

1 Introduction

1.1 Project Objective and Scope

The Objective of the test plan Document is:

- Identifying the assignment 2 requirements.
- Create test cases for the use cases which have been implemented so far up to assignment 2.
- The implementation of the unit test for test cases up to assignment 2
- Implementation of Automated API tests up to assignment 2
- Prove and show the test results

Scope:

In this iteration all the tests will be implemented and tested including Unit, function, and the API tests.

1.2 System overview

The software product that is being developed is a half-finished web application which is the foundation of a library system for books. The system can add books, modify the books, and remove the books. the system stack is implemented in a modern way and the two modules (the client and the server) will communicate via HTTP requests. The front end is implemented using a Single Page Application (SPA) architecture, which means that no rendering is done on server side all logic for the visual part is done locally and no reload of the page is needed. This also means that the border between back end and front end is very sharp, the only way to know how to communicate is to use the API. The application server is executed in a virtual machine that is managed via Vagrant, the client is executed in your web browser.

1.2 Stakeholders

- Customers: Fully functional Crud that have been tested.
- Developers: Test all functionality and make sure it is working according to API.

2 Test Requirements

Below is a list of the uses cases, functional and non-functional requirements that will be targeted for testing.

2.1 Manual Testing

- 2.1.1 Use Case 1: View a list of books.
- 2.1.2 Use Case 2: Delete a book

2.2 Unit Testing

- 2.2.1 Class: book.
- 2.2.2 Class: catalog.
- 2.2.3 Class: booksDAO.
- 2.2.4 Class: GetBooksResouruces.

2.3 API Testing

- 2.3.1 Class: GetBooksResouruces.
- 2.3.2 Class: RemoveBookResources.

3 Test Strategy

This section describes how the tests will be executed.

3.1 Manual Testing

- Test Objectives: Ensure the system is functioning properly.
- Testing techniques: Executing each Use case with vailed and invalid data to make sure of the results
- Criteria: Failure tests should be reported
- Consideration: None

3.2 Unit Testing

- Test Objectives: Ensure the Classes are functioning properly.
- Testing techniques: Create tests and implement then execute the tests and analyze the results.
- Criteria: Failure tests should be reported
- Consideration: Server must be running before testing

3.3 API Testing

- Test Objectives: Ensure the application is responding to the API.

- Testing techniques: Create tests and implement then execute the tests and analyze the results.
- Criteria: Failure tests should be reported
- Consideration: None

4 Resources and responsibilities

4.1 Man Power

Since this project is individual all responsibilities will be on me. It is my role to be the test designer which will identify and implement test cases, Tester which execute tests and analyze results and the programmer to implement other and missing functionalities.

4.2 Tools

Resource	type	Objective
Vagrant	Software, virtual machine	Software which runs virtual machine and code executer and execute manual testing
ATOM	Text editor.	Software for edition java codes
Eclipse	IDE	Execute tests

5 Milestones

Test Plan:

- Stated on: 3/25/2018
- Time needed: 2 hours
- finished on: 3/25/2018

Test Design:

- Stated on: 3/25/2018

- **Time needed: 1 hour**
- **finished on: 3/25/2018**

Implementation:

- **Stated on: 3/25/2018**
- **Time needed: 4 hours**
- **finished on: 3/25/2018**

Test Cases

Test ID	1
Test Name	User should view a list of books
Requirement Covered	2.1.1
Use Case	1
Test Scenario	User run the server and click on the books button from localhost9090
Pre-Conditions	Server must run and the system must have 1 or more book(s) and show it in the browser
Test Steps	User click on the book button
Test Data	Localhost:9090
Expected Results	System should display a list of books and response with 200 ok
Post-Conditions	User should see the list of books
Actual results	System displays the list of books and response with 200 ok
Status: (pass/fail)	pass
Date:	3/25/2018
Comments	The list must have some books

Test ID	2
Test Name	User should view an empty list of books
Requirement Covered	2.1.1
Use Case	1
Test Scenario	User run the server and click on the books button from localhost9090
Pre-Conditions	Server must run and the system must be empty of book(s) and an empty list must show up in the browser
Test Steps	User click on the book button
Test Data	Localhost:9090
Expected Results	System should display an empty list of books and response with 200 ok
Post-Conditions	User should see an empty list of books
Actual results	System displays an empty list of books and response with 200 ok
Status: (pass/fail)	pass
Date:	3/25/2018
Comments	The list should be empty of books

