DMIT2008 Assignment 4b: Routing in React, Pages, and Next.js

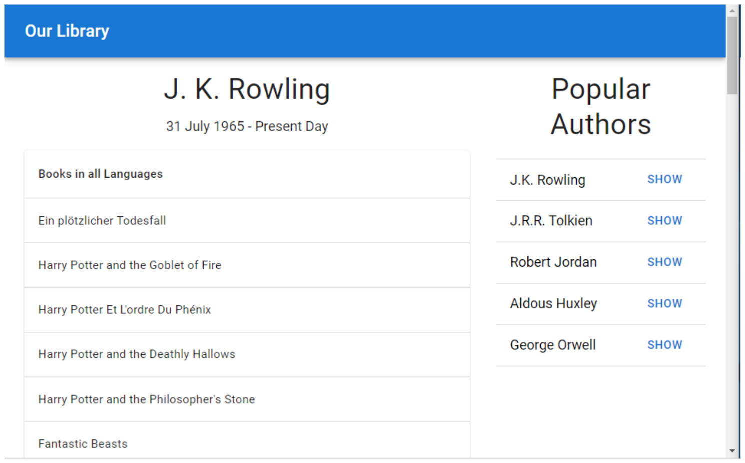
Introduction

This assignment will be a continuation of assignment 4A, where we built out our library application a bit more. This assignment will test everything that you've learned so far including the section "Next.js Pages, Routing and React". Why is this important to learn? Every site/application you go has different pages, that needs different data. In this assignment we're going to create a page so that we can display covers of the books (if they have any).

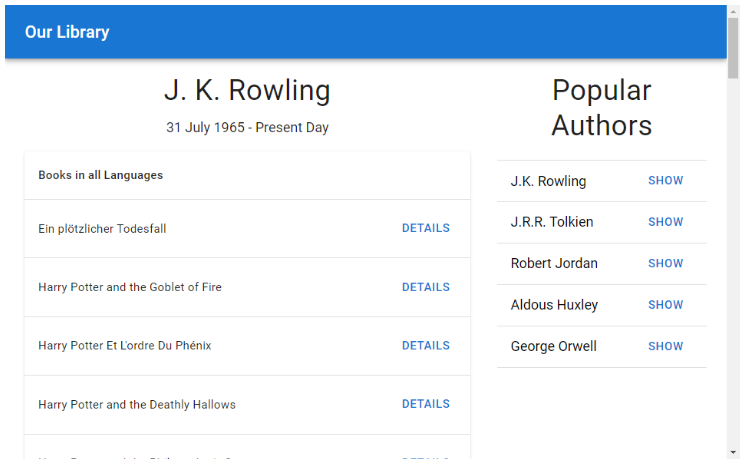
This assignment will be using the Internet Archives Open Library books api endpoint you can access here https://openlibrary.org/dev/docs/api/books, and the Covers api endpoint you can access here https://openlibrary.org/dev/docs/api/covers.

Overview of functionality

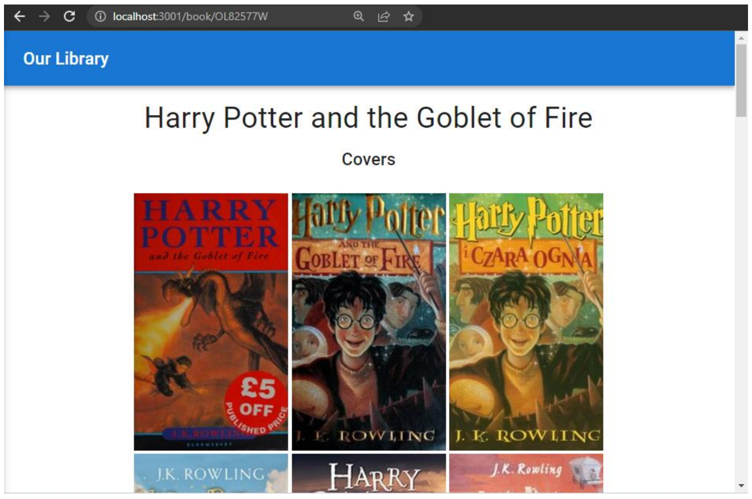
- The project's index.js (with the path "/") starter looks like the image below once you install all of the dependencies and run the next.js project (and wait for the author to load)


