Book Automation Software

(TEST CASES & TEST PLAN DOCUMENT)

S.VISHNU PRANEETH(14CS30031)
KINSUK DAS(14CS10025)

Introduction

This document is an overview of the testing strategies and plans of Book Automation Software(BAS).It outlines the nature and purpose of all testing procedures.Testing criteria under the white box, black box, and system-testing paradigm will be utilized.The test plan is aimed at verifying the functionality and correct working of every aspect and part of the Book Automation Software(BAS). Test document specifications as per IEEE Standard 829-1983 for Software Test Documentation are implemented in this document.

Objective

The objective of testing the software is to exhaustively test the entire program for finding all the bugs.The user interface is designed user-friendly and provides easy access to all the functions.The definition of testing is the process of analyzing a software item, to detect the differences between existing and required conditions i.e. defects/errors/bugs and to evaluate the features of the software item.

Testing Process flow

This matches the present software with the required standards for the proper functioning of a software.

The flow of the testing strategy is described as follows:

- Figure out all the requirements to be tested in the software item.
- Find the type of test that is used to test a specific property of the software item.
- Obtain the test results for each such testing procedure.

- Tabulate the tested data and the outcome or the results of the testing procedure done.
- A particular field test has to be successful before the software item is tested for system testing.
- If a particular test is unsuccessful a bug/defect report is to be produced showing the defect and the causes of the problem.

Scope

There are several stages at which testing is done when the product is constructed.Testing is a very dependent activity.Test plans must be developed at each level of product testing and development.

Purpose

The sole purpose of testing is to validate and check a software to check if it fits the requirements specified and to identify the errors present in the software item. The faults found in the software are then fixed later.

Testing strategy

This part emphasizes on the test types done for unit testing,integration testing system testing ,interface testing ,security testing ,etc.

● Unit Testing

This testing is done to identify the errors and bugs involved at the code level(program level) . The test cases are defined in order to test the correctness of the program.

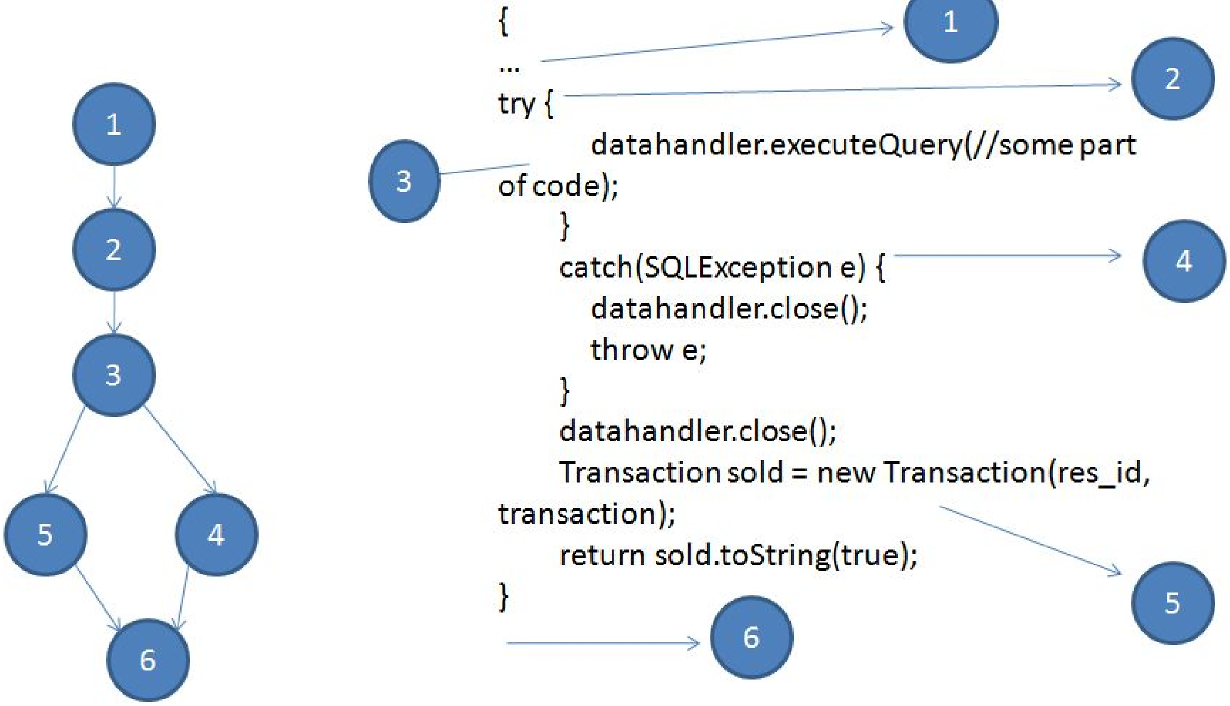
White Box Testing

This form of testing is done to pay main attention on the code part,so as to identify any errors or bugs if present to improve the functionality of the software . Test cases are designed accordingly so as to identify all such defects.

In white Box testing Control-Flow-Diagrams of Condition coverage are drawn to determine the flow through paths created in such diagram.

White Box Testing for addTransaction()

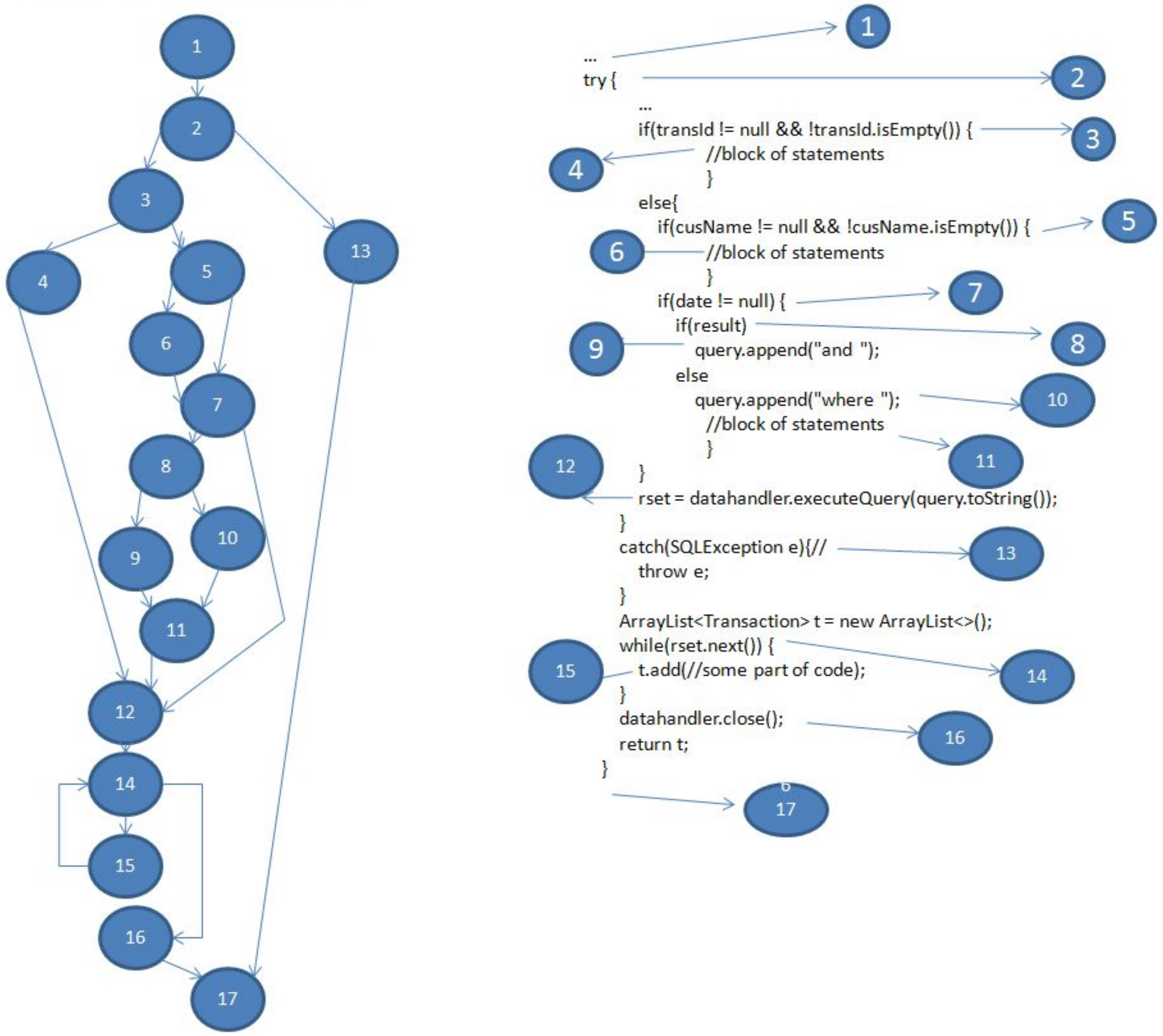
CFD for addTransaction()



```
{
...
try {
      datahandler.executeQuery(//some part
of code);
   }
   catch(SQLException e) {
      datahandler.close();
      throw e;
   }
   datahandler.close();
   Transaction sold = new Transaction(res_id,
transaction);
   return sold.toString(true);
}
```

