



UNIVERSITI TEKNOLOGI MALAYSIA

OBJECT ORIENTED PROGRAMMING

(SECJ2154)

SEMESTER 1, SESSION 2023/2024

GROUP PROJECT

BOOK STORE INVENTORY MANAGEMENT SYSTEM

SECTION 01

LECTURER:

MADAM LIZAWATI BINTI MI YUSUF

NAME	MATRIC NUMBER
SARVEISH A/L BALASUNDRAM	B23CS0016
TAN CHEE SEN	B23CS0072
CHONG WEI YANG	A23CS5027
LIEW YONG ZHENG	A23CS5008

Table Of Contents

SECTION A: PROJECT DESCRIPTION	3
1. Problem Statement and Project Overview	3
2. Objective and Scope	5
3. Project Workflow	6
4. OO Concepts	22
SECTION B: UML CLASS DIAGRAM	37
1. Name Class	54
2. User Class	55
3. Menu Class	57
4. Report Class	57
5. Admin Class	58
6. Customer Class	60
7. BookSupplier Class	62
8. Book Class	63
9. InventorySystem Class	66
10. OrderManagement Class	67
SECTION C: SOURCE CODE AND USER MANUAL	71
1. User manual	71
A. Admin User Manual	75
B. Customer User Manual	85
C. Book SupplierUser Manual	89
2. Source Code	93
SECTION D: TASK DISTRIBUTION.....	150
SECTION E: CONCLUSION.....	151

SECTION A: PROJECT DESCRIPTION

1. Problem Statement and Project Overview

Although online applications and platforms have been widely embraced for book purchases and monitoring product inventory levels, small-town or rural bookshops mostly depend on conventional, face-to-face interactions to sell and manage their stock. One such bookstore is Kamal Book Store.

Mr Kamal is facing a lot of difficulties in uplifting his bookstore sales and keeping track of his book store's stores' management. This is because Mr. Kamal still uses manual filing to keep track of book records, book stock quantity and many more. He felt very stressed due to the large amount of work that he needed to put in order to manage the book records, customers' details, and his bookstore as a whole. Moreover, he is also facing problems in keeping track of his restocking schedule for his products to the book supplier as he is encountering dilemmas including inventory discrepancies, out-of-stock, and challenges in accurately forecasting demand, resulting in delays or lack of accurate information from suppliers, supplying the wrong quantity and wrong type of books, indirectly impacting customer demand and satisfaction. So, he decided to implement an effective bookstore management system that would help him ease his work and improve his bookstore's sales.

Our team has planned to develop a system named Kamal BookStore Inventory Management System. This system is an efficient and comprehensive programme used to facilitate the day-to-day operations of the bookstore. This system is designed to be used by the admin (Mr. Kamal), the book supplier and the customers of his bookstore. The system mainly focuses on enhancing seamless book inventory management and secured authentication for the user types, the overall shopping experience for the customers and seamless management of restocking books by the supplier.

From an admin perspective, the demand for extra human resources to handle book records and stock quantity is significantly reduced for the administrator, Mr.Kamal. This is because the administrator is only required to concentrate on important activities like adding new books, changing information, and ensuring accurate inventory records by automating

time-consuming chores due to the system's effective features. For the bookstore and Mr Ali, this increase in efficiency means significant time and resource savings. Moreover, the system also offers a secured authentication, which ensures a secure sign-in process for the admin by limiting functionalities and features for other user roles, which are customers and suppliers. The system also provides Mr. Kamal with reports that give him important information about the functioning of the bookstore, including a list of the top ten best-selling books, total profit earned, and much more. As a result, it is possible to make good decisions on book promotion and inventory. In addition, Mr Kamal can effectively stock up by using the system for placing orders with suppliers so as to ensure a suitable level of inventory in order to sustain his bookshop's profitability.

From a customer perspective, the system will improve the whole user experience by offering an easy-to-use interface and a smooth book ordering procedure whenever they use the system. The system's user experience starts with a simple registration or sign-in process that gives users access to various features and an extensive range of book catalogues. Regarding purchasing a book from the bookstore, the system offers them an easy-to-understand book addition to their cart, and they can view their book cart before the ordering procedure is performed completely.

Lastly, from the book supplier's perspective, they can access the system by logging in to the system. They can view the pending admin orders towards them with in-detail information. Later on, they can provide approval status of either approve or reject to the admin's order request by even searching the order quickly and in an easy manner.

In conclusion, this system offers Mr Kamal's bookshop operations by providing customers with an easy-to-use platform to buy books from the store, an elegant book-ordering management process for both customers to the bookstore as well as admin to the book supplier, and an administrator-friendly approach to book management.

2. Objective and Scope

The main objectives of the system are:-

1. To offer a secure authentication to access the system by providing user registration and sign-in processes.
2. To provide an automated system with features for adding, editing, and removing book entries that help manage book inventory.
3. To increase efficiency and reduce data errors, resulting in improved accuracy, faster decision-making, and a more responsive approach to inventory control.
4. To offer suppliers to make faster decisions, respond promptly to admin's book supply requests, and participate in a more seamlessly integrated ordering process.
5. To offer the customers a seamless book ordering and purchasing experience.

Scope of the system are :-

1. The system only supports only one book supplier.
2. The system only supports one administrator to access the system, Mr. Kamal, as it doesn't allow new registration for admin roles.
3. The system doesn't offer a payment system for the customer to pay directly for their order through the system.
4. The system provides a simple yet effective reporting to the administrator.
5. The system offers no user types with sorting options to view the book catalogue.
6. The system is only recommended for desktops and laptops as it is not mobile-friendly.

3. Project Workflow

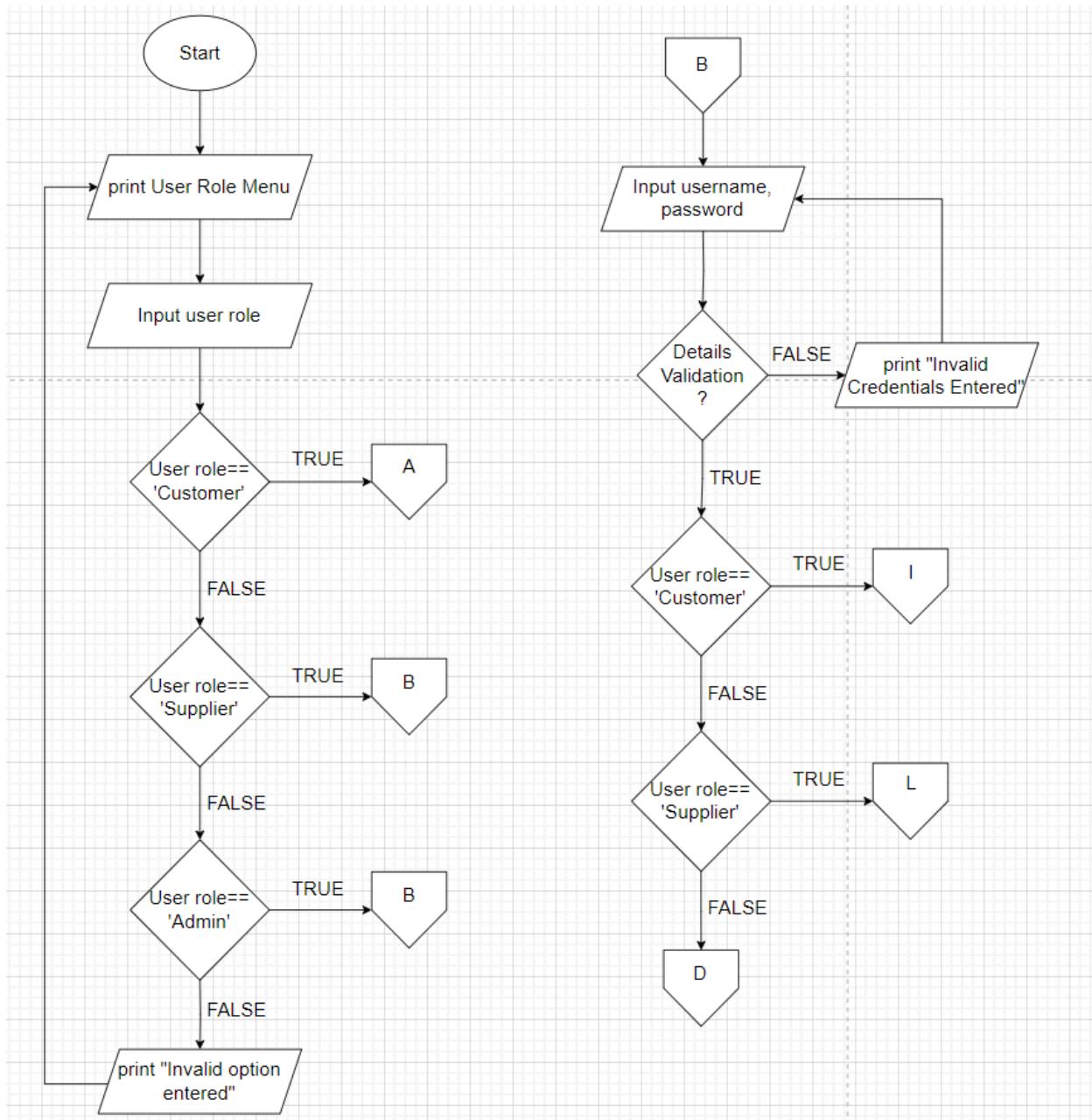
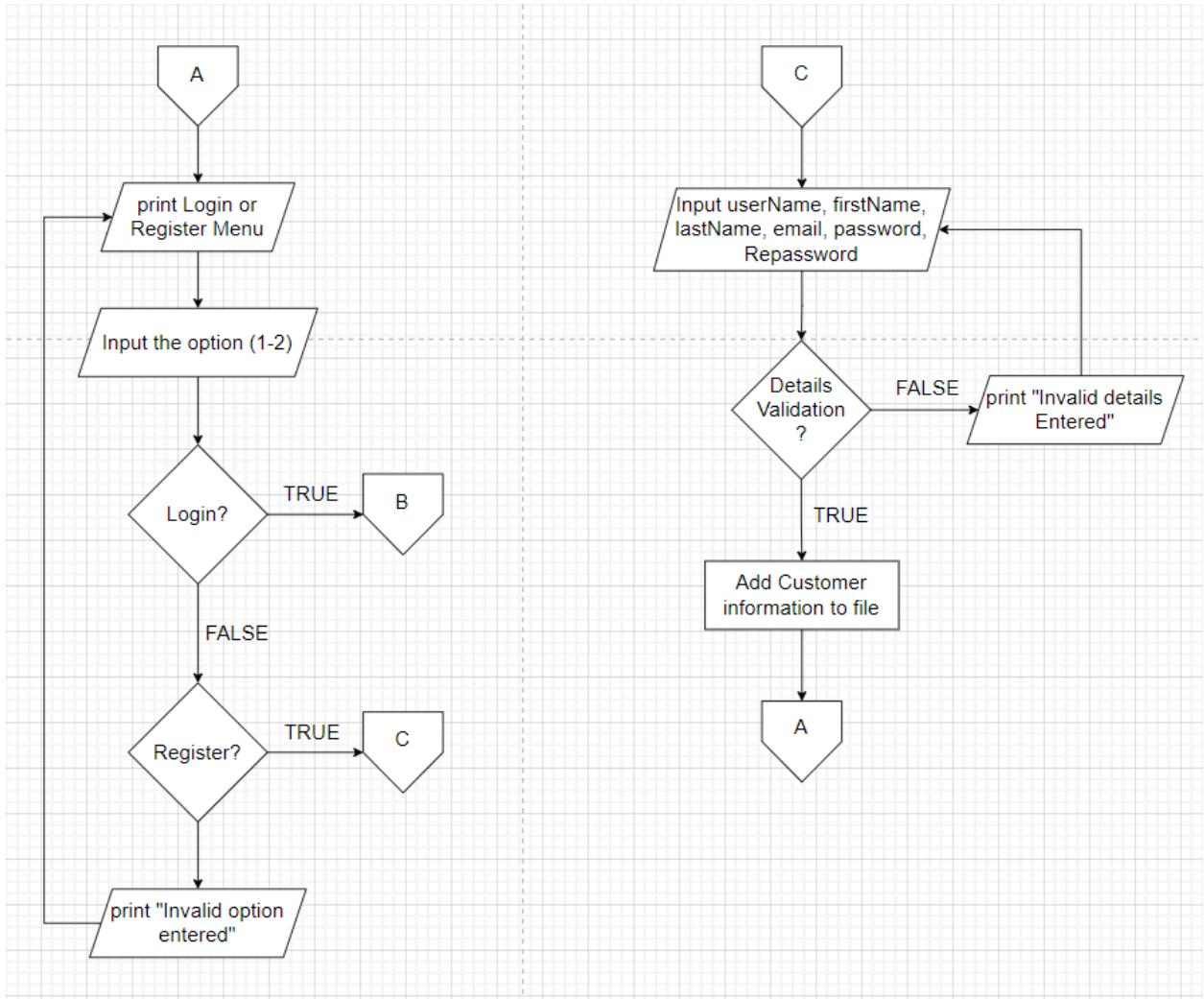


