SE 318

SOFTWARE VERIFICATION AND VALIDATION

SPRING 2021

BookStore Management System

Eyup Batuhan Sevinc

Berk Muslu

Tuğgun Asrak

UNIT TEST Document

Version *3.0*

24/06/2021

VERSION HISTORY

VERSION 1.0 -14.05.2021
-UI added. -Stock data added. -Shopping cart added. -Current price,tax,shipping expenses added. -Current profit and sales ratio added
VERSION 2.0 -05.06.2021
-Txt based database added. -UI is improved. -Admin panel is added. -Profit panel is added for admin.
VERSION 3.0 -24.06.2021

-Search features added.

-5 clients can operate the system simultaneously.

-20 seconds time limit added for search operation.

-The system supports Windows8.1 and higher, Debian based Linux, macOS operating systems.

1 Introduction

1.1 Purpose of The Test Case Document

A test case is a set of actions executed to verify a particular feature or functionality of the software system. A test case contains test steps, test data, precondition, postcondition developed for specific test scenarios to verify any requirement.

The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

The Test Case document is documenting the functional requirements of the Test Case Documentation test case. The target group is the project manager, project team, and testing team. A few parcels of this archive may on event be shared with the client/user and other partner whose input/approval into the testing preparation is needed.

1.2 CONSTRAINTS

Junit5 was utilized for the unit tests system and the framework was built with the Java programming language. The test cases worked easily inside the given

imperatives. In any case, there were a few issues with not being able to multi-return in Java, so additional strategies were composed.

2 UNIT TEST FRAMEWORK: *JUNIT*

JUnit 5 was utilized for the unit test system. It may be a unit test library utilized for testing Java based codes. JUnit rearranges testing and abbreviates this time by advertising an assortment of additional features. It moreover gives test-driven improvement. In expansion, it guarantees that the code is composed by testing the arrangement that's made, some time recently moving to the program development.

3 TEST CASES

z

Test Case 1	
Test Definition	
This thes for testing the function of username checking fur admin	
Input Value	
`int id = 1;`	
Expected Value	**Actual Value**
admin	**admin**
Result of Test Case	***successful***
Test Script	

```java
@Test
void getAdminUsernamePositive(){
    int id = 1;
    PreparedStatement ps;
    ResultSet rs;
    String username = "";
    String query = "SELECT * FROM admin_accounts WHERE `id` =?";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=tr
        ps = con.prepareStatement(query);
        ps.setInt( parameterIndex: 1, id);
        rs = ps.executeQuery();
        if (rs.next()) {
            username = rs.getString( columnLabel: "username");
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Get Admin Username Positive");
    assertEquals( expected: "admin",username);
}
```

Test Case 2	
Test Definition	
This test case is checking the total sale counter.	
Input Value	
Expected Value	**Actual Value**
4	**4**
Result of Test Case	***successfull***

Test Script

```
@Test
void getSaleCountPositive() {

    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT quantity,total FROM transactions";
    int sales = 0;

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();

        while (rs.next()){
            sales += rs.getInt( columnLabel: "quantity");
        }

        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Get Sales Count Positive");
    assertEquals( expected: 4,sales);

}
```

Test Case 3

Test Definition

This test case is checking the total sale counter

Input Value

<Write input>

Expected Value	Actual Value
!=4	0

Result of Test Case	*successfull*
Test Script	

```
@Test
void getSaleCountNegative() {

    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT quantity,total FROM transactions";
    int sales = 0;

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();

        while (rs.next()){
            sales = rs.getInt( columnLabel: "quantity");
        }

        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Get Sales Count Negative");
    assertNotEquals( unexpected: 4,sales);
```

Test Case 4
Test Definition
This test case is checking the total income counter
Input Value
<Write input>

Expected Value	Actual Value
108	**108**
Result of Test Case	***syccessfull***
Test Script	

```
@Test
void getTotalIncomePositive() {

    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT total FROM transactions";
    double income = 0;

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                 user: 'j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();

        while (rs.next()){
            income += rs.getInt( columnLabel: "total");
        }

        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Get Income Positive");
    assertEquals( expected: 108,income);
```

Test Case 5
Test Definition
This test case is checking the total income counter
Input Value

<Write input>	
Expected Value	**Actual Value**
!=108	**0**
Result of Test Case	***<successful OR fail>***
Test Script	

```
@Test
void getTotalIncomeNegative() {

    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT total FROM transactions";
    double income = 0;

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();

        while (rs.next()){
            income = rs.getInt( columnLabel: "total");
        }

        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Get Income Negative");
    assertNotEquals( unexpected: 108,income);
```

