Assignment Cover Sheet

Qualification		Module Number and Title
HND in Computing/HND in Software Engineering		**Introduction to OOP- SEC4207**
Student Name & No.		**Assessor**
M.A. Chamath Shyamal & CL/HDCSE/95/43		Upeka Wijeshinge
Hand out date		**Submission Date**
02/07/2021		24/07/2021
Assessment type WRIT1-Coursework	**Duration/Length of Assessment Type** 3000 words equivalent	**Weighting of Assessment** 100%

Learner declaration
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Marks Awarded			
First assessor			
IV marks			
Agreed grade			
Signature of the assessor		Date	

FEEDBACK FORM

INTERNATIONAL COLLEGE OF BUSINESS & TECHNOLOGY

Module:

Student:

Assessor:

Assignment:

Strong features of your work:

Areas for improvement:

Marks Awarded:

Learning outcomes covered

LO1: Explain the fundamentals of Object-Oriented Programming concepts

LO2: Design Object-Oriented based applications

LO3: Develop Object-Oriented applications

Scenario and the Task

"City Bookshop" is one of the new bookshop located in the heart of the city. The company planned to implement an application in order to automate transaction process

User levels and functionalities are follows

Cashier

1. View all the book details
2. Add new book details and category
3. Search book details based on category, Name, Price etc

Manager: (can perform all the functionality as cashier plus the following)

1. Create a new account (can have different account types)
2. Create new user account for cashier

You are required to apply OOP concepts for the above scenario. Data need to be saved and retrieved from a File

Part A: Report

Task 1.Provide design solution (UML diagrams) for the above mention Scenario. Provide clear explanation for all the diagrams mention below. (Provide assumption if necessary) **(30 marks) (LO2)**

a) Use case Diagram
b) Class Diagram
c) Sequence Diagram

Task 2: Develop suitable system for the above scenario based on the design. Required to use Object Oriented concepts (Object, Class, Abstraction, Inheritance, Encapsulation and Polymorphism) for the development. Document the main functionalities and Object Oriented concepts applied with proper explanation and source code. **(Marks 20) (LO1, LO3)**

Task 3: Provide a user manual for the developed solution **(Marks 10) (LO3)**

Guidelines for the report format

Paper A4
Margins 1.5” left, 1” right, top and bottom
Page numbers – bottom, right
Line spacing 1.5
Font
Headings 14pt, Bold
Normal 12pt
Font face- Times New Roman

Part B: Demonstration

Task 4: System demonstration. System should work according to the expected functionalities. Should be able to demonstrate Object Oriented concepts (Object, Class, Abstraction, Inheritance, Encapsulation, and Polymorphism) applied to the given scenario. **(Marks 40) (LO1, LO3)**

Acknowledgement

Primarily I thank God for being able to complete this assignment in a successful manner. Thus, I take this opportunity to express my deep sense of gratitude and my profound respect to the lecturer who guided and inspired me in doing this assignment. I had to get the guidance of a respected and responsible person at the preparation time and while continuing the assignment. So, I would like to thank our lecturer Miss. Upeka Wijeshinge whose valuable guidelines and consultations been the ones that helped me patch this assignment and make this full proof success and finalize this successfully.

And also, her instructions which were given underlying the structure has served as the major contributor towards in completing this assignment and her instructions about the programming and coding was more helpful in implementing this City Bookshop's system. Through those and by this golden opportunity, I got the full knowledge on basics and some advance in java Programming. I hope this knowledge you gave me will help me in learning more advance java programming as well as my career. I really thankful for you because of the massive courage you had to teach us.

Then I would like to thank Mr. Chathura, Miss Manoda and Miss Ishani who is with us and help us in many sides since the beginning of the Bridging Program. However finally, I am really grateful because I managed to complete this assignment within the time period given by our lecturer Miss. Upeka Wijeshinge. Thank you all!

Contents

Introduction

This Assignment is regarding the Introduction to Object Oriented Programming Module and this is the seventh assignment we got in our HD Program. Basically, this coursework covers fundamentals of Object-Oriented Programming concepts, Designing of Object-Oriented based applications and developing of Object-Oriented applications. Studying about Object Oriented Programming is more important for a student who hopes to become a Software Engineer as Java programming directly explains the basics in building software and how complex codes are made using the basic stuffs. In my assignment, I have ideally shown some theory used along with the evidences from the developed system for City Bookshop.

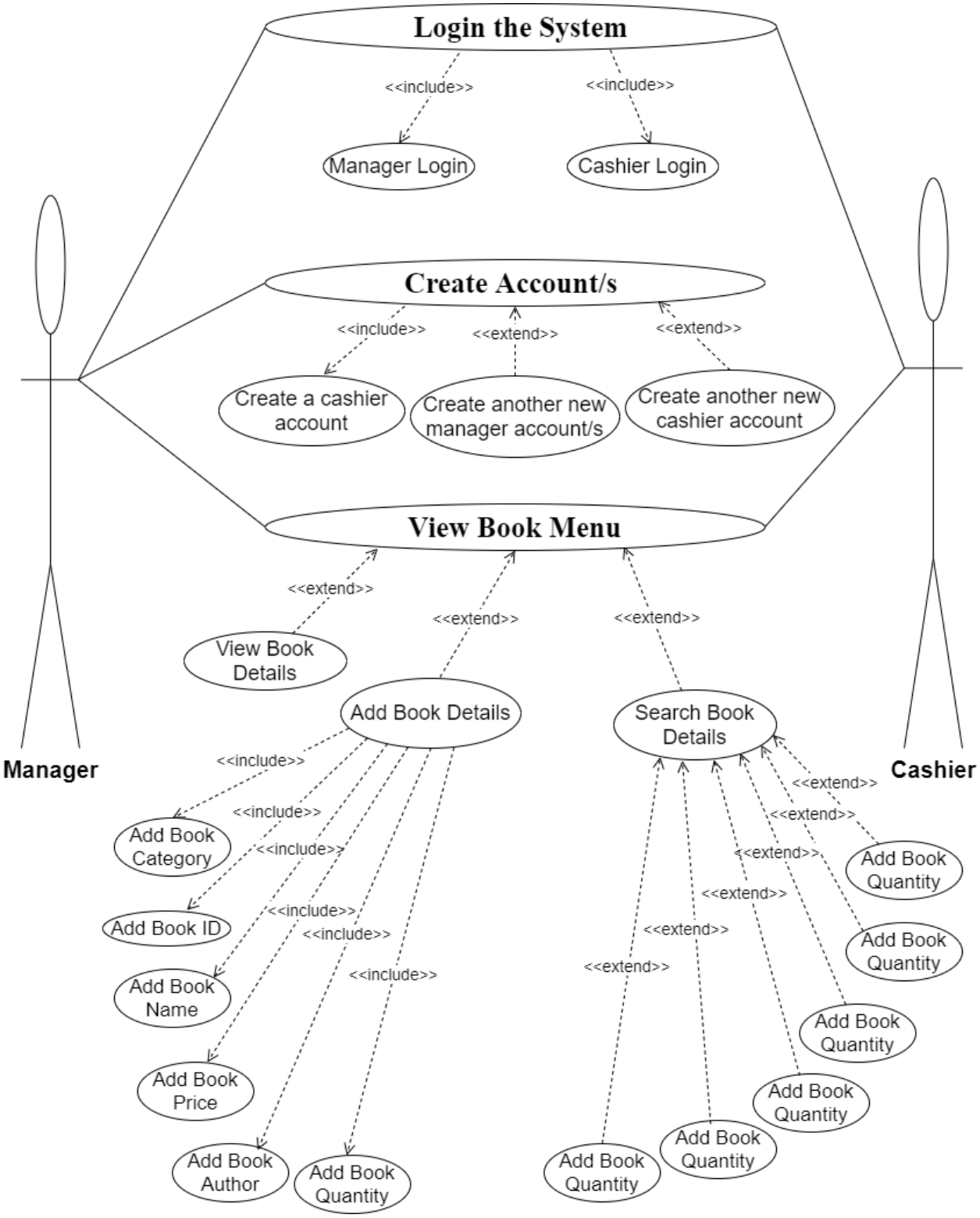
The primary and main objective of implementing this application for City Bookshop is to automate their transaction process. City Bookshop is one of the new bookshops located in the heart of the city. Therefore, I had to include OOP concepts which are more considerable when using an Object-Oriented Programming language like Java. Thus, they hope to make it easier their transaction process by an automated application. Mainly, there are two users who can use this system. They are Manager of this City Bookshop and Cashier. Basically, those two users can perform functions such as Login, creating new accounts, viewing of book details, adding new book details, searching for new books and so on.

This document includes explanations about concepts in Object Oriented Programming, three types of UML diagrams as Use Case, Class and Sequence. Further, I have included some screenshots of the coding along with assumptions and descriptions by highlighting the OOP concepts. Moreover, a user manual has provided in this document which makes user of this system more understandable.

Hope this assignment will be a clear and ideal one.

Task 1- UML Diagrams

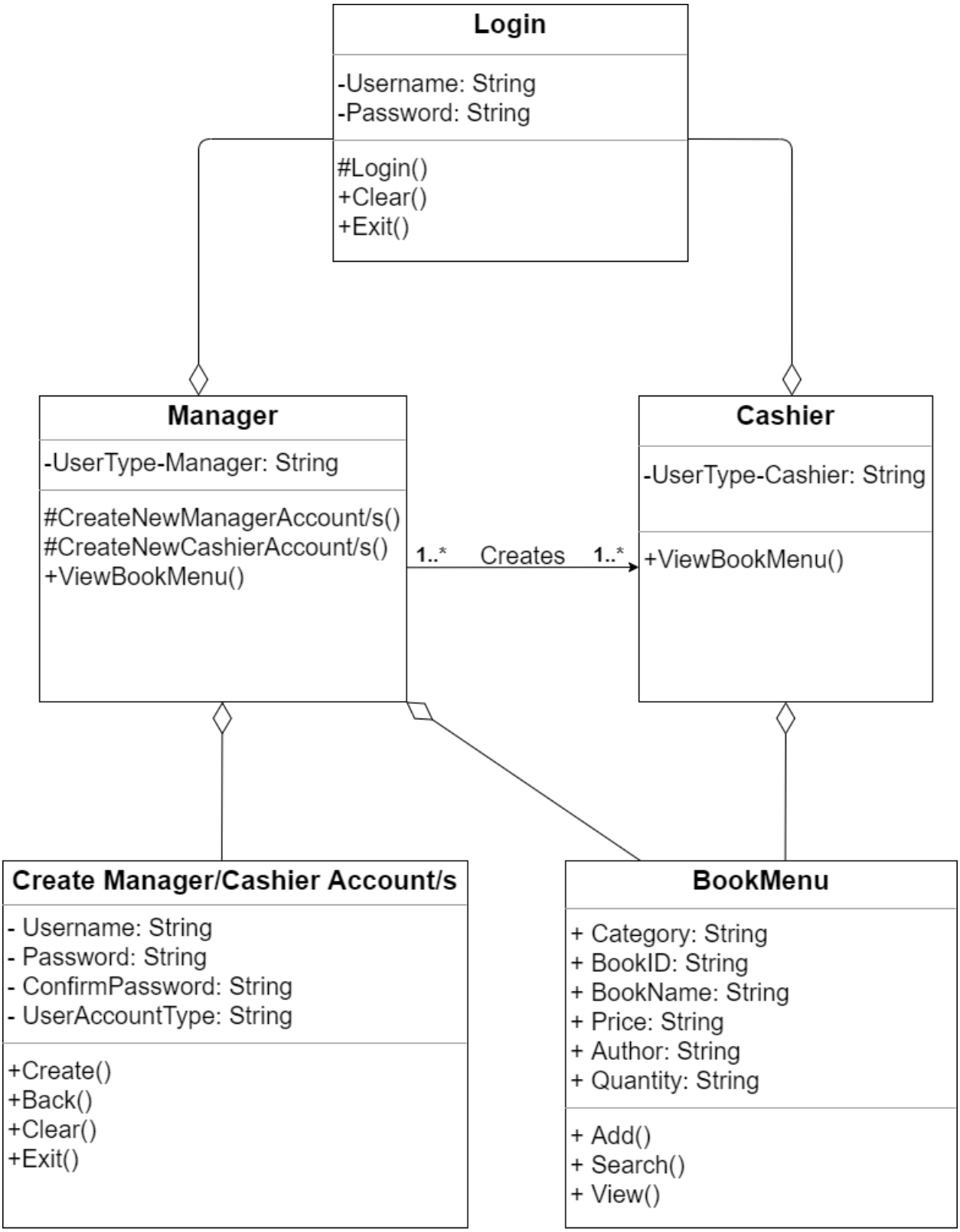
1.1) Use Case Diagram



Use Case Diagram is a type of Unified Modeling Language diagrams. Basically, Use Case diagrams demonstrate the interactions of users/actors with functional requirements of a particular system. Therefore, it is convenient to mention that Use Case diagrams include actors, main use cases and sub use cases mainly.

