# READING LIST APP
# PROJECT DOCUMENTATION

Asawer Khan

BAE14

# Table Of Contents

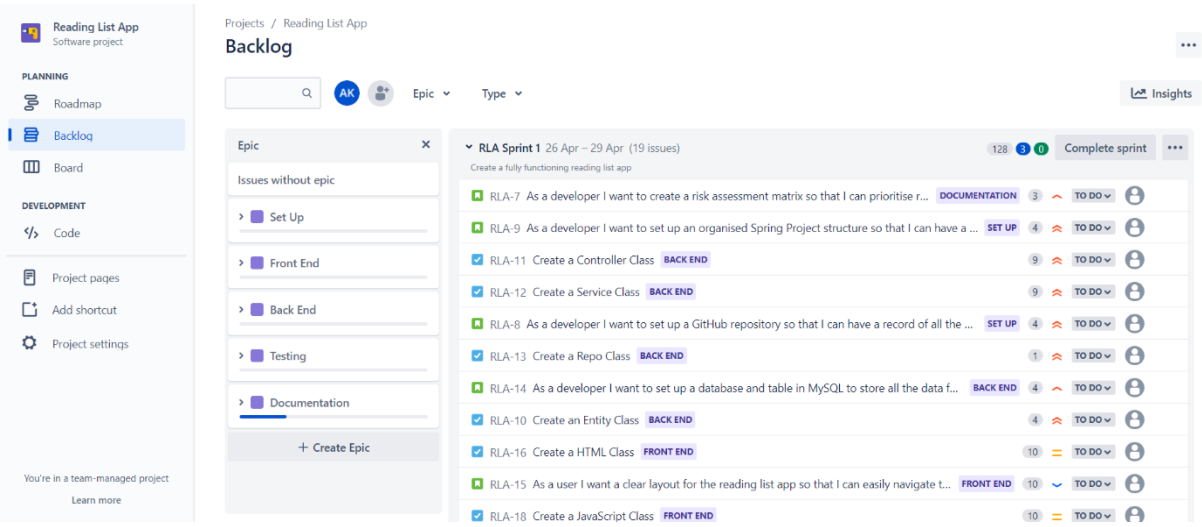
## Contents

Setup Jira Board

Created a Jira Board which consisted of a Sprint made up of 5 Epics:

- Setup
- Documentation
- Frontend
- Backend
- Testing

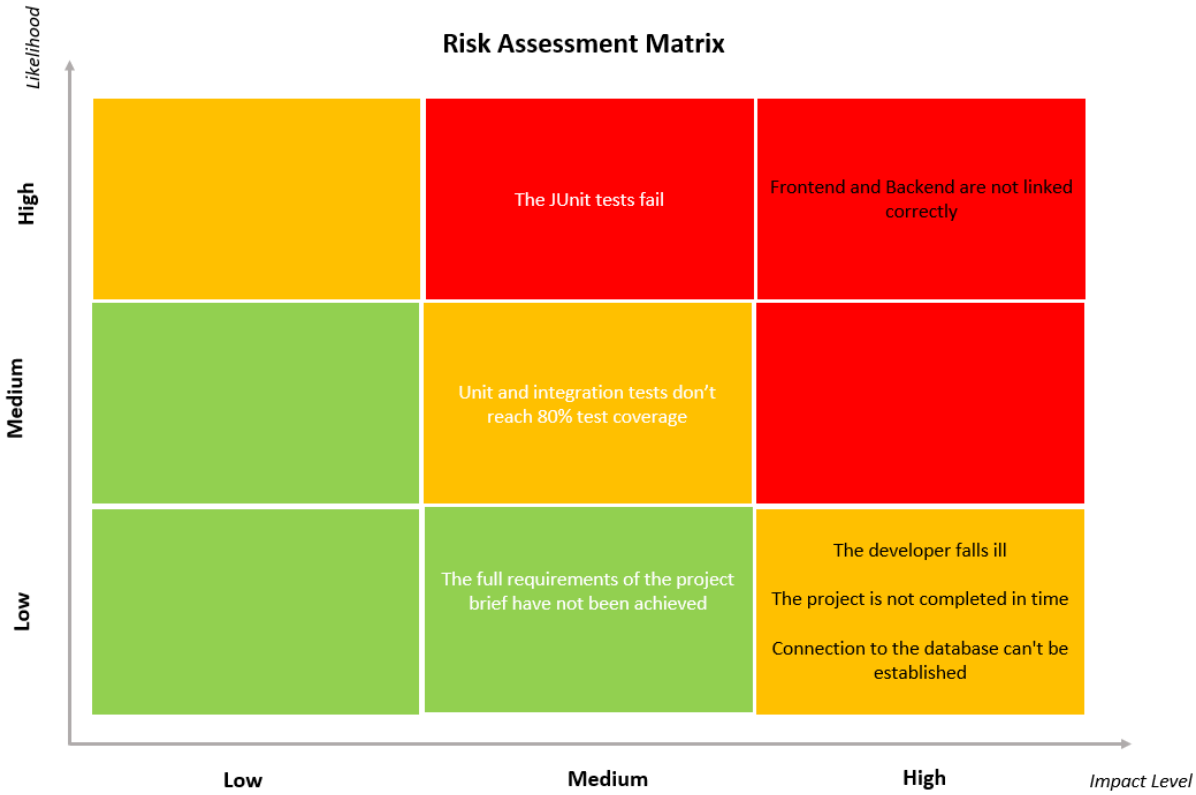
Each Epic included stories, mostly from the perspective of myself as the developer, but also some as the user. The Epics also included tasks. Each story and task were given a story point and a priority label. This will be revisited and updated as stories and tasks are completed or to create new ones.