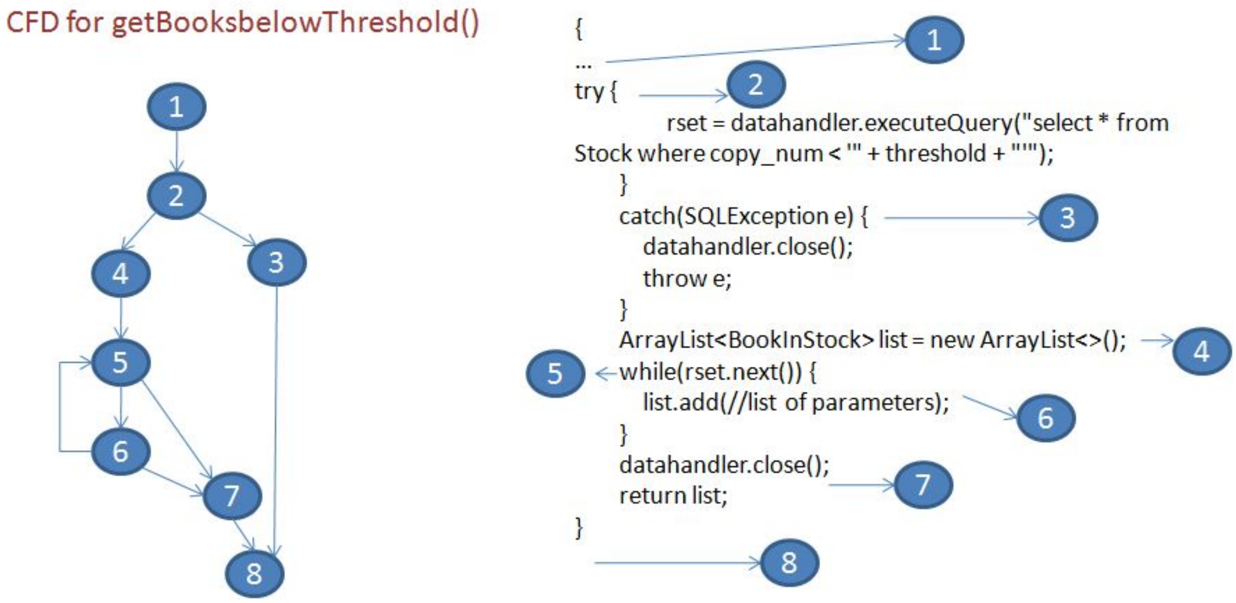
White Box Testing for getTransaction()

CFD for getTransaction()



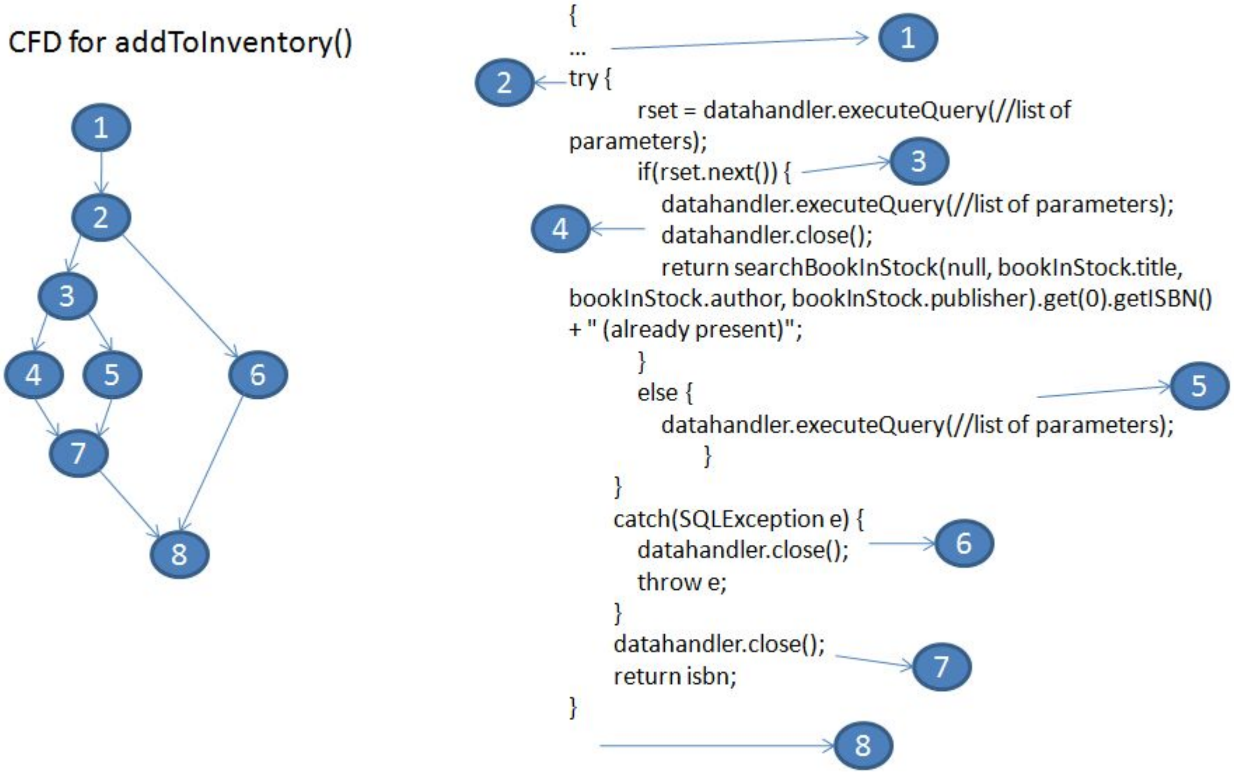
```
...
try {
    ...
    if(transId != null && !transId.isEmpty()) {
          //block of statements
          }
    else{
      if(cusName != null && !cusName.isEmpty()) {
          //block of statements
          }
      if(date != null) {
        if(result)
          query.append("and ");
        else
          query.append("where ");
          //block of statements
          }
    }
    rset = datahandler.executeQuery(query.toString());
  }
  catch(SQLException e){//
    throw e;
  }
  ArrayList<Transaction> t = new ArrayList<>();
  while(rset.next()) {
    t.add(//some part of code);
  }
  datahandler.close();
  return t;
}
```

White Box Testing for getBooksBelowThreshold()


CFD for getBooksbelowThreshold()

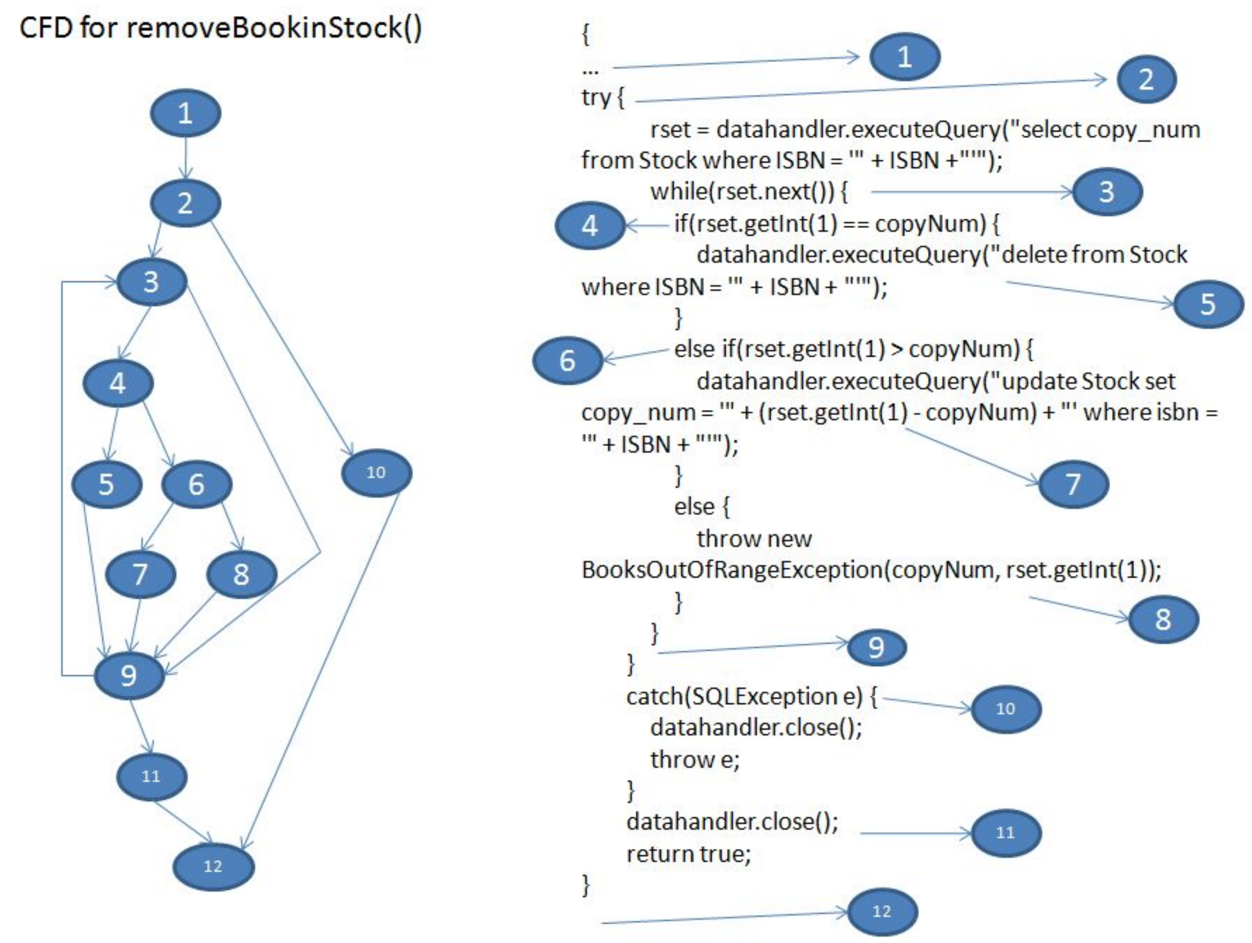
```
{
...
try {
            rset = datahandler.executeQuery("select * from
Stock where copy_num < '" + threshold + "'");
      }
      catch(SQLException e) {
         datahandler.close();
         throw e;
      }
      ArrayList<BookInStock> list = new ArrayList<>();
      while(rset.next()) {
         list.add(//list of parameters);
      }
      datahandler.close();
      return list;
}
```

