# Test Plan

## Rough plan

Testing will be done using JavaScript. The goal of tests is too make sure that system works properly with security and performance.

Two types of test is required API tests and Unit tests.

Requirements - API test will use the node module ‘supertest’ to be able to request calls and get expected status results were as unit test will use ‘chai’ module for expected results.

Each test will be documented with expected results, actual results and possible fixing that needs to be done.

## Unit test

Test objectives – use cases must work properly

Test objects – GetBooksResource and RemoveBookResource

Test technique – for getting books it is enough to call the method and check if a list is returned. As for the remove that has input and modifies the book database it is required test security with input data (incorrect input and correct input) then need to compare the original database with the modified one to see if a book was indeed removed or not.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test no | Type | Target | Purpose of test | Expected result | Actual result |
| 1 | Input security | removeBook | How will remove handle a wrong type of input | Database will be the same | As expected |
| 2 | database | removeBook | How will remove change the database | Database contain one less book | As expected |

The get books will always be called when removing a book in order to compare original database and the modified one.

## API test

In order for the system to perform as expected the different use cases will need to work properly. By testing the response status in different cases this will be achieved.

Test objectives – API calls must work properly with different input values

Test objects – all resources (get books, get book, remove book, add book, edit book)

Test technique – call the API with different input values (correct and incorrect ones) and check if response status is as expected.

Quick note – add and edit is not implemented yet, therefore no data will be sent only checking status response.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Target | Test no | Input type | Expected status result | Actual result | Action that needs to be done |
| Get books | 1 | Don’t need input | 200 | 200 | Nothing |
| Get book | 1 | no input | 404 | 200 | Implement 404 response when calling get book (GET) (a) |
| 2 | incorrect input | 404 | 200 | Same as (a) |
| 3 | correct input | 200 | 200 | Nothing |
| Remove | 1 | no input | 404 | 200 | Implement 404 response when calling remove (REMOVE) (b) |
| 2 | incorrect input | 404 | 200 | Same as (b) |
| 3 | correct input | 200 | 200 | Nothing |
| Add | 1 | No input required | 200 | fail | Need to add callback to addBookResouce |
| Edit | 1 | incorrect input | 404 | 200 | Implement 404 response when calling edit (POST) |
| 2 | correct input | 200 | 200 | Nothing |

As the edit part is not implemented yet I don’t test it as much

* Implementation is made
* Test return expected response

# Appendences

### Rough tree