Figure 1.1.1: login() - select user role

Figure 1.1.2: login() - all user login



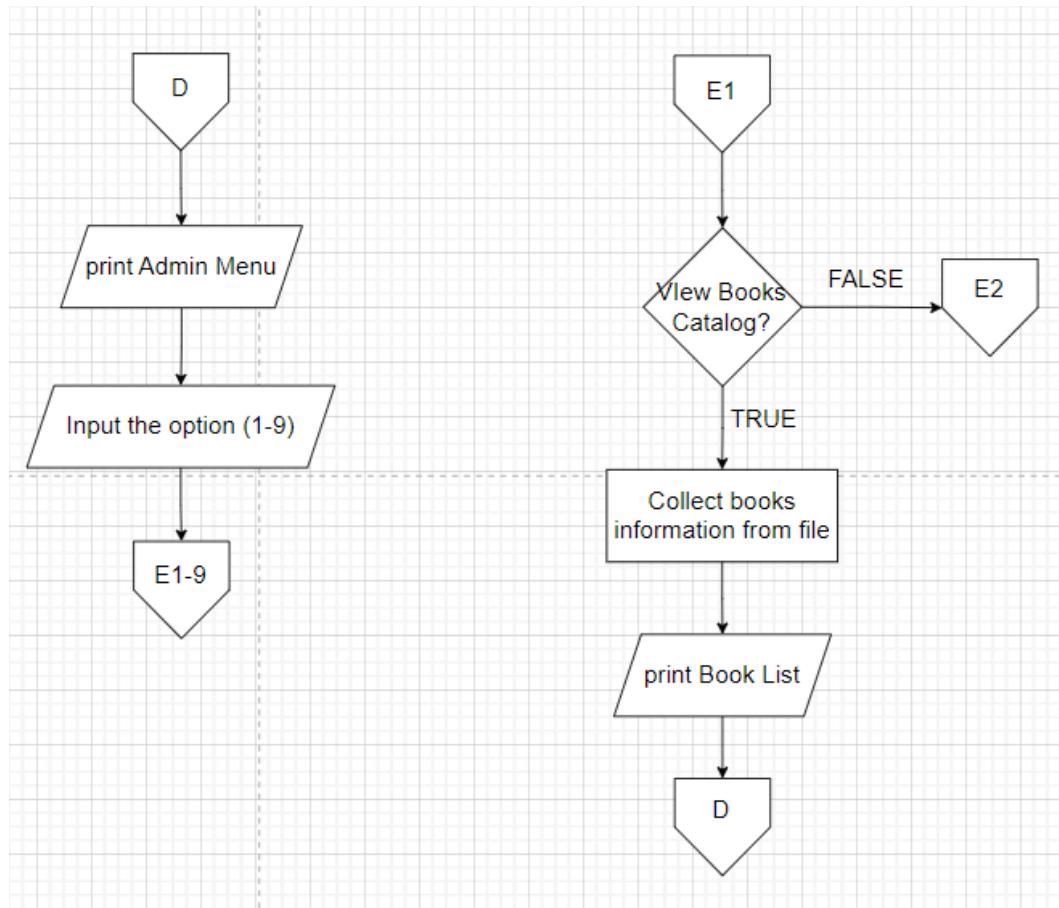


Figure 1.1.5: `viewMenu()` - admin

Figure 1.1.6: `viewAllBooks()` - admin

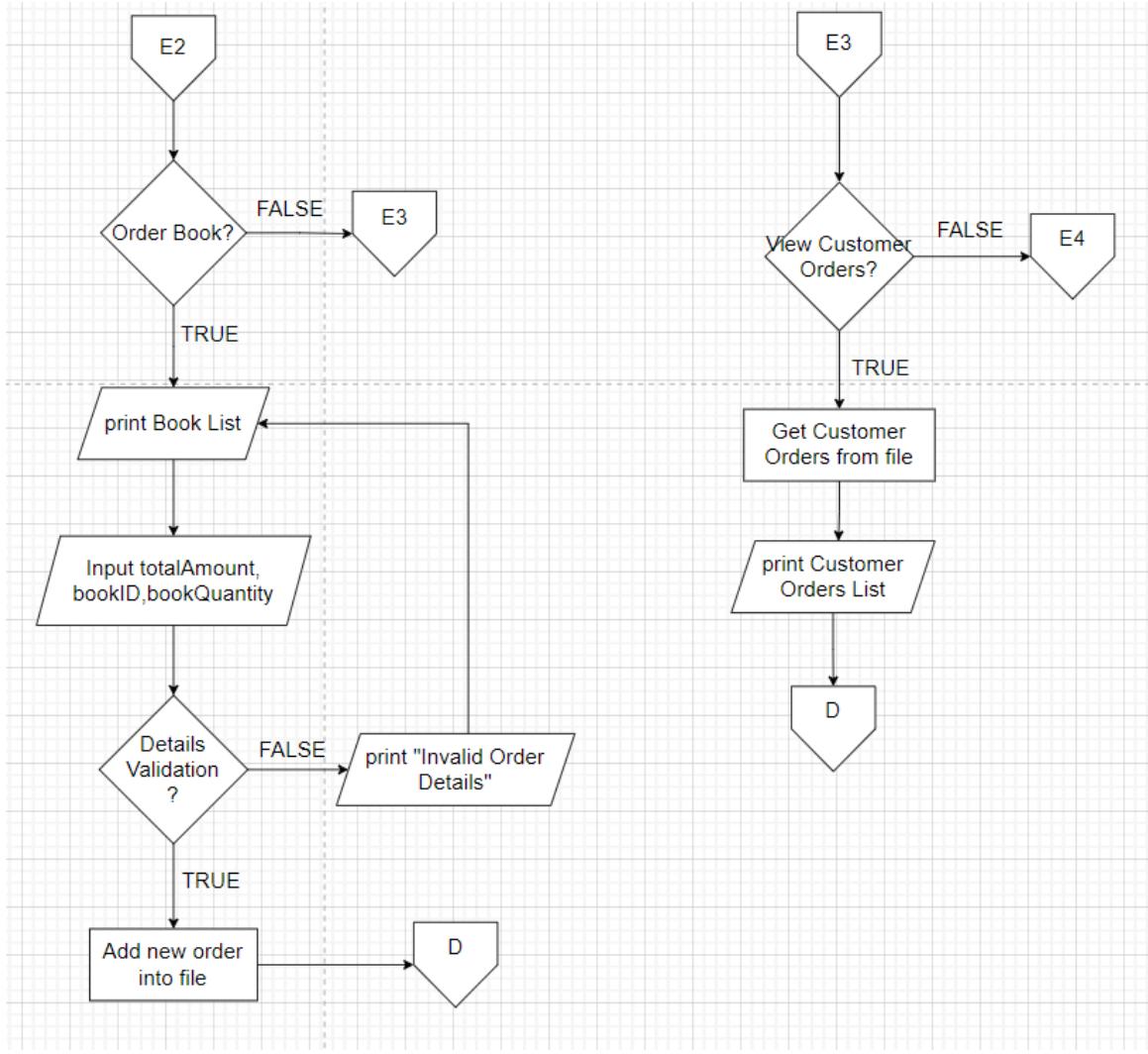


Figure 1.1.7: orderBooks() - admin

Figure 1.1.8: viewAllOrders() - Customer Orders

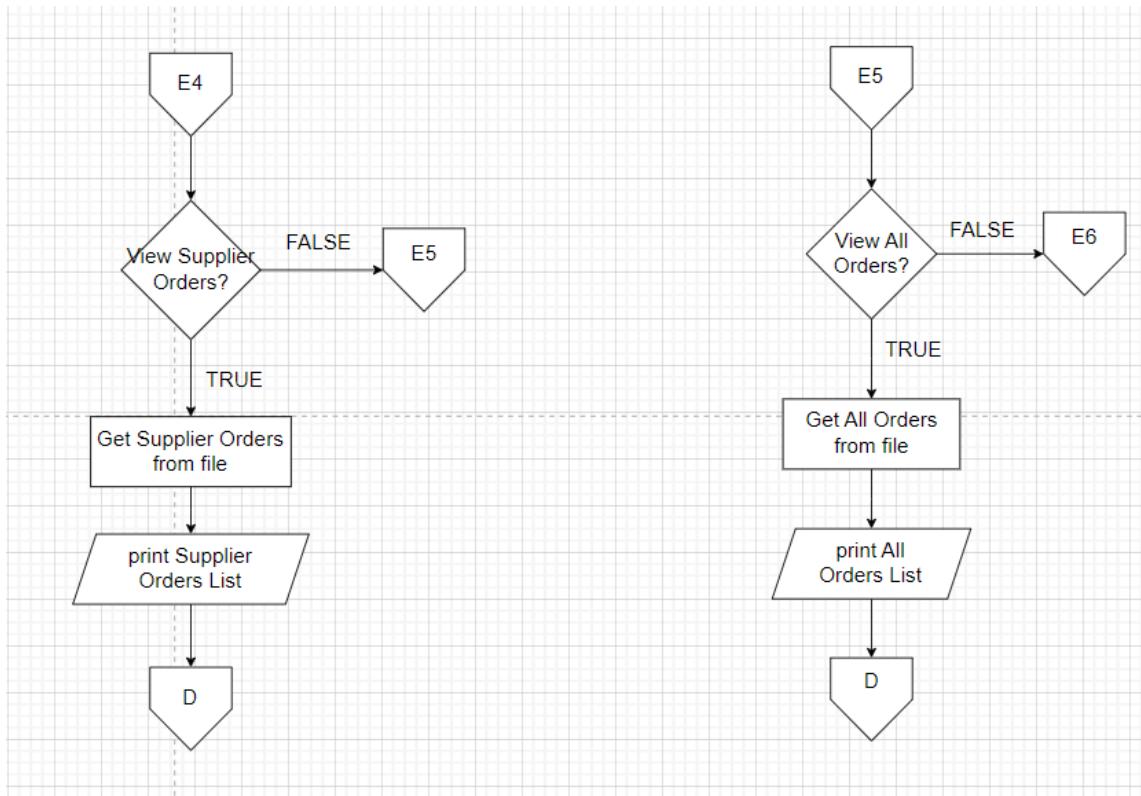


Figure 1.1.9: viewAllOrders()-Supplier Orders

Figure 1.1.10: viewAllOrders()-All Orders

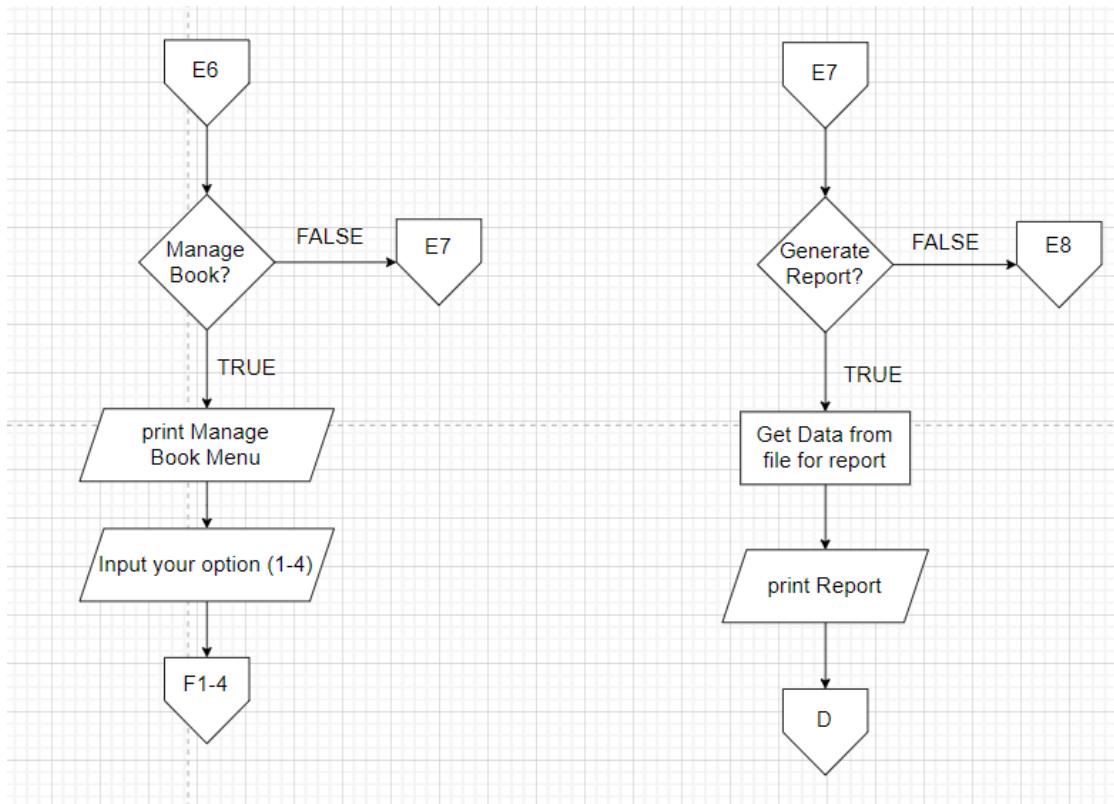


Figure 1.1.11: manageMenu() - admin

Figure 1.1.12: generateSalesReport() - admin

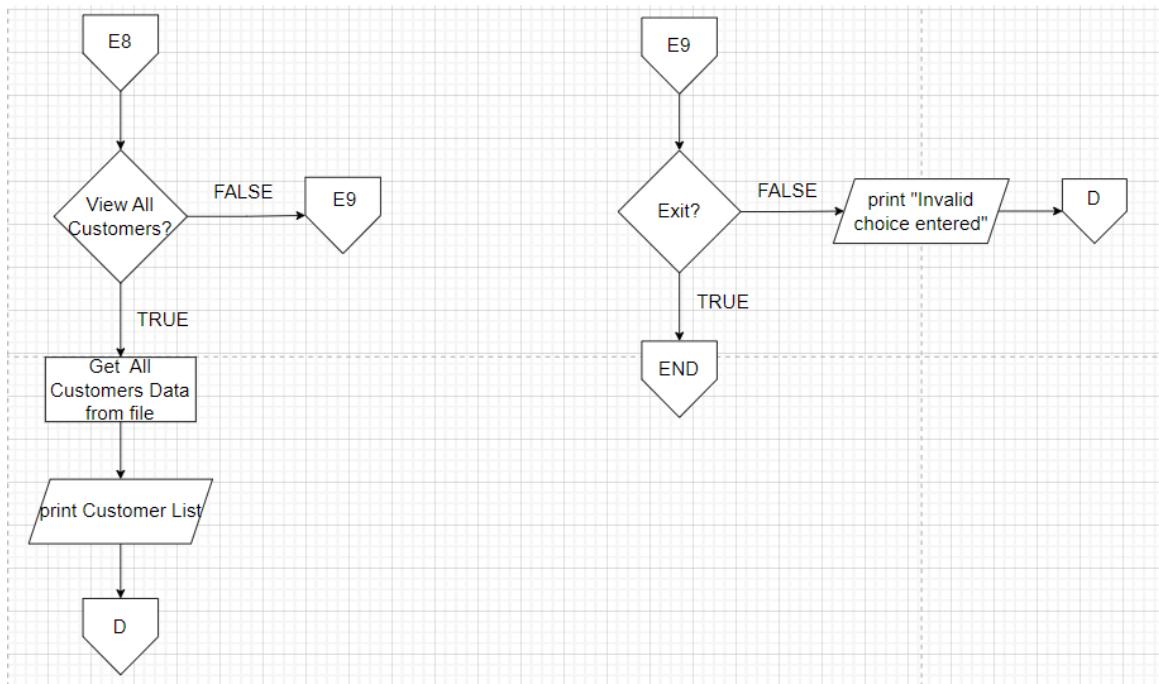


Figure 1.1.13: viewAllCustomer() - admin

Figure 1.1.14: Exit - admin

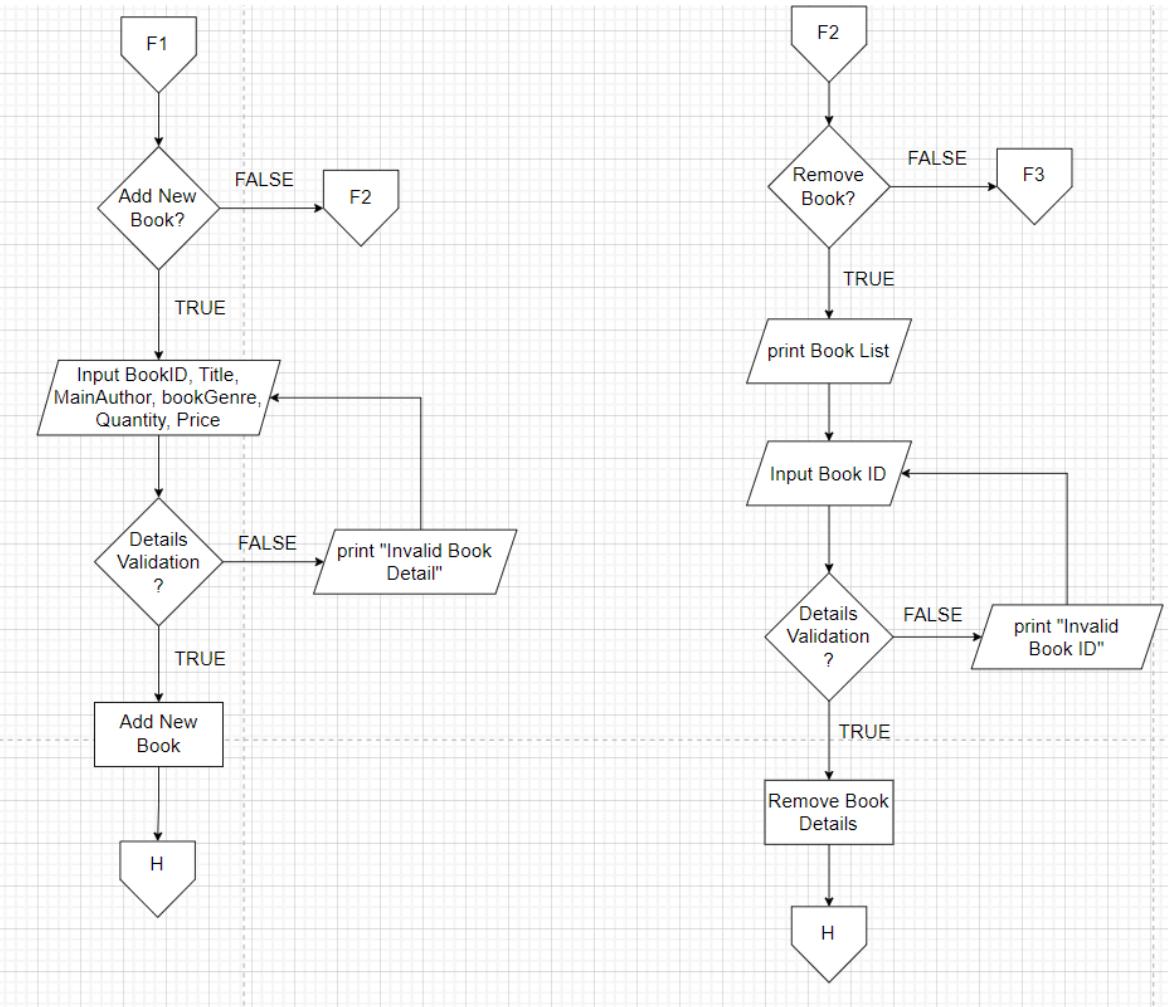


Figure 1.1.15: addBooksInfoFile()- admin

Figure 1.1.16: removeBookFromFile()- admin

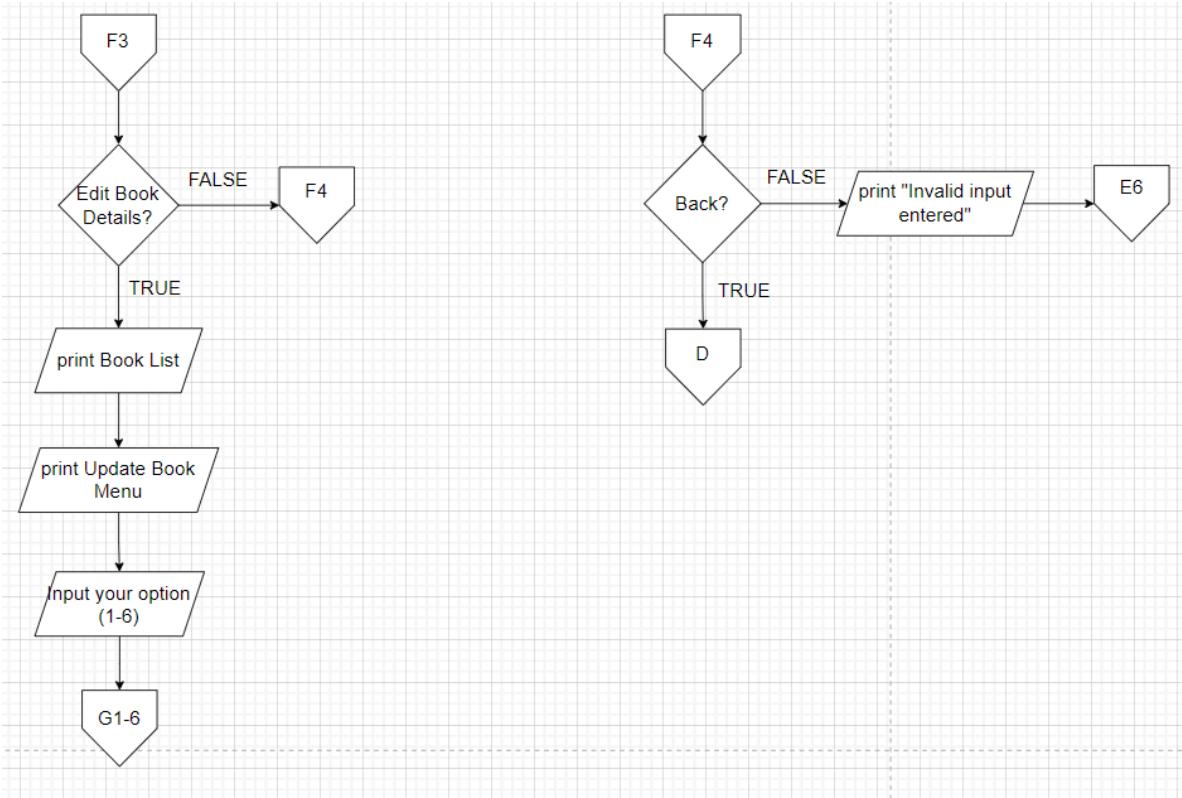


Figure 1.1.17: editBookDetails() - admin

Figure 1.1.18: Back - admin

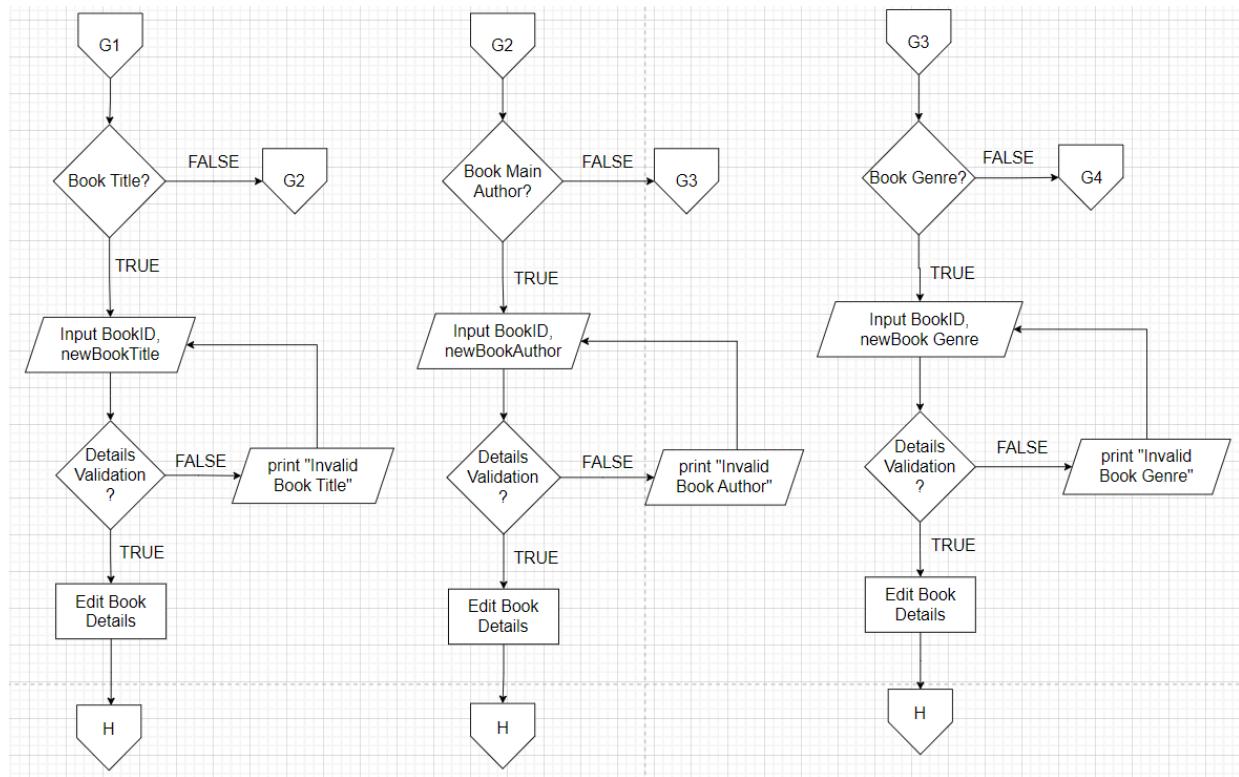


Figure 1.1.19: editBookDetails() - Title, MainAuthor, Genre (admin)

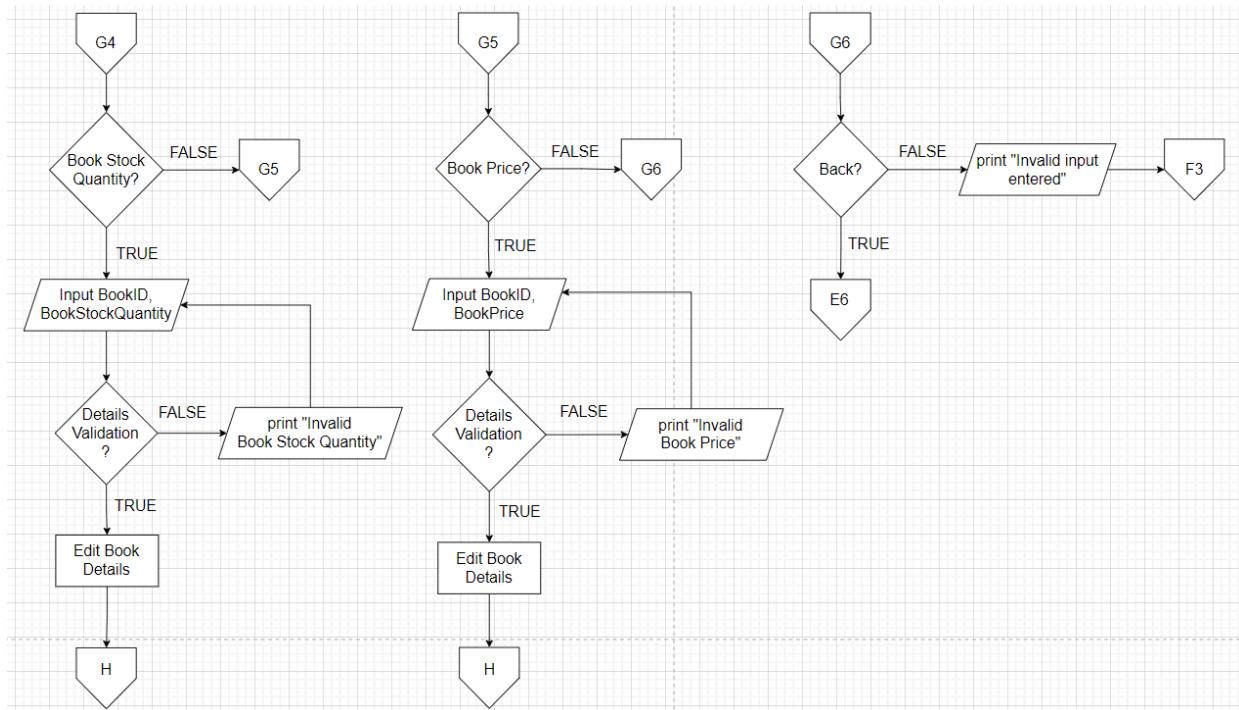


Figure 1.1.20: editBookDetails() - QuantityinStock, BookPrice, Back (admin)

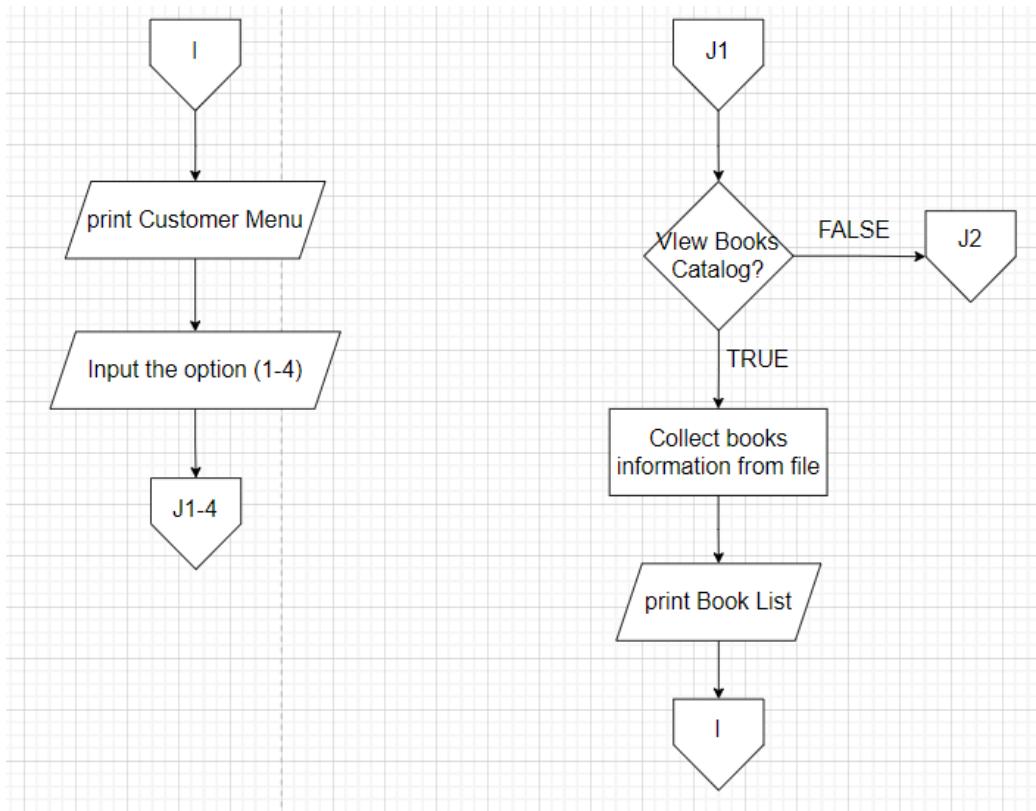


Figure 1.1.21: `viewMenu()` - Customer

Figure 1.1.22: `displayBooks()` - Customer

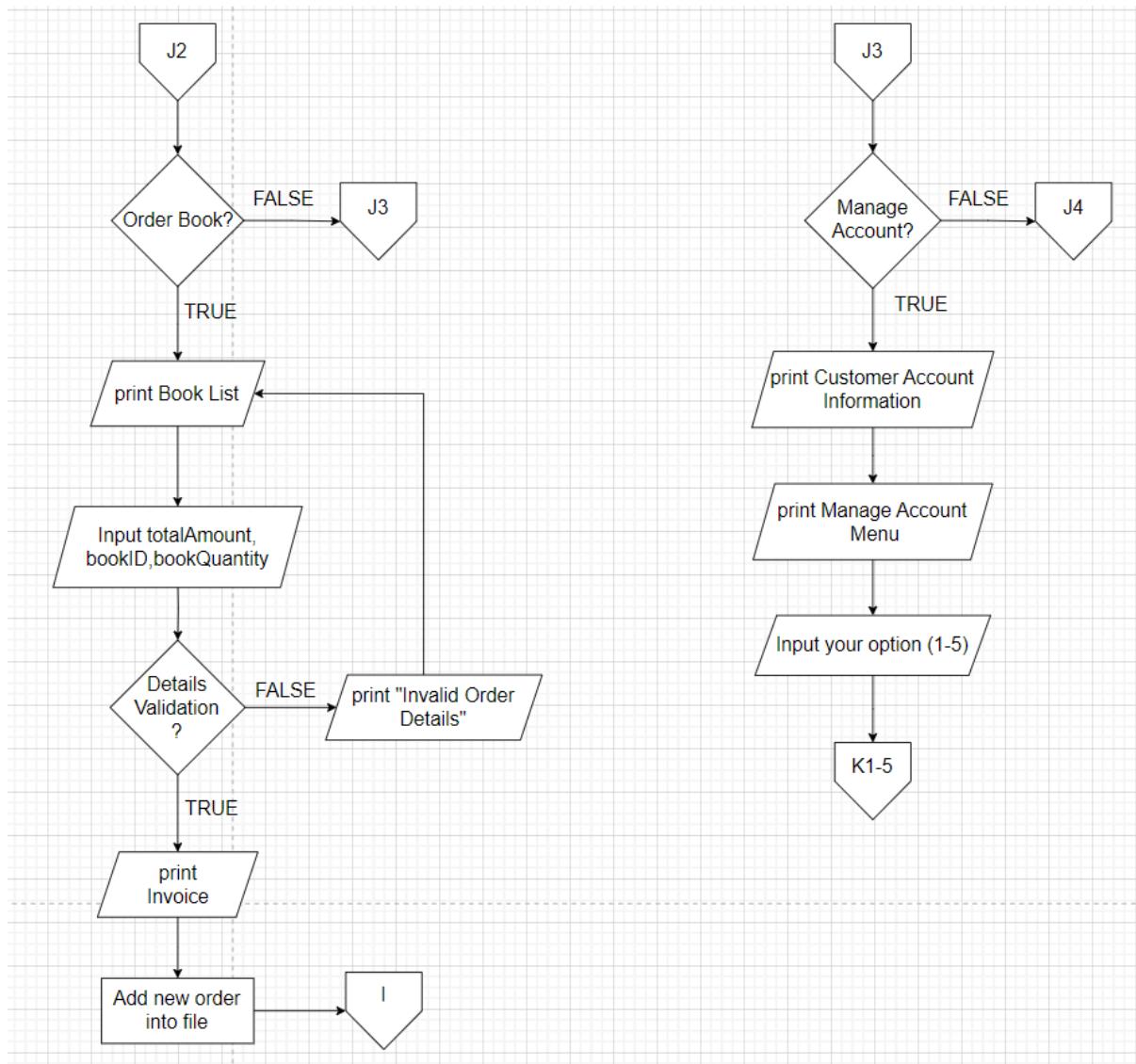


Figure 1.1.23: orderNewBooks()-Customer

Figure 1.1.24: updateCustomerAcc()-Customer

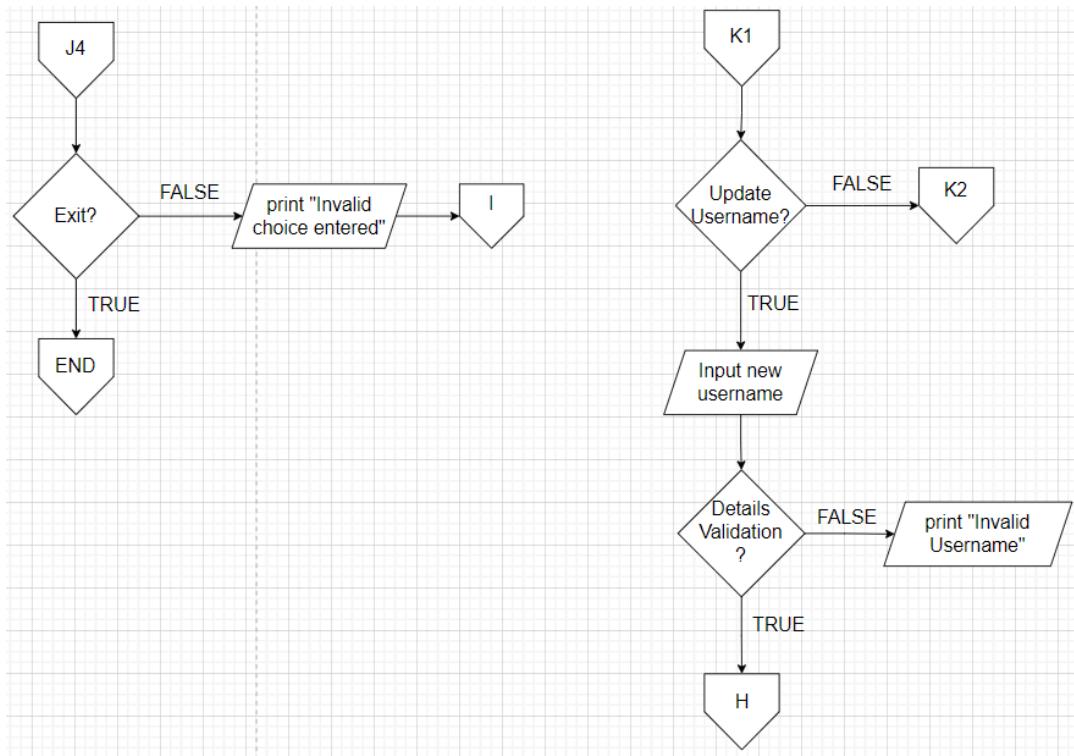


Figure 1.1.25: Exit - Customer

Figure 1.1.26: updateCustomerAcc() - username

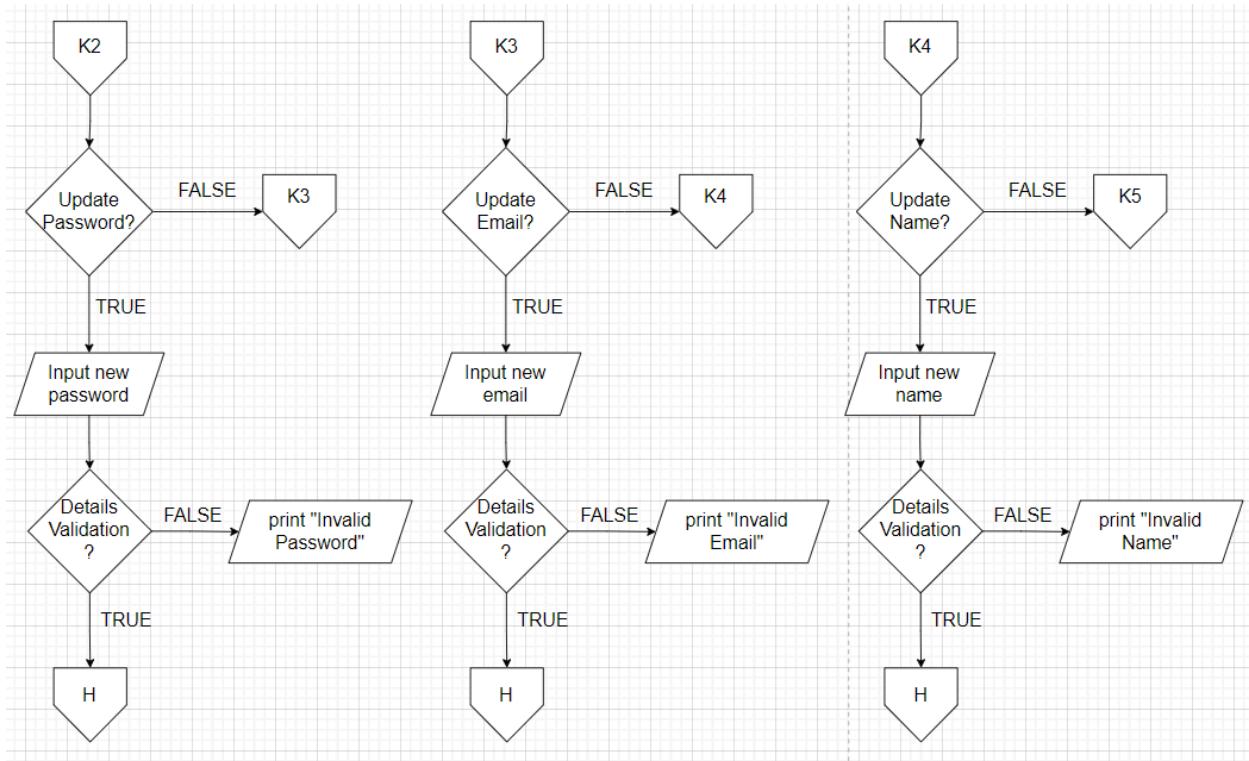
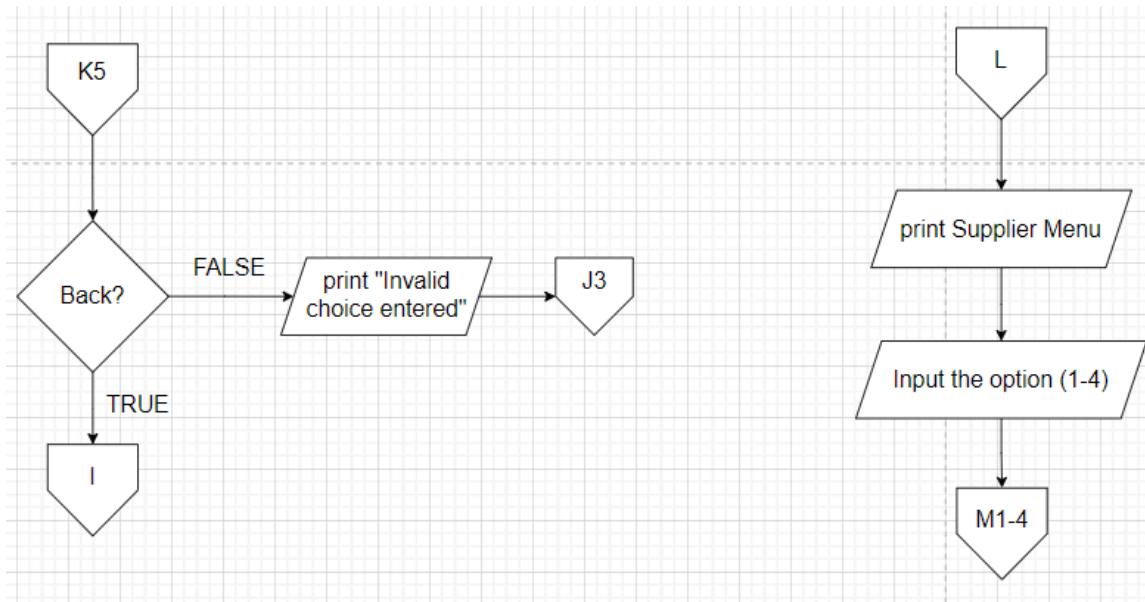


Figure 1.1.27: updateCustomerAcc() - password, email, name Customer



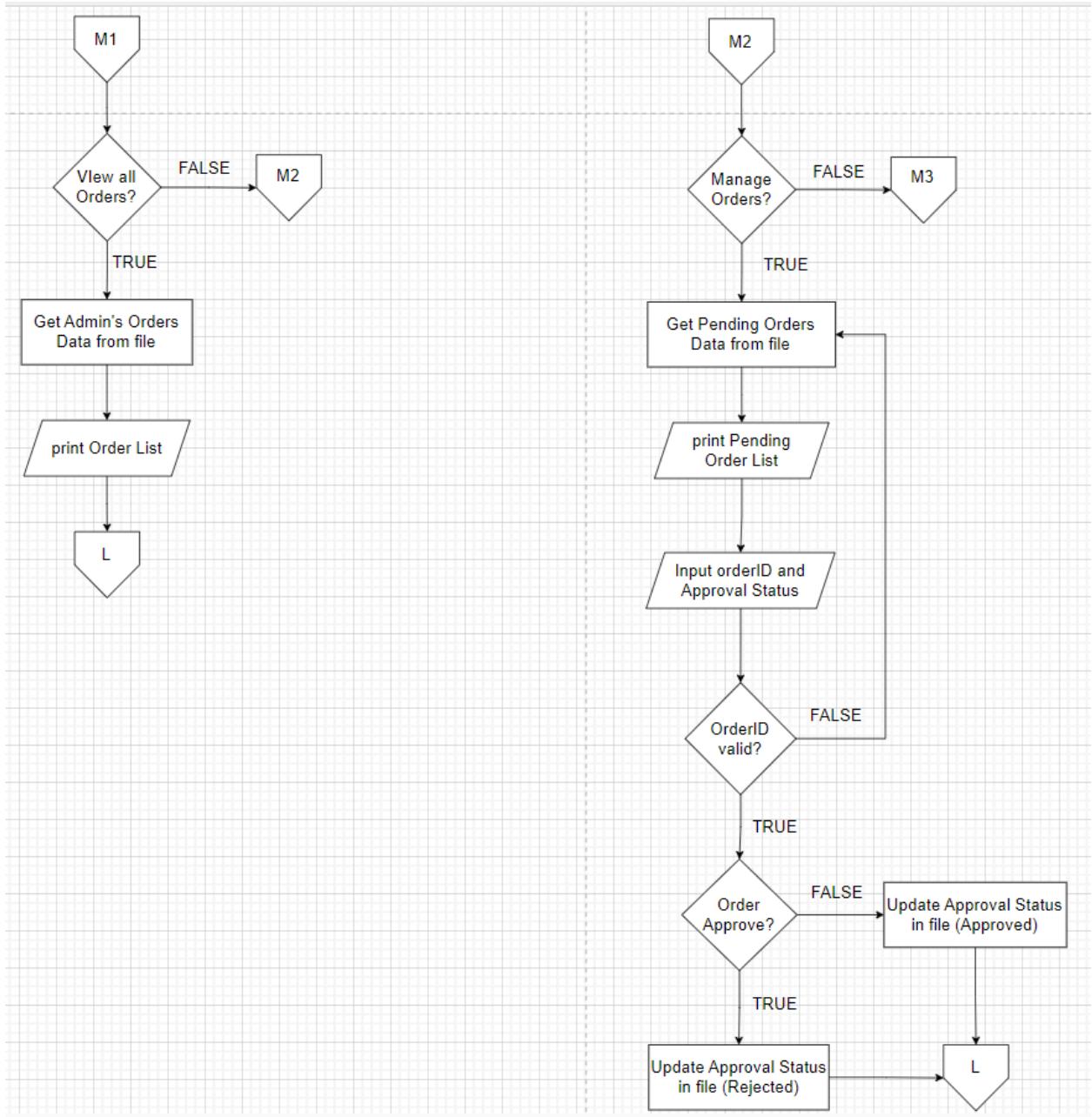


Figure 1.1.30: viewAllOrders() - Supplier

Figure 1.1.31: updateOrderStatus() - Supplier

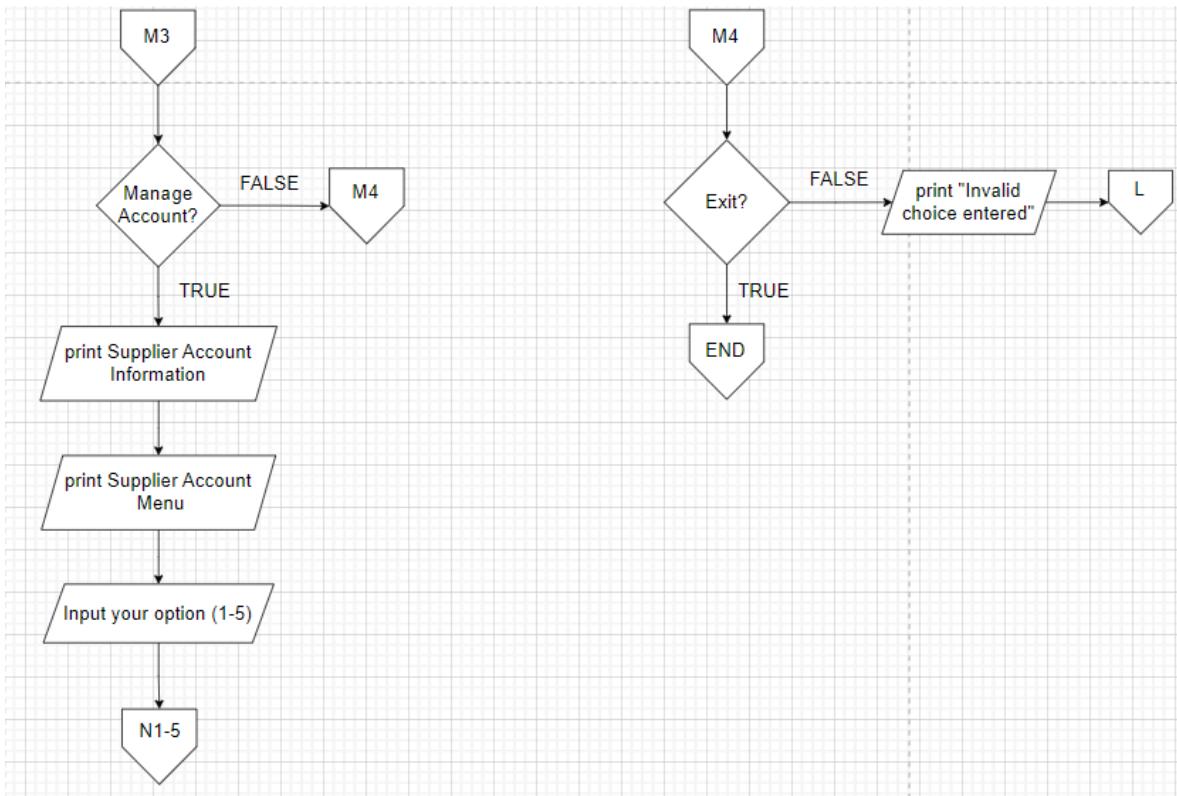


Figure 1.1.32: manageAccount() - Supplier

Figure 1.1.33: Exit - Supplier

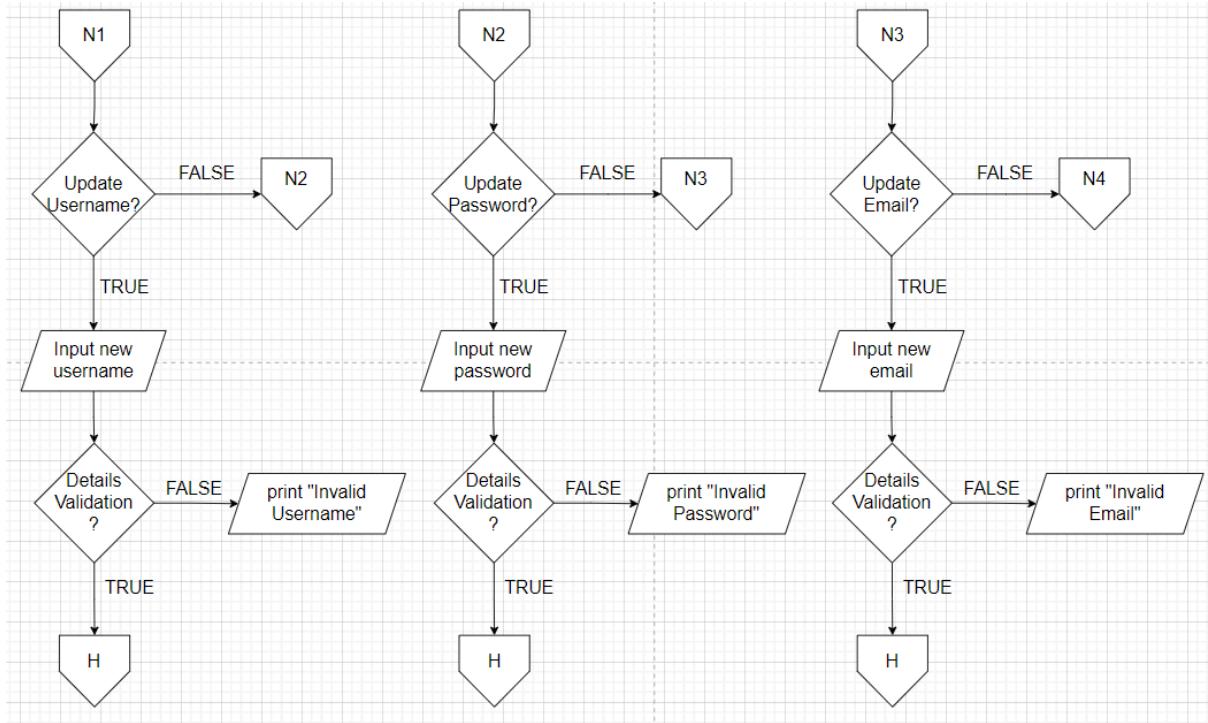
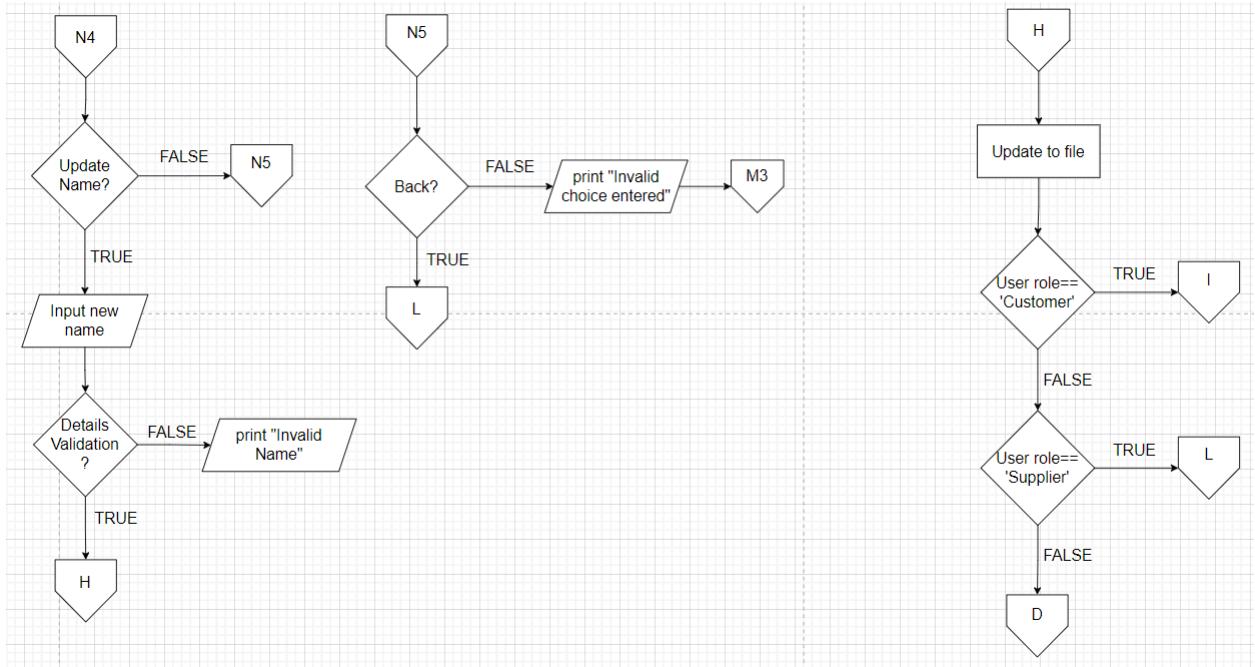


Figure 1.1.34: manageAccount() - username, password, email Supplier



4. OO Concepts

A. Encapsulation and data hiding

- Encapsulation involves bundling attributes and methods within a class, controlling access to the data through methods while hiding the internal implementation details. In the inventory system, classes encapsulate data related to books, orders, and user information. Public methods enable controlled access to this data, maintaining data integrity and security.

```
class Name{  
    private String fName;  
    private String lName;  
  
    Name(String f, String l){  
        fName = f;  
        lName = l;  
    }  
    //GETTERS  
    public String getfName(){  
        return fName;  
    }  
    public String getlName(){  
        return lName;  
    }  
  
    //SETTERS  
    public void setfName(String f){  
        fName = f;  
    }  
  
    public void setlName(String l){  
        lName = l;  
    }  
}
```

Figure 1.2.1: Illustrates an example of data encapsulation and hiding that have been implemented in the system under Name class.