White Box Testing for addToInventory()


CFD for addToInventory()

```
{
...
try {
         rset = datahandler.executeQuery(//list of
parameters);
         if(rset.next()) {
            datahandler.executeQuery(//list of parameters);
            datahandler.close();
            return searchBookInStock(null, bookInStock.title,
bookInStock.author, bookInStock.publisher).get(0).getISBN()
+ " (already present)";
         }
         else {
            datahandler.executeQuery(//list of parameters);
               }
      }
      catch(SQLException e) {
         datahandler.close();
         throw e;
      }
      datahandler.close();
      return isbn;
}
```

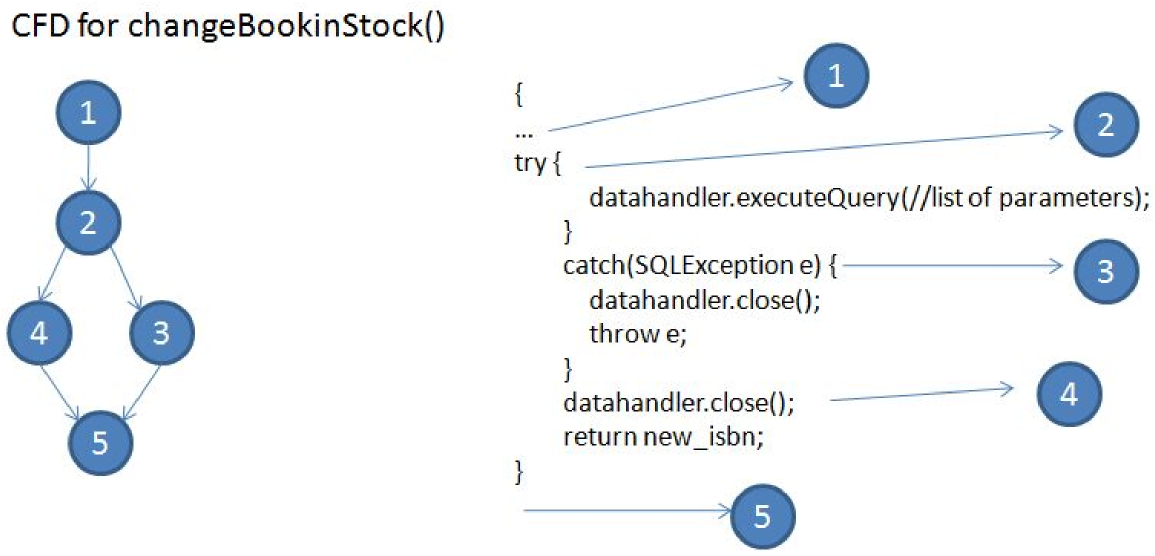
White Box Testing for removeBookInStock()


CFD for removeBookinStock()

```
{
...
try {
        rset = datahandler.executeQuery("select copy_num
from Stock where ISBN = '" + ISBN +"'");
        while(rset.next()) {
          if(rset.getInt(1) == copyNum) {
            datahandler.executeQuery("delete from Stock
where ISBN = '" + ISBN + "'");
          }
          else if(rset.getInt(1) > copyNum) {
            datahandler.executeQuery("update Stock set
copy_num = '" + (rset.getInt(1) - copyNum) + "' where isbn =
'" + ISBN + "'");
          }
          else {
            throw new
BooksOutOfRangeException(copyNum, rset.getInt(1));
          }
        }
    }
    catch(SQLException e) {
      datahandler.close();
      throw e;
    }
    datahandler.close();
    return true;
}
```

Bug Found:Even if the book is not in stock,it still returns true.

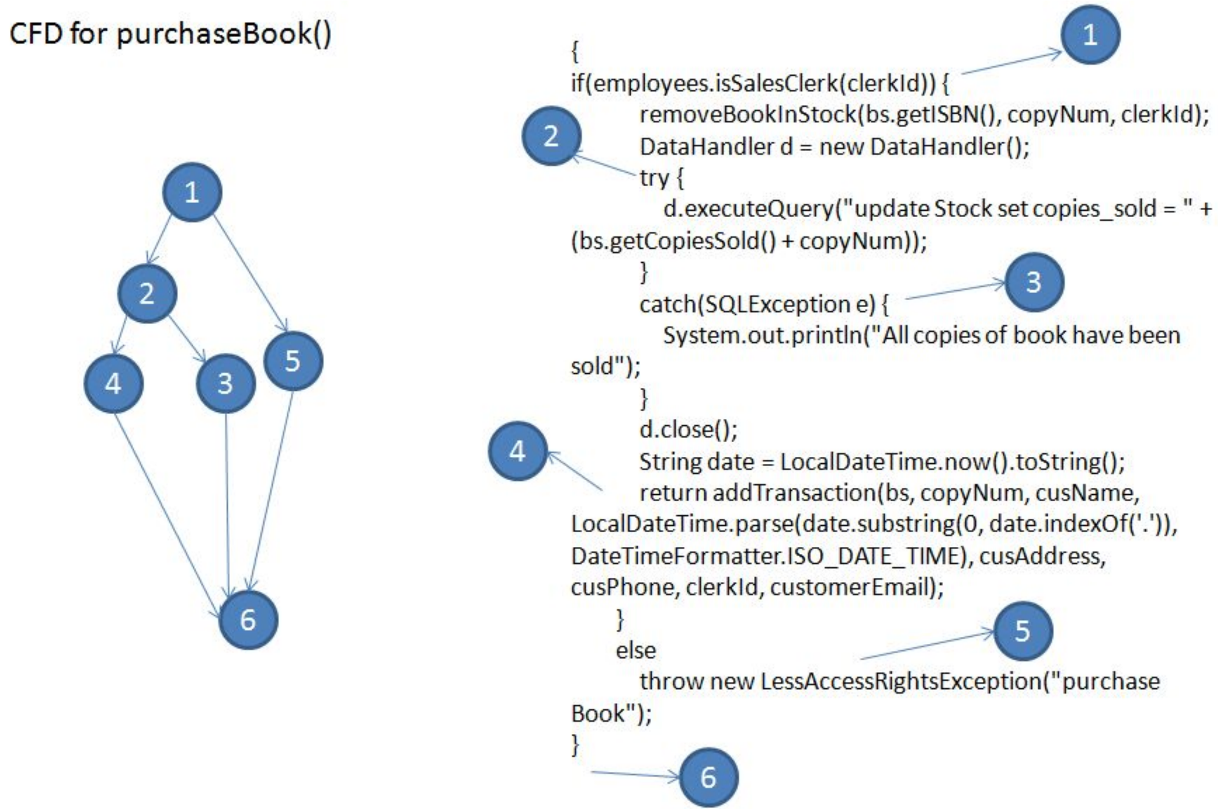
White Box Testing for changeBookInStock()



CFD for changeBookinStock()

```
{
...
try {
    datahandler.executeQuery(//list of parameters);
  }
  catch(SQLException e) {
    datahandler.close();
    throw e;
  }
  datahandler.close();
  return new_isbn;
}
```

