## Login/Registration Testing

| | |
|---|---|
| Test Objective: | Ensuring that correctly entered registration details result in successful registration of new user |
| Technique: | Send new Registration request with a correctly formatted inputs to register api and check the response returned |
| Completion Criteria: | Response returns successful user creation |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensuring that Blank Usernames aren't allowed as input |
| Technique: | Send new Registration request with a null username input to UserValidator and check the response returned |
| Completion Criteria: | Registration Request fails to be created and indicates blank username error |
| Special Considerations: | From a frontend perspective, this should show an error on the registration page rather than redirecting them anywhere |

| | |
|---|---|
| Test Objective: | Ensuring that Non-email usernames aren't allowed as input |
| Technique: | Send new Registration request with a username that's not an email input to UserValidator and check the response returned |
| Completion Criteria: | Registration Request fails to be created and indicates incorrectly formatted username error |
| Special Considerations: | From a frontend perspective, this should show an error on the registration page rather than redirecting them anywhere |

| | |
|---|---|
| Test Objective: | Ensuring that Blank Passwords aren't allowed as input |
| Technique: | Send new Registration Request with a null password input to UserValidator and check the response returned |
| Completion Criteria: | Registration Request fails to be created and indicates blank password error |
| Special Considerations: | From a frontend perspective, this should show an error on the registration page rather than redirecting them anywhere |

| | |
|---|---|
| Test Objective: | Ensuring that Passwords less than 6 characters aren't allowed as input |
| Technique: | Send new Registration Request with a 5-character input to UserValidator and check the response returned |
| Completion Criteria: | Registration Request fails to be created and indicates small password error |
| Special Considerations: | From a frontend perspective, this should show an error on the registration page rather than redirecting them anywhere |

| | |
|---|---|
| Test Objective: | Ensuring that registration forms with non-matching passwords are rejected |
| Technique: | Send new Registration Request with a different password/confirm password to UserValidator and check the response returned |
| Completion Criteria: | Registration Request fails to be created and indicates unmatching passwords |
| Special Considerations: | From a frontend perspective, this should show an error on the registration page rather than redirecting them anywhere |

| | |
|---|---|
| Test Objective: | Ensure that if a valid username/password combination is sent, the user can log in |
| Technique: | Pass a known valid username and password in a loginRequest to the Login API, check the response returned |
| Completion Criteria: | If the response returned indicates the login was successful and returns a JWT |
| Special Considerations: | This action should redirect users to the sites main page on completion |

| | |
|---|---|
| Test Objective: | Ensure that if invalid username is entered the login does not proceed |
| Technique: | Pass a known invalid username in a loginRequest to the Login API, check the response returned |
| Completion Criteria: | If the response returned indicates that there were no username matches |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Ensure that if invalid password is entered the login does not proceed |
| Technique: | Pass a known invalid password with a known valid Username in a loginRequest to the Login API, check the response returned |
| Completion Criteria: | If the response returned indicates that the password didn't match |
| Special Considerations: | - |

## Books Testing

| | |
|---|---|
| Test Objective: | Ensure that a valid book object can be created |
| Technique: | Pass a valid JSON object to the books microservice (via POST) and check the response status |
| Completion Criteria: | If the response status returned is: CREATED (201). |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Ensure that an invalid book object cannot be created |
| Technique: | Pass an invalid JSON object to the books microservice (via POST) and check the response status |
| Completion Criteria: | If the response status returned is: 400 (bad request) |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Ensure that a book object can be returned from the books microservice |
| Technique: | Use a GET request to query the books microservice api and then check the mock object returned / the response status |
| Completion Criteria: | If the returned object meets the criteria and the response status is OK (200) |
| Special Considerations: | Use a mock object being returned using mockito to ensure that the results are the same each time even if the persistent backend data is changed |

| | |
|---|---|
| Test Objective: | Ensure a book object using the wrong parameters cannot be returned from the books microservice |
| Technique: | Use a GET request to query the books micrcoservice with the wrong parameters and check the response status |
| Completion Criteria: | If the returned response status is 400 (bad request) |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Test that a PUT request to update a book can be executed correctly when the correct parameters are given |
| Technique: | Use a PUT request to query the books microservice using the correct body and then check the response status |
| Completion Criteria: | If the returned body contains the updated book and the response status is 200 (OK) |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Test that a request to update a book is not executed when incorrect parameters are given |
| Technique: | Use a PUT request containing incorrect body and check the response status |
| Completion Criteria: | If the returned response status is 400 (bad request) |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Test that a request to delete a book is executed when given the correct parameters |
| Technique: | Query the backend using a DELETE request and check the returned response status |
| Completion Criteria: | If the returned status is OK and the book has been deleted from the database |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Test that a request to delete a book is not executed when given incorrect parameters |
| Technique: | Query the backend with a delete request with incorrect parameters and check the returned response status |
| Completion Criteria: | If the returned response status is 400 and no book has been deleted from the database |
| Special Considerations: | - |