Our Library
J. K. Rowling
31 July 1965 - Present Day
Books in all Languages
Ein plötzlicher Todesfall
Harry Potter and the Goblet of Fire
Harry Potter Et Lordre Du Phénix
Harry Potter and the Deathly Hallows
Harry Potter and the Philosopher's Stone
Fantastic Beasts
Popular Authors
J.K. Rowling SHOW
J.R.R. Tolkien SHOW
Robert Jordan SHOW
Aldous Huxley SHOW
George Orwell SHOW

- Once your index page is complete, it should look like the image below, where the "details" button should link you to the proper book (see image below)


Our Library
J. K. Rowling
31 July 1965 - Present Day
Books in all Languages
Ein plötzlicher Todesfall DETAILS
Harry Potter and the Goblet of Fire DETAILS
Harry Potter Et Lordre Du Phénix DETAILS
Harry Potter and the Deathly Hallows DETAILS
Popular Authors
J.K. Rowling SHOW
J.R.R. Tolkien SHOW
Robert Jordan SHOW
Aldous Huxley SHOW
George Orwell SHOW

- Once you click one of the favourite authors books "DETAILS" button (here we clicked "harry potter and the goblet of fire" as an example) the new page is rendered with the url "/book/**bookIdHere**" where in this example **bookIdHere** is OL82577W



localhost:3001/book/OL82577W

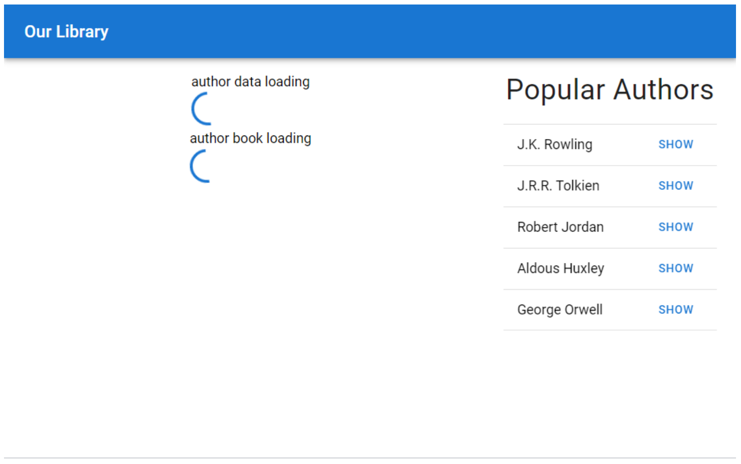
Our Library

Harry Potter and the Goblet of Fire

Covers

Note: this has to be a separate page using a "Dynamic Route".

- Bonus: This Data takes long to load so use a circular progress bar (https://mui.com/material-ui/react-progress/) use a stateful Boolean to wait to load the data (for each request as well)



Our Library

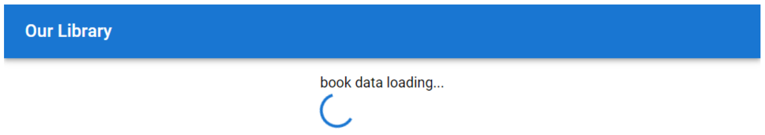
author data loading

author book loading

Popular Authors

J.K. Rowling	SHOW
J.R.R. Tolkien	SHOW
Robert Jordan	SHOW
Aldous Huxley	SHOW
George Orwell	SHOW

Do the same for the book page so it looks like this when it's loading.



Our Library

book data loading...