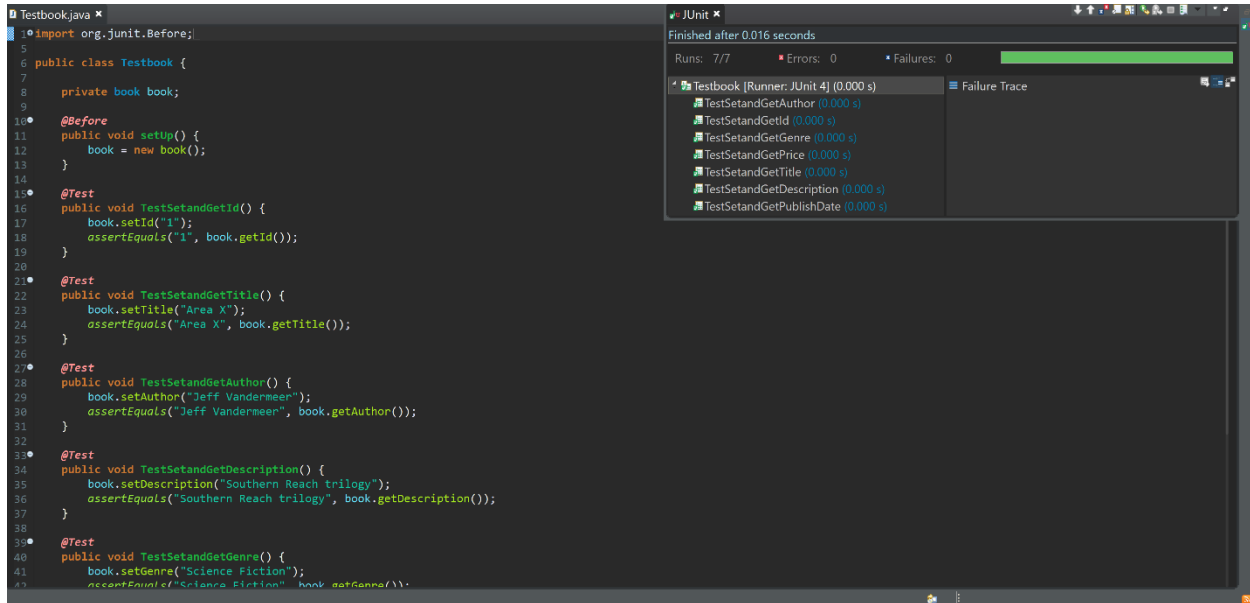
Test ID	3
Test Name	User should delete the selected book
Requirement Covered	2.1.2
Use Case	2
Test Scenario	User run the server and selects a book from the book list by clicking on the TITLE or AUTHOR of the specified book
Pre-Conditions	Server must run and the system must have 1 or more book(s)
Test Steps	1. User selects a book from the book list by clicking on the TITLE or AUTHOR of the selected book
	2. System opens the details page of the selected book
	3. User click on the delete button
Test Data	localhost:9090
Expected Results	System should delete the selected book and response with 200 ok
Post-Conditions	The selected book should be deleted
Actual results	System delete the selected book and response with 200 ok
Status: (pass/fail)	pass
Date:	3/25/2018
Comments	The list must have 1 or few books

Test ID	4
Test Name	User receive respond 404 not found
Requirement Covered	2.1.2
Use Case	2
Test Scenario	User run the server and sends a delete request through the system
Pre-Conditions	Server must run and the system does not contain the selected book the user wants to delete
Test Steps	User sends a request through the system
Test Data	localhost:9090/api/books/100
Expected Results	System should response with 404 not found
Post-Conditions	System should response with 404 not found
Actual results	System response with 404 not found
Status: (pass/fail)	pass
Date:	3/25/2018
Comments	The list should not have the specified book