When it comes to the above Use Case diagram, there are two actors named as Manager and Cashier who are the users of that system. Both of them can perform the function called Login to the system. As login to the system is an essential function in order to continue the other functions in the system, login of those two users have mentioned as include relationships. When carefully observe the above use case diagram, its evident that Manager can perform all the functions in the system while Cashier can perform limited functions. So, manager can login to the system at first an then he/she can create one or multiple accounts for both the users. Common function for both the actors or users in the above use case diagram is View Book Menu. Simply its possible to mention that manager can login at first and create multiple user accounts for both users and then both the users can View Book Menu as a common.

1.2) Class Diagram

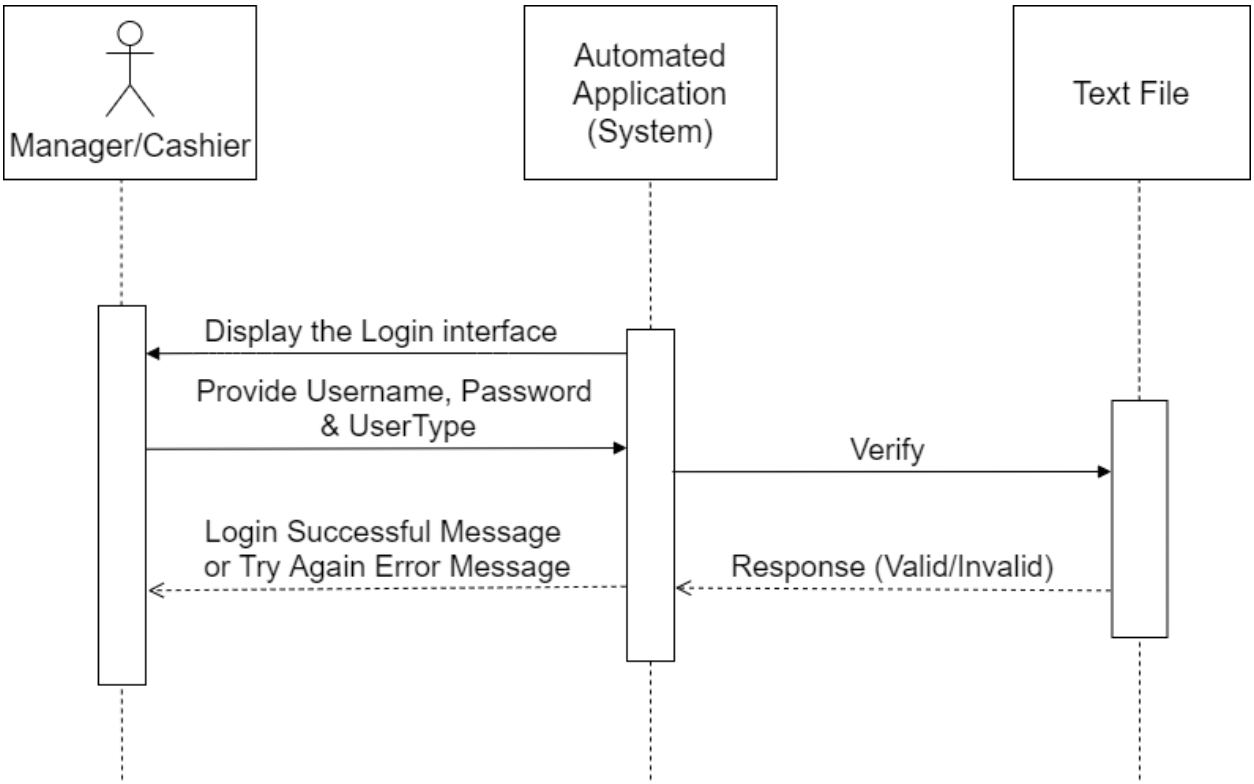


Class Diagram is also a Unified Modeling Language diagrams which depicts the structure of a particular system based on the classes, attributes and methods.

In the above Class Diagram, there can be seen three main classes as Login of two users, Create manager/Cashier Accounts and Book Menu. In that, I have considered Login as a parent class while Manager and Cashier are child classes of that parent class. Further, there can be seen a multiplicity indicator of one to many (1...*) mentioned as create where manager can create many managers accounts and many cashier accounts for the system. Moreover, in the Login parent class login function has been depicted with a protected access modifier as login function should be a protected one. And also, inside the manger child class, both two create new manager account/s an create new cashier account/s methods have included with protected access modifiers in order to protect creating accounts and it can be done only by manager. However, all the methods and attributes inside the Book Menu class are public access modifiers as all those functions and attributes inside that class can be accessed by both the users and other classes. It is visible that in the above class diagram of City Bookshop system's, Book Menu and Create Manager/Cashier Account/s classes have joined with the two child classes in the diagram by using the relationship called Association Relationship.

1.3) Sequence Diagram

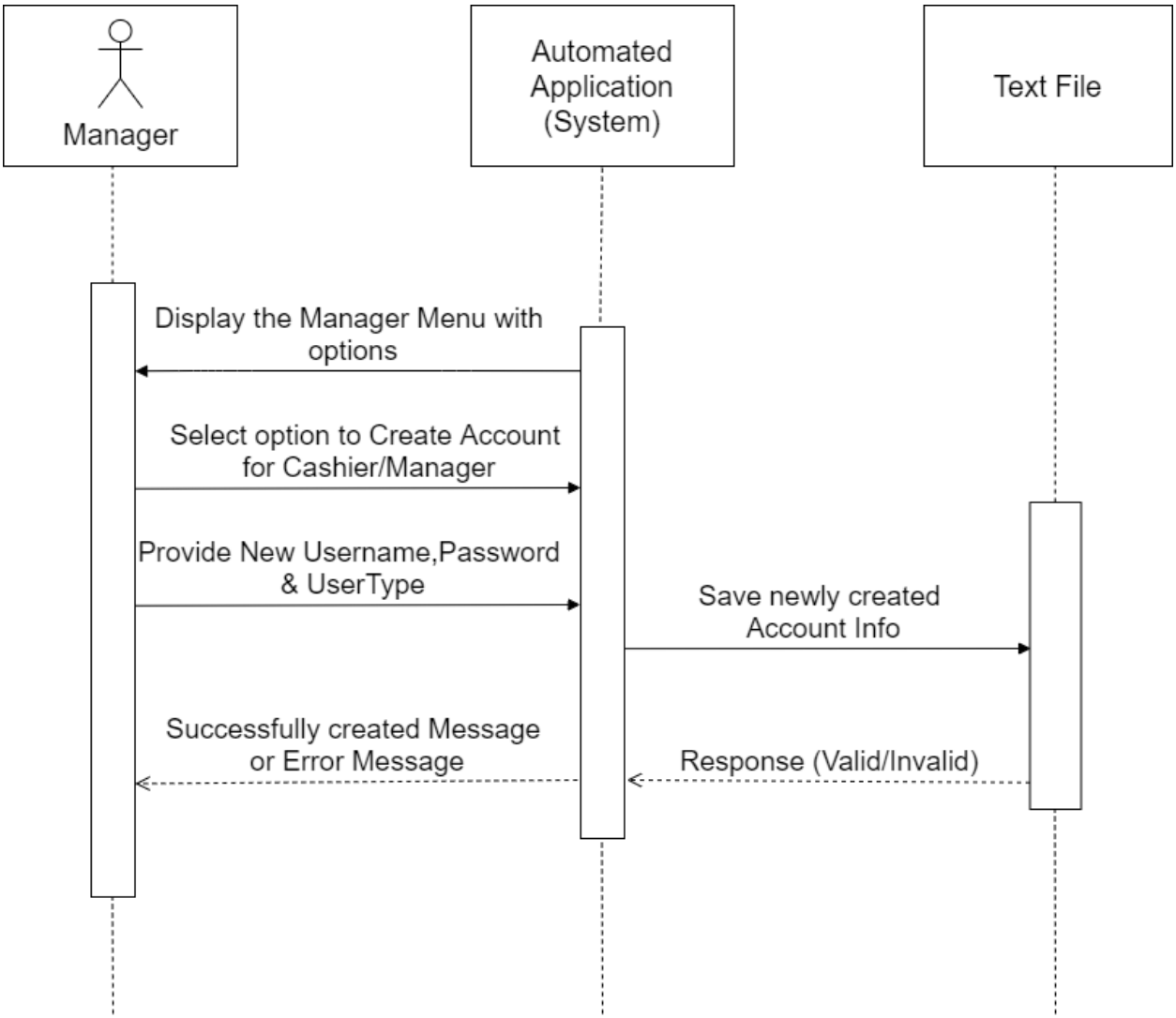
1.3.1) Sequence Diagram for Login



Sequence Diagram is also another kind of a Unified Modeling Language diagrams which describe the flow of messages, events and actions among objects. When it comes to the City Bookshop's System, there can be identified three objects as Manager/Cashier, Automated System and the relevant Text File.

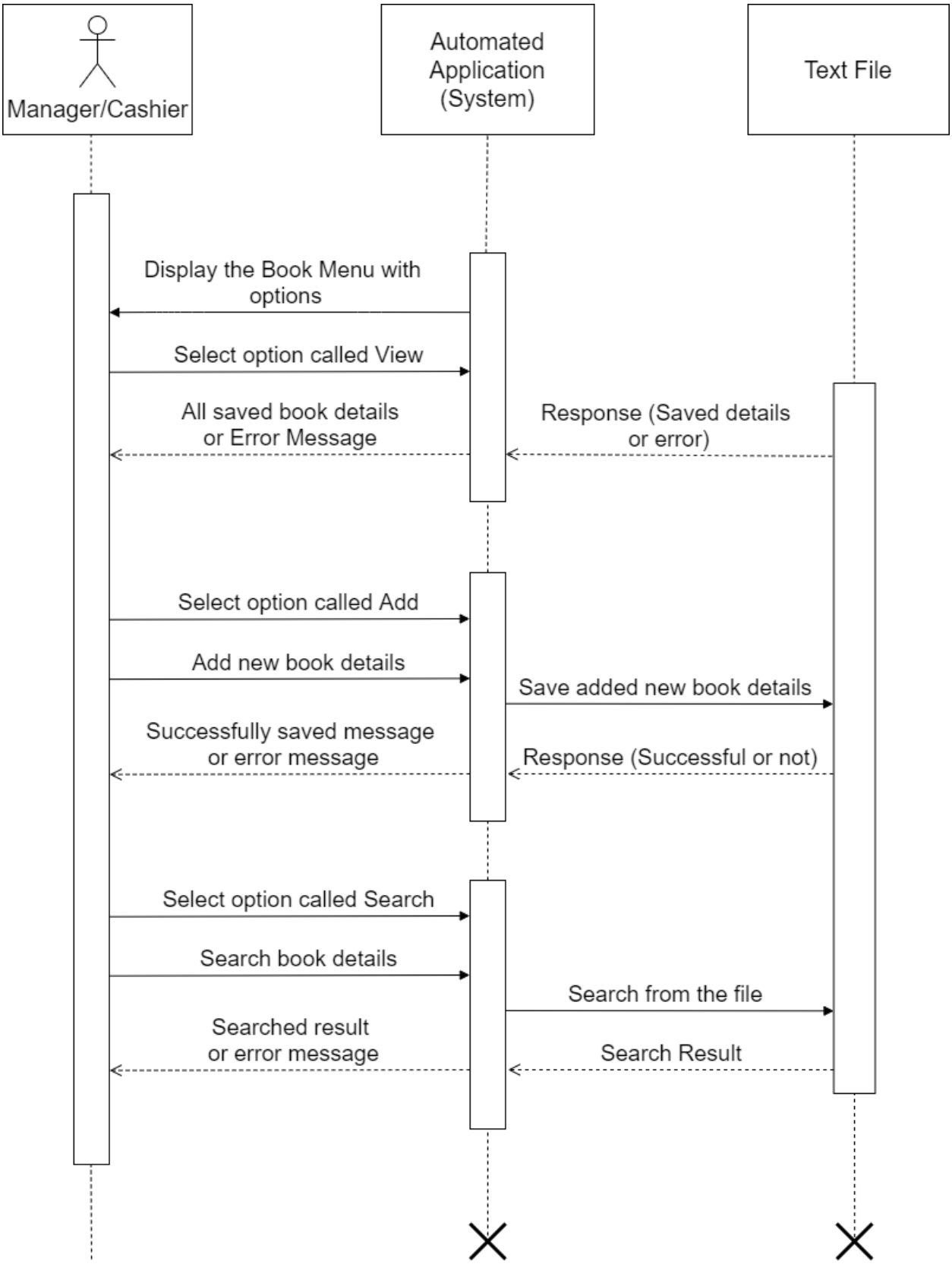
In the above sequence diagram for login, at first the login interface will be displayed by the system for the user. If manager has created an account for cashier, then cashier can provide Username, Password & User Type and login. Otherwise, manager should provide necessary details and login. After user entering the relevant data, the system verifies them with the use of text file and then return the response as Successful or not to the user.

