Front-End Technologies Basics Regular Exam

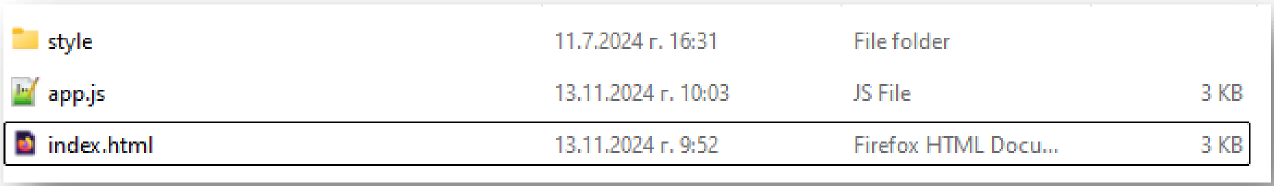
1. DOM Manipulation

Use the provided skeleton to solve this problem.

Write the missing functionality of this user interface.

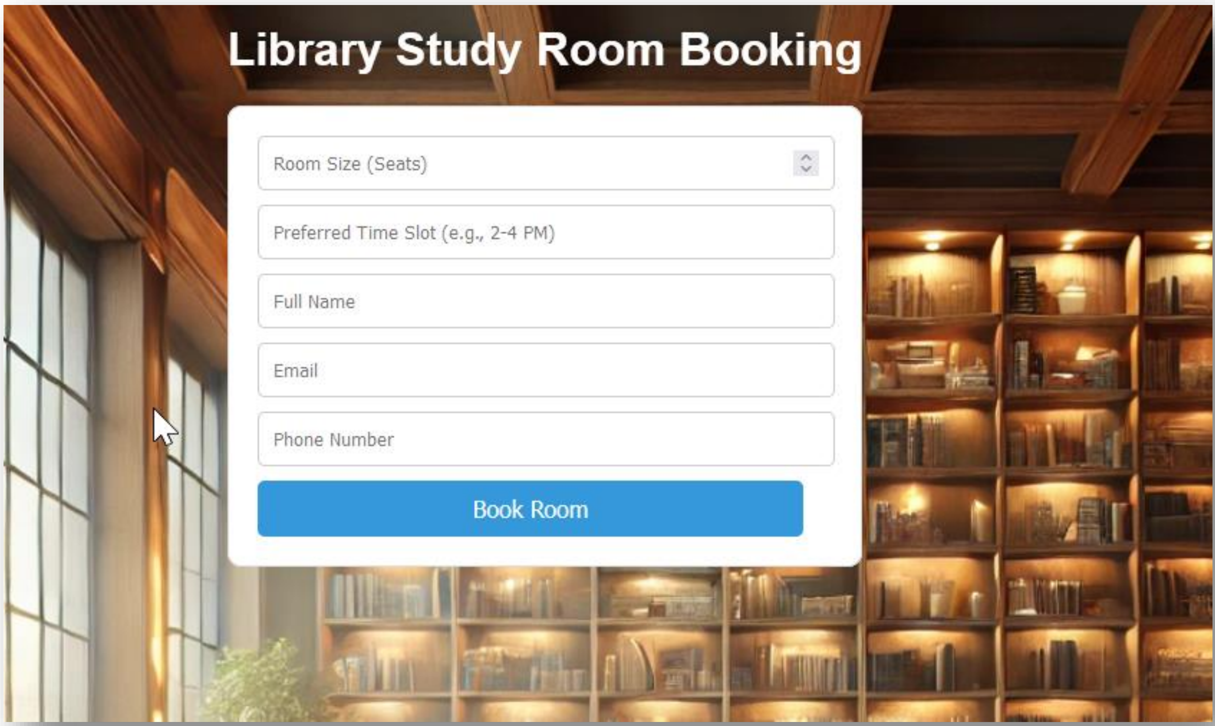
Application structure

You are provided with an HTML application - **Library Study Room Booking** with the following structure:

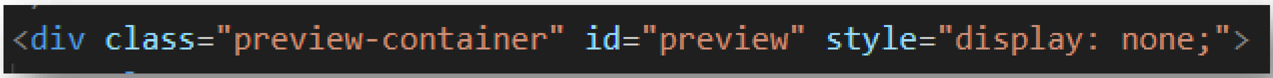


style	11.7.2024 г. 16:31	File folder	
app.js	13.11.2024 г. 10:03	JS File	3 KB
index.html	13.11.2024 г. 9:52	Firefox HTML Docu...	3 KB

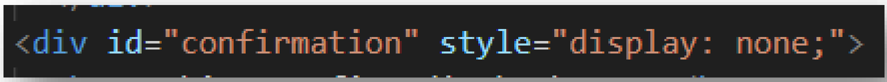
Open the index.html to start the application. Its purpose is to book library study rooms. The application looks like this:


Library Study Room Booking
Room Size (Seats)
Preferred Time Slot (e.g., 2-4 PM)
Full Name
Email
Phone Number
Book Room

Look at the html structure in the index.html file. You will notice that there are 2 hidden elements:



```html
<div class="preview-container" id="preview" style="display: none;">
```



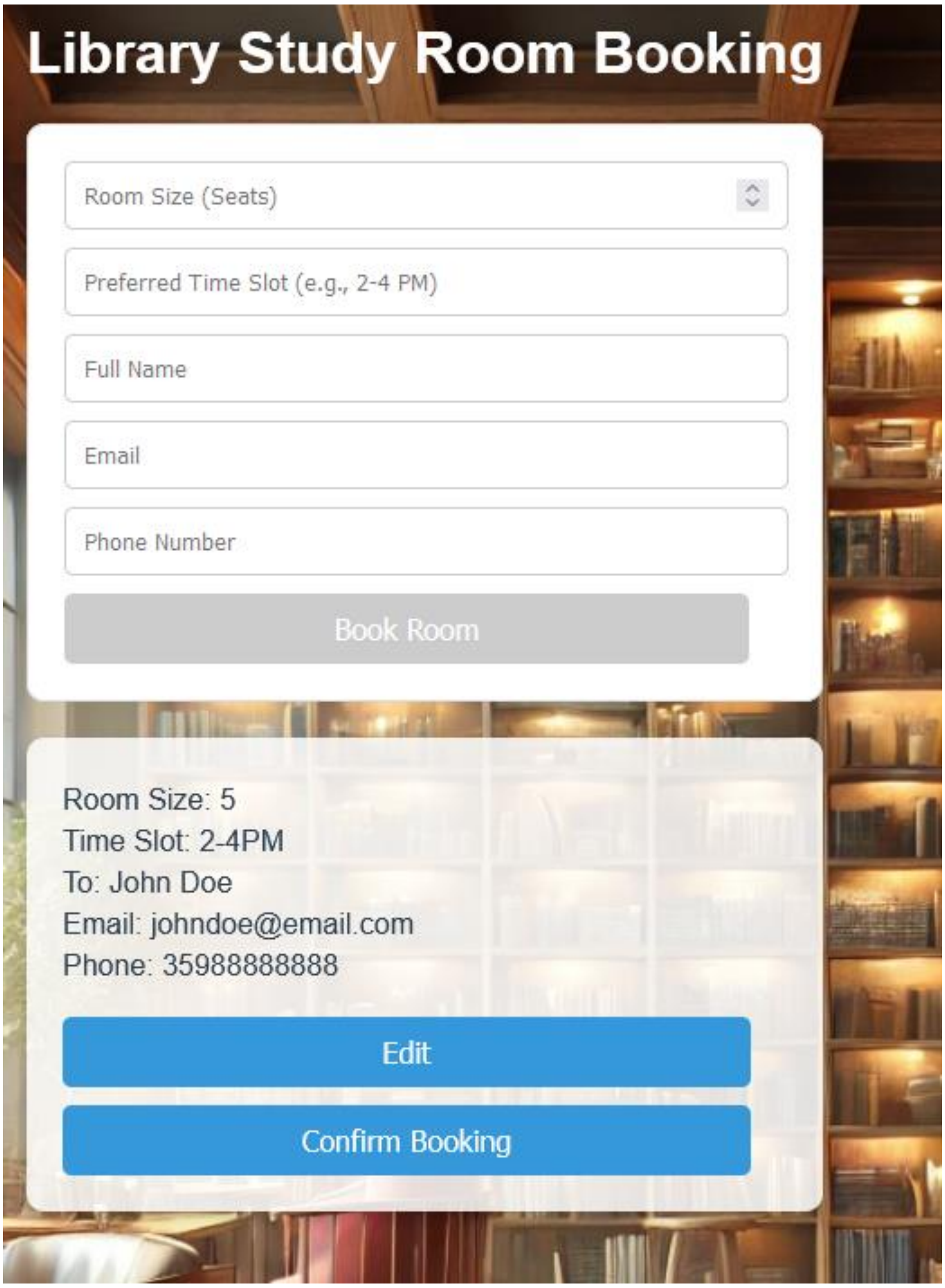
```html
<div id="confirmation" style="display: none;">
```