Implementation and screenshots

Task 3: Unit Tests

Test Class: book

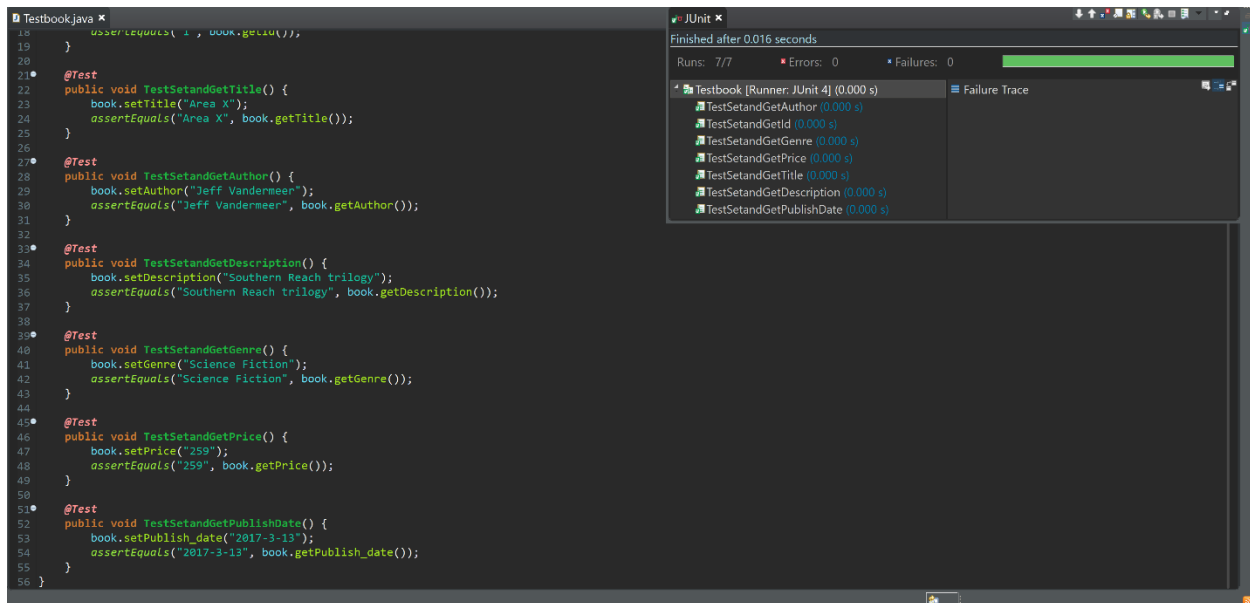


The screenshot shows an IDE with two panels. The left panel displays the `Testbook.java` file, which contains the following code:

```
1 import org.junit.Before;
2
3 public class Testbook {
4
5     private book book;
6
7     @Before
8     public void setUp() {
9         book = new book();
10    }
11
12    @Test
13    public void TestSetandGetId() {
14        book.setId("1");
15        assertEquals("1", book.getId());
16    }
17
18    @Test
19    public void TestSetandGetTitle() {
20        book.setTitle("Area X");
21        assertEquals("Area X", book.getTitle());
22    }
23
24    @Test
25    public void TestSetandGetAuthor() {
26        book.setAuthor("Jeff Vandermeer");
27        assertEquals("Jeff Vandermeer", book.getAuthor());
28    }
29
30    @Test
31    public void TestSetandGetDescription() {
32        book.setDescription("Southern Reach trilogy");
33        assertEquals("Southern Reach trilogy", book.getDescription());
34    }
35
36    @Test
37    public void TestSetandGetGenre() {
38        book.setGenre("Science Fiction");
39        assertEquals("Science Fiction", book.getGenre());
40    }
41}
```

The right panel shows the JUnit test runner output, indicating that the tests passed successfully. The output is as follows:

```
JUnit
Finished after 0.016 seconds
Runs: 7/7 Errors: 0 Failures: 0
Testbook [Runner:JUnit 4] (0.000 s)
TestSetandGetAuthor (0.000 s)
TestSetandGetId (0.000 s)
TestSetandGetGenre (0.000 s)
TestSetandGetPrice (0.000 s)
TestSetandGetTitle (0.000 s)
TestSetandGetDescription (0.000 s)
TestSetandGetPublishDate (0.000 s)
```



The screenshot shows an IDE with two panels. The left panel displays the `Testbook.java` file, which contains the following code:

```
18 assertEquals("1", book.getId());
19 }
20
21 @Test
22 public void TestSetandGetTitle() {
23     book.setTitle("Area X");
24     assertEquals("Area X", book.getTitle());
25 }
26
27 @Test
28 public void TestSetandGetAuthor() {
29     book.setAuthor("Jeff Vandermeer");
30     assertEquals("Jeff Vandermeer", book.getAuthor());
31 }
32
33 @Test
34 public void TestSetandGetDescription() {
35     book.setDescription("Southern Reach trilogy");
36     assertEquals("Southern Reach trilogy", book.getDescription());
37 }
38
39 @Test
40 public void TestSetandGetGenre() {
41     book.setGenre("Science Fiction");
42     assertEquals("Science Fiction", book.getGenre());
43 }
44
45 @Test
46 public void TestSetandGetPrice() {
47     book.setPrice("259");
48     assertEquals("259", book.getPrice());
49 }
50
51 @Test
52 public void TestSetandGetPublishDate() {
53     book.setPublish_date("2017-3-13");
54     assertEquals("2017-3-13", book.getPublish_date());
55 }
56 }
```

The right panel shows the JUnit test runner output, indicating that the tests passed successfully. The output is as follows:

```
JUnit
Finished after 0.016 seconds
Runs: 7/7 Errors: 0 Failures: 0
Testbook [Runner:JUnit 4] (0.000 s)
TestSetandGetAuthor (0.000 s)
TestSetandGetId (0.000 s)
TestSetandGetGenre (0.000 s)
TestSetandGetPrice (0.000 s)
TestSetandGetTitle (0.000 s)
TestSetandGetDescription (0.000 s)
TestSetandGetPublishDate (0.000 s)
```