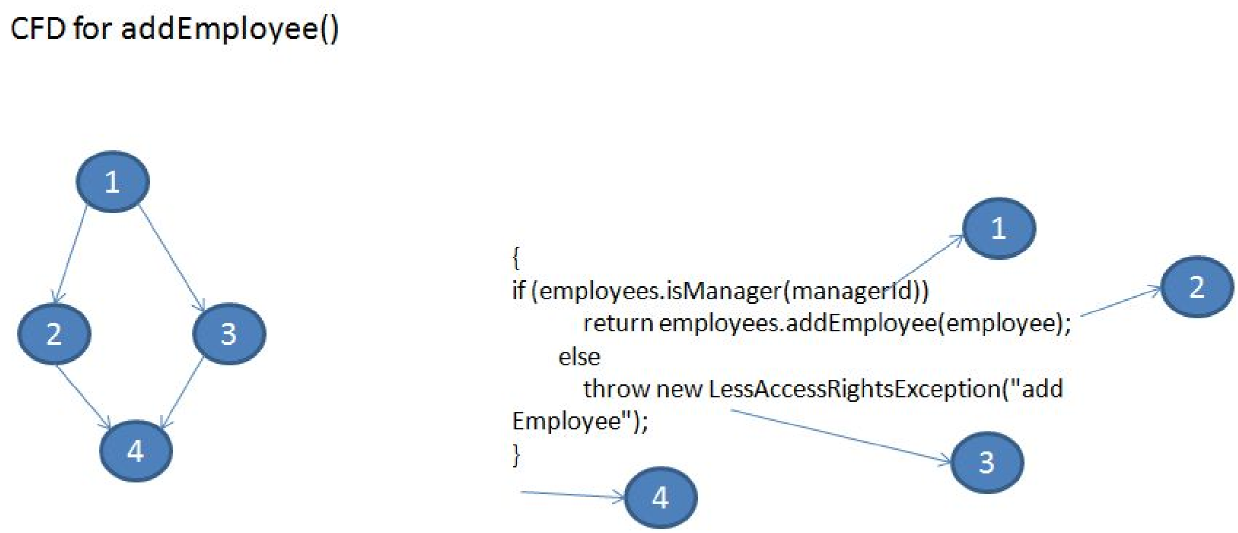
White Box Testing for purchaseBook()



CFD for purchaseBook()

```
{
if(employees.isSalesClerk(clerkId)) {
        removeBookInStock(bs.getISBN(), copyNum, clerkId);
        DataHandler d = new DataHandler();
        try {
          d.executeQuery("update Stock set copies_sold = " +
(bs.getCopiesSold() + copyNum));
        }
        catch(SQLException e) {
          System.out.println("All copies of book have been
sold");
        }
        d.close();
        String date = LocalDateTime.now().toString();
        return addTransaction(bs, copyNum, cusName,
LocalDateTime.parse(date.substring(0, date.indexOf('.')),
DateTimeFormatter.ISO_DATE_TIME), cusAddress,
cusPhone, clerkId, customerEmail);
      }
      else
        throw new LessAccessRightsException("purchase
Book");
}
```

White Box Testing for addEmployee()



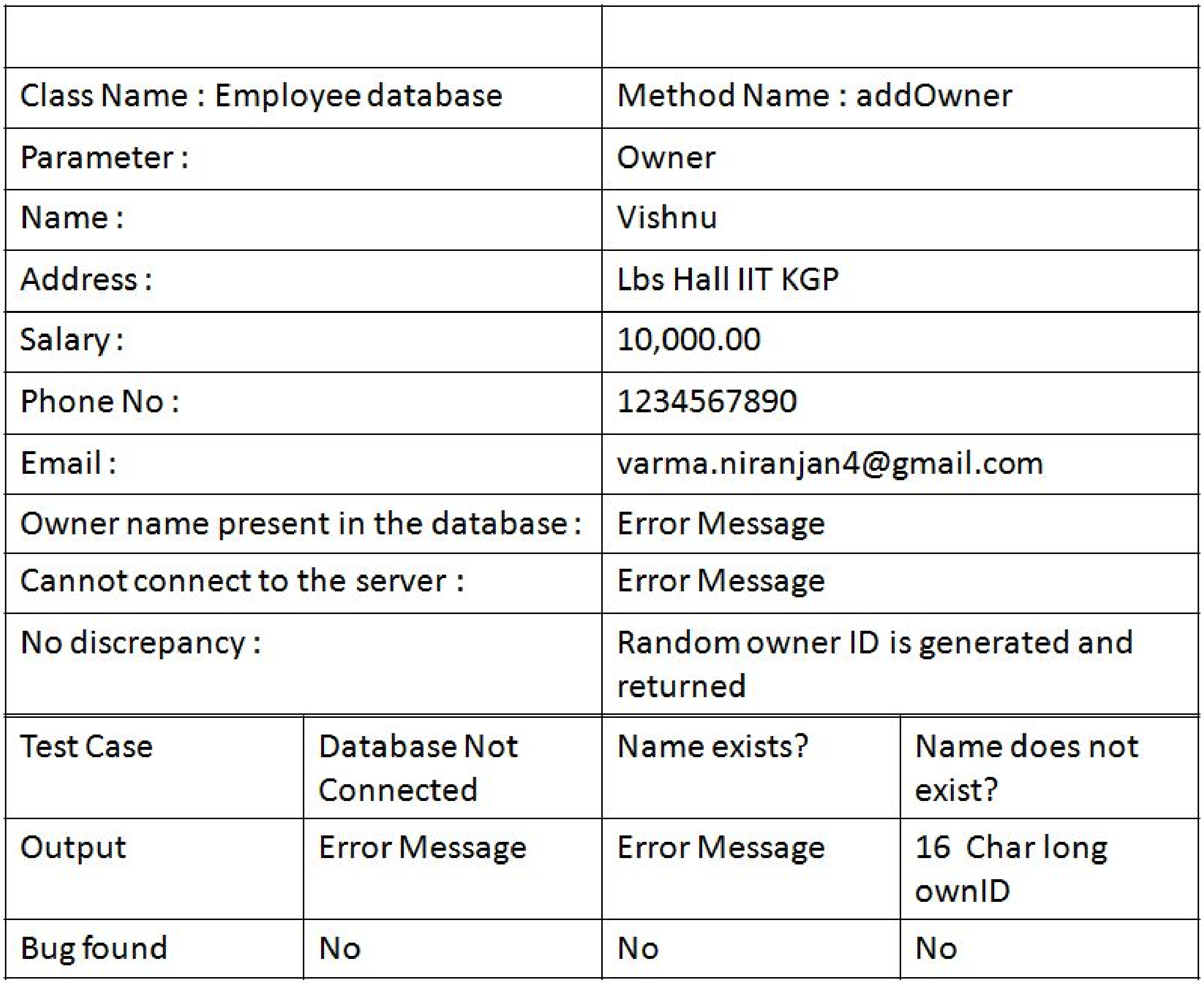
CFD for addEmployee()

```
{
if (employees.isManager(managerId))
        return employees.addEmployee(employee);
     else
        throw new LessAccessRightsException("add
Employee");
}
```

Black Box Testing

This testing is done to ensure proper output is obtained for every test case and is not concerned with internal functionalities of the program.Appropriate equivalence class cases and boundary cases are checked for in such a testing

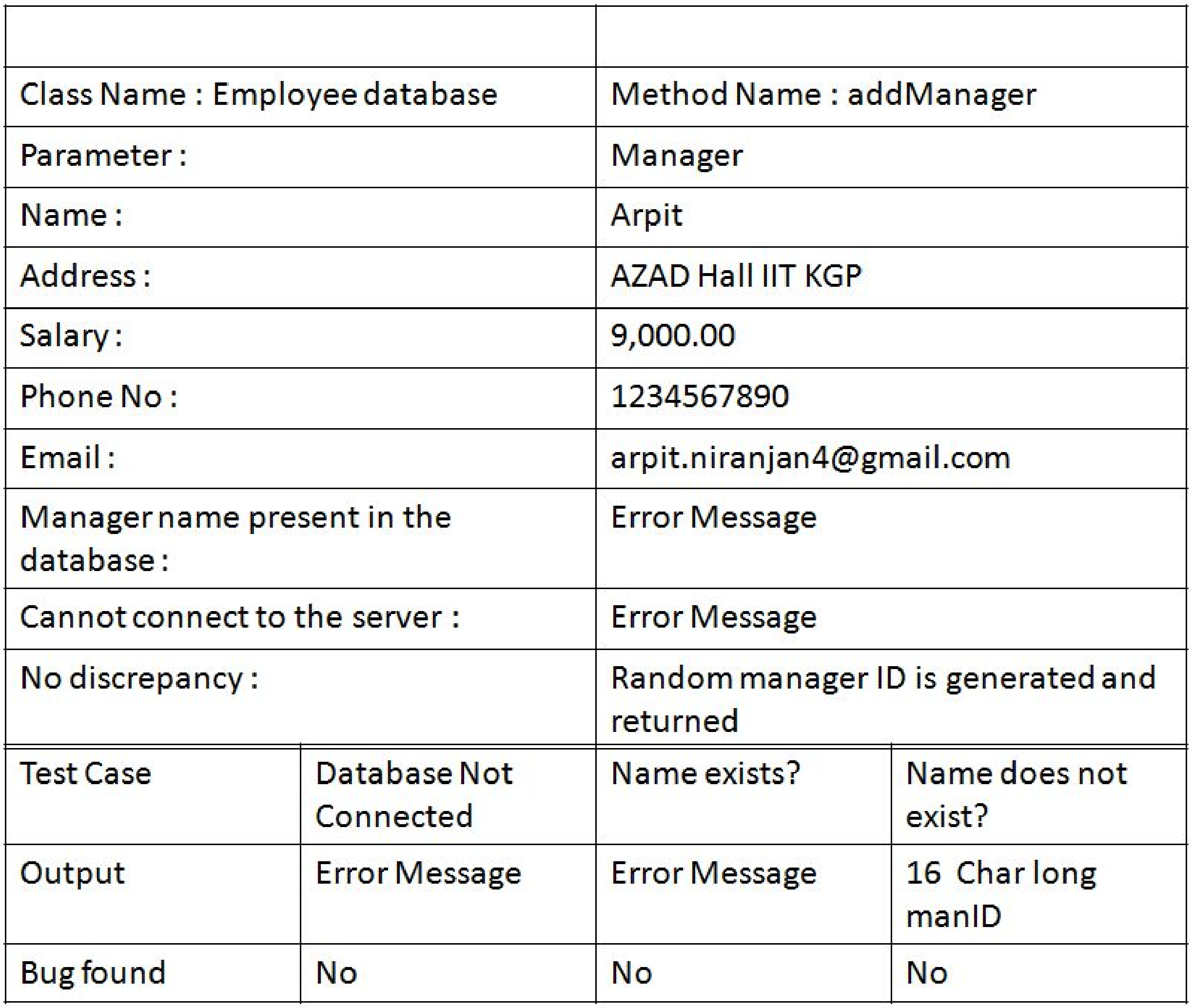
Black Box testing for addOwner()