The preview element (**class="preview-container"**) is supposed to be displayed after the 'Book Room' button is clicked. The element with **id="confirmation"** element is supposed to be displayed after the booking is finalized.

Your Task

Write the missing JavaScript code in the app.js file to make the **Library Study Room Booking** work as expected:

1.1. Book Room Button Functionality



Room Size: 5
Time Slot: 2-4PM
To: John Doe
Email: johndoe@email.com
Phone: 35988888888

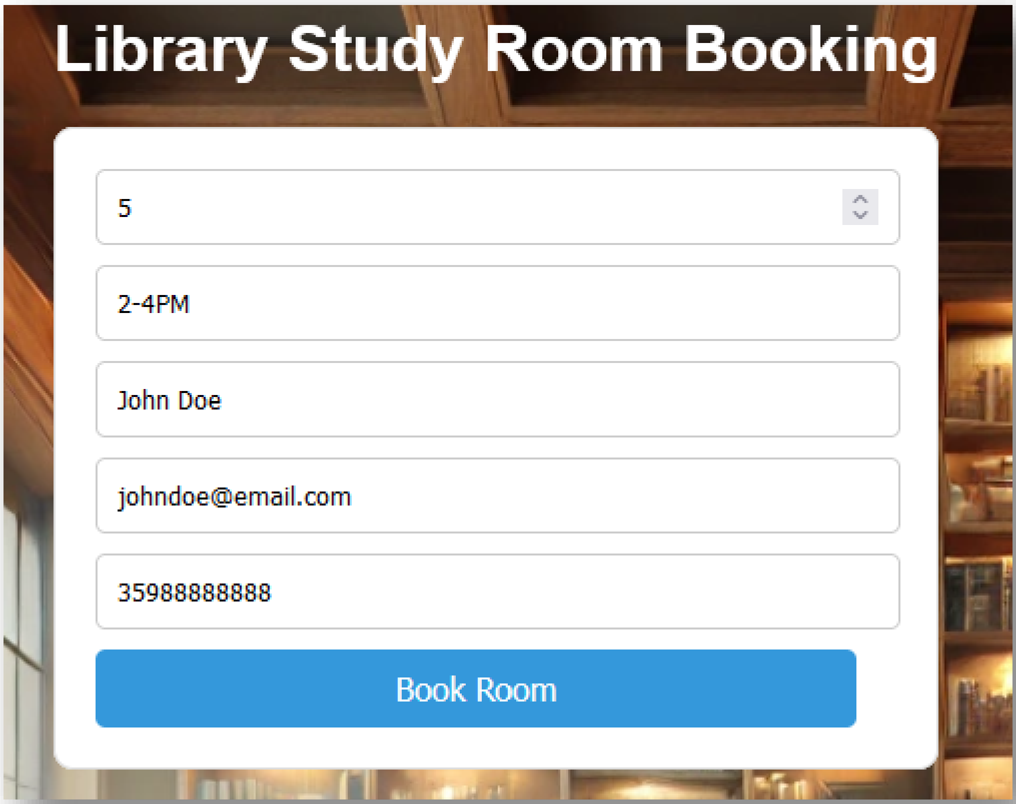
When the **[Book Room]** button is clicked, you need to validate that all fields are filled. If this is not the case, do not proceed.
If the validation is successful, the following functionality needs to be implemented:

- The information about the booked study room is filled in the preview element – room size, time slot, full name, email, phone number.
- The preview element must be displayed.
- The 'Book Room' button must be disabled.