1.3.2) Sequence Diagram for Create Manager/Cashier Account/s



At first it is convenient to mention that this sequence diagram is relatable with manager as only manager can create more new accounts. Thus, in the above sequence diagram for create Manager/Cashier Account/s, the system will display the manager menu primarily after logging. Then manager can select option to create account for Cashier/Manager and provide necessary information. That information will be saved in the relevant file. Finally, a response will be sent to the user (Manager) mentioning whether account is successfully created or not.

1.3.3) Sequence Diagram for Book Menu (View, Add & Search)



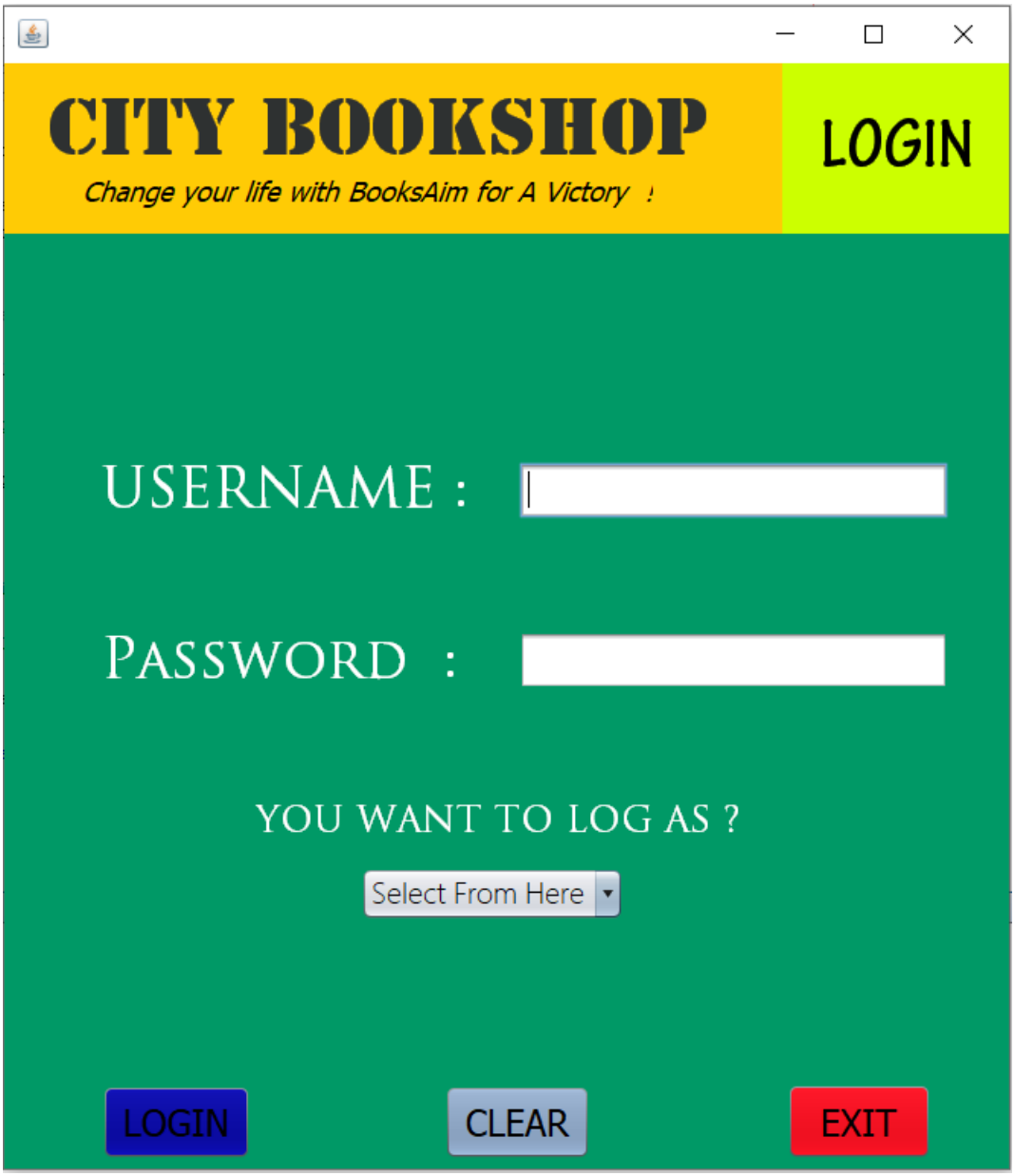
As in the above sequence diagram for Book Menu, its visible that there can be seen three functions in the system as View, Add and Search book details. All the functions in the main menu can be accessed by both the users. If any user wants to view book details, first the book menu will be displayed. Then user has to select the option called view and a response will be sent to the user by showing all the saved book details or an error message.

If any of the users want to add book details, first the book menu will be displayed. Then user has to select the option called add and enter particular book's details on relevant fields. After that, particular book detail will be saved in the file and a response will be sent to the user as successfully saved or not.

If user needs to search for a particular book, first the book menu will be displayed. Then user has to select the option called search. Then the system will search that particular record from the file and send search result to the user or else an error message.

Task 2 – Developed System

2.1) Login



This is the Login interface which is common for both the users in this City Bookshop's System. This is the very first interface which is visible for any user. There are three panels used along with labels, a text field, a password field, a combo box and buttons.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package city_bookshop;

import javax.swing.JOptionPane;
import java.io.*;
/**
 *
 * @author Chamath__Shyamal
 */
public class LoginIF extends javax.swing.JFrame {

      File LoginFile;

    public LoginIF() {

        try
        {
          LoginFile=new File("C:\\Users\\User\\Desktop\\City Bookshop Files\\LoginIF.txt");

          if(LoginFile.createNewFile())
          {
           System.out.println("File Created:"+LoginFile.getName());
          }
          else
          {
              System.out.println("File already exists !");
          }
        }

        catch (Exception e)

       {
           System.out.println("An error occured creating file !"+e);
        }

        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
```

When it comes to the Login interface that Login interface has named as LoginIF (Login InterFace). Thus, in the above screenshot, it proves that coding of the Login has done under the package called city_bookshop. After that, there can be seen some imported library functions such as javax.swing.JOptionPane and java.io.*. Then the public class called LoginIF can be seen. After that the constructor called public LoginIF() is visible where creation of the file called LoginFile is taken. Then the function happen inside the login button is depicted in the below screenshot.

```java
    private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {

if(txtUsername.getText().length()<=0 || txtPassword.getText().length()<0)
{
JOptionPane.showMessageDialog(null, "Fields can't be blank ! Fill & Try Agin ! ", "Login", JOptionPane.WARNING_MESSAGE);
}
else
  try {
      String UserName = txtUsername.getText();
      String Password = txtPassword.getText();
      String UserType = cmbUserType.getSelectedItem().toString();

      FileReader fr = new FileReader(LoginFile);
      BufferedReader br = new BufferedReader(fr);
      String line, UName, Passd, UType;
      boolean LoginSuccessfull = false;

      while ((line = br.readLine()) != null) {
      UName = line.split(" ")[0];
      Passd = line.split(" ")[1];
      UType = line.split(" ")[2];

    if(UName.equals(UserName) && Passd.equals(Password) && UType.equals(UserType)){
    LoginSuccessfull = true;
    JOptionPane.showMessageDialog(null, "Successfully Logged-In");

            if (UserType.equals("Manager")) {
            Managers_MenuIF mm= new Managers_MenuIF();
            mm.setVisible(true);
            this.setVisible(false);
            }
            else{
            Book_MenuIF mm = new Book_MenuIF ();
            mm.setVisible(true);
            this.setVisible(false);
           }
         }
       }
     if (!LoginSuccessfull)     {
      JOptionPane.showMessageDialog(null, "Incorrect Username , Password or User Type Entered", "Login", JOptionPane.WARNING_MESSAGE);
      }

      br.close();
      fr.close();
  }
      catch(Exception e)    {
      System.out.println("An error occured creating file"+e);
      }
    }
```

Inside this LoginIF interace there can be seen a private method called Login as a button which includes the function taken palace when user login to the system. In the above screenshot, there is an if…else statement at the beginning which means if any of the fields are empty (length is less than zero), a warning message will be displayed as "Fields can't be blank! Fill & Try Again!". So, it is clear that this is one of the validations in the code. If not, rest of the code will be executed. As in the above screenshot, rest of the coding under the login has been included inside a try-catch block.

Further, if user entered Username and Password are matched, then a message will pop up mentioning "Successfully Logged-In". And based on the user type selected by the user, he/she will be directed into the Manager Menu or to the Book Menu (Main Menu). For that function also, if…else statement has been used and class name called Manager_MenuIF and

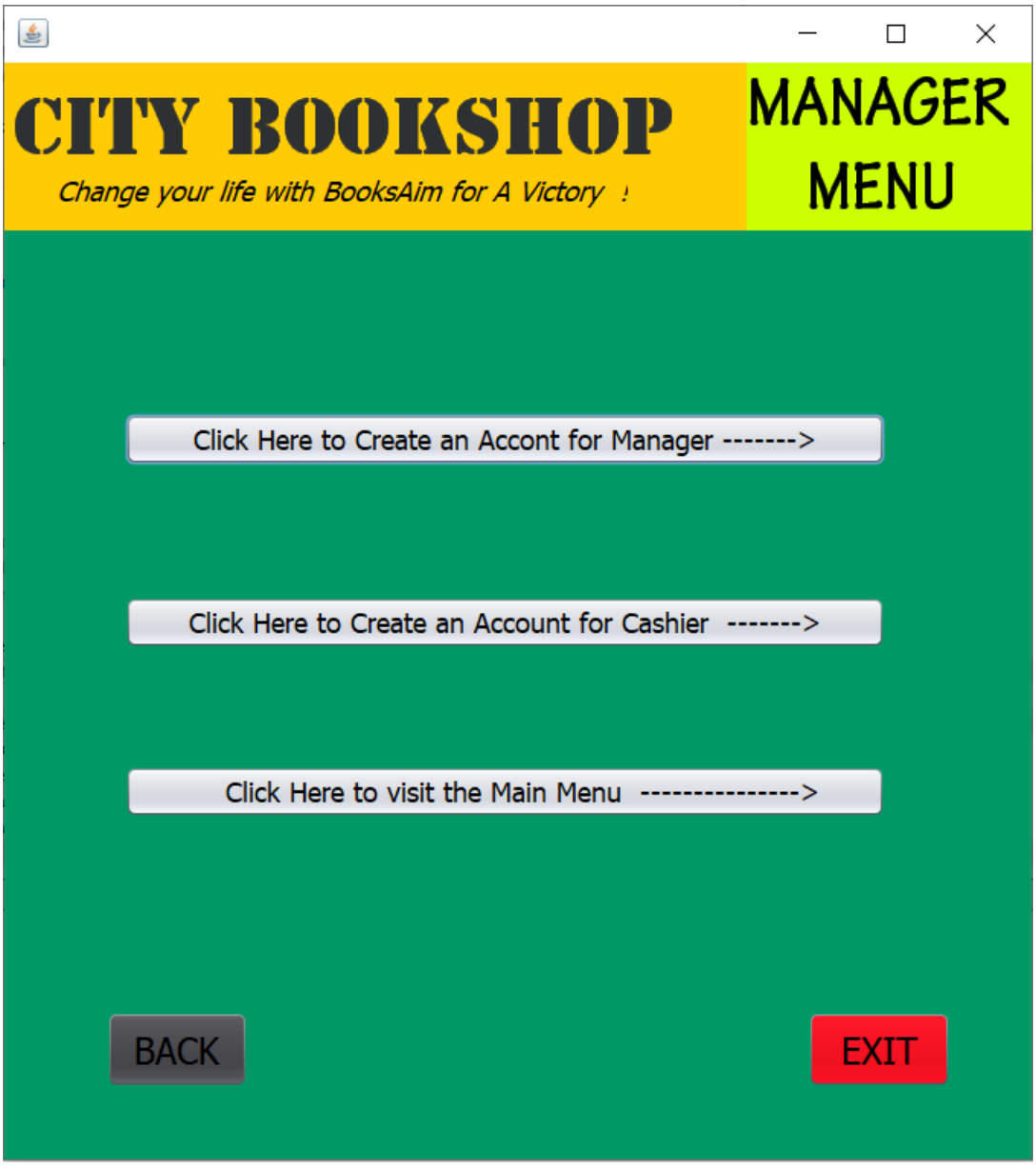
Book_MenuIF with the object called mm has used too. There are two methods as mm.setVisible(true) and this.setVisible(false). mm.setVisible(true) will show the Manager Menu while this.setVisible(false) close the previous window.

However, if user entered username, password and user type are incorrect, system will show a message as “Incorrect Username, Password or User Type Entered”. That’s also a validation in this system. Finally, the catch part has used for error handling. Moreover, as encapsulation leads for hiding the complexity of the implementation inside the class, inside the code there can be seen txtUsername, txtPassword and cmbUserType where encapsulation technique has used in order to hide user’s details inside the class.

