***Individual Project***

**Library Management System**

Table of Contents

[**1.** **Introduction** 3](#_Toc26658115)

[**2.** **Assumptions** 3](#_Toc26658116)

[**3.** **Use Case Diagrams** 4](#_Toc26658117)

[**4.** **Activity Diagram** 5](#_Toc26658118)

[**5.** **Class Diagram** 6](#_Toc26658119)

[**6.** **Object Oriented Concept** 7](#_Toc26658120)

[**6.1** **Classes** 7](#_Toc26658121)

[**6.2** **Objects** 8](#_Toc26658122)

[**6.3** **Constructor** 9](#_Toc26658123)

[**6.4** **Inheritance** 10](#_Toc26658124)

[**6.5** **Encapsulation** 10](#_Toc26658125)

[**6.6** **Packages** 11](#_Toc26658126)

[**7.** **User Manual** 12](#_Toc26658127)

[**7.1** **Login page** 12](#_Toc26658128)

[**7.2** **System Admin** 13](#_Toc26658129)

[**7.3** **Librarian** 18](#_Toc26658130)

[**7.4** **Guest** 21](#_Toc26658131)

[**8.** **Additional features** 24](#_Toc26658132)

[**8.1** **Edit, delete, search and view Guest** 24](#_Toc26658133)

# **Introduction**

Library Management System was developed in order to maintain some of the library activities and roles of people involved in the library daily activities (librarian, guest and etc.) such as Search, adding new book and etc... The application will be used by several users including librarians, system admin as well as guests. In the Library Management System, each user (librarian, system admin and guest) can perform certain tasks, except system admin who can perform all tasks provided by the application. For example, system admin can create/edit/delete books, librarians, guest, as well as generating reports such as total borrowed/available books and etc.…. In the other side, librarian can get access to Library Management System if he/she was granted the access by system admin. In addition, librarian can add, edit, search, view, check in/check out books and register guests of the Library Management System. Finally, the system guest can get access to the system as registered or non-registered guest. Guest registration can be done by system admin and librarian. The Guests can log into the system without authentication. The only activity that guest is allowed to perform is to search books and view the various details of books such as book title, book author, date of publication and etc.

# **Assumptions**

1. **Use Case Diagrams**



Figure 1Usecase Diagram for Library Management System

# **Activity Diagram**



Figure 2: Activity Diagram for Library Management System

# **Class Diagram**



Figure 3:Class Diagram of Library Management System

# **Object Oriented Concept**

OOP is an approach for constructing a software system based in the concept of classes and objects which are used based on the modeling of the real world entities. For example, objects are the physical entities which can be found in the universe around us such as hardware, building, car, chair, and also ball are all seen as examples of objects. In addition to that, objects also have state and the state of the object is the condition of it or a set of circumstances which describe the object. For example, the state of a bank account object would have the current balance, and the state of a clock object would be the current time. *(Edward V. Unknown data of publish).*

## **Classes**

Classes are the fundamental building blocks of a Java. In object-oriented programming approach, a class is considered as a prototype from which objects are created. Those object contains the instance variables and methods defined by the class. The blue print describes the state and behavior that is used as a blueprint to create objects of that class. In Library Management System, class concept is used to define variables and methods that describes the state and behavior of the class. For example, book class has variables (Book ID, Book Author, Book Title) and Behaviors (Search book, add book and etc.).