- The values of the fields in the form must be cleared.

1.2. Edit Room Info

When the **[Edit] button** is **clicked, all of the information is loaded back into the input fields from step 1, while the [Book Room] button is enabled again.**



5

2-4PM

John Doe

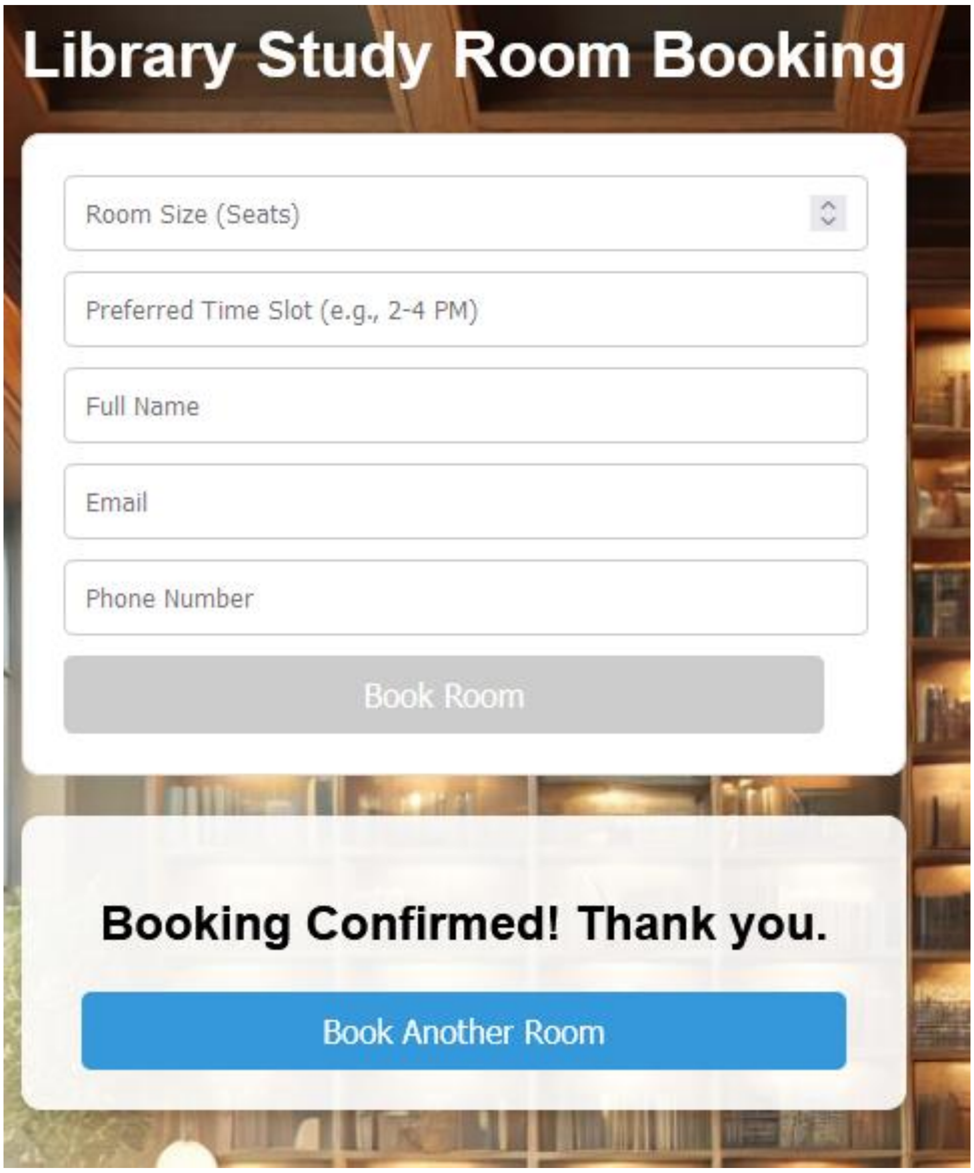
johndoe@email.com

35988888888

The preview element must be hidden.