B. Inheritance

- It is used to create a hierarchy in which classes declared as subclasses inherit properties and behaviour from superclasses. It established an “is-a” relationship between classes.

- Figure 2.0 shows the User class, which represents the parent class in the system. Meanwhile, Figure 2.1 Admin, Figure 2.2 Customer and Figure 2.3 BookSupplier classes are subclasses for the parent class. These subclasses inherit the methods and variables of their parent class, which are declared as public or protected as their access modifiers.

```

class User extends Menu{ //Parent class
    protected String userID;
    protected String userName;
    protected String password;
    protected String email;
    protected int userRole;
    private Name name;

    public User( String id, String names ,String pw, String mail, int roleID, String fName, String lName){
        userID = id;
        userName = names;
        password = pw;
        email = mail;
        userRole = roleID;
        name = new Name(fName,lName); //composition done
    }

    public User(){}
}

public String getUserID() {
    return userID;
}
public void setUserID(String userID) {
    this.userID = userID;
}
public String getUserName() {
    return userName;
}
public void setUserName(String userName) {
    this.userName = userName;
}

```

Figure 1.2.2: User parent class

```

class Admin extends User { //subclass
    private Vector<Customer> customers;
    private static Vector<Book> books;
    private static Vector<OrderManagement> orderList = new Vector<OrderManagement>();
    private Report report;

    public Admin(String id, String name ,String pw, String mail, int roleID, String fName, String lName, Vector<Book> bks){
        super(id, name, pw, mail, roleID, fName, lName);
        customers = new Vector<Customer>();
        books = bks; //Aggregation done
    }

    public Admin(){}
}

public void viewAllCustomers(Customer c) throws FileNotFoundException{
    System.out.print(s:"\u033[H\u033[2J");
    System.out.flush();
    InventorySystem.header();
    customers = c.getCustomersFromFile();
    c.viewAllCustomers(customers);
}

public Vector<Customer> getCustomers() {
    return customers;
}

public static Vector<Book> getBooks() {
    return books;
}

```

Figure 1.2.3: Admin subclass

```

public class Customer extends User { //subclass
    private Vector<Book> bookList;
    private Vector<OrderManagement> order = new Vector<OrderManagement>();

    public Customer(){}
    public Customer(String id, String name ,String pw, String mail, int roleID, String fName, String lName){
        super(id, name, pw, mail, roleID,fName,lName);
    }

    public static void registration() throws IOException{
        Vector<Customer> cust = new Vector<Customer>();
        cust = getCustomersFromFile();
        Scanner sc = new Scanner(System.in);
        System.out.print(s:"\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        System.out.print(s:"\n\nEnter your username : ");
        String username = sc.nextLine();

        for(Customer c:cust){
            while (c.getUserName().equals(username)) {
                System.out.println(x:"Username already exists. Please choose a different username.");
                System.out.print(s:"\nEnter a new username : ");
                username = sc.nextLine();
            }
        }
    }
}

```

Figure 1.2.4: Customer subclass

```

class BookSupplier extends User{ //Subclass
    private Vector<OrderManagement> orderlist;
    public BookSupplier(String id, String name ,String pw, String mail, int roleID,String fName, String lName){
        super(id, name, pw, mail, roleID,fName,lName);
    }

    public BookSupplier(){}

    public static void displaySupplier(User u){
        System.out.println(x:");
        System.out.println(x:"| Supplier ID | Username | Full Name | Email |");
        System.out.println(x:"|-----|-----|-----|-----|");

        System.out.printf(format:"| %15s | %19s | %19s | %19s |%n", u.getUserID(), u.getUserName(), u.getName().getfName() + " " + u.getName().getlName());
        System.out.println(x:"|-----|-----|-----|-----|");
    }
}

```

Figure 1.2.5: BookSupplier subclass

C. Composition

- A strong relationship in which one class contains objects of another class as part of its structure. Composition is a way to design classes where one class contains an object of another class, establishing a "has-a" relationship.

- Figure 3.2 shows the User class and its composition relationship with a name class in Figure 3.1. The User class has a Name object in its structure, according to this composition which demonstrates how the User class integrates and depends on the attributes and functions of the Name object. This implies that the User class has a Name object, which leads to an interconnection of those two classes.

```

class Name{
    private String fName;
    private String lName;

    Name(String f, String l){
        fName = f;
        lName = l;
    }
    //GETTERS
    public String getfName(){
        return fName;
    }
    public String getlName(){
        return lName;
    }

    //SETTERS
    public void setfName(String f){
        fName = f;
    }

    public void setlName(String l){
        lName = l;
    }
}

```

Figure 1.2.6: Name Class

```

class User extends Menu{ //Parent class
    protected String userID;
    protected String userName;
    protected String password;
    protected String email;
    protected int userRole;
    private Name name;

    public User( String id, String names ,String pw, String mail, int roleID, String fName, String lName){
        userID = id;
        userName = names;
        password = pw;
        email = mail;
        userRole = roleID;
        name = new Name(fName,lName); //composition done
    }

    public User(){}
}

```

Figure 1.2.7: User Class

D. Aggregation

- A loose relationship in which one class is associated with another class but does not own its objects.
- An aggregate relationship, which includes a static Vector<Book> called books, is shown in Figure 4.2 of the Administration class. The Admin class constructor receives a Book class vector as an argument, indicating that the Admin class is merging the Book objects. We have developed the getter and setter methods for a book vector to show that we can retrieve and modify any combination of books. Figure 4.1 shows the Book class implementation.

```

class Book[] //aggregate to Admin
private String bookID;
private String title;
private String mainAuthor;
private int genre;
private int quantityInStock;
private double bookPrice;

public Book(){}
public Book(String id, String titleBook, String author, int genreCat, int stock,double price){
    bookID = id;
    title = titleBook;
    mainAuthor = author;
    genre = genreCat;
    quantityInStock = stock;
    bookPrice = price;
}

tabnine: test | explain | document | ask
public double getBookPrice() {
    return bookPrice;
}
tabnine: test | explain | document | ask
public void setBookPrice(double bookPrice) {
    this.bookPrice = bookPrice;
}

```

Figure 1.2.8: Book Class

```

class Admin extends User {
    private Vector<Customer> customers;
    private static Vector<Book> books;
    private static Vector<OrderManagement> orderList = new Vector<OrderManagement>();
    private Report report;

    public Admin(String id, String name ,String pw, String mail, int roleID, String fName, String lName, Vector<Book> bks){
        super(id, name, pw, mail, roleID,fName,lName);
        customers = new Vector<Customer>();
        books = bks; //Aggregation done
    }

    public Admin(){}

    tabnine: test | explain | document | ask
    public void viewAllCustomers(Customer c) throws FileNotFoundException{
        System.out.print(s:"\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        customers = c.getCustomersFromFile();
        c.viewAllCustomers(customers);
    }
}

```

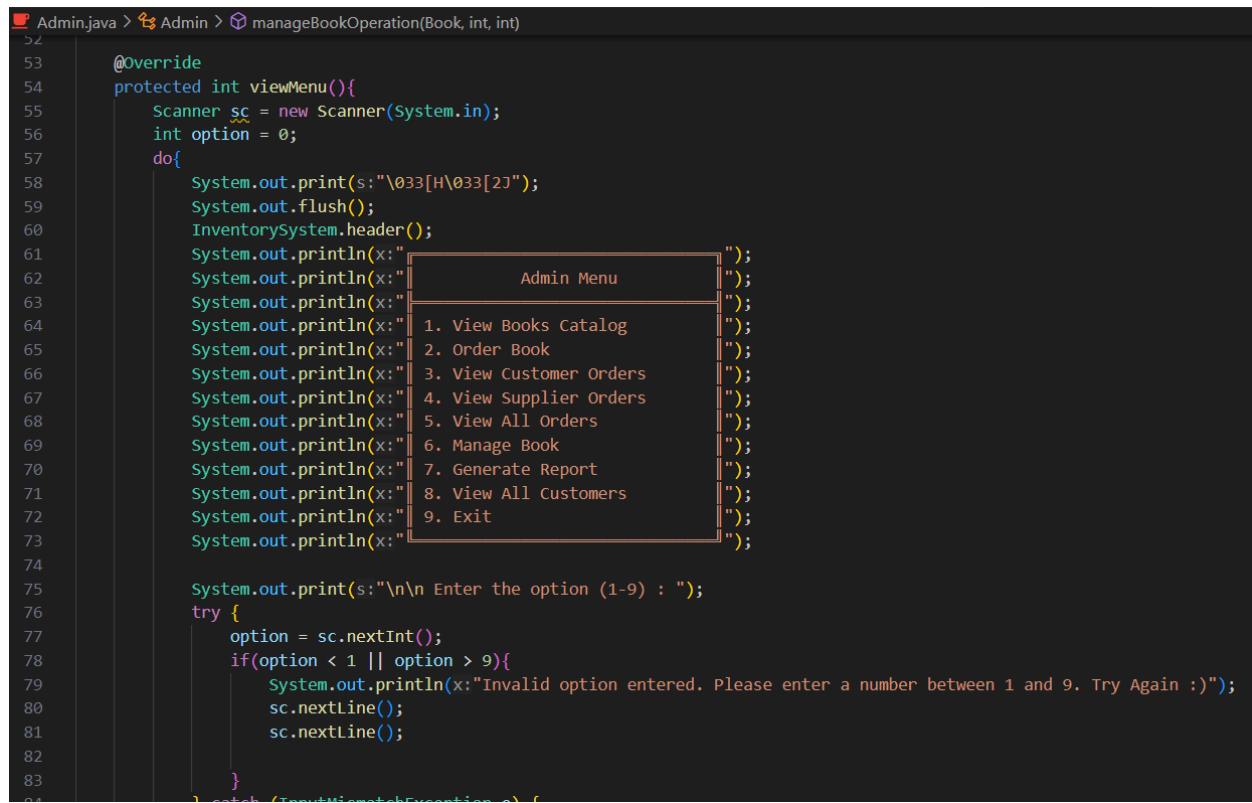
Figure 1.2.9: Admin Class

E. Polymorphism:

- The ability to treat different classes as instances of the same class through inheritance and method overriding.
- The Menu class in Figure 5.0 declares an abstract method called viewMenu(). The method viewMenu() is implemented with polymorphism features. The Admin class and User class in Figure 5.1 and 5.2 inherits from the Menu class and demonstrates polymorphism by offering a distinct implementation for the viewMenu() method. The method's implementation in both classes overrides the abstract method declared in the base class, showcasing the flexibility and adaptability of polymorphism within the structure of inheritance and method overriding.

```
abstract class Menu { //abstract class that is used by the user superclass and its subclasses to perform polymorphism
    abstract protected int viewMenu(); //abstract function
}
```

Figure 1.2.10: Menu Class



```
Admin.java > Admin > manageBookOperation(Book, int, int)
52
53     @Override
54     protected int viewMenu(){
55         Scanner sc = new Scanner(System.in);
56         int option = 0;
57         do{
58             System.out.print(s:"\033[H\033[2J");
59             System.out.flush();
60             InventorySystem.header();
61             System.out.println(x:"" Admin Menu "");
62             System.out.println(x:"" 1. View Books Catalog "");
63             System.out.println(x:"" 2. Order Book "");
64             System.out.println(x:"" 3. View Customer Orders "");
65             System.out.println(x:"" 4. View Supplier Orders "");
66             System.out.println(x:"" 5. View All Orders "");
67             System.out.println(x:"" 6. Manage Book "");
68             System.out.println(x:"" 7. Generate Report "");
69             System.out.println(x:"" 8. View All Customers "");
70             System.out.println(x:"" 9. Exit "");
71             System.out.println(x:"" );
72
73             system.out.print(s:"\n\n Enter the option (1-9) : ");
74             try {
75                 option = sc.nextInt();
76                 if(option < 1 || option > 9){
77                     System.out.println(x:"Invalid option entered. Please enter a number between 1 and 9. Try Again :");
78                     sc.nextLine();
79                     sc.nextLine();
80
81                 }
82             } catch (InputMismatchException e) {
83                 System.out.println(x:"Input mismatch detected. Please enter a valid integer value.");
84             }
85         }
86     }
```

Figure 1.2.11: viewMenu() method in Admin class

```
User.java > User > setPassword(String)
103     String fName = ary[0];
104     String lName = ary[1];
105     User temp = new User(id, name, pw, mail, roleID,fName,lName);
106     cust.add(temp);
107 }
108 }
109 }
110
111 sc.nextLine();
112 sc.close();
113
114 return cust;
115
116 }
117
118 protected int viewMenu(){
119     return 0;
120 } //Polymorphism done using abstraction.
121 }
```

Figure 1.2.12: Viewmenu method

F. Exception handling:

- It is used to manage exceptions that may occur during program execution and ensure that errors or unexpected situations are properly handled. In an inventory system, this might involve handling scenarios such as incorrect user input during order processing or data entry about products.
- In order to effectively manage certain exceptions, the viewMenu(), generateReport(), and validation() methods have been integrated with "try" and "catch" blocks as shown in Figures 6.0, 6.1 and 6.2 respectively. Exceptions such as FileNotFoundException and InputMismatchException are used to provide special information about the type of exception that occurred. Moreover, this feature is used to update and include error messages that provide more information in order to improve clarity and facilitate the process of understanding and resolving issues.

```

protected int viewMenu(){
    Scanner sc = new Scanner(System.in);
    int option = 0;
    do{
        System.out.print(s:"\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        System.out.println(x:"          Supplier Menu          ");
        System.out.println(x:" 1. View all Orders   ");
        System.out.println(x:" 2. Manage Orders   ");
        System.out.println(x:" 3. Manage Account   ");
        System.out.println(x:" 4. Exit           ");
        System.out.println(x:"");

        System.out.print(s:"\n\n Enter the option (1-4) : ");
        try {
            option = sc.nextInt();
            if(option < 1 || option > 4){
                System.out.println(x:"Invalid option entered. Please enter a number between 1 and 4. Try Again :");
                sc.nextLine();
                sc.nextLine();
            }
        } catch (InputMismatchException e) {
            System.out.println(x:"Invalid option entered. Please Enter an appropriate number.\nPress any key to continue...");
            sc.nextLine();
            sc.nextLine();
            option = 10;
        }
    }while(option < 1 || option > 4);
    return option;
}

```

Figure 1.2.13: `viewMenu()` method with `InputMismatchException` try-catch block.

```

public void generateReport(int roleID) throws FileNotFoundException{
    System.out.print(s:"\033[H\033[2J");
    System.out.flush();
    InventorySystem.header();
    Scanner sc = new Scanner(System.in);
    OrderManagement orders = new OrderManagement();
    orderList = orders.getOrderFromFile(roleID);
    if(orderList.size() == 0){
        System.out.println(x:"No Order Found.\nPress any key to continue..");
        return;
    }
    System.out.println(
        x:""
    );
    System.out.println(
        x:"          REPORT KEDAI BUKU KAMAL          "
    );
    System.out.println(
        x:""
    );
    System.out.println(
        x:"          TOP 10 BOOKS SOLD          "
    );
    System.out.println(
        x:"-----"
    );
    System.out.printf(
        format:"%-8s | %-30s | %-10s\n", ...args:"Book ID", "Title", "Quantity");
    System.out.println(
        x:"-----"
    );

    double totalAmount = 0;
    double totalAmountOrder = 0;
}

```

Figure 1.2.14: generateReport() method with FileNotFoundException.

```
public boolean validation(String bookId, int bookQuantity, int roleId) throws FileNotFoundException{
    Vector<Book> bkList = new Vector<Book>();
    Book book = new Book();
    Boolean check = false;
    bkList = book.getBooksFromFile();

    try {
        for (Book b : bkList) {
            if (b.getBookID().equals(bookId)) {
                check = true;
                break; // Exit the loop if the book is found
            }
        }

        if (!check) {
            System.out.println("Sorry, the book id you entered is not found.");
        }
    } catch (InputMismatchException e) {
        System.out.println(e.getMessage());
        check = false;
    }

    if(roleId == 2){
        for(Book b:bkList){
            if(b.getBookID().equals(bookId)){
                if(b.getQuantityInStock() >= bookQuantity){
                    check = true;
                    break; // Exit the loop if the book is found
                }else{
                    System.out.println("Sorry, the quantity you order is more than the quantity in stock.");
                    check = false;
                }
            }
        }
    }
}
```

Figure 1.2.15: validation() method with InputMismatchException try-catch block.

G. Association

- An association can be one-to-one, one-to-many, or many-to-many. The Admin class has associations with the Report and OrderManagement classes. Figure 7.0 shows the Admin class implementing association by declaring orderList variable that represents OrderManagement class and report variable that represents Report class within the class.
- The generateSalesReport() method represents an association between the Admin class with the Report class, where an Admin can generate sales reports using a Report object in Figure 7.1. Meanwhile, the orderBooks() method represents an association between Admin class and OrderManagement class, where an Admin can add orders to its orderList variable using an OrderManagement object in Figure 7.1.

```

class Admin extends User { //subclass
    private Vector<Customer> customers;
    private static Vector<Book> books;
    private static Vector<OrderManagement> orderList = new Vector<OrderManagement>();
    private Report report;

    public Admin(String id, String name ,String pw, String mail, int roleID, String fName, String lName, Vector<Book> bks){
        super(id, name, pw, mail, roleID,fName,lName);
        customers = new Vector<Customer>();
        books = bks; //Aggregation done
    }

    public Admin(){}

```

Figure 1.2.16: Implementation of declaring variables that represents Report and OrderManagement class.

```

Admin.java > Admin > manageBookOperation(Book, int, int)

94
95     public void generateSalesReport(Report r) throws FileNotFoundException{
96         report = r;
97         report.generateReport(roleID:1);
98     }
99
100    public void orderBooks(OrderManagement o) {
101        orderList.add(o);
102    }
103

```

Figure 1.2.17: Functions that are used to create association between Admin and Report Class using generateSalesReport() and association between Admin and OrderManagement class using orderBooks().

H. File Operation (Add, Delete, Edit, View)

- Figure 8.0 until Figure 8.3 shows how the system implements file operation concept to perform new book addition into file, removing a certain book detail from file, reading all the details from the file and modifying certain book details and updating it into the file.

- For example, figure 8.1 shows addBooksIntoFile() method that adds new book details entered by the user into the booksDatabase.txt text file.

```

public Vector<Book> getBooksFromFile() throws FileNotFoundException {
    Vector<Book> books = new Vector<Book>();
    Scanner sc = new Scanner(new File(pathname:"booksDatabase.txt"));

    while(sc.hasNext()){
        String bookId = sc.next();
        String title = sc.next();
        String mainAuthor = sc.next();
        int genre = sc.nextInt();
        int quantityInStock = sc.nextInt();
        double bookPrice = sc.nextDouble();
        Book book = new Book(bookId, title, mainAuthor, genre, quantityInStock,bookPrice);
        books.add(book);
    }
    return books;
}

```

Figure 1.2.18: getBooksFromFile() method

```

public void addBooksIntoFile(Vector<Book> bk) throws IOException{
    Scanner scan = new Scanner(System.in);
    System.out.println(x:"=====ADD BOOK=====");
    System.out.print(s:"\nEnter Book ID: ");
    String id = scan.nextLine();

    Vector<Book> bkList = new Vector<Book>();
    bkList = getBooksFromFile();

    for(Book c:bkList){
        while (c.getBookID().equals(id.toUpperCase())) {
            System.out.println(x:"Book ID already exists. Please choose a different Book ID.");
            System.out.print(s:"\nEnter a new Book ID : ");
            id = scan.nextLine().toUpperCase();
        }
    }

    System.out.print(s:"\nEnter Title: ");
    String ttl = scan.nextLine();

    System.out.print(s:"\nEnter Main Author: ");
    String mainAuthor = scan.nextLine();
    int genreOption = 0;
    do{
        System.out.print(s:"\nEnter new book genre (1-Romance , 2-Mystery , 3-Fantasy , 4-Comedy , 5-Thriller) : ");
        try {
            genreOption = scan.nextInt();
            if(genreOption <1 || genreOption>5){
                System.out.println(x:"Invalid option entered. Please enter a number between 1 and 5. Try Again :");
            }
        } catch (InputMismatchException e) {
            System.out.println(x:"Invalid option entered. Please Enter an appropriate number.\nPress any key to continue...");
            scan.nextLine();
            genreOption = 8;
        }
    }
}

```

Figure 1.2.19: addBooksIntoFile() method

```

public Boolean editBookDetails(Vector<Book> books,int category, String id, String value) throws IOException{
    value = value.replaceAll(regex:" ", replacement:"_");
    for(Book book:books){
        if(book.getBookID().equals(id.toUpperCase())){
            switch (category) {
                case 1:
                    book.setTitle(value);
                    break;
                case 2:
                    book.setMainAuthor(value);
                    break;
                case 3:
                    book.setGenre(Integer.parseInt(value));
                    break;
                case 4:
                    book.setQuantityInStock(Integer.parseInt(value));
                    break;
                case 5:
                    book.setBookPrice(Double.parseDouble(value));
                    break;
                default:
                    return false;
            }
        }
    }
    if(books !=null){
        FileWriter file = new FileWriter(fileName:"booksDatabase.txt",append:false);
        for (Book bk : books) {
            file.write(bk.getBookID().toUpperCase()+" "+bk.getTitle()+" "+bk.getMainAuthor()+" "+bk.getGenre()+" "+bk.getQuantityInStock());
        }
        file.close();
    }
    return true;
}

```

Figure 1.2.20: editBookDetails() method

```

public void removeBookFromFile(Vector<Book> bk2) throws IOException{
    Scanner scan = new Scanner(System.in);

    if(bk2.size() != 0){
        viewAllBooks(bk2,role:1);
        String bookId = "";
        int index =0;

        System.out.print(s:"\nEnter Book ID : ");
        bookId = scan.nextLine().toUpperCase();
        boolean validation = false;

        for (Book bk : bk2) {
            if (bk.getBookID().equals(bookId.toUpperCase())) {
                validation = true;
                break;
            }
        }

        while(validation == false){
            System.out.println(x:"Book ID not exists in our database. Please Try Again!");
            System.out.print(s:"\nEnter Book ID : ");
            bookId = scan.nextLine().toUpperCase();
            for (Book bk : bk2) {
                if (bk.getBookID().equals(bookId.toUpperCase())) {
                    validation = true;
                    break;
                }
            }
        }
        PrintWriter outputFile = new PrintWriter(new FileWriter(fileName:"booksDatabase.txt"),autoFlush:false);
        for(Book book:bk2){
            if(book.getBookID().equals(bookId.toUpperCase())){
                break;
            }
        }
    }
}

```

Figure 1.2.21: removeBookFromFile() method

SECTION B: UML CLASS DIAGRAM

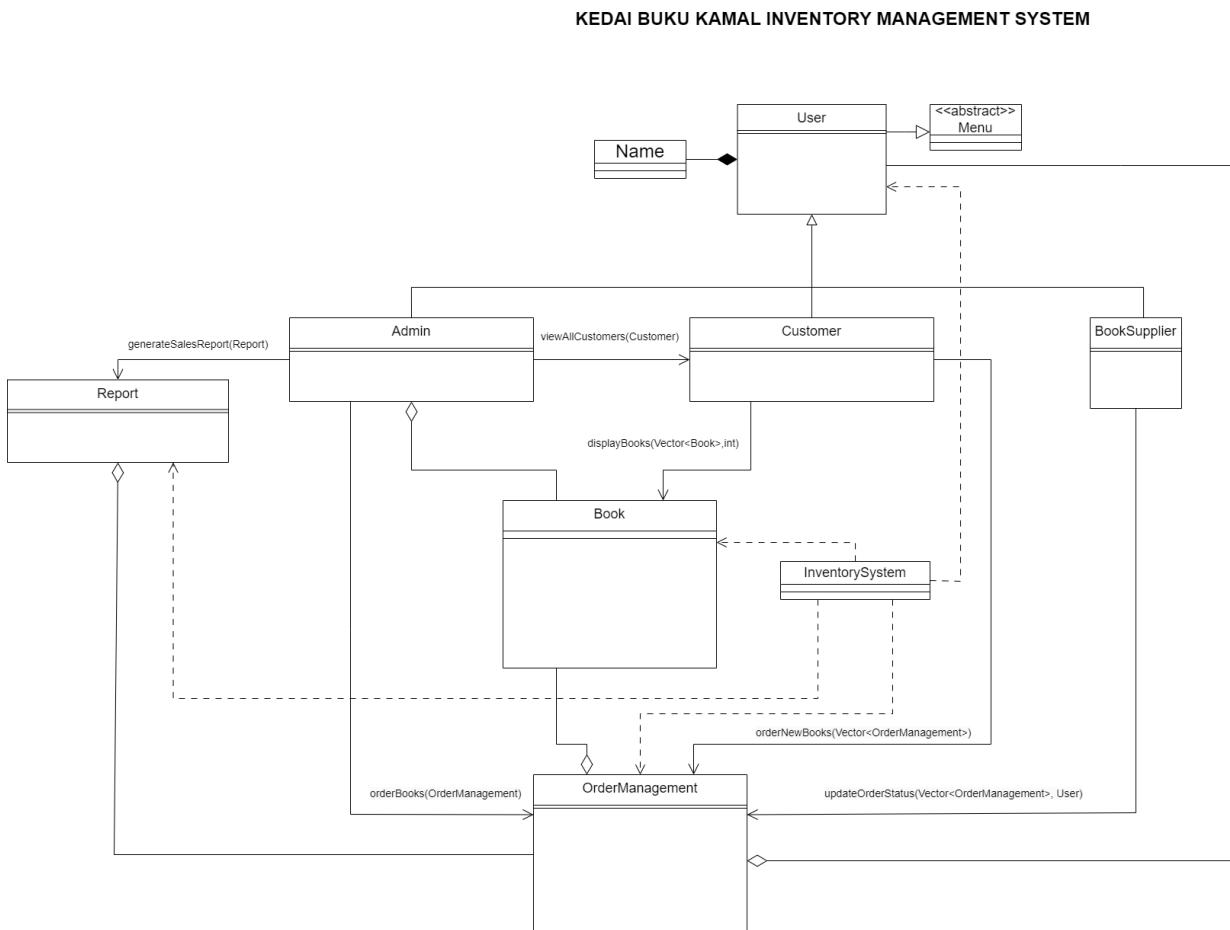


Figure 2.1 : UML class Figure (class with no data)