Class Name : Employee database	Method Name : addOwner
Parameter :	Owner
Name :	Vishnu
Address :	Lbs Hall IIT KGP
Salary :	10,000.00
Phone No :	1234567890
Email :	varma.niranjan4@gmail.com
Owner name present in the database :	Error Message
Cannot connect to the server :	Error Message
No discrepancy :	Random owner ID is generated and returned

Test Case	Database Not Connected	Name exists?	Name does not exist?
Output	Error Message	Error Message	16 Char long ownID
Bug found	No	No	No

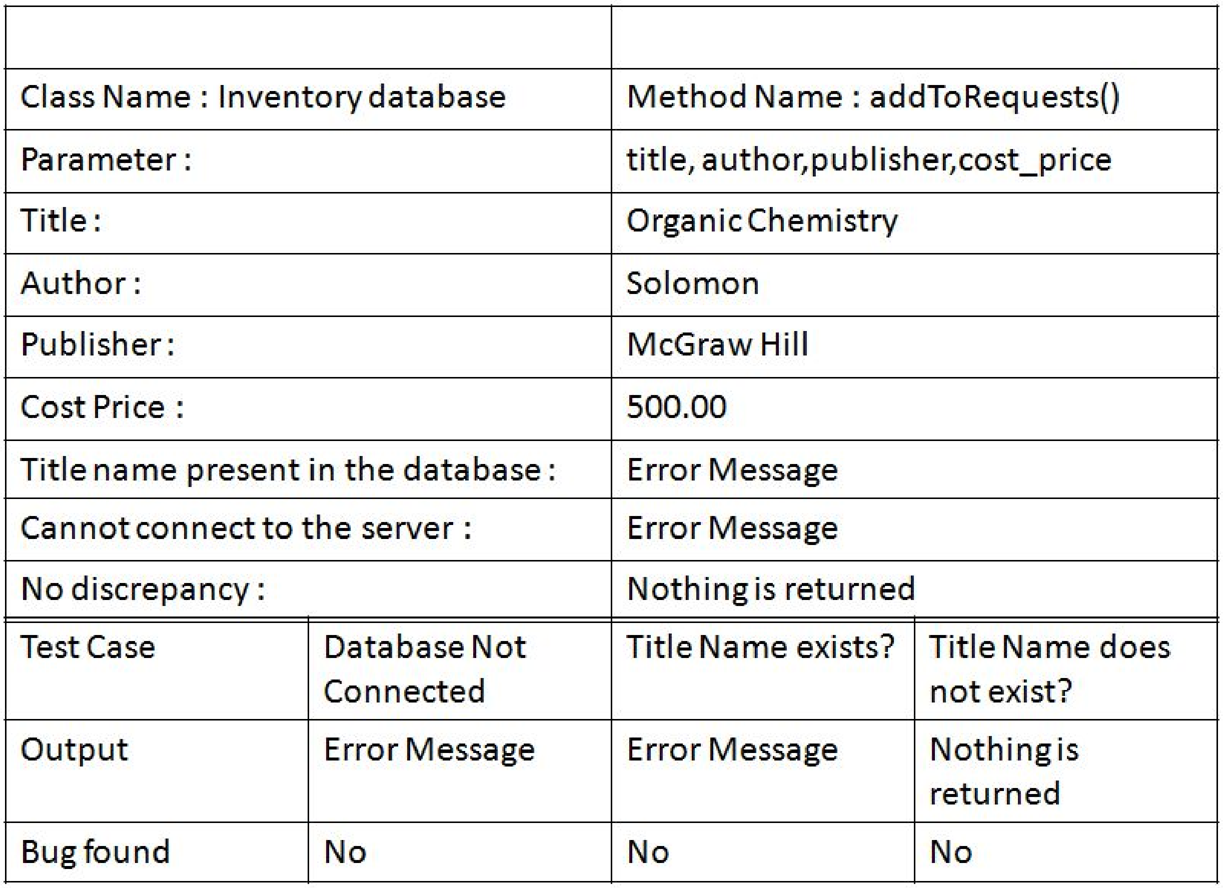
Black Box Testing for addManager()



Class Name : Employee database	Method Name : addManager
Parameter :	Manager
Name :	Arpit
Address :	AZAD Hall IIT KGP
Salary :	9,000.00
Phone No :	1234567890
Email :	arpit.niranjan4@gmail.com
Manager name present in the database :	Error Message
Cannot connect to the server :	Error Message
No discrepancy :	Random manager ID is generated and returned

Test Case	Database Not Connected	Name exists?	Name does not exist?
Output	Error Message	Error Message	16 Char long manID
Bug found	No	No	No

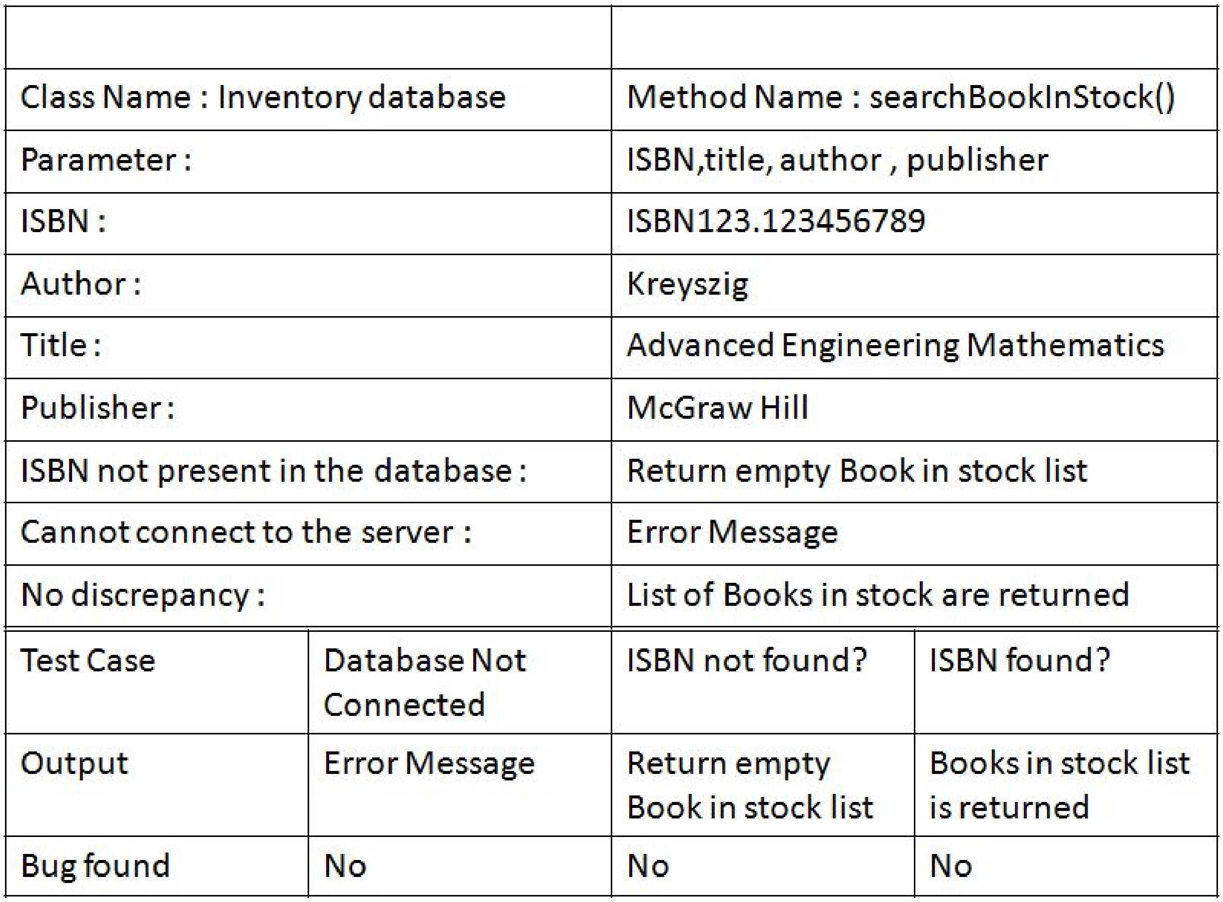
Black Box testing for addToRequests()