```
315
316     private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
317             System.exit(0);
318     }
319
320     private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
321             txtUsername.setText("");
322             txtPassword.setText("");
323             cmbUserType.setSelectedIndex(0);
324     }
325
```

In the above screenshot, there can be seen two private methods as btnExit and btnClear assigned for the Exit and Clear buttons. So, if user wants to exit from the system all he/she has to do is press the exit button only. Then the system will be closed automatically. Also, if user wants to clear the stuffs entered in the fields, he/she can press the button called Clear.

2.2) Manager Menu



This is the interface which is only accessible for Manager. As in the first interface, three panels have used along with buttons and labels. When manager provide correct username and password and user type as manager, he/she will be directed into this interface which has named as Manager Menu.

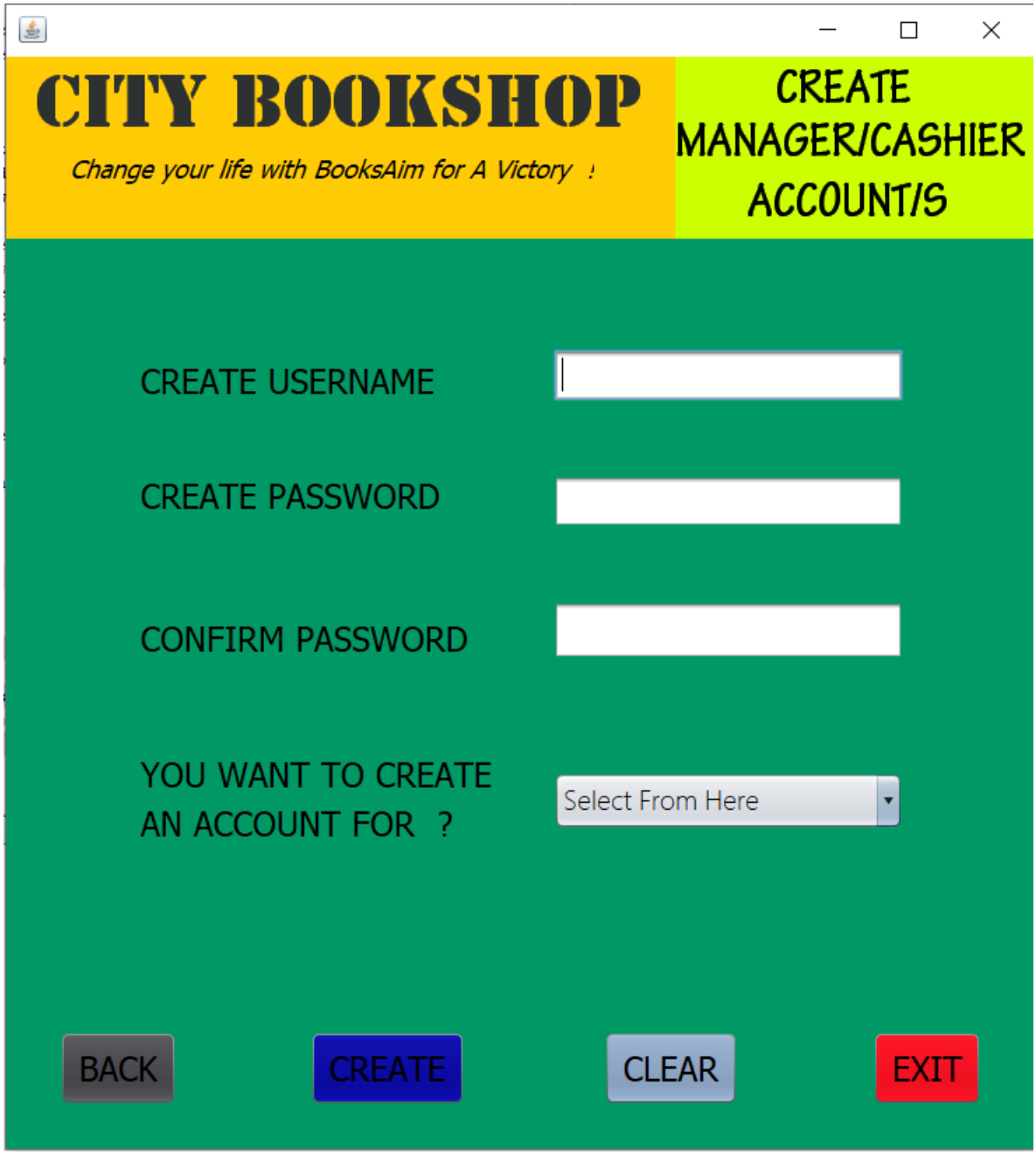
```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package city_bookshop;

/**
 *
 * @author Chamath__Shyamal
 */
public class Managers_MenuIF extends javax.swing.JFrame {

    /**
     * Creates new form Managers_MenuIF
     */
    public Managers_MenuIF() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```

It is visible in the above screenshot that the class called Managers_MenuIF is there. Moreover, the constructor of that class also can be seen as Managers_MenuIF(). Below screenshot will highlight how each button in this interface works.

```java

    private void btnCAmanagerActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        Create_New_MC_AcIF cma = new Create_New_MC_AcIF ();
        cma.setTitle ( "Create Manager Account");
        cma.setVisible(true);
    }

    private void btnCacashierActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        Create_New_MC_AcIF cca = new Create_New_MC_AcIF ();
        cca.setTitle ( "Create Cashier Account");
        cca.setVisible(true);
    }

    private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        LoginIF bk= new LoginIF();
        bk.setTitle ( "Login");
        bk.setVisible(true);
    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void btnMainMenuActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        Book_MenuIF mm = new Book_MenuIF ();
        mm.setTitle ( "View Main Menu");
        mm.setVisible(true);
    }

```

In here, both btnCAmanager and btnCacashier will be loaded into one interface called Create_New_MC_AcIF where manager is able to create new accounts for cashier and manager. The method named as dispose () will close all the previous windows. There, two objects have created as cma and cca for Create_New_MC_AcIF classes. Because those two buttons are loaded into the same interface. Otherwise, if user click ok btnMainMenu, the interface with the Book Menu (main menu) will be displayed. There, an object has created as mm for the Book_MenuIF class. Further, its visible that a back button also added. The object created for LoginIF in btnBack is bk. When user click in Back button, user can see the login interface again. Those private methods are private access modifiers in here. Same as mentioned in the login, method called btnExit is used in here too. So, when user click on Exit button, system will be closed.

2.3) Create Manager/Cashier Account/s



Basically, this is the interface which manager can see when manager click on any of the buttons named as Click Here to Create an Account for Manager and Click Here to Create an Account for Cashier. I have provided two buttons for the same interface because it's easier for manager to click on any of those two buttons and create an account. Anyhow, for this interface

also three panels have used along with labels, a text field, password fields, a combo box and buttons.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package city_bookshop;

import java.io.*;
import javax.swing.JOptionPane;
/**
 *
 * @author Chamath__Shyamal
 */
public class Create_New_MC_AcIF extends javax.swing.JFrame {

    File LoginFile;

    public Create_New_MC_AcIF() {

        try
        {
          LoginFile=new File("C:\\Users\\User\\Desktop\\City Bookshop Files\\LoginIF.txt");

          if(LoginFile.createNewFile())
          {
           System.out.println("File Created:"+LoginFile.getName());
          }
          else
          {
              System.out.println("File already exists !");
          }
        }

        catch (Exception e)

       {
           System.out.println("An error occured creating file !"+e);
        }


        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```

Create_New_MC_AcIF interface also falls under the package called city_bookshop. In the above screenshot, it shows the class name for this as Create_New_MC_AcIF and constructor as Create_New_MC_AcIF(). Then can see that same file created part done in the login has coded here also inside the constructor. Reason for that is newly created account details are also saved inside the same file called LoginFile.

```
Start Page | City_Bookshop.java | LoginIF.java | Managers_MenuIF.java | Create_New_MC_AcIF.java | Book_MenuIF.java | SearchAddIF.java
Source  Design  History
329     private void btnCreateActionPerformed(java.awt.event.ActionEvent evt) {
330
331         boolean valid= true;
            if ( txtUsername.getText().toString().isEmpty() || txtPassword. getText().toString().isEmpty()|| txtConfirmPassword.getText().toString().isEmpty()||
333             cmbUserType.getSelectedItem().toString().isEmpty() )
334           {
335            JOptionPane.showMessageDialog(null,"Fields cannot be blank !","Create Manager/Cashier Account",JOptionPane.ERROR_MESSAGE );
336            valid = false;
337           }
338
339         else
340           {
            if ( txtPassword.getText().toString().length()<8)
342            {
343               JOptionPane.showMessageDialog(null," Password should contain eight(8) characters ","Create Manager/Cashier Account", JOptionPane.ERROR_MESSAGE);
344               valid = false;
345            }
346            else if (!txtPassword.getText().equals (txtConfirmPassword.getText()))
347            {
348            JOptionPane.showMessageDialog(null,"ERROR! Password and the Confirm Password should match !", "Create Manager/Cashier Account" ,JOptionPane.ERROR_MESSAGE);
349            valid=false;
350            }
351           }
352
353         if(valid) {
354         String Username = txtUsername.getText();
355         String Password = txtPassword.getText();
356         String UserType = cmbUserType.getSelectedItem().toString();
357         String Record;
358         Record=Username+" "+Password+" "+UserType;
359
360         try  {
             FileWriter writer=new FileWriter(LoginFile,true);
362          BufferedWriter br=new BufferedWriter(writer);
363          writer.write (System.getProperty("line.separator"));
364
365          br.write(Record);
366         // br.newLine();
367          br.close();
368          writer.close();
369
370          JOptionPane.showMessageDialog(null,"Account Created Successfully");
371         }
372         catch(Exception e)
373          {
374          JOptionPane.showMessageDialog(null,"An Error Occured when creating the Account ! TRY AGAIN !");
375          }
376     }
377 }
```

All coding depicts by the above screenshot comes under the private method called btnCreate. As shown in the screenshot, if…elseif…else statements have used. First if statement have been used in order to check whether manager tries to press create button with empty fields. If so, the system will show an error message as "Fields cannot be blank!". That is also a validation in this system. The second if statement will execute if user entered a password less than eight characters. So, when user entered a password less than eight characters "Password should contain eight (8) characters". Then can found an else if statement. That statement will run when user entered password and confirm password are not same. If password and confirm password does not match the system will throw an error message as "ERROR! Password and the Confirm Password should match!". That is also a validation part used.