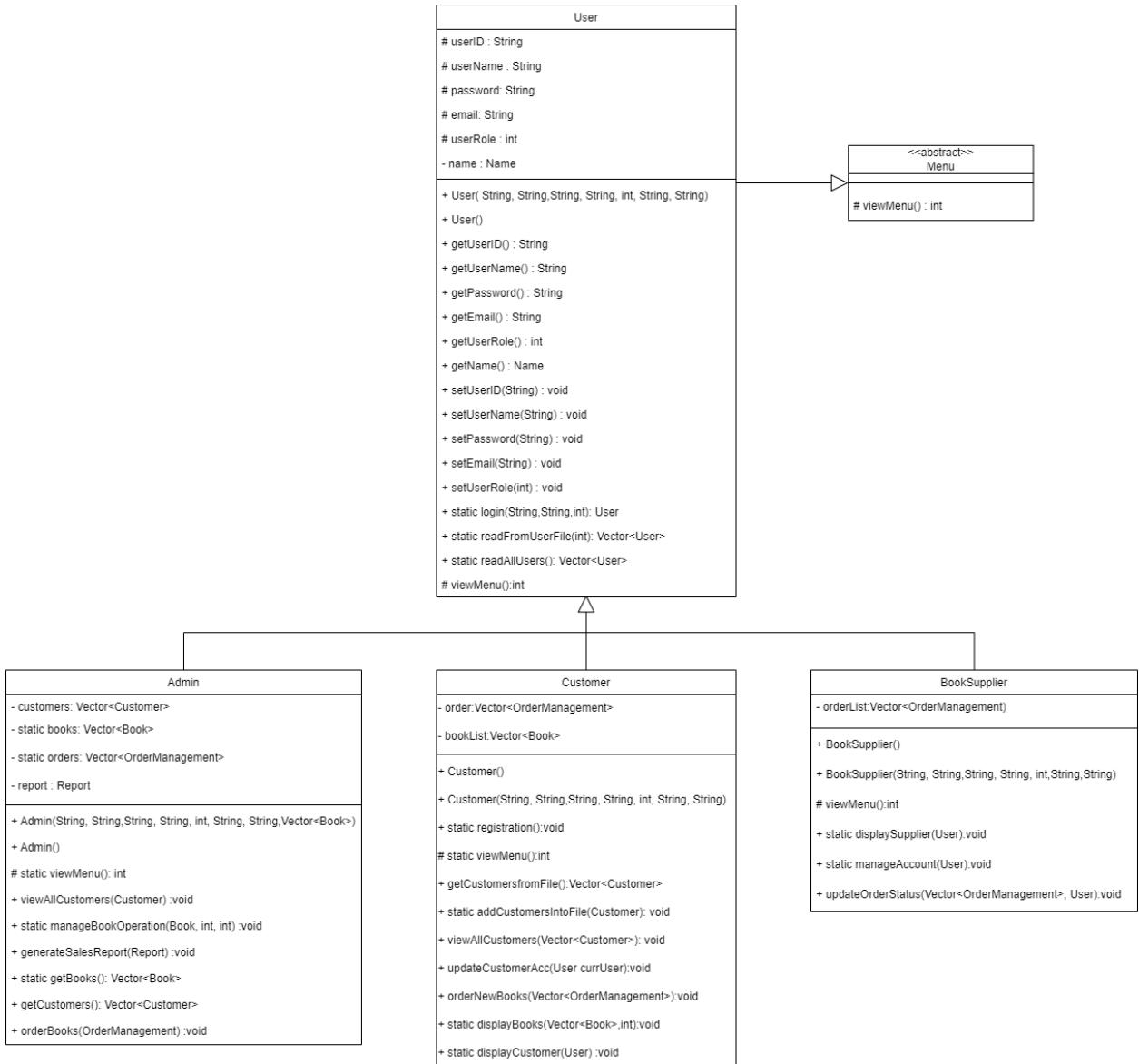


Figure 2.2: 3 subclass to user

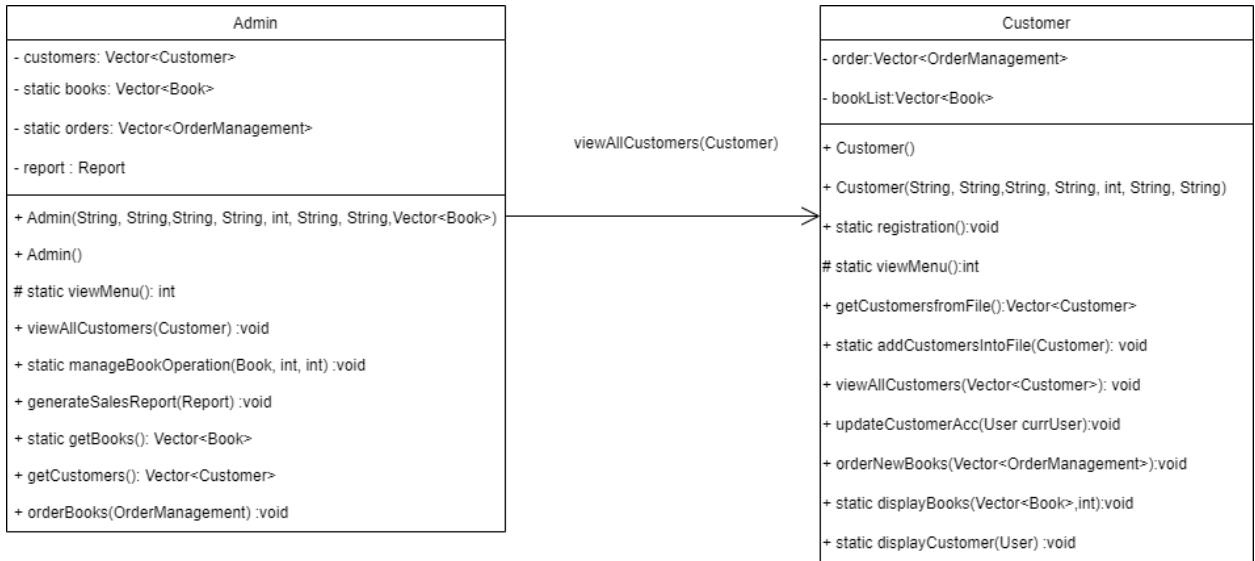


Figure 2.3 : Admin to customer

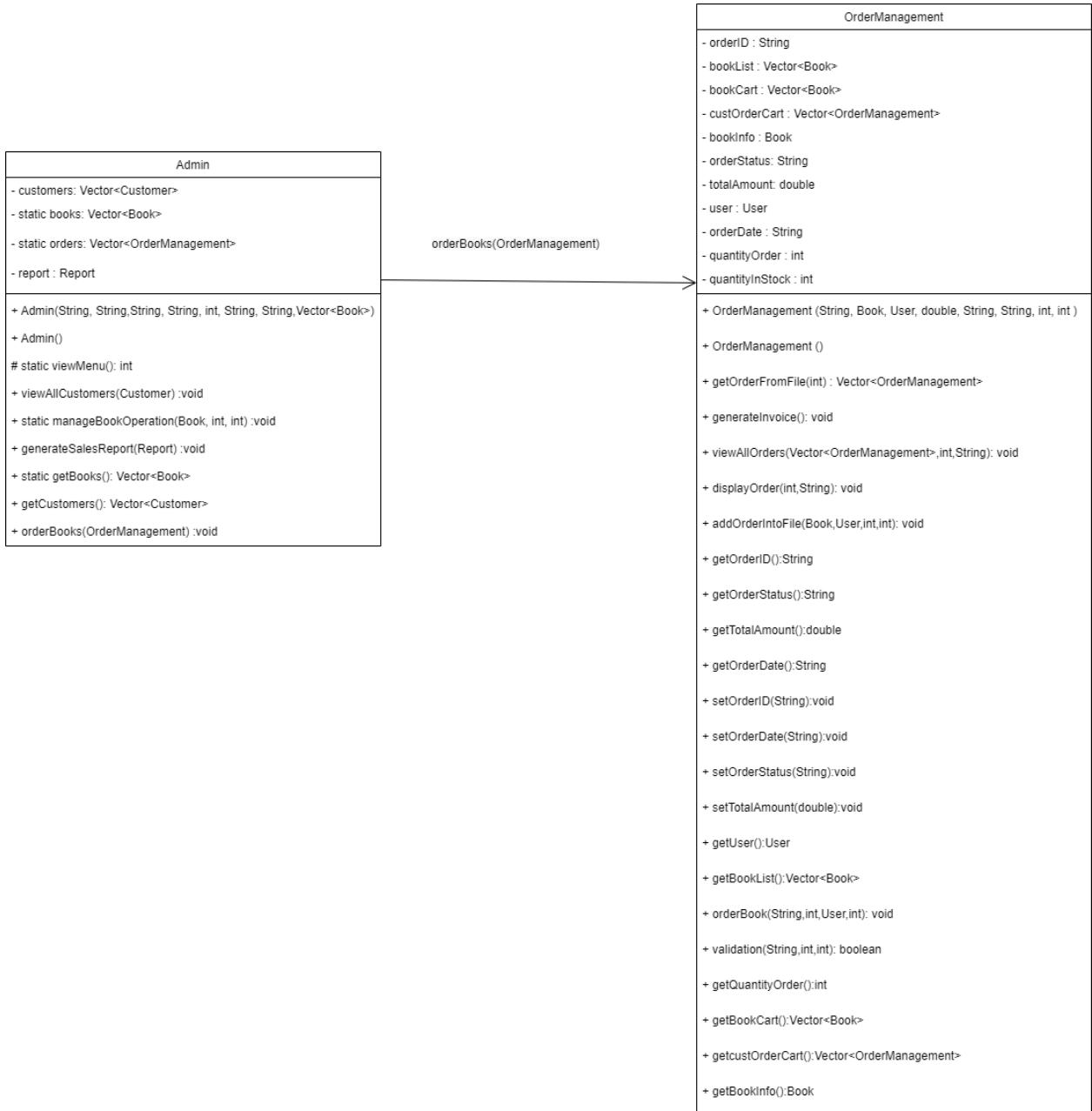


Figure 2.4 : Admin to orderManagement

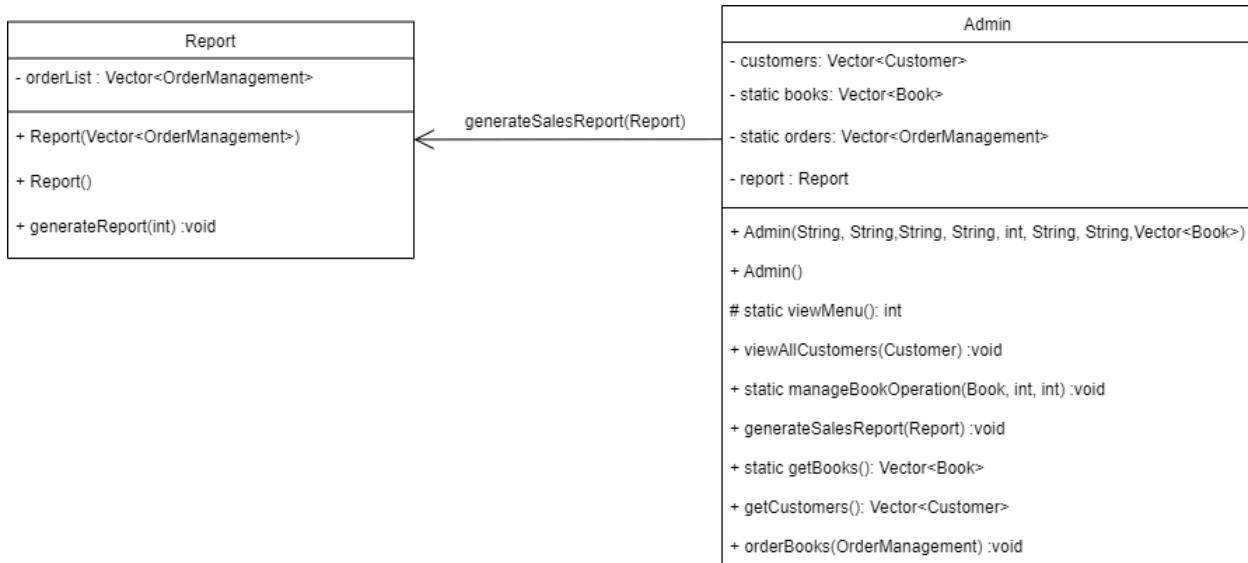


Figure 2.5 : Admin to report

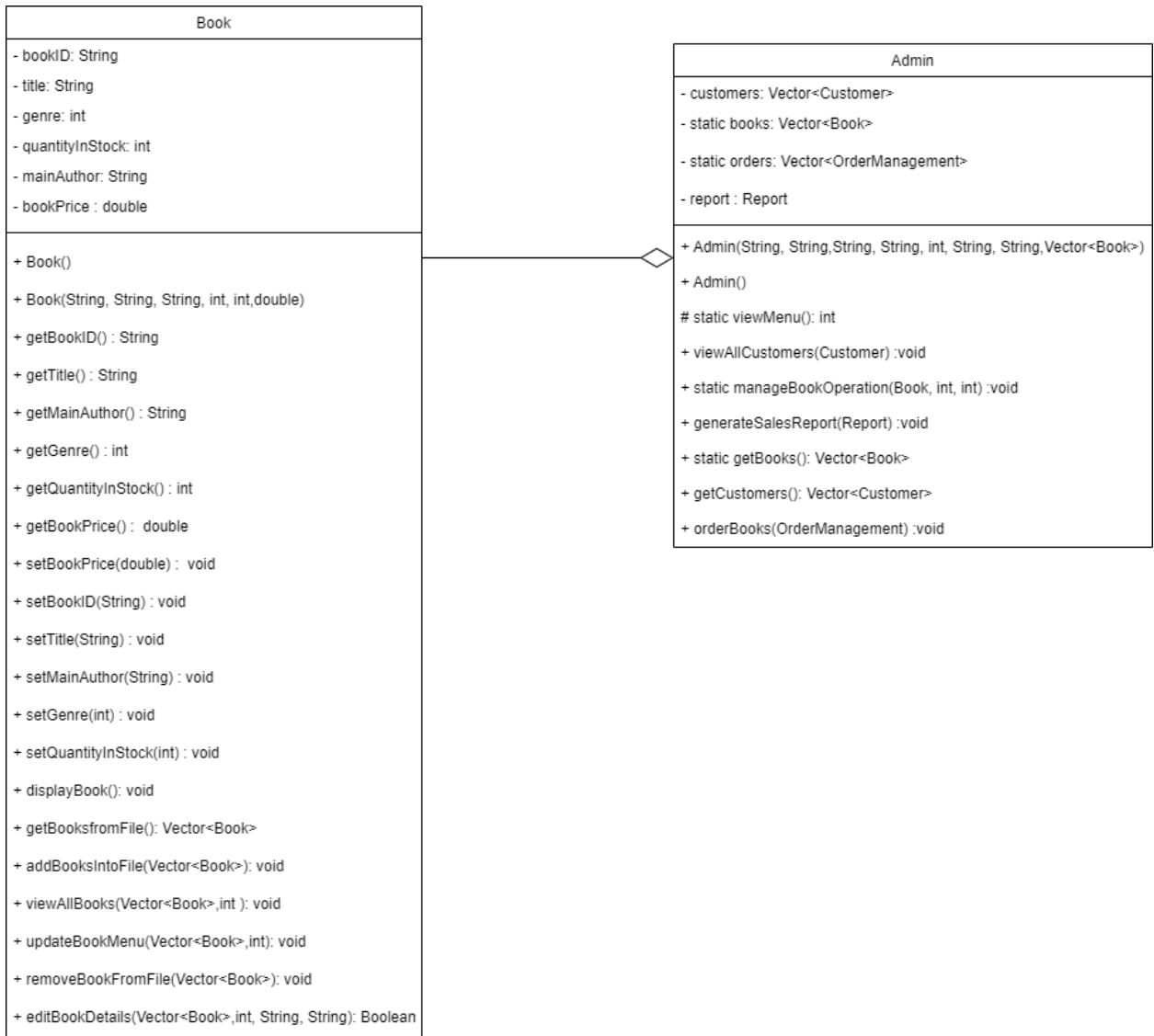


Figure 2.6 : Book to admin



Figure 2.7 : Book to orderManagement

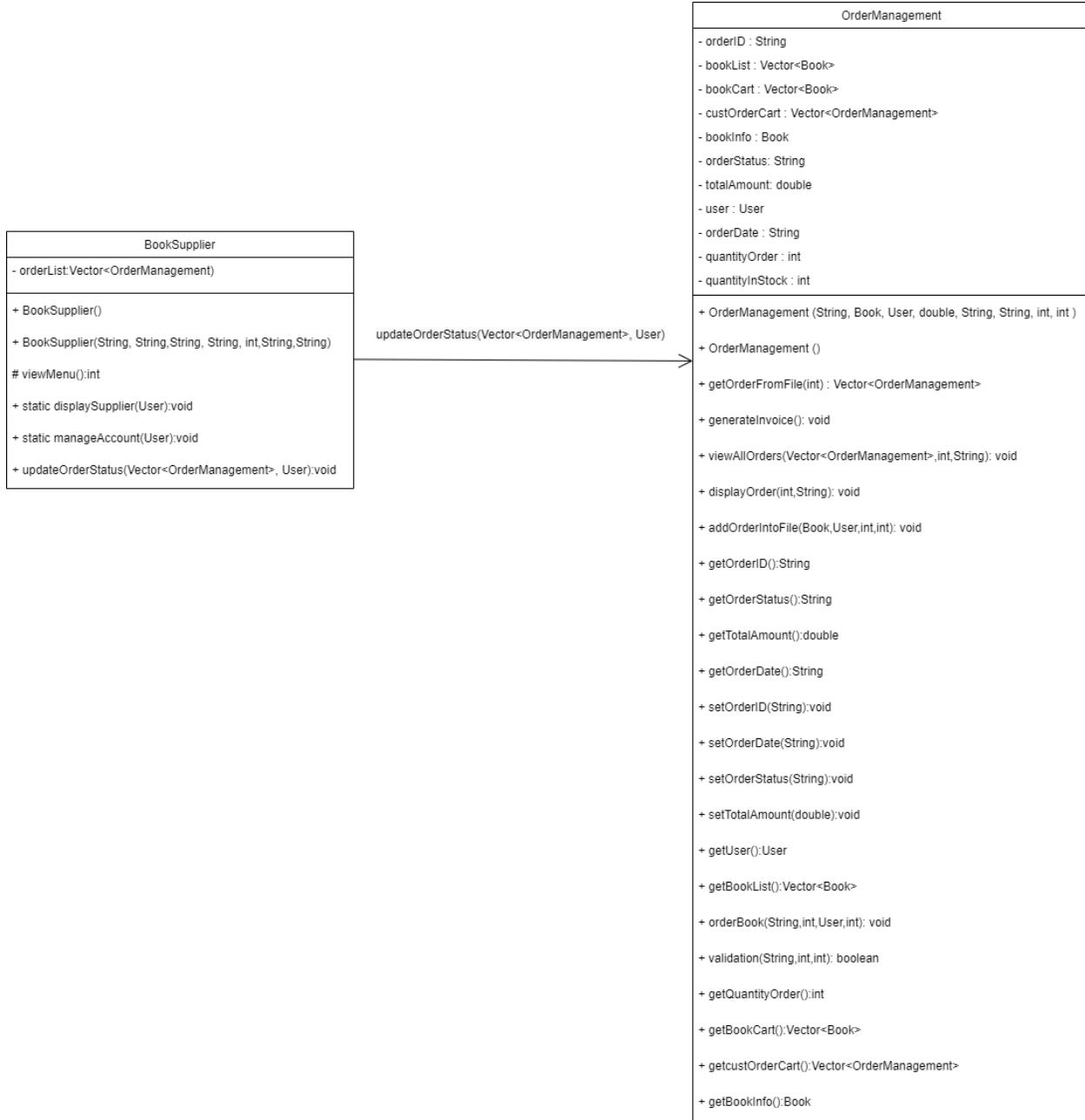


Figure 2.8 : BookSupplier to orderManagement

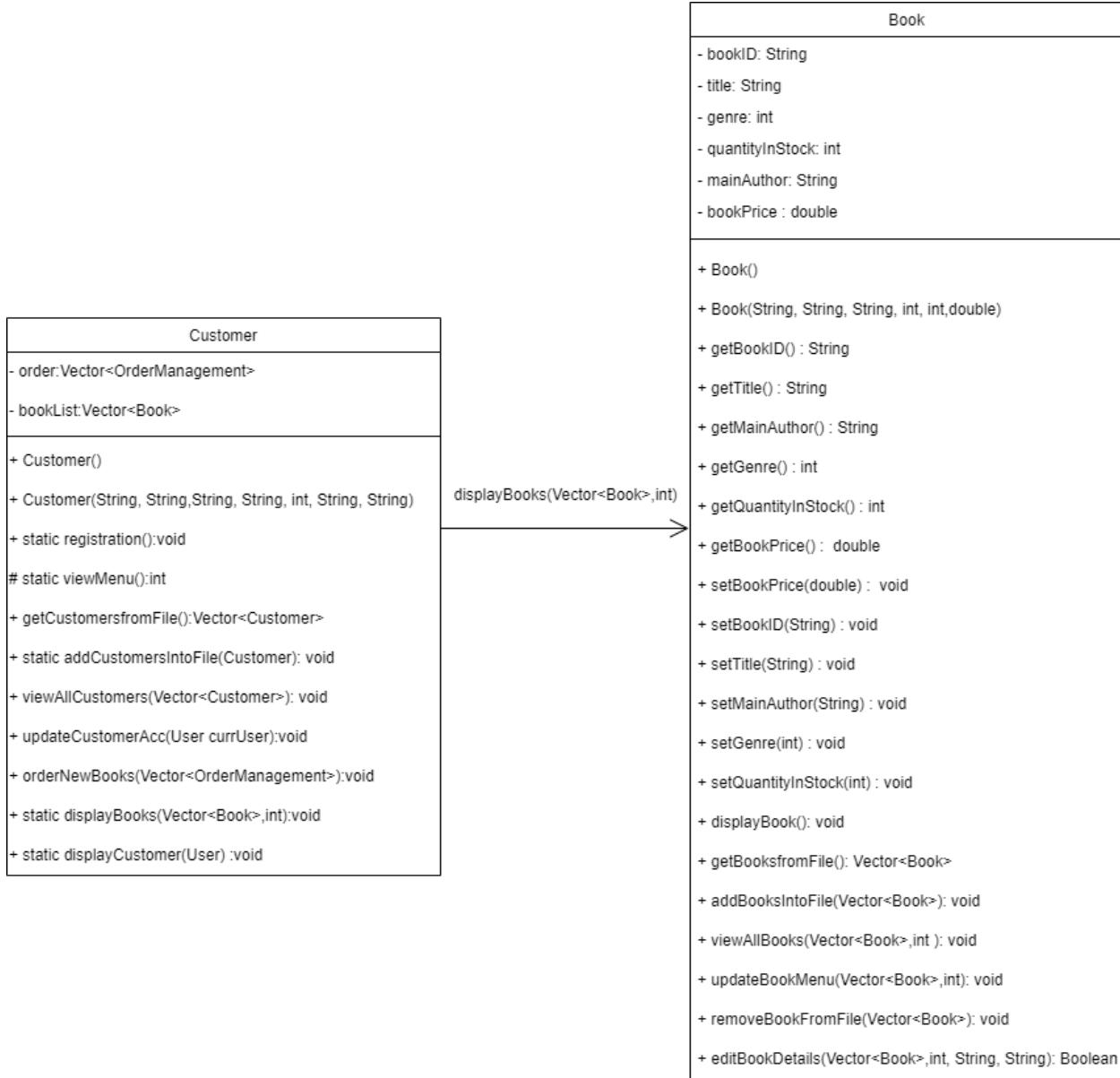


Figure 2.9 : Customer to book



Figure 2.10 : Customer to orderManagement

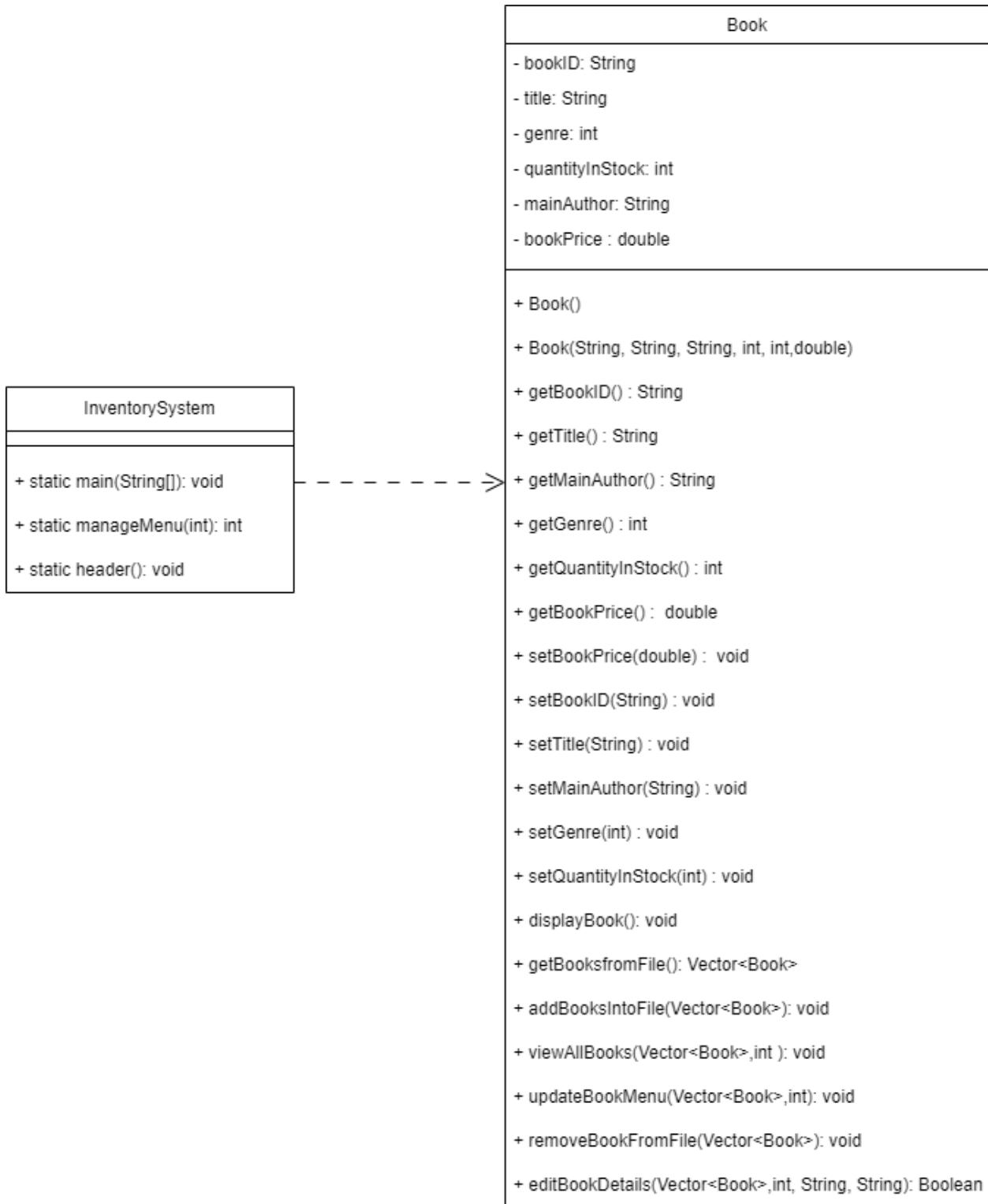


Figure 2.11 : InventorySystem to book

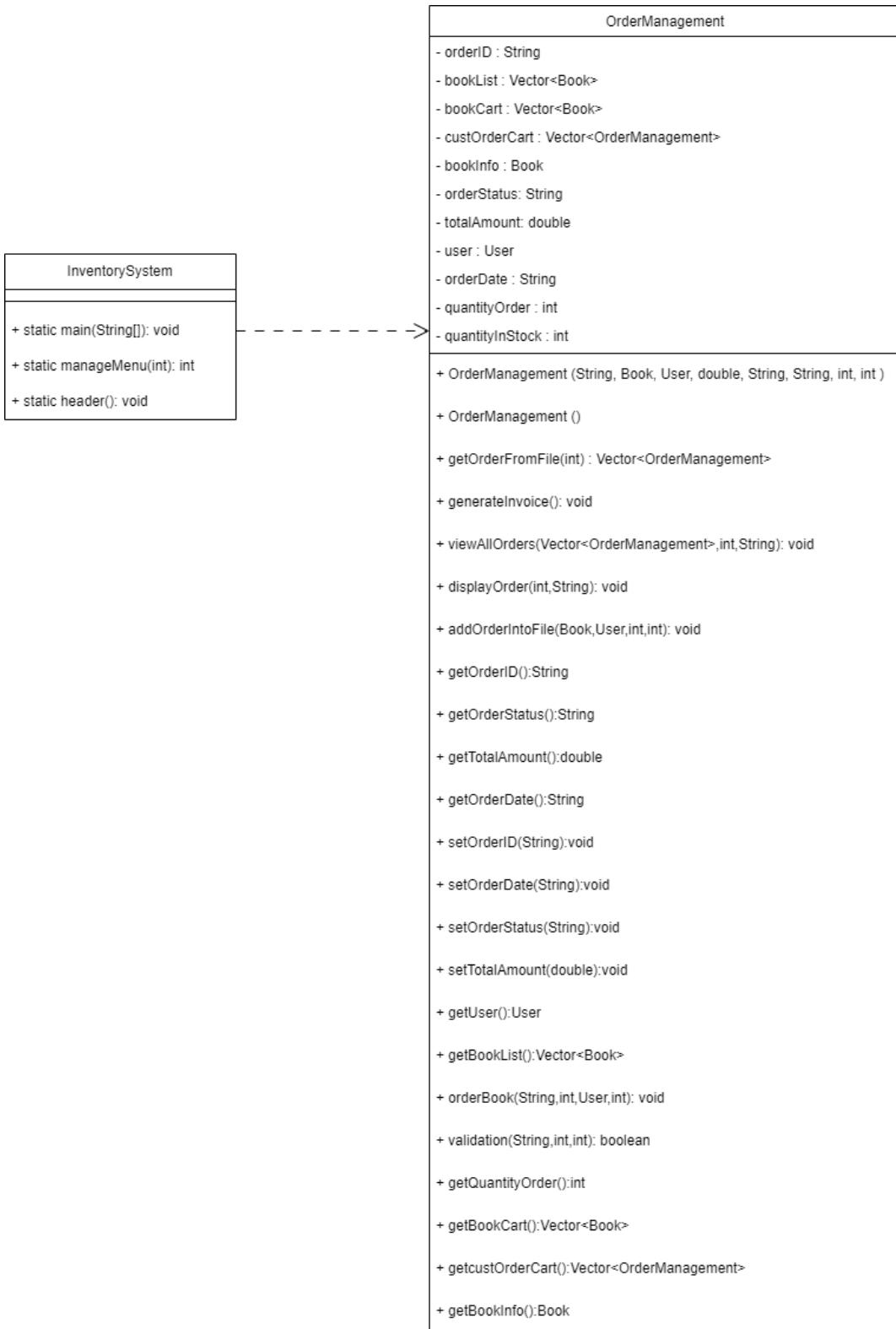


Figure 2.12 : InventorySystem to orderManagement

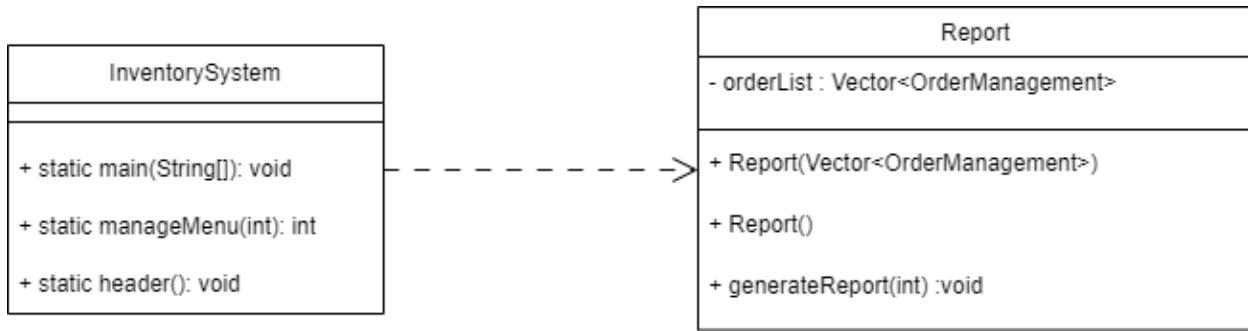


Figure 2.13 : InventorySystem to report

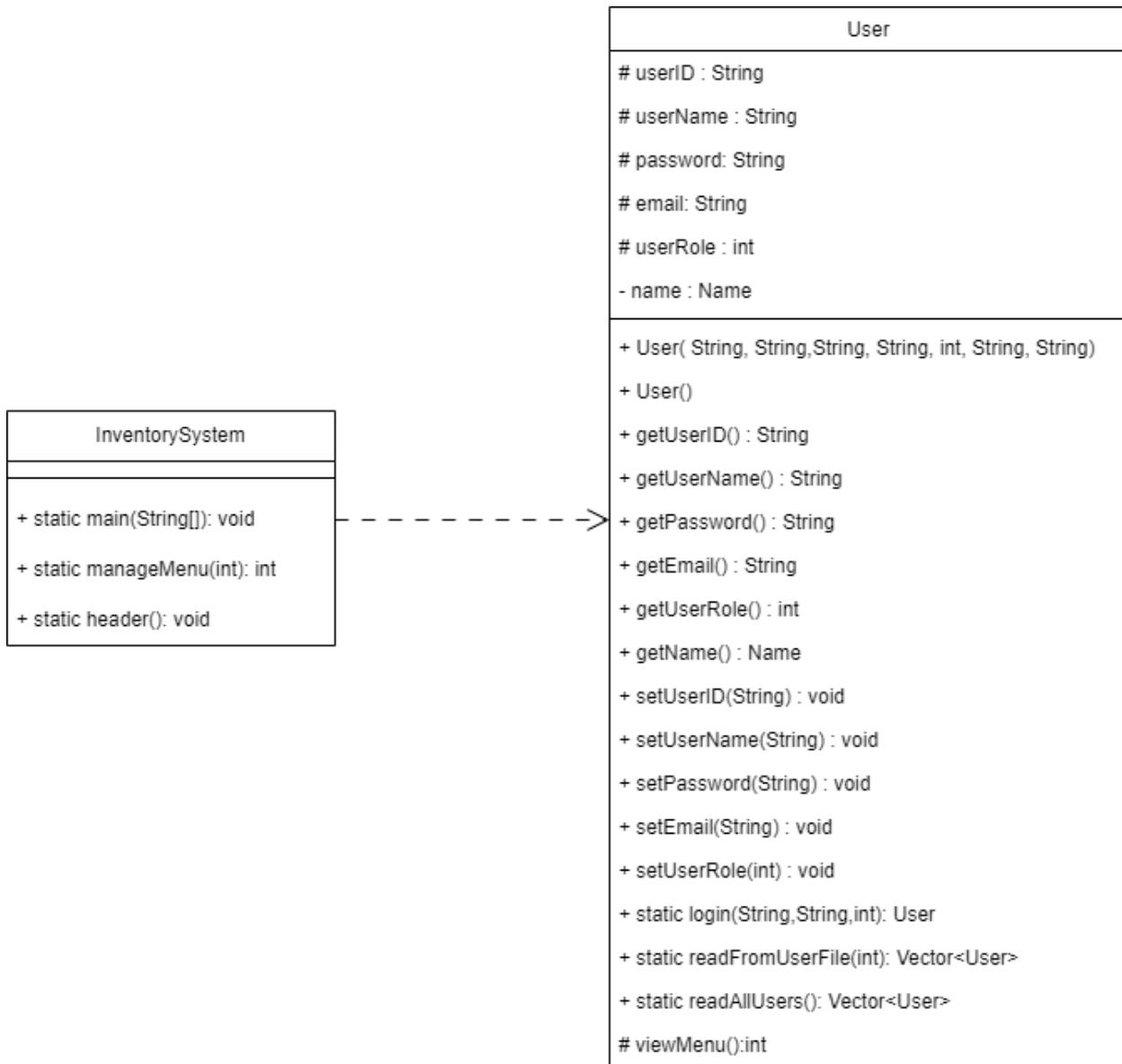


Figure 2.14 : InventorySystem to user

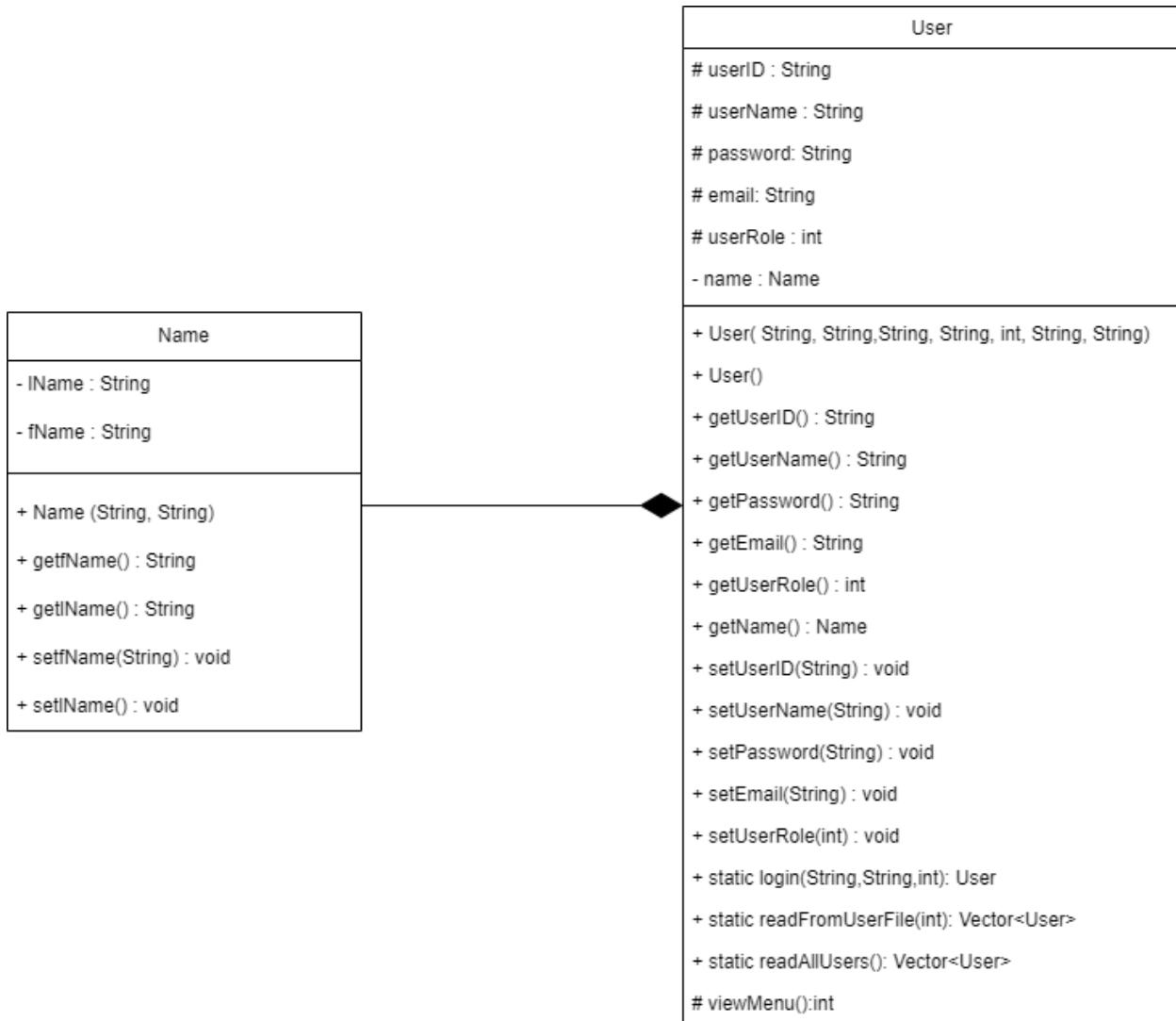


Figure 2.15 : Name to user

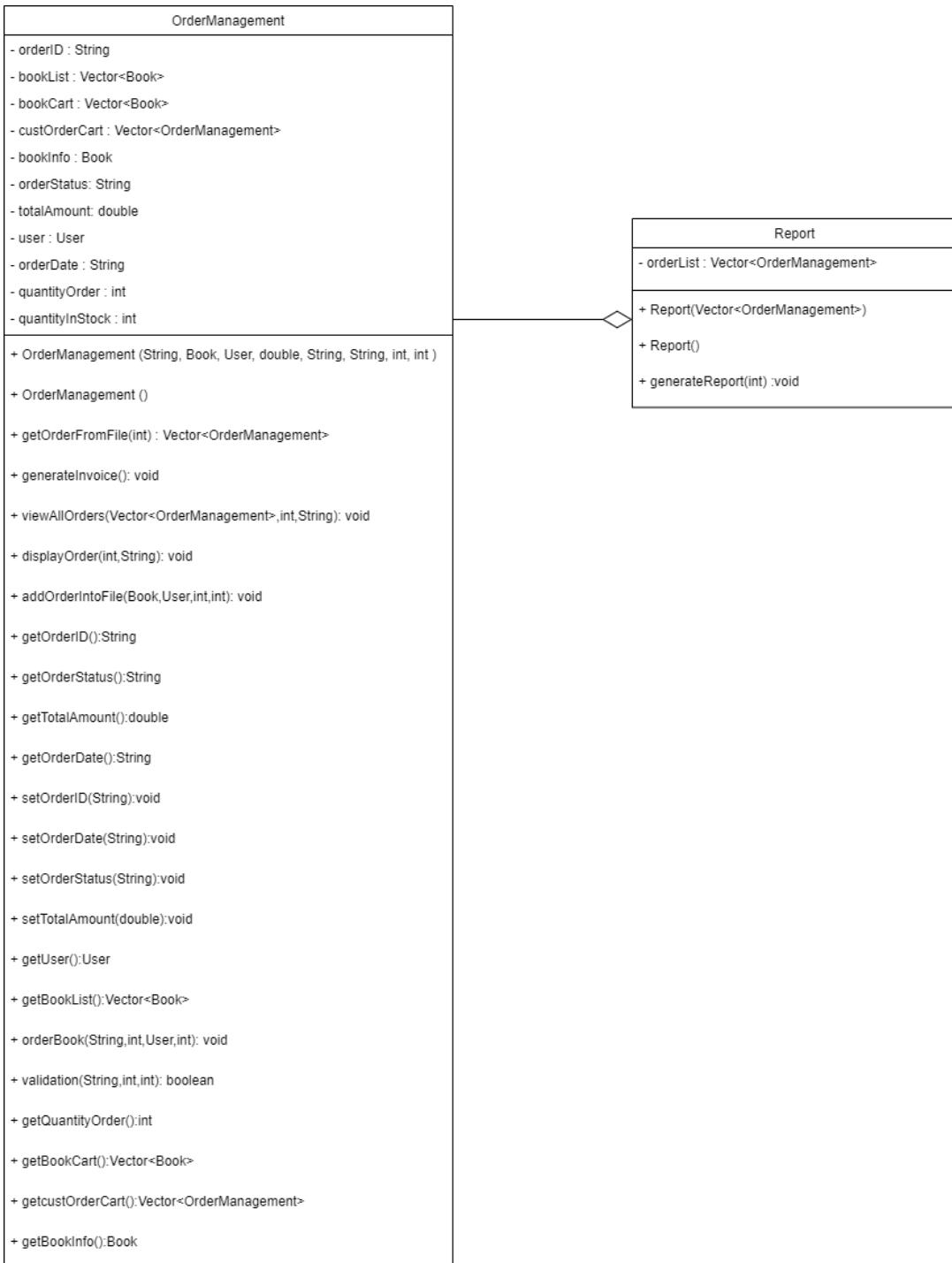


Figure 2.16 : OrderManagement to report

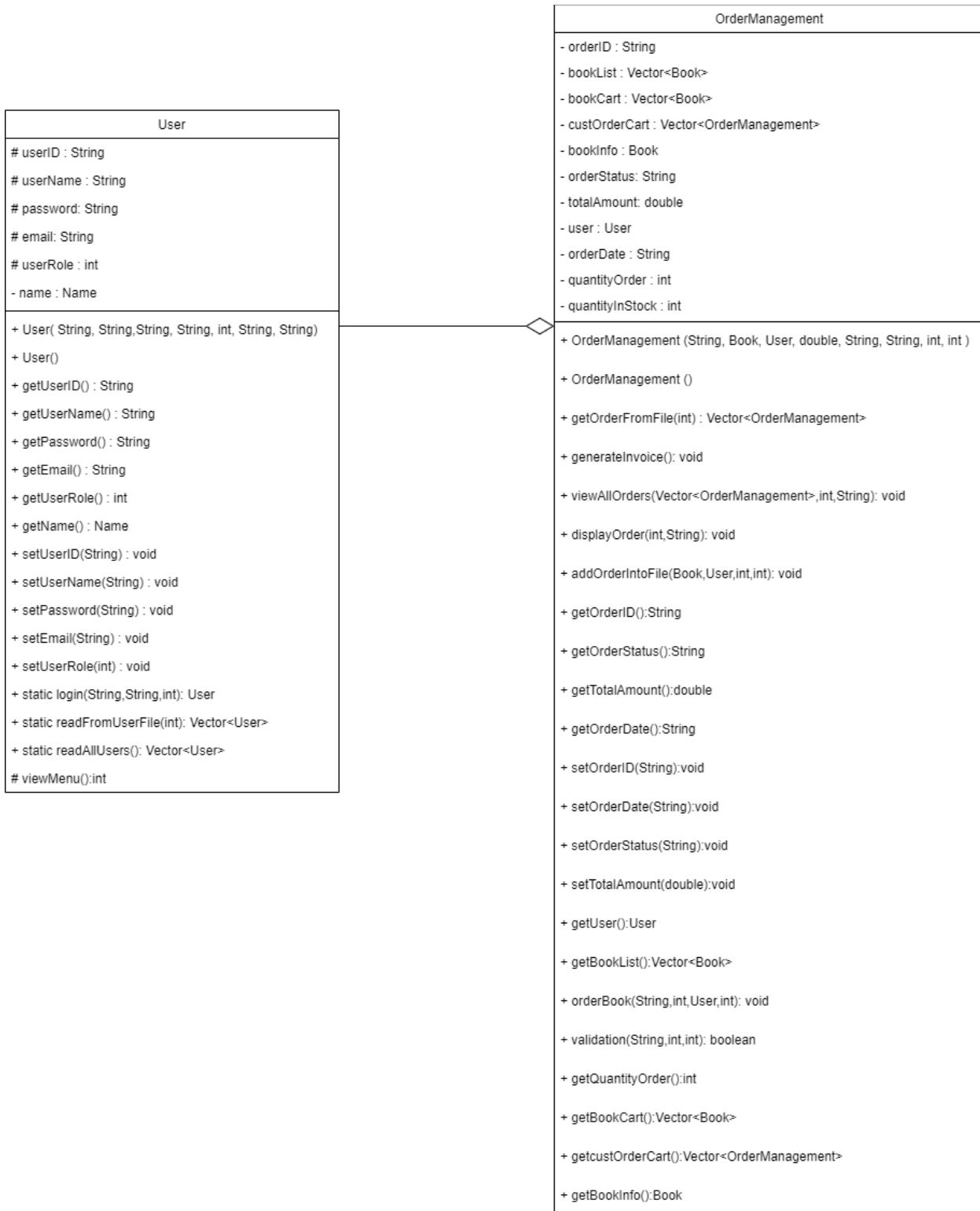


Figure 2.17 : User to orderManagement

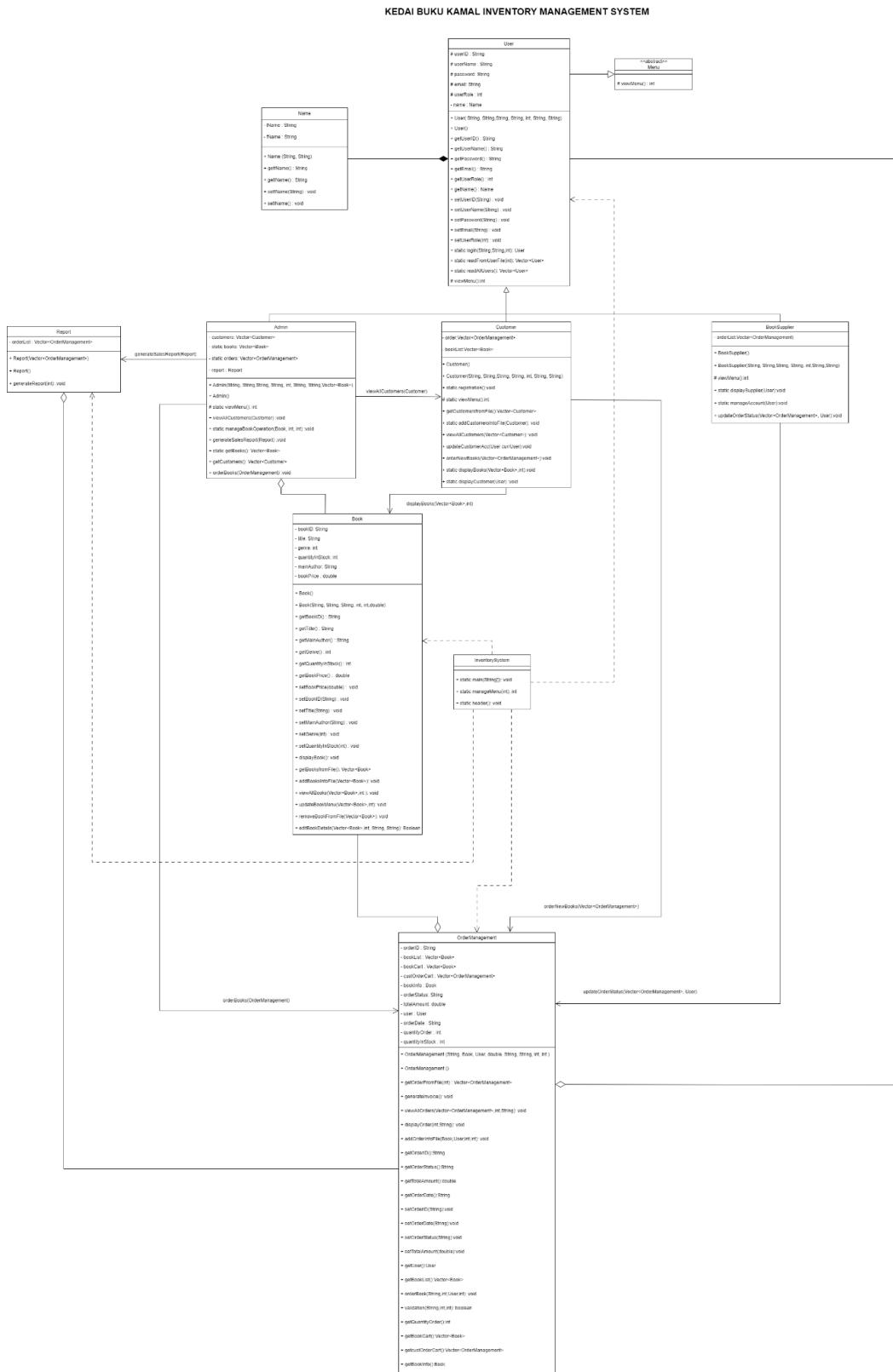


Figure 2.18 : UML class Figure

1. NAME CLASS

Attributes	Data Type	Description
fName	String	A variable to store first name.
lName	String	A variable to store last name.
Methods	Return / void	Description
Name(String, string)	void	A function used to find name.
getfName()	return String	A function used to get first name.
getlName()	return String	A function used to get last name.
setfName(String)	void	A function used to set first name.
setlName()	void	A function used to set last name.

2. USER CLASS

Attributes	Data Type	Description
userID	String	A variable to store userID.
userName	String	A variable to store userName.
password	String	A variable to store password.
email	String	A variable to store email.
userRole	int	A variable to store userRole.
name	Name	A variable to store name.
Method	Return / void	Description
User(String, String, String, String, int, String, String)	-	Constructor for initializing User with specified parameters.
User()	-	Default constructor for creating an empty User object.
getUser(String)	return String	A function used to get user.

getUserName(String)	return String	A function used to get user name.
getPassword(String)	return String	A function used to get password.
getEmail(String)	return String	A function used to get email.
getUserRole(int)	return int	A function used to get user role.
getName(Name)	return name	A function used to get name.
getUserID(String)	return String	A function used to get userID.
static login(String, String, int)	return Boolean	Static method for user login.
static readFormUserFile(int)	return Vector<User>	Static method to read user data from a file based on user ID.
static readAllUsers()	return Vector<User>	Static method to read all user data from a file.
viewMenu()	return int	Method to view the user menu and return the selected option.

3. MENU CLASS

Method	Return / void	Description
viewMenu()	return int	A function used to view menu.

4. REPORT CLASS

Attributes	Data Type	Description
orderList	Vector<OrderManagement>	A vector containing instances of OrderManagement
Method	Return / void	Description
Report(Vector<OrderManagement>)	-	Constructor that takes a vector of OrderManagement to initialize the Report.
Report	-	Default constructor for creating an empty Report object.
generateReport()	void	A function used to generate report
getOrderList()	void	A function used to get order list.

5. ADMIN CLASS

Attributes	Data Type	Description
static customers	Vector<Customer>	Static vector containing instances of Customer.
static books	Vector<Book>	Static vector containing instances of Book.
static orders	Vector<OrderManagement>	Static vector containing instances of OrderManagement.
report	Report	An instance of the Report class
Method	Return / void	Description
Admin(String, String, String, int, String, String, Vector<Book>)	-	Constructor for initializing Admin with specified parameters.
Admin()	-	Default constructor for creating an empty Admin object.
static viewMenu()	return int	Static method to view menu and return the selected option.

viewAllCustomers(Customer)	void	Displays information about a specific Customer.
static manageBookOperation(Book, int, int)	void	Static method to manage book operations
generateSalesReport(Report)	void	Generates a sales report based on the provided Report object.
orderBooks(Vector<OrderManagement>)	void	Places orders for books based on a vector of OrderManagement.

6. CUSTOMER CLASS

Attributes	Data Type	Description
order	Vector<OrderManagement>	Vector containing instances of OrderManagement.
Method	Return / void	Description
Customer()	-	Default constructor for creating an empty Customer object.
Customer(String, String, String, String, int, String, String)	-	Constructor for initializing Customer with specified parameters.
static registration()	void	Static method for customer registration.
static viewMenu()	return int	Static method to view menu and return the selected option.
getCustomersFromFile()	return Vector<Customer>	Reads and returns customer data from a file.

static addCustomersInto File(Customer)	void	Static method to add a customer to the file.
viewAllCustomers (Vector<Customer >)	void	Displays information about all customers in a vector.
updateCustomerA cc(int, String, String)	Return Boolean	Updates customer account information based on parameters.
orderNewBooks(V ector<OrderMana gement>)	void	Places new orders for books based on a vector of OrderManagement.
displayBooks(Vect or<Book>, int)	void	Displays information about books based on a vector of Book and an integer parameter.

7. BOOKSUPPLIER CLASS

Attributes	Data Type	Description
static order	Vector<OrderManagement>	Static vector containing instances of OrderManagement.
Method	Return / void	Description
BookSupplier()	-	Default constructor for creating an empty BookSupplier object.
BookSupplier(String, String, String, String, int, String, String)	-	Constructor for initializing BookSupplier with specified parameters.
static viewMenu()	return int	Static method to view menu and return the selected option.
static displaySupplier(User)	void	Static method to display information about a specific User.
static manageAccount(User)	void	Static method to manage the account of a specific User.

updateOrderStatus(Vector<OrderManagement>)	void	Updates the status of orders based on a vector of OrderManagement.
--	------	--

8. BOOK CLASS

Attributes	Data Type	Description
bookID	String	A variable to store bookID.
title	String	A variable to store title.
mainAuthor	String	A variable to store mainAuthor.
genre	int	A variable to store genre.
quantityInStock	int	A variable to store quantityInStock.
Method	Return / void	Description
Book()	-	Default constructor for creating an empty Book object.
Book(String, String, String, int, int)	-	Constructor for initializing Book with specified parameters.

getBookID()	return String	A function used to get bookID.
getTitle()	return String	A function used to get title.
getMainAuthor()	return String	A function used to get main author.
getGenre()	return int	A function used to get genre.
getQuantityInStock()	return int	A function used to get quantity in stock.
setBookID(String)	void	A function used to set bookID.
setTitle(String)	void	A function used to set title.
setMainAuthor(String)	void	A function used to set main author.
setGenre(int)	void	A function used to set genre.
setQuantityInStock(int)	void	A function used to set quantity in stock.
displayBook()	void	Displays information about the book.
getBooksFromFile()	return Vector<Book>	Reads and returns book data from a file.

addBooksIntoFile(Vector<Book>)	void	Adds books to the file.
viewAllBooks(Vec tor<Book>, int)	void	Displays information about all books in a vector.
updateBookMenu(Vector<Book>, int)	void	Updates the book menu based on a vector of Book and an integer parameter.
removeBookFrom File(Vector<Book>)	void	Removes books from the file based on a vector of Book.