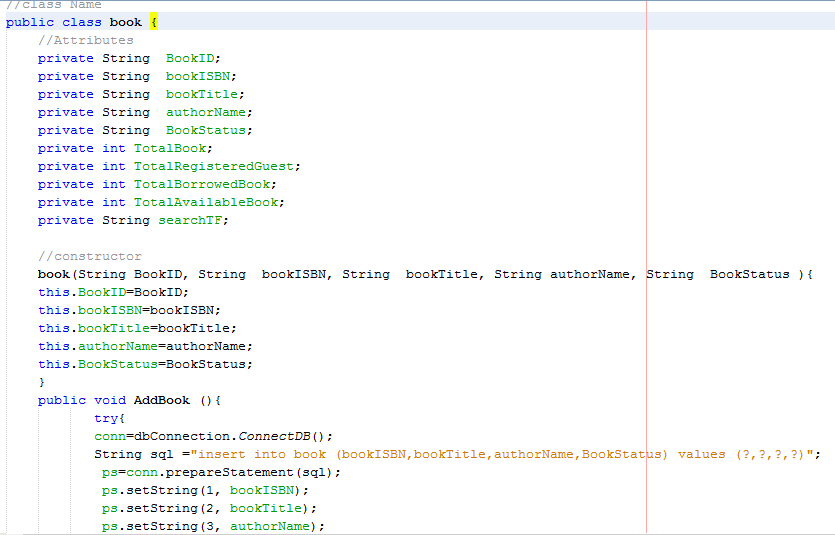


Figure 4: Book Class Showing Attributes and Methods

## **Objects**

Objects are an instance of the class. It is being used to access the attributes of the class which enhances the security of the system. Objects contain data, and code to manipulate that data. The entire set of data and code of an object can be made user defined data type with the help of a class. In fact, objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created.

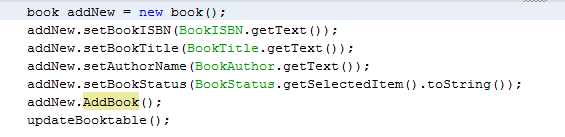


Figure 5: Figure 4: Book Object

The above code is based on Library Management System, object concept is used to instantiate class in order to use the defined variables and methods that describes the state and behavior of the class. For example, book class has variables (Book ID, Book Author, Book Title) and Behaviors (Search book, add book and etc.), an object of book class can be made as book addBook= new book() as showing the above figure.

## **Constructor**

A class has constructors which are invoked in order to create objects from the class prototype. Constructor declarations look like method declarations, except that they use the same name of the class and have no return type.

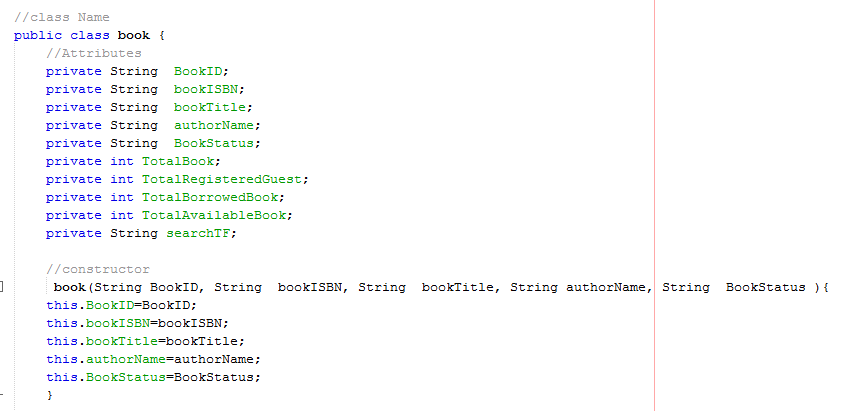


Figure 6 Book Constructor of Book Class in the Library Management System

The above figure shows the use of constructor*, in* Library Management System, it is used to create objects from the class prototype. its declarations look like method declarations, except that they use the same name of the class and have no return type as it is showing in the above figure.

## **Inheritance**

Inheritance is property whereby one class extends another class by including additional methods and/or variables. The original class is called the superclass of the extending class, and the extending class is called the subclass of the class that is extended. A class able to inherits another class where by adding the keyword ‘extend’ after class name followed by the class that is to be inherited. The use of inheritance is beneficial for the purpose of security and reusability so there is no need to declare more functions in the child classes and simply can inherit them from the parent classes and execute their operation in any class. For instance, if we had a class of “fruit”, we could extend it by defining classes of “apple”, “orange” and “banana”. Each of these subclasses could be described in the following way:

apple "is a" fruit

orange "is a" fruit

banana "is a" fruit

Because each of our subclasses extends the “fruit” class, it has all the attributes and methods of a “fruit” plus any *specific* characteristics of its own. *(NefariousDesign.2006).*

