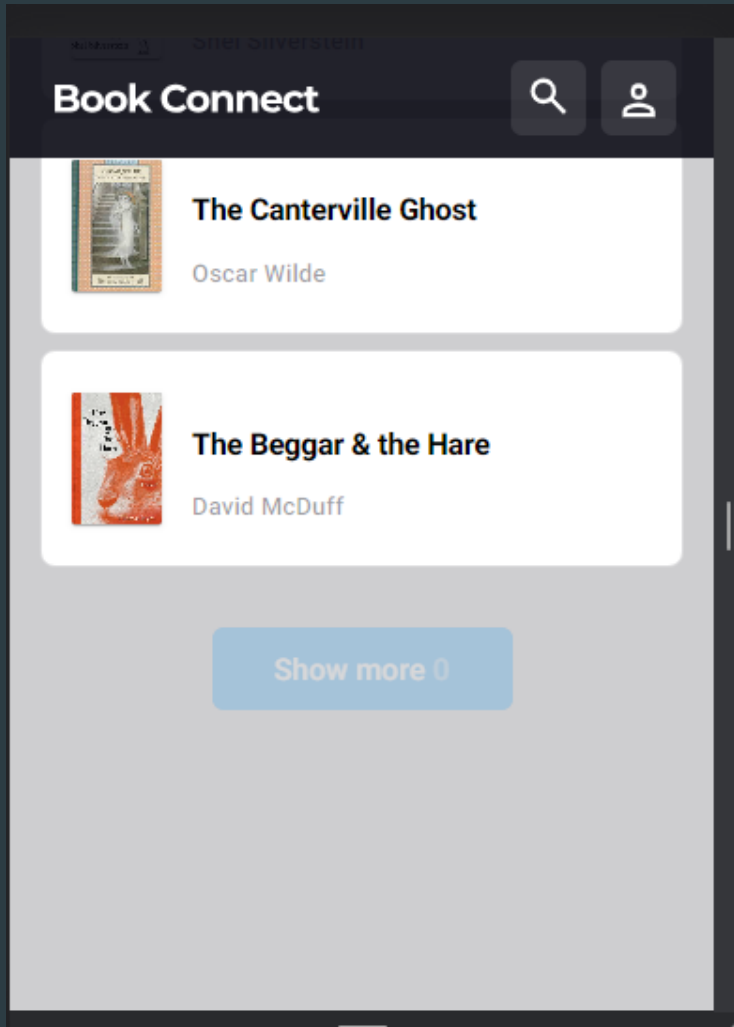
- ▶ The show more button allows the user to render or load more book into the page and display the number of books remaining

```
const updateShowMoreButton = () => {  
  const remainingBooks = bookArray.length - (page* BOOKS_PER_PAGE);  
  html.main.button.disabled = remainingBooks <= 0;  
  html.main.button.innerHTML = /* html */`  
    <span>Show more</span>  
    <span class="list__remaining">${remainingBooks > 0 ? remainingBooks: 0}</span>`;  
}
```



