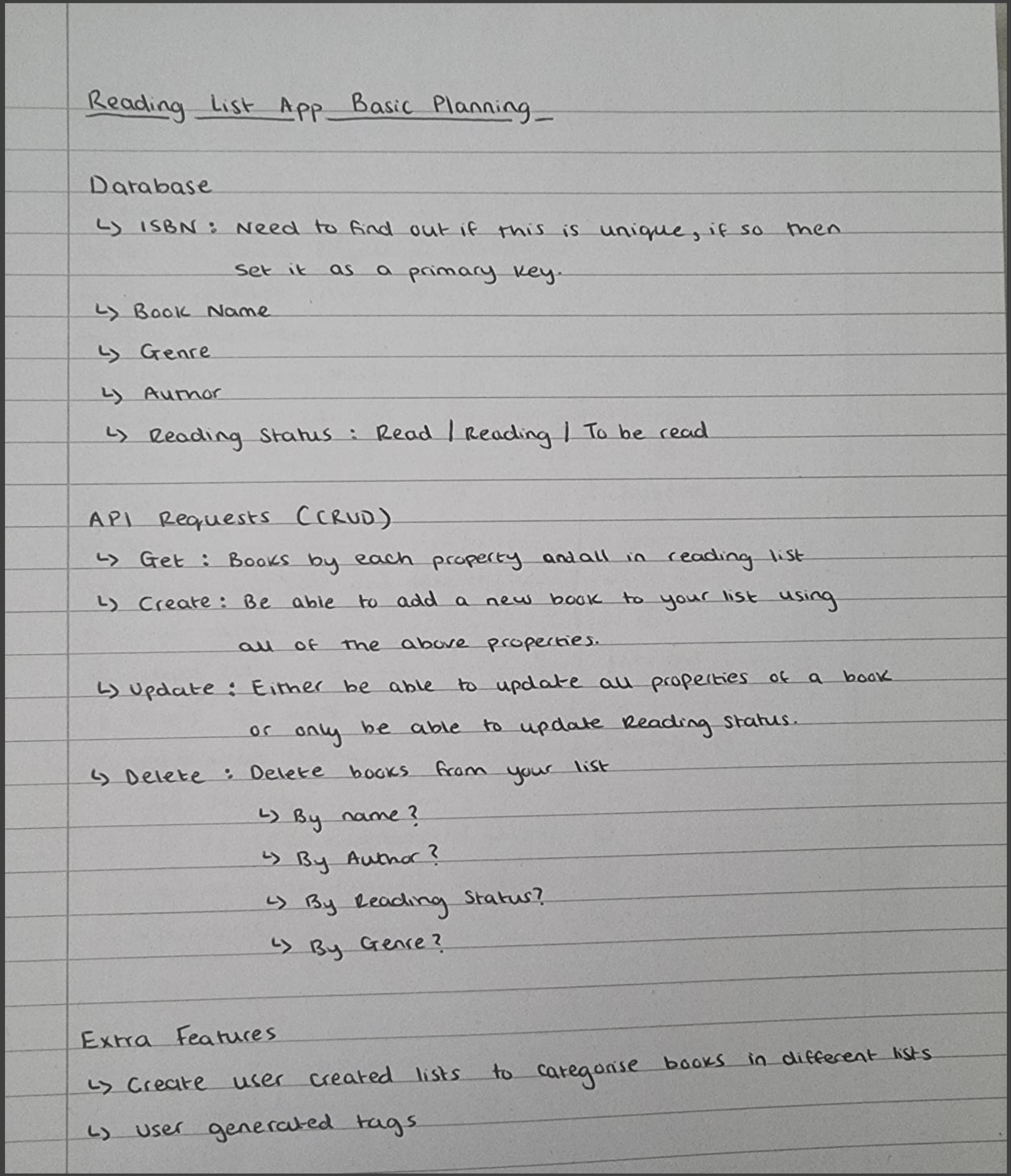
Reading List App


Reading List App Basic Planning

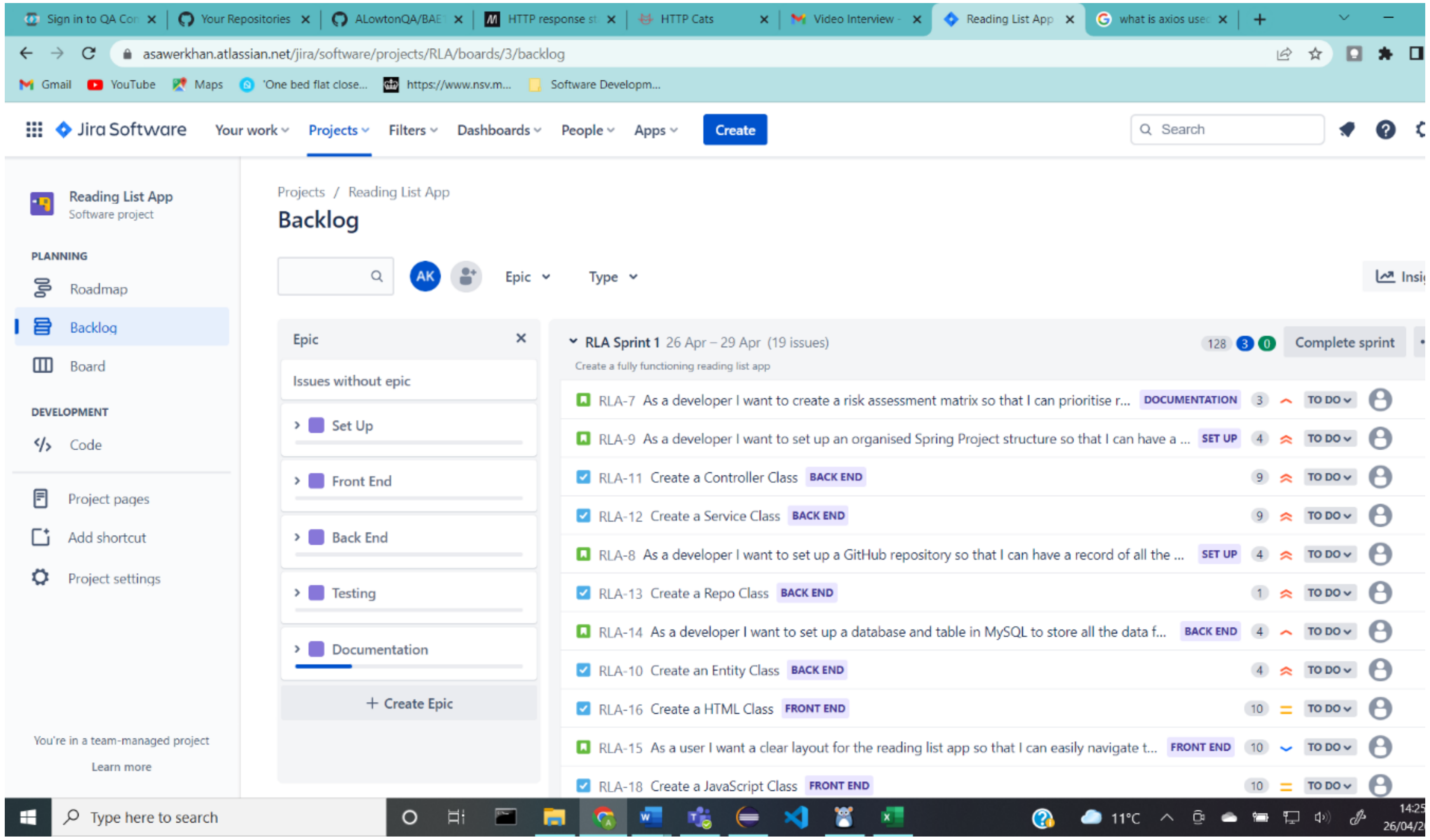
Database
↳ ISBN : Need to find out if this is unique, if so then set it as a primary key.
↳ Book Name
↳ Genre
↳ Author
↳ Reading Status : Read / Reading / To be read

API Requests (CRUD)
↳ Get : Books by each property and all in reading list
↳ Create : Be able to add a new book to your list using all of the above properties.
↳ Update : Either be able to update all properties of a book or only be able to update Reading status.
↳ Delete : Delete books from your list
  ↳ By name?
  ↳ By Author?
  ↳ By Reading Status?
  ↳ By Genre?

Extra Features
↳ Create user created lists to categorise books in different lists
↳ User generated tags

Vision

Jira Board



asawerkhan.atlassian.net/jira/software/projects/RLA/boards/3/backlog

Jira Software · Your work · Projects · Filters · Dashboards · People · Apps · Create

Reading List App
Software project

PLANNING
- Roadmap
- Backlog
- Board

DEVELOPMENT
- Code
- Project pages
- Add shortcut
- Project settings

You're in a team-managed project
Learn more

Projects / Reading List App

Backlog

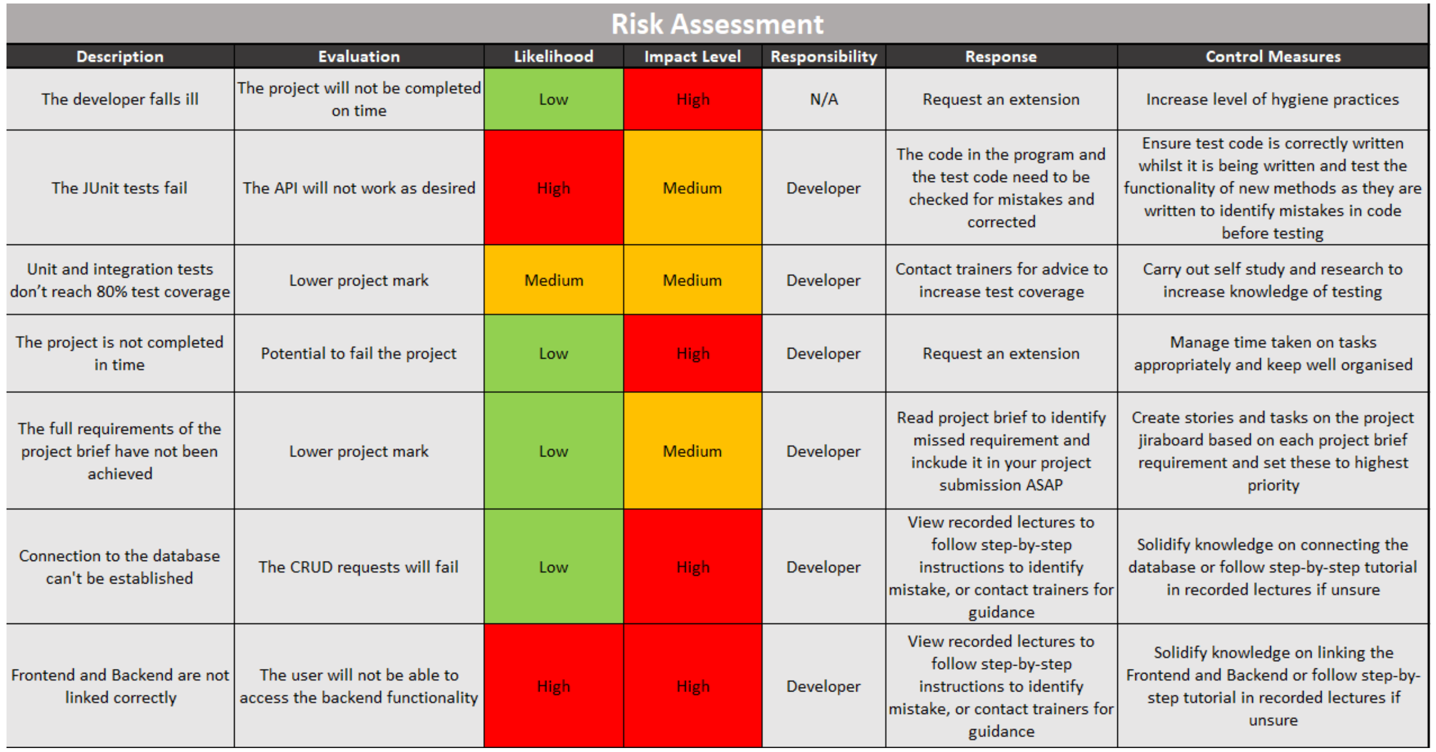
Epic
- Issues without epic
- Set Up
- Front End
- Back End
- Testing
- Documentation
- + Create Epic