editBookDetails(V ector<Book>, int, String, String)	return bool	Edits book details based on a vector of Book, an integer parameter, and new details. Returns true if successful.

9. INVENTORYSYSTEM CLASS

Method	Return / void	Description
static main(String[])	void	Main method to start the inventory system.
static header()	void	Static method to display the system header.
static manageMenu()	return int	Static method to manage the system menu and return the selected option.

10. ORDERMANAGEMENT CLASS

Attributes	Data Type	Description
orderID	String	A variable to store orderID.
bookInfo	Vector<Book>	Vector containing instances of Book representing the ordered books.
bookCart	Vector<Book>	Vector containing instances of Book representing the items in the customer's cart.
custOrderCart	Vector<OrderManagement>	Vector containing instances of OrderManagement representing the customer's order cart.
orderStatus	String	A variable to store order status.
totalAmount	double	A variable to store total amount.
user	User	Instance of the User class representing the customer who placed the order.
orderDate	String	A variable to store order date.
quantityOrder	int	A variable to store quantity order.
quantityInStock	int	A variable to store quantity in stock.

Method	Return / void	Description
OrderManagement(String, Book, User, double, String, String, int, int)	-	Constructor for initializing OrderManagement with specified parameters.
OrderManagement()	-	Default constructor for creating an empty OrderManagement object.
getOrderFromFile(int)	Return Vector<OrderManagement>	Retrieves orders from a file based on order ID.
generateInvoice()	void	A function used to generate invoice.
viewAllOrders(Vector <OrderManagement>, int, String)	void	A function used to view all orders.
displayOrder(int, String)	void	A function used to display order.
addOrderIntoFile(Book, User, int, int)	void	Adds an order to the file based on book, user, and quantity information.
getOrderID()	return String	A function used to get orderID.

getOrderStatus()	return String	A function used to get order status
getTotalAmount()	return double	A function used to get total amount.
getOrderDate()	return String	A function used to get order date
getOrderStatus()	return String	A function used to get order status.
setOrderID(String)	void	A function used to set orderID.
setOrderDate(String)	void	A function used to set order date.
setOrderStatus(String)	void	A function used to set order status.
setTotalAmount()	void	A function used to set total amount.
getUser()	return User	A function used to get user from the order.
getBookList()	return Vector<Book>	A function used to get book list.
orderBook(String, int, User, int)	return boolean	Places a new order based on book ID, quantity, user, and order ID.
validation(String, int, int)	return boolean	Validates book ID, quantity, and user information for ordering. Returns true if valid.

getQuantityOrder()	return int	A function used to get quantity order.
getBookCart()	return Vector<Book>	A function used to get book cart.
getcustOrderCart()	return Vector<OrderM anagement>	A function used to get cust order cart.
getBookInfo()	return Book	A function used to get book info.

SECTION C: SOURCE CODE AND USER MANUAL

1. User manual

- a) First, this screen shows the role to be chosen by the user, if you are an administrator, enter the number 1, if you are a customer, enter the number 2, if you are a supplier, enter the number 3, as shown in **Figure 3.1**.

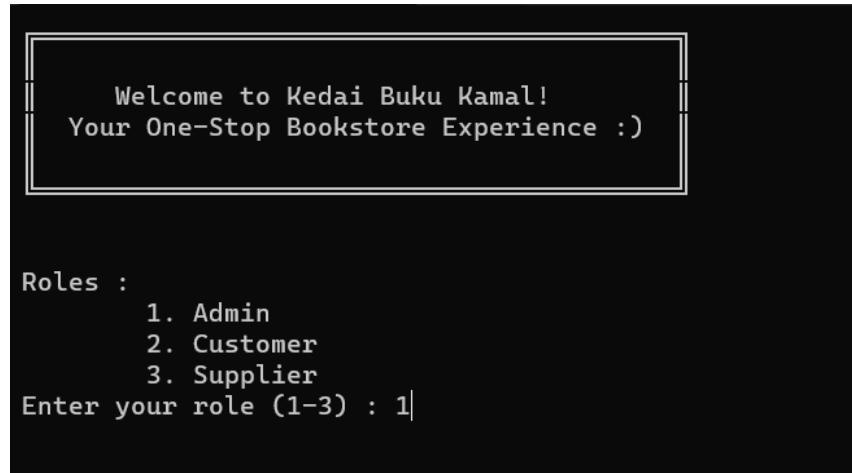


Figure 3.1 : Role choice screen

- b) After selecting your role, you will be asked to enter your username and password, as shown in **Figure 3.2**. Please make sure you enter your username and password correctly. Otherwise the credentials will be invalid.

```
Welcome to Kedai Buku Kamal!
Your One-Stop Bookstore Experience :)

Enter username : okk
Enter password : okk
Invalid Credentials Entered. Please Try Again!

Enter username : admin
Enter password : admin123
```

Figure 3.2 : Enter username and password

- c) If you are a customer and do not have any account, you can select number 2 to register a new account, as shown in **Figure 3.3**. If you already have an account, you can choose number 1 to login.

```
Welcome to Kedai Buku Kamal!
Your One-Stop Bookstore Experience :)

Customer Options:
    1. Login
    2. Register

Enter the option (1-2) : A
Invalid option entered. Please Enter an appropriate number.
Press any key to continue...

Customer Options:
    1. Login
    2. Register

Enter the option (1-2) : 3
Invalid option entered. Please Try Again!

Customer Options:
    1. Login
    2. Register

Enter the option (1-2) : 2|
```

Figure 3.3 : Customer options to login and register

- d) If you select to register an account, then it will show you this screen as shown in **Figure 3.4** to enter your username and password to register an account. After successfully register, you can continue to login as shown in **Figure 3.5**.

```
Welcome to Kedai Buku Kamal!
Your One-Stop Bookstore Experience :)

Enter your username : sar
Username already exists. Please choose a different username.

Enter a new username : Wei Yang
Enter your first name: Wei
Enter your last name: Yang
Enter your email: weiY@example.com
Enter a password: wei123
Re-enter your password: wei123

Your username : WeiYang
Your Password: wei123
Successfully Registered..You will be exited from the system to login :)
```

Figure 3.4 : Enter username and password for register

```
Welcome to Kedai Buku Kamal!
Your One-Stop Bookstore Experience :)

Customer Options:
1. Login
2. Register

Enter the option (1-2) : 1

Enter username : WeiYang

Enter password : wei12
Invalid Credentials Entered. Please Try Again!

Enter username : WeiYang
Enter password : wei123
```

Figure 3.5 : Enter username and password to login

- e) After successful login, if you are an administrator, the menu page will be displayed, as shown in **Figure 3.6**. If you are a customer, the menu page will be displayed, as shown in **Figure 3.7**. If you are a supplier, the menu screen will be displayed, as shown in **Figure 3.8**. Please make sure to enter the number if you want to select the menu.

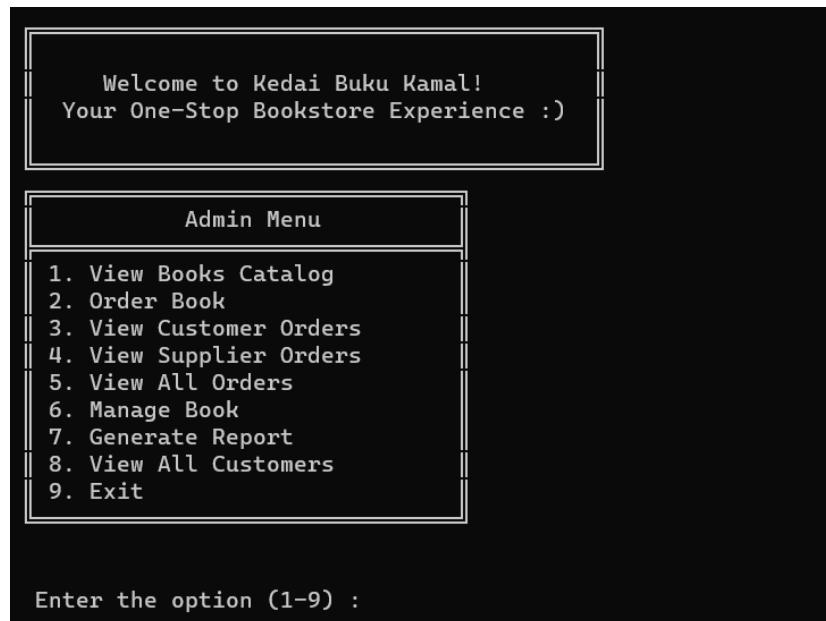


Figure 3.6 : Admin menu screen

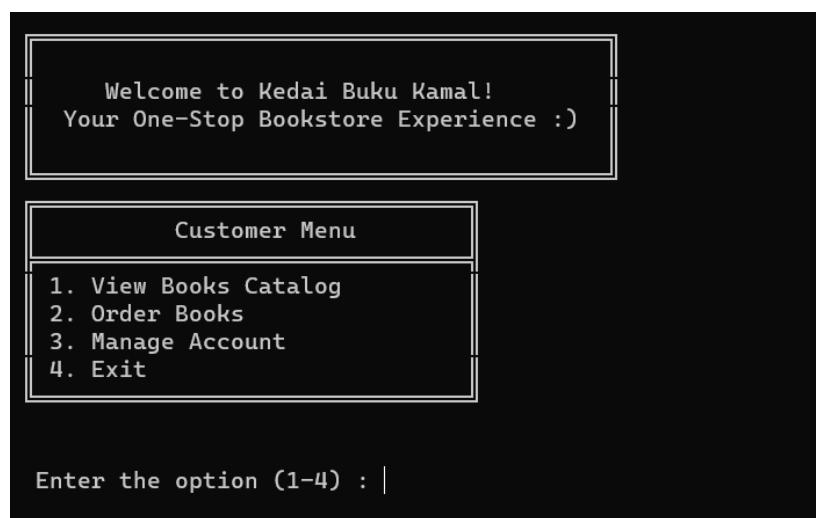


Figure 3.7 : Customer menu screen

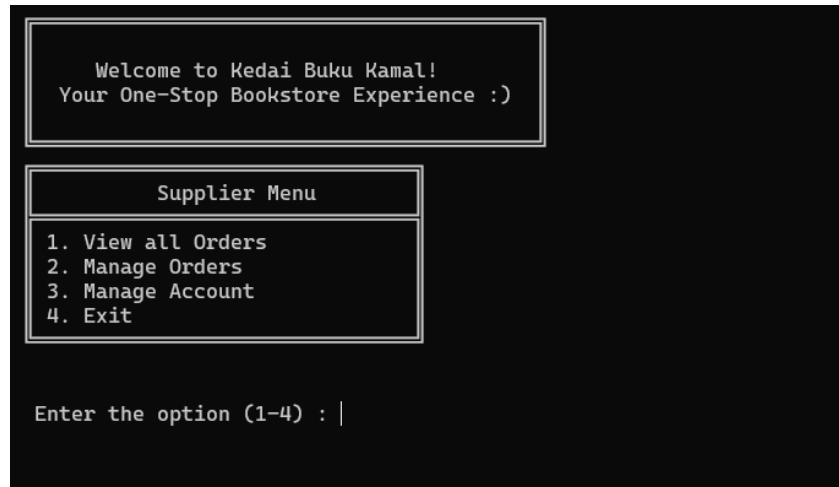


Figure 3.8 : Supplier menu screen

A. Admin User Manual

f) If you select admin menu number 1, the book catalog list will be displayed, as shown in

Figure 3.9.

Welcome to Kedai Buku Kamal!
Your One-Stop Bookstore Experience :)

Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	50	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	30	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	40	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	58	18.90
ok	ok	ok	Romance	1	12.80

Press Enter to continue...|

Figure 3.9 : View books catalog

g) If you select admin menu number 2, the order books screen will be displayed, as shown in **Figure 3.10**. Admin needs to enter a valid BookID and quantity of books that the admin wants to order. As a result, the order will be successfully sent to supplier to be approved.

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	10	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	3	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	58	18.90

Press Enter to continue...
How many books do you want to order: A
Invalid option entered. Please Enter an appropriate number.
Press any key to continue...
How many books do you want to order: -1
Please enter a non-negative number.
How many books do you want to order: 2
Enter Book ID to order : a
Book ID not exists in our database. Please Try Again!
Enter Book ID : b002
Enter book quantity : 7
Total Amount: 97.3
Order Successfully, Please wait Supplier to Approve the order.....
Enter Book ID to order : b001
Enter book quantity : 1
Total Amount: 23.9
Order Successfully, Please wait Supplier to Approve the order.....

Figure 3.10 : Orders books

h) If you select admin menu number 3, the customer orders will be displayed, as shown in

Figure 3.11.