# After loading all books:

When all the books have been loaded, the button is disabled



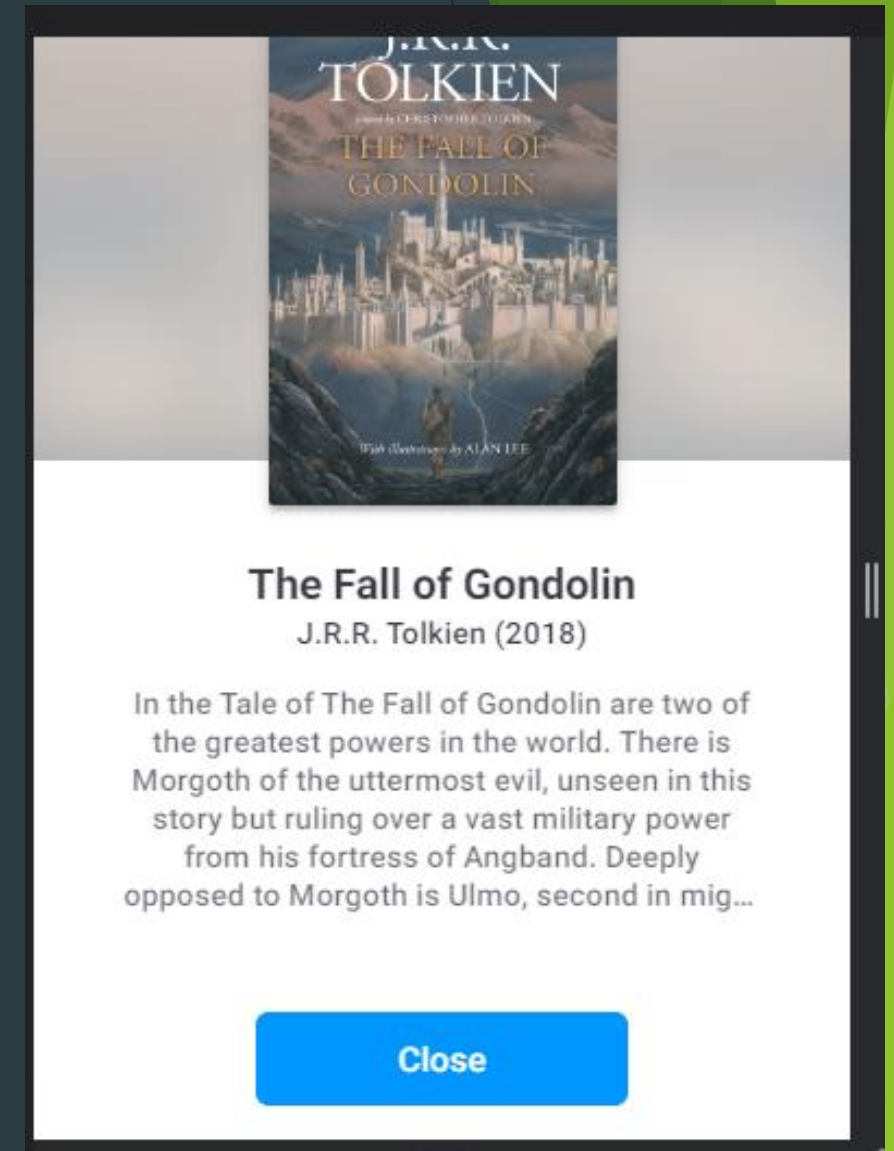
```
*/  
const handleShowMoreClick = () =>{  
  page = page +1;  
  renderPreview(bookArray.slice(((page -1) *BOOKS_PER_PAGE), (page*BOOKS_PER_PAGE)));  
  updateShowMoreButton();  
}
```

# Book Preview:

- ▶ A book preview which displays the book details:

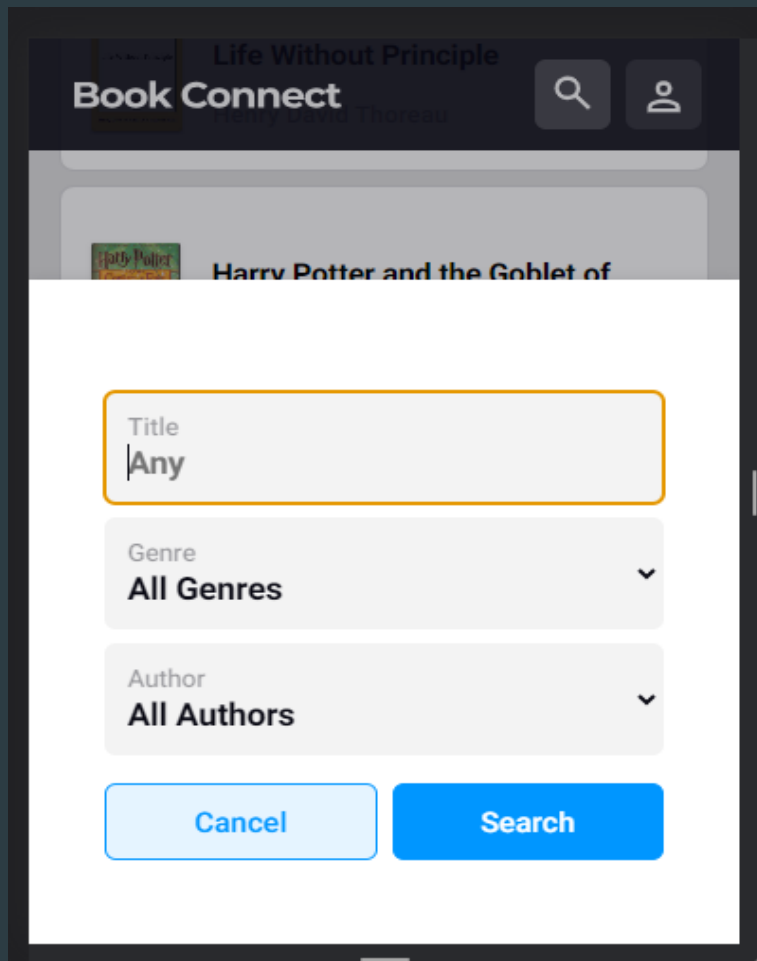
```
function activeHandle (event){
  const {active: {overlay, image, imageBlur, title, summary, subtitle, close}} = html;

  const bookElement = event.target.closest('.preview').id
  let currentBook = []
  for (const book of bookArray) {
    if (bookElement === book.id){
      currentBook = book;
    }
  }
  title.innerHTML = currentBook.title;
  subtitle.innerHTML = `${currentBook.author} (${new Date((currentBook.published)).getFullYear()})`;
  summary.innerHTML = currentBook.summary;
  image.src = currentBook.image;
  imageBlur.src = currentBook.image;
  overlay.show();
  close.addEventListener('click', function(event){
    event.preventDefault();
    overlay.close();
  })
}
```



## Book filter/ Search:

- ▶ A dialog pop-up or overlay that display when the search button is clicked



```
const handlerSearch = (event) => {  
  const {search: {overlay, form, cancel}} = html;  
  overlay.show();  
  cancel.addEventListener('click', function(){  
    overlay.close();  
    form.reset();  
  })  
}
```

