BOOKIFY

List of React components:

1. Navbar:
    a. Functionality: A navigation bar component to navigate between different books
    b. Interactions: Display links to various pages (ex: home), book listings, shopping cart, and checkout. Respond to user clicks to redirect to corresponding routes
2. Home:
    a. Functionality: The landing page of Bookify
    b. Interactions: May display content or updates
3. Book Listing:
    a. Functionality: Display a grid of books available
    b. Interactions: Allow searching for books and display book titles similar to the search functionality. Users can click on a book to view information.
4. Book Details:
    a. Functionality: Display detailed information about book
    b. Interactions: Show book detail (title, author, description, and cover image)
5. User Authentication:
    a. Functionality: Allow users to register and sign in
    b. Interactions: Users can sign in and log in and determines access to certain features
6. Shopping Cart:
    a. Functionality: Enables users to add and remove books from their shopping cart
    b. Interactions: Display the added books and calculate the total price of items in cart
7. Checkout:
    a. Functionality: Facilitate the checkout process for users to complete their purchases
    b. Interactions: Gather shipping information, handle payment options, and process the order
8. User Bookshelf:

    a. Functionality: Allow users to add and remove books from their personal bookshelf
    b. Interactions: Display the user's bookshelf with added books and provide options to remove books

Functions and Logic:

1. User Authentication Functions:
    a. registerUser: handle user registration process, including validating user input, passwords, and storing user details in database
    b. loginUser: handle user login, verify credentials, and issue authentication tokens upon login being successful
    c. logoutUser: handle user logout, invalidating the authentication token and ending the user's session
    d. getBooks: retrieve a list of books from the database and return them for display
    e. searchBooks: take a search query as input and find books that match the query. It will return the search results for display
2. Shopping Cart Functions:
    a. addToCart: add selected books to the user's shopping cart. It will manage the cart state, update the cart contents, and calculate the total price of items in cart
    b. removeFromCart: remove a book from the user's shopping cart and update the cart state accordingly
3. Checkout Functions:
    a. processOrder: handle the processing of the order including validating the cart, calculating the total order amount, and initiating the payment process
    b. handlePayment: integrate with the payment gateway API to securely process the user's payment and complete the order
    c. addToBookshelf: add selected books to the user's bookshelf in the database, associating them with the user's profile
    d. removeFromBookshelf: remove a book from the user's bookshelf in the database
4. API Integration Functions:

a. fetchBooksFromAPI: handle API requests to fetch book data from an external database and update the application's state with retrieved information
b. sendOrderToAPI: handle API requests to send the order details to the backend for processing and storing in the database

API Request-Response Formats AND Endpoint Routes:

1. User Registration API Request & Response:
    a. Route C - register
    b. Method: Post
    c. Request:
        i.
```
{
  "username": "example_user",
  "email": "user@example.com",
  "password": "password123"
}
```
    d. Response:
        i. Message is successful or an error message if registration fails
2. User Login API Request & Response:
    a. Route R - login
    b. Method: Get
    c. Request:
        i.
```
{
  "email": "user@example.com",
  "password": "password123"
}
```
    d. Response:
        i. An authentication token if login is successful, or an error message if login fails
3. Get Books API Request & Response:
    a. Route R - get books
    b. Method: Get
    c. Response:

i.

```
{
  "books": [
    {
      "id": 1,
      "title": "Book Title 1",
      "author": "Author 1",
      "description": "Description of Book 1",
      "coverImage": "http://example.com/book1.jpg",
      "price": 29.99
    },
```

4. Search Books API Request & Response:
    a. Route R - search
    b. Method: Get
    c. Request:

i.

```
{
  "searchQuery": "book title"
}
```

    d. Response:

i.

```
{
  "results": [
    {
      "id": 1,
      "title": "Book Title 1",
      "author": "Author 1",
      "description": "Description of Book 1",
      "coverImage": "http://example.com/book1.jpg",
      "price": 29.99
    },
```

5. Shopping Cart API Request & Response:
    a. Route U - shopping cart
    b. Method: Put/Post
    c. Request:

i.

```
{
  "userId": "1234567890",
  "books": [
    {
      "bookId": 1,
      "quantity": 2
    },
    {
      "bookId": 3,
      "quantity": 1
    }
  ]
}
```

d. Response:

i.

```
{
  "message": "Successfully updated the shopping cart."
}
```

6. Checkout Cart API Request & Response:
   a. Route C - checkout
   b. Method: Post
   c. Request:

i.

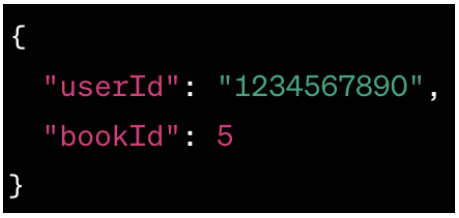
```
{
  "userId": "1234567890",
  "cartId": "abcd1234",
  "totalAmount": 59.97,
  "paymentMethod": "credit_card",
  "billingAddress": {
    "street": "123 Main Street",
    "city": "Cityville",
    "zipcode": "12345",
    "country": "Countryland"
  }
}
```

   d. Response:

i.

```
{
  "orderId": "order_1234567890",
  "message": "Order successfully placed!"
}
```

7. User Bookshelf API Request & Response (**Add** and **Remove**):
   a. Route C - add & Route D - remove

b. Method: Post (add) & Delete (remove)
c. Request:
    i. 

d. Response:
    i. 

    ii. Or removed from the user's bookshelf

Design References:

- We want a user-friendly interface meaning the color scheme will go good together but not turn away users. We are basing it off of Kindle or Apple books. Having that simple way of looking at their bookshelf, searching up books, and logging in and signing out.