OBJECT-ORIENTED PROGRAMMING COURSE PROJECT REPORT:

LIBRARY MANAGEMENT SYSTEM

Date of Report: 8/1/2021

Group members:
Do Quang Minh – ITITIU19028
Nguyen Hoang Linh – ITITIU19023

Table of Contents

I. Introduction

Our group project is a library management system for two types of user which are students who borrow books and managers who manage the borrowing of the students.

II. Contribution

- Đỗ Quang Minh: Idea contribution, Planning, Report, Powerpoint, UI Design
- Nguyễn Hoàng Linh: Idea contribution, Back-end, support UI, class diagram

Overall:

Full name	Đỗ Quang Minh	Nguyễn Hoàng Linh
Student ID	ITITIU19028	ITITIU19023
Percentage task (%)	50%	50%

III. App features:

1) User types:

- Our system is used for keeping track of all the books that are borrowed, returned, and added to the library. Also, students can find the books they want in book list in "E-Library". So, there are two types of users in our system:

+ **Manager**: Manage information of students, books, borrowing books and returning books

+ **Student**: Students can track their borrowing status and find some of the books that are available in the library

- Before using the system, users need to log into the system first. If not have an account, the user can sign up at any time

- In addition, users can completely delete the account if they do not want to use the system.

2) Main features

Main features	Access
Search for the books in the library	Student + Manager
Tracking book lending	Student
Tracking loan status of students	Manager
Manage borrowing books	Manager
Manage returning books	Manager
Update book lists (Add new book + Delete book)	Manager

3) Special features

Special features	Access
Read pdf books	Student + Manager
Audio books	Student + Manager

IV. Class Diagrams and Design

- We have separate parts of class diagram as an attached .pdf and .jpg file. Due to the size of each parts is quite large, in this document, we just show some highlight parts in our system.

- We simply divided our class diagram into 2 parts:

  + **Back-end**

  + **User interface**

1) Back-end:

- About the back-end, we have **6** classes:

  + **User** (Abstract class): Contain general information of all users

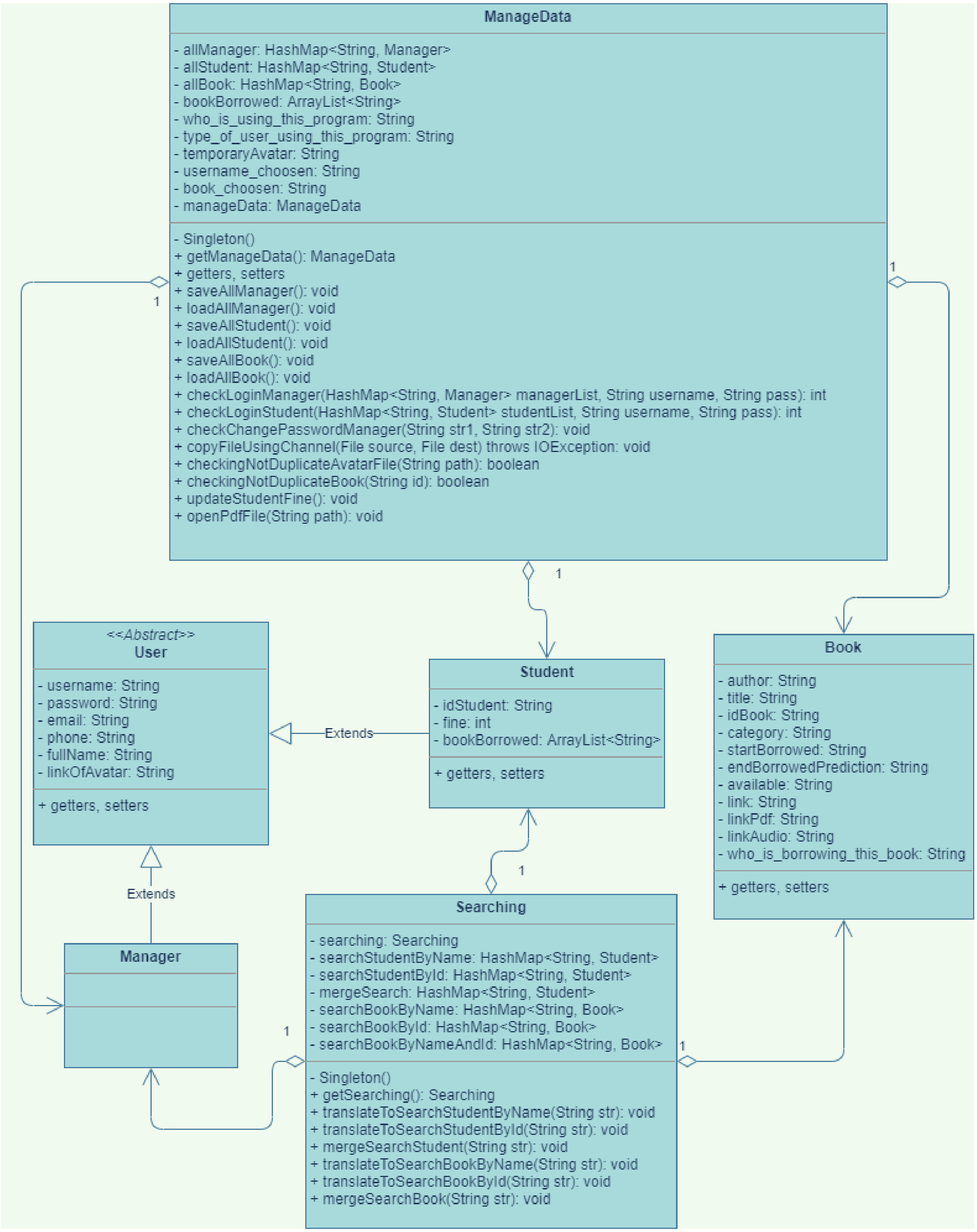
  + **Student** (extends from User): Contain unique information of all students

