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 one of the next planned functions, being the functionality to get a specific book rather than a list of all. The third is to write API tests to check that these work, both ones already implemented and at least one of the ones to be implemented.

Static testing

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

Manual test cases

Test 2.1

Test ID: MT5.1. Add new book, The GraveYard Book by Neil Gaiman. 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
2. Click on Book
3. Click on New Book
4. Write in "The Graveyard Book" in Title field
5. Write "Neil Gaiman" in Author field
6. Click on Save

Expected result: Book should be added to database, and user should be returned to main library page.

☐ Test succeeded

☐ Test failed

Comments by tester:_____

_____

_____

Test 3.1

Test ID MT3.1. Testing use case 'find a specific book', with The Graveyard Book by Neil Gaiman. 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.

Prerequisites: Server up and running; Test MT2.1.

Test steps:

1. User goes to library page
2. Click in search field
3. Type “the graveyard book” in field
4. Click submit

Expected result: System returns a page listing The Graveyard book by Neil Gaiman.

☐ Test succeeded

☐ Test failed

Comments by tester:_____

Automated unit tests

Unit test #1: Test the xmlToJson function to make sure it is returning an object that has an ID. This because the ID is used for almost all other functions in the system, and thus very important to make sure it gets right.

Unit test #2: Test xmltoJson with empty input to make sure it doesn't crash.

Unit test #3: Test FindBookById to see if it returns book with correct id. Unfinished method.

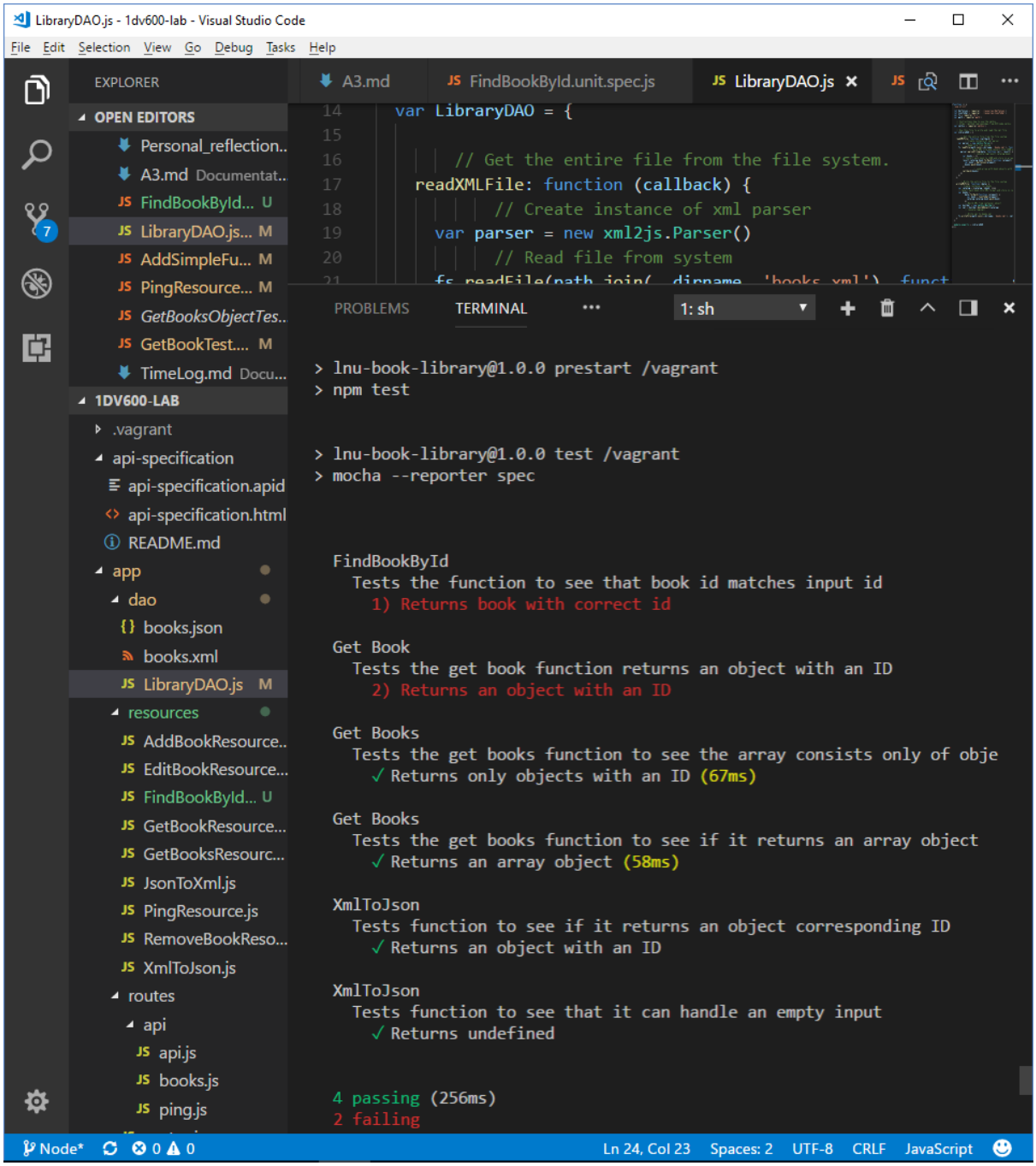
Automated API tests

API 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 an array.