1.3. Confirm Booking

When the [Confirm Booking] button is clicked, you must hide the preview element and show the element with **id="confirmation".**


Library Study Room Booking
Room Size (Seats)
Preferred Time Slot (e.g., 2-4 PM)
Full Name
Email
Phone Number
Book Room
Booking Confirmed! Thank you.
Book Another Room

1.4. Book Another Room

When the `[Book Another Room]` button is clicked, you must **hide the <u>element with id="confirmation</u>** and enable the **`[Book Room]` button**.

2. JS Application Testing

You have been given a JavaScript application. All the necessary setup is complete, allowing you to start writing your tests. Your goal is to write end-to-end (Front-End) tests using Playwright.

2.1. App Introduction

The **application for testing** is called **"Book Collection Management"**. It's a user-friendly application offering functionalities and permissions based on user login status:

1) For **guest user**:
   - **Home Page**: View a brief introduction to the app.
   - **Collection page**: See all created books.
   - **Search Page**: Search for books by title.
   - **Register/Login page**: Register a new account or log in.
2) For **Logged-In user**:
   - **Create Book Page**: Create new books.

- **Collection**: See all created books (both their own and others').
- **Search Page**: Search for books by title.
- **Logout**: Log out of the app.

3) **Book details functionality**:
   - **Detail View**: Only logged-in users can see detailed information about a book.
   - **Edit/Delete Buttons**: Only the owner of a book can see and use the Edit and Delete buttons in the detail view to modify or remove the book.
4) **Navigation bar**: Provides easy access to application functionalities based on your login status (logged in or guest).

Also, there will be seeded data for 3 books, that will always be loaded when you start the application.

2.2. Instructions

A **folder named "tests"** is prepared for you. There is **a single file named "e2e.test.js"**. In it, you must write your front-end tests with the Playwright framework.

As for the execution of the tests, **you need to start the back-end server of the application and the HTTP server**. Everything is configured for you, so you need just **to execute three commands in two terminals**:

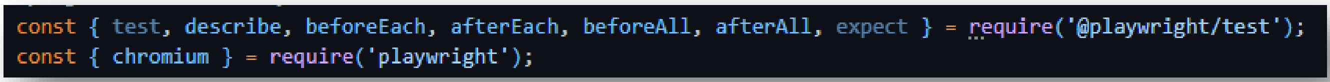
- "**npm install**" (or "**npm i** ") – to install the dependencies for your app.
- **"npm run server"** – to start the application back-end server.
- **"npm start"** (in another terminal)– to start the HTTP server.

Note: You will need to use **a third terminal window for the execution of Playwright tests**.
Use the command "**npm test**".

2.3. Front-End Testing with Playwright

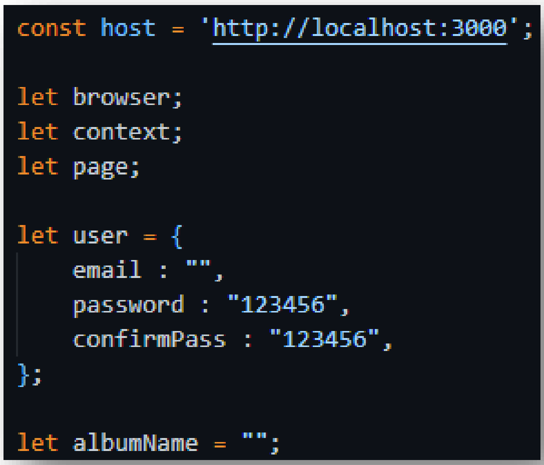
You are provided with predefined configurations in the e2e.test.js file:

- Needed imports for Playwright:



```
const { test, describe, beforeEach, afterEach, beforeAll, afterAll, expect } = require('@playwright/test');
const { chromium } = require('playwright');
```

- Predefined variables that you can use:



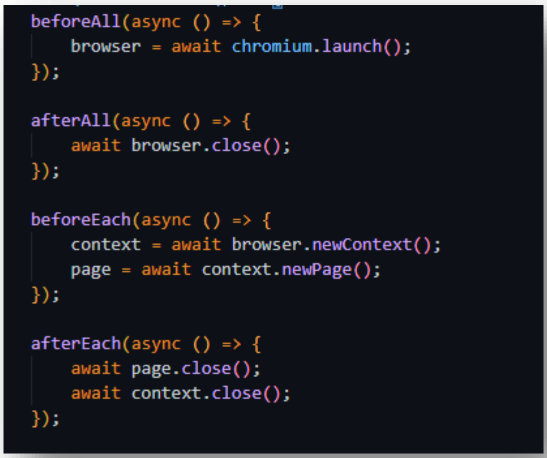
```
const host = 'http://localhost:3000';

let browser;
let context;
let page;

let user = {
    email : "",
    password : "123456",
    confirmPass : "123456",
};

let albumName = "";
```

- Before and after test configurations:

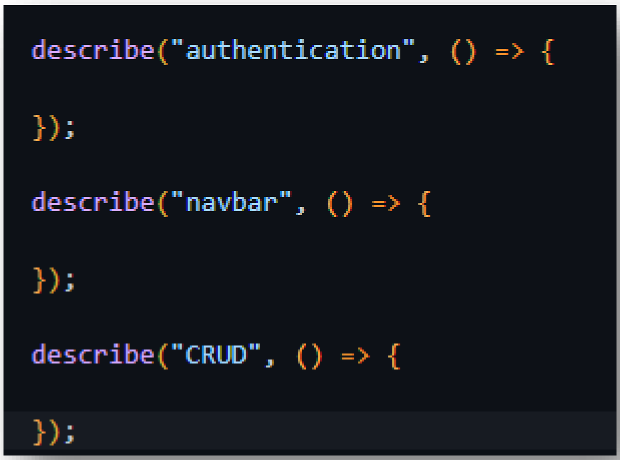

```
beforeAll(async () => {
    browser = await chromium.launch();
});

afterAll(async () => {
    await browser.close();
});

beforeEach(async () => {
    context = await browser.newContext();
    page = await context.newPage();
});

afterEach(async () => {
    await page.close();
    await context.close();
});
```

- Test suits:


```
describe("authentication", () => {

});

describe("navbar", () => {

});

describe("CRUD", () => {

});
```

Use them and write the following e2e tests with Playwright for the "Book Collection Management" application:

2.3.1. Registration with Valid Data (Authentication Functionality)

1. Create a test scope.
2. **Go to http://localhost:3000**
3. **Locate and click on the Register button**.
4. **Wait for the register form to load**.
5. **Create a unique email value**.
6. **Locate and fill the input field for email.**
7. **Locate and fill the input field for password**.
8. **Locate and fill the input field for confirm password**.
9. **Press the submit button**.
10. **Assert that you are redirected to the home page and the Logout button is visible**.

Hint: Use the predefined user object to hold and reuse user data.

2.3.2. Login with Valid Data (Authentication Functionality)

1. Create a test scope.
2. **Go to http://localhost:3000**
3. **Locate and click on the Login button**.
4. **Wait for the login form to load**.

5. **Locate and fill the input field for email.**
6. **Locate and fill the input field for password**.
7. **Press the submit button**.
8. **Assert that you are redirected to the home page and the Logout button is visible**.