## **Encapsulation**

Encapsulation is also a fundamental object-oriented concept. It is the bundling of data that represents the state of an object together with the code responsible for manipulating that data. Encapsulation means shielding. Each object-oriented object has a shield around it. Objects cannot see other objects. They can exchange things though, as if they are interconnected through a hatch. Figure 1 shows the concept of the encapsulation. It separates the external aspects of an object from the internal implementation details of the object, which are hidden from other objects. The object encapsulates both data and the logical procedures required to manipulate the data. *(Bill. V. 2012). For example, in* Library Management System, class book has data (book ID, Book Title), so, those data can are encapsulated using the set and get function as showing below:

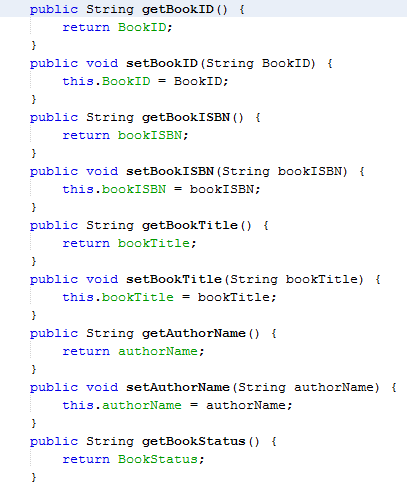


Figure 7: Data Encapsulation of book class

## **Packages**

Packages are basically collection of files and folders all kept and maintained within a single directory. This one directory is what is known as the package. In java, a package is declared using the “package” keyword before any other declarations in the java file. In Library Management System, package concept is used to collectively classify a group of similar or related classes. All classes created for GUI designs are kept in one package and functional classes are kept in another package. This mechanism helps to organize classes into groups where related classes are kept together. For example, in

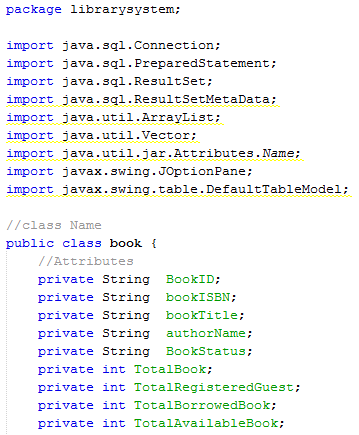
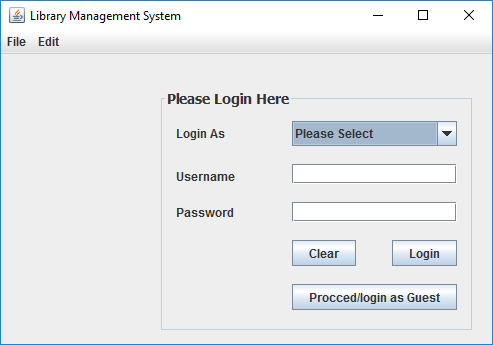


Figure 8: the use of Package in the book class

The above figure shows the use of package*, in* Library Management System, it is used inside the book class in order to get accessed to the outside resources (classes, packages and etc.) and get accessed from outside (classes, packages and etc.).