Class Testbook

[all](#) > [default-package](#) > Testbook

7	0	0	0.006s
tests	failures	ignored	duration

100%

successful

Tests

Test	Duration	Result
TestSetandGetAuthor	0.001s	passed
TestSetandGetDescription	0.001s	passed
TestSetandGetGenre	0.001s	passed
TestSetandGetId	0s	passed
TestSetandGetPrice	0s	passed
TestSetandGetPublishDate	0.003s	passed
TestSetandGetTitle	0s	passed

TestRemoveBooks

```
TestRemoveBooks.java
1 import static org.junit.Assert.assertEquals;
2
3 public class TestRemoveBooks {
4     booksDAO dao = new booksDAO();
5
6     String jsonWithTwoBooks = "[{"id":"1","title":"Area X"}, {"id":"2","title":"The Name of the Wind"}]";
7     String jsonBookTest = "[{"id":"1","title":"Area X"}, {"id":"2","title":"The Name of the Wind"}, {"id":"3","title":"A Song of Ice and Fire"}, {"id":"4","title":"Minecraft: War of the Ar"}]";
8
9     /*fail method*/
10    @Test
11    public void SearchAuthorRecursive() throws Exception {
12        catalog = dao.ToCatalog();
13        assertEquals("Jeff Vandermeer", catalog.binarySearchNonRecursive("Jeff Vandermeer"));
14    }
15
16    /*fail method*/
17    @Test
18    public void SearchAuthorNoneRecursive() throws Exception {
19        catalog = dao.ToCatalog();
20        assertEquals(true, catalog.binarySearchRecursive("Jeff Vandermeer"));
21    }
22
23    @Test
24    public void ShouldReturnAllbooksToJSON() throws Exception {
25        catalog = dao.ToCatalog();
26        assertEquals(jsonBookTest, catalog.ConvertToJSON());
27    }
28
29    @Test
30    public void testSizeWithAllBooks() throws Exception {
31        int size = 4;
32        catalog = dao.ToCatalog();
33        assertEquals(size, catalog.getListSize());
34    }
35 }
```

JUnit

Finished after 0.566 seconds

Runs: 6/6 Errors: 0 Failures: 2

TestRemoveBooks [Runner: JUnit 4] (0.002 s)

ShouldReturnAllbooksToJSON (0.001 s)

testSizeWithAllBooks (0.000 s)

SearchAuthorNoneRecursive (0.001 s)

SearchAuthorRecursive (0.000 s)

RemoveAndTestSizeofTwoBooks (0.000 s)

ShouldRemoveAndReturnTwoBooksToJSON (0.000 s)

Failure Trace

! java.lang.AssertionError: expected:<true> but was:<false>

at TestRemoveBooks.SearchAuthorNoneRecursive(TestRemoveBooks.java:26)

```
TestRemoveBooks.java
23 @Test
24 public void SearchAuthorNoneRecursive() throws Exception {
25     catalog = dao.ToCatalog();
26     assertEquals(true, catalog.binarySearchRecursive("Jeff Vandermeer"));
27 }
28
29
30
31
32 @Test
33 public void ShouldReturnAllbooksToJSON() throws Exception {
34     catalog = dao.ToCatalog();
35     assertEquals(jsonBookTest, catalog.ConvertToJSON());
36 }
37
38 @Test
39 public void testSizeWithAllBooks() throws Exception {
40     int size = 4;
41     catalog = dao.ToCatalog();
42     assertEquals(size, catalog.getListSize());
43 }
44
45 @Test
46 public void ShouldRemoveAndReturnTwoBooksToJSON() throws Exception {
47     catalog = dao.ToCatalog();
48     catalog.removeBook("3");
49     catalog.removeBook("4");
50     assertEquals(jsonWithTwoBooks, catalog.ConvertToJSON());
51 }
52
53 @Test
54 public void RemoveAndTestSizeofTwoBooks() throws Exception {
55     int size = 3;
56     catalog = dao.ToCatalog();
57     catalog.removeBook("3");
58     assertEquals(size, catalog.getListSize());
59 }
60 }
```

JUnit

Finished after 0.532 seconds

Runs: 6/6 Errors: 0 Failures: 2

TestRemoveBooks [Runner: JUnit 4] (0.002 s)

ShouldReturnAllbooksToJSON (0.001 s)

testSizeWithAllBooks (0.000 s)

SearchAuthorNoneRecursive (0.001 s)

SearchAuthorRecursive (0.000 s)

RemoveAndTestSizeofTwoBooks (0.000 s)

ShouldRemoveAndReturnTwoBooksToJSON (0.000 s)

Failure Trace

! org.junit.ComparisonFailure: expected:<[Jeff Vandermeer]> but was:<[-1]>

at TestRemoveBooks.SearchAuthorRecursive(TestRemoveBooks.java:20)