# 1. Risk Assessment

| Risk Assessment | | | | | | |
|---|---|---|---|---|---|---|
| **Description** | **Evaluation** | **Likelihood** | **Impact Level** | **Responsibility** | **Response** | **Control Measures** |
| The developer falls ill | The project will not be completed on time | Low | High | N/A | Request an extension | Increase level of hygiene practices |
| The JUnit tests fail | The API will not work as desired | High | Medium | Developer | The code in the program and the test code need to be checked for mistakes and corrected | Ensure test code is correctly written whilst it is being written and test the functionality of new methods as they are written to identify mistakes in code before testing |
| Unit and integration tests don't reach 80% test coverage | Lower project mark | Medium | Medium | Developer | Contact trainers for advice to increase test coverage | Carry out self study and research to increase knowledge of testing |
| The project is not completed in time | Potential to fail the project | Low | High | Developer | Request an extension | Manage time taken on tasks appropriately and keep well organised |
| The full requirements of the project brief have not been achieved | Lower project mark | Low | Medium | Developer | Read project brief to identify missed requirement and inckude it in your project submission ASAP | Create stories and tasks on the project jiraboard based on each project brief requirement and set these to highest priority |
| Connection to the database can't be established | The CRUD requests will fail | Low | High | Developer | View recorded lectures to follow step-by-step instructions to identify mistake, or contact trainers for guidance | Solidify knowledge on connecting the database or follow step-by-step tutorial in recorded lectures if unsure |
| Frontend and Backend are not linked correctly | The user will not be able to access the backend functionality | High | High | Developer | View recorded lectures to follow step-by-step instructions to identify mistake, or contact trainers for guidance | Solidify knowledge on linking the Frontend and Backend or follow step-by-step tutorial in recorded lectures if unsure |

# 2. Risk Assessment Matrix



# 3. Setting up a Spring Project and GitHub Repository

MINGW64:/c/Users/User/BAE NSAC/Reading List Project

```
User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (master)
$ git branch
* master

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (master)
$ git branch -M main

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (main)
$ git branch
* main

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (main)
$ git remote add origin https://github.com/AsawerKhan/ReadingListProject.git

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (main)
$ git remote -v
origin  https://github.com/AsawerKhan/ReadingListProject.git (fetch)
origin  https://github.com/AsawerKhan/ReadingListProject.git (push)

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (main)
$ git push -u origin main
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (25/25), 58.90 KiB | 5.35 MiB/s, done.
Total 25 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AsawerKhan/ReadingListProject.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (main)
$ git branch
* main

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (main)
$ git checkout -b dev
Switched to a new branch 'dev'

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (dev)
$ git branch
* dev
  main

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (dev)
$ git push -u origin dev
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/AsawerKhan/ReadingListProject/pull/new/dev
remote:
To https://github.com/AsawerKhan/ReadingListProject.git
 * [new branch]      dev -> dev
branch 'dev' set up to track 'origin/dev'.

User@DESKTOP-S58IBQ0 MINGW64 ~/BAE NSAC/Reading List Project (dev)
$ |
```

AsawerKhan / ReadingListProject Public

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

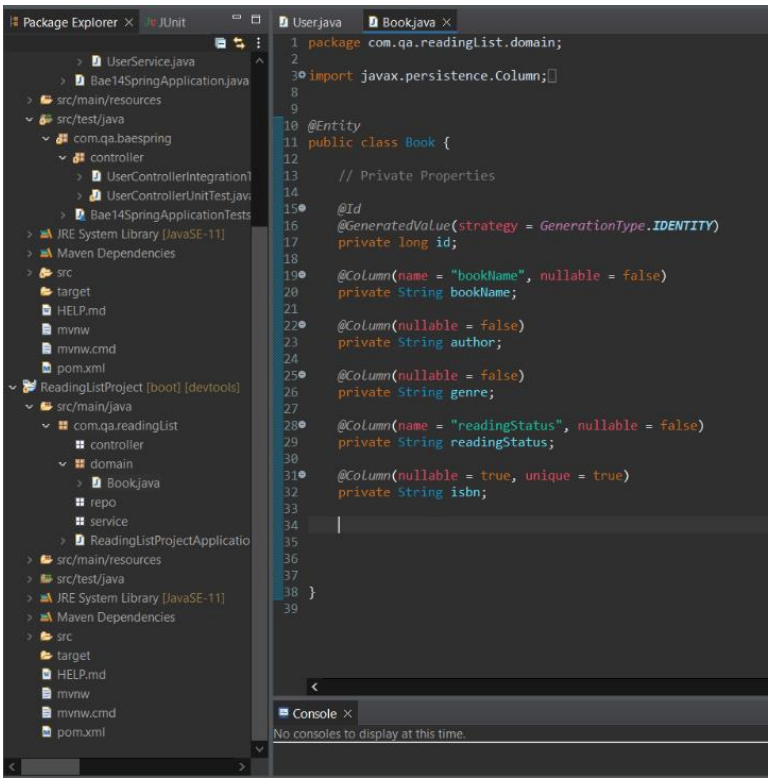
main | 2 branches | 0 tags | Go to file | Add file | Code