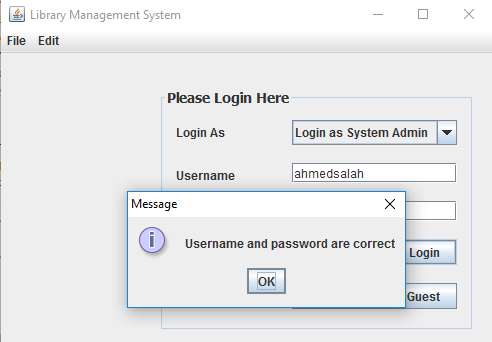
# **User Manual**

## **Login page**

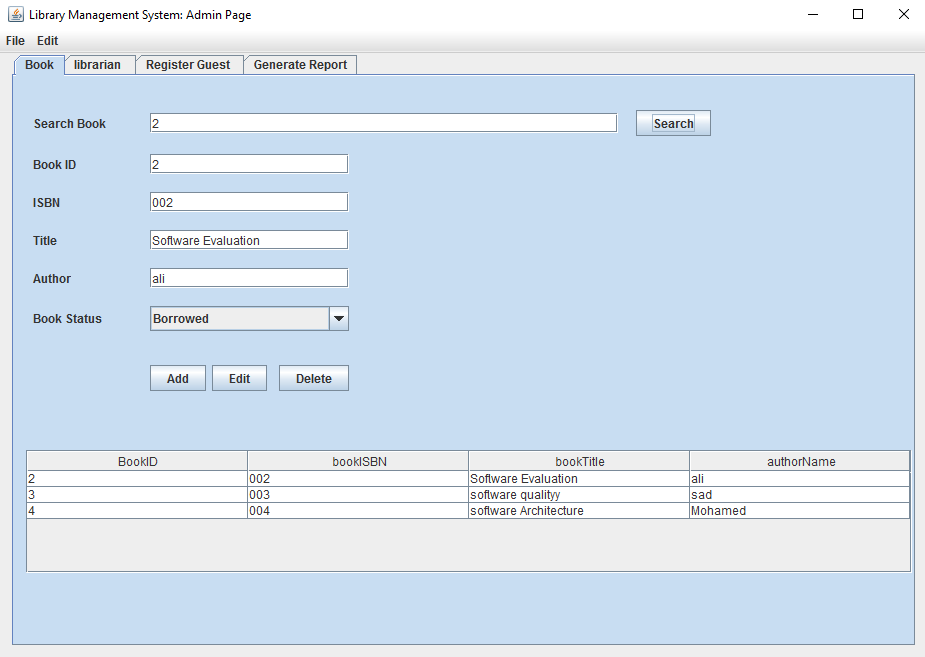


## **System Admin**

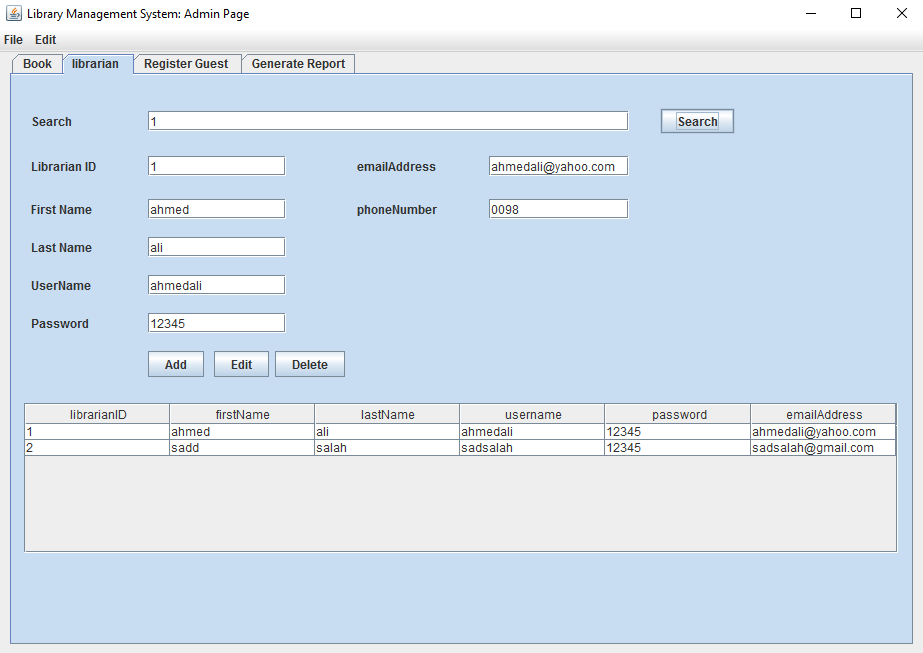
* **Login as System Admin**