Class TestRemoveBooks

all > default-package > TestRemoveBooks

6

tests

2

failures

0

ignored

0.478s

duration

66%

successful

Failed tests

Tests

Test	Duration	Result
RemoveAndTestSizeofTwoBooks	0.084s	passed
SearchAuthorRecursive	0.063s	failed
SearchAuthornoneRecursive	0.075s	failed
ShouldRemoveAndReturnTwoBooksToJSON	0.026s	passed
ShouldReturnAllbooksToJSON	0.154s	passed
testSizewithAllBooks	0.076s	passed

Class TestRemoveBooks

all > default-package > TestRemoveBooks

6

tests

2

failures

0

ignored

0.478s

duration

66%

successful

Failed tests

Tests

SearchAuthorRecursive

```
org.junit.ComparisonFailure: expected:<[Jeff Vandermeer]> but was:<[-1]>
    at org.junit.Assert.assertEquals(Assert.java:115)
    at org.junit.Assert.assertEquals(Assert.java:144)
    at TestRemoveBooks.SearchAuthorRecursive(TestRemoveBooks.java:20)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
    at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)
    at org.junit.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
    at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:325)
    at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:78)
    at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:57)
    at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)
    at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:71)
    at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:208)
    at org.junit.runners.ParentRunner.access$000(ParentRunner.java:58)
    at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:268)
    at org.junit.runners.ParentRunner.run(ParentRunner.java:363)
    at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.runTestClass(JUnit4TestClassExecutor.java:116)
    at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.execute(JUnit4TestClassExecutor.java:59)
    at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.execute(JUnit4TestClassExecutor.java:39)
    at org.gradle.api.internal.tasks.testing.junit.AbstractJUnit4TestClassProcessor.processTestClass(AbstractJUnit4TestClassProcessor.java:66)
```

```

at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.execute(JUnit4TestClassExecutor.java:110)
at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.execute(JUnit4TestClassExecutor.java:59)
at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.execute(JUnit4TestClassExecutor.java:39)
at org.gradle.api.internal.tasks.testing.junit.AbstractJUnit4TestClassProcessor.processTestClass(AbstractJUnit4TestClassProcessor.java:66)
at org.gradle.api.internal.tasks.testing.SuiteTestClassProcessor.processTestClass(SuiteTestClassProcessor.java:51)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:35)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:24)
at org.gradle.internal.dispatch.ContextClassLoaderDispatch.dispatch(ContextClassLoaderDispatch.java:32)
at org.gradle.internal.dispatch.ProxyDispatchAdapter$DispatchingInvocationHandler.invoke(ProxyDispatchAdapter.java:93)
at com.sun.proxy.$Proxy3.processTestClass(Unknown Source)
at org.gradle.api.internal.tasks.testing.worker.TestWorker.processTestClass(TestWorker.java:109)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:35)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:24)
at org.gradle.internal.remote.internal.hub.MessageHubBackedObjectConnection$DispatchWrapper.dispatch(MessageHubBackedObjectConnection.java:146)
at org.gradle.internal.remote.internal.hub.MessageHubBackedObjectConnection$DispatchWrapper.dispatch(MessageHubBackedObjectConnection.java:128)
at org.gradle.internal.remote.internal.hub.MessageHub$Handler.run(MessageHub.java:404)
at org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(ExecutorPolicy.java:63)
at org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:46)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at org.gradle.internal.concurrent.ThreadFactoryImpl$ManagedThreadRunnable.run(ThreadFactoryImpl.java:55)
at java.lang.Thread.run(Thread.java:748)

```

SearchAuthorNoneRecursive

```

java.lang.AssertionError: expected:<true> but was:<false>
at org.junit.Assert.fail(Assert.java:88)
at org.junit.Assert.failNotEquals(Assert.java:834)
at org.junit.Assert.assertEquals(Assert.java:118)
at org.junit.Assert.assertEquals(Assert.java:144)
at TestRemoveBooks.SearchAuthorNoneRecursive(TestRemoveBooks.java:26)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)