Class Name : Inventory database	Method Name : addToRequests()
Parameter :	title, author,publisher,cost_price
Title :	Organic Chemistry
Author :	Solomon
Publisher :	McGraw Hill
Cost Price :	500.00
Title name present in the database :	Error Message
Cannot connect to the server :	Error Message
No discrepancy :	Nothing is returned

Test Case	Database Not Connected	Title Name exists?	Title Name does not exist?
Output	Error Message	Error Message	Nothing is returned
Bug found	No	No	No

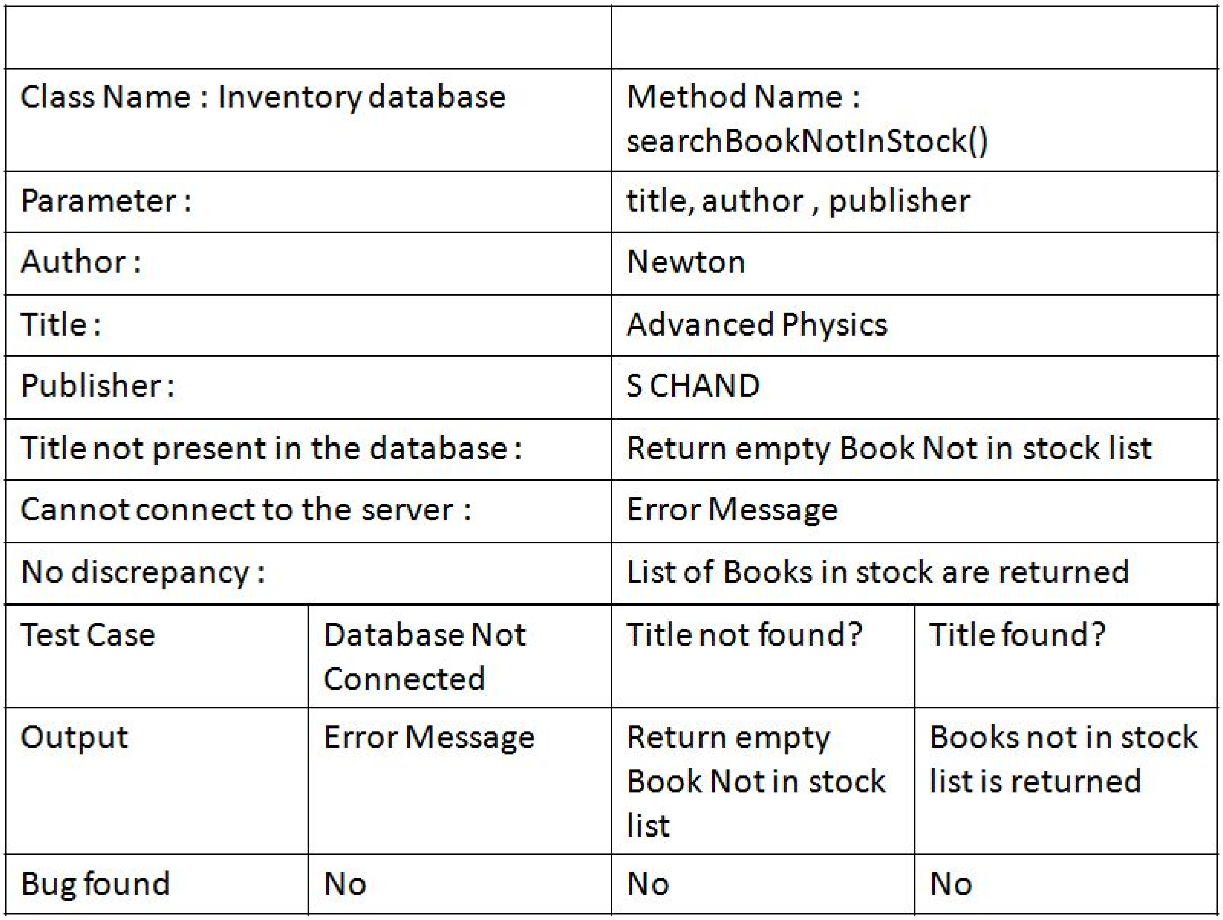
Black Box Testing for searchBookInStock()



Class Name : Inventory database	Method Name : searchBookInStock()
Parameter :	ISBN,title, author , publisher
ISBN :	ISBN123.123456789
Author :	Kreyszig
Title :	Advanced Engineering Mathematics
Publisher :	McGraw Hill
ISBN not present in the database :	Return empty Book in stock list
Cannot connect to the server :	Error Message
No discrepancy :	List of Books in stock are returned

Test Case	Database Not Connected	ISBN not found?	ISBN found?
Output	Error Message	Return empty Book in stock list	Books in stock list is returned
Bug found	No	No	No

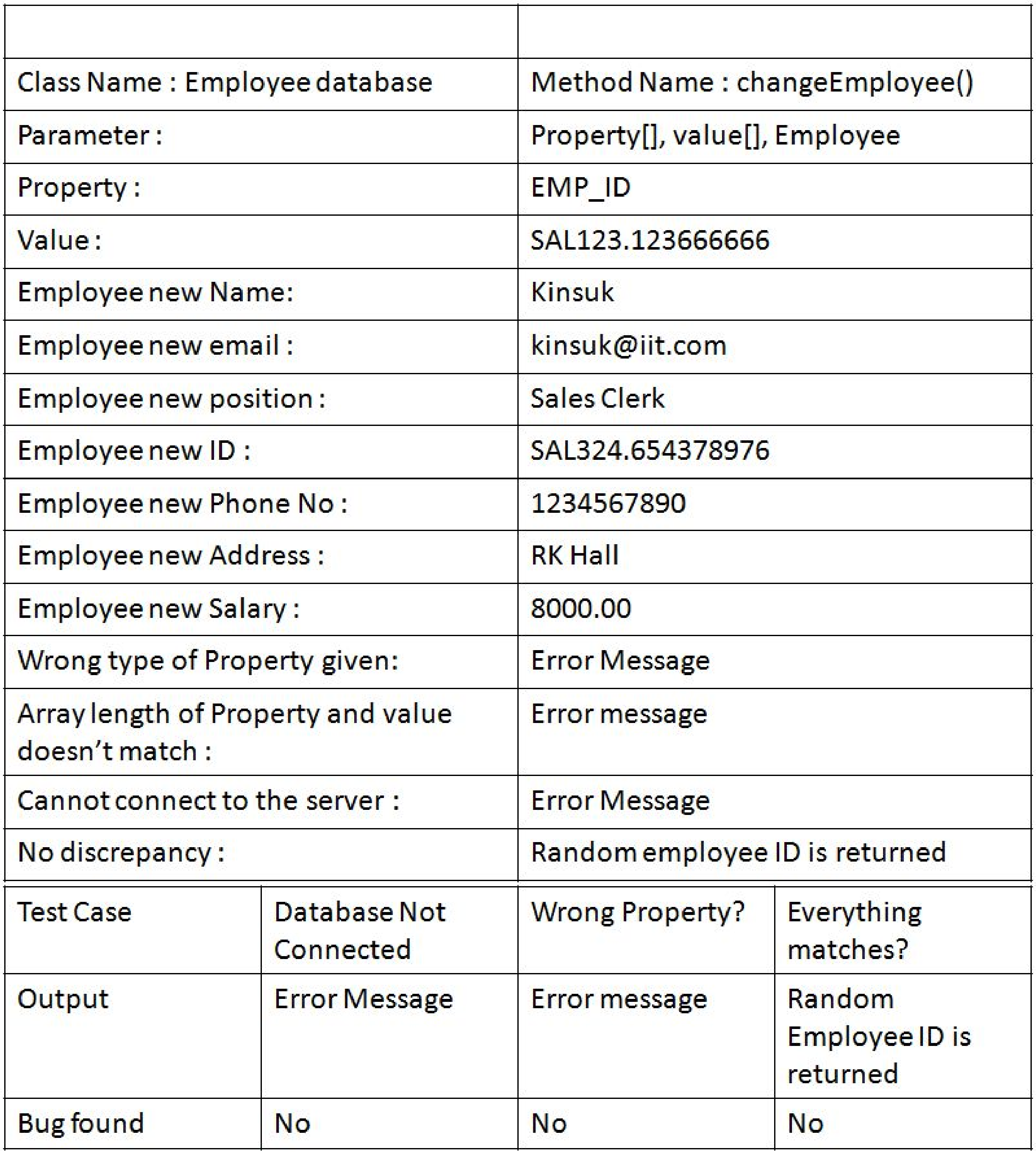
Black Box Testing for searchBookNotInStock()



Class Name : Inventory database	Method Name : searchBookNotInStock()
Parameter :	title, author , publisher
Author :	Newton
Title :	Advanced Physics
Publisher :	S CHAND
Title not present in the database :	Return empty Book Not in stock list
Cannot connect to the server :	Error Message
No discrepancy :	List of Books in stock are returned