****

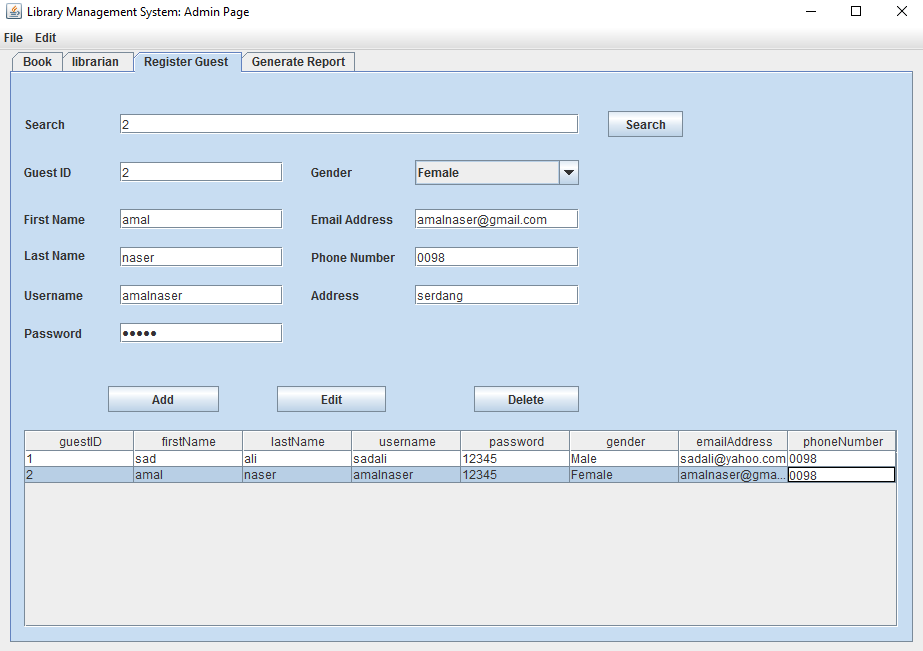
* **Main Page** **of** **System Admin: Book Section**



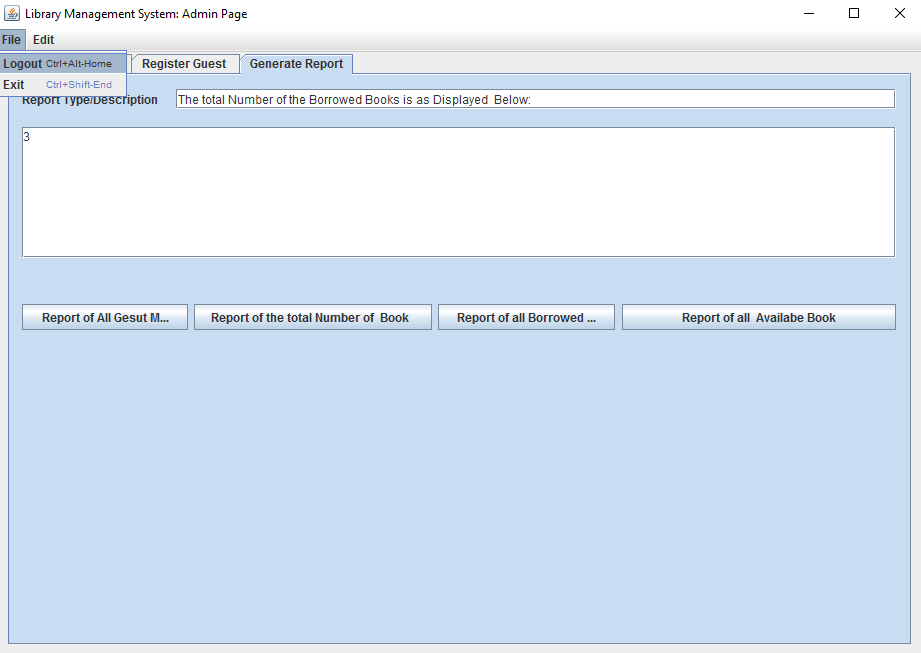
* **Main Page** **of** **System Admin: Librarian Section**



