# Test Plan

## Test objectives

The objectives for the tests in this stage of the development process are threefold. One is to prepare manual tests for the most important parts of the functionality, to make sure that not only the technical aspects of the system is on par, but that it is also usable from the client side. Two is to implement unit tests for both one of the implemented functions, to continuously check that I don't break what's working as I go forth, and for the next planned function, bring the function to add a book to the library. The third is to write API tests to check that everything will work as intended going forward.

## Static testing

Standard JS for code structure evaluation, check before each commit.

## Manual test cases

### Test 1.1

Add new book, ID: MT1. Test of use case ‘User adds new book’.

This manual test is meant to test the Add new book functionality of the library system, this being one of the primary use cases for the system and a prerequisite of most other uses of the system.

Prerequisites: Server up and running.

Test steps:

1. User goes to library page

### Test #2

ID MT2. Testing use case 'find a specific book'. This seems to me to be one of the most important, and most used, functions of a system like this - to check if a certain book is in there.

\* Name and id of the test-case

\* Reference to what requirement (use-case) that is tested

\* Short description of what is tested and why.

\* Preconditions that need to be fulfilled before this test is can be started (can be other test-cases or things like "Server must be started")

\* Test-steps including test-data. This is an exact description of what the tester is supposed to do. "Click 'Send' button", "Write 'Daniel' in Name box" etc.

\* A description of the expected result that should happen if the system works as specified.

\* Checkboxes if the test did succeed or fail.

\* A space for comments by the tester

## Automated unit tests

Unit test #1: Get books method, being a more important method than the delete method of the ones already implemented as you generally want to find a book in the system before deleting it. First test: Check that list books returns a list of objects.

Unit test # 2: Get books method. Check that every object in the list return is a book object, and nothing else, to make sure there is no unnecessary data cluttering the database.

Unit test #3: Add book, being the next method to implement. Test that Add book returns a book JSON object.

## Automated API tests

Delete book test, test with existing book ID, and erroneous book ID.