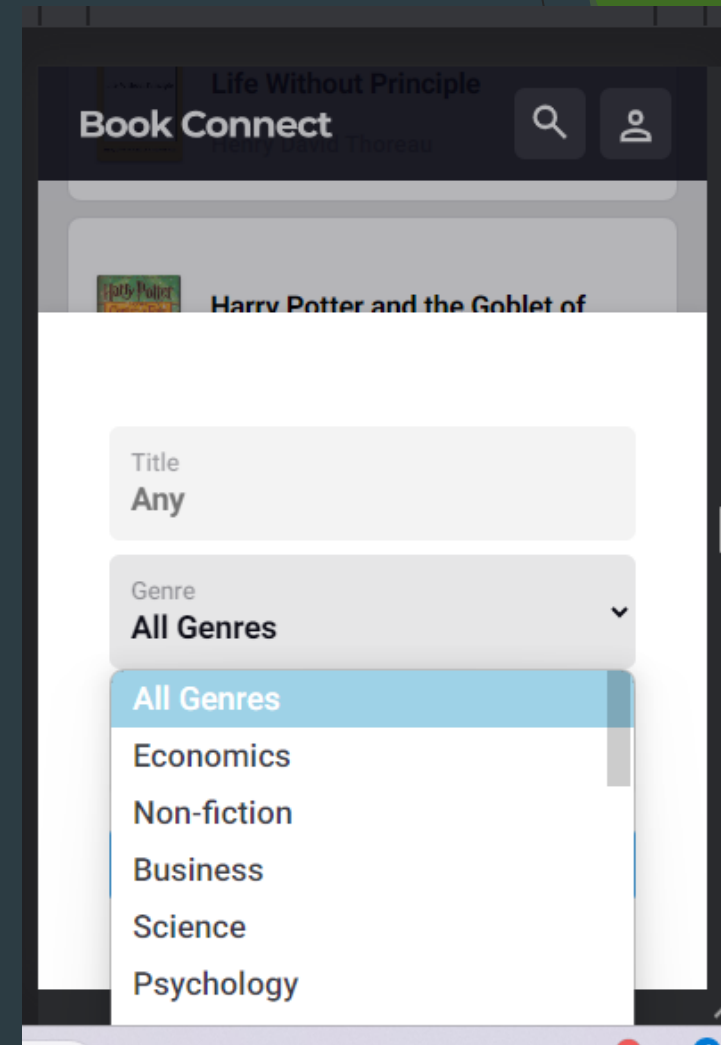


# Genres Options:

- Function that populates the DOM with genre options

```
const createGenresHtmlOptions = () =>{
  const fragment = document.createDocumentFragment()
  const genresArray = ['All Genres'].concat(Object.values(genres))
  const {search: {genres: genresElement}} = html;
  for (const genre of genresArray) {
    const option = document.createElement('option')
    option.value = genre;
    option.innerHTML = genre;
    fragment.appendChild(option)
  }
  genresElement.appendChild(fragment);
  return fragment
}
createGenresHtmlOptions();
```

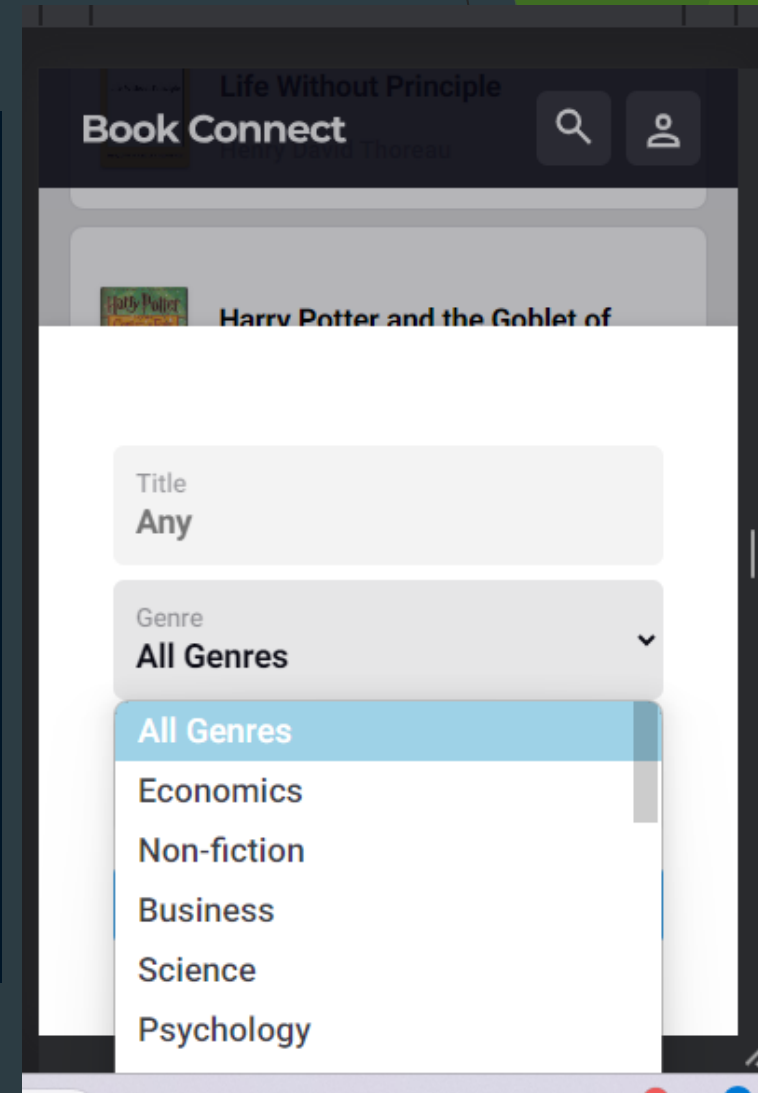
You, 2 hours ago • Added IWA19 gitHub



# Author options:

- A code snippet that populates the DOM with Author Options

```
const createAuthorsHtmlOptions = () => {  
  const authorsArray = ['All Authors'].concat(Object.values(authors));  
  const fragment = document.createDocumentFragment();  
  const {search: {authors: authorsElement}} = html;  
  for (const author of authorsArray){  
    const option = document.createElement('option');  
    option.value = author;  
    option.innerHTML = author;  
    fragment.appendChild(option);  
  }  
  authorsElement.appendChild(fragment);  
  return fragment;  
}  
createAuthorsHtmlOptions();
```