Asawer Khan AK: init commit 4b77ee0 1 hour ago 1 commit

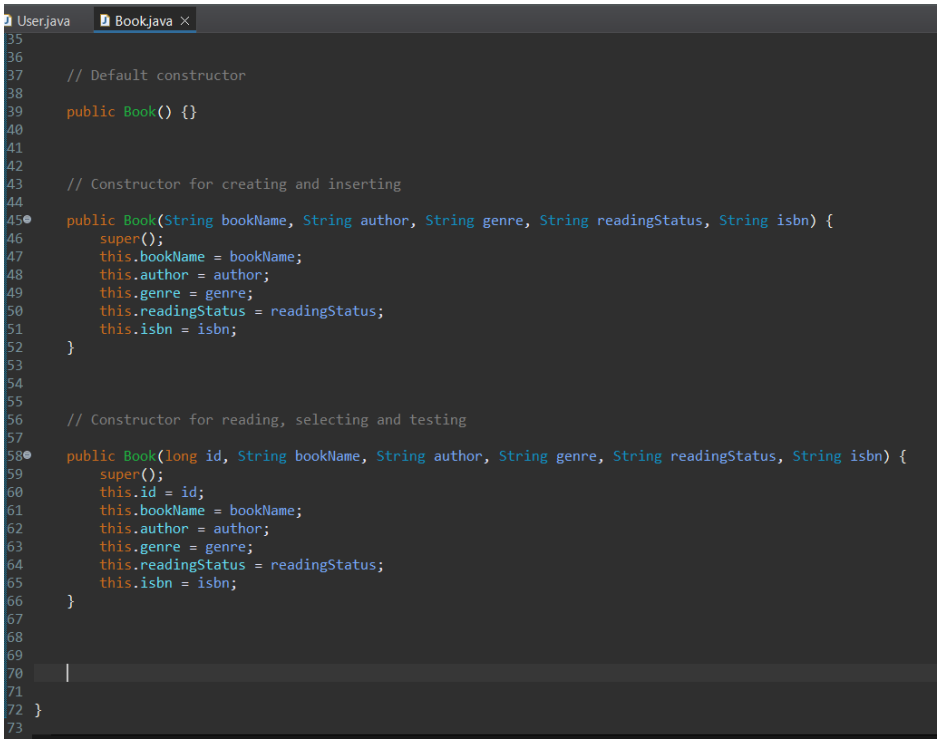
| | | |
|---|---|---|
| .mvn/wrapper | AK: init commit | 1 hour ago |
| src | AK: init commit | 1 hour ago |
| .gitignore | AK: init commit | 1 hour ago |
| mvnw | AK: init commit | 1 hour ago |
| mvnw.cmd | AK: init commit | 1 hour ago |
| pom.xml | AK: init commit | 1 hour ago |

# 4. Creating a Book Entity Class

## 4.1 Properties




## 4.2 Constructors



## 4.3 Getters and Setters

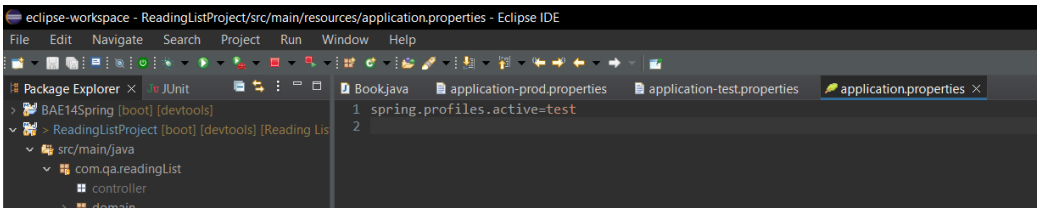
```java
User.java   *Book.java

    // Getters and setters

    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }

    public String getBookName() {
        return bookName;
    }
    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }

    public String getGenre() {
        return genre;
    }
    public void setGenre(String genre) {
        this.genre = genre;
    }

    public String getReadingStatus() {
        return readingStatus;
    }
    public void setReadingStatus(String readingStatus) {
        this.readingStatus = readingStatus;
    }

    public String getIsbn() {
```

## 4.4 toString, hashCodes and equals

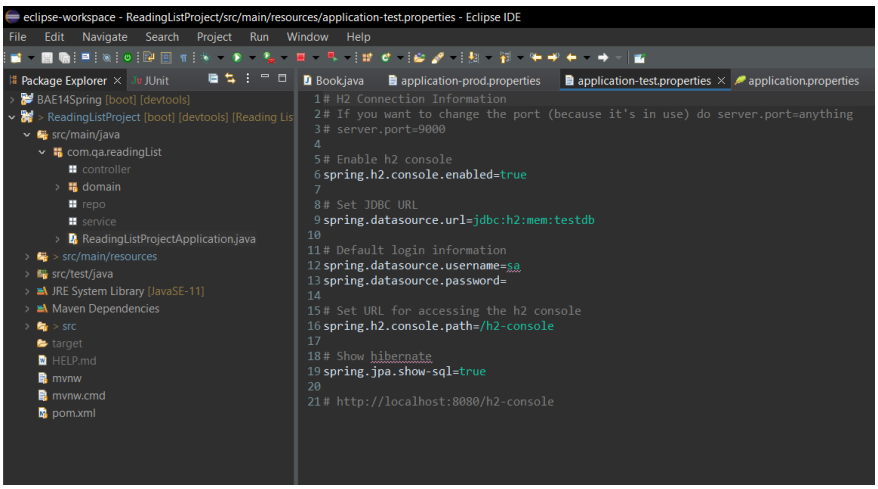
```java
User.java   Book.java
    }



    // toString

    @Override
    public String toString() {
        return "Book [id=" + id + ", bookName=" + bookName + ", author=" + author + ", genre=" + genre
                + ", readingStatus=" + readingStatus + ", isbn=" + isbn + "]";
    }


    // hashCodes and equals
        // hashCodes

    @Override
    public int hashCode() {
        return Objects.hash(author, bookName, genre, id, isbn, readingStatus);
    }


        // equals

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Book other = (Book) obj;
        return Objects.equals(author, other.author) && Objects.equals(bookName, other.bookName)
                && Objects.equals(genre, other.genre) && id == other.id && Objects.equals(isbn, other.isbn)
                && Objects.equals(readingStatus, other.readingStatus);
    }
}
```