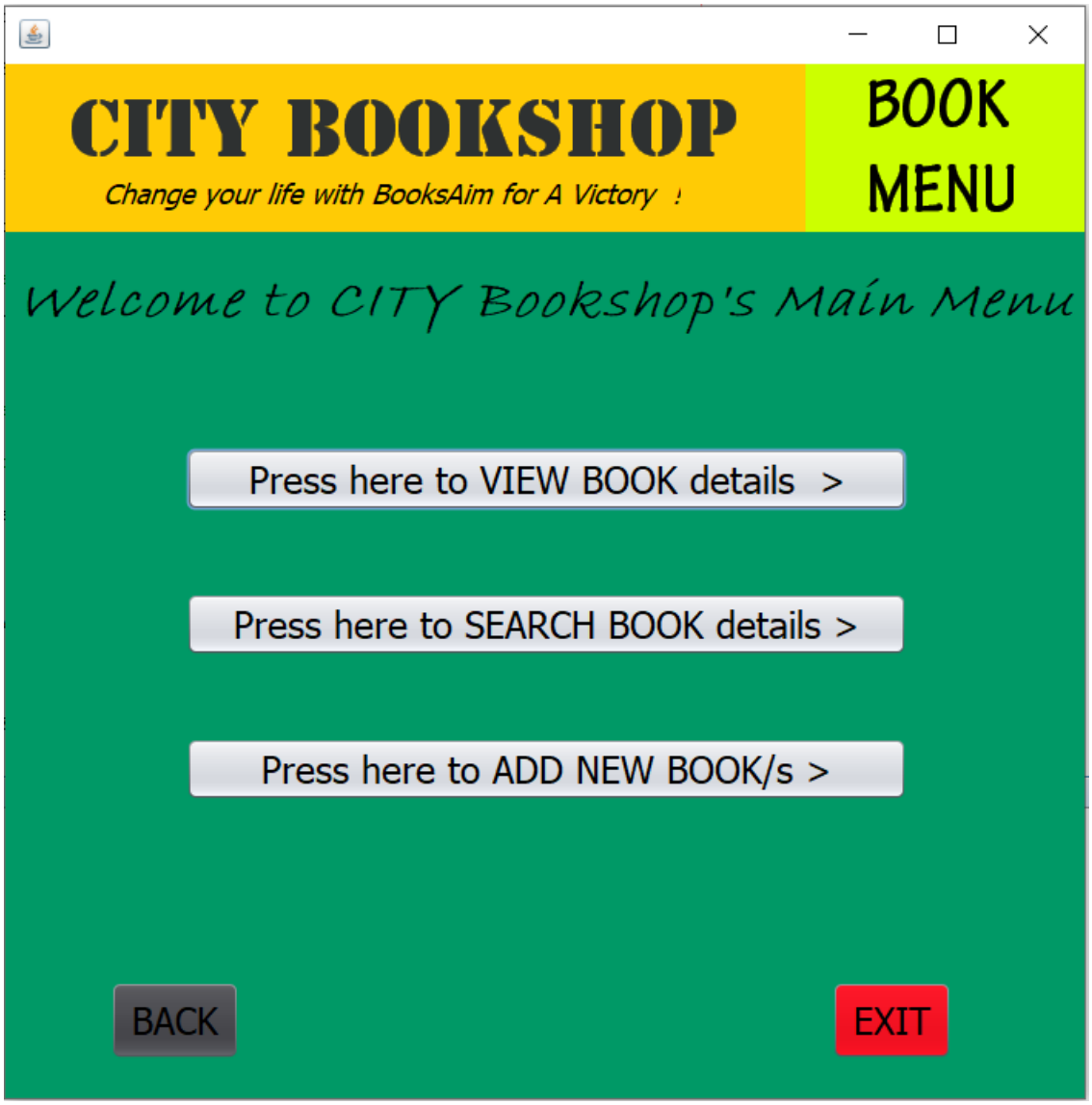
However, user entered everything are valid or fine with the conditions, then those user details will be saved into the LoginFile. And show a message as "Account Created Successfully". Otherwise, an error message will be displayed as "An Error Occurred when creating the

Account! TRY AGAIN!". This part also a validation used in this system and used inside a try-catch block.

```
310
311     private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
312         dispose();
313         Managers_MenuIF bk= new Managers_MenuIF();
314         bk.setTitle ( "Managers Menu");
315         bk.setVisible(true);
316     }
317
318     private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
319         txtUsername.setText("");
320         txtPassword.setText("");
321         txtConfirmPassword.setText("");
322         cmbUserType.setSelectedIndex(0);
323     }
324
325     private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
326         System.exit(0);
327     }
328
```

In the above screenshot, three private methods names as btnBack, btnClear and btnExit are visible. If manager clicks on Back button where I have created the object as bk and directed for the class called Managers_MenuIF, manager will be directed into the Manager Menu interface. If user click Clear button, all the fields in the Create_New_MC_AcIF form will be vanished. As usual, if user click on Exit button, the system will be closed.

2.4) Book Menu (Main Menu)



When it comes to this Book Menu interface, this interface is only accessible after login to the system by providing correct username, password and user type. This interface will be directly displayed to the cashier after login. But for manager, he/she can view this interface only after moving to the main menu. Anyhow, this interface contains three panels along with labels and buttons.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package city_bookshop;

import java.io.File;

/**
 *
 * @author Chamath__Shyamal
 */
public class Book_MenuIF extends javax.swing.JFrame {

    File BookMenu;
    public Book_MenuIF() {

        try
        {
          BookMenu=new File("C:\\Users\\User\\Desktop\\City Bookshop Files\\Book_MenuIF.txt");

          if(BookMenu.createNewFile())
          {
           System.out.println("File Created:"+BookMenu.getName());
          }
          else
          {
              System.out.println("File already exists !");
          }
        }

        catch (Exception e)

       {
           System.out.println("An error occured creating file !"+e);
        }



        initComponents();

    }
```

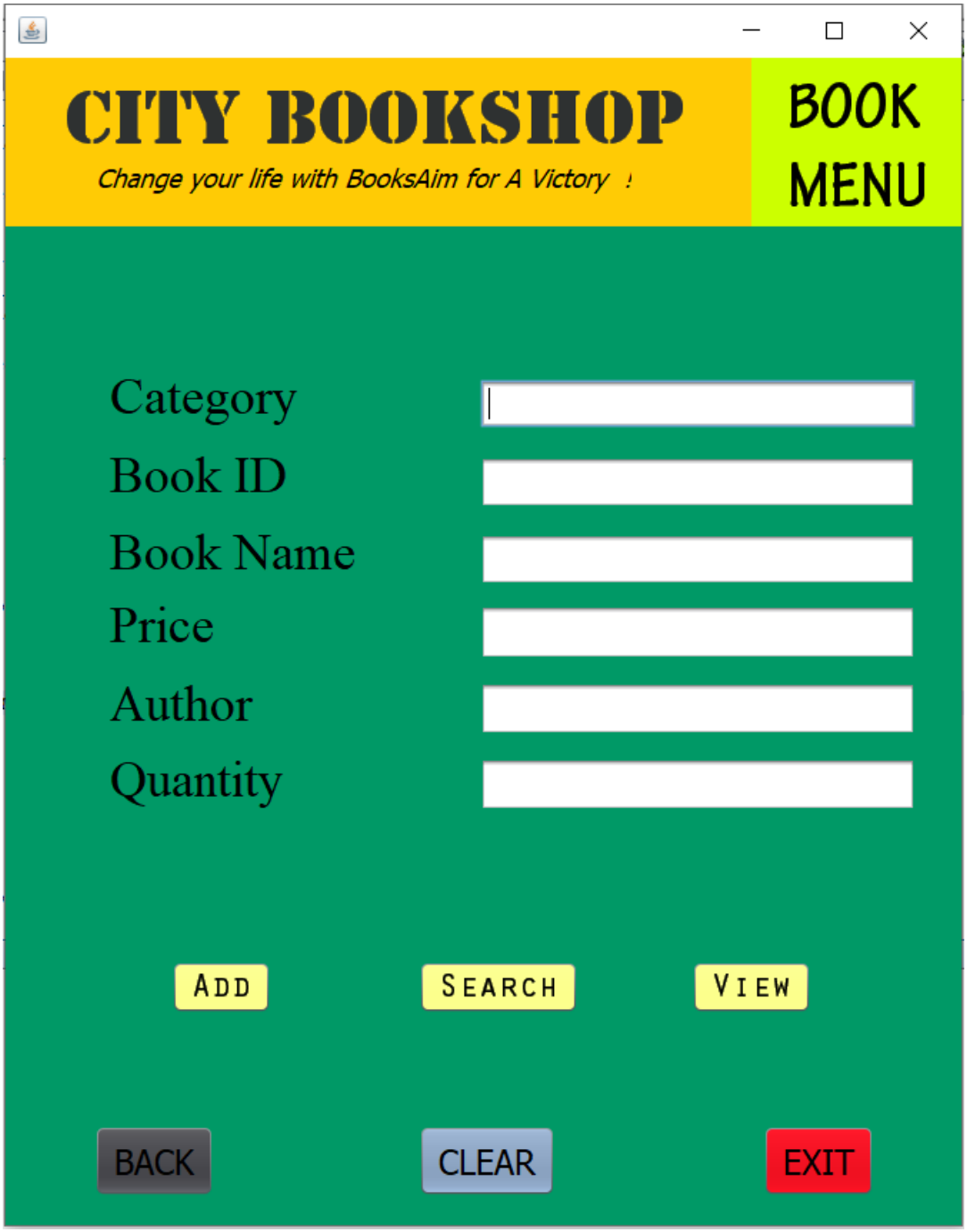
Book_MenuIF interface also falls under the package called city_bookshop. In the above screenshot, it shows the class name for this as Book_MenuIF and constructor as Book_MenuIF(). Then inside the constructor, new txt file has created as Book_MenuIF.txt to save the book details. That section has included inside a try-catch block.