# Search submit:

```
const handlerSearchSave = (event) => {
  const {search: {overlay, title, form, authors, genres:genresValue}} = html;
  const {main: {list}} = html;
  event.preventDefault();
  const titleFilter = html.search.title.value.toLowerCase();
  const authorFilter = authors.value;
  const genreFilter = genresValue.value;

  const searchedBooks = bookArray.filter((book) =>{
    let genreArray = [];
    for (let i=0; i < book.genre.length; i++){
      let genreItem1 = '';
      for (const genreItem of book.genre){
        genreItem1 = genres[genreItem];
      }
      genreArray.push(genreItem1);
    }

    const titleMatch = book.title.toLowerCase().includes(titleFilter);
    const authorMatch = authorFilter === 'All Authors' || book.author.includes(authorFilter);
    const genreMatch = genreFilter === 'All Genres' || genreArray.includes(genreFilter);
    return titleMatch && authorMatch && genreMatch;
  })

  if (searchedBooks.length < 1){
    overlay.close();
    form.reset();
    const allBook = renderPreview(bookArray);
    const a = allBook.querySelectorAll('button')
    for (const book of a){
      html.main.button.style.display = 'none'
      book.classList.replace('preview', 'preview_hidden');
    }
  }
```

```
form.reset();
const allBook = renderPreview(bookArray);
const a = allBook.querySelectorAll('button')
for (const book of a){
  html.main.button.style.display = 'none'
  book.classList.replace('preview', 'preview_hidden');
}
html.main.button.style.display = 'none'
html.main.message.style.display = 'block';
html.header.search.addEventListener('click', function (event){
  html.main.message.style.display = 'none';
});

}else{
  const allBook = renderPreview(bookArray);
  const a = allBook.querySelectorAll('button')
  for (const book of a){
    html.main.button.style.display = 'none'
    book.classList.replace('preview', 'preview_hidden');
  }

  for (let i = 0; i < searchedBooks.length; i++) {
    const bookId = searchedBooks[i].id;
    const matchingBook = document.getElementById(`_${bookId}`);
    if (matchingBook) {
      matchingBook.classList.replace('preview_hidden', 'preview');
    }
    overlay.close();
    form.reset();
  }
}
```