RLA Sprint 1 26 Apr – 29 Apr (19 issues) · 128 · 3 · 0 · Complete sprint
Create a fully functioning reading list app

RLA-7	As a developer I want to create a risk assessment matrix so that I can prioritise r...	DOCUMENTATION	3	TO DO
RLA-9	As a developer I want to set up an organised Spring Project structure so that I can have a ...	SET UP	4	TO DO
RLA-11	Create a Controller Class	BACK END	9	TO DO
RLA-12	Create a Service Class	BACK END	9	TO DO
RLA-8	As a developer I want to set up a GitHub repository so that I can have a record of all the ...	SET UP	4	TO DO
RLA-13	Create a Repo Class	BACK END	1	TO DO
RLA-14	As a developer I want to set up a database and table in MySQL to store all the data f...	BACK END	4	TO DO
RLA-10	Create an Entity Class	BACK END	4	TO DO
RLA-16	Create a HTML Class	FRONT END	10	TO DO
RLA-15	As a user I want a clear layout for the reading list app so that I can easily navigate t...	FRONT END	10	TO DO
RLA-18	Create a JavaScript Class	FRONT END	10	TO DO

- Setup
- Documentation
- Frontend
- Backend
- Testing

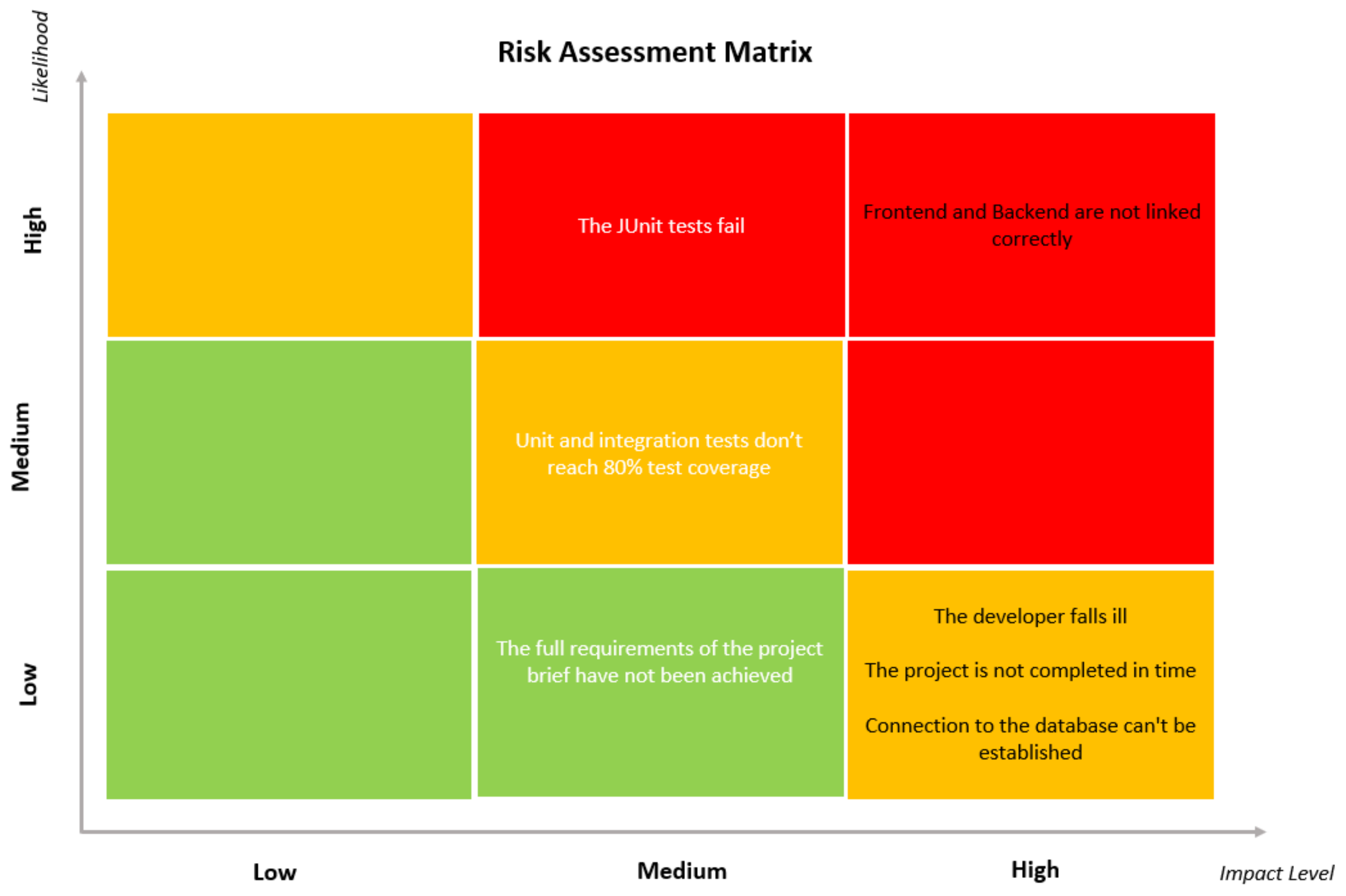
Risk Assessment



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

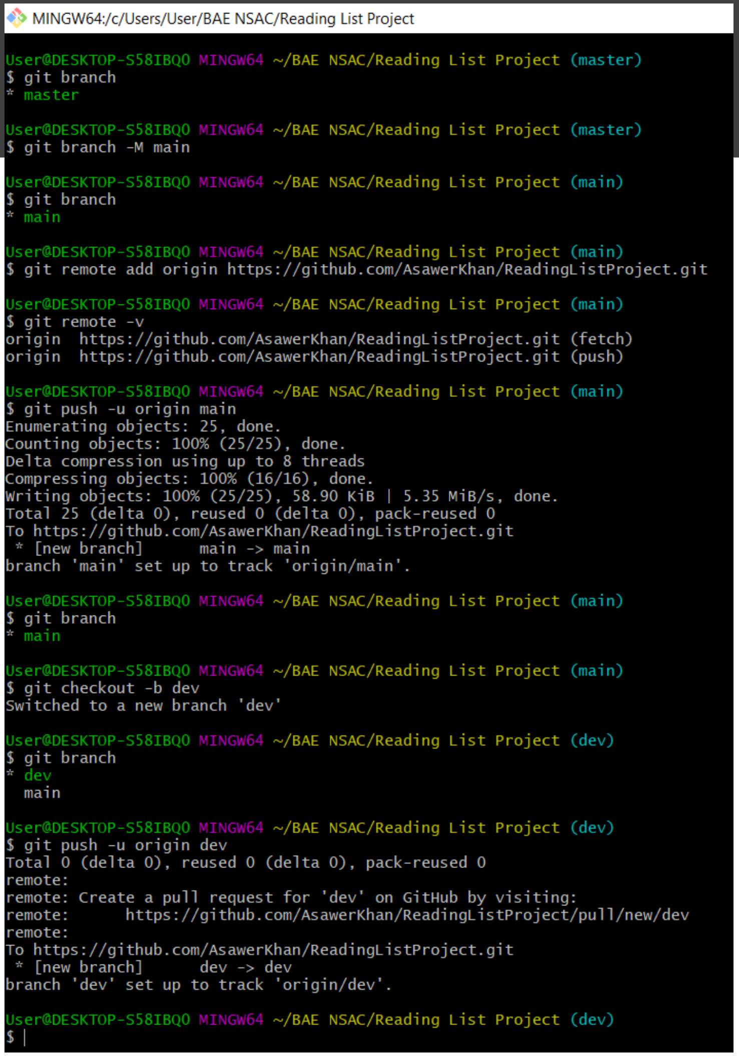
Risk Assessment Matrix



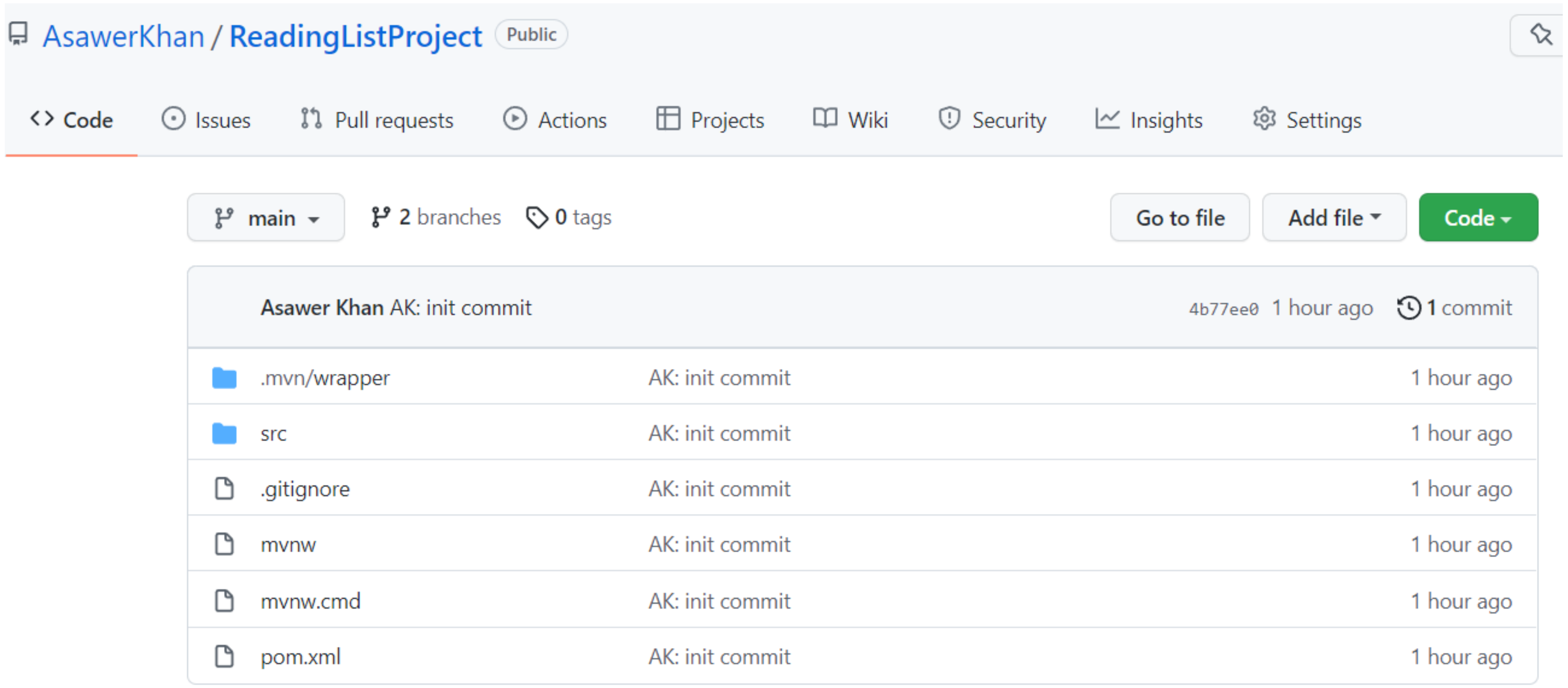
Risk Assessment Matrix

Likelihood / Impact Level	Low	Medium	High
High	🟧	The JUnit tests fail	Frontend and Backend are not linked correctly
Medium	🟩	Unit and integration tests don't reach 80% test coverage	🟥
Low	🟩	The full requirements of the project brief have not been achieved	The developer falls ill The project is not completed in time Connection to the database can't be established