  + **Manager** (extends from User): Contain unique information of all managers

  + **Book**: Contain information of each book in the library

  + **Searching**: Manage output of data when users use the "Search" feature

  + **ManageData**: Monitor all the system as managing activities of users, save/load the data and do some other important tasks related to the operation of the system

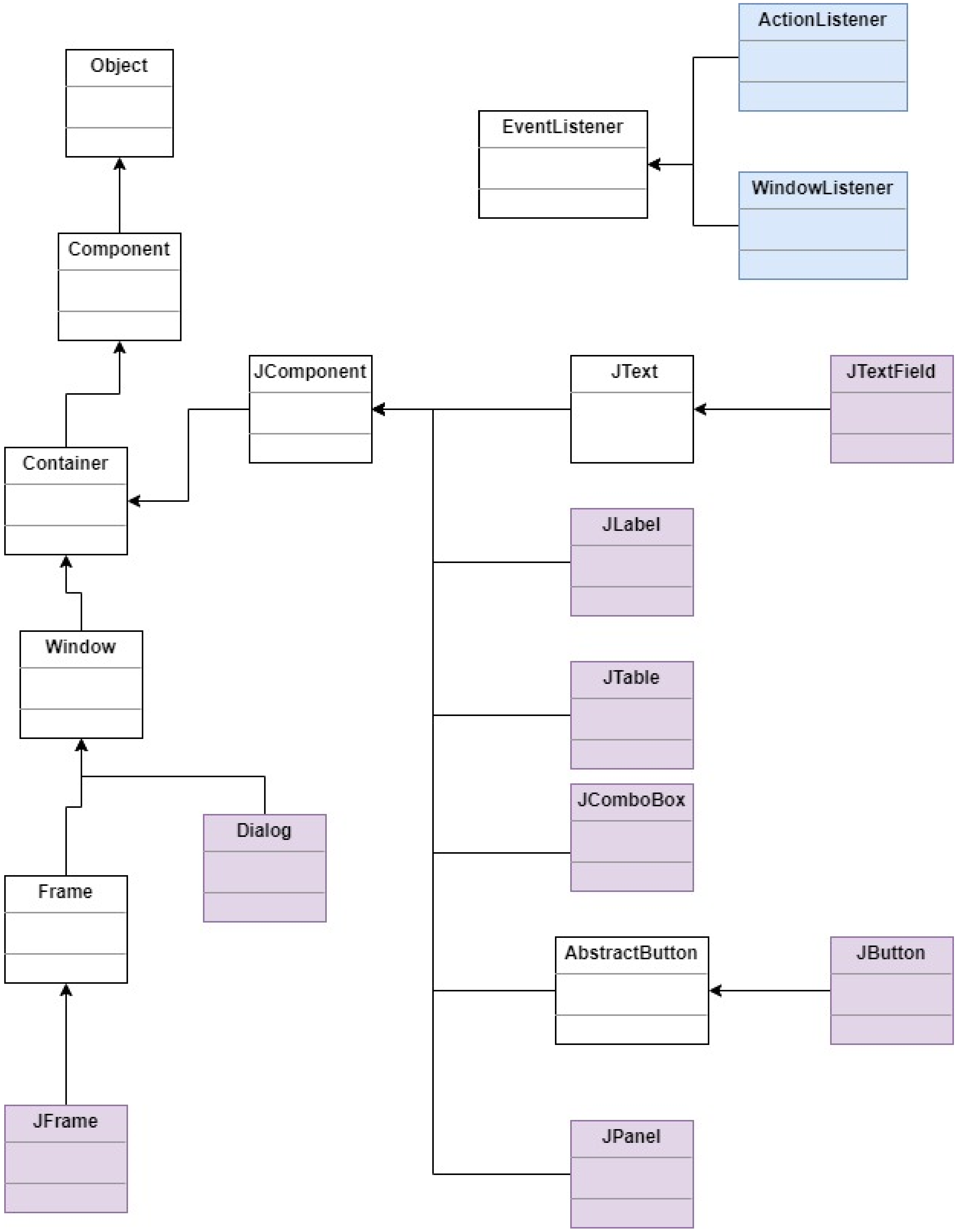

ManageData
- allManager: HashMap<String, Manager>
- allStudent: HashMap<String, Student>
- allBook: HashMap<String, Book>
- bookBorrowed: ArrayList<String>
- who_is_using_this_program: String
- type_of_user_using_this_program: String
- temporaryAvatar: String
- username_choosen: String
- book_choosen: String
- manageData: ManageData
- Singleton()
+ getManageData(): ManageData
+ getters, setters
+ saveAllManager(): void
+ loadAllManager(): void
+ saveAllStudent(): void
+ loadAllStudent(): void
+ saveAllBook(): void
+ loadAllBook(): void
+ checkLoginManager(HashMap<String, Manager> managerList, String username, String pass): int
+ checkLoginStudent(HashMap<String, Student> studentList, String username, String pass): int
+ checkChangePasswordManager(String str1, String str2): void
+ copyFileUsingChannel(File source, File dest) throws IOException: void
+ checkingNotDuplicateAvatarFile(String path): boolean
+ checkingNotDuplicateBook(String id): boolean
+ updateStudentFine(): void
+ openPdfFile(String path): void
1
1
1
<<Abstract>>
User
- username: String
- password: String
- email: String