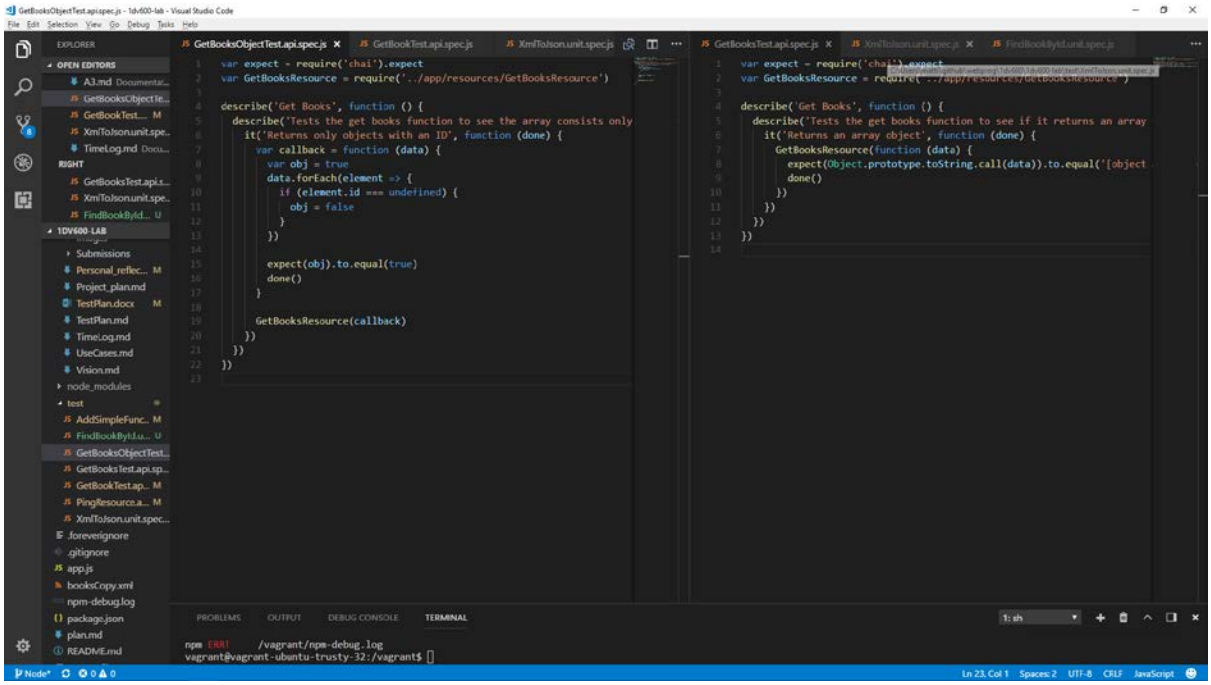
API test # 2: Get books method. Check that every object in the list return is an object with an ID.

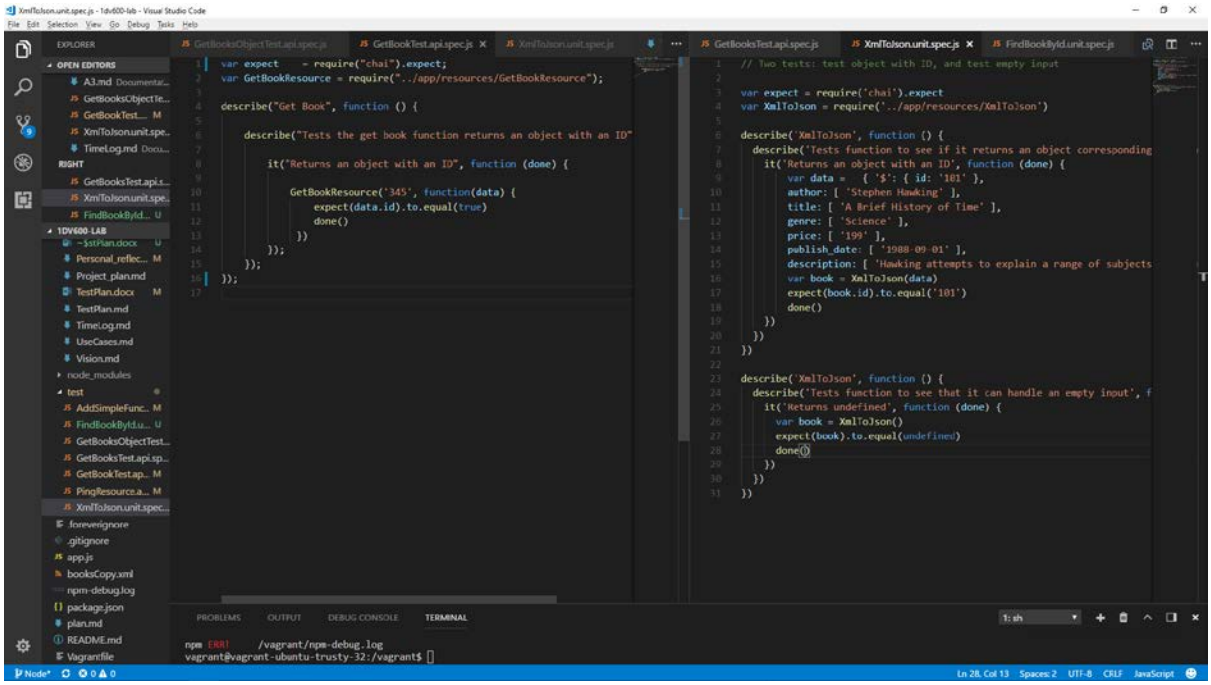
API test #3: Get book, being the next method to implement. Test that Get Book returns an object with an ID. Unfinished method.