Test Case 6
Test Definition

This test case is checking the total profit counter	
Input Value	
<Write input>	
Expected Value	**Actual Value**
20	**20**
Result of Test Case	***Successfull***
Test Script	

```
@Test
void calculateProfitPositive(){

    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT quantity FROM transactions";
    int sales = 0;
    final int profit = 5;

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();

        while (rs.next()){
            sales += rs.getInt( columnLabel: "quantity");
        }

        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Calculate Profit Positive");
    int last_prof = sales * profit;
    assertEquals( expected: 20,last_prof);
```

Test Case 7	
Test Definition	
This test case is checking the total profit counter	
Input Value	
<Write input>	
Expected Value	**Actual Value**
!=20	**6**
Result of Test Case	***Successfull***
Test Script	

```java
@Test
void calculateProfitNegative(){

    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT quantity FROM transactions";
    int sales = 0;
    final int profit = 6;

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();

        while (rs.next()){
            sales += rs.getInt( columnLabel: "quantity");
        }

        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Calculate Profit Positive");
    int last_prof = sales * profit;
    assertNotEquals( unexpected: 20,last_prof);
```

Test Case 8	
Test Definition	
This test case for checking the get admin username function	
Input Value	
<Write input>	
Expected Value	**Actual Value**
!= “Admin”	**“ ”**

Result of Test Case	***success***
Test Script	

```
@Test
void getAdminUsernameNegative(){

    int id = 2;
    PreparedStatement ps;
    ResultSet rs;
    String username = "";

    String query = "SELECT * FROM admin_accounts WHERE `id` =?";

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");

        ps = con.prepareStatement(query);
        ps.setInt( parameterIndex: 1, id);
        rs = ps.executeQuery();

        if (rs.next()) {

            username = rs.getString( columnLabel: "username");
        }

        con.close();

    } catch (Exception e) {
        e.printStackTrace();
```

Test Case 9	
Test Definition	
This test case for checking the get admin password function	
Input Value	
<Write input>	
Expected Value	**Actual Value**

“admin”	“admin”
Result of Test Case	***Successfull***
Test Script	

```
@Test
void getAdminPasswordPositive(){

    int id = 1;
    PreparedStatement ps;
    ResultSet rs;
    String password = "";

    String query = "SELECT * FROM admin_accounts WHERE `id` =?";

    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");

        ps = con.prepareStatement(query);
        ps.setInt( parameterIndex: 1, id);
        rs = ps.executeQuery();

        if (rs.next()) {

            password = rs.getString( columnLabel: "password");
        }

        con.close();

    } catch (Exception e) {
        e.printStackTrace();
    }

    System.out.println("Get Admin Password Positive");
```

Test Case 10
Test Definition
This test case for checking the get admin username function

Input Value	
<Write input>	
Expected Value	**Actual Value**
!=”admin”	**“ “**
Result of Test Case	***Successful***
Test Script	

```
@Test
void getAdminPasswordNegative(){
    int id = 2;
    PreparedStatement ps;
    ResultSet rs;
    String password = "";
    String query = "SELECT * FROM admin_accounts WHERE `id` =?";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        ps.setInt( parameterIndex: 1, id);
        rs = ps.executeQuery();
        if (rs.next()) {
            password = rs.getString( columnLabel: "password");
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Get Admin Password Negative");
    assertNotEquals( unexpected: "test",password);


}
```

Test Case 11
Test Definition

This test case for checking the get total users function	
Input Value	
<Write input>	
Expected Value	**Actual Value**
2	**2**
Result of Test Case	***Successfull***
Test Script	

```
@Test
void getTotalUsers(){
    int user_count = 0;
    String query = "SELECT COUNT(*) FROM accounts";
    ResultSet rs;


    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        Statement statement = con.createStatement();
        rs = statement.executeQuery(query);
        if(rs.next()){
            user_count = rs.getInt( columnLabel: "COUNT(*)");
        }


    }catch (Exception e){
        e.printStackTrace();
    }

    assertEquals( expected: 2,user_count);


}
```