## Orders Testing

| | |
|---|---|
| Test Objective: | Ensure that a valid order is created |
| Technique: | Pass a valid JSON object to the orders microservice (via POST) and check the response status |
| Completion Criteria: | If the response status returned is: CREATED (201). |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Ensure that an invalid order object cannot be created |
| Technique: | Pass an invalid JSON object to the orders microservice (via POST) and check the response status |
| Completion Criteria: | If the response status returned is: 400 (bad request) |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Ensure that a valid cart is created |
| Technique: | Pass a valid JSON object to the orders microservice (via POST) and check the response status |
| Completion Criteria: | If the response status returned is: CREATED (201). |
| Special Considerations: | - |

| | |
|---|---|
| Test Objective: | Ensure that an invalid cart object cannot be created |
| Technique: | Pass an invalid JSON object to the orders microservice (via POST) and check the response status |
| Completion Criteria: | If the response status returned is: 400 (bad request) |
| Special Considerations: | - |

# FRONTEND TESTING

- COMPONENTS RENDERED TESTING

| | |
|---|---|
| Test Objective: | Ensure that the BookCardMaterial component is rendered successfully |
| Technique: | Pass  book details and check if the book title is being shown in the home page correctly |
| Completion Criteria: | If in the home page, the book title can be seen by checking, in the BookCardMaterial is showing the title |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the Login Page component is rendered successfully |
| Technique: | By checking if the heading “Sign In” exists, by passing the “Sign in” and checking the length |
| Completion Criteria: | It returns that the length is greater than 0, asserting that it is there. |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the AdminBookCard component is rendered successfully |
| Technique: | Pass  book details and check if the book title is being shown in the admin home page correctly |
| Completion Criteria: | If in the admin page, the book title can be seen, by checking in the AdminBookCard is showing the title |

| | |
|---|---|
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the AdminDashboard component is rendered successfully |
| Technique: | By checking if the bottom navigation button “Users” exists, by passing the “Users” and checking the length |
| Completion Criteria: | It returns that the length is equal to 5, asserting that it is there. |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the AdminDashboard component is rendered successfully |
| Technique: | By checking if the heading “Orders” exists, by passing the “Orders” and checking the length |
| Completion Criteria: | It returns that the length is equal to 5, asserting that it is there. |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the AdminDashboard component is rendered successfully |
| Technique: | By checking if the heading “Role Requests” exists, by passing the “Role Requests” and checking the length |
| Completion Criteria: | It returns that the length is equal to 5, asserting that it is there. |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the OrderHistory component is rendered successfully |
| Technique: | By checking if the heading “Order history” exists, by passing the “Order history” and checking the length |
| Completion Criteria: | It returns that the length is equal to 5, asserting that it is there. |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the ShoppingCart component is rendered successfully |
| Technique: | By checking if the heading “Cart Summary” exists, by passing the “Cart Summary” and checking if it is found in the screen |
| Completion Criteria: | It returns true, asserting that it is there. |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the ShoppingCart component is rendered successfully |
| Technique: | By checking if the button “Checkout” exists, by passing the “Checkout” and checking if it is found in the screen |
| Completion Criteria: | It returns true, asserting that it is there. |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | Ensure that the RegistrationPage component is rendered successfully |
| Technique: | By checking if the button “Register” exists, by passing the “Register” and checking if it is found in the screen |
| Completion Criteria: | It returns true, asserting that it is there. |
| Special Considerations: | |

- FRONTEND INTEGRATED WITH BACKEND (API REQUESTS SUCCESS) TESTING

| | |
|---|---|
| Test Objective: | API GET BOOKS and show all the books |
| Technique: | Call the post API requests and check the length of the getched books |
| Completion Criteria: | The length of the fetched books is greater than 0, which means it is working |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | API DELETE BOOK and the required book is deleted |
| Technique: | Call the delete API request and then checking the return response is true |
| Completion Criteria: | The return response is true,which means the book is deleted |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | API getAllUsers and all the users are shown |
| Technique: | Call the getAllUsers API request and then checking the return length is greater than 0 |
| Completion Criteria: | The return response is greater than 0,which means there are role users |
| Special Considerations: | |

| | |
|---|---|
| Test Objective: | API  getAllRoleRequests and the requested users are shown |
| Technique: | Call the getAllRoleRequests API request and then checking the return length is greater than 0 |
| Completion Criteria: | The return response is greater than 0,which means there are role requested users |
| Special Considerations: | |