# 5. Connecting to databases

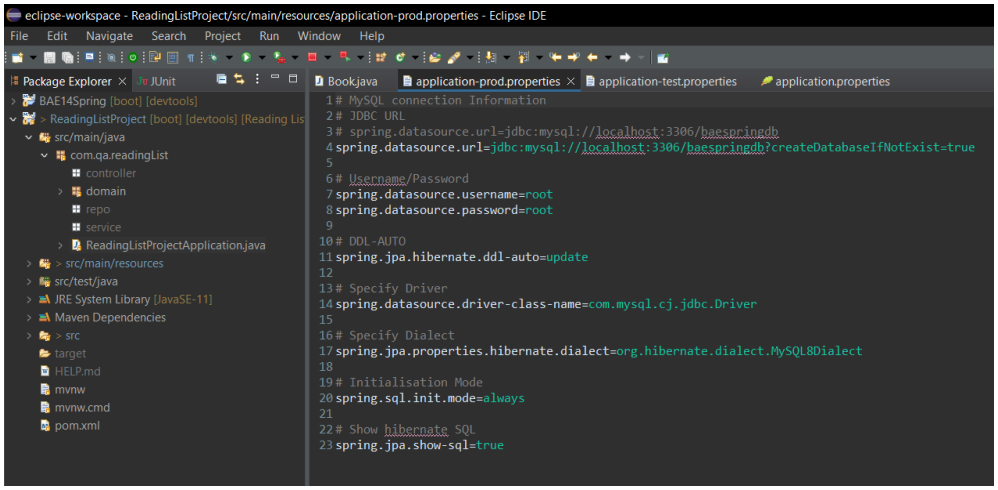
## 5.1 File to Switch Between Database Connections