Setting up a Spring Project and GitHub Repo



```
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)
$ |
```

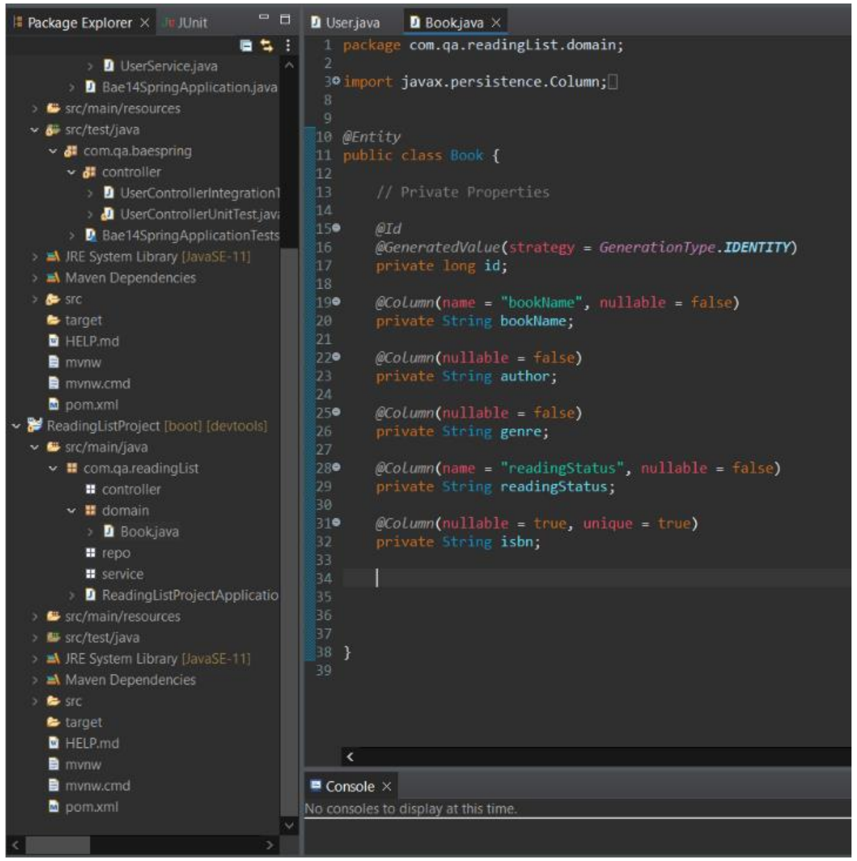


Asawer Khan AK: init commit — 4b77ee0 1 hour ago — 1 commit

.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

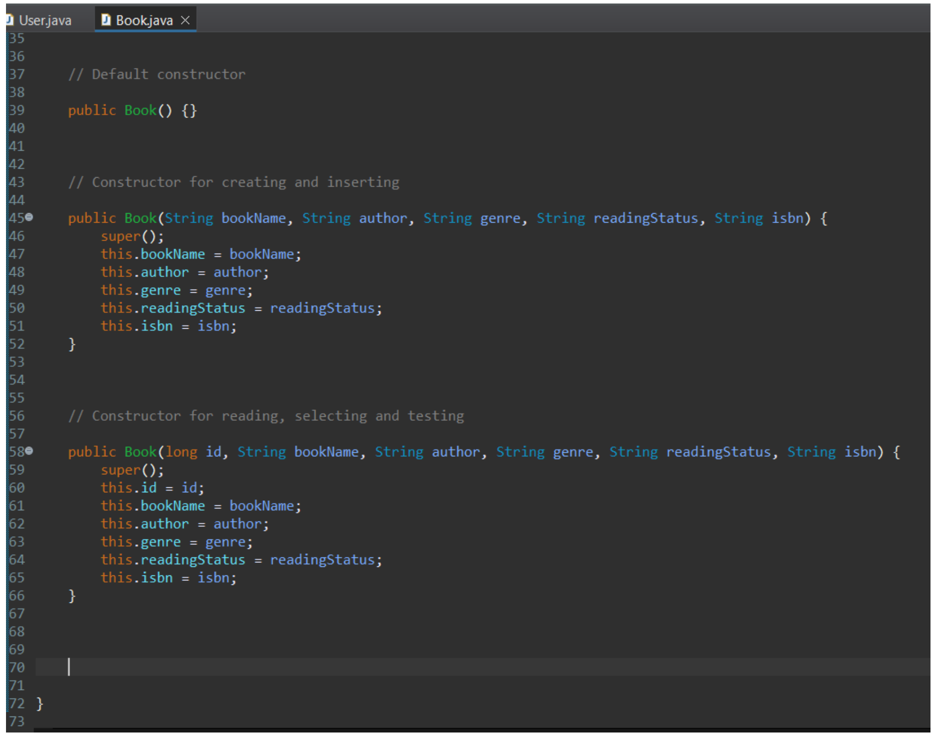
Creating a Book Entity Class

- Properties
- Constructors
- Getters and Setters
- toString
- hashCodes and equals

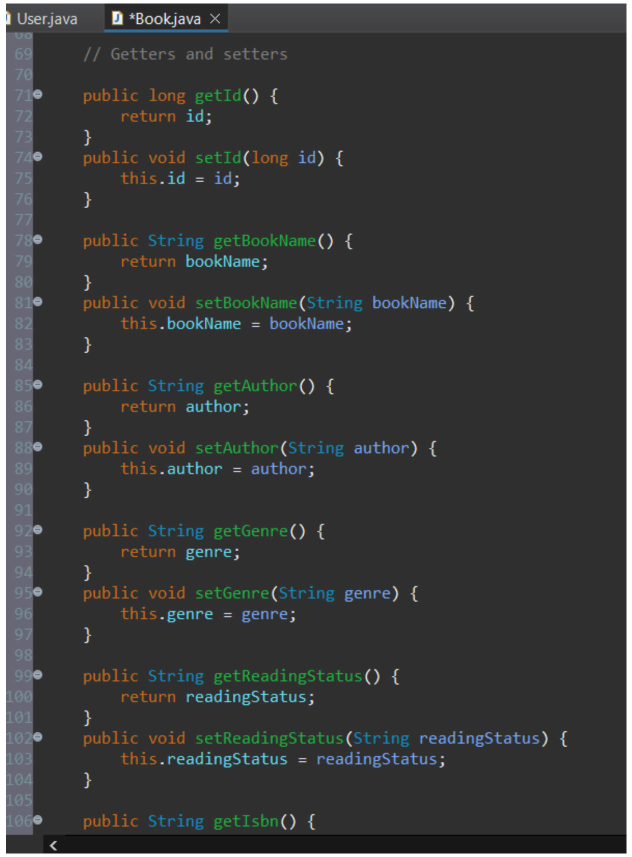


```java
package com.qa.readingList.domain;

import javax.persistence.Column;


@Entity
public class Book {

    // Private Properties

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(name = "bookName", nullable = false)
    private String bookName;

    @Column(nullable = false)
    private String author;

    @Column(nullable = false)
    private String genre;

    @Column(name = "readingStatus", nullable = false)
    private String readingStatus;

    @Column(nullable = true, unique = true)
    private String isbn;

    



}
```

```
No consoles to display at this time.
```



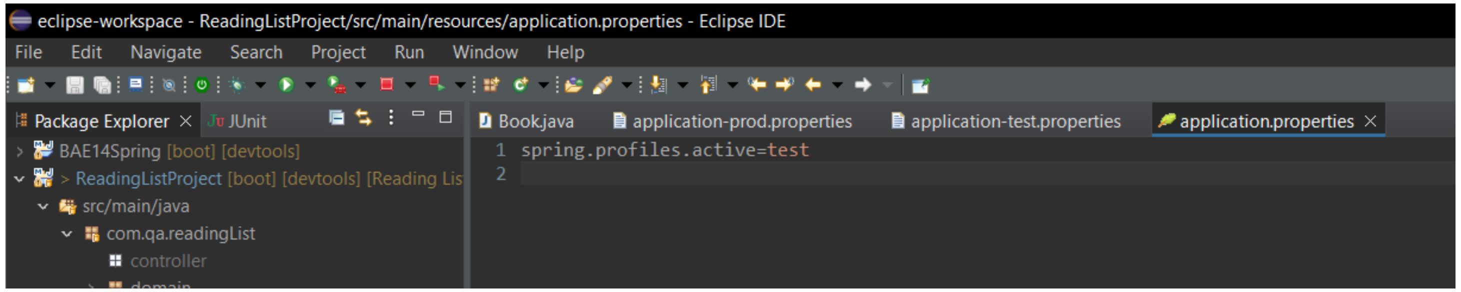
```java


    // Default constructor

    public Book() {}



    // Constructor for creating and inserting

    public Book(String bookName, String author, String genre, String readingStatus, String isbn) {
        super();
        this.bookName = bookName;
        this.author = author;
        this.genre = genre;
        this.readingStatus = readingStatus;
        this.isbn = isbn;
    }




    // Constructor for reading, selecting and testing

    public Book(long id, String bookName, String author, String genre, String readingStatus, String isbn) {
        super();
        this.id = id;
        this.bookName = bookName;
        this.author = author;
        this.genre = genre;
        this.readingStatus = readingStatus;
        this.isbn = isbn;
    }




}
```



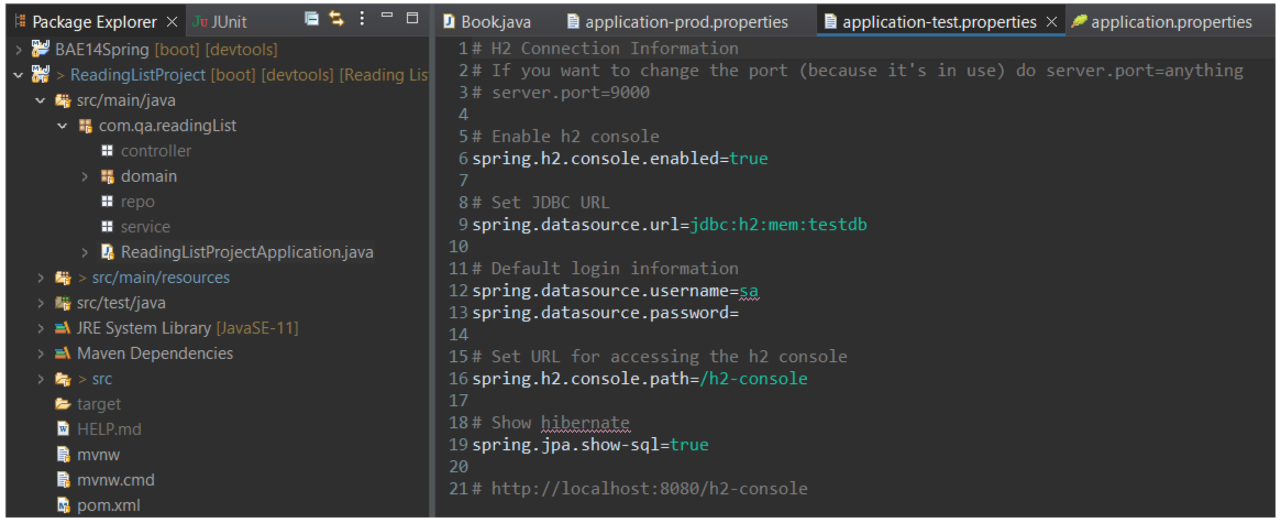
```java
    // Getters and setters

    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() {
```