Test Case 12	
Test Definition	
This test case for checking the get total admins function	
Input Value	
<Write input>	
Expected Value	**Actual Value**
1	**1**
Result of Test Case	***Successfull***
Test Script	

```
@Test
void getTotalAdmins(){
    int user_count = 0;
    String query = "SELECT COUNT(*) FROM admin_accounts";
    ResultSet rs;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        Statement statement = con.createStatement();
        rs = statement.executeQuery(query);
        if(rs.next()){
            user_count = rs.getInt( columnLabel: "COUNT(*)");
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    assertEquals( expected: 1,user_count);
}
```

Test Case 13	
Test Definition	
This test case for checking the get books function	
Input Value	
<Write input>	
Expected Value	**Actual Value**
20	**20**
Result of Test Case	***Successfull***
Test Script	

```
@Test
void getTotalBooks(){
    int book_count = 0;
    String query = "SELECT COUNT(*) FROM books";
    ResultSet rs;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        Statement statement = con.createStatement();
        rs = statement.executeQuery(query);
        if(rs.next()){
            book_count = rs.getInt( columnLabel: "COUNT(*)");
        }


    }catch (Exception e){
        e.printStackTrace();
    }

    assertEquals( expected: 20,book_count);
```

Test Case 14	
Test Definition	
This test case for checking stock status	
Input Value	
<Write input>	
Expected Value	**Actual Value**
56	**56**
Result of Test Case	***Successfull***
Test Script	

```java
@Test
void stockCheckPositive(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT stock FROM books WHERE id = 1";
    int stock = 0;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            stock = rs.getInt( columnLabel: "stock");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Stock Check Positive");
    assertEquals( expected: 56,stock);
}
```

Test Case 15	
Test Definition	
This test case for checking stock status	
Input Value	
<Write input>	
Expected Value	**Actual Value**
!=52	**0**
Result of Test Case	***Successfull***
Test Script	

```java
@Test
void stockCheckNegative(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT stock FROM books WHERE id = 1";
    int stock = 0;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            stock = rs.getInt( columnLabel: "stock");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Stock Check Negative");
    assertNotEquals( unexpected: 52,stock);
}
```

Test Case 16	
Test Definition	
This test case is for book name check function	
Input Value	
<Write input>	
Expected Value	**Actual Value**
“Harry Potter”	**“Harry Potter”**
Result of Test Case	***Successfull***

Test Script

```
@Test
void bookNameCheckPositive(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT name FROM books WHERE id = 1";
    String name = "";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            name = rs.getString( columnLabel: "name");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Bookname Positive");
    assertEquals( expected: "Harry Potter",name);
}
```

Test Case 17

Test Definition

This test case is for book name check function

Input Value

<Write input>

Expected Value	Actual Value
!= “Harry Potter”	**“”**

Result of Test Case ***Successfull***

Test Script


```
@Test
void bookNameCheckNegative(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT name FROM books WHERE id = 2";
    String name = "";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UE82tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            name = rs.getString( columnLabel: "name");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Booknane Negative");
    assertNotEquals( unexpected: "Harry Potter",name);
}
```

Test Case 18	
Test Definition	
This test case is for author name check function	
Input Value	
“SELECT author FROM books WHERE id=1”	
Expected Value	Actual Value
“J.K. Rowling”	“J.K. Rowling”
Result of Test Case	*Successfull*

Test Script

```
@Test
void bookAuthorCheckPositive(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT author FROM books WHERE id = 1";
    String author = "";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();

        if(rs.next()){
            author = rs.getString( columnLabel: "author");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Book Author Positive");
    assertEquals( expected: "J.K. Rowling",author);
```

Test Case 19

Test Definition

This test case is for author name check function

Input Value

<Write input>

Expected Value	Actual Value
!="J.K. Rowling"	**“”**

Result of Test Case ***Successfull***

Test Script

```
@Test
void bookAuthorCheckNegative(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT author FROM books WHERE id = 2";
    String author = "";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            author = rs.getString( columnLabel: "author");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Book Author Negative");
    assertNotEquals( unexpected: "J.K. Rowling",author);
}
```

Test Case 20	
Test Definition	
This test case is for genre check function	
Input Value	
<Write input>	
Expected Value	**Actual Value**
“Fantasy”	**“Fantasy”**

Result of Test Case	*Success*

Test Script

```
@Test
void bookGenreCheckPositive(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT category FROM books WHERE id = 1";
    String genre = "";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            genre = rs.getString( columnLabel: "category");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Book Genre Positive");
    assertEquals( expected: "Fantasy",genre);
}
```