```

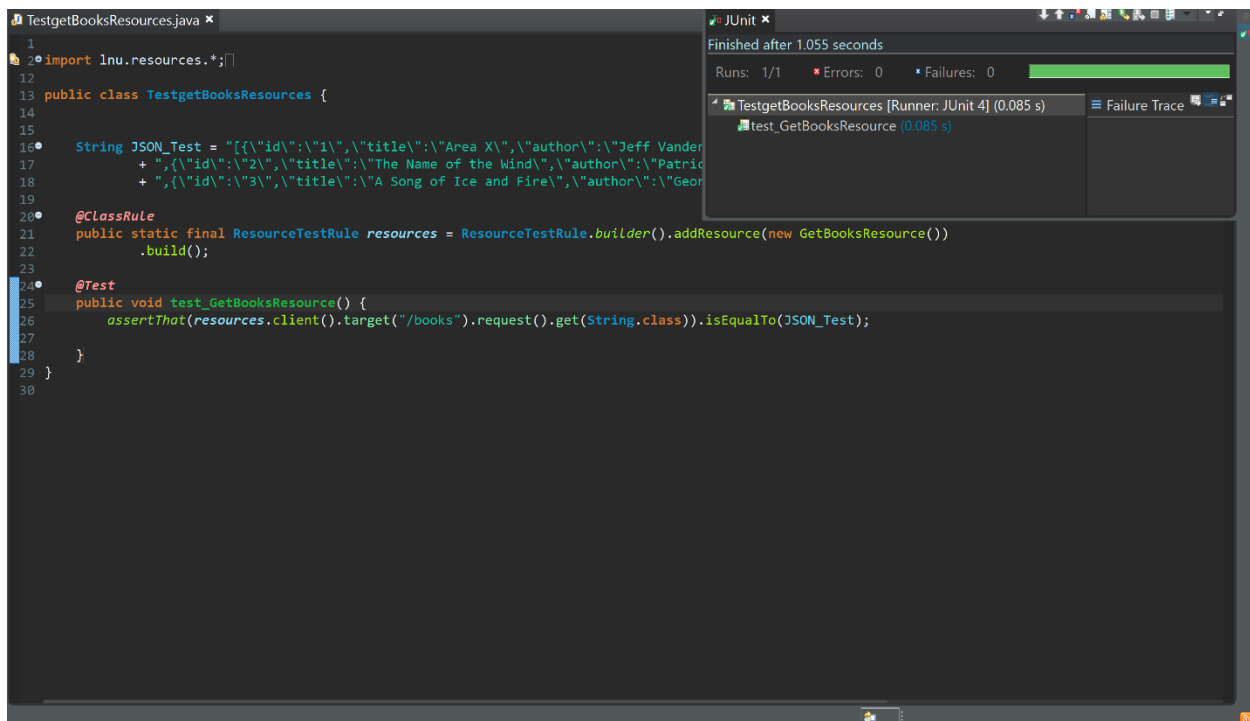
```

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)
at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:325)
at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:78)
at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:57)
at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)
at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:71)
at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:288)
at org.junit.runners.ParentRunner.access$000(ParentRunner.java:58)
at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:268)
at org.junit.runners.ParentRunner.run(ParentRunner.java:363)
at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.runTestClass(JUnit4TestClassExecutor.java:116)
at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.execute(JUnit4TestClassExecutor.java:59)
at org.gradle.api.internal.tasks.testing.junit.JUnit4TestClassExecutor.execute(JUnit4TestClassExecutor.java:39)
at org.gradle.api.internal.tasks.testing.junit.AbstractJUnit4TestClassProcessor.processTestClass(AbstractJUnit4TestClassProcessor.java:66)
at org.gradle.api.internal.tasks.testing.SuiteTestClassProcessor.processTestClass(SuiteTestClassProcessor.java:51)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:35)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:24)
at org.gradle.internal.dispatch.ContextClassLoaderDispatch.dispatch(ContextClassLoaderDispatch.java:32)
at org.gradle.internal.dispatch.ProxyDispatchAdapter$DispatchingInvocationHandler.invoke(ProxyDispatchAdapter.java:93)
at com.sun.proxy.$Proxy3.processTestClass(Unknown Source)
at org.gradle.api.internal.tasks.testing.worker.TestWorker.processTestClass(TestWorker.java:109)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:35)
at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:24)
at org.gradle.internal.remote.internal.hub.MessageHubBackedObjectConnection$DispatchWrapper.dispatch(MessageHubBackedObjectConnection.java:146)
at org.gradle.internal.remote.internal.hub.MessageHubBackedObjectConnection$DispatchWrapper.dispatch(MessageHubBackedObjectConnection.java:128)
at org.gradle.internal.remote.internal.hub.MessageHub$Handler.run(MessageHub.java:404)
at org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(ExecutorPolicy.java:63)
at org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:46)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at org.gradle.internal.concurrent.ThreadFactoryImpl$ManagedThreadRunnable.run(ThreadFactoryImpl.java:55)
at java.lang.Thread.run(Thread.java:748)

```

In testRemoveBooks I test the the “ConvertToJSON()” Method and “removeBook(String id)”. I also, tested the “getListSize()” before and after removing book(s) form the system. The method “searchAuthorfRecursive()” and “searchAuthornoneRecursive” are an incomplete methods which supposed to search for Author name using recursive Binary Search where it returns “searchAuthorfRecursive()” should return a Boolean and “searchAuthornoneRecursive” should return a String . I expected the method to return true since the Author is in the list but it returned falls since I didn’t take into consideration that the array is not alphabetically sorted based on Author names.

TestgetBooksResources



The screenshot displays an IDE with two panels. The left panel shows the source code for `TestgetBooksResources.java`, and the right panel shows the JUnit test runner output.