Connecting to Databases

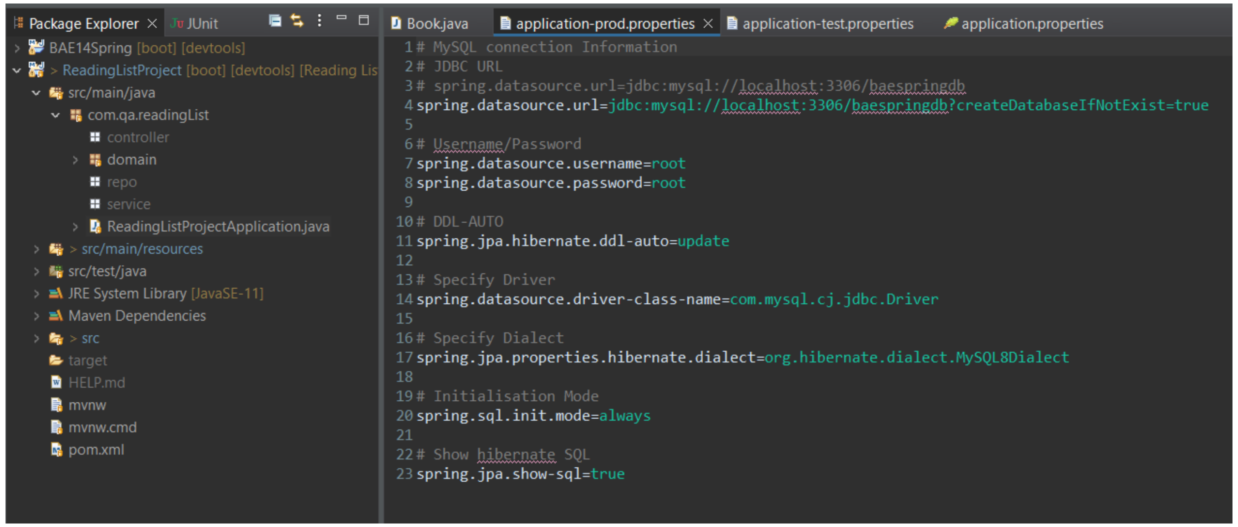
- H2 Database
- MySQL Database


eclipse-workspace - ReadingListProject/src/main/resources/application.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
JUnit
BAE14Spring [boot] [devtools]
> ReadingListProject [boot] [devtools] [Reading Lis
src/main/java
com.qa.readingList
controller
Book.java
application-prod.properties
application-test.properties
application.properties

```
spring.profiles.active=test
```


Package Explorer
JUnit
BAE14Spring [boot] [devtools]
> ReadingListProject [boot] [devtools] [Reading Lis
src/main/java
com.qa.readingList
controller
domain
repo
service
ReadingListProjectApplication.java
> src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
> src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
Book.java
application-prod.properties
application-test.properties
application.properties

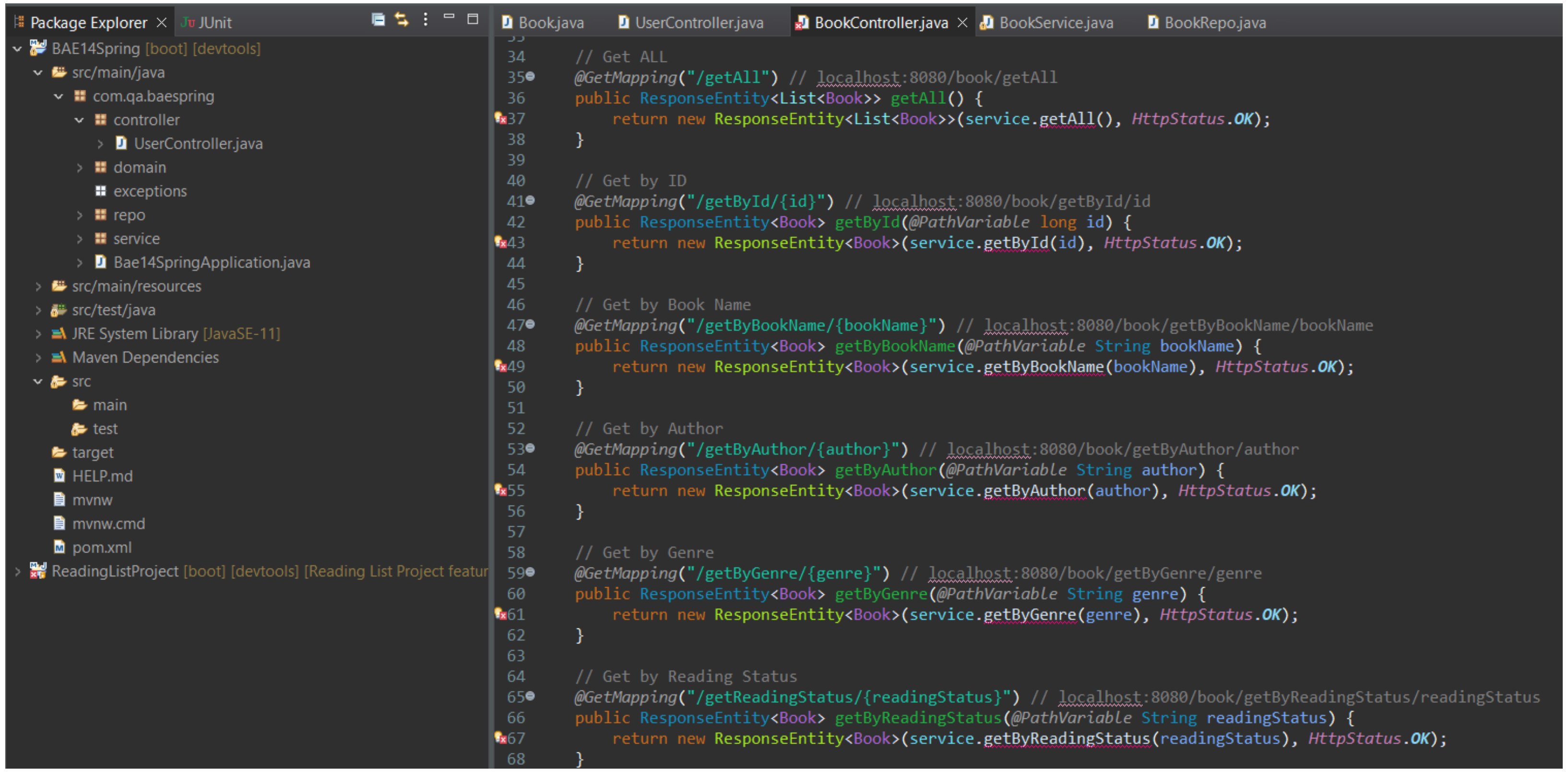
```
# H2 Connection Information
# If you want to change the port (because it's in use) do server.port=anything
# server.port=9000

# Enable h2 console
spring.h2.console.enabled=true

# Set JDBC URL
spring.datasource.url=jdbc:h2:mem:testdb

# Default login information
spring.datasource.username=sa
spring.datasource.password=

# Set URL for accessing the h2 console
spring.h2.console.path=/h2-console

# Show hibernate
spring.jpa.show-sql=true

# http://localhost:8080/h2-console
```


Package Explorer
JUnit
BAE14Spring [boot] [devtools]
> ReadingListProject [boot] [devtools] [Reading Lis
src/main/java
com.qa.readingList
controller
domain
repo
service
ReadingListProjectApplication.java
> src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
> src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
Book.java
application-prod.properties
application-test.properties
application.properties

```
# MySQL connection Information
# JDBC URL
# spring.datasource.url=jdbc:mysql://localhost:3306/baespringdb
spring.datasource.url=jdbc:mysql://localhost:3306/baespringdb?createDatabaseIfNotExist=true

# Username/Password
spring.datasource.username=root
spring.datasource.password=root

# DDL-AUTO
spring.jpa.hibernate.ddl-auto=update

# Specify Driver
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Specify Dialect
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Initialisation Mode
spring.sql.init.mode=always

# Show hibernate SQL
spring.jpa.show-sql=true
```

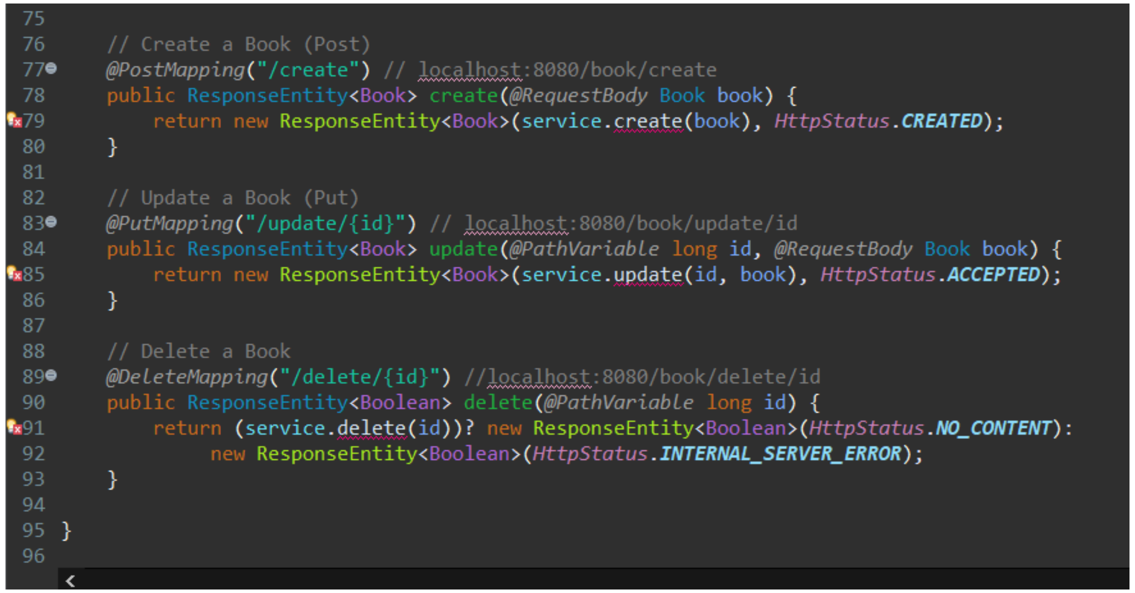
Rest API - Controller

- Constructor
- Methods
- Get
- Create
- Update
- Delete



```
BAE14Spring [boot] [devtools]
  src/main/java
    com.qa.baespring
      controller
        UserController.java
      domain
      exceptions
      repo
      service
      Bae14SpringApplication.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  src
    main
    test
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
ReadingListProject [boot] [devtools] [Reading List Project featur
```