Test Case 21

Test Definition

This test case is for genre check function

Input Value

<Write input>

Expected Value	Actual Value
!=”Fantasy”	**“”**

Result of Test Case	*Success*
Test Script	

```
@Test
void bookGenreCheckNegative(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT category FROM books WHERE id = 3";
    String genre = "";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            genre = rs.getString( columnLabel: "category");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Book Genre Negative");
    assertNotEquals( unexpected: "Fantasy",genre);
}
```

Test Case 22	
Test Definition	
This case is for checking price check function	
Input Value	
<Write input>	
Expected Value	**Actual Value**

25	**25**
Result of Test Case	***Success***
Test Script	

```
void bookPriceCheckPositive(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT price FROM books WHERE id = 1";
    double price = 0;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            price = rs.getDouble( columnLabel: "price");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Book price Positive");
    assertEquals( expected: 25,price);
}
```

Test Case 23	
Test Definition	
This case is for checking price check function	
Input Value	
<Write input>	
Expected Value	**Actual Value**

!=25	0
Result of Test Case	***Success***
Test Script	

```
@Test
void bookPriceCheckNegative(){
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT price FROM books WHERE id = 2";
    double price = 0;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            price = rs.getDouble( columnLabel: "price");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("Check Book price Positive");
    assertNotEquals( unexpected: 25,price);
}
```

Test Case 24
Test Definition
This case is for checking stock decreasing function
Input Value
<Write input>

Expected Value	Actual Value
4	4
Result of Test Case	***Success***
Test Script	

```
@Test
void stockDecreaseTestPositive() {
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT stock FROM books WHERE id = 1";
    int stock = 0;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            stock = rs.getInt( columnLabel: "stock");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    int quantity = 4;
    int last_stock = stock-quantity;
    System.out.println("Stock Decrease Test Positive");
    assertNotEquals(stock,last_stock);
```

Test Case 25
Test Definition
This case is for checking stock decreasing function
Input Value
<Write input>

Expected Value	Actual Value
<0	-2
Result of Test Case	***Success***
Test Script	

```
@Test
void stockDecreaseTestNegative() {
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT stock FROM books WHERE id = 1";
    int stock = 0;
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        rs = ps.executeQuery();
        if(rs.next()){
            stock = rs.getInt( columnLabel: "stock");
        }
        con.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    int last_stock = stock;
    int quantity = -2;
    if(quantity > 0) {
        last_stock  = stock - quantity;
    }
    System.out.println("Stock Decrease Test Negative");
    assertEquals(stock,last_stock);
```

Test Case 26
Test Definition
This case is for checking search by name function

Input Value	
“Harry Potter”	
Expected Value	**Actual Value**
True	**True**
Result of Test Case	***Success***
Test Script	

```
@Test
void searchByNamePositive() {
    boolean bookFound = false;
    String book_name = "Harry Potter";
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT * FROM books WHERE `name` =?";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        ps.setString( parameterIndex: 1, book_name);
        rs = ps.executeQuery();
        if (rs.next()) {
            bookFound = true;
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Search by Name Test Positive");
    assertEquals( expected: true,bookFound);

}
```

Test Case 27
Test Definition

This case is for checking search by name function	
Input Value	
“Harry Poter”	
Expected Value	**Actual Value**
False	**False**
Result of Test Case	***Success***
Test Script	

```
@Test
void searchByNameNegative() {
    boolean bookFound = false;
    String book_name = "Harry Poter";
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT * FROM books WHERE `name` =?";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        ps.setString( parameterIndex: 1, book_name);
        rs = ps.executeQuery();
        if (rs.next()) {
            bookFound = true;
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Search by Name Test Negative");
    assertNotEquals( unexpected: true,bookFound);

}
```

Test Case 28	
Test Definition	
This case is for checking search by author function	
Input Value	
“J.K. Rowling”	
Expected Value	**Actual Value**
True	**True**
Result of Test Case	***Success***
Test Script	

```
@Test
void searchByAuthorPositive() {

    boolean bookFound = false;
    String author = "J.K. Rowling";
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT * FROM books WHERE `author` =?";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        ps.setString( parameterIndex: 1, author);
        rs = ps.executeQuery();
        if (rs.next()) {
            bookFound = true;
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Search by Author Test Positive");
    assertEquals( expected: true,bookFound);

}
```