- phone: String
- fullName: String
- linkOfAvatar: String
+ getters, setters
Extends
Student
- idStudent: String
- fine: int
- bookBorrowed: ArrayList<String>
+ getters, setters
Book
- author: String
- title: String
- idBook: String
- category: String
- startBorrowed: String
- endBorrowedPrediction: String
- available: String
- link: String
- linkPdf: String
- linkAudio: String
- who_is_borrowing_this_book: String
+ getters, setters
Extends
1
Manager
Searching
- searching: Searching
- searchStudentByName: HashMap<String, Student>
- searchStudentById: HashMap<String, Student>
- mergeSearch: HashMap<String, Student>
- searchBookByName: HashMap<String, Book>
- searchBookById: HashMap<String, Book>
- searchBookByNameAndId: HashMap<String, Book>
- Singleton()
+ getSearching(): Searching
+ translateToSearchStudentByName(String str): void
+ translateToSearchStudentById(String str): void
+ mergeSearchStudent(String str): void
+ translateToSearchBookByName(String str): void
+ translateToSearchBookById(String str): void
+ mergeSearchBook(String str): void
1
1

Class diagram of back-end part

2) User interface:

- About the user interface, we have about **18** classes. All of them extend from the Jframe, and the components of each class is taken from some of the classes in Java Swing library as JPanel, JLabel, JButton, …
- Also, most of the events in the user interface are taken from class **ActionListener** extended from **EventListener** class which is in the java.awt library
- Here is the function of each UI class:

Class	Function
Frame1_Intro	The first menu of this system
Frame2_LoginManager	Login form for "Manager"
Frame3_LoginStudent	Login form for "Student"
Frame4_RegisterManager	Register form for "Manager"
Frame5_RegisterStudent	Register form for "Student"
Frame6_ProfileManager	Show the personal information of "Manager" users
Frame7_ProfileStudent	Show the personal information of "Student" users
Frame8_EditProfile	The interface helps "Manager" change the information
Frame10_ManageStudents_General	Manage the list of students
Frame11_ManageStudents_Detail	See detail of each students
Frame13_SeeBorrowedBooks	See borrowing book situation of each students
Frame14_Book_Student	E-Library for students
Frame15_EditProfileStudent	The interface helps "Student" change the information
Frame16_Book_Manager	E-Library for managers
Frame18_ForgotPassword	Manager can get the forgotten password
Frame19_ForgotPassword_Student	Student can get the forgotten password
Frame20_AddBook	Manager can add a book here
Frame21_StudentBorrowBook	Manager can lend a book here



Hierarchy of some used components of Java Swing in our system

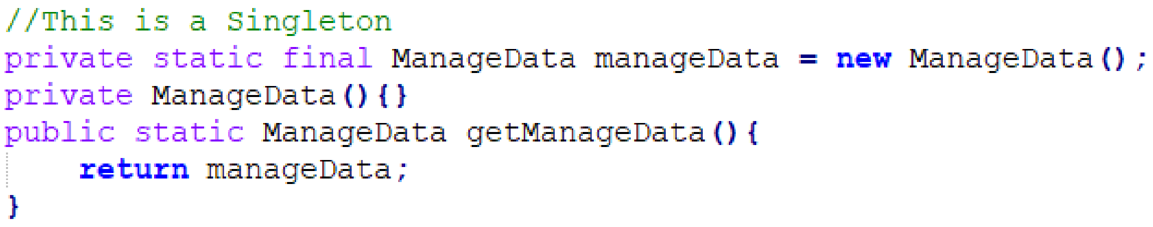
V. Design pattern used:

- The "Library Management System" is a management system, so it is important to create a component that oversees and manage the entire system. In our system, there are 2 classes that are used to monitor the system.

+ **ManageData**: Manage the whole system: monitor user activities, save/load data, ...

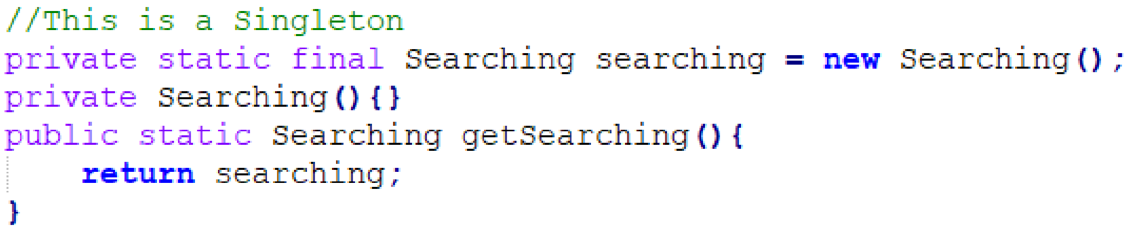
+ **Searching**: Manage outputting the data on the table when users use the "Search" function.

- To make the 2 classes work effectively, we use the **<u>Singleton Pattern</u>** because each class should have only 1 instance.



```
//This is a Singleton
private static final ManageData manageData = new ManageData();
private ManageData(){}
public static ManageData getManageData(){
    return manageData;
}
```

*Singleton example in class **ManageData***



```
//This is a Singleton
private static final Searching searching = new Searching();
private Searching(){}
public static Searching getSearching(){
    return searching;
}
```