```java

    private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        SearchAddIF sd = new SearchAddIF();
        sd.setTitle ("Search");
        sd.setVisible(true);
    }

    private void btnViewActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        SearchAddIF sd = new SearchAddIF();
        sd.setTitle ("View");
        sd.setVisible(true);
    }

    private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        SearchAddIF sd = new SearchAddIF();
        sd.setTitle ("Add");
        sd.setVisible(true);
    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
          System.exit(0);
    }

    private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        LoginIF bk= new LoginIF();
        bk.setTitle ( "Login");
        bk.setVisible(true);
    }

```

In the above screenshot, private access modifiers can be seen. Inside all btnSearch, btnView and btnAdd methods, there is a method called dispose () which close the previous interface window in order for new one to be appeared. Same object has created as sd and called the class called SearchAddIF under all first three buttons. All those three buttons will be loaded into the same interface where user can add, search and view book details and that interface is called SearchAddIF. Reason for adding three separate buttons is to make a clear understanding on user's mind. It's like user will identify that when he/she click on a particular function definitely he/she is on the right interface.

Additionally, there are two more methods as btnExit and btnBack. As usual, when user click on Exit button in the interface, the system will close and when user click on Back button, he/she will be able to see the Logic interface which is the very first interface in this system. For that, inside the btnBack private method, an object has created as bk for the LoginIF class and the Login interface will pop up.

2.5) Book Menu (View, Add & Search)



For the above interface, three panels have used along with labels, text fields and buttons. This interface is also accessible for both the users and can perform the functions in it.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package city_bookshop;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;

/**
 *
 * @author Chamath__Shyamal
 */
public class SearchAddIF extends javax.swing.JFrame {

    File BookMenu;

    public SearchAddIF() {

        try
        {
          BookMenu=new File("C:\\Users\\User\\Desktop\\City Bookshop Files\\Book_MenuIF.txt");

          if(BookMenu.createNewFile())
          {
           System.out.println("File Created:"+BookMenu.getName());
          }
          else
          {
              System.out.println("File already exists !");
          }
        }

        catch (Exception e)

       {
           System.out.println("An error occured creating file !"+e);
        }

        initComponents();
    }
```

In the above code there can be seen some imported java files. SearchAddIF interface also falls under the package called city_bookshop. In the above screenshot, it shows the class name for this as SearchAddIF and constructor as SearchAddIF (). Then can see that same file created part done in the Book_Menu has coded here also inside the constructor. Reason for that is when user search, add and try to view book details, that file needs to be updated and retrieve saved data from that file. That section has included inside a try-catch block.

```java
    private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {

        try {
          boolean Resultfound=true;
          String[] words=null;

          FileReader fr=new FileReader(BookMenu);
          BufferedReader br=new BufferedReader(fr);

          String FindLine;
          String Category=txtCategory.getText();
          String BookID=txtBookID.getText();
          String BookName=txtBookName.getter();
          String Price=txtPrice.getText();
          String Author=txtAuthor.getText();
          String Quantity=txtQuantity.getText();


          GoOutFromLoop:
          while((FindLine=br.readLine())!=null){
              words=FindLine.split(" ");

               for (String word:words){


                  if (word.equals(Category)){
                      Resultfound=true;
                      System.out.println(FindLine);

                      break GoOutFromLoop;
                  }

                  else if (word.equals(BookID)){
                      Resultfound=true;
                      System.out.println(FindLine);

                      break GoOutFromLoop;
                  }

                  else if (word.equals(BookName)){
                     Resultfound=true;
                     System.out.println(FindLine);

                      break GoOutFromLoop;
                  }

                  else if (word.equals(Price)){
                     Resultfound=true;
                     System.out.println(FindLine);
```

As visible in the above screenshot inside the btnSearch private method, try-catch block is used and inside that FileReader and BufferedReader objects have created. A while…loop also have used in order to find the line including if…elseif…else statement. If the system found the user searched line from the text file, it will come out from the loop and the next screenshot will display what happens after system found the correct searched result.

```
                    break GoOutFromLoop;
                }

                else if (word.equals(Price)){
                    Resultfound=true;
                    System.out.println(FindLine);

                    break GoOutFromLoop;
                }

                else if (word.equals(Author)){
                    Resultfound=true;
                    System.out.println(FindLine);

                    break GoOutFromLoop;
                }

                else if (word.equals(Quantity)){
                    Resultfound=true;
                    System.out.println(FindLine);

                    break GoOutFromLoop;
                }

                else{
                    Resultfound=false;
                }
            }
        }
    if(!Resultfound)
    {
        JOptionPane.showMessageDialog(rootPane, "Searched Result is NOT FOUND");

    }
    else{
        JOptionPane.showMessageDialog(rootPane,"Searched Result is FOUND"+"\n"+FindLine);
    }

    fr.close();
    br.close();

     }
     catch (Exception e) {
        JOptionPane.showMessageDialog(null,"An error occured Searching"+ e,"Search",JOptionPane.ERROR_MESSAGE);
     }

}
```

As mentioned above, if system found the user searched record from the file, it will display a message as "Searched Result is FOUND" along with the found result. If not, the system will display a message to the user as "Searched Result is NOT FOUND" as those validation parts are visible in the above screenshot. Further, the message inside the catch block will execute and display for the user as an error message.

```
419     private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
420
421  if(txtCategory.getText().length()<=0 || txtBookID.getText().length()<0 || txtBookName.getText().length()<=0 || txtPrice.getText().length()<=0 || txtAuthor.getText().length()<=0 ||
422                 txtQuantity.getText().length()<=0 )
423       {
424        JOptionPane.showMessageDialog(null, "Fields can't be blank ! Fill & Try Again ! ", "Add", JOptionPane.WARNING_MESSAGE);
425       }
426  else{
427
428          String Category=txtCategory.getText();
429          String BookID=txtBookID.getText();
430          String BookName=txtBookName.getText();
431          String Price=txtPrice.getText();
432          String Author=txtAuthor.getText();
433          String Quantity=txtQuantity.getText();
434
435
436          String Record=Category+" "+BookID+" "+BookName+" "+Price+" "+Author+" "+Quantity;
437
438          try
439          {
441              FileWriter writer=new FileWriter(BookMenu,true);
442              BufferedWriter buffer=new BufferedWriter(writer);
443
444              buffer.write(Record);
445              buffer.newLine();
446              buffer.close();
447              writer.close();
448
449              JOptionPane.showMessageDialog(null,"Successfully ADDED to the file");
450          }
451          catch(IOException e){
452              System.out.println("An ERROR occured Adding values"+e);
453          }
454       }
455     }
```

In the above screenshot, private method called btnAdd method have used. In here, the system will get user inputs and add them into the text file as a record. Basically, the writing to the file has taken place in here using try-catch block. If user filled all the fields and press add button, the system will display a message as "Successfully ADDED to the file". Otherwise, system will display an error message as "An ERROR occured Adding values". However, if user tries to press add button without filling any of the fields or with any empty field, system will throw a warning message as "Fields can't be blank! Fill & Try Again! " to the user. So, those are the validations used when inside the btnAdd method.

```java
    private void btnViewActionPerformed(java.awt.event.ActionEvent evt) {

        String line,Lines=" ";

        try {
            FileReader fr=new FileReader(BookMenu);
            BufferedReader br=new BufferedReader(fr);

            while((line=br.readLine())!=null)
            {
                System.out.println(line);
                Lines=Lines+line+"\n";
            }
            JOptionPane.showMessageDialog(rootPane, Lines);
        }
        catch (Exception e) {
            System.out.println("ERROR in Viewing data ! TRY AGAIN !"+e);
        }

    }
```

In here, the method called btnView is visible. What happens inside this view method is it will read all the lines in the BookMenu text file. For that also try-catch block have used and if user press this View button, user will be able to view all the saved records inside the file. Otherwise, it will display an error message to the user like "ERROR in Viewing data! TRY AGAIN!".

```java
    private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
        txtCategory.setText("");
        txtBookID.setText("");
        txtBookName.setText("");
        txtPrice.setText("");
        txtAuthor.setText("");
        txtQuantity.setText("");
    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

```

Here, the private method called btnClear and btnExit are visible. Thus, if user click on the Clear button, all the stuffs in the fields will be disappeared. If user click on Exit button, the system will be closed.

```java
    private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        Book_MenuIF bk= new Book_MenuIF();
        bk.setTitle ( "Book Menu");
        bk.setVisible(true);
    }
```

In the above screenshot, it depicts that when user click on back button, user will be redirected into the Book_Menu interface. For that, an object called bk has created for the Book_MenuIF class.

Task 3 – User Manual

Summary how the system works

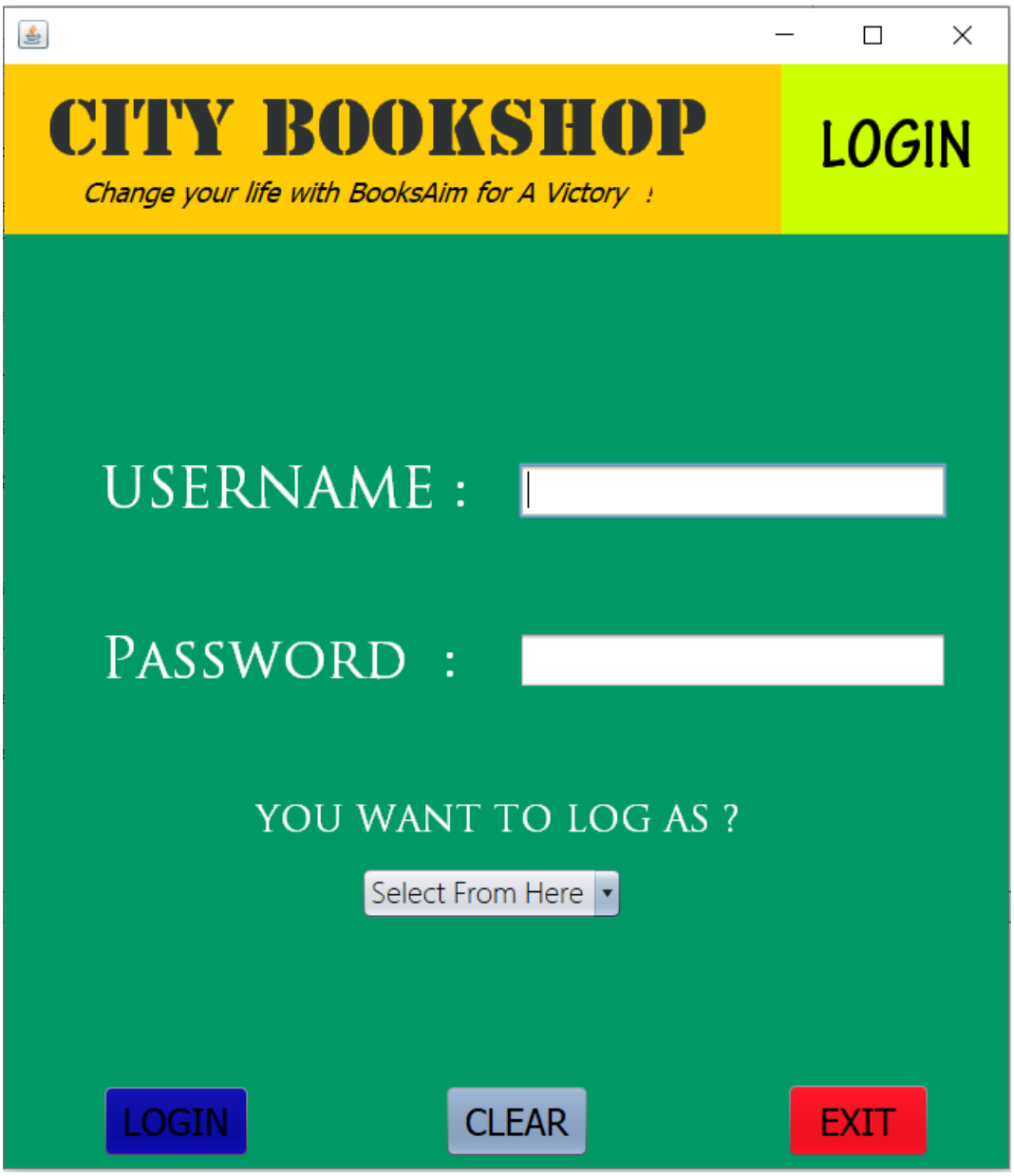
For Cashier

Displaying the Interface to Login → If login credentials are matched with Cashier, Bok Menu (Main Menu) Interface will be displayed → Then another interface will be loaded where user can add, view & search book details

For Manager

Displaying the Interface to Login → If login credentials are matched with Manager, Manager Menu Interface will be displayed → If manager select option to create new accounts for cashier/manager, Create New MC Ac/s interface will be loaded → If manager select option to view Book Menu (Main Menu), interface with add, view & search buttons will be loaded.

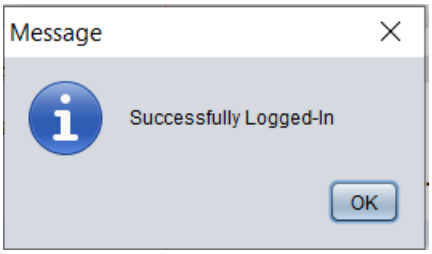
3.1) Login



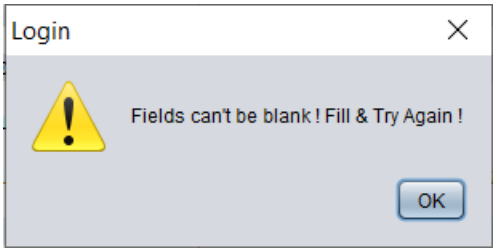
- This is the interface user can see at the beginning. This will be accessible for cashier only when manager logged and successfully created a cashier account.
- Manager can enter Username and Password written inside the file. Then have to select Manager as User Type. And press login button. If a mistake happened when typing username

or password, user can press clear button. If user wants to close this system, user can press on exit button. After press login button, following interface will be displayed.