Test Case	Database Not Connected	Title not found?	Title found?
Output	Error Message	Return empty Book Not in stock list	Books not in stock list is returned
Bug found	No	No	No

Black Box Testing for changeEmployee()



Class Name : Employee database	Method Name : changeEmployee()
Parameter :	Property[], value[], Employee
Property :	EMP_ID
Value :	SAL123.123666666
Employee new Name:	Kinsuk
Employee new email :	kinsuk@iit.com
Employee new position :	Sales Clerk
Employee new ID :	SAL324.654378976
Employee new Phone No :	1234567890
Employee new Address :	RK Hall
Employee new Salary :	8000.00
Wrong type of Property given:	Error Message
Array length of Property and value doesn't match :	Error message
Cannot connect to the server :	Error Message
No discrepancy :	Random employee ID is returned

Test Case	Database Not Connected	Wrong Property?	Everything matches?
Output	Error Message	Error message	Random Employee ID is returned
Bug found	No	No	No

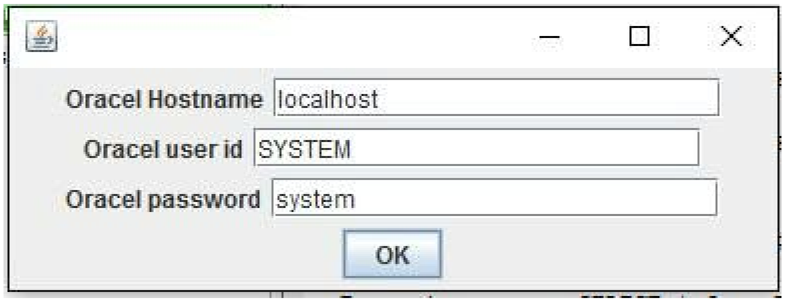
Integration Testing

Graphic User Interface (GUI) Module :

This part shows the testing of the software showing how the user interface works.

User Interface Testing :

Login Window:



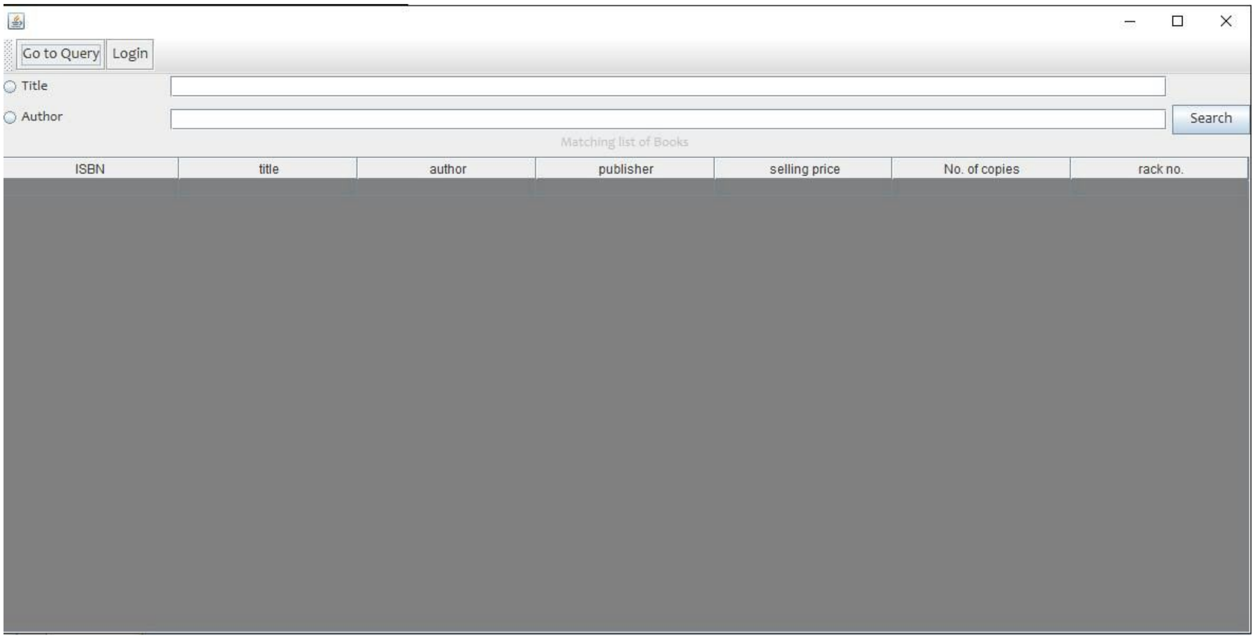
Oracel Hostname localhost

Oracel user id SYSTEM

Oracel password system

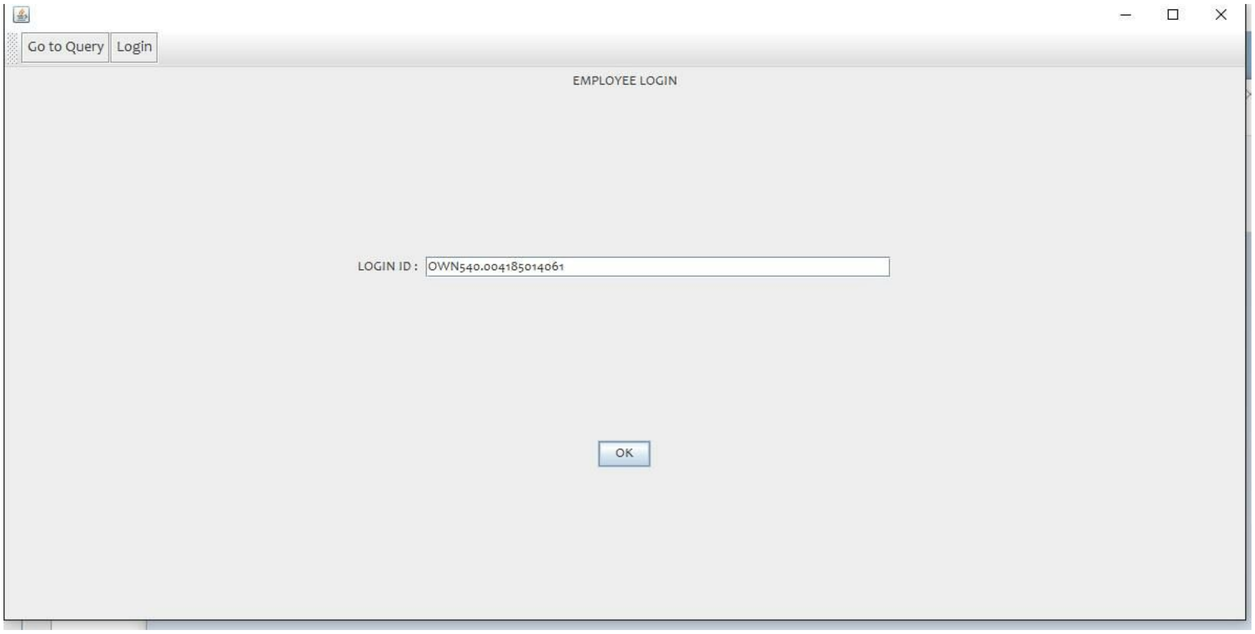
What is Tested?	**Connection to database**
Inputs	**Hostname,username,password**
Success scenario	**Image below opens up.**
Failure scenario	**Nothing happens**

Main Page :



ISBN	title	author	publisher	selling price	No. of copies	rack no.

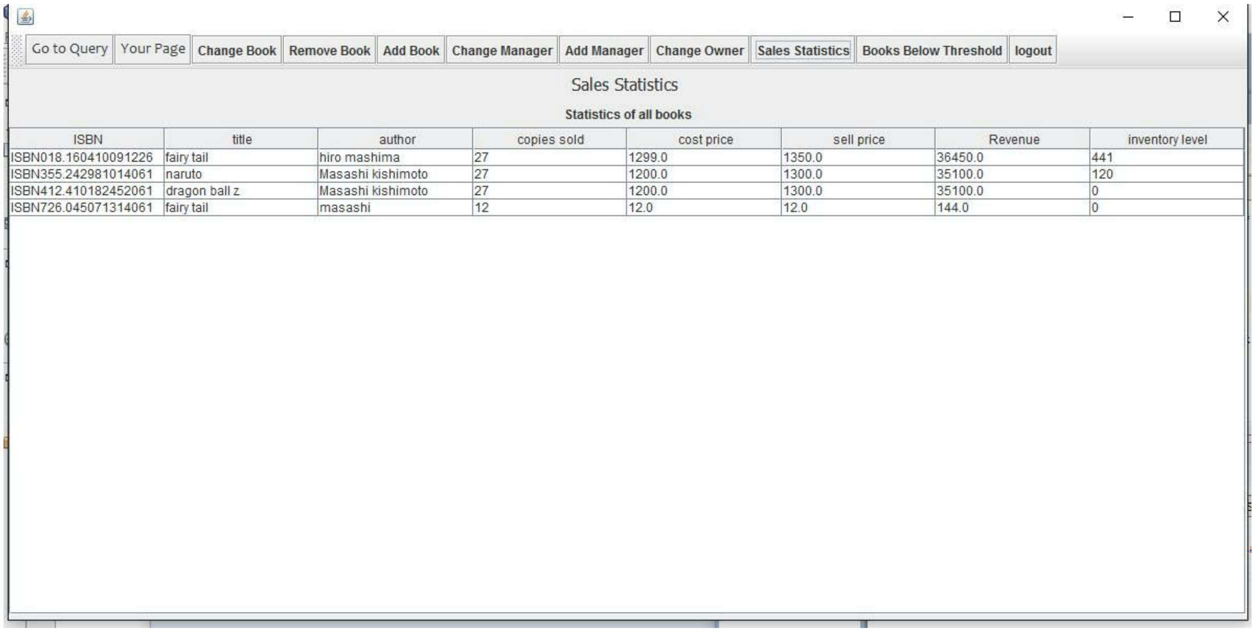
Employee Login



LOGIN ID : OWN540.004185014061

What is Tested?	**Employee ID**
Inputs	**Employee ID**
Success scenario	**Id present in database**
Failure scenario	**Error message**