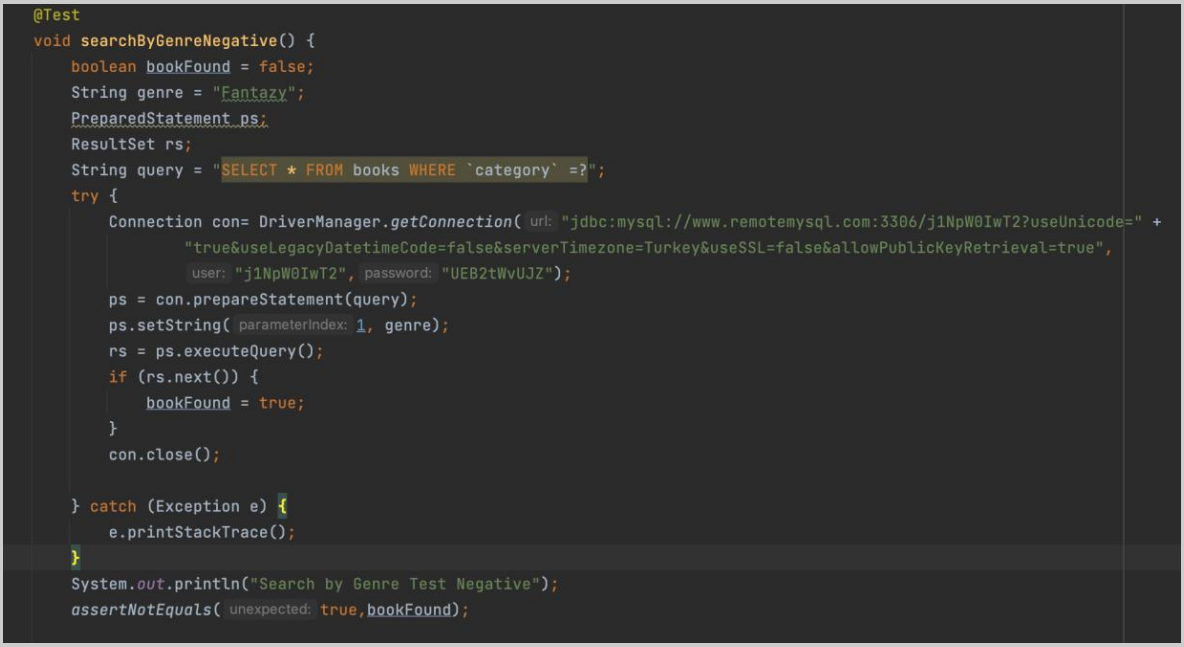
Test Case 29	
Test Definition	
This case is for checking search by author function	
Input Value	
“J.K. Bowling”	
Expected Value	**Actual Value**
False	**False**
Result of Test Case	***Success***

Test Script

```
@Test
void searchByAuthorNegative() {
    boolean bookFound = false;
    String author = "J.K. Bowling";
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT * FROM books WHERE `author` =?";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        ps.setString( parameterIndex: 1, author);
        rs = ps.executeQuery();
        if (rs.next()) {
            bookFound = true;
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Search by Author Test Negative");
    assertNotEquals( unexpected: true,bookFound);
}
```

Test Case 30	
Test Definition	
This case is for checking search by genre function	
Input Value	
“Fantazy”	
Expected Value	Actual Value

False	**False**
Result of Test Case	***Success***
Test Script	



```
@Test
void searchByGenreNegative() {
    boolean bookFound = false;
    String genre = "Fantazy";
    PreparedStatement ps;
    ResultSet rs;
    String query = "SELECT * FROM books WHERE `category` =?";
    try {
        Connection con= DriverManager.getConnection( url: "jdbc:mysql://www.remotemysql.com:3306/j1NpW0IwT2?useUnicode=" +
                "true&useLegacyDatetimeCode=false&serverTimezone=Turkey&useSSL=false&allowPublicKeyRetrieval=true",
                user: "j1NpW0IwT2", password: "UEB2tWvUJZ");
        ps = con.prepareStatement(query);
        ps.setString( parameterIndex: 1, genre);
        rs = ps.executeQuery();
        if (rs.next()) {
            bookFound = true;
        }
        con.close();

    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Search by Genre Test Negative");
    assertNotEquals( unexpected: true,bookFound);
```

4.Conclusion

In this project, we developed a BookStore Management System.The system written with Java Programming Language.In addition we used JUNIT framework for Unit testing and requirements list documentation. The version history, test cases and the results of the specified test cases included in this test document. Test cases check if the system is working as dedicated and functional requirements are working properly.