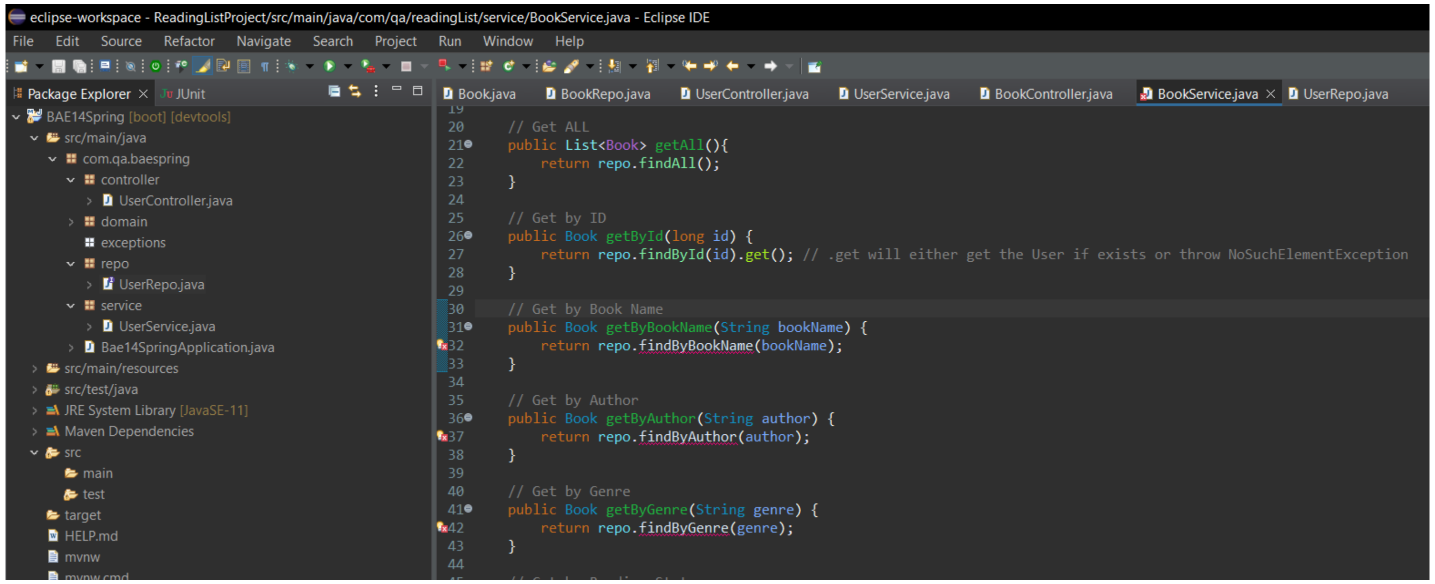
```java
    // Get ALL
    @GetMapping("/getAll") // localhost:8080/book/getAll
    public ResponseEntity<List<Book>> getAll() {
        return new ResponseEntity<List<Book>>(service.getAll(), HttpStatus.OK);
    }

    // Get by ID
    @GetMapping("/getById/{id}") // localhost:8080/book/getById/id
    public ResponseEntity<Book> getById(@PathVariable long id) {
        return new ResponseEntity<Book>(service.getById(id), HttpStatus.OK);
    }

    // Get by Book Name
    @GetMapping("/getByBookName/{bookName}") // localhost:8080/book/getByBookName/bookName
    public ResponseEntity<Book> getByBookName(@PathVariable String bookName) {
        return new ResponseEntity<Book>(service.getByBookName(bookName), HttpStatus.OK);
    }

    // Get by Author
    @GetMapping("/getByAuthor/{author}") // localhost:8080/book/getByAuthor/author
    public ResponseEntity<Book> getByAuthor(@PathVariable String author) {
        return new ResponseEntity<Book>(service.getByAuthor(author), HttpStatus.OK);
    }

    // Get by Genre
    @GetMapping("/getByGenre/{genre}") // localhost:8080/book/getByGenre/genre
    public ResponseEntity<Book> getByGenre(@PathVariable String genre) {
        return new ResponseEntity<Book>(service.getByGenre(genre), HttpStatus.OK);
    }

    // Get by Reading Status
    @GetMapping("/getReadingStatus/{readingStatus}") // localhost:8080/book/getByReadingStatus/readingStatus
    public ResponseEntity<Book> getByReadingStatus(@PathVariable String readingStatus) {
        return new ResponseEntity<Book>(service.getByReadingStatus(readingStatus), HttpStatus.OK);
    }
```



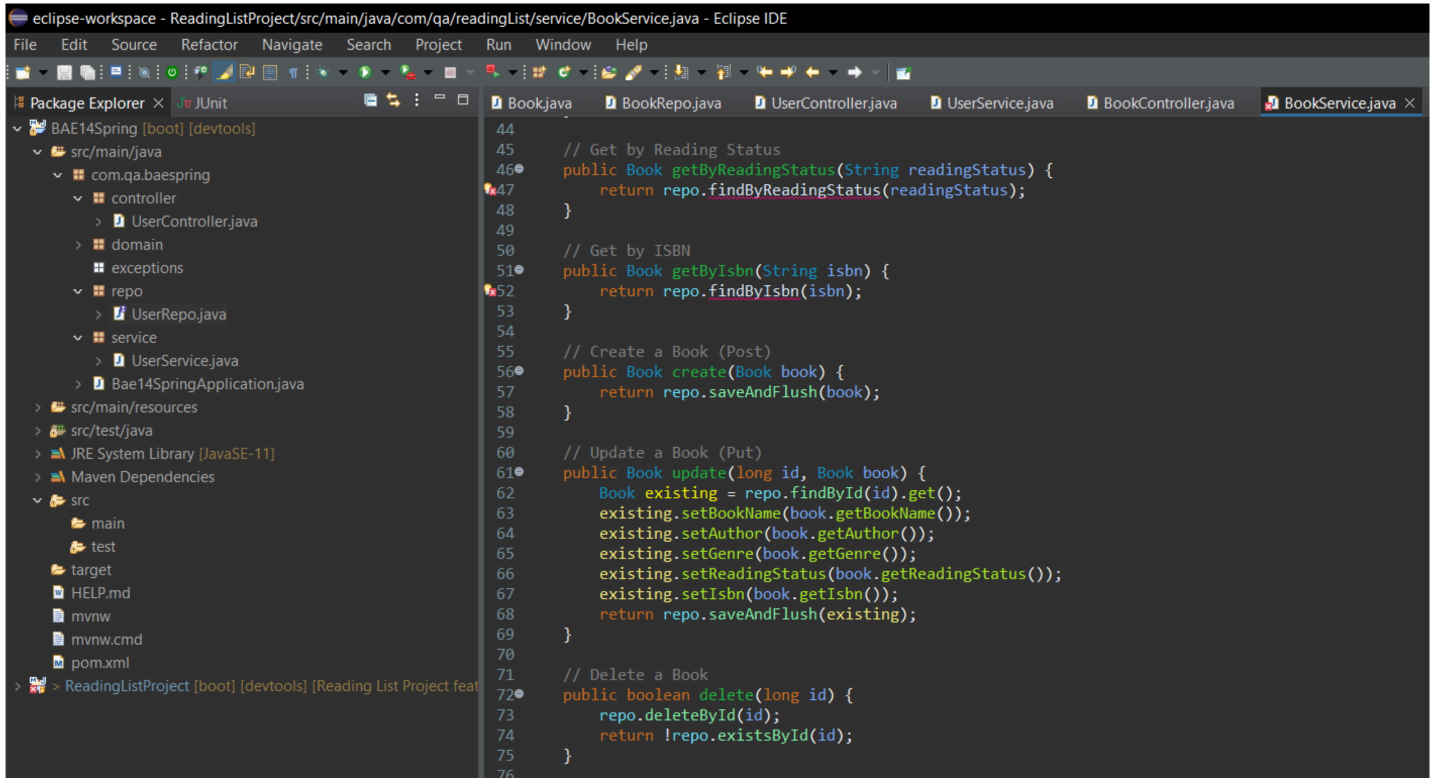
```java

    // Create a Book (Post)
    @PostMapping("/create") // localhost:8080/book/create
    public ResponseEntity<Book> create(@RequestBody Book book) {
        return new ResponseEntity<Book>(service.create(book), HttpStatus.CREATED);
    }

    // Update a Book (Put)
    @PutMapping("/update/{id}") // localhost:8080/book/update/id
    public ResponseEntity<Book> update(@PathVariable long id, @RequestBody Book book) {
        return new ResponseEntity<Book>(service.update(id, book), HttpStatus.ACCEPTED);
    }

    // Delete a Book
    @DeleteMapping("/delete/{id}") //localhost:8080/book/delete/id
    public ResponseEntity<Boolean> delete(@PathVariable long id) {
        return (service.delete(id))? new ResponseEntity<Boolean>(HttpStatus.NO_CONTENT):
            new ResponseEntity<Boolean>(HttpStatus.INTERNAL_SERVER_ERROR);
    }

}

```

Rest API - Service

- Constructor
- Methods
- Get
- Create
- Update
- Delete



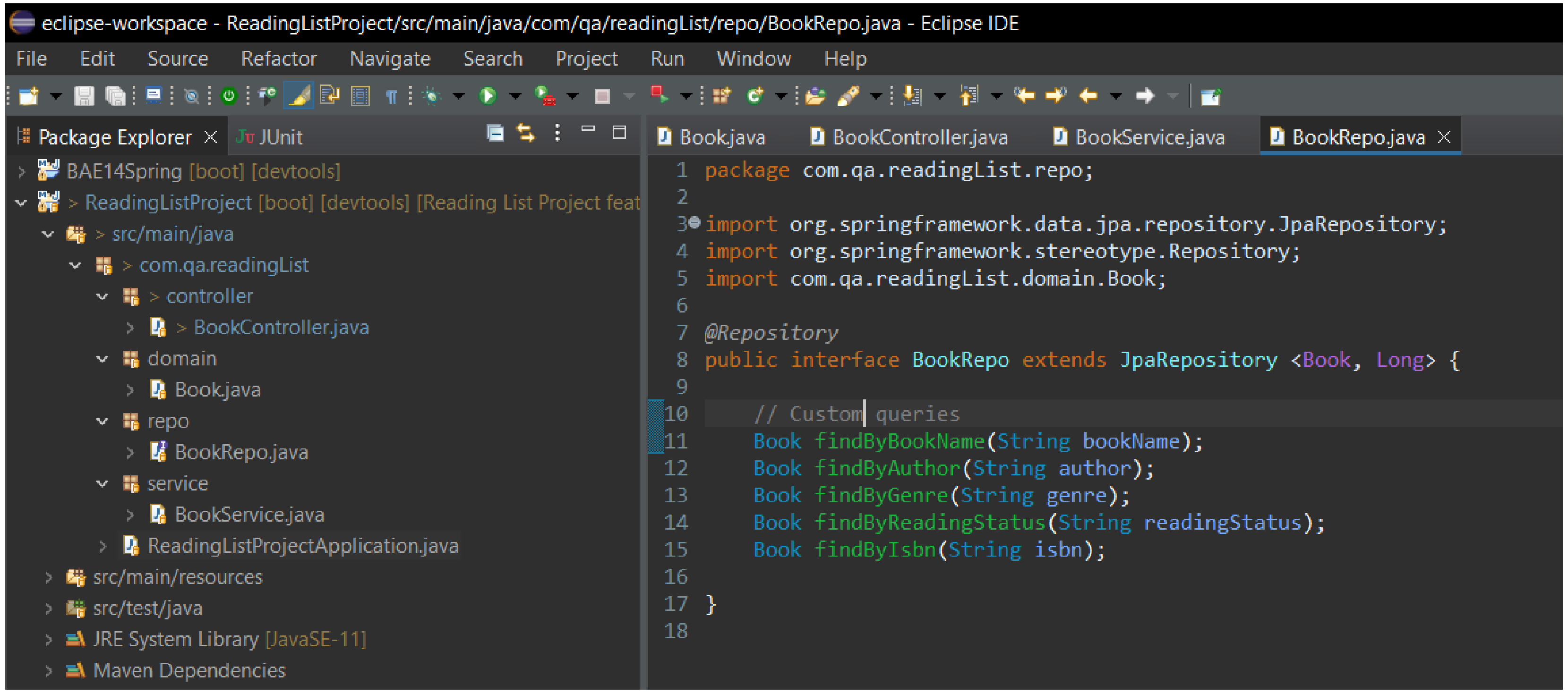
```java
    // Get ALL
    public List<Book> getAll(){
        return repo.findAll();
    }

    // Get by ID
    public Book getById(long id) {
        return repo.findById(id).get(); // .get will either get the User if exists or throw NoSuchElementException
    }

    // Get by Book Name
    public Book getByBookName(String bookName) {
        return repo.findByBookName(bookName);
    }

    // Get by Author
    public Book getByAuthor(String author) {
        return repo.findByAuthor(author);
    }

    // Get by Genre
    public Book getByGenre(String genre) {
        return repo.findByGenre(genre);
    }
```