Sales Statistics Successful :



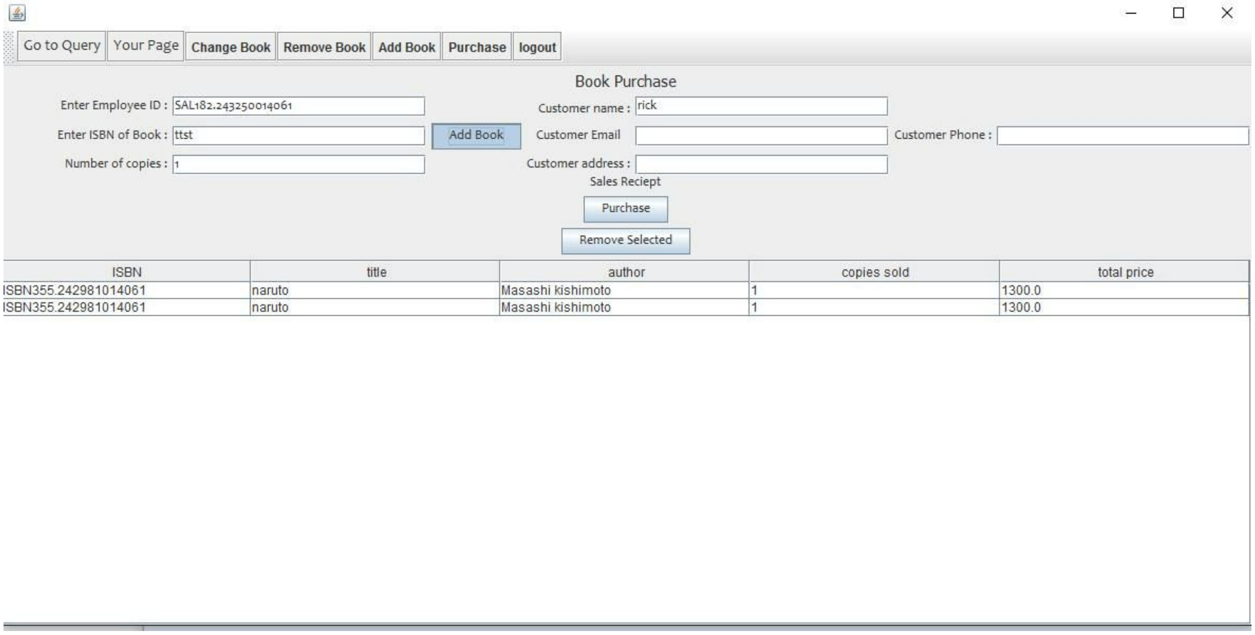
Sales Statistics

Statistics of all books

ISBN	title	author	copies sold	cost price	sell price	Revenue	inventory level
ISBN018.160410091226	fairy tail	hiro mashima	27	1299.0	1350.0	36450.0	441
ISBN355.242981014061	naruto	Masashi kishimoto	27	1200.0	1300.0	35100.0	120
ISBN412.410182452061	dragon ball z	Masashi kishimoto	27	1200.0	1300.0	35100.0	0
ISBN726.045071314061	fairy tail	masashi	12	12.0	12.0	144.0	0

What is Tested?	**Owner ID**
Inputs	**Owner ID**
Success scenario	**All sales statistics are generated**
Failure scenario	**Error Message**

Purchase :



Book Purchase

Enter Employee ID : SAL182.243250014061

Enter ISBN of Book : ttst

Number of copies : 1

Customer name : rick

Customer Email

Customer Phone :

Customer address :

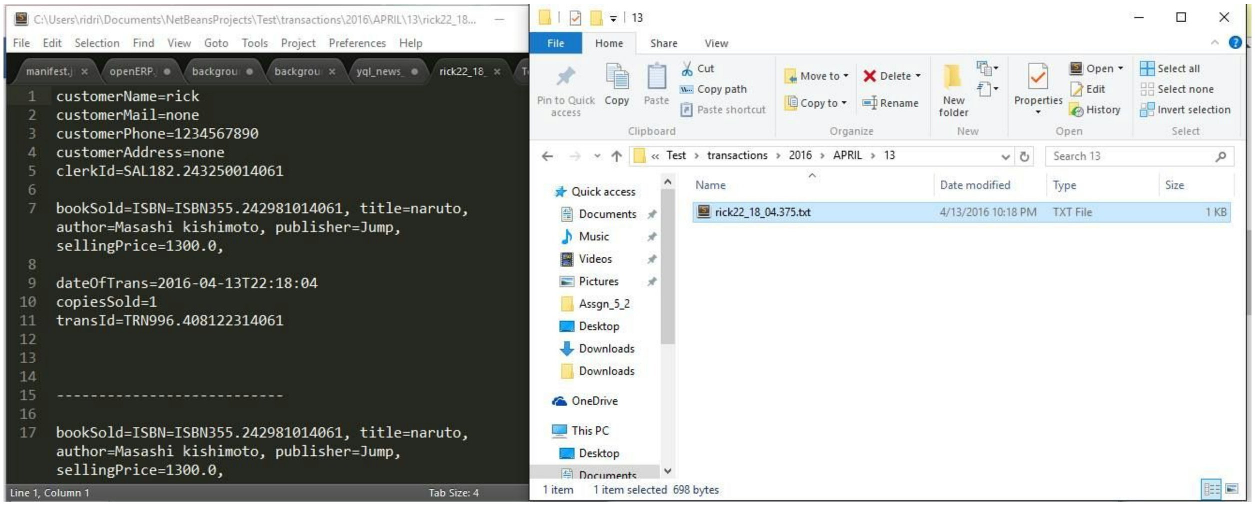
Sales Reciept

ISBN	title	author	copies sold	total price
ISBN355.242981014061	naruto	Masashi kishimoto	1	1300.0
ISBN355.242981014061	naruto	Masashi kishimoto	1	1300.0

What is Tested?	**Valid book purchase**
Inputs	**Emp ID** **ISBN** **Number of Copies** **Customer name** **Customer address** **Customer Phone No.**

Success scenario	**The transaction txt file is saved**
Failure scenario	**Error message.**

Purchase Success scenario



C:\Users\ridri\Documents\NetBeansProjects\Test\transactions\2016\APRIL\13\rick22_18...

```
customerName=rick
customerMail=none
customerPhone=1234567890
customerAddress=none
clerkId=SAL182.243250014061

bookSold=ISBN=ISBN355.242981014061, title=naruto,
author=Masashi kishimoto, publisher=Jump,
sellingPrice=1300.0,

dateOfTrans=2016-04-13T22:18:04
copiesSold=1
transId=TRN996.408122314061



--------------------------

bookSold=ISBN=ISBN355.242981014061, title=naruto,
author=Masashi kishimoto, publisher=Jump,
sellingPrice=1300.0,
```

Line 1, Column 1 Tab Size: 4

Test › transactions › 2016 › APRIL › 13

Name	Date modified	Type	Size
rick22_18_04.375.txt	4/13/2016 10:18 PM	TXT File	1 KB

1 item 1 item selected 698 bytes

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case we focused only on function validation and performance. And in both cases we used the black-box method of testing.

In addition, we tested:

- The interfaces to ensure they are functioning as desired (i.e. check if each interface is behaving as expected, specifically verifying the appropriate action is associated with each mouse_click event).

- The interaction between the GUI and the backend controller classes. In this case the data will be inserted and check if they are sent to the server properly and the expected results are obtained.

Performance testing

This test was conducted to evaluate the fulfillment of a system with specified performance requirements. It was done using black-box testing method.
Following things were tested:
Adding large number of products to the database to see how much time it takes to retrieve them from the server.
Logging in large number of employees from various terminals to test the maximum stress which the server can handle without substantial loss in performance.
Calculating the sales statistics for a very large sales history to test the performance of chart generation.

Deliverables

- Program function specifications

- Program source code

- Test plan document - this document should address testing objectives, criteria, standards, schedule and assignments, and testing tools.
  Unit Testing Plan
  Integration Plan
  System Testing Plan

Environmental Needs

As the project has been developed in Java, the software works in both Windows and Linux platforms.
MySQL is also required to function to be able to connect to the database.