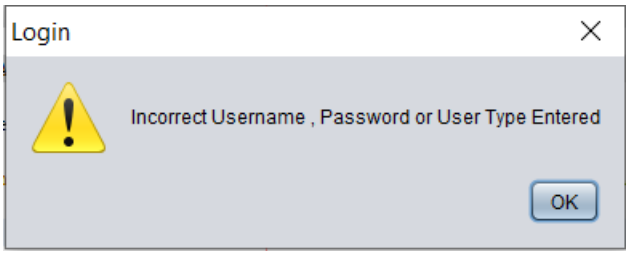
-If user successfully logged in, following notification will pop up.



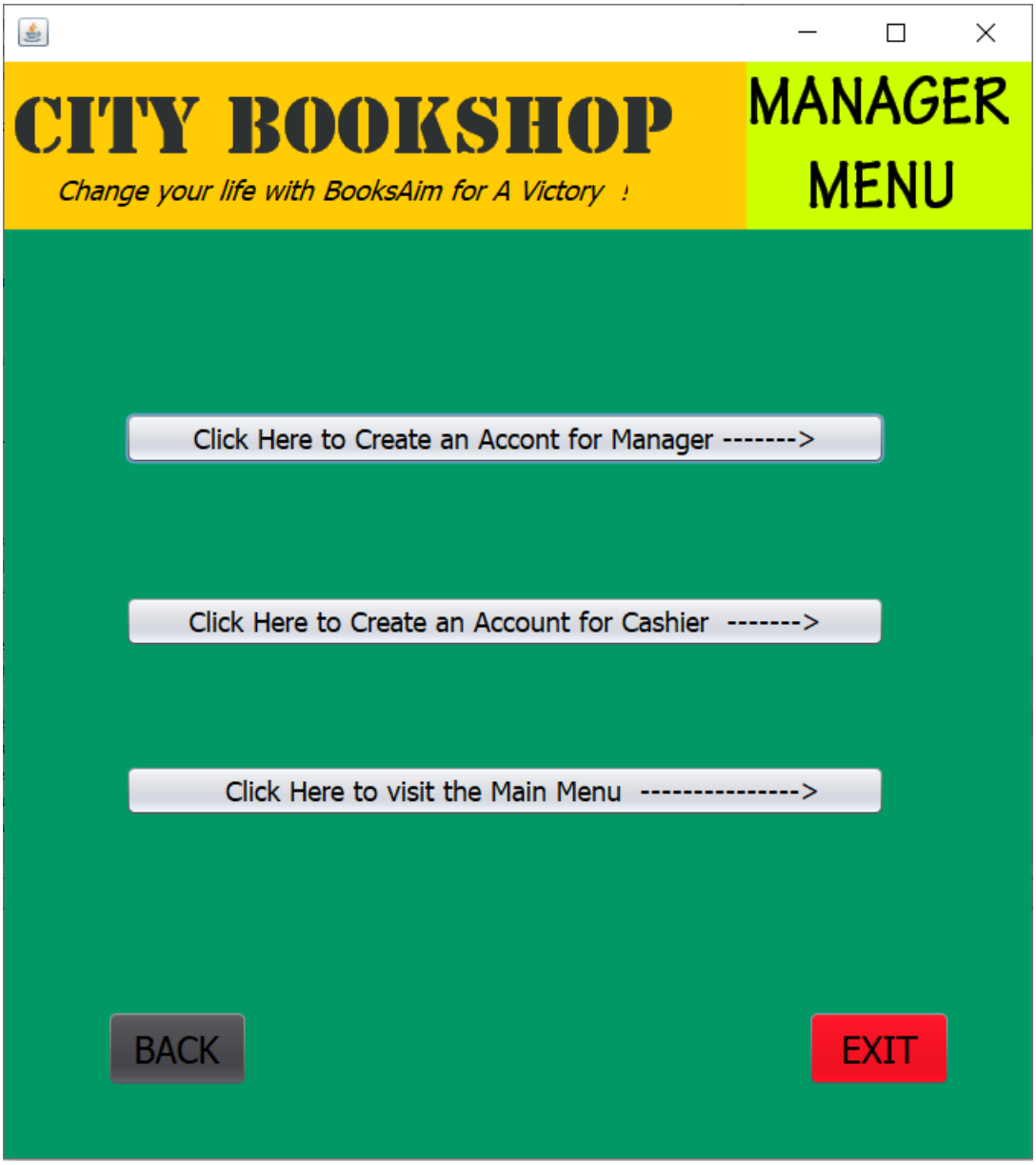
-If user try to click on login button with any empty fields, following notification will pop up.



-If user enters incorrect username, password or user type, following notification will pop up.



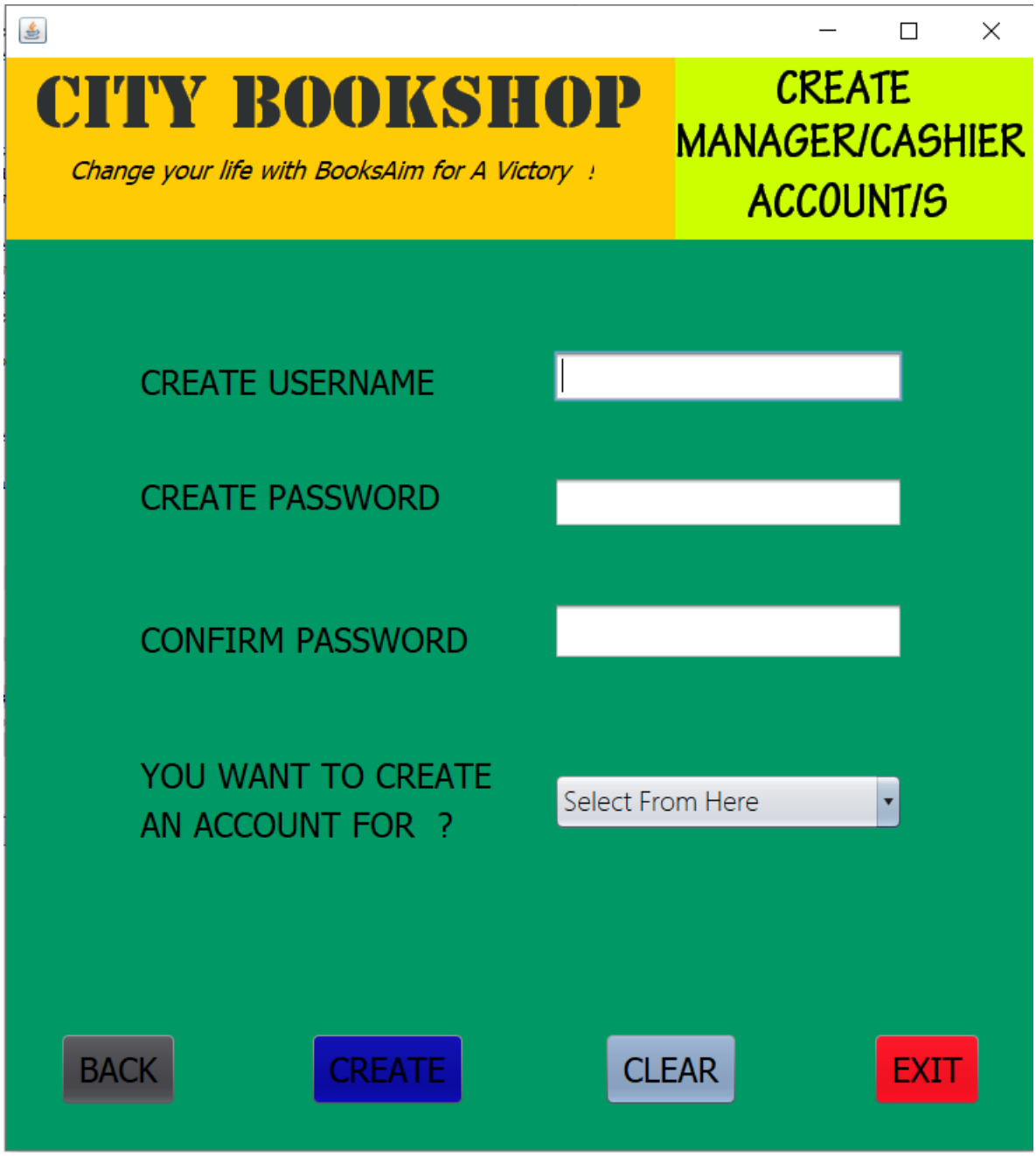
3.2) Manager Menu



- The above interface will be visible only for manager who log into the system by providing username and password assigned for manager. User who enters cashier username and password cannot view this interface.

- Manager can press one button from the first two buttons. Because those two buttons are connected with a same interface which has depicted below. From there, manager can create new manage or cashier accounts.
- If manager needs to view the Main Menu (Book Menu) interface where view, add and search functions there, user can press on Click Here to visit the Main Menu button.
- Otherwise, if manager wants to go back to the login interface, user can click on back button while if user wants to close the system, user can click on exit button.

3.3) Create Manager/Cashier Account/s



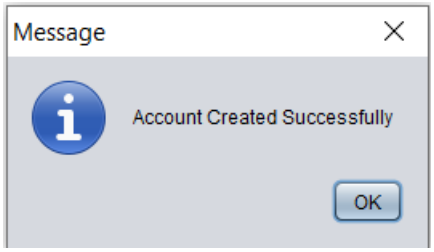
- When user click on of the first two buttons in the manager menu interface, user will be directed into the above interface.

- Manager can create a suitable any kind of a username and should create a password which contains eight characters. Further, the same password enters inside the Password filed should enter in the Confirm Password field as well. Then manager has to select the user type whether the new account is creating for Cashier or Manager. And press on create button.

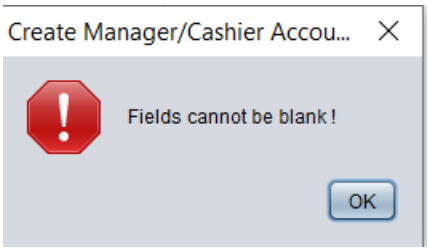
- If a mistake happened when typing username, password and confirm password, user can press clear button and enter again.

- If manager wants to go back to the manager menu, user needs to press back button and if user wants to close the system, user can press exit button.

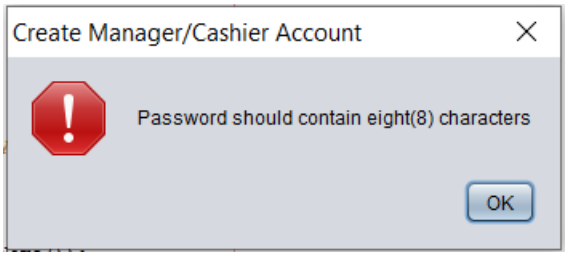
-If User-Manager enter create button after filling all the fields successfully, following notification will pop up.



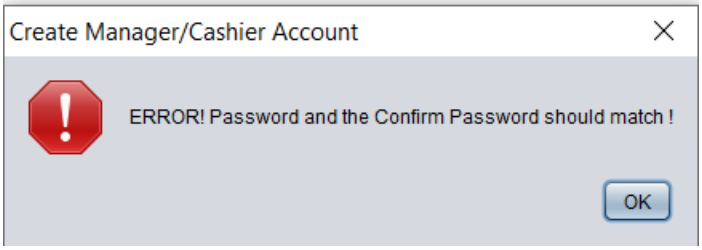
-If User-Manager tries to press on create button with empty fields, following notification will pop up.