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)									
OrderID	BookID	Title	Main Author	Genre	Order Qty	UserId	Status	TotalAmount	OrderDate
OR03	B001	The Great Gatsby	Scott Fitzgerald	Romance	10	user2D	Completed	RM406.30	2024-01-01
OR04	B002	To Kill a Mockingbird	Harper Lee	Mystery	10	user2D	Completed	RM222.40	2024-01-01
OR05	B003	1984	George Orwell	Mystery	10	user2D	Completed	RM418.50	2024-01-01
OR06	B001	The Great Gatsby	Scott Fitzgerald	Romance	1	user2D	Completed	RM23.90	2024-01-01
OR07	B002	To Kill a Mockingbird	Harper Lee	Mystery	1	user2D	Completed	RM13.90	2024-01-01
OR08	B003	1984	George Orwell	Mystery	1	user2D	Completed	RM29.90	2024-01-01
OR09	B004	The Catcher in the Rye	J.D.Salinger	Thriller	1	user2D	Completed	RM33.90	2024-01-01
OR011	B008	The Harry Potter Series	J.K. Rowling	Romance	2	hahaCust	Completed	RM37.80	2024-01-05
OR012	B008	The Harry Potter Series	J.K. Rowling	Romance	11	hahaCust	Completed	RM140.80	2024-01-05
OR020	B001	The Great Gatsby	Scott Fitzgerald	Romance	2	WeiYangCust	Completed	RM47.80	2024-01-05
OR020	B003	1984	George Orwell	Mystery	3	WeiYangCust	Completed	RM89.70	2024-01-05
OR018	B001	The Great Gatsby	Scott Fitzgerald	Romance	2	WeiYangCust	Completed	RM47.80	2024-01-05
OR018	B003	1984	George Orwell	Mystery	3	WeiYangCust	Completed	RM89.70	2024-01-05

Press Enter to continue...]

Figure 3.11 : View customer orders

i) If you select menu number 4, the supplier orders will be displayed, as shown in **Figure 3.12.**

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)									
OrderID	BookID	Title	Main Author	Genre	Order Qty	UserId	Status	TotalAmount	OrderDate
OR01	B008	The Harry Potter Series	J.K.Rowling	Romance	10	admin123	Approved	RM189.00	2024-01-01
OR02	B007	The Da Vinci	Dan Brown	Mystery	10	admin123	Rejected	RM397.10	2024-01-01
OR010	B002	To Kill a Mockingbird	Harper Lee	Mystery	2	admin123	Approved	RM27.80	2024-01-05
OR013	B002	To Kill a Mockingbird	Harper Lee	Mystery	7	admin123	Approved	RM97.30	2024-01-05
OR014	B002	To Kill a Mockingbird	Harper Lee	Mystery	7	admin123	Pending	RM97.30	2024-01-05
OR015	B001	The Great Gatsby	Scott Fitzgerald	Romance	1	admin123	Pending	RM23.90	2024-01-05
OR020	B001	The Great Gatsby	Scott Fitzgerald	Romance	1	admin123	Pending	RM23.90	2024-01-06

Press Enter to continue...|

Figure 3.12 : View supplier orders

- j) If you select menu number 5, all orders will be displayed, as shown in **Figure 3.13**.

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)									
OrderID	BookID	Title	Main Author	Genre	Order Qty	UserId	Status	TotalAmount	OrderDate
OR01	B008	The Harry Potter Series	J.K.Rowling	Romance	10	admin123	Approved	RM189.00	2024-01-01
OR02	B007	The Da Vinci	Dan Brown	Mystery	10	admin123	Rejected	RM397.10	2024-01-01
OR03	B001	The Great Gatsby	Scott Fitzgerald	Romance	10	user2D	Completed	RM406.30	2024-01-01
OR04	B002	To Kill a Mockingbird	Harper Lee	Mystery	10	user2D	Completed	RM222.40	2024-01-01
OR05	B003	1984	George Orwell	Mystery	10	user2D	Completed	RM418.50	2024-01-01
OR06	B001	The Great Gatsby	Scott Fitzgerald	Romance	1	user2D	Completed	RM23.90	2024-01-01
OR07	B002	To Kill a Mockingbird	Harper Lee	Mystery	1	user2D	Completed	RM13.90	2024-01-01
OR08	B003	1984	George Orwell	Mystery	1	user2D	Completed	RM29.90	2024-01-01
OR09	B004	The Catcher in the Rye	J.D.Salinger	Thriller	1	user2D	Completed	RM33.90	2024-01-01
OR010	B002	To Kill a Mockingbird	Harper Lee	Mystery	2	admin123	Approved	RM27.80	2024-01-05
OR011	B008	The Harry Potter Series	J.K.Rowling	Romance	2	hahaCust	Completed	RM37.80	2024-01-05
OR012	B008	The Harry Potter Series	J.K.Rowling	Romance	11	hahaCust	Completed	RM140.80	2024-01-05
OR013	B002	To Kill a Mockingbird	Harper Lee	Mystery	7	admin123	Approved	RM97.30	2024-01-05
OR014	B002	To Kill a Mockingbird	Harper Lee	Mystery	7	admin123	Pending	RM97.30	2024-01-05
OR015	B001	The Great Gatsby	Scott Fitzgerald	Romance	1	admin123	Pending	RM23.90	2024-01-05
OR020	B001	The Great Gatsby	Scott Fitzgerald	Romance	2	WeiYangCust	Completed	RM47.80	2024-01-05
OR020	B003	1984	George Orwell	Mystery	3	WeiYangCust	Completed	RM89.70	2024-01-05
OR018	B001	The Great Gatsby	Scott Fitzgerald	Romance	2	WeiYangCust	Completed	RM47.80	2024-01-05
OR018	B003	1984	George Orwell	Mystery	3	WeiYangCust	Completed	RM89.70	2024-01-05
OR020	B001	The Great Gatsby	Scott Fitzgerald	Romance	1	admin123	Pending	RM23.90	2024-01-06

Press Enter to continue...|

Figure 3.13 : View all orders

k) If you select menu number 6, the managed book menu will be displayed, as shown in

Figure 3.14.

- i) If you want to add a new book, you need to select number 1 to add a new book, as shown in **Figure 3.15**.
- ii) If you want to remove a book, you need to select number 2 to remove a book, as shown in **Figure 3.16** and **Figure 3.17**.
- iii) If you want to edit a book's details, you need to select number 3 to edit a book's details, as shown in **Figure 3.18**.

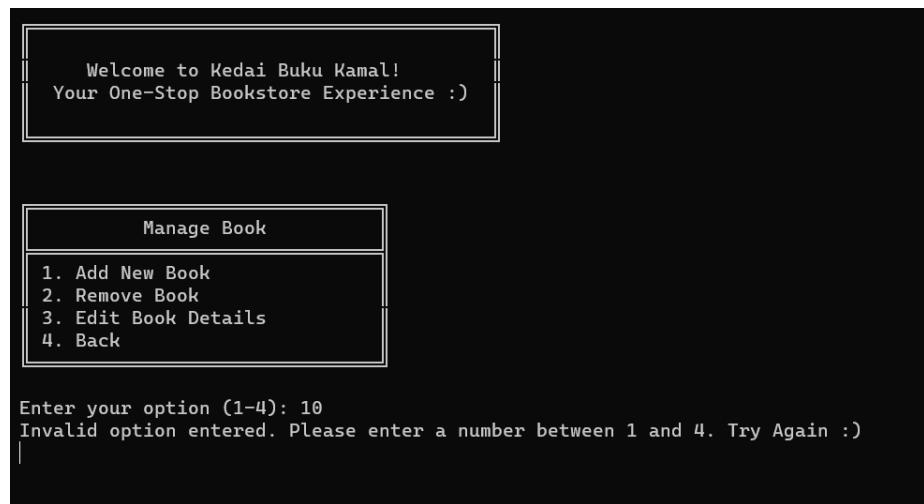


Figure 3.14 : Manage book

```

Welcome to Kedai Buku Kamal!
Your One-Stop Bookstore Experience :)

Manage Book
1. Add New Book
2. Remove Book
3. Edit Book Details
4. Back

Enter your option (1-4): 1
=====ADD BOOK=====

Enter Book ID: B001
Book ID already exists. Please choose a different Book ID.

Enter a new Book ID : B0011

Enter Title: Upin Ipin

Enter Main Author: Mohd Radzi

Enter new book genre (1-Romance , 2-Mystery , 3-Fantasy , 4-Comedy , 5-Thriller) : a
Invalid option entered. Please Enter an appropriate number.
Press any key to continue...

Enter new book genre (1-Romance , 2-Mystery , 3-Fantasy , 4-Comedy , 5-Thriller) : -1
Invalid option entered. Please enter a number between 1 and 5. Try Again :)

Enter new book genre (1-Romance , 2-Mystery , 3-Fantasy , 4-Comedy , 5-Thriller) : 3

Enter Quantity in Stock: 12

Enter Book Price (RM): 12.90

```

Figure 3.15 : Manage book (add new book)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	0	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	10	19.90
B003	1984	George Orwell	Mystery	16	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	28.90

Press Enter to continue...

Enter Book ID : B001

Figure 3.16 : Manage book (before remove book)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B002	To Kill a Mockingbird	Harper Lee	Mystery	10	19.90
B003	1984	George Orwell	Mystery	16	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90

Press Enter to continue... █

Figure 3.17 : Manage book (after remove book)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	10	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	3	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90
B008	The Harry Potter Series	J. K.Rowling	Romance	58	18.90
B0011	Upin Ipin	Mohd Radzi	Fantasy	12	12.90

Press Enter to continue... .

Update Book
1. Book Title
2. Book Main Author
3. Book Genre
4. Book Stock Quantity
5. Book Price
6. Back

Enter your option (1-6): 1
 Enter Book ID : B0
 Book ID not exists in our database. Please Try Again!

Enter Book ID : B0011
 Enter new book title : Upin & Ipin|

Figure 3.18 : Manage book (edit book details)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	10	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	3	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	58	18.90
B0011	Upin Ipin	Mohd Radzi	Fantasy	12	12.90

Press Enter to continue...

Update Book
 1. Book Title
 2. Book Main Author
 3. Book Genre
 4. Book Stock Quantity
 5. Book Price
 6. Back

Enter your option (1-6): 1
 Enter Book ID : B0
 Book ID not exists in our database. Please Try Again!

Enter Book ID : B0011
 Enter new book title : Upin & Ipin|

Figure 3.19 : Manage book (Edit book title)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	10	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	3	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	58	18.90
B0011	Upin & Ipin	Mohd Radzi	Fantasy	12	12.90

Press Enter to continue...

Update Book
 1. Book Title
 2. Book Main Author
 3. Book Genre
 4. Book Stock Quantity
 5. Book Price
 6. Back

Enter your option (1-6): 2
 Enter Book ID : B0011
 Enter new book author : Mohammad Radzi|

Figure 3.20 : Manage book (Edit book main author)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	10	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	3	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	58	18.90
B0011	Upin & Ipin	Mohammad Radzi	Fantasy	12	12.90

Press Enter to continue...

Update Book					
1. Book Title					
2. Book Main Author					
3. Book Genre					
4. Book Stock Quantity					
5. Book Price					
6. Back					

Enter your option (1-6): 3
Enter Book ID : B0011
Enter new book genre (1-Romance , 2-Mystery , 3-Fantasy , 4-Comedy , 5-Thriller) : 0
Invalid option entered. Please enter a number between 1 and 5. Try Again :)
Enter new book genre (1-Romance , 2-Mystery , 3-Fantasy , 4-Comedy , 5-Thriller) : 4

Figure 3.21 : Manage book (Edit book genre)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)					
Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	10	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	3	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	58	18.90
B0011	Upin & Ipin	Mohammad Radzi	Comedy	12	12.90

Press Enter to continue...

Update Book					
1. Book Title					
2. Book Main Author					
3. Book Genre					
4. Book Stock Quantity					
5. Book Price					
6. Back					

Enter your option (1-6): 4
Enter Book ID : B0011
Enter new book stock Quantity : 19

Figure 3.22 : Manage book (Edit book stock quantity)

Welcome to Kedai Buku Kamal!
Your One-Stop Bookstore Experience :)

Book ID	Title	Main Author	Genre	Stock Quantity	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	10	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	3	13.90
B003	1984	George Orwell	Mystery	25	29.90
B004	The Catcher in the Rye	J.D. Salinger	Thriller	19	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	35	13.90
B006	The Hobbit	J.R.R. Tolkien	Mystery	20	23.60
B007	The Da Vinci	Dan Brown	Mystery	45	26.90
B008	The Harry Potter Series	J.K. Rowling	Romance	58	18.90
B0011	Upin & Ipin	Mohammad Radzi	Comedy	19	12.90

Press Enter to continue...

Update Book

- 1. Book Title
- 2. Book Main Author
- 3. Book Genre
- 4. Book Stock Quantity
- 5. Book Price
- 6. Back

```

Enter your option (1-6): 5
Enter Book ID : B0011
Enter new book price (RM) : -1
The price is lower than zero. Please enter an appropriate price. Try Again :)
Enter new book price (RM) : A
Invalid option entered. Please Enter an appropriate number.
Press any key to continue...
Enter new book price (RM) : 13.90

```

Figure 3.23 : Manage book (Edit book price)

- l) If you select menu number 7, the manage book menu will be displayed, as shown in **Figure 3.24**.

REPORT KEDAI BUKU KAMAL		
TOP 10 BOOKS SOLD		
Book ID	Title	Quantity
B008	The Harry Potter Series	23
B002	To Kill a Mockingbird	13
B001	The Great Gatsby	11
B003	1984	11
B004	The Catcher in the Rye	1
Total Amount Gained (RM):		RM1327.40
Total Amount Ordered (RM):		RM216.80
Status: Profit Total Profit: RM1110.60		
Press any key to continue...		

Figure 3.24 : Generate report

- m) If you select menu number 8, the all customer will be displayed, as shown in **Figure 3.25**.

Customer ID				Username	Full Name	Email
abu123Cust	liewCust	sarCust		abu123	Abu Bakar William Liew Sarveish Bala	abu@example.com liew@example.com sar@example.com
Press Enter to continue...						

Figure 3.25 : View all customers

B. Customer User Manual

- n) If you select customer menu number 1, the book catalogue will be displayed, as shown in

Figure 3.26.

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)				
Book ID	Title	Main Author	Genre	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	19.90
B003	1984	George Orwell	Mystery	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	23.60
B007	The Da Vinci	Dan Brown	Mystery	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	18.90
B0011	Upin & Ipin	Mohammad Radzi	Comedy	13.90

Press Enter to continue...|

Figure 3.26 : View books catalogue

- o) If you select customer menu number 2, the order book screen will be displayed, as shown in **Figure 3.27.**

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)				
Book ID	Title	Main Author	Genre	Price (RM)
B001	The Great Gatsby	Scott Fitzgerald	Romance	23.90
B002	To Kill a Mockingbird	Harper Lee	Mystery	19.90
B003	1984	George Orwell	Mystery	29.90
B004	The Catcher in the Rye	J.D.Salinger	Thriller	33.90
B005	Pride and Prejudice	Jane Austen	Fantasy	13.90
B006	The Hobbit	J.R.R.Tolkien	Mystery	23.60
B007	The Da Vinci	Dan Brown	Mystery	20.90
B008	The Harry Potter Series	J.K.Rowling	Romance	18.90
B0011	Upin & Ipin	Mohammad Radzi	Comedy	13.90

Press Enter to continue...
How many books do you want to order: a
Invalid input. Please enter a valid integer.
How many books do you want to order: 2
Enter the book ID to order : b0
Enter book Quantity : 2
Sorry, the book id you entered is not found.
Enter the book ID to order : b001
Enter book Quantity : 2
Purchase Successfully.....

Enter the book ID to order : b003
Enter book Quantity : 3
Purchase Successfully.....

KEDAI BUKU KAMAL				
Address: No 12, Jalan Sri Sabah 25, Taman Universiti, 41200 Klang, Selangor Phone Number: 03-9973628777				
Book Name Quantity Price				
The Great Gatsby 2 47.80				
1984 3 89.70				
Total Amount to Pay (RM): RM137.50				

Figure 3.27 : Order book

- p) If you select customer menu number 3, the manage account menu will be displayed, as shown in **Figure 3.28**.
- i) After you select customer menu number 3, if you want to update username, you need to select number 1, as shown in **Figure 3.29**
 - ii) If you want to update the password, you need to select number 2, as shown in **Figure 3.30**.
 - iii) If you want to update the email, you need to select number 3, as shown in **Figure 3.31**.
 - iv) If you want to update the name, you need to select number 4, as shown in **Figure 3.32**.



Figure 3.28 : Manage account

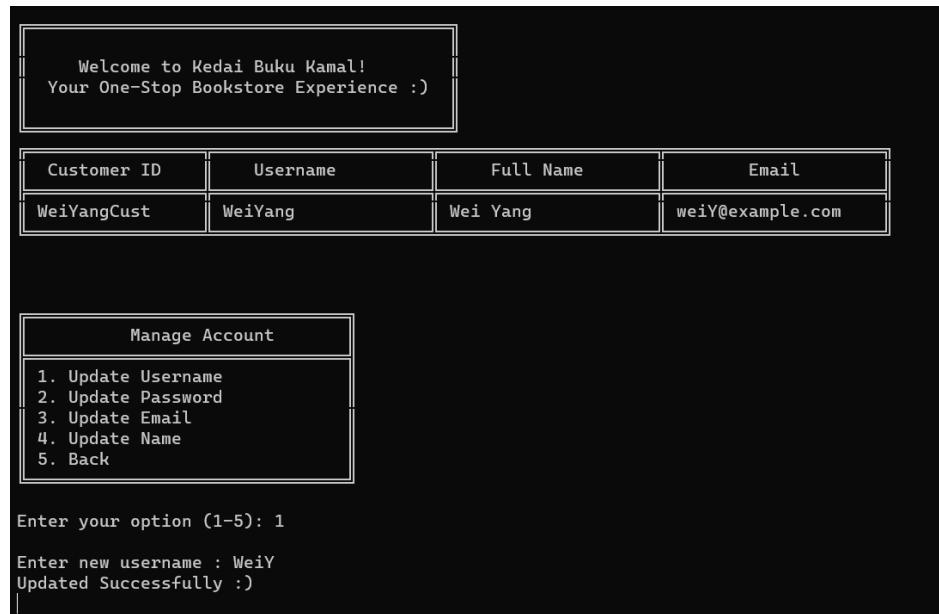


Figure 3.29 : Update username

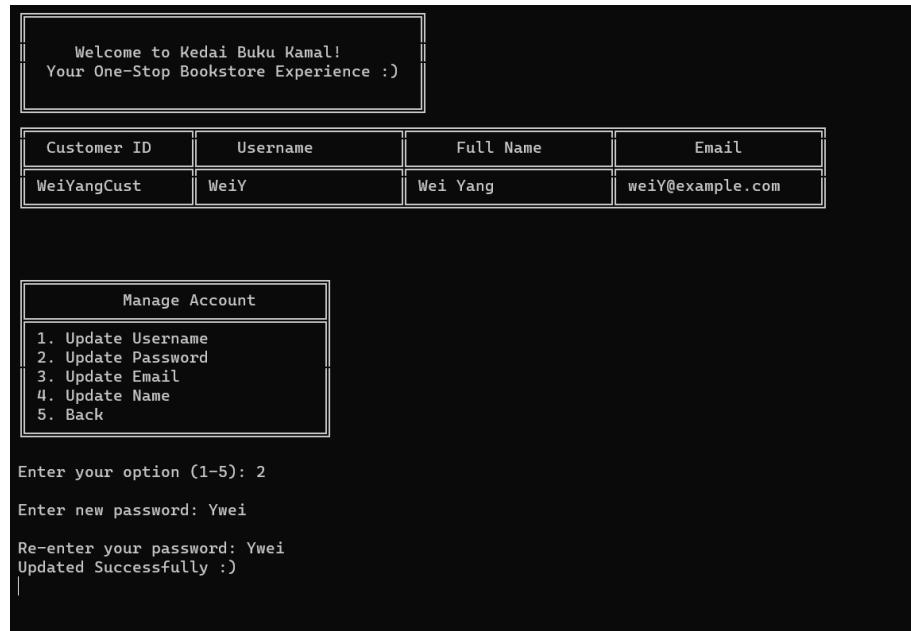


Figure 3.30 : Update password



Figure 3.31 : Update email

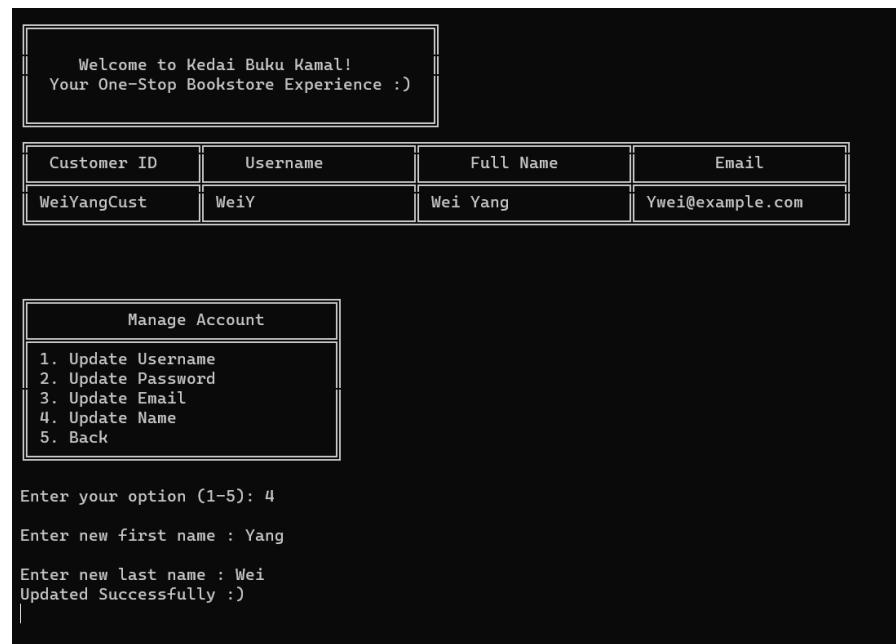


Figure 3.32 : Update name

C. Book Supplier User Manual

- q) If you select supplier menu number 1, all orders will be displayed, as shown in Figure 3.33.

OrderID	BookID	Title	Main Author	Genre	Order Qty	UserId	Status	TotalAmount	OrderDate
OR013	B002	To_Kill_a_Mockingbird	Harper_Lee	Mystery	7	admin123	Pending	RM97.30	2024-01-05
OR014	B002	To_Kill_a_Mockingbird	Harper_Lee	Mystery	7	admin123	Pending	RM97.30	2024-01-05
OR015	B001	The_Great_Gatsby	Scott_Fitzgerald	Romance	1	admin123	Pending	RM23.90	2024-01-05
Press Enter to continue...									

Figure 3.33 : View all orders

- r) If you select supplier menu number 2, manage orders will be displayed, as shown in Figure 3.34 and Figure 3.35.

OrderID	BookID	Title	Main Author	Genre	Order Qty	UserId	Status	TotalAmount	OrderDate
OR013	B002	To_Kill_a_Mockingbird	Harper_Lee	Mystery	7	admin123	Pending	RM97.30	2024-01-05
OR014	B002	To_Kill_a_Mockingbird	Harper_Lee	Mystery	7	admin123	Pending	RM97.30	2024-01-05
OR015	B001	The_Great_Gatsby	Scott_Fitzgerald	Romance	1	admin123	Pending	RM23.90	2024-01-05
Press Enter to continue...									
Enter order ID to edit : or0									
Order ID not found									
Enter order ID to edit : OR013									
Enter Status (1- Approve / 2- Reject) :a									
Invalid option entered. Please Enter an appropriate number.									
Press any key to continue...									
Enter Status (1- Approve / 2- Reject) :10									
Invalid option entered. Please enter a number between 1 and 2. Try Again :)									
Enter Status (1- Approve / 2- Reject) :1									

Figure 3.34 : Manage order (before approve)

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)									
OrderID	BookID	Title	Main Author	Genre	Order Qty	User ID	Status	Total Amount	Order Date
OR014	B002	To_Kill_a_Mockingbird	Harper_Lee	Mystery	7	admin123	Pending	RM97.30	2024-01-05
OR015	B001	The_Great_Gatsby	Scott_Fitzgerald	Romance	1	admin123	Pending	RM23.90	2024-01-05
Press Enter to continue... Enter order ID to edit :									

Figure 3.35 : Manage order (After approve)

- s) If you select supplier menu number 3, the manage account menu will be displayed, as shown in **Figure 3.36**.
- i) After you select supplier menu number 1, if you want to update username, you need to select number 1, as shown in **Figure 3.37**.
 - ii) If you want to update the password, you need to select number 2, as shown in **Figure 3.38**.
 - iii) If you want to update the email, you need to select number 3, as shown in **Figure 3.39**.
 - iv) If you want to update the name, you need to select number 4, as shown in **Figure 3.40**.

Welcome to Kedai Buku Kamal! Your One-Stop Bookstore Experience :)			
Supplier ID	Username	Full Name	Email
supplier123	SriJaya	SriJaya Publication	jayaS@example.com
Manage Account			
1. Update Username 2. Update Password 3. Update Email 4. Update Name 5. Back			
Enter your option (1-5): 1			
Enter new username : Jaya			

Figure 3.37 : Update username

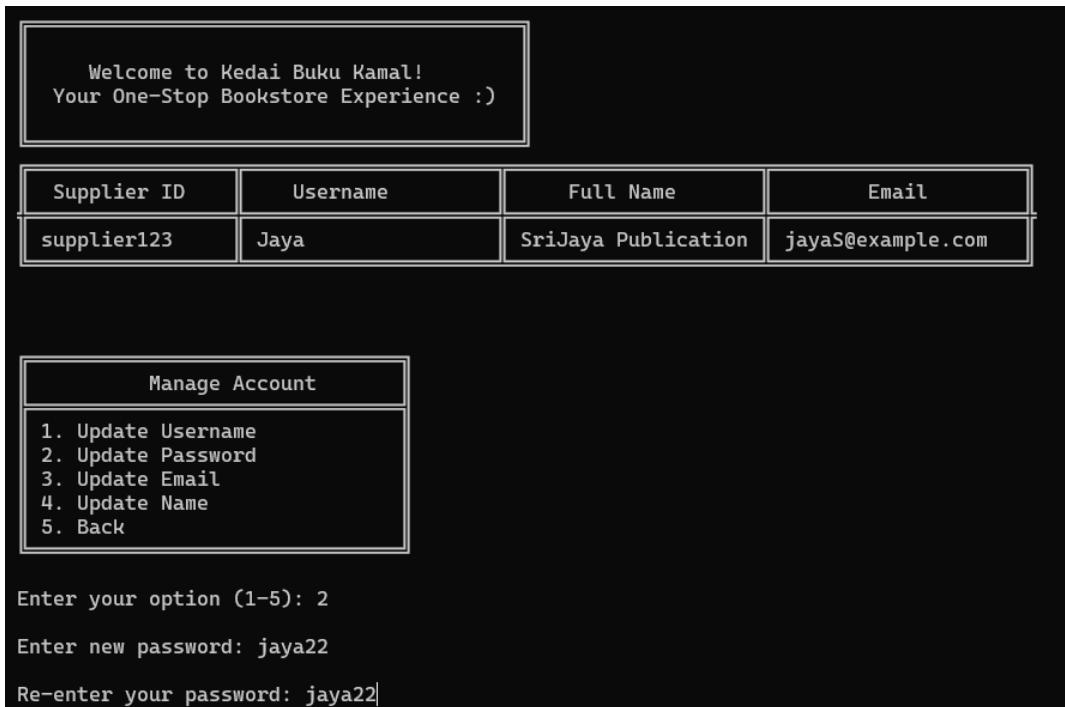


Figure 3.38 : Update password

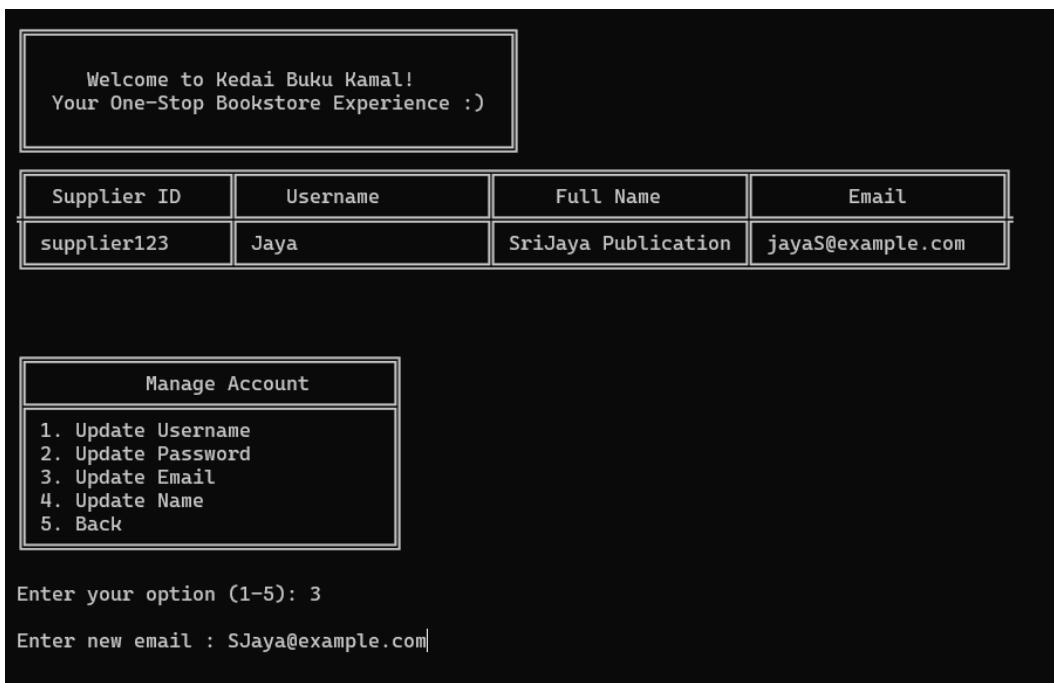


Figure 3.39 : Update email

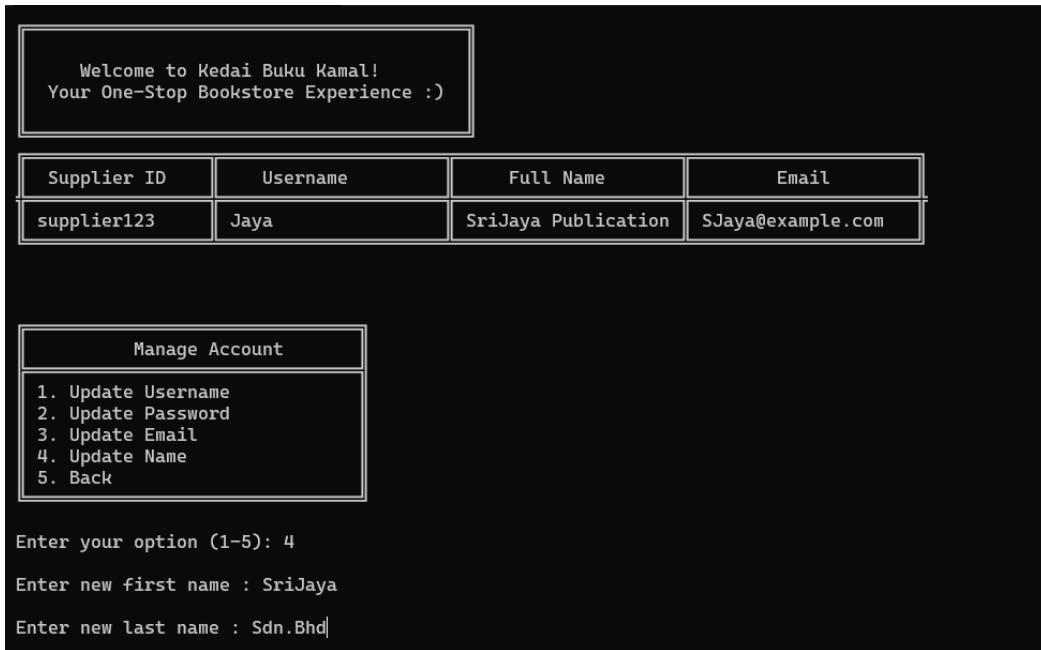


Figure 3.40 : Update name

2. Source Code

Admin.java

```
import java.io.*;
import java.util.*;

class Admin extends User {
    private Vector<Customer> customers;
    private static Vector<Book> books;
    private static Vector<OrderManagement> orderList = new Vector<OrderManagement>();
    private Report report;

    public Admin(String id, String name ,String pw, String mail, int roleID, String fName,
    String lName, Vector<Book> bks){
        super(id, name, pw, mail, roleID,fName,lName);
        customers = new Vector<Customer>();
        books = bks;
    }

    public Admin(){}

    public void viewAllCustomers(Customer c) throws FileNotFoundException{
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        customers = c.getCustomersFromFile();
        c.viewAllCustomers(customers);
    }

    public Vector<Customer> getCustomers() {
        return customers;
    }

    public static Vector<Book> getBooks() {
        return books;
    }

    public static void manageBookOperation(Book b, int option, int roleID) throws
    IOException{
        switch (option) {
            case 1:
                b.addBooksIntoFile(books);//this method is in book class
                break;
            case 2:
                b.removeBookFromFile(books);
                break;
        }
    }
}
```

```

        case 3:
            b.updateBookMenu(books,roleID);
            break;
        case 4:
            b.viewAllBooks(books,roleID);
            break;
        default:
            break;
    }
}

@Override
protected int viewMenu(){
    Scanner sc = new Scanner(System.in);
    int option = 0;
    do{
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        System.out.println(" Admin Menu || ");
        System.out.println("|| ");
        System.out.println("1. View Books Catalog || ");
        System.out.println("2. Order Book || ");
        System.out.println("3. View Customer Orders || ");
        System.out.println("4. View Supplier Orders || ");
        System.out.println("5. View All Orders || ");
        System.out.println("6. Manage Book || ");
        System.out.println("7. Generate Report || ");
        System.out.println("8. View All Customers || ");
        System.out.println("9. Exit || ");
        System.out.println("|| ");

        System.out.print("\n\n Enter the option (1-9) : ");
        try {
            option = sc.nextInt();
            if(option < 1 || option > 9){
                System.out.println("Invalid option entered. Please enter a number between 1 and
9. Try Again :)");
                sc.nextLine();
                sc.nextLine();
            }
        } catch (InputMismatchException e) {
            System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...\"");
        }
    }
}

