Table of Contents

1. Introduction

Utilizing library resources is a shared experience for almost every citizen in a civilized society. In the old days, borrowing books from libraries is inconvenient and time-consuming because all the operations handling transactions are done by librarians manually. To illustrate, those operation includes scanning library card, scanning bar code on top of books, and recording ingoing and outgoing books. When all requests are handled sequentially, it is no surprise that queues exist when it comes to library services. Furthermore, since there was no proper maintenance of databases about issues of books, searching for a book can also be a hassle process for the readers.

Fortunately, with the advancement of technology, different library management systems (LMS) are released to solve the problem. Typical examples are Unpad Library Management System (ULiMS) introduced by Universitas Padjadjaran and University of Gitwe Library Management Information System (UG-LMIS) introduced by University of Gitwe [3][8].

According to S. Sharma et al. [9], LMS refers to the digitalized way of performing daily operations of a library, so that it can boost efficiency and make tasks more manageable. It can handle operations like book borrowing, book return, and book record tracking, as well as user authentication, in a shorter time with the help of computers. Meanwhile, information like book location, borrower, on-shelf and off-shelf quantity, and book authors can also be tracked easily so that users can search for a book easily by just inputting the book id in the search section of the application.

However, the currently proposed solutions are not perfect.

First, the time needed to handle requests from multiple users depends on the number of computers that exist in the library. As the software is installed on a computer and can only handle requests from one single user at a time, it is understandable that the computerized system can still be slow when the number of computers not meeting the number of users. Taking ULiMS as an example [8], fewer people have chosen to use the digitalized system due to the number of computers available.

Second, it takes time for users to find the location of the computerized system and some of them may just don’t know the existence of the system. In that case, the ease of accessing digital

systems hinders user experience in book searching and book borrowing. According to past studies related to ULiMS [8], it is found that users tend to reach for help from the librarian when they get no idea about accessing the computerized system through computers in the library.

This project aims to introduce a mobile-based LMS which can solve the problem mentioned above, speeding up the required time for each service