eclipse-workspace - ReadingListProject/src/main/java/com/qa/readingList/service/BookService.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
BAE14Spring [boot] [devtools]
src/main/java
com.qa.baespring
controller
UserController.java
domain
exceptions
repo
UserRepo.java
service
UserService.java
Bae14SpringApplication.java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
src
main
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
ReadingListProject [boot] [devtools] [Reading List Project feat
Book.java
BookRepo.java
UserController.java
UserService.java
BookController.java
BookService.java

```java
    // Get by Reading Status
    public Book getByReadingStatus(String readingStatus) {
        return repo.findByReadingStatus(readingStatus);
    }

    // Get by ISBN
    public Book getByIsbn(String isbn) {
        return repo.findByIsbn(isbn);
    }

    // Create a Book (Post)
    public Book create(Book book) {
        return repo.saveAndFlush(book);
    }

    // Update a Book (Put)
    public Book update(long id, Book book) {
        Book existing = repo.findById(id).get();
        existing.setBookName(book.getBookName());
        existing.setAuthor(book.getAuthor());
        existing.setGenre(book.getGenre());
        existing.setReadingStatus(book.getReadingStatus());
        existing.setIsbn(book.getIsbn());
        return repo.saveAndFlush(existing);
    }

    // Delete a Book
    public boolean delete(long id) {
        repo.deleteById(id);
        return !repo.existsById(id);
    }
```

Rest API- Repo

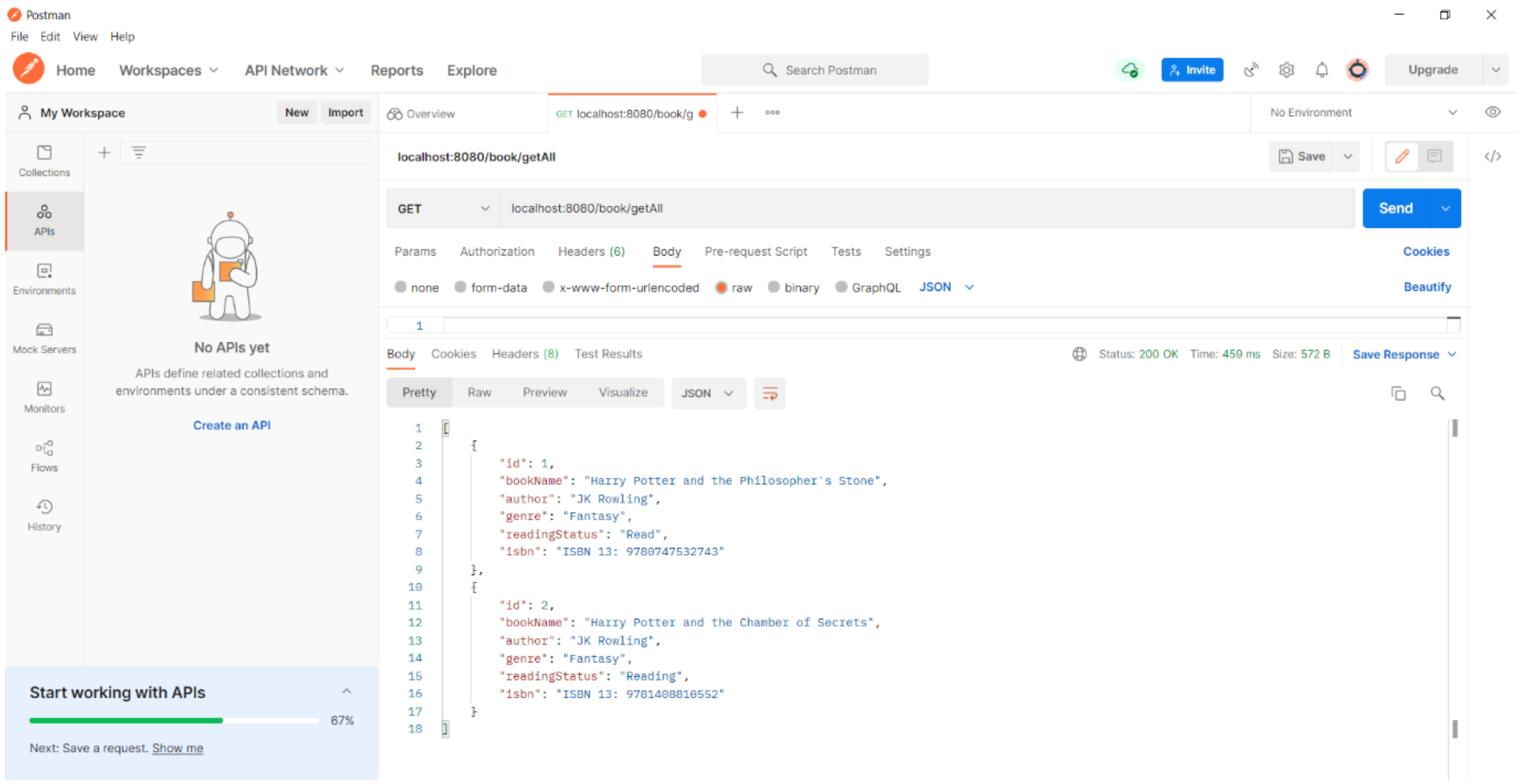
- Custom Queries


eclipse-workspace - ReadingListProject/src/main/java/com/qa/readingList/repo/BookRepo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
BAE14Spring [boot] [devtools]
ReadingListProject [boot] [devtools] [Reading List Project feat
src/main/java
com.qa.readingList
controller
BookController.java
domain
Book.java
repo
BookRepo.java
service
BookService.java
ReadingListProjectApplication.java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
Book.java
BookController.java
BookService.java
BookRepo.java

```java
package com.qa.readingList.repo;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.qa.readingList.domain.Book;

@Repository
public interface BookRepo extends JpaRepository <Book, Long> {

    // Custom queries
    Book findByBookName(String bookName);
    Book findByAuthor(String author);
    Book findByGenre(String genre);
    Book findByReadingStatus(String readingStatus);
    Book findByIsbn(String isbn);

}
```