```

```
        sc.nextLine();
        sc.nextLine();
        option = 18;
    }

}while(option < 1 || option > 9);
return option;
}

public void generateSalesReport(Report r) throws FileNotFoundException{
    report = r;
    report.generateReport(1);
}

public void orderBooks(OrderManagement o) {
    orderList.add(o);
}

}
```

Book.java

```
import java.io.*;
import java.util.*;

class Book{
    private String bookID;
    private String title;
    private String mainAuthor;
    private int genre;
    private int quantityInStock;
    private double bookPrice;

    public Book(){}
    public Book(String id, String titleBook, String author, int genreCat, int stock,double price){
        bookID = id;
        title = titleBook;
        mainAuthor = author;
        genre = genreCat;
        quantityInStock = stock;
        bookPrice = price;
    }

    public double getBookPrice() {
        return bookPrice;
    }
    public void setBookPrice(double bookPrice) {
        this.bookPrice = bookPrice;
    }

    public String getBookID() {
        return bookID;
    }
    public void setBookID(String bookID) {
        this.bookID = bookID;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getMainAuthor() {
        return mainAuthor;
    }
```

```

    }
    public void setMainAuthor(String mainAuthor) {
        this.mainAuthor = mainAuthor;
    }
    public int getGenre() {
        return genre;
    }
    public void setGenre(int genre) {
        this.genre = genre;
    }
    public int getQuantityInStock() {
        return quantityInStock;
    }
    public void setQuantityInStock(int quantityInStock) {
        this.quantityInStock = quantityInStock;
    }

    public void displayBook(int role){ //Testing since need to check with Sarveish.
        String genreStr ="";
        switch (genre) {
            case 1:
                genreStr = "Romance";
                break;
            case 2:
                genreStr = "Mystery";
                break;
            case 3:
                genreStr = "Fantasy";
                break;
            case 4:
                genreStr = "Comedy";
                break;
            case 5:
                genreStr = "Thriller";
                break;
            default:
                break;
        }
        String title = getTitle().replaceAll("_", " ");
        String author = getMainAuthor().replaceAll("_", " ");
        if(role == 1){
            System.out.printf(" %-8s %-30s %-30s %-22s %-20d %-20.2f%n",
            getBookID(), title, author, genreStr, getQuantityInStock(), getBookPrice());
        } else {
            System.out.printf(" %-8s %-30s %-30s %-22s %-20.2f%n", getBookID(), title,
            author, genreStr, getBookPrice());
        }
    }
}

```

```

        }

    public Vector<Book> getBooksFromFile() throws FileNotFoundException {
        Vector<Book> books = new Vector<Book>();
        try {
            Scanner sc = new Scanner(new
File("Submission/sec01_perdana/Group3/source_code/src/booksDatabase.txt"));

            while(sc.hasNext()){
                String bookId = sc.next();
                String title = sc.next();
                String mainAuthor = sc.next();
                int genre = sc.nextInt();
                int quantityInStock = sc.nextInt();
                double bookPrice = sc.nextDouble();
                Book book = new Book(bookId, title, mainAuthor, genre,
quantityInStock,bookPrice);
                books.add(book);
            }
        } catch (Exception e) {
            System.out.println("An error occurred during file operation..Please try again
later\nPress any key to continue..");
            Scanner scn = new Scanner(System.in);
            scn.nextLine();
        }
        return books;
    }

    public void addBooksToFile(Vector<Book> bk) throws IOException{
        Scanner scan = new Scanner(System.in);
        System.out.println("=====ADD BOOK=====");
        System.out.print("\nEnter Book ID: ");
        String id = scan.nextLine();

        Vector<Book> bkList = new Vector<Book>();
        bkList = getBooksFromFile();

        for(Book c:bkList){
            while (c.getBookID().equals(id.toUpperCase())) {
                System.out.println("Book ID already exists. Please choose a different Book ID.");
                System.out.print("\nEnter a new Book ID : ");
                id = scan.nextLine().toUpperCase();
            }
        }
    }
}

```

```

System.out.print("\nEnter Title: ");
String ttl = scan.nextLine();

System.out.print("\nEnter Main Author: ");
String mainAuthor = scan.nextLine();
int genreOption = 0;
do{
    System.out.print("\nEnter new book genre (1-Romance , 2-Mystery , 3-Fantasy , 4-
Comedy , 5-Thriller) : ");
    try {
        genreOption = scan.nextInt();
        if(genreOption<1 || genreOption>5){
            System.out.println("Invalid option entered. Please enter a number between 1 and
5. Try Again :");
        }
    } catch (InputMismatchException e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");  

        scan.nextLine();
        genreOption = 8;
    }
}while(genreOption<1 || genreOption>5);

int quantityInStock=0;
do{
    System.out.print("\nEnter Quantity in Stock: ");
    try {
        quantityInStock = scan.nextInt();
        if (quantityInStock >= 0) {
            break;
        } else {
            System.out.println("Stock quantity cannot be lower than 0. Please enter a valid
stock quantity.");
        }
    } catch (Exception e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");  

        scan.nextLine();
        scan.nextLine();
        quantityInStock = -10;
    }
}

}while(quantityInStock<=0);

```

```

double bookPrice=0.0;
do{
    System.out.print("\nEnter Book Price (RM): ");
    try {
        bookPrice = scan.nextDouble();
        if (bookPrice >= 0) {
            break;
        } else {
            System.out.println("Invalid price entered. Please try again.");
        }
    } catch (InputMismatchException e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");
        scan.nextLine();
        scan.nextLine();
        quantityInStock = -10;
    }
}while(bookPrice<=0);

Book c = new Book(id.toUpperCase(), ttl, mainAuthor, genreOption,
quantityInStock,bookPrice);
try {
    PrintWriter outputFile = new PrintWriter(new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/booksDatabase.txt",true));
    String title = c.getTitle().replaceAll(" ", "_");
    String author = c.getMainAuthor().replaceAll(" ", "_");
    outputFile.write(c.getBookID() + " " + title + " " + author + " " + c.getGenre() +
"+c.getQuantityInStock() + " +c.getBookPrice() + "\n");
    outputFile.close();
    bk.add(c);
} catch (Exception e) {
    System.out.print("An error is occurred during the file operation. The book is not
added..Please Try again later..\nPress Any Key to continue..");
    scan.nextLine();
}
}

public void viewAllBooks(Vector<Book> books,int role) throws FileNotFoundException{
    System.out.print("\033[H\033[2J");
    System.out.flush();
    InventorySystem.header();
    if(books.size() == 0){
        System.out.println("\nNo books in the database.");
    }
}

```

```

}else{
    if(role == 1){
        System.out.print(
    "|||"
    "|||"
    "|||\n");
        System.out.print(
            "||Book ID || Title || Main Author || Genre ||"
        Stock Quantity || Price (RM) ||\n");
        System.out.print(
    "|||"
    "|||"
    "|||\n");
        for(Book bk:books){
            bk.displayBook(role);
        }
        System.out.print(
    "|||"
    "|||"
    "|||\n");
    }else{
        System.out.print(
    "|||"
    "|||"
    "|||\n");
        System.out.print(
            "||Book ID || Title || Main Author || Genre ||"
        Price (RM) ||\n");
        System.out.print(
    "|||"
    "|||"
    "|||\n");
        for(Book bk:books){
            bk.displayBook(role);
        }
        System.out.print(
    "|||"
    "|||"
    "|||\n");
    }
}

```

```

    }

    System.out.print("Press Enter to continue... ");
    Scanner scan = new Scanner(System.in);
    scan.nextLine();
}

public void updateBookMenu(Vector<Book> books, int role) throws IOException{
    Scanner scan = new Scanner(System.in);
    if(books.size() != 0){
        int option=0, newOption=0;
        do {
            System.out.print("\033[H\033[2J");
            System.out.flush();
            viewAllBooks(books,role);
            System.out.println("\n\n");
            System.out.println("Update Book   ||");
            System.out.println("||");
            System.out.println("1. Book Title   ||");
            System.out.println("2. Book Main Author   ||");
            System.out.println("3. Book Genre   ||");
            System.out.println("4. Book Stock Quantity   ||");
            System.out.println("5. Book Price   ||");
            System.out.println("6. Back   ||");
            System.out.println("||");
            System.out.println("Enter your option (1-6): ");
            try {
                option = scan.nextInt();
                if(option < 1 || option >6){
                    System.out.println("Invalid option entered. Please enter a number between 1 and 6.\n Try Again :)");
                    scan.nextLine();
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid option entered. Please Enter an appropriate number.\nPress any key to continue... ");
                scan.nextLine();
                option = 10;
            }
        } while (option < 1 || option >6);
        scan.nextLine();
        if(option == 6){
            return;
        }
    }
}

```

```

        }
        System.out.print("Enter Book ID :");
        String id = scan.nextLine().toUpperCase();
        boolean validation = false;

        for (Book bk : books) {
            if (bk.getBookID().equals(id.toUpperCase())) {
                validation = true;
                break;
            }
        }

        while(validation == false){
            System.out.println("Book ID not exists in our database. Please Try Again!");
            System.out.print("\nEnter Book ID :");
            id = scan.nextLine().toUpperCase();
            for (Book bk : books) {
                if (bk.getBookID().equals(id.toUpperCase())) {
                    validation = true;
                    break;
                }
            }
        }

        switch (option) {
            case 1:
                System.out.print("Enter new book title :");
                String newVal = scan.nextLine();
                Boolean val = editBookDetails(books,option, id.toUpperCase(), newVal);
                break;
            case 2:
                System.out.print("Enter new book author :");
                newVal = scan.nextLine();
                val = editBookDetails(books,option, id, newVal);
                break;
            case 3:
                do{
                    System.out.print("Enter new book genre (1-Romance , 2-Mystery , 3-
Fantasy , 4-Comedy , 5-Thriller) :");
                    newVal = scan.nextLine();

                    try {
                        newOption = Integer.parseInt(newVal);
                        if(newOption <1 || newOption>5){
                            System.out.println("Invalid option entered. Please enter a number
between 1 and 5. Try Again :)");
                        }
                    }
                }
        }
    }
}

```

```

        }
    } catch (Exception e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");

        scan.nextLine();
        scan.nextLine();
        newOption = -10;
    }

} while(newOption <1 || newOption>5);
val = editBookDetails(books,option, id.toUpperCase(), newVal);
break;
case 4:
int temp = 0;
do {
    System.out.print("Enter new book stock Quantity : ");
    newVal = scan.nextLine();
    try {
        temp = Integer.parseInt(newVal);
        if(temp < 0){
            System.out.println("The stock quantity is lower than zero. Please enter
an appropriate quantity. Try Again :");
        }
    } catch (Exception e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");

        scan.nextLine();
        temp = -1;
    }
} while (temp <0 );
val = editBookDetails(books,option, id.toUpperCase(), newVal);
break;
case 5:
double tempDouble = 0;
do {
    System.out.print("Enter new book price (RM) : ");
    newVal = scan.nextLine();
    try {
        tempDouble = Double.parseDouble(newVal);
        if(tempDouble < 0){
            System.out.println("The price is lower than zero. Please enter an
appropriate price. Try Again :");
        }
    } catch (Exception e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");
```

```

        scan.nextLine();
        tempDouble = -1 ;
    }
} while (tempDouble <0);

val = editBookDetails(books,option, id.toUpperCase(), newVal);
break;
default:
break;
}
}else{
System.out.print("\nNo books in the database.\nPress any key to continue..");
scan.nextLine();
}

}

public void removeBookFromFile(Vector<Book> bk2) throws IOException{
Scanner scan = new Scanner(System.in);
Vector<OrderManagement> orderList = new Vector<OrderManagement>();
OrderManagement orders = new OrderManagement();
orderList = orders.getOrderFromFile(3);
if(bk2.size() != 0){
viewAllBooks(bk2,1);
String bookId = "";
int index =0;

System.out.print("\nEnter Book ID : ");
bookId = scan.nextLine().toUpperCase();
boolean validation = false;

for (Book bk : bk2) {
if (bk.getBookID().equals(bookId.toUpperCase())) {
validation = true;
break;
}
}

while(validation == false){
System.out.println("Book ID not exists in our database. Please Try Again!");
System.out.print("\nEnter Book ID : ");
bookId = scan.nextLine().toUpperCase();
for (Book bk : bk2) {
if (bk.getBookID().equals(bookId.toUpperCase())) {
validation = true;
break;
}
}
}
}
}

```

```

        }
    }
}

try {
    PrintWriter outputFile = new PrintWriter(new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/booksDatabase.txt"),false);
    PrintWriter outputOrderFile = new PrintWriter(new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/ordersDatabase.txt"),false);
    Vector<Book> updatedBookV = new Vector<Book>();
    for(Book book:bk2){
        if (!book.getBookID().equals(bookId.toUpperCase())){
            outputFile.write(book.getBookID().toUpperCase() + " " + book.getTitle() + " "
+
            book.getMainAuthor() + " " + book.getGenre() + " " +
            book.getQuantityInStock() + " " + book.getBookPrice() + "\n");
            updatedBookV.add(book);
        }
    }
    bk2 = updatedBookV;
    outputFile.close();
    Vector<OrderManagement> updatedOrderV = new Vector<OrderManagement>();
    for(OrderManagement order: orderList){
        if(!order.getBookInfo().getBookID().equals(bookId.toUpperCase())){
            outputOrderFile.write(order.getOrderID() +
"+Math.round(order.getTotalAmount() * 10) / 10.0+" +
                order.getBookInfo().getBookID() + " " + order.getQuantityOrder() +
"+order.getUser().getUserID() +
                order.getUser().getUserRole() + " " + order.getOrderStatus() +
"+order.getOrderDate() + "\n");
            updatedOrderV.add(order);
        }
    }
    orderList = updatedOrderV;
    outputOrderFile.close();
} catch (Exception e) {
    System.out.print("An error is occurred during the file operation. The book is not
removed..Please Try again later..\nPress Any Key to continue..");
    scan.nextLine();
}

} else{
    System.out.print("\nNo books in the database.\nPress any key to continue..");
    scan.nextLine();
}
}

```

```

    public Boolean editBookDetails(Vector<Book> books,int category, String id, String value)
throws IOException{
    value = value.replaceAll(" ", "_");
    for(Book book:books){
        if(book.getBookID().equals(id.toUpperCase())){
            switch (category) {
                case 1:
                    book.setTitle(value);
                    break;
                case 2:
                    book.setMainAuthor(value);
                    break;
                case 3:
                    book.setGenre(Integer.parseInt(value));
                    break;
                case 4:
                    book.setQuantityInStock(Integer.parseInt(value));
                    break;
                case 5:
                    book.setBookPrice(Double.parseDouble(value));
                    break;
                default:
                    return false;
            }
        }
    }
    if(books !=null){
        try {
            FileWriter file = new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/booksDatabase.txt",false);
            for (Book bk : books) {
                file.write(bk.getBookID().toUpperCase()+" "+bk.getTitle()+" "
"+bk.getMainAuthor()+" "+bk.getGenre()+" "+bk.getQuantityInStock()+" "
"+bk.getBookPrice()+"\n");
            }
            file.close();
        } catch (Exception e) {
            System.out.print("An error is occurred during the file operation. The book is not
removed..Please Try again later..\nPress Any Key to continue..");
            Scanner scan = new Scanner(System.in);
            scan.nextLine();
        }
    }
}

```

```
    return true;  
}  
  
}
```

BookSupplier.java:

```
//Wei Yang
import java.io.*;
import java.time.LocalDate;
import java.util.*;

class BookSupplier extends User{
    private Vector<OrderManagement> orderList;
    public BookSupplier(String id, String name ,String pw, String mail, int roleID,String fName, String lName){
        super(id, name, pw, mail, roleID,fName,lName);
    }

    public BookSupplier(){
    }

    public static void displaySupplier(User u){

        System.out.println("||-----||-----||-----||-----");
        System.out.println("|| Supplier ID || Username || Full Name || Email");
        System.out.println("||-----||-----||-----||-----");

        System.out.printf("|| %-15s || %-19s || %-19s || %-19s ||%n", u.getUserID(),
u.getUserName(), u.getName().getfName() + " " + u.getName().getlName(), u.getEmail());

        System.out.println("||-----||-----||-----||-----");
    }

    @Override
    protected int viewMenu(){
        Scanner sc = new Scanner(System.in);
        int option = 0;
        do{
            System.out.print("\033[H\033[2J");
            System.out.flush();
            InventorySystem.header();
            System.out.println("||-----||-----");
        }
    }
}
```

```

System.out.println("Supplier Menu");
System.out.println("1. View all Orders");
System.out.println("2. Manage Orders");
System.out.println("3. Manage Account");
System.out.println("4. Exit");
System.out.println("Enter the option (1-4) : ");
try {
    option = sc.nextInt();
    if(option < 1 || option > 4){
        System.out.println("Invalid option entered. Please enter a number between 1 and
4. Try Again :)");
        sc.nextLine();
        sc.nextLine();
    }
} catch (InputMismatchException e) {
    System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...\"");
    sc.nextLine();
    sc.nextLine();
    option = 10;
}
}while(option < 1 || option > 4);
return option;
}

public void updateOrderStatus(Vector<OrderManagement> orderList, User realUser)
throws IOException{
    this.orderList = orderList;
    Boolean isPending = false;
    for(OrderManagement order : orderList){
        if(order.getOrderStatus().equals("Pending")){
            isPending = true;
            break;
        }
    }
    if(isPending == true){
        OrderManagement orderApproval = new OrderManagement();
        LocalDate UpdateAt = LocalDate.now();
        Scanner sc = new Scanner(System.in);
        Book books = new Book();
        Vector<Book> bkList = new Vector<Book>();
        bkList = books.getBooksFromFile();
        String bookId = "";
    }
}

```

```

int quantityOrder = 0;
boolean checkOrderId = false;
String orderId = "";
do{
    System.out.print(" \nEnter order ID to edit : ");
    orderId = sc.nextLine().toUpperCase();
    for(OrderManagement order : orderList){
        if(order.getOrderID().equals(orderId) &&
order.getOrderStatus().equals("Pending")){
            checkOrderId = true;
        }
    }
    if(!checkOrderId){
        System.out.println("Order ID not found");
    }
}while(checkOrderId == false);
int optionApproval = 0;
do{
    try{
        System.out.print("Enter Status (1- Approve / 2- Reject) :");
        optionApproval = sc.nextInt();
        if(optionApproval < 1 || optionApproval > 2){
            System.out.println("Invalid option entered. Please enter a number between 1
and 2. Try Again :");
        }
    }catch(Exception e){
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");
        sc.nextLine();
        sc.nextLine();
        optionApproval = 10;
    }
}while(optionApproval < 1 || optionApproval > 2);
if(optionApproval == 1){
    for(OrderManagement order : orderList){
        if(order.getOrderID().equals(orderId)){
            order.setOrderStatus("Approved");
            bookId = order.getBookInfo().getBookID();
            quantityOrder = order.getQuantityOrder();
            for(Book bks : bkList){
                if(bks.getBookID().equals(bookId)){
                    bks.setQuantityInStock(bks.getQuantityInStock()+quantityOrder);
                }
            }
            System.out.println("Order Approved Successfully");
        }
    }
}

```

```

        }
    }else if(optionApproval == 2){
        for(OrderManagement order : orderList){
            if(order.getOrderID().equals(orderId)){
                order.setOrderStatus("Rejected");
                bookId = order.getBookInfo().getBookID();
                System.out.println("Order Rejected Successfully");
            }
        }
    }
    try {
        FileWriter file = new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/booksDatabase.txt",false);
        FileWriter fileOrders = new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/ordersDatabase.txt",false);
        for(Book bks : bkList){
            if(bks.getBookID().equals(bookId)){
                file.write(bks.getBookID()+" "+bks.getTitle()+" "+bks.getMainAuthor()+" "
"+bks.getGenre()+" "+bks.getQuantityInStock()+" "+bks.getBookPrice()+"\n");
            }else{
                file.write(bks.getBookID()+" "+bks.getTitle()+" "+bks.getMainAuthor()+" "
"+bks.getGenre()+" "+bks.getQuantityInStock()+" "+bks.getBookPrice()+"\n");
            }
        }
        int i =1;
        for (OrderManagement order : orderList) {
            fileOrders.write("OR0"+i+" "+order.getTotalAmount()+" "+
                order.getBookInfo().getBookID()+" "+order.getQuantityOrder()+" "+
                "+order.getUser().getUserID()+" "+order.getUser().getUserRole()+" "+
                order.getOrderStatus()+" "+order.getOrderDate()+"\n");
            i++;
        }
        fileOrders.close();
        file.close();
        for(OrderManagement order : orderList){
            System.out.println(order.getOrderID()+" "+order.getOrderStatus());
        }
    } catch (Exception e) {
        System.out.print("An error is occurred during file operation..Please try again..\nPress
Any Key to Continue..");
        Scanner scan = new Scanner(System.in);
        scan.nextLine();
    }
}
else{
    return;
}

```

```

        }

    }

public static void manageAccount(User currUser) throws IOException{
    Scanner scan = new Scanner(System.in);
    int option =0;
    String newValue = "";
    do {
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        displaySupplier(currUser);
        System.out.println("\n\n");
        System.out.println("          Manage Account      || ");
        System.out.println("          || ");
        System.out.println("          || ");
        System.out.println("          1. Update Username   || ");
        System.out.println("          2. Update Password   || ");
        System.out.println("          3. Update Email     || ");
        System.out.println("          4. Update Name      || ");
        System.out.println("          5. Back             || ");
        System.out.println("          || ");
        System.out.println("          || ");
        System.out.print("\nEnter your option (1-5): ");
        try {
            option = scan.nextInt();

            if(option < 1 || option >5){
                System.out.println("Invalid option entered. Please enter a number between 1 and
5. Try Again :)");
            }
        } catch (Exception e) {
            System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");

            scan.nextLine();
            scan.nextLine();
            option = 10;
        }
    }

} while (option < 1 || option >5);
scan.nextLine();

switch (option) {
    case 1:
        System.out.print("\nEnter new username : ");
        newValue = scan.nextLine();
}

```

```

currUser.setUserName(newValue);
break;
case 2:
System.out.print("\nEnter new password: ");
newValue= scan.nextLine();

System.out.print("\nRe-enter your password: ");
String confirmPassword = scan.nextLine();

while (newValue.equals(confirmPassword) == false) {
    System.out.println("Passwords do not match. Please re-enter your password.");
    System.out.print("\nEnter a password: ");
    newValue = scan.nextLine();

    System.out.print("\nRe-enter your password: ");
    confirmPassword = scan.nextLine();
}
currUser.setPassword(newValue);
break;
case 3:
System.out.print("\nEnter new email : ");
newValue = scan.nextLine();
currUser.setEmail(newValue);
break;
case 4:
System.out.print("\nEnter new first name : ");
newValue = scan.nextLine();
currUser.getName().setfName(newValue);
System.out.print("\nEnter new last name : ");
newValue= scan.nextLine();
currUser.getName().setlName(newValue);
break;

case 5:
return;

default:
break;
}
Vector<User> users = new Vector<User>();
users = User.readAllUsers();
try {
    FileWriter file = new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/usersDatabase.txt",false);
    for (User c : users) {

```

```

String fullName = c.getName().getfName()+"_"+c.getName().getlName();
if(c.getUserID().equals(currUser.getUserID())){
    String username = currUser.getUserName().replaceAll(" ", "");
    String em = currUser.getEmail().replaceAll(" ", "");
    fullName =
currUser.getName().getfName()+"_"+currUser.getName().getlName();
    String line = String.format("%s %s %s %s %s %d\n", currUser.getUserID(),
username, currUser.getPassword(), em,fullName, 3);
    file.write(line);
} else{
    String line = String.format("%s %s %s %s %s %d\n", c.getUserID(),
c.getUserName(), c.getPassword(), c.getEmail(),fullName, c.getUserRole());
    file.write(line);
}

}
file.close();
System.out.println("Updated Successfully :)");
} catch (Exception e) {
    System.out.print("An error is occurred during file operation..Please try again..\nPress
Any Key to Continue..");
    scan.nextLine();
}
};

}

```

Customer.java:

```
import java.io.*;
import java.util.*;

public class Customer extends User {
    private Vector<Book> bookList;
    private Vector<OrderManagement> order = new Vector<OrderManagement>();

    public Customer(){}
    public Customer(String id, String name ,String pw, String mail, int roleID, String fName,
String lName){
        super(id, name, pw, mail, roleID,fName,lName);
    }

    public static void registration() throws IOException{
        Vector<Customer> cust = new Vector<Customer>();
        cust = getCustomersFromFile();
        Scanner sc = new Scanner(System.in);
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        System.out.print("\n\nEnter your username : ");
        String username = sc.nextLine();

        for(Customer c:cust){
            while (c.getUserName().equals(username)) {
                System.out.println("Username already exists. Please choose a different username.");
                System.out.print("\nEnter a new username : ");
                username = sc.nextLine();
            }
        }

        username = username.replaceAll("\\s","");
        System.out.print("Enter your first name: ");
        String nameFirst = sc.nextLine();

        System.out.print("Enter your last name: ");
        String nameLast = sc.nextLine();

        System.out.print("Enter your email: ");
        String email = sc.nextLine();

        System.out.print("Enter a password: ");
    }
}
```

```

String pw = sc.nextLine();

System.out.print("Re-enter your password: ");
String confirmPassword = sc.nextLine();

while (pw.equals(confirmPassword) == false) {
    System.out.println("Passwords doesn't match. Please re-enter your password.");
    System.out.print("Enter a password: ");
    pw = sc.nextLine();

    System.out.print("Re-enter your password: ");
    confirmPassword = sc.nextLine();
}

String userID = username+Cust;
Customer c = new Customer(userID, username, pw, email, 2,nameFirst,nameLast);
addCustomersIntoFile(c);
sc.close();
}

@Override
protected int viewMenu(){
    Scanner sc = new Scanner(System.in);
    int option = 0;
    do{
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        System.out.println("Customer Menu || ");
        System.out.println("1. View Books Catalog || ");
        System.out.println("2. Order Books || ");
        System.out.println("3. Manage Account || ");
        System.out.println("4. Exit || ");
        System.out.println("|| ");

        System.out.print("\n\n Enter the option (1-4) : ");
        try{
            option = sc.nextInt();
            if(option < 1 || option > 4){
                System.out.println("Invalid option entered. Please enter a number between 1 and
4. Try Again :");
            }
        }catch(Exception e){
            System.out.println("Invalid option entered. Please Enter an appropriate
        }
    }
}

```

```

number.\nPress any key to continue...");
        sc.nextLine();
        sc.nextLine();
        option = 10;
    }

}while(option < 1 || option > 4);
return option;
}

public static Vector<Customer> getCustomersFromFile() throws FileNotFoundException {
    Vector<Customer> customers = new Vector<Customer>();
    Vector<User> users = User.readFromFile(2);

    for (User user : users) {
        Customer customer = new Customer(user.getUserID(), user.getUserName(),
        user.getPassword(), user.getEmail(),
        user.getUserRole(),user.getName().getfName(),user.getName().getlName());
        customers.add(customer);
    }
    return customers;
}

public static void addCustomersToFile(Customer c) throws IOException{
    try {
        PrintWriter outputFile = new PrintWriter(new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/usersDatabase.txt",true));
        String fullName = c.getName().getfName()+" "+c.getName().getlName();
        outputFile.write(c.getUserID()+" "+c.getUserName()+" "+c.getPassword()+" "
"+c.getEmail()+" "+fullName+" "+2+"\n");
        outputFile.close();

        System.out.println("\nYour username : "+c.getUserName()+"\nYour Password:
"+c.getPassword());
    } catch (Exception e) {
        System.out.print("An error is occurred during file operation..Please try again..\nPress
Any Key to Continue..");
        Scanner scan = new Scanner(System.in);
        scan.nextLine();
    }
}

public void viewAllCustomers(Vector<Customer> customers) throws
FileNotFoundException {
    if(customers.size() == 0){

```

```

        System.out.println("\nNo customers in the database.");
    }else{

        System.out.println("||-----");
        System.out.println("|| Customer ID || Username || Full Name ||");
        System.out.println("||-----");
        System.out.println("||-----");
        System.out.println("||-----");

        for (Customer u : customers) {
            System.out.printf("|| %-15s || %-19s || %-19s || %-19s ||%n", u.getUserID(),
u.getUserName(), u.getName().getfName() + " " + u.getName().getlName(), u.getEmail());
        }

        System.out.println("||-----");
        System.out.println("||-----");
    }

    System.out.print("Press Enter to continue...");
    Scanner scan = new Scanner(System.in);
    scan.nextLine();
}

public static void displayCustomer(User u){

    System.out.println("||-----");
    System.out.println("|| Customer ID || Username || Full Name ||");
    System.out.println("||-----");
    System.out.println("||-----");
    System.out.println("||-----");

    System.out.printf("|| %-15s || %-19s || %-19s || %-19s ||%n", u.getUserID(),
u.getUserName(), u.getName().getfName() + " " + u.getName().getlName(), u.getEmail());

    System.out.println("||-----");
    System.out.println("||-----");
}
}

public static void updateCustomerAcc(User currUser) throws IOException{
    Scanner scan = new Scanner(System.in);
}

```

```

int option=0;
String newValue = "";
do {
    System.out.print("\033[H\033[2J");
    System.out.flush();
    InventorySystem.header();
    displayCustomer(currUser);
    System.out.println("\n\n");
    System.out.println(" Manage Account || ");
    System.out.println(" 1. Update Username || ");
    System.out.println(" 2. Update Password || ");
    System.out.println(" 3. Update Email || ");
    System.out.println(" 4. Update Name || ");
    System.out.println(" 5. Back || ");
    System.out.println("|| ");
    System.out.print("\nEnter your option (1-5): ");
    try {
        option = scan.nextInt();
        if(option < 1 || option >5){
            System.out.println("Invalid option entered. Please enter a number between 1 and
5. Try Again :)");
            System.out.print("Press any key to continue..");
            scan.nextLine();
            scan.nextLine();
        }
    } catch (InputMismatchException e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue...");
        scan.nextLine();
        scan.nextLine();
        option = 10;
    }
} while (option < 1 || option >5);
scan.nextLine();

switch (option) {
    case 1:
        System.out.print("\nEnter new username : ");
        newValue = scan.nextLine();
        currUser.setUserName(newValue);
        break;
    case 2:
        System.out.print("\nEnter new password: ");

```

```

newValue= scan.nextLine();

System.out.print("\nRe-enter your password: ");
String confirmPassword = scan.nextLine();

while (newValue.equals(confirmPassword) == false) {
    System.out.println("Passwords do not match. Please re-enter your password.");
    System.out.print("\nEnter a password: ");
    newValue = scan.nextLine();

    System.out.print("\nRe-enter your password: ");
    confirmPassword = scan.nextLine();
}
currUser.setPassword(newValue);
break;
case 3:
    System.out.print("\nEnter new email : ");
    newValue = scan.nextLine();
    currUser.setEmail(newValue);
    break;
case 4:
    System.out.print("\nEnter new first name : ");
    newValue = scan.nextLine();
    currUser.getName().setfName(newValue);
    System.out.print("\nEnter new last name : ");
    newValue= scan.nextLine();
    currUser.getName().setlName(newValue);
    break;

case 5:
    return;

default:
    break;
}
Vector<User> us = new Vector<User>();
us = User.readAllUsers();
try {
    FileWriter file = new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/usersDatabase.txt",false);
    for (User c : us) {
        String fullName = c.getName().getfName()+"_"+c.getName().getlName();
        if(c.getUserID().equals(currUser.getUserID())){
            String username = currUser.getUserName().replaceAll(" ", "");
            String em = currUser.getEmail().replaceAll(" ", "");

```

```

        fullName =
currUser.getName().getfName()+"_" +currUser.getName().getlName();
        String line = String.format("%s %s %s %s %s %d%n", currUser.getUserID(),
username, currUser.getPassword(), em,fullName, 2);
        file.write(line);
    }else{
        String line = String.format("%s %s %s %s %s %d%n", c.getUserID(),
c.getUserName(), c.getPassword(), c.getEmail(), fullName, c.getUserRole());
        file.write(line);
    }

}
file.close();
System.out.println("Updated Successfully.");
} catch (Exception e) {
    System.out.print("An error is occurred during file operation..Please try again.\nPress
Any Key to Continue..");
    scan.nextLine();
}

}

public void orderNewBooks(OrderManagement o){
    order.add(o);
}

public void displayBooks(Vector<Book>b,int role) throws FileNotFoundException{
    bookList = b;
    bookList.get(0).viewAllBooks(bookList,2);
}
}