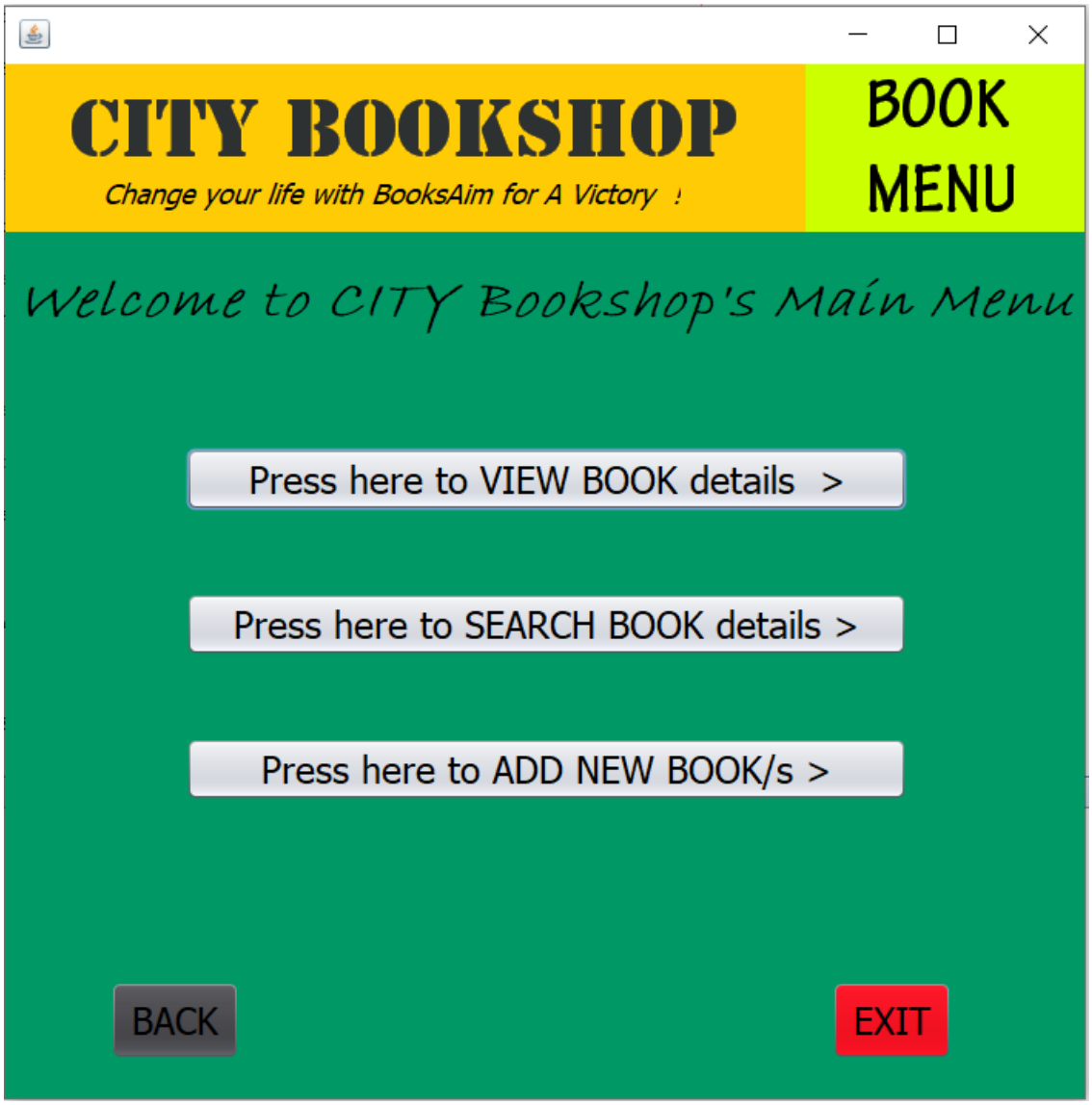
-If User-Manager tries to press create button with password less than or more than 8 characters, following notification will pop up.



- If User-Manager tries to press create button without matching characters inside the password and confirm password fields, following notification will pop up.

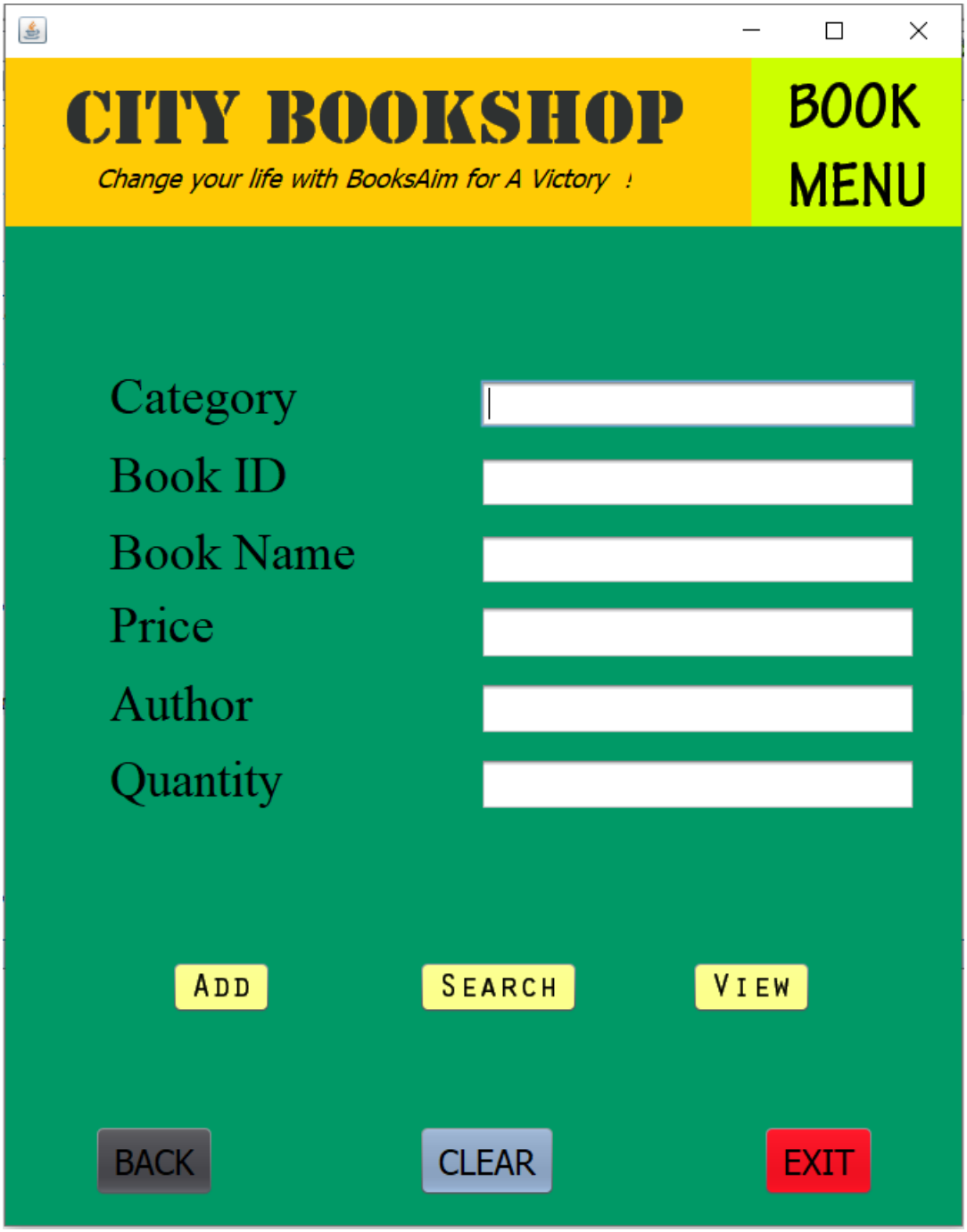


3.4) Book Menu (Main Menu)



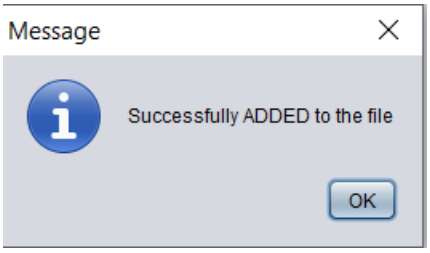
- According to the username and password created by Manager, cashier/s can provide relevant username, password and they will be directly loaded into the above interface which is called Book Menu. This is the common interface shown to all the users who log into the system.
- All three main buttons inside the interface will be loaded into a different interface where user can add, search and view book details.
- If user wants to go back to the login interface, user can click on back button and if user wants to close the system, user can press exit button.

3.5) Book Menu (View, Add & Search)

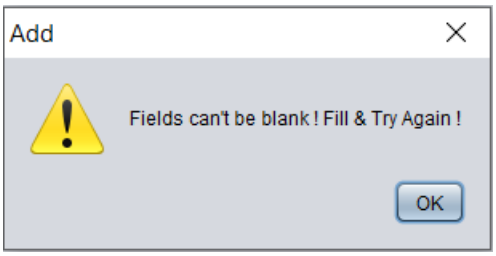


- Once user clicks on one button out of three buttons in the Main Menu (Book Menu), the above interface will be loaded.
- As this interface is commonly accessible for any user who logged in, user/s can view all the saved book details by clicking on the button called view.
- If user wants to add a new book detail, user/s can fill all the relevant fields and press on button called add.
- If user wants to search an added record from the file, user can fill any of the fields with a keyword and press search button. Assume that user wants to find a book from its name. So, user can enter that particular name inside the field called book name and click on search button.
- If a mistake happened when typing username, password and confirm password, user can press clear button and enter again.
- If user wants to go back to the login interface, user can click on back button and if user wants to close the system, user can press exit button.

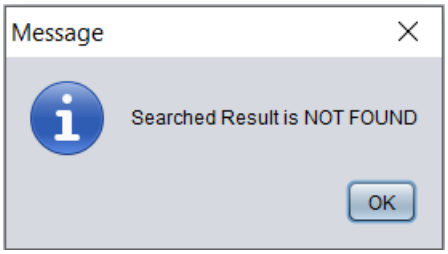
-If user completed all the fields and press on add button, following notification will pop up.



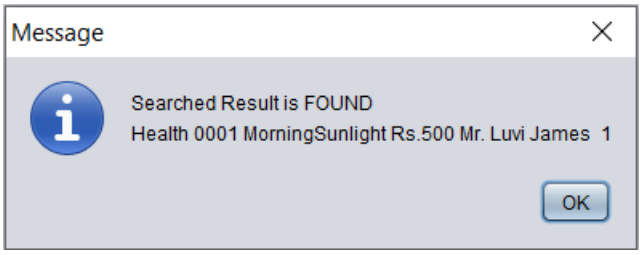
-If user tries to press on add button with empty fields, following notification will pop up.



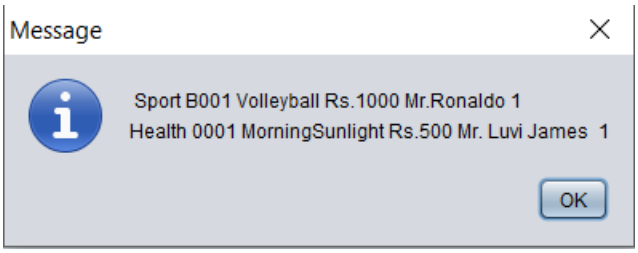
-If user clicks on search button with empty fields or with a wrong searching keyword in a text field, following notification will pop up.



-If user clicks on search button by entering a correct keyword, a notification will pop up like in below.



-If user clicks on view button, following notification will pop u.



Conclusion

However, I could get many advantages from this course work such as how to solve doubts and issues, how to gather requirements needed, how to do the analyzation properly, how to manage time effectively, how to overcome from errors which are arising when writing the code and so on. Finally…with all above mentioned details and me proper time management, I could prepare a successful document to complete this assignment. Thus, I hope this assignment been a great help for me to get learnt about Java Programming and to prove that I have successfully completed my seventh assignment in my HD Program. Moreover, during this OOP-Java assessment, I learnt about the basics of object-oriented programming while learning Java programming language and many more things. Developing of this system for City Bookshop gave me the chance to try my new skills in practice. Specially this recalled my knowledge for drawing UML diagrams. So, I have built the application in a user-friendly manner including login part, manager menu, book menu, etc. While doing this project I also gained deep understanding on programming/coding and how it can be implemented in real life situations as now I have the experience in creating an application for City Bookshop. Thus, I believe that this was a great chance for me to improve my computer programming skills.