## 5.2 H2 Database Connection

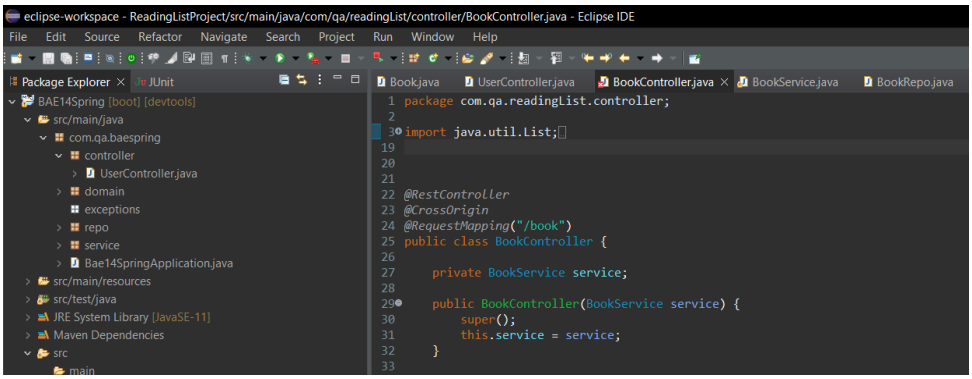



## 5.3 MySQL Database Connection

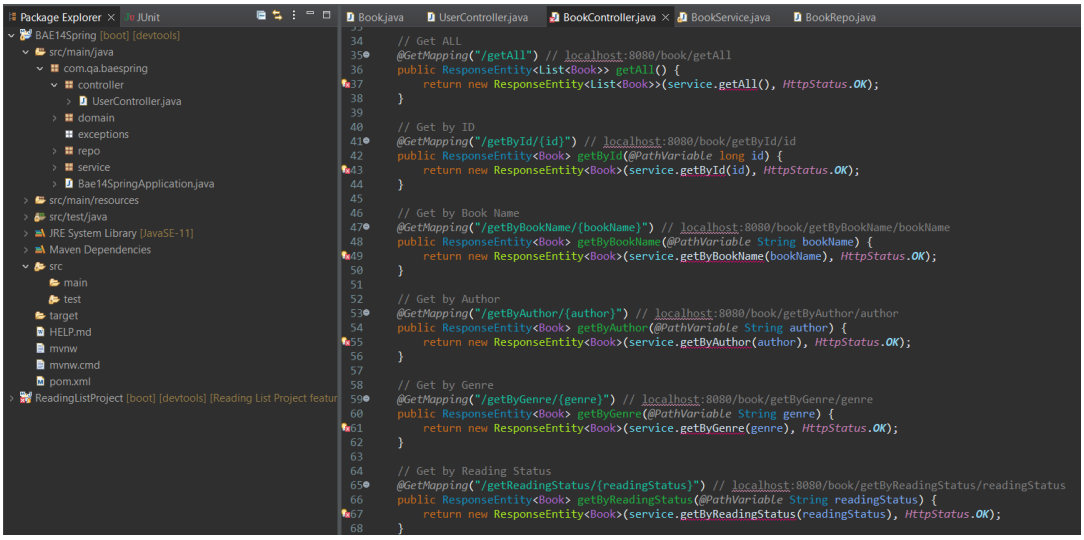


# 6. Rest API - Controller

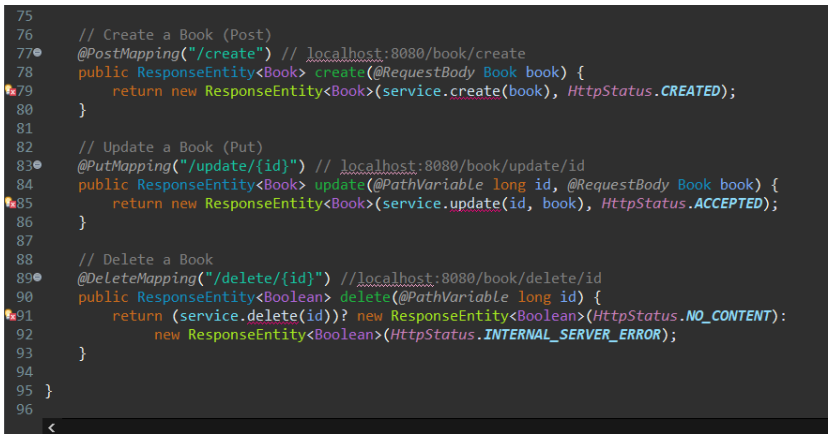
## 6.1 Constructor




## 6.2 Get Methods

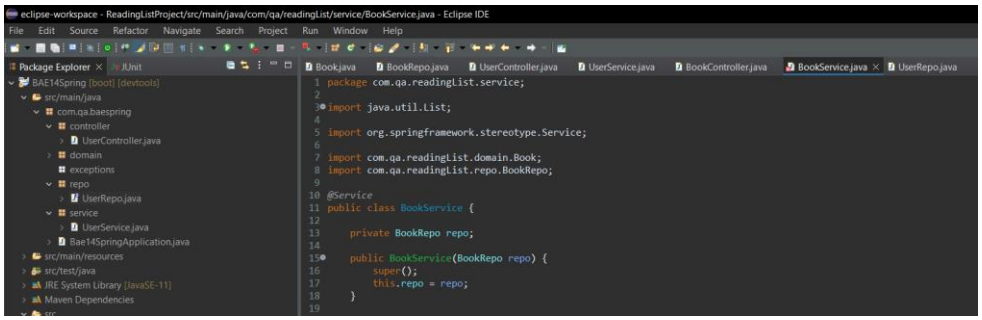



## 6.3 Create, Update and Delete Methods

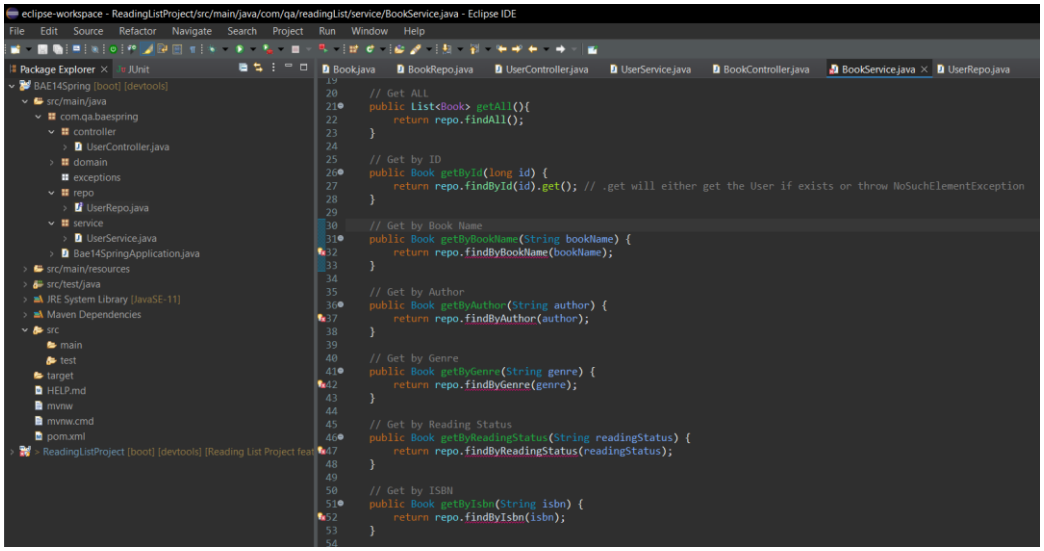


# 7. Rest API – Service

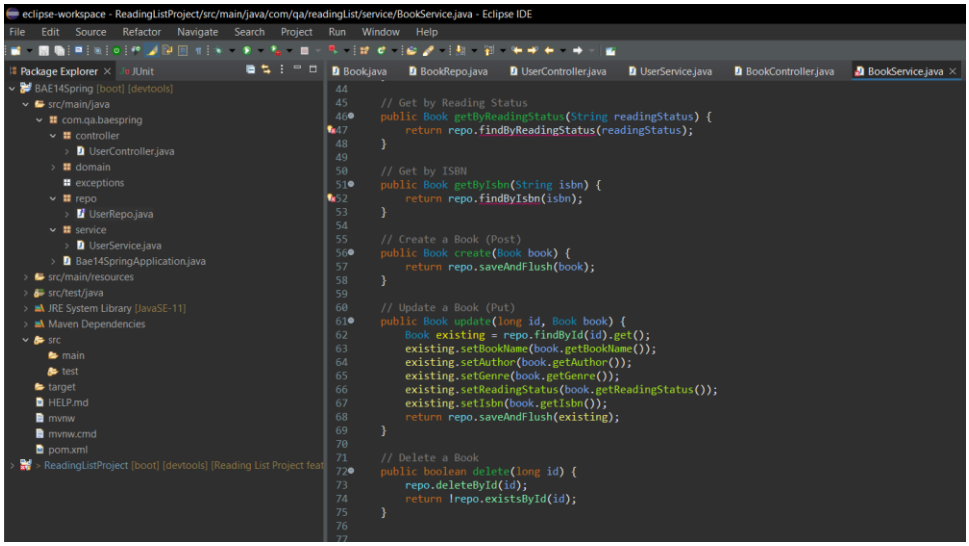
## 7.1 Constructor




## 7.2 Get Methods

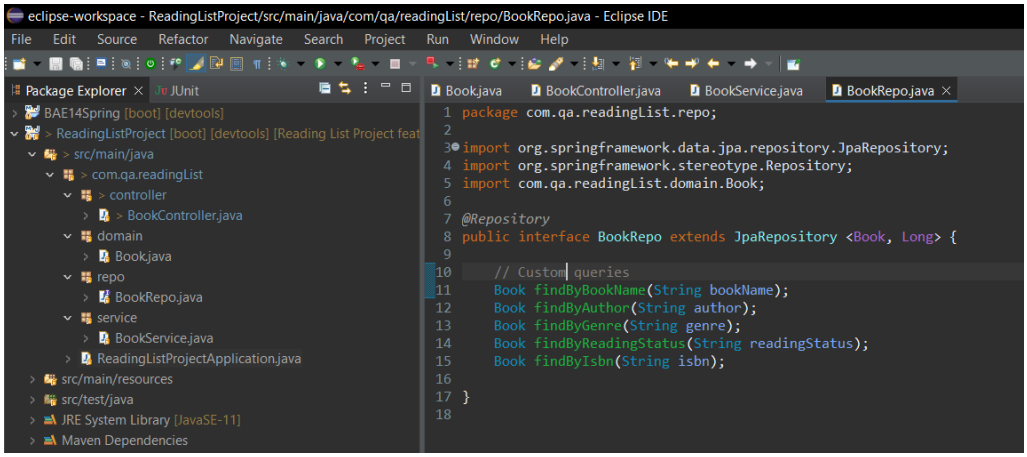



## 7.3 Create, Update and Delete Methods



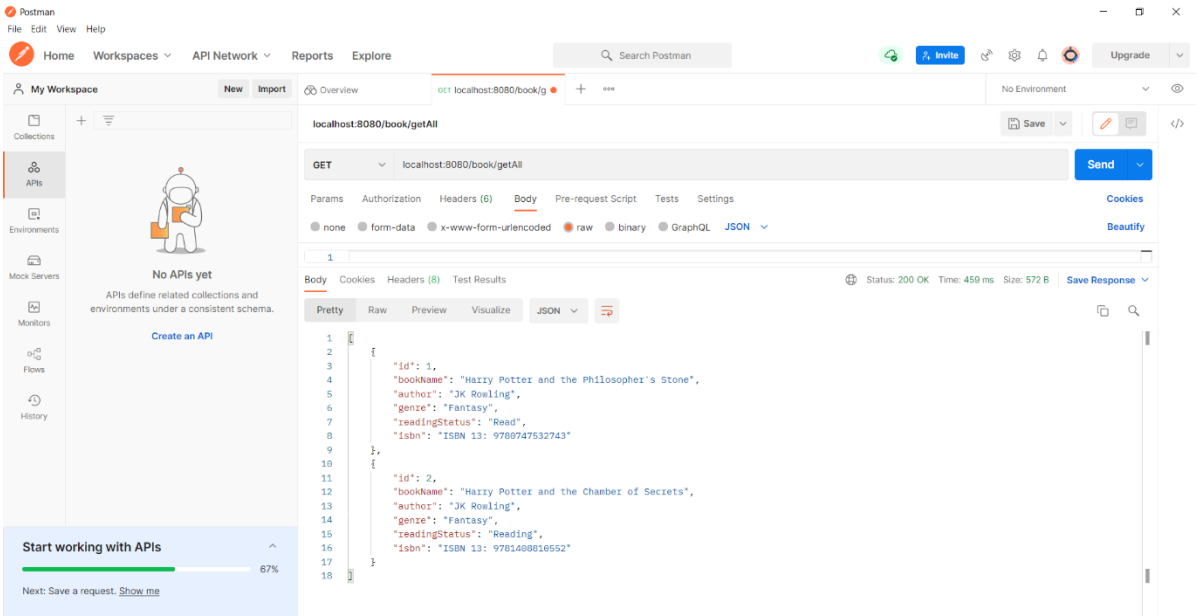
# 8. Rest API – Repo

## 8.1 Customer Queries




# 9. Postman

Checked all queries using Postman to ensure they worked.



# 10. Integration Test

The following tests were created, were ran as a Junit test, failed initially due to error in casing of a word not matching with that in testdata.sql file. This error was amended and the test passed.

## 10.1 Test for Create

```
gList/controller/BookControllerIntegrationTest.java - Eclipse IDE
Run  Window  Help
BookController.java  Book.java  BookControllerUnitTest.java  UserControllerUnitTest.java  BookControllerIntegrationTest.java
1  package com.qa.readingList.controller;
2
3  import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.delete;
25
26 @SpringBootTest
27 @AutoConfigureMockMvc
28 @Sql(scripts = {"classpath:testschema.sql", "classpath:testdata.sql"}, executionPhase = ExecutionPhase.BEFORE_TEST_METHOD)
29 @ActiveProfiles("test")
30 public class BookControllerIntegrationTest {
31
32     @Autowired
33     private MockMvc mvc;
34
35     @Autowired
36     private ObjectMapper mapper;
37
38     // Create