```

InventorySystem.java:

```
import java.io.*;
import java.util.*;

public class Customer extends User {
    private Vector<Book> bookList;
    private Vector<OrderManagement> order = new Vector<OrderManagement>();

    public Customer(){}
    public Customer(String id, String name ,String pw, String mail, int roleID, String fName,
String lName){
        super(id, name, pw, mail, roleID,fName,lName);
    }

    public static void registration() throws IOException{
        Vector<Customer> cust = new Vector<Customer>();
        cust = getCustomersFromFile();
        Scanner sc = new Scanner(System.in);
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        System.out.print("\n\nEnter your username : ");
        String username = sc.nextLine();

        for(Customer c:cust){
            while (c.getUserName().equals(username)) {
                System.out.println("Username already exists. Please choose a different username.");
                System.out.print("\nEnter a new username : ");
                username = sc.nextLine();
            }
        }

        username = username.replaceAll("\\s","");
        System.out.print("Enter your first name: ");
        String nameFirst = sc.nextLine();

        System.out.print("Enter your last name: ");
        String nameLast = sc.nextLine();

        System.out.print("Enter your email: ");
        String email = sc.nextLine();

        System.out.print("Enter a password: ");
    }
}
```

```

String pw = sc.nextLine();

System.out.print("Re-enter your password: ");
String confirmPassword = sc.nextLine();

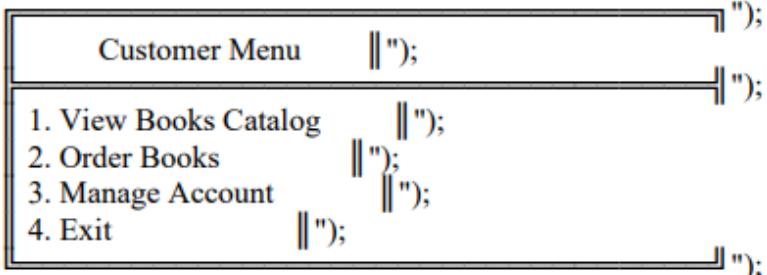
while (pw.equals(confirmPassword) == false) {
    System.out.println("Passwords doesn't match. Please re-enter your password.");
    System.out.print("Enter a password: ");
    pw = sc.nextLine();

    System.out.print("Re-enter your password: ");
    confirmPassword = sc.nextLine();
}

String userID = username+"Cust";
Customer c = new Customer(userID, username, pw, email, 2, nameFirst, nameLast);
addCustomersIntoFile(c);
sc.close();
}

@Override
protected int viewMenu(){
    Scanner sc = new Scanner(System.in);
    int option = 0;
    do{
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        System.out.println("Customer Menu");
        System.out.println("1. View Books Catalog");
        System.out.println("2. Order Books");
        System.out.println("3. Manage Account");
        System.out.println("4. Exit");
        System.out.println("Enter the option (1-4) : ");
        try{
            option = sc.nextInt();
            if(option < 1 || option > 4){
                System.out.println("Invalid option entered. Please enter a number between 1 and 4. Try Again :");
            }
        }catch(Exception e){
            System.out.println("Invalid option entered. Please Enter an appropriate");
        }
    }
}

```



```

number.\nPress any key to continue...");
    sc.nextLine();
    sc.nextLine();
    option = 10;
}

}while(option < 1 || option > 4);
return option;
}

public static Vector<Customer> getCustomersFromFile() throws FileNotFoundException {
    Vector<Customer> customers = new Vector<Customer>();
    Vector<User> users = User.readFromFile(2);

    for (User user : users) {
        Customer customer = new Customer(user.getUserID(), user.getUserName(),
user.getPassword(), user.getEmail(),
user.getUserRole(),user.getName().getfName(),user.getName().getlName());
        customers.add(customer);
    }
    return customers;
}

public static void addCustomersToFile(Customer c) throws IOException{
    try {
        PrintWriter outputFile = new PrintWriter(new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/usersDatabase.txt",true));
        String fullName = c.getName().getfName()+" "+c.getName().getlName();
        outputFile.write(c.getUserID()+" "+c.getUserName()+" "+c.getPassword()+" "
"+c.getEmail()+" "+fullName+" "+2+"\n");
        outputFile.close();

        System.out.println("\nYour username : "+c.getUserName()+"\nYour Password:
"+c.getPassword());
    } catch (Exception e) {
        System.out.print("An error is occurred during file operation..Please try again..\nPress
Any Key to Continue..");
        Scanner scan = new Scanner(System.in);
        scan.nextLine();
    }
}

public void viewAllCustomers(Vector<Customer> customers) throws
FileNotFoundException {
    if(customers.size() == 0){

```

```

        System.out.println("\nNo customers in the database.");
    }else{

        System.out.println("||-----");
        System.out.println("|| Customer ID || Username || Full Name ||");
        System.out.println("|| Email ||");
        System.out.println("||-----");

        for (Customer u : customers) {
            System.out.printf("|| %-15s || %-19s || %-19s || %-19s ||%n", u.getUserID(),
u.getUserName(), u.getName().getfName() + " " + u.getName().getlName(), u.getEmail());
        }

        System.out.println("||-----");
    }

    System.out.print("Press Enter to continue...");
    Scanner scan = new Scanner(System.in);
    scan.nextLine();
}

public static void displayCustomer(User u){

    System.out.println("||-----");
    System.out.println("|| Customer ID || Username || Full Name ||");
    System.out.println("|| Email ||");
    System.out.println("||-----");

    System.out.printf("|| %-15s || %-19s || %-19s || %-19s ||%n", u.getUserID(),
u.getUserName(), u.getName().getfName() + " " + u.getName().getlName(), u.getEmail());

    System.out.println("||-----");
}
}

public static void updateCustomerAcc(User currUser) throws IOException{
    Scanner scan = new Scanner(System.in);
}

```

```

int option=0;
String newValue = "";
do {
    System.out.print("\033[H\033[2J");
    System.out.flush();
    InventorySystem.header();
    displayCustomer(currUser);
    System.out.println("\n\n");
    System.out.println("Manage Account || ");
    System.out.println("1. Update Username || ");
    System.out.println("2. Update Password || ");
    System.out.println("3. Update Email || ");
    System.out.println("4. Update Name || ");
    System.out.println("5. Back || ");
    System.out.println("|| ");
    System.out.print("\nEnter your option (1-5): ");
    try {
        option = scan.nextInt();
        if(option < 1 || option >5){
            System.out.println("Invalid option entered. Please enter a number between 1 and
5. Try Again :)");
            System.out.print("Press any key to continue..");
            scan.nextLine();
            scan.nextLine();
        }
    } catch (InputMismatchException e) {
        System.out.println("Invalid option entered. Please Enter an appropriate
number.\nPress any key to continue..");
        scan.nextLine();
        scan.nextLine();
        option = 10;
    }
} while (option < 1 || option >5);
scan.nextLine();

switch (option) {
    case 1:
        System.out.print("\nEnter new username : ");
        newValue = scan.nextLine();
        currUser.setUserName(newValue);
        break;
    case 2:
        System.out.print("\nEnter new password: ");

```

```

newValue= scan.nextLine();

System.out.print("\nRe-enter your password: ");
String confirmPassword = scan.nextLine();

while (newValue.equals(confirmPassword) == false) {
    System.out.println("Passwords do not match. Please re-enter your password.");
    System.out.print("\nEnter a password: ");
    newValue = scan.nextLine();

    System.out.print("\nRe-enter your password: ");
    confirmPassword = scan.nextLine();
}
currUser.setPassword(newValue);
break;
case 3:
    System.out.print("\nEnter new email : ");
    newValue = scan.nextLine();
    currUser.setEmail(newValue);
    break;
case 4:
    System.out.print("\nEnter new first name : ");
    newValue = scan.nextLine();
    currUser.getName().setfName(newValue);
    System.out.print("\nEnter new last name : ");
    newValue= scan.nextLine();
    currUser.getName().setlName(newValue);
    break;

case 5:
    return;
default:
    break;
}
Vector<User> us = new Vector<User>();
us = User.readAllUsers();
try {
    FileWriter file = new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/usersDatabase.txt",false);
    for (User c : us) {
        String fullName = c.getName().getfName()+"_"+c.getName().getlName();
        if(c.getUserID().equals(currUser.getUserID())){
            String username = currUser.getUserName().replaceAll(" ", "");
            String em = currUser.getEmail().replaceAll(" ", "");

```

```

        fullName =
currUser.getName().getfName()+"_"+currUser.getName().getlName();
        String line = String.format("%s %s %s %s %s %d\n", currUser.getUserID(),
username, currUser.getPassword(), em,fullName, 2);
        file.write(line);
    }else{
        String line = String.format("%s %s %s %s %s %d\n", c.getUserID(),
c.getUserName(), c.getPassword(), c.getEmail(), fullName, c.getUserRole());
        file.write(line);
    }

}
file.close();
System.out.println("Updated Successfully.");
} catch (Exception e) {
    System.out.print("An error is occurred during file operation..Please try again..\nPress
Any Key to Continue..");
    scan.nextLine();
}

}

public void orderNewBooks(OrderManagement o){
    order.add(o);
}

public void displayBooks(Vector<Book>b,int role) throws FileNotFoundException{
    bookList = b;
    bookList.get(0).viewAllBooks(bookList,2);
}
}

```

Menu.java:

```
abstract class Menu {  
    abstract protected int viewMenu();  
}
```

Name.java:

```
class Name{  
    private String fName;  
    private String lName;  
  
    Name(String f, String l){  
        fName = f;  
        lName = l;  
    }  
  
    public String getfName(){  
        return fName;  
    }  
    public String getlName(){  
        return lName;  
    }  
  
    public void setfName(String f){  
        fName = f;  
    }  
  
    public void setlName(String l){  
        lName = l;  
    }  
}
```

OrderManagement.java:

```
import java.io.*;  
// to generate random order ID  
import java.util.*;  
import java.time.LocalDate;// to get latest date  
  
public class OrderManagement {  
    private String orderID;  
    private Vector<Book> bookList;  
    private Vector<Book> bookCart;
```

```

private Vector<OrderManagement> custOrderCart;
private Book bookInfo;
private String orderStatus;
private double totalAmount;
private User user;
private String orderDate;
private int quantityOrder;
private int quantityInStock;

public OrderManagement(String orderID, Book bookInfo, User user, double totalAmount,
String orderStatus, String orderDate, int quantityInStock
, int quantityOrder){
    this.orderID = orderID;
    this.bookInfo = bookInfo;
    this.orderStatus = orderStatus;
    this.user = user;
    this.totalAmount = totalAmount;
    this.orderDate = orderDate;
    this.quantityInStock = quantityInStock;
    this.quantityOrder = quantityOrder;
}

public OrderManagement(){
    this.orderID = "";
    this.orderStatus = "";
    this.user = new User();
    this.totalAmount = 0.0;
    this.orderDate = "";
    bookCart = new Vector<Book>();
    custOrderCart = new Vector<OrderManagement>();
    // this.report = new Report();
}

public Vector<OrderManagement> getOrderFromFile(int role) throws
FileNotFoundException {
    Vector<OrderManagement> orders = new Vector<OrderManagement>();
    Vector<Book> books = new Vector<Book>();
    try {
        Scanner sc = new Scanner(new
File("Submission/sec01_perdana/Group3/source_code/src/ordersDatabase.txt")); //need to
change path when export to github
        while(sc.hasNext()){

            String orderId = sc.next();
            double totalAmount = sc.nextDouble();
            String bookId = sc.next();
            int quantityOrder = sc.nextInt();


```

```

// double price = sc.nextDouble();
String userID = sc.next();
int userRole = sc.nextInt();
String orderStatus = sc.next();
String orderDate = sc.next();

Book bookSelected = new Book();
books = bookSelected.getBooksfromFile();
for(Book b:books){
    if(b.getBookID().equals(bookId.toUpperCase())){
        bookInfo = b;
    }
}
//System.out.println(bookInfo.getTitle() + " " + bookInfo.getMainAuthor() + " " +
bookInfo.getGenre() + " " + bookInfo.getQuantityInStock() + " " + bookInfo.getBookPrice());
User users = new User(userID,"userName", "password", "mail",
userRole,"fname", "lname");
OrderManagement order = new OrderManagement(orderId,
bookInfo,users,totalAmount, orderStatus, orderDate, quantityInStock, quantityOrder);
orders.add(order);

}
} catch (Exception e) {
    System.out.print("An error is occurred during the file operation..Please Try again
later..\nPress Any Key to continue..");
    Scanner scan = new Scanner(System.in);
    scan.nextLine();
}

return orders;
}
public void generateInvoice() throws FileNotFoundException {
    System.out.println(
    " _____
    || KEDAI BUKU KAMAL ||
    System.out.println(
    "|| Address: " + "No 12, Jalan Sri Sabah 25, Taman University, 41200 Klang,
Selangor ||");
}

```

```

System.out.println(
    "|| Phone Number: " + "03-9973628777" ||");
System.out.println(
    "|| ----- ||");
System.out.printf(
    "||%-30s | %-15s | %-20s%16s\n", "Book Name", "Quantity", "Price", "||");
System.out.println(
    "|| ----- ||");

double totalAmount = 0;

Vector<String> bookTitles = new Vector<String>();
Vector<Integer> bookQuantities = new Vector<Integer>();

for (OrderManagement order : custOrderCart) {
    Book book = order.getBookInfo();
    String title = book.getTitle().replaceAll("_", " ");
    int quantity = order.getQuantityOrder();

    int index = bookTitles.indexOf(title);
    if (index == -1) { //adding books in order to avoid duplicate book title to be shown in
invoice.
        bookTitles.add(title);
        bookQuantities.add(quantity);
    } else {
        bookQuantities.set(index, bookQuantities.get(index) + quantity);
    }
}

for (int i = 0; i < bookTitles.size(); i++) {
    String t = bookTitles.get(i);
    int quantity = bookQuantities.get(i);

    Book book = new Book();
    Vector<Book> books = new Vector<Book>();
    Book bk = new Book();

    books = bk.getBooksFromFile();

    for (Book b : books) { //double checking with title
        if (b.getTitle().replaceAll("_", " ").toUpperCase().equals(t.replaceAll("_", " ").toUpperCase())) {
            book = b;
            break;
        }
    }
}

```

```

        if(book != null) {
            double price = book.getBookPrice();
            System.out.printf(" %-30s | %-15d | %-20.2f%16s\n", t, quantity, price *
quantity," ");
            totalAmount += price * quantity;
        }
    }

    System.out.println("-----");
    System.out.printf(" %-29s RM%.2f%49s\n", "Total Amount to Pay (RM):",
totalAmount," ");
}

System.out.println("-----");
}
}

public void viewAllOrders(Vector<OrderManagement> orders,int role,String sortBy)
throws FileNotFoundException{
System.out.print("\033[H\033[2J");
System.out.flush();
InventorySystem.header();
Boolean isPending = false;
if(role ==3){
    for(OrderManagement order : orders){
        if(order.getOrderStatus().equals("Pending")){
            isPending = true;
            break;
        }
    }
    if(isPending == true){
        System.out.print(
"
-----\n"
);
        System.out.print(
" OrderID || BookID || Title || Main Author || Genre\n"
" Order Qty || UserId || Status || TotalAmount || OrderDate ||\n"
);
        System.out.print(
"
-----\n"
);
        for(OrderManagement ord:orders){
}
    }
}
}

```

```

        ord.displayOrder(role,sortBy);
    }
    System.out.print(
    "|||-----|||-----|||-----|||-----|||\n");
    System.out.print("Press Enter to continue...");  

    return;
}else{
    System.out.print("No orders to provide approval status. Please wait for new admin  

orders.");
    System.out.print("\nPress Enter to continue...");  

    Scanner scan = new Scanner(System.in);
    scan.nextLine();
    return;
}
System.out.print(
    "|||-----|||-----|||-----|||-----|||\n");
if(role == 1){
    System.out.print(
        "||OrderID||BookID||      Title      || Main Author ||  Genre  ||Order Qty||\n");
    UserId   || Status || TotalAmount || OrderDate ||\n");
}else{
    System.out.print(
        "||OrderID||BookID||      Title      || Main Author ||  Genre  ||Order Qty||\n");
    UserId   || Status || TotalAmount || OrderDate ||\n");
}
System.out.print(
    "|||-----|||-----|||-----|||-----|||\n");
for(OrderManagement ord:orders){
    ord.displayOrder(role,sortBy);
}
System.out.print(
    "|||-----|||-----|||-----|||-----|||\n");

```

```

        System.out.print("Press Enter to continue...");  

        Scanner scan = new Scanner(System.in);  

        scan.nextLine();  

    }  
  

    public void displayOrder(int role,String sortBy){ //Testing since need to check with  

Sarveish.  

        String genreStr ="";  

        switch (bookInfo.getGenre()) {  

            case 1:  

                genreStr = "Romance";  

                break;  

            case 2:  

                genreStr = "Mystery";  

                break;  

            case 3:  

                genreStr = "Fantasy";  

                break;  

            case 4:  

                genreStr = "Comedy";  

                break;  

            case 5:  

                genreStr = "Thriller";  

                break;  

            default:  

                break;  

        }  

        String title = bookInfo.getTitle().replaceAll("_", " ");  

        String author = bookInfo.getMainAuthor().replaceAll("_", " ");  

        if (role == 1) {  
  

            if(sortBy.equals("Customer")){  

                if(getOrderStatus().equals("Completed")){  

                    System.out.printf(" %-7s %-6s %-30s %-18s %-11s %-9d %-13s %-  

10s RM%-10.2f %-11s %n",getOrderID(), bookInfo.getBookID(),  

                    title,author, genreStr,getQuantityOrder(),user.getUserID()  

                    ,getOrderStatus(),getTotalAmount(),getOrderDate());  

                }  

            }else if(sortBy.equals("Supplier")){  
  

                if(getOrderStatus().equals("Pending") || getOrderStatus().equals("Approved") ||  

getOrderStatus().equals("Rejected")){  
  

                    System.out.printf(" %-7s %-6s %-30s %-18s %-11s %-9d %-13s %-  

10s RM%-10.2f %-11s %n",getOrderID(), bookInfo.getBookID(),  

                    title,author, genreStr,getQuantityOrder(),user.getUserID()
}

```

```

        ,getOrderStatus(), getTotalAmount(), getOrderDate());
    }
} else if(sortBy.equals("All")){
    System.out.printf(" %-7s %-6s %-30s %-18s %-11s %-9d %-13s %-10s RM%-10.2f %-11s %n",getOrderID(), bookInfo.getBookID(),
        title,author, genreStr,getQuantityOrder(),user.getUserID()
        ,getOrderStatus(), getTotalAmount(),getOrderDate());
}
} else if(role == 3 && getOrderStatus().equals("Pending")){
    System.out.printf(" %-7s %-6s %-30s %-18s %-11s %-9d %-13s %-10s RM%-10.2f %-11s %n",getOrderID(), bookInfo.getBookID(),
        title,author, genreStr,getQuantityOrder(),user.getUserID()
        ,getOrderStatus(), getTotalAmount(),getOrderDate());
}
}

public void addOrderIntoFile(Book bk, User user, int quantityOrder,int n) throws
IOException{
    Scanner scan = new Scanner(System.in);
    LocalDate UpdateAt = LocalDate.now();
    // Generate a random UUID
    UUID randomUUID = UUID.randomUUID();
    try {
        PrintWriter outputFile = new PrintWriter(new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/ordersDatabase.txt",true));
        Book books = new Book();
        Vector<Book> bkList = new Vector<Book>();
        bkList = books.getBooksFromFile();
        if(user.getUserRole() == 1){

            outputFile.write("OR0"+n+" "+getTotalAmount()+" "+
bk.getBookID()+" "+quantityOrder+" "+user.getUserID()+" "+user.getUserRole()+" "+
"+Pending+" "+UpdateAt+"\n");

        } else if(user.getUserRole() == 2){
            FileWriter file = new
FileWriter("Submission/sec01_perdana/Group3/source_code/src/booksDatabase.txt",false);
            for (Book bks : bkList) {
                if(bks.getBookID().equals(bk.getBookID())){
                    file.write(bks.getBookID()+" "+bks.getTitle()+" "+bks.getMainAuthor()+" "+
"+bks.getGenre()+" "+(bks.getQuantityInStock()-quantityOrder)+" "+
"+bks.getBookPrice()+"\n");
                }else{
                    file.write(bks.getBookID()+" "+bks.getTitle()+" "+bks.getMainAuthor()+" "+
"+bks.getGenre()+" "+bks.getQuantityInStock()+" "+bks.getBookPrice()+"\n");
                }
            }
        }
    }
}
```

```

        }
    }
    file.close();
    outputFile.write("OR0"+n+" "+Math.round(getTotalAmount() * 10) / 10.0+" "+
        bk.getBookID()+" "+quantityOrder+" "+user.getUserID()+" "+user.getUserRole()+" "+
        +"Completed"+" "+UpdateAt+"\n");

    OrderManagement order = new OrderManagement("OR0"+n, bk, user,
    Math.round(getTotalAmount() * 10) / 10.0, "Completed", UpdateAt.toString(),
    bk.getQuantityInStock(), quantityOrder);
    custOrderCart.add(order);
}
outputFile.close();
} catch (Exception e) {
    System.out.print("An error is occurred during the file operation. The order is not
added..Please Try again later..\nPress Any Key to continue..");
    scan.nextLine();
}

}

public void orderBook(String bookId, int quantityOrder, User realUser, int n) throws
IOException{
    Vector<Book> bkList = new Vector<Book>();
    Book book = new Book();
    // book.getBooksFromFile();
    bkList = book.getBooksFromFile();
    for(Book b:bkList){
        if(b.getBookID().equals(bookId)){
            this.setTotalAmount(b.getBookPrice()*quantityOrder);
            if(realUser.getUserRole() == 1){
                System.out.println("Total Amount: " + getTotalAmount());
                System.out.println("Order Successfully, Please wait Supplier to Approve the
order.....");
            }else{
                System.out.println("Purchase Successfully.....");
            }
            this.addOrderIntoFile(b, realUser,quantityOrder, n);
            this.getcustOrderCart();
            bookCart.add(b);
        }
    }
}

public boolean validation(String bookId, int bookQuantity, int roleId) throws
FileNotFoundException{

```

```

Vector<Book> bkList = new Vector<Book>();
Book book = new Book();
Boolean check = false;
bkList = book.getBooksFromFile();

try {
    for (Book b : bkList) {
        if (b.getBookID().equals(bookId)) {
            check = true;
            break; // Exit the loop if the book is found
        }
    }

    if (!check) {
        System.out.println("Sorry, the book id you entered is not found.");
    }
} catch (InputMismatchException e) {
    System.out.println(e.getMessage());
    check = false;
}
if(roleId == 2){
    for(Book b:bkList){
        if(b.getBookID().equals(bookId)){
            if(b.getQuantityInStock() >= bookQuantity){
                check = true;
                break; // Exit the loop if the book is found
            }else{
                System.out.println("Sorry, the quantity you order is more than the quantity in stock.");
                check = false;
            }
        }
    }
}

return check;
}

public String getOrderID(){
    return orderID;
}

public String getOrderStatus(){
    return orderStatus;
}

```

```
public double getTotalAmount(){
    return totalAmount;
}

public String getOrderDate(){
    return orderDate;
}

public int getQuantityOrder(){
    return quantityOrder;
}

public void setOrderID(String orderID){
    this.orderID = orderID;
}
public void setOrderDate(String orderDate){
    this.orderDate = orderDate;
}

public void setOrderStatus(String orderStatus){
    this.orderStatus = orderStatus;
}

public void setTotalAmount(double totalAmount){
    this.totalAmount = totalAmount;
}

public User getUser(){
    return user;
}

public Vector<Book> getbookList(){
    return bookList;
}

public Vector<Book> getBookCart(){
    return bookCart;
}

public Vector<OrderManagement> getcustOrderCart(){
    return custOrderCart;
}

public Book getBookInfo(){
```

```
        return bookInfo;
    }
}
```

Report.java:

```
import java.io.*;
import java.util.*;
```

```

public class Report {

    private Vector<OrderManagement> orderList;

    Report(Vector<OrderManagement> orderList){
        this.orderList = orderList;
    }

    Report(){
        orderList = new Vector<OrderManagement>();
    }

    public void generateReport(int roleID) throws FileNotFoundException{
        System.out.print("\033[H\033[2J");
        System.out.flush();
        InventorySystem.header();
        Scanner sc = new Scanner(System.in);
        OrderManagement orders = new OrderManagement();
        orderList = orders.getOrderFromFile(roleID);
        if(orderList.size() == 0){
            System.out.println("\nNo Order Found.\nPress any key to continue..");
            return;
        }
        System.out.println(
    "-----");
    System.out.println(
        "||      REPORT KEDAI BUKU KAMAL      ||");
    System.out.println(
    "-----");

    System.out.println(
        "||      TOP 10 BOOKS SOLD      ||");
    System.out.println(
        "|| ----- ||");
    System.out.printf(
        "||%-8s | %-30s | %-10s||\n", "Book ID", "Title", "Quantity");
    System.out.println(
        "|| ----- ||");

    double totalAmount = 0;
    double totalAmountOrder = 0;
}

```

```

Vector<String> bookId = new Vector<String>();
Vector<String> bookTitles = new Vector<String>();
Vector<Integer> bookQuantities = new Vector<Integer>();

String status = "";
int j= 0;
for (OrderManagement order : orderList) {
    Book book = order.getBookInfo();
    String title = book.getTitle().replaceAll("_", " ");
    int quantity = order.getQuantityOrder();
    String id = book.getBookID();
    String st = order.getOrderStatus();

    int index = bookTitles.indexOf(title);
    int min = 0;
    if(order.getOrderStatus().equals("Approved")){
        totalAmountOrder += order.getTotalAmount();
    } else if(order.getOrderStatus().equals("Completed")){
        totalAmount += order.getTotalAmount();
    }
    if (index == -1) {
        if(st.equals("Rejected") || st.equals("Pending")){
            }else{
                bookId.add(id);
                bookTitles.add(title);
                bookQuantities.add(quantity);
            }
    } else {
        if(st.equals("Rejected") || st.equals("Pending")){
            }else{
                bookQuantities.set(index, bookQuantities.get(index) + quantity);
            }
    }
}
for (int i = 0; i < bookQuantities.size() - 1; i++) {
    for (int k = i + 1; k < bookQuantities.size(); k++) {
        if (bookQuantities.get(i) < bookQuantities.get(k)) {
            int tempQuantity = bookQuantities.get(i);
            bookQuantities.set(i, bookQuantities.get(k));
            bookQuantities.set(k, tempQuantity);
        }
    }
}

```

```

        String tempTitle = bookTitles.get(i);
        bookTitles.set(i, bookTitles.get(k));
        bookTitles.set(k, tempTitle);

        String tempId = bookId.get(i);
        bookId.set(i, bookId.get(k));
        bookId.set(k, tempId);
    }
}

// Display the top 10 books
int n = Math.min(bookQuantities.size(), 10);
System.out.println("-----");
for (int i = 0; i < n; i++) {
    String id = bookId.get(i);
    String t = bookTitles.get(i);
    int quantity = bookQuantities.get(i);

    Book book = new Book();
    Vector<Book> books = new Vector<Book>();
    Book bk = new Book();

    books = bk.getBooksFromFile();

    for (Book b : books) { // double checking with title
        if (b.getTitle().replaceAll("_", " ").equalsIgnoreCase(t.replaceAll("_", " "))) {
            book = b;
            break;
        }
    }
    if (book != null) {
        System.out.printf("%-8s | %-31s | %-10d\n", id, t, quantity);
    }
}
if(totalAmount > totalAmountOrder){
    status = "Profit";
}else{
    status = "Loss";
}
double totalProfit = totalAmount - totalAmountOrder;
System.out.println("-----");
System.out.printf("%-31sRM%.2f%16s\n", "Total Amount Gained (RM):",
totalAmount, " ");
System.out.println("-----");
System.out.printf("%-31sRM%.2f%17s\n", "Total Amount Ordered (RM):",

```

```
totalAmountOrder,"||");
    System.out.println("-----||");
    System.out.printf("||%25s%5s%25s\n", "Status:", status, "||");
    System.out.printf("||%25sRM%2.2f%22s\n", "Total "+ status+": ", totalProfit,"||");

System.out.println("||-----||");
System.out.print("Press any key to continue... ");
sc.nextLine();
}

}
```

User.java:

```
import java.io.*;
import java.util.*;
```

```
class User extends Menu{  
    protected String userID;  
    protected String userName;  
    protected String password;  
    protected String email;  
    protected int userRole;  
    private Name name;  
  
    public User( String id, String names ,String pw, String mail, int roleID,String fName, String lName){  
        userID = id;  
        userName = names;  
        password = pw;  
        email = mail;  
        userRole = roleID;  
        name = new Name(fName,lName); //composition done  
    }  
  
    public User(){  
}  
  
    public String getUserID() {  
        return userID;  
    }  
    public void setUserID(String userID) {  
        this.userID = userID;  
    }  
    public String getUserName() {  
        return userName;  
    }  
    public void setUserName(String userName) {  
        this.userName = userName;  
    }  
    public String getPassword() {  
        return password;  
    }  
    public void setPassword(String password) {  
        this.password = password;  
    }  
    public String getEmail() {  
        return email;  
    }  
    public void setEmail(String email) {  
        this.email = email;  
    }  
    public int getUserRole() {
```

```

        return userRole;
    }
    public void setUserRole(int userRole) {
        this.userRole = userRole;
    }
    public Name getName(){
        return name;
    }

    public static User login(String name, String pw, int roleID) throws FileNotFoundException{
        Vector<User> users = new Vector<User>();
        users = readFromFile(roleID);
        for(User cust : users){
            if(cust.getUserName().equals(name) && cust.getPassword().equals(pw)){
                return cust;
            }
        }
        return null;
    }

    public static Vector<User> readAllUsers() throws FileNotFoundException{
        Vector<User> cust = new Vector<User>();
        try {
            Scanner sc = new Scanner(new
File("Submission/sec01_perdana/Group3/source_code/src/usersDatabase.txt"));

            while(sc.hasNext()){
                String id = sc.next();
                String name = sc.next();
                String pw = sc.next();
                String mail = sc.next();
                String fullName = sc.next();
                int roleID = sc.nextInt();
                String[] ary = fullName.split("_");
                String fName = ary[0];
                String lName = ary[1];
                User tempUser = new User(id, name, pw, mail, roleID,fName,lName);
                cust.add(tempUser);
            }
        } catch (Exception e) {
            System.out.print("An error is occurred during the file operation..Please Try again
later..\nPress Any Key to continue..");
            Scanner scan = new Scanner(System.in);
            scan.nextLine();
        }
    }
}

```

```

        return cust;

    }

    public static Vector<User> readFromUserFile(int userRoleID) throws
FileNotFoundException{
    Vector<User> cust = new Vector<User>();
    try {
        Scanner sc = new Scanner(new
File("Submission/sec01_perdana/Group3/source_code/src/usersDatabase.txt"));

        while(sc.hasNext()){
            String id = sc.nextLine();
            String name = sc.nextLine();
            String pw = sc.nextLine();
            String mail = sc.nextLine();
            String fullName = sc.nextLine();
            int roleID = sc.nextInt();

            if(roleID == userRoleID){
                String[] ary = fullName.split(" ");
                String fName = ary[0];
                String lName = ary[1];
                User temp = new User(id, name, pw, mail, roleID,fName,lName);
                cust.add(temp);
            }
        }

        sc.nextLine();
        sc.close();
    } catch (Exception e) {
        System.out.print("An error is occurred during the file operation..Please Try again
later..\nPress Any Key to continue..");
        Scanner scan = new Scanner(System.in);
        scan.nextLine();
    }

    return cust;
}

protected int viewMenu(){
    return 0;
} //Polymorphism done using abstraction.

```

}

SECTION D: TASK DISTRIBUTION

Category	Task	Person-In-Charge
Coding Implementation	Class InventorySystem	LIEW YONG ZHENG
	Class User	SARVEISH
	Class Admin	SARVEISH
	Class Customer	SARVEISH
	Class BookSupplier	CHONG WEI YANG
	Class Book	CHONG WEI YANG
	Class Report	LIEW YONG ZHENG & TAN CHEE SEN
	Class OrderManagement	TAN CHEE SEN
	Class Name	SARVEISH
	Class Menu	SARVEISH
Others (Diagrams, Reports and Slides)	Introduction	LIEW YONG ZHENG
	Class Diagram	CHONG WEI YANG & TAN CHEE SEN
	Flowchart Workflow	LIEW YONG ZHENG & SARVEISH
	OO Concept Explanation	LIEW YONG ZHENG
	User Manual	CHONG WEI YANG
	Presentation Slide	ALL

SECTION E: CONCLUSION

In summary, by using JAVA to create the system, we were able to gain some understanding of OOP concepts. We can implement the OOP concepts covered in the Object-Oriented Programming course in creating relationships between all the classes and that make up the entire system.