Source Code:

```
1
2 import lnw.resources.*;
3
4 public class TestgetBooksResources {
5
6     String JSON_Test = "[{"id\":\"1\", \"title\":\"Area X\", \"author\":\"Jeff Vander
7     + \", {\"id\":\"2\", \"title\":\"The Name of the Wind\", \"author\":\"Patric
8     + \", {\"id\":\"3\", \"title\":\"A Song of Ice and Fire\", \"author\":\"Geor
9
10
11     @ClassRule
12     public static final ResourceTestRule resources = ResourceTestRule.builder().addResource(new GetBooksResource())
13         .build();
14
15
16     @Test
17     public void test_GetBooksResource() {
18         assertEquals(resources.client().target("/books").request().get(String.class)).isEqualTo(JSON_Test);
19     }
20 }
```

JUnit Runner Output:

```
Finished after 1.055 seconds
Runs: 1/1   Errors: 0   Failures: 0
TestgetBooksResources [Runner: JUnit 4] (0.085 s)
test_GetBooksResource (0.085 s)
```

Class TestgetBooksResources

[all](#) > [default-package](#) > TestgetBooksResources

1	0	0	0.111s
tests	failures	ignored	duration

100%

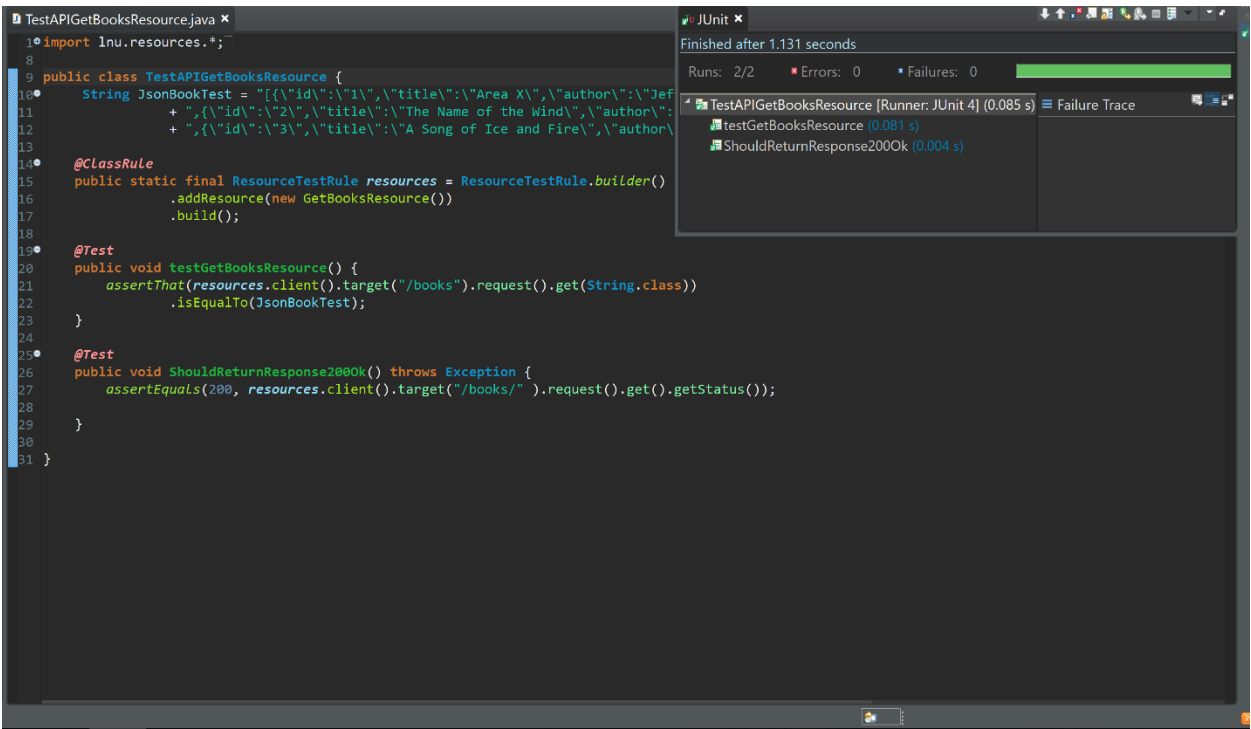
successful

Tests

Test	Duration	Result
test_GetBooksResource	0.111s	passed

Task 4: API Test

TestAPIGetBooksResource



Class TestAPIGetBooksResource

all > default-package > TestAPIGetBooksResource

2

0

0

0.401s

tests

failures

ignored

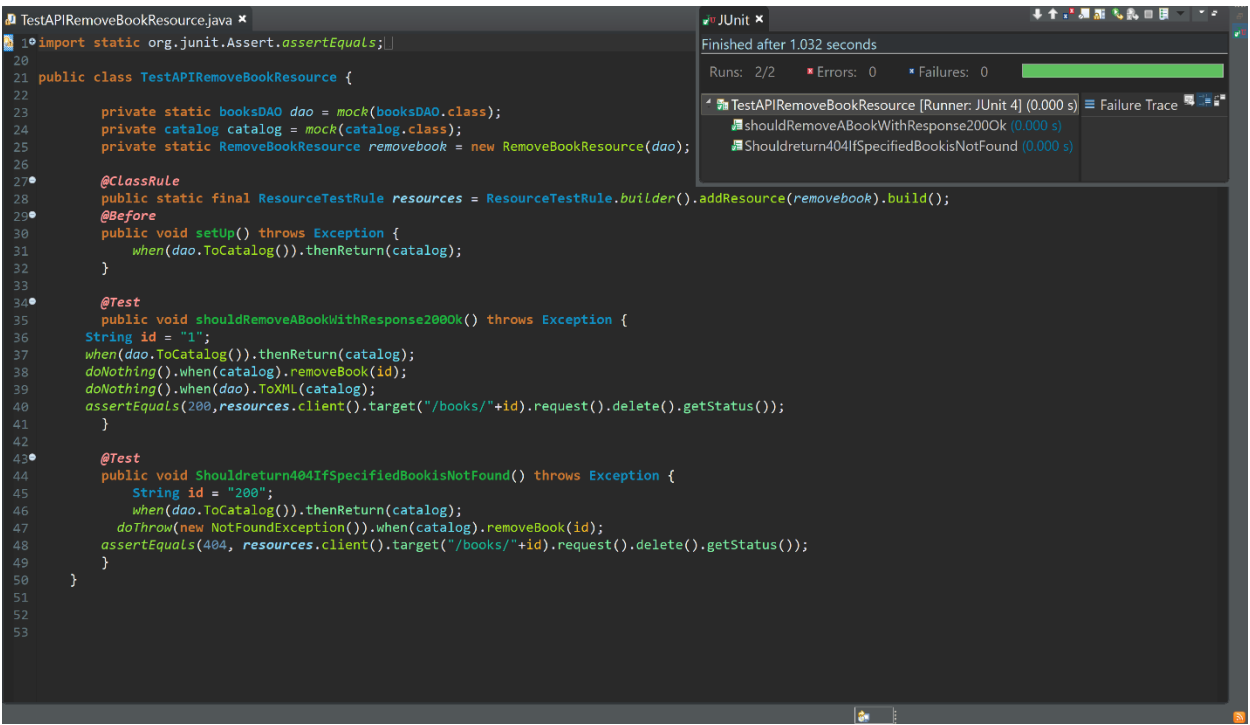
duration

100%
successful

Tests

Test	Duration	Result
ShouldReturnResponse200Ok	0.030s	passed
testGetBooksResource	0.371s	passed

TestAPIRemoveBookResource



Class TestAPIRemoveBookResource

all > default-package > TestAPIRemoveBookResource

2

tests

0

failures

0

ignored

0.166s

duration

100%
successful

Tests

Test	Duration	Result
Shouldreturn404IfSpecifiedBookisNotFound	0.010s	passed