Required Tasks

- Install and run the starter project.
- Create a link back to the Home page in the Navbar.
    - Using the Link from next.js create the functionality so that if you're on any other page the navbar will link back to "/".
      Note: Later on when you have your "book" page you can click it and see if it works correctly (it should navigate to the index.js page)
- In the Book Table Add a "TableCell" with a "Button" that navigates to the "book" page.
    - When the user clicks the button the site is navigated to "/book/**SOMEWORKID**" where **SOMEWORKID** will be the id from "key" of the bookData object. You'll need to use the Next Router to do so (docs here)
      Note: you'll have to get just the id out of the value of the key: "/works/**SOMEWORKID**".
- Create a "book" page using Dynamic Routes (with a param of **bookId**) that will have the path "/book/**bookId**".
    - Using the Next Router get the "**bookId**".
    - Using "useEffect" and the a dependency array that listens to changes in the "**bookId**" variable load the book data from the open archive.
    - You'll use the endpoint https://openlibrary.org/dev/docs/api/books to get the data for the specific book.
    - In the "book" page using the Image List component (https://mui.com/material-ui/react-image-list/#main-content) load each image using the **Medium Covers** of the books.
        - Here's a short eyou'll be using the api with the following say you have a coverId of 10521270 your url that you'll use for the image is as follows.
          https://covers.openlibrary.org/b/id/10521270-M.jpg
          More Details on how to fetch the books are
          https://openlibrary.org/dev/docs/api/covers.

          Note: the cover API **might be a bit slow** so be patient, and just refresh the page or wait a few moments if you get a code of 502 bad gateway or 503 service unavailable.
    - Using all of your knowledge of the course so far and the examples you'll be expected to build the page and JSX so that it looks like the image in the "Overview" of functionality section.
        - It is suggested that you use MUI components such as "Container", "Grid", "Typography", "ImageList" and "ImageListItem" (as well as your normal html pieces"
- Bonus:
    - Create a Circular Loading component that will take a "text" prop. Here's the docs for MUI circular loading. (https://mui.com/material-ui/react-progress/)
    - Create stateful variables that will either display the Loading component with the text or the JSX if the endpoint was loaded successfully.
    - Refer to the "Overview of functionality" section for look at and feel.

Marking key

Tasks	Grade	Marks	Total
Link Component Used in the Navbar • Link Component used in the navbar correctly and links back to the "/" path.		**3**	
BookTable Component and Routing to Book page • useRouter hook imported properly into component and initialized properly in the component. • Table Cell and Button created in correct location and button's onClick properly navigates to books page using the router and the correct book id. Note: Functionality should look like the sample functionality images.		**1** **5**	
Book Page Component • useRouter hook imported properly into component and initialized properly in the component. • The param bookId successfully initialized from the next router. • "getBook" function uses correct endpoint (and it's own file), takes one "bookId" parameter defined and exported properly. • getBook imported properly in the page. • useEffect used with correct dependency array. The "getBook" function used correctly sets the stateful value of booksData (or whatever its' called) • The ImageList is rendered with the book covers • JSX of the book id page looks somewhat like the "overview of functionality" picture Note: Functionality should look like the sample functionality images.		**1** **1** **3** **1** **3** **5** **3**	
Loading Bonus • Loading Component created with MUI Circular progress component and used properly.		**[3]**	
Formatting and Style • Code Formatting and Style (run "npm run lint") and fix any errors that come up. • Errors when running the project.		**-3** **-3**	

Marking Rubric

Marks	5 Marks Criteria
5	**Task was completed with the highest of proficiency adhering to best practices and followed subject matter guidelines all tasks were completed to a professional standard.**
4	**Task was completed well some minor mistakes. Well above average work shows good understanding of the task and high degree of competence**
3	**Satisfactory work some features missing or incorrectly implemented. Show a moderate level of understanding in the task with room for improvement.**
2	**Below average work. Task was poorly complete. Show understanding of the task and the requirements to implement but implementation was poorly executed.**
1	**Some of the task was completed. Showed a lack of understanding in the subject matter and very poorly executed**
0	**Not completed.**

Marks	**3 Marks Criteria**
3	**Proficient shows a high degree of competence in completing task.**
2	**Capable above average degree of competence in completing task**
1	**Satisfactory shows a satisfactory degree of competence in completing task.**
0	**Shows a limited degree of competence in completing task.**

Marks	**1 Marks Criteria**
1	**Task Completed satisfactorily**
0	**Task was not executed.**