* **Main Page** **of** **System Admin: Register Guest Section**

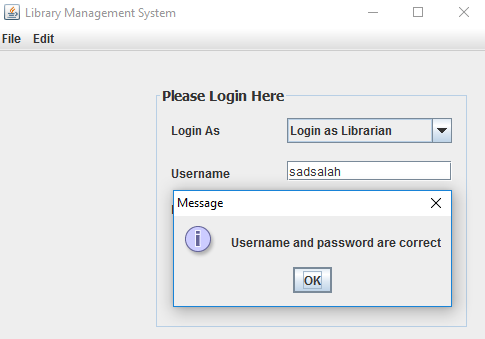


* **Main Page** **of** **System Admin: Register Guest Section**

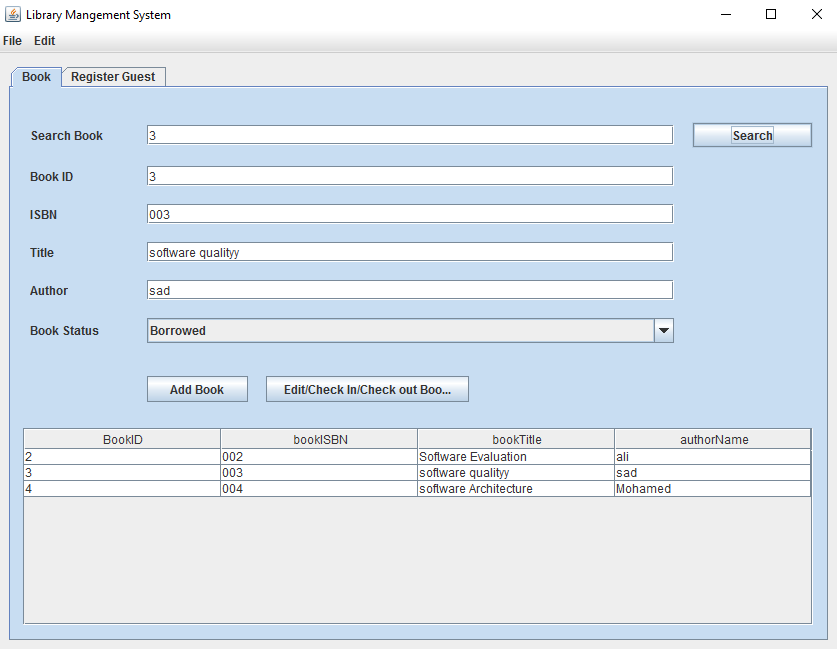


## **Librarian**

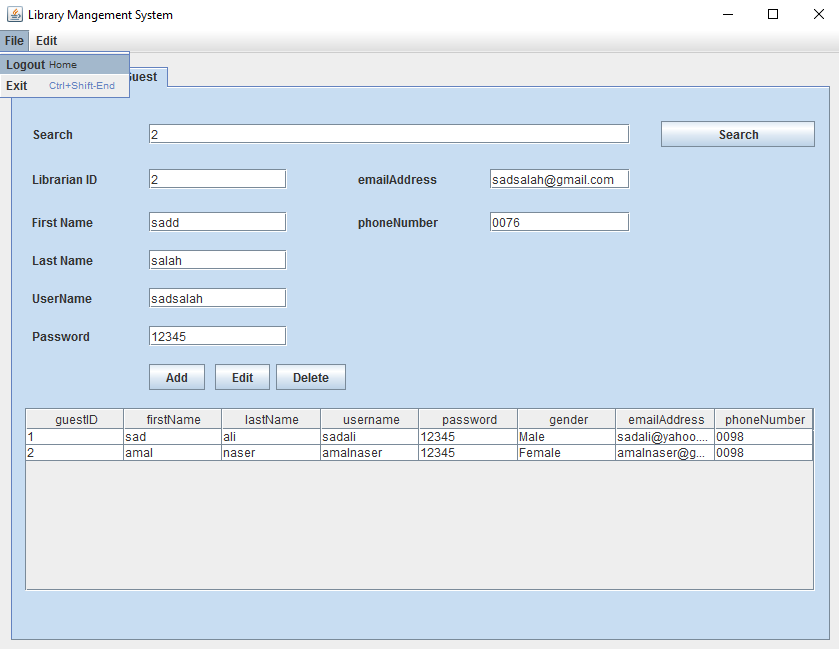
* **Login as Librarian**

****

* **Main Page of Librarian: Book Section**

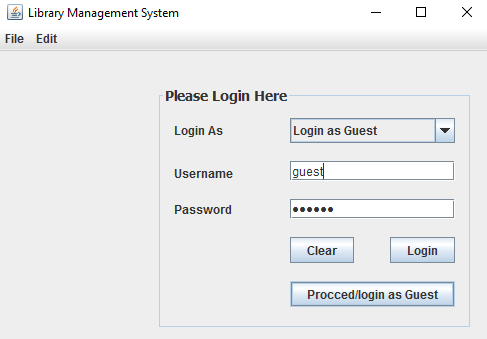
****

* **Main Page of Librarian: Register Book Section**

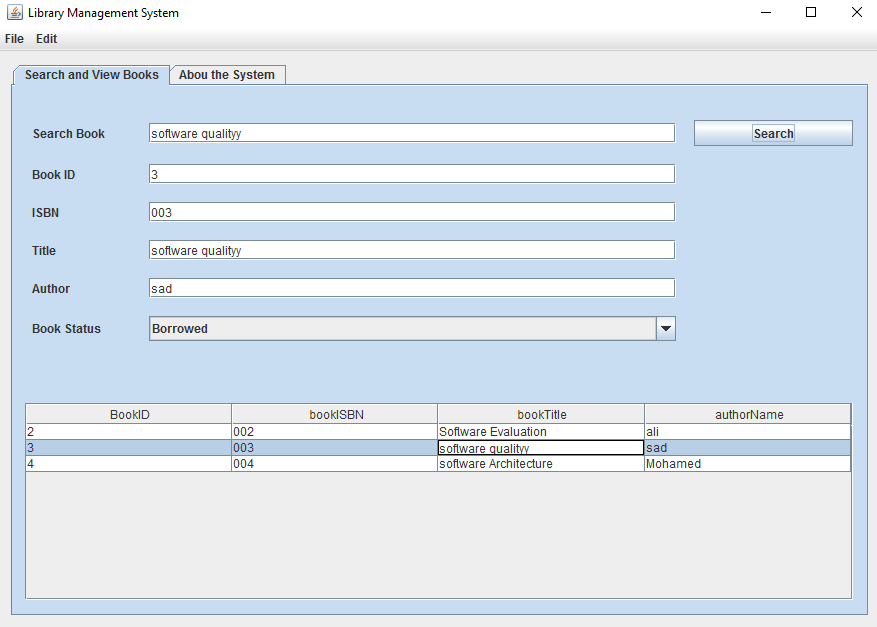
****

## **Guest**

* **Login to the system as guest**

****

* **Main Page of Guest: Search and View Book Detail**