39     @Test
40     public void createTest() throws Exception {
41         Book entry = new Book("The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
42         String entryAsJSON = mapper.writeValueAsString(entry);
43
44         Book result = new Book(2L, "The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
45         String resultAsJSON = mapper.writeValueAsString(result);
46
47         mvc.perform(post("/book/create")
48                 .contentType(MediaType.APPLICATION_JSON)
49                 .content(entryAsJSON))
50                 .andExpect(status().isCreated())
51                 .andExpect(content().json(resultAsJSON));
52     }
```

## 10.2 Test for getAll

```
54     // getAll
55     @Test
56     public void getAllTest() throws Exception {
57         Book book = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
58         List<Book> output = new ArrayList<>();
59         output.add(book);
60         String outputAsJSON = mapper.writeValueAsString(output);
61
62         mvc.perform(get("/book/getAll")
63                 .contentType(MediaType.APPLICATION_JSON))
64                 .andExpect(status().isOk())
65                 .andExpect(content().json(outputAsJSON));
66     }
```

## 10.3 Test for getByID

```
68     // getByID
69     @Test
70     public void getByIdTest() throws Exception {
71         Book entry = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
72         String entryAsJSON = this.mapper.writeValueAsString(entry);
73
74         mvc.perform(get("/book/getById/1")
75             .contentType(MediaType.APPLICATION_JSON))
76             .andExpect(status().isOk())
77             .andExpect(content().json(entryAsJSON));
78     }
```