*Singleton example in class **Searching***

VI. App workflows:

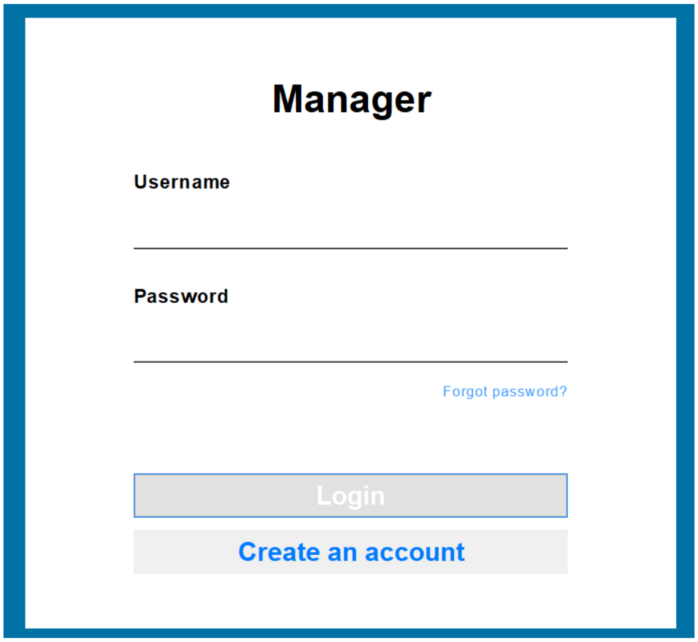
1) Manager workflows:

- At the login form, manager has 3 options:

+ **Login**

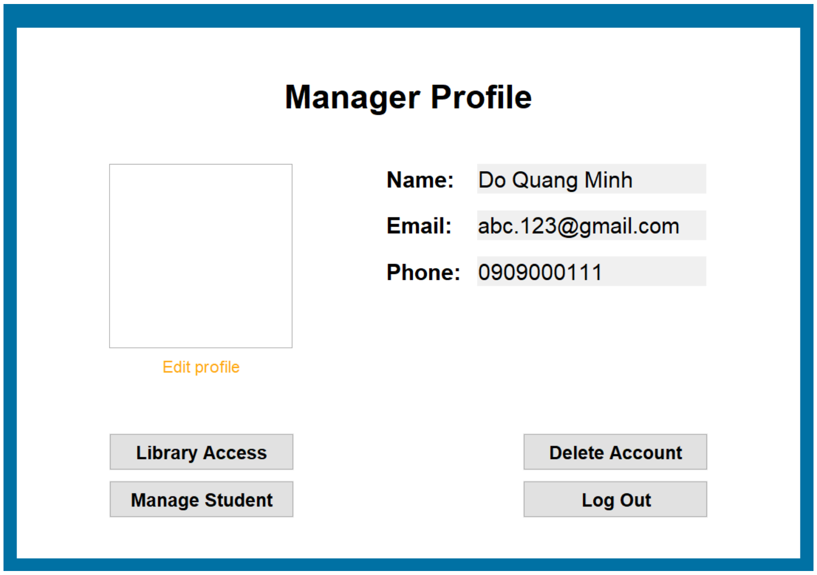
+ If they do not have account, they can click on the “**Create an account**” to get an account

+ If they had an account but forgot their password, they can click on “**Forgot password?**” to get the password again



Login form of Manager

- After the manager login, they will see the profile


Manager Profile

Name: Do Quang Minh

Email: abc.123@gmail.com

Phone: 0909000111

Manager profile

- If they want to see who is borrowing some books, they can go to "**Manage Student**". At this feature, manager can see list of student's accounts and record if someone return books.
- If they want to go to E-library, they can click on "**Library Access**". At the E-library, they can see and edit the book list, lend a book to someone, read a book by pdf file and listen audio book.

2) Student workflows:

- Similar to Manager, at first each student will see a login form with 3 options: "**Login**", "**Create an account**" and "**Forgot password**?". And after the student login to the system, they also see their information in the profile.
- Also, student can go to E-library to see the book list, read pdf book and listen audio books.
- The difference between student and manager is the student can not edit the book lists and manage borrowing/returning books.

VII. Conclusion

In conclusion, our project will help people who manage the library system monitor the information of the books, borrowing and returning of students. Also, it can support reading the e-book through pdf files and listening audio books in e-library section.

However, the system is still in v1.0 which means that there are many things needed to upgrade, maintain and update new features. Also, the system currently uses very simple database, so the data can be exposed. In the future, after learning any subjects related to database, we will try the best to upgrade our system again.

Moreover, we will try to learn experience from users to know what is necessary for a good software. We will improve the performance the best day by day to serve everyone the best.