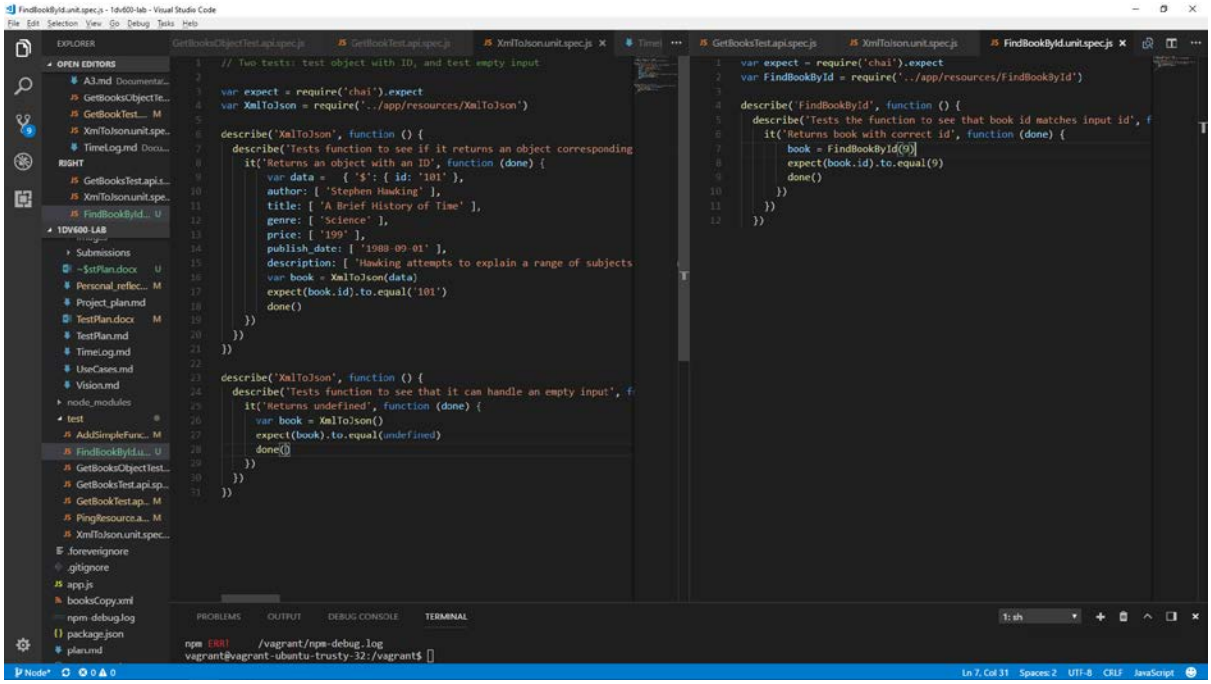
Screenshot showing unit tests and API tests, one failing and two succeeding of both.



Test code screenshots






FindBookById.unit.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
RIGHT
1DV600-LAB
Submissions
node_modules
test
// Two tests: test object with ID, and test empty input
var expect = require('chai').expect
var XmlToJson = require('../app/resources/XmlToJson')
describe('XmlToJson', function () {
describe('Tests function to see if it returns an object corresponding
it('Returns an object with an ID', function (done) {
var data = { '$': { id: '101' },
author: ['Stephen Hawking'],
title: ['A Brief History of Time'],
genre: ['Science'],
price: ['199'],
publish_date: ['1988-09-01'],
description: ['Hawking attempts to explain a range of subjects
var book = XmlToJson(data)
expect(book.id).to.equal('101')
done()
})
})
})
describe('XmlToJson', function () {
describe('Tests function to see that it can handle an empty input', f
it('Returns undefined', function (done) {
var book = XmlToJson()
expect(book).to.equal(undefined)
done()
})
})
})
var expect = require('chai').expect
var FindBookById = require('../app/resources/FindBook9Id')
describe('FindBookById', function () {
describe('Tests the function to see that book id matches input id', f
it('Returns book with correct id', function (done) {
book = FindBookById(9)
expect(book.id).to.equal(9)
done()
})
})
})
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
npm ERR! /vagrant/npm-debug.log
vagrant@vagrant-ubuntu-trusty-32:/vagrant$
Node*
Ln 7, Col 31
Spaces: 2
UTF-8
CRLF
JavaScript