shouldRemoveABookWithResponse200Ok	0.156s	passed

Personal Reflection

Writing a Test Plan was a bit confusing I had to do some research and dig up templates on how to write a good Test Plan. Also, I had to do some research on how to write a proper Test Case in which I had found some templates on google that have helped me. The most difficult part was Unit and API Tests in which I was lost and confused on how to implement them. I ran the tests on eclipse as a Junit tests since I sometimes use Eclipse as my “text editor” since I have installed addons to help me instead of Atom and I find Eclipse is much better than Atom. After trying it on Eclipse I ran it through vagrant in which I had some mistakes at first but later I managed to get them to work and I discovered that vagrant makes a HTML documents of the tests that I executed throughout vagrant in Path “Project/Build/reports/tests” in which I took screen shots. Testing was confusing at first but I learned new methods and testing Techniques.

Time Log

Test Plan

Started	Task	Timed estimated	Turn out time	ended
3/25/2018	Researching	1 hour	1 hour	3/25/2018
3/25/2018	Writing	2 hours	1hour	3/25/2018

Test Cases

Started	Task	Timed estimated	Turn out time	ended
3/25/2018	Researching	2 hours	1 hour	3/25/2018
3/25/2018	Writing and Implementation	1 hour	40 minutues	3/25/2018

Unit Tests

Started	Task	Timed estimated	Turn out time	ended
3/25/2018	Researching	2 hour	2 hours	3/25/2018
3/25/2018	Implementation	1 hour	2 hours	3/25/2018

API Tests

Started	Task	Timed estimated	Turn out time	ended
3/25/2018	Researching	2 hours	1 hour	3/25/2018
3/25/2018	Implementation	1 hour	2 hours	3/25/2018

Reflection

Started	Task	Timed estimated	Turn out time	ended
3/25/2018	Writing Reflection	30 minutes	15 minutes	3/25/2018