Postman

- Checked all queries with Postman to confirm they work



localhost:8080/book/getAll

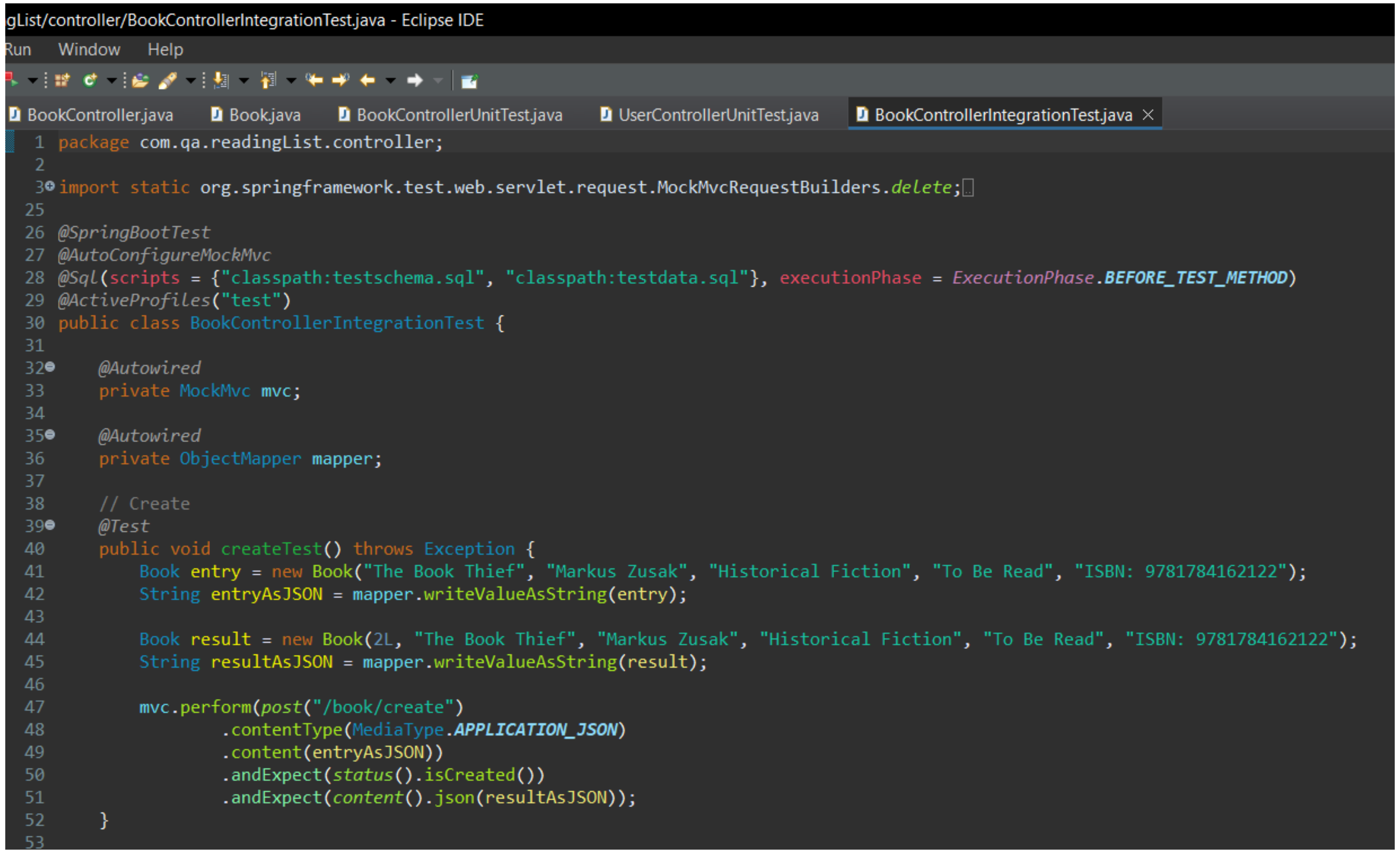
GET localhost:8080/book/getAll

Status: 200 OK Time: 459 ms Size: 572 B

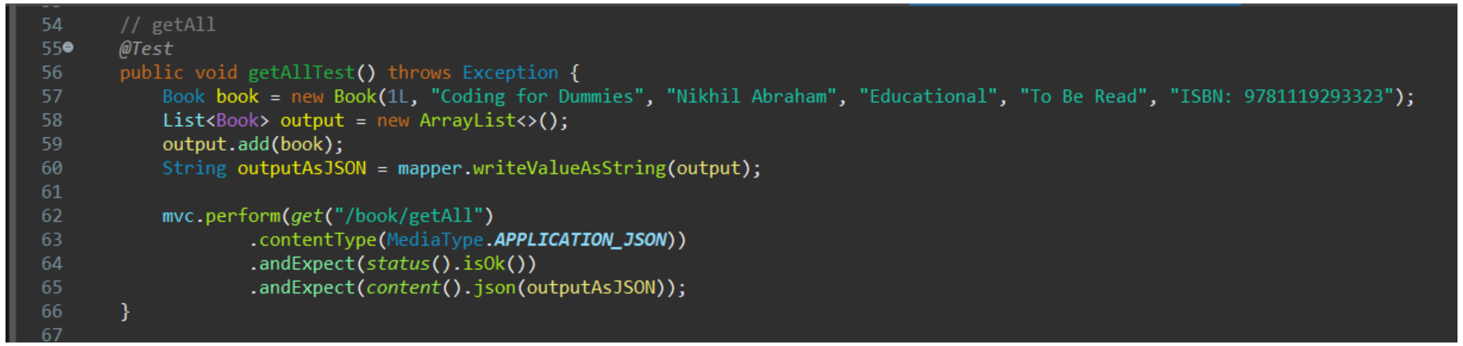
```
[
    {
        "id": 1,
        "bookName": "Harry Potter and the Philosopher's Stone",
        "author": "JK Rowling",
        "genre": "Fantasy",
        "readingStatus": "Read",
        "isbn": "ISBN 13: 9780747532743"
    },
    {
        "id": 2,
        "bookName": "Harry Potter and the Chamber of Secrets",
        "author": "JK Rowling",
        "genre": "Fantasy",
        "readingStatus": "Reading",
        "isbn": "ISBN 13: 9781408810552"
    }
]
```

Integration Test

- Methods
- Create
- getAll
- getByID
- Update
- Delete


gList/controller/BookControllerIntegrationTest.java - Eclipse IDE
Run Window Help
BookController.java
Book.java
BookControllerUnitTest.java
UserControllerUnitTest.java
BookControllerIntegrationTest.java

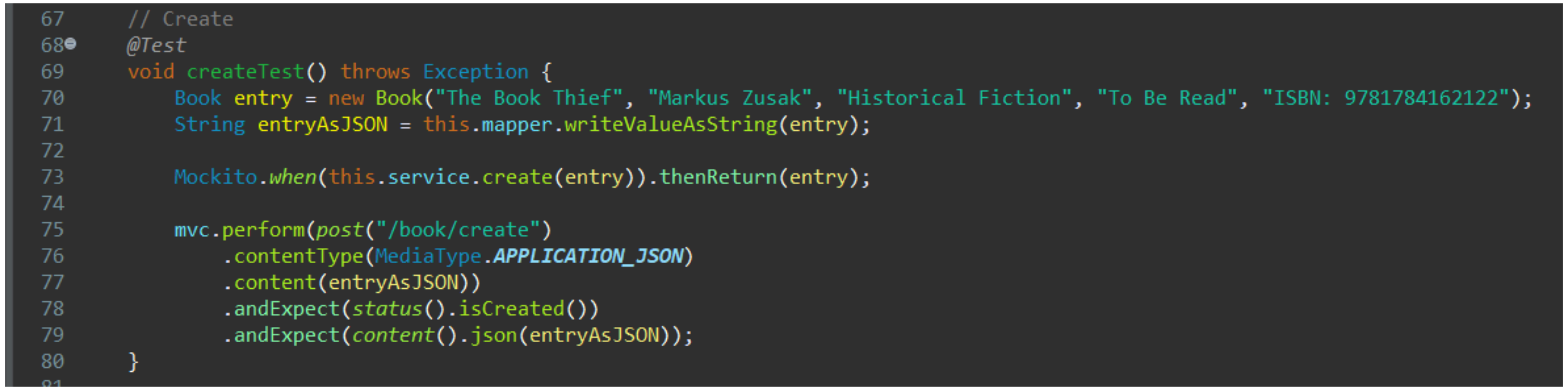
```java
package com.qa.readingList.controller;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.delete;

@SpringBootTest
@AutoConfigureMockMvc
@Sql(scripts = {"classpath:testschema.sql", "classpath:testdata.sql"}, executionPhase = ExecutionPhase.BEFORE_TEST_METHOD)
@ActiveProfiles("test")
public class BookControllerIntegrationTest {

    @Autowired
    private MockMvc mvc;

    @Autowired
    private ObjectMapper mapper;

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

        Book result = new Book(2L, "The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
        String resultAsJSON = mapper.writeValueAsString(result);

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

```



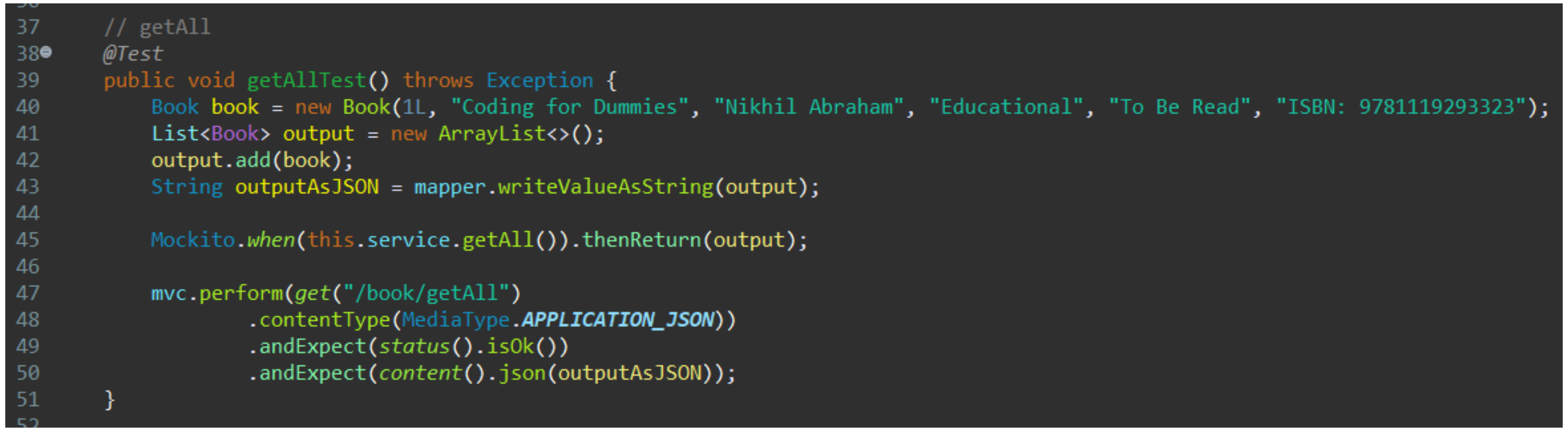
```java
    // getAll
    @Test
    public void getAllTest() throws Exception {
        Book book = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
        List<Book> output = new ArrayList<>();
        output.add(book);
        String outputAsJSON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

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

```

Unit Test - Controller

- Same methods used as integration testing.
- An addition of one line of code is needed in the methods.
- Annotations are different to integration testing.



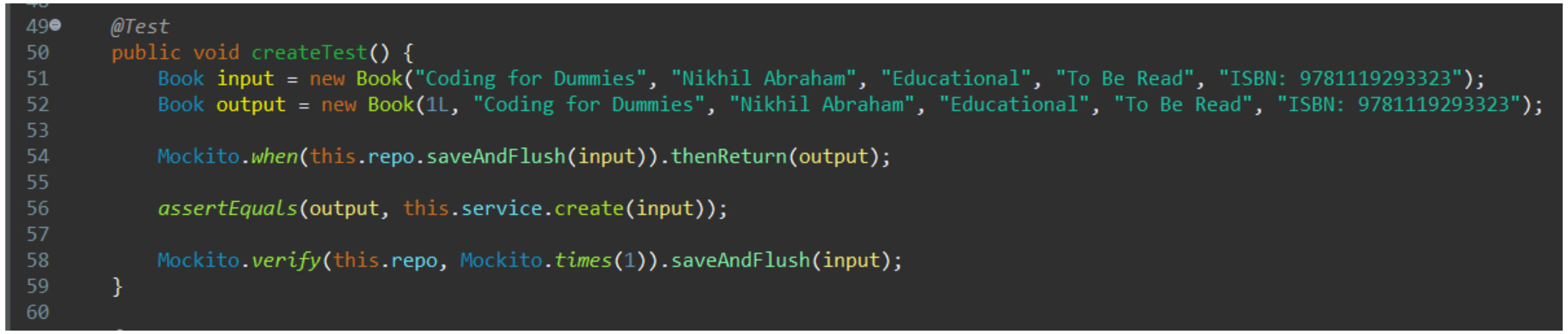
```java
// Create
@Test
void createTest() throws Exception {
    Book entry = new Book("The Book Thief", "Markus Zusak", "Historical Fiction", "To Be Read", "ISBN: 9781784162122");
    String entryAsJSON = this.mapper.writeValueAsString(entry);

    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));
}
```



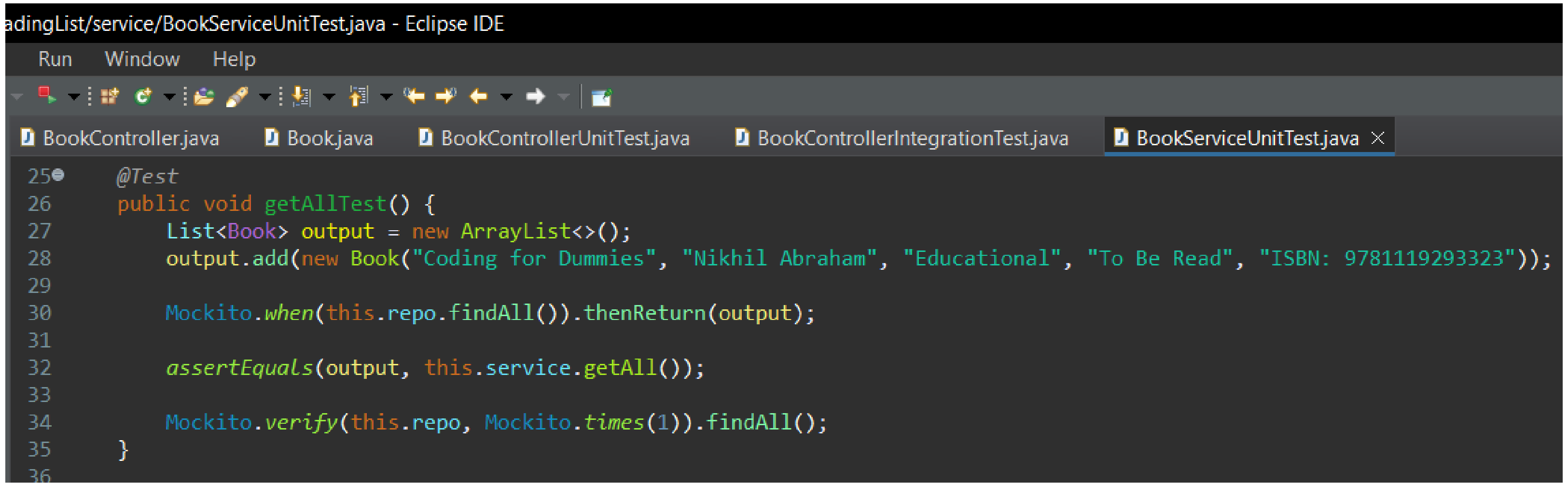
```java
// getAll
@Test
public void getAllTest() throws Exception {
    Book book = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
    List<Book> output = new ArrayList<>();
    output.add(book);
    String outputAsJSON = mapper.writeValueAsString(output);

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

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

Unit Test - Service

- For all the same methods as unit test for controller