# Search Example:

Book Connect

Title

Harry

Genre

All Genres


Author

All Authors

Cancel


Search

Book Connect




Harry Potter and the Goblet of Fire

Mary GrandPré



Harry Potter Boxed Set, Books 1-5

Mary GrandPré

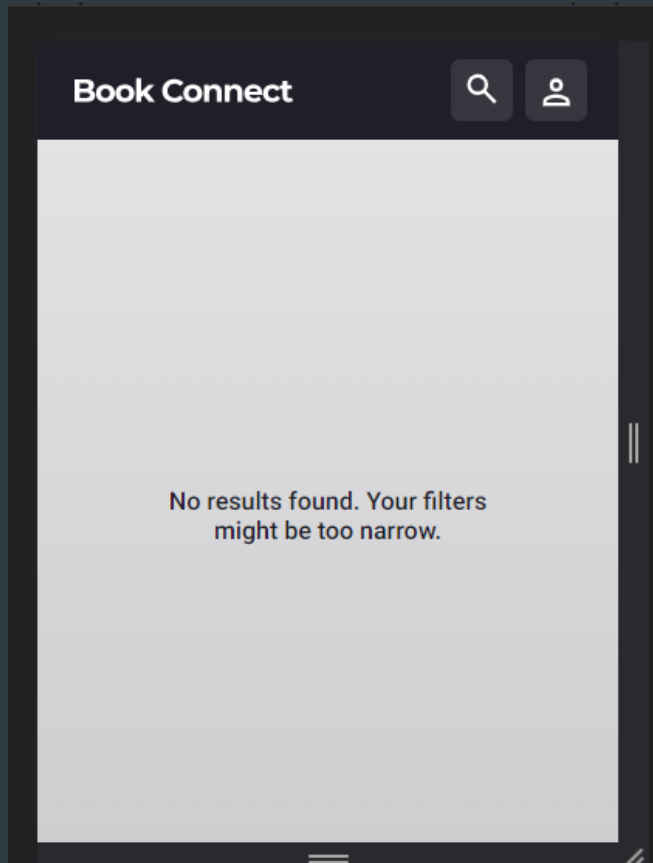


Harry Potter and the Order of the Phoenix

J.K. Rowling

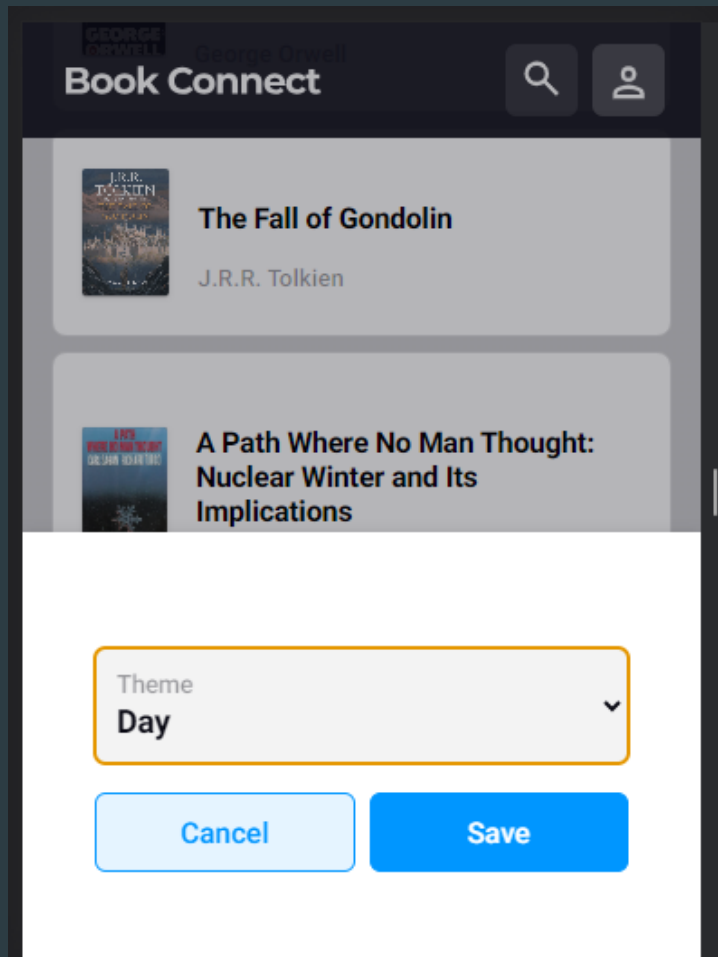
## Search Example Part2

- ▶ This shows what happens when no matches are found:

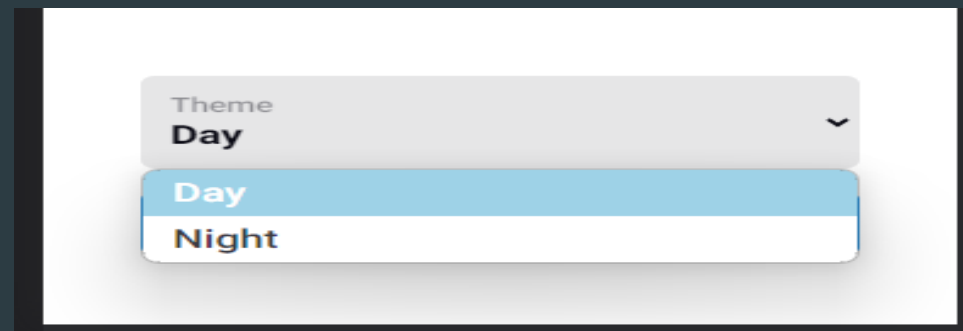


## Theme Settings:

- ▶ The dialog overlay that pops up that allows the user to toggle between
- ▶ Dark and light mode settings.