****

# **Additional features**

## **Edit, delete, search and view Guest**

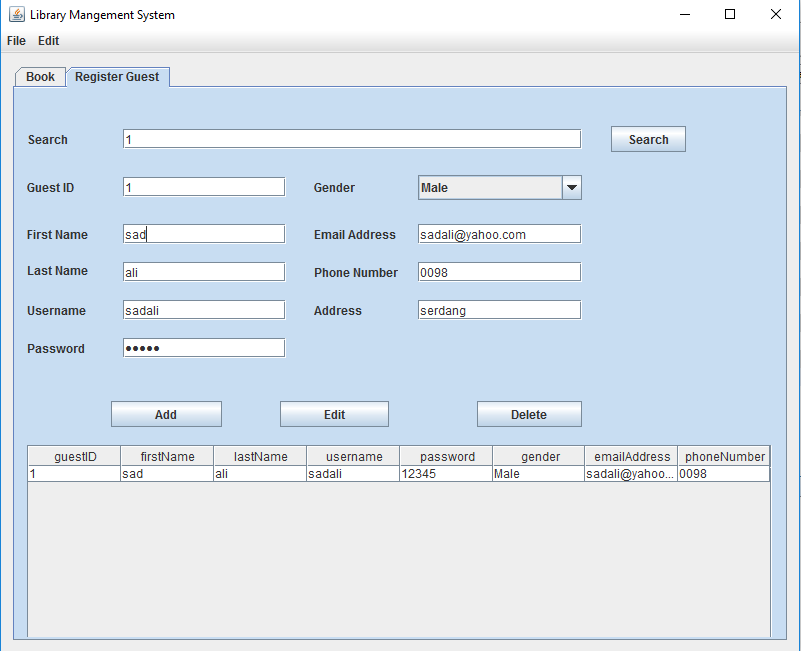


Figure 9: Edit, delete, search and view Guest

The above figure shows four additional features (additional of four Functionalities). The functionalities are editing, searching, viewing and deleting guests. Those functionalities were not mentioned and required in the assignment question, but as part of the requirement to have additional features, the author of this document decided to add those features as additional features of the library Management System.