## 10.4 Test for Update

```
80     // Update
81     @Test
82     public void updateTest() throws Exception {
83         Book entry = new Book("The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
84         Book result = new Book(1L, "The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
85         String entryAsJSON = this.mapper.writeValueAsString(entry);
86         String resultAsJSON = this.mapper.writeValueAsString(result);
87
88         mvc.perform(put("/book/update/1")
89             .contentType(MediaType.APPLICATION_JSON)
90             .content(entryAsJSON))
91             .andExpect(status().isAccepted())
92             .andExpect(content().json(resultAsJSON));
93     }
```

## 10.5 Test for Delete

```
94
95     // Delete
96     @Test
97     public void deleteTest() throws Exception {
98         mvc.perform(delete("/book/delete/1")
99             .contentType(MediaType.APPLICATION_JSON))
100            .andExpect(status().isNoContent());
101    }
102 }
```

# 11. Unit Test – Controller

## 11.1 Test for getAll

```java
    // getAll
    @Test
    public void getAllTest() throws Exception {
        Book book = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
        List<Book> output = new ArrayList<>();
        output.add(book);
        String outputAsJSON = mapper.writeValueAsString(output);

        Mockito.when(this.service.getAll()).thenReturn(output);

        mvc.perform(get("/book/getAll")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(content().json(outputAsJSON));
    }
```

## 11.2 Test for getByID

```java
    // getByID
    @Test
    public void getByIdTest() throws Exception {
        Book entry = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
        String entryAsJSON = this.mapper.writeValueAsString(entry);

        Mockito.when(this.service.getById(1L)).thenReturn(entry);

        mvc.perform(get("/book/getById/1")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk())
            .andExpect(content().json(entryAsJSON));
    }
```

## 11.3 Test for Create

```java
    // Create
    @Test
    void createTest() throws Exception {
        Book entry = new Book("The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
        String entryAsJSON = this.mapper.writeValueAsString(entry);

        Mockito.when(this.service.create(entry)).thenReturn(entry);

        mvc.perform(post("/book/create")
            .contentType(MediaType.APPLICATION_JSON)
            .content(entryAsJSON))
            .andExpect(status().isCreated())
            .andExpect(content().json(entryAsJSON));
    }
```

## 11.4 Test for Update

```java
    // Update
    @Test
    public void updateTest() throws Exception {
        Book entry = new Book("The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
        String entryAsJSON = this.mapper.writeValueAsString(entry);

        Mockito.when(this.service.update(1L, entry)).thenReturn(entry);

        mvc.perform(put("/book/update/1")
            .contentType(MediaType.APPLICATION_JSON)
            .content(entryAsJSON))
            .andExpect(status().isAccepted())
            .andExpect(content().json(entryAsJSON));
    }
```

## 11.5 Test for Delete

```java
    // Delete
    @Test
    public void deleteTest() throws Exception {
        Mockito.when(this.service.delete(1L)).thenReturn(true);

        mvc.perform(delete("/book/delete/1")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isNoContent());
    }


    @Test
    public void deleteFailTest() throws Exception {
        Mockito.when(this.service.delete(2L)).thenReturn(false);

        mvc.perform(delete("/book/delete/5")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isInternalServerError());
    }
}
```

# 12. Unit Test – Service

## 12.1 Test for getAll

adingList/service/BookServiceUnitTest.java - Eclipse IDE

Run Window Help

BookController.java | Book.java | BookControllerUnitTest.java | BookControllerIntegrationTest.java | BookServiceUnitTest.java

```java
25●     @Test
26      public void getAllTest() {
27          List<Book> output = new ArrayList<>();
28          output.add(new Book("Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323"));
29
30          Mockito.when(this.repo.findAll()).thenReturn(output);
31
32          assertEquals(output, this.service.getAll());
33
34          Mockito.verify(this.repo, Mockito.times(1)).findAll();
35      }
36
```

## 12.2 Test for getByID

```java
37●     @Test
38      public void getByIdTest() {
39          Optional<Book> OptionalOutput = Optional.of(new Book(1L,"Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323"
40          Book output = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
41
42          Mockito.when(this.repo.findById(1L)).thenReturn(OptionalOutput);
43
44          assertEquals(output, this.service.getById(1L));
45
46          Mockito.verify(this.repo, Mockito.times(1)).findById(1L);
47      }
48
```

## 12.3 Test for Create

```java
49●     @Test
50      public void createTest() {
51          Book input = new Book("Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
52          Book output = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
53
54          Mockito.when(this.repo.saveAndFlush(input)).thenReturn(output);
55
56          assertEquals(output, this.service.create(input));
57
58          Mockito.verify(this.repo, Mockito.times(1)).saveAndFlush(input);
59      }
60
```

## 12.4 Test for Update

```java
61●     @Test
62      public void updateTest() {
63          Book input = new Book("Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
64          Optional<Book> existing = Optional.of(new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323"));
65          Book output = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
66
67          Mockito.when(this.repo.findById(1L)).thenReturn(existing);
68          Mockito.when(this.repo.saveAndFlush(output)).thenReturn(output);
69
70          assertEquals(output, this.service.update(1L, input));
71
72          Mockito.verify(this.repo, Mockito.times(1)).findById(1L);
73          Mockito.verify(this.repo, Mockito.times(1)).saveAndFlush(output);
74      }
```