```java
    @Test
    public void createTest() {
        Book input = new Book("Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");
        Book output = new Book(1L, "Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323");

        Mockito.when(this.repo.saveAndFlush(input)).thenReturn(output);

        assertEquals(output, this.service.create(input));

        Mockito.verify(this.repo, Mockito.times(1)).saveAndFlush(input);
    }

```

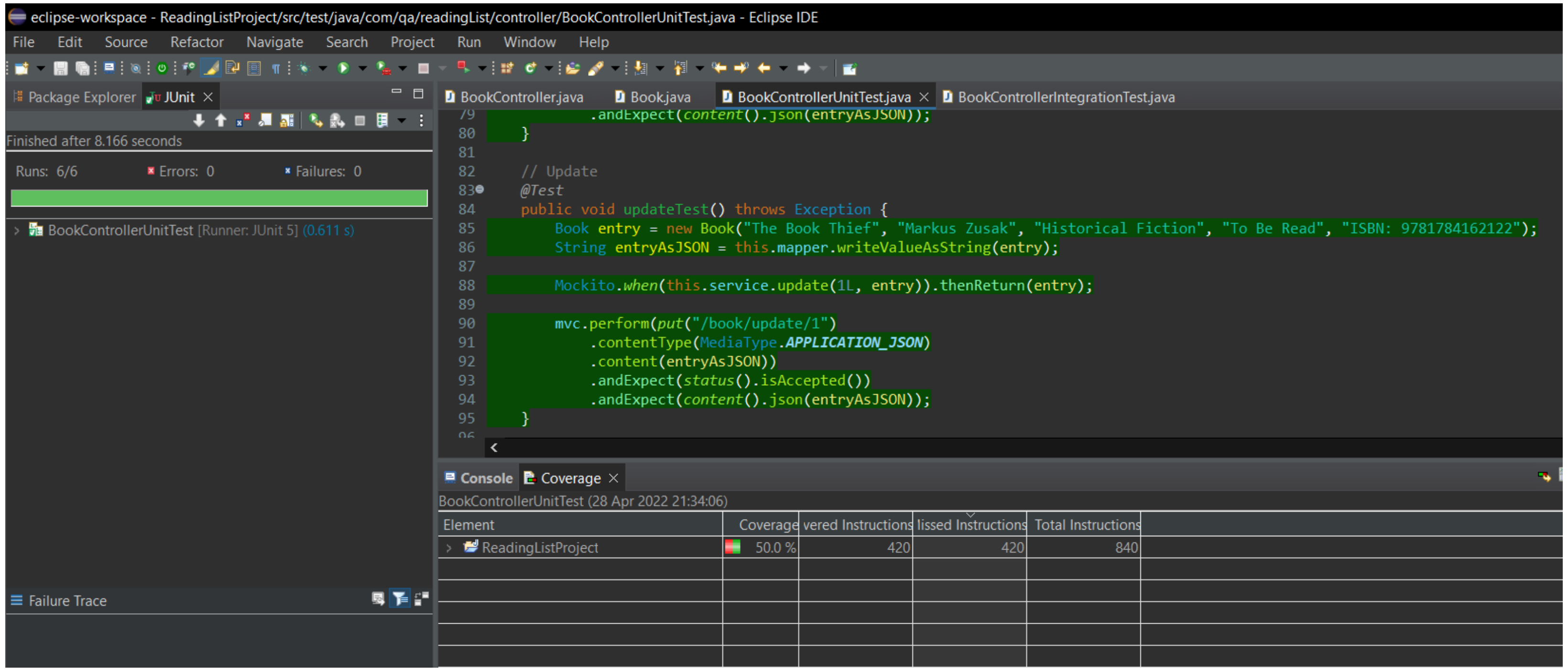


adingList/service/BookServiceUnitTest.java - Eclipse IDE

```java
    @Test
    public void getAllTest() {
        List<Book> output = new ArrayList<>();
        output.add(new Book("Coding for Dummies", "Nikhil Abraham", "Educational", "To Be Read", "ISBN: 9781119293323"));

        Mockito.when(this.repo.findAll()).thenReturn(output);

        assertEquals(output, this.service.getAll());

        Mockito.verify(this.repo, Mockito.times(1)).findAll();
    }

```

Test Coverage

- All tests passed.
- All tests coverage approx. 40– 50%
- Not met the industry standard of 80%.



eclipse-workspace - ReadingListProject/src/test/java/com/qa/readingList/controller/BookControllerUnitTest.java - Eclipse IDE

Finished after 8.166 seconds

Runs: 6/6 Errors: 0 Failures: 0

BookControllerUnitTest [Runner: JUnit 5] (0.611 s)

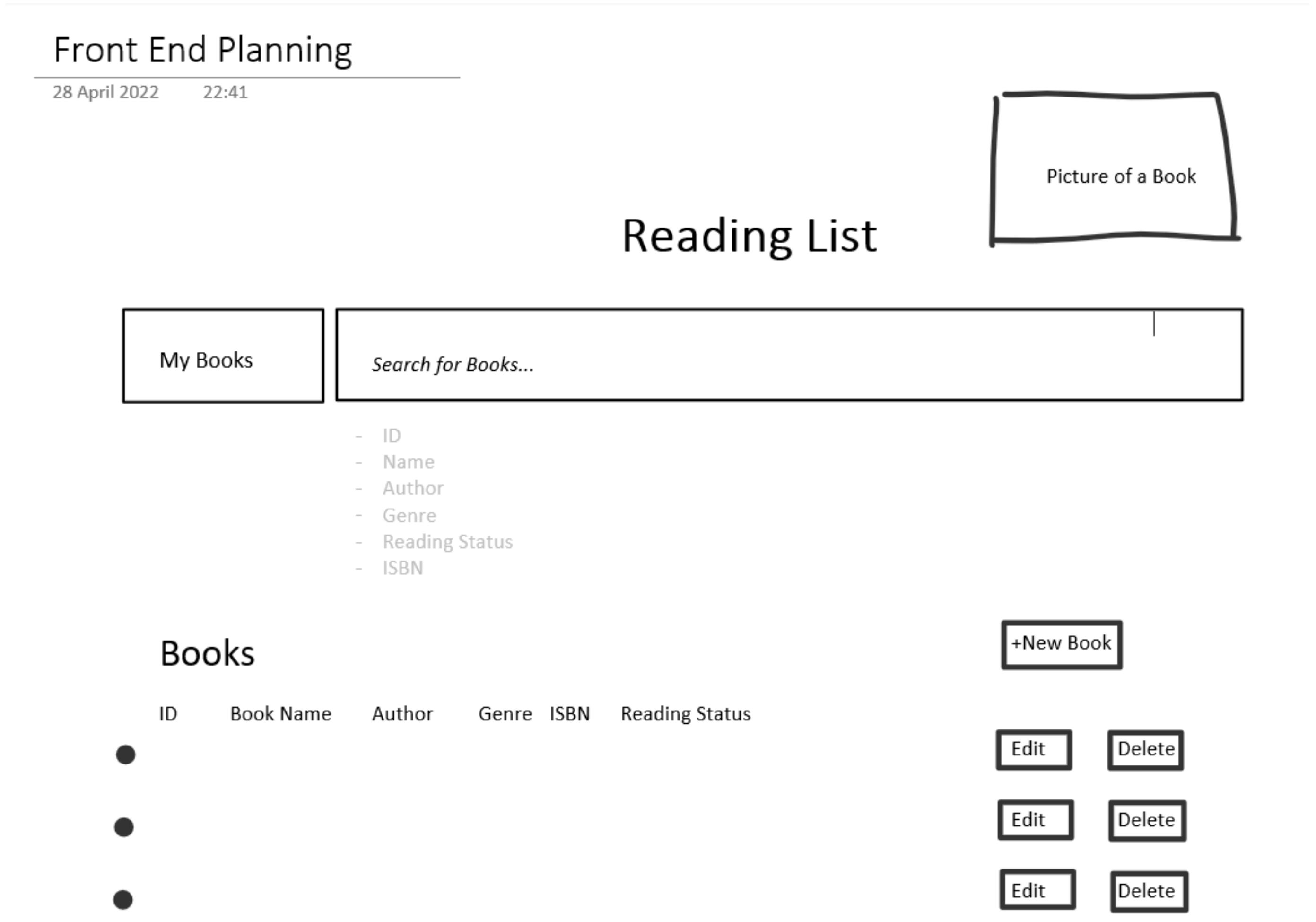
```java
            .andExpect(content().json(entryAsJSON));
    }

    // 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));
    }
```

BookControllerUnitTest (28 Apr 2022 21:34:06)

Element	Coverage	vered Instructions	lissed Instructions	Total Instructions
ReadingListProject	50.0 %	420	420	840

Front End - Planning

- The design of desired front-end layout

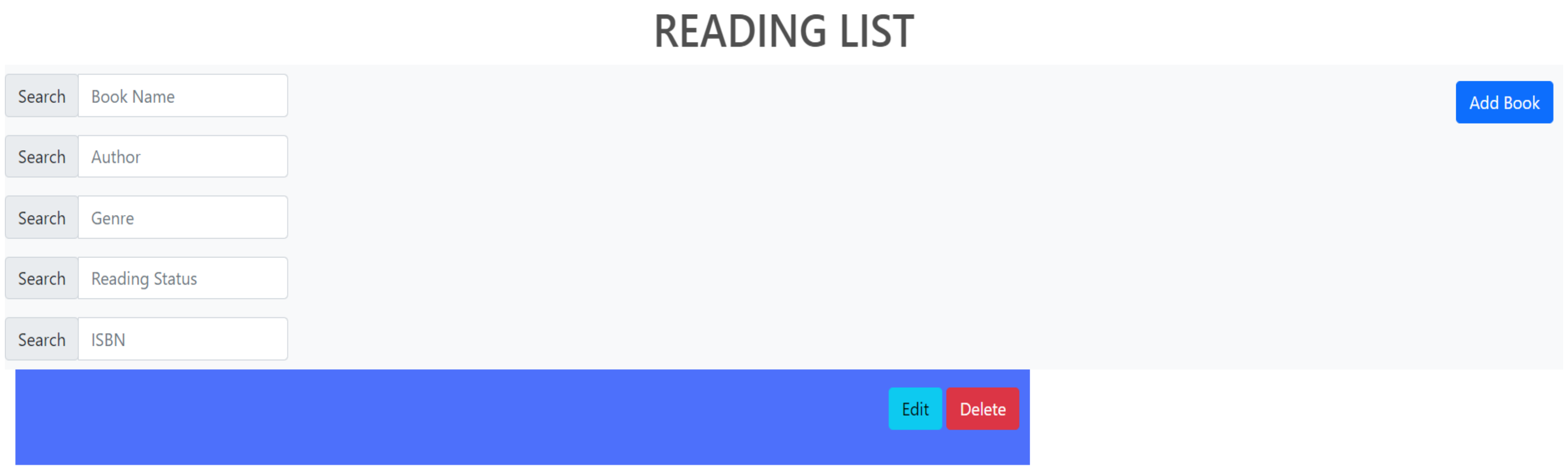


Front End Planning

28 April 2022 22:41

Front End

- The front-end is much harder than it seems
- Used Bootstrap in html file
- CSS
- JavaScript



Thank you for listening!