2.3.3. Logout from the Application (Authentication Functionality)

1. **Create a test scope**.
2. **Go to http://localhost:3000**
3. **Log in to the application**.
4. **Click the Logout button**.
5. **Wait for Login button.**
6. **Assert that the URL is for home page.**

2.3.4. Navigation for Logged-In User Testing

1. Create a test scope.
2. **Go to http://localhost:3000**
3. **Log in to the application**.
4. **Assert that "Home", "Collection", "Search", "Create Book" and "Logout" buttons are visible**, and **"Login" and "Register" buttons are hidden.**

2.3.5. Navigation for Guest User Testing

1. Create a test scope.
2. **Go to http://localhost:3000**
3. **Assert that "Home", "Collection", "Search", "Login" and "Register" buttons are visible, and "Create Book" and "Logout" buttons are hidden.**

2.3.6. Create a Book Testing (CRUD Functionality)

1. Create a test scope.
2. **Go to http://localhost:3000**
3. **Log in to the application**.
4. **Locate and click the "Create Book" button**.
5. **Wait for the create book form to load**.
6. **Generate random book title** and save it in predefined variable.
7. **Locate and fill the input field for title with random generated value**.
8. **Locate and fill the input field for cover image.**
9. **Locate and fill the input field for year.**
10. **Locate and fill the input field for author.**
11. **Locate and fill the input field for genre.**
12. **Locate and fill the input field for description**.
13. **Press the submit button**.
14. **Assert that the url is http://localhost:3000/collection**.
15. **Assert that a book with the title you just added is present in the list**.

2.3.7. Edit a Book Testing (CRUD Functionality)

1. Create a test scope.
2. **Go to http://localhost:3000**
3. **Log in to the application**.
4. **Locate and click on "Search" button**.
5. **Locate and fill the input field for search.**

6. **Locate and click on the "Search" button for searching result.**
7. **Locate and click on the first book's title from the resuls**.
8. **Locate and click on Edit button**.
9. **Wait for the Edit form to load**.
10. **Locate and fill the title field in the edit form with new value to edit a book.**
11. **Press the submit button**.
12. **Assert that the title's value is as expected with edited value**.

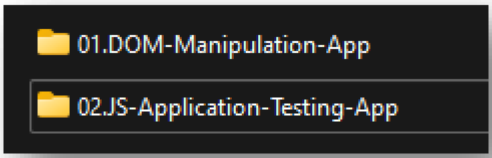
2.3.8. Delete a Book Testing (CRUD Functionality)

1. Create a test scope.
2. **Go to** http://localhost:3000
3. **Log in to the application**.
4. **Locate and click on "Search" button.**
5. **Locate and fill the input field for search.**
6. **Locate and click on "Search" button for searching result**.
7. **Locate and click on first book's Detail button with searched book title**.
8. **Press the delete button**.
9. **Assert that the url is** http://localhost:3000/collection.
10. **Assert that a book with the title you just deleted is NOT present in the list**.

3. How to Submit Your Work

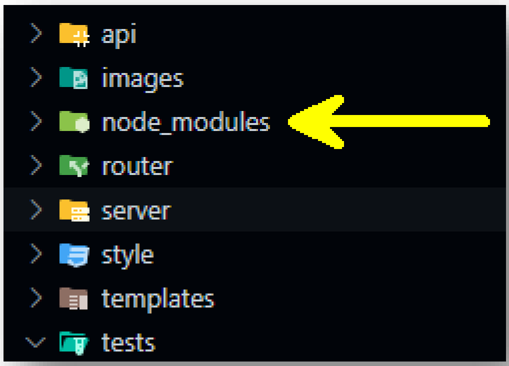
You need to submit your work on the SoftUni website in the Exam Section.

1. Create a folder.
2. Put the folders of both applications in it – the DOM manipulation app and the JS Application Testing app:



```
01.DOM-Manipulation-App
02.JS-Application-Testing-App
```

3. Go to the JS Application Testing app folder and delete the "node_modules" folder:



```
> api
> images
> node_modules
> router
> server
> style
> templates
v tests
```

4. Archive the folder that contains both applications and your solutions.
5. Upload the archive to the SoftUni website in the course section for your exam.