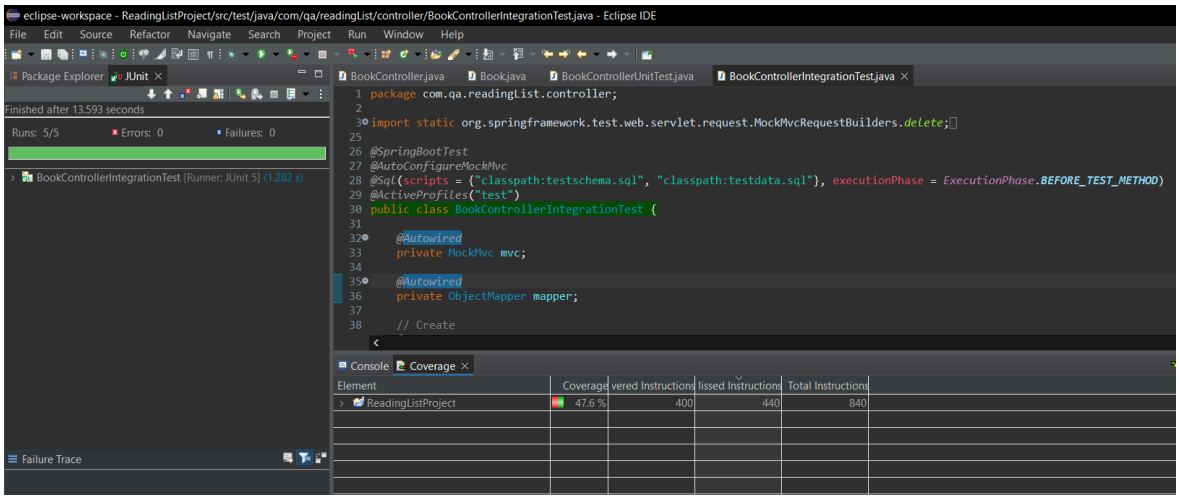
## 12.5 Test for Delete

```java
76●     @Test
77      public void deleteTest() {
78          Mockito.when(this.repo.existsById(1L)).thenReturn(false);
79
80          assertTrue(this.service.delete(1L));
81
82          Mockito.verify(this.repo, Mockito.times(1)).deleteById(1L);
83          Mockito.verify(this.repo, Mockito.times(1)).existsById(1L);
84      }
85
```

# 13.Test Coverage

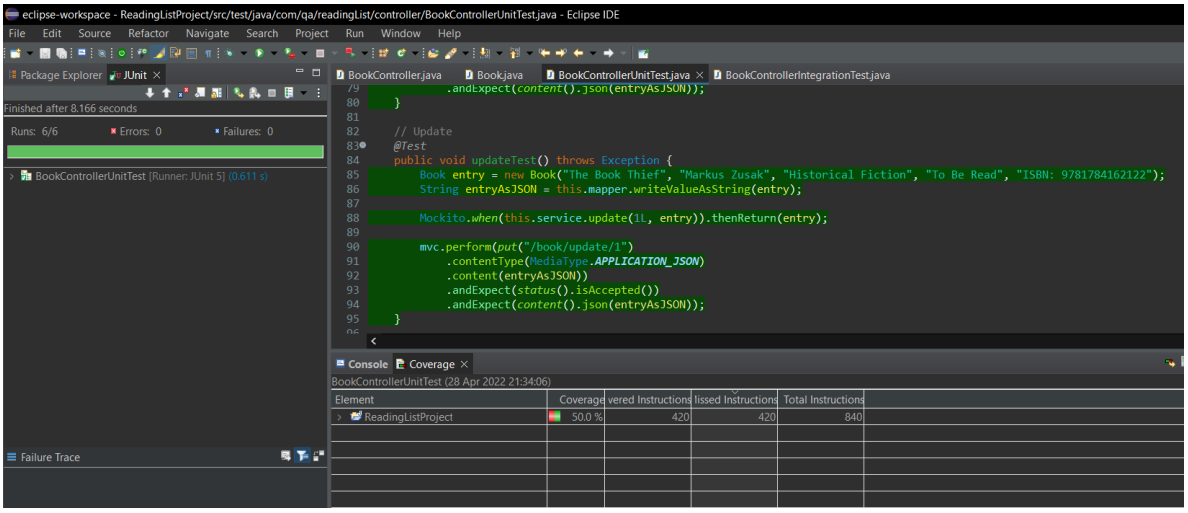
## 13.1 Integration Test

The integration test coverage for the project is at 47.6%, below the industry standard of 80%.



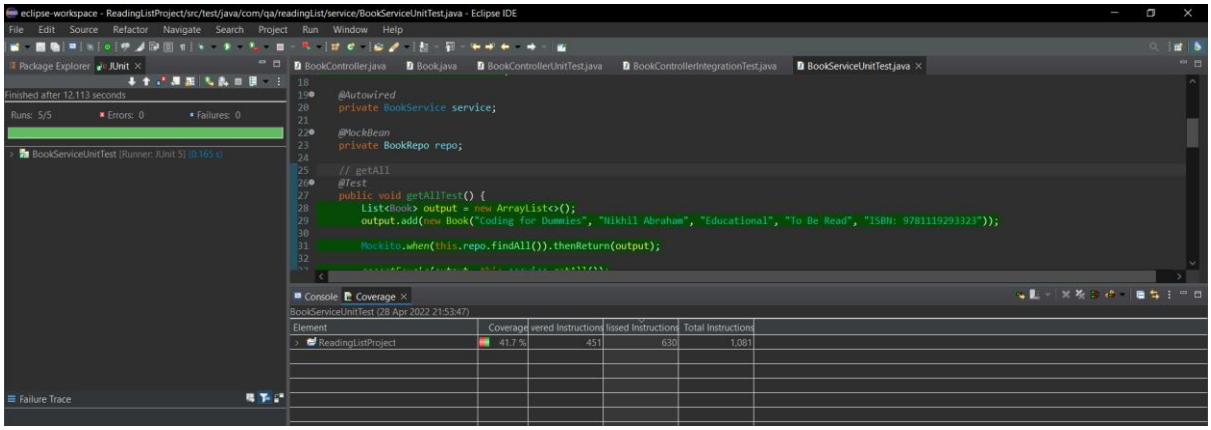

## 12.2 Unit Test – Controller

The unit test coverage for the controller is at 50%, below the industry standard of 80%.




## 13.3 Unit Test – Service

The unit test coverage for the service is at 41.7%, below the industry standard of 80%.



# 14. Front-End

## 14.1 Planning

Front End Planning

28 April 2022 22:41