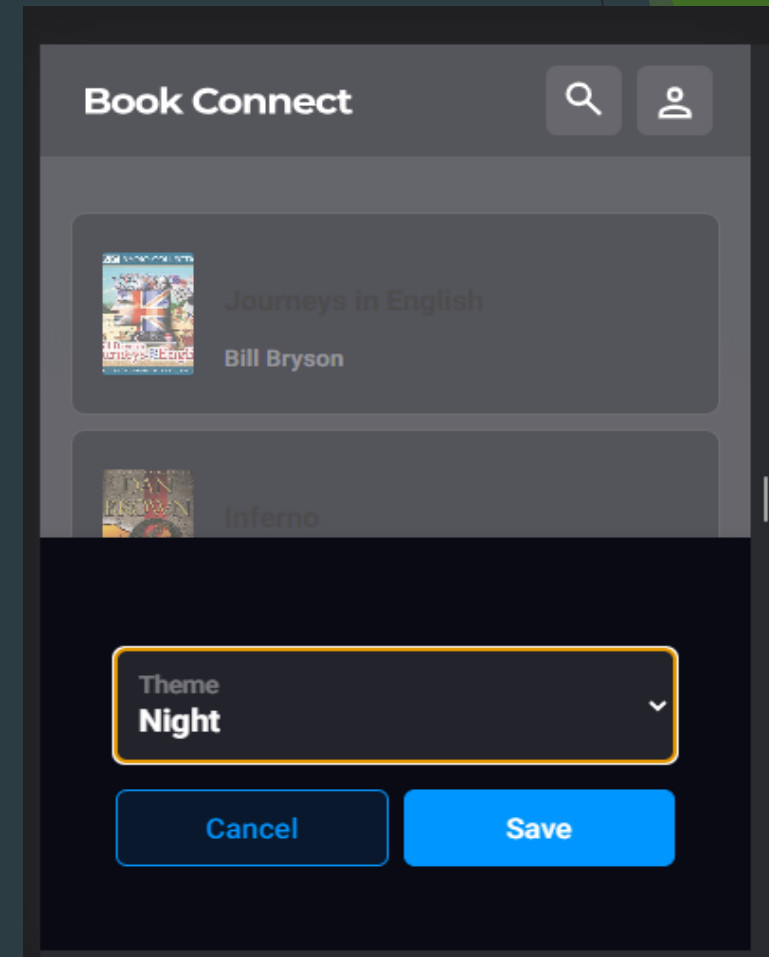
```
*/  
function handlerSettings(event){  
  const {settings: {overlay, cancel}} = html;  
  overlay.show();  
  
  cancel.addEventListener('click', function(){  
    overlay.close();  
  })  
}
```



# Applying theme Settings:

- A code snippet of the function that applies a theme chosen by the user

```
const handlerSaveSettings = (event) => {  
  // You, 1 second ago • Uncommitted changes  
  const {settings: {overlay, theme, save}} = html;  
  event.preventDefault();  
  const selectedTheme = theme.value;  
  
  settingsTheme.value = window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches ? 'night' : 'day'  
  
  if (selectedTheme === 'night'){  
    document.documentElement.style.setProperty('--color-light', `rgb(${css[selectedTheme][0]})`);  
    document.documentElement.style.setProperty('--color-dark', `rgb(${css[selectedTheme][1]})`);  
  }  
  
  if (selectedTheme === 'day'){  
    document.documentElement.style.setProperty('--color-light', `rgb(${css[selectedTheme][0]})`);  
    document.documentElement.style.setProperty('--color-dark', `rgb(${css[selectedTheme][1]})`);  
  }  
  
  overlay.close();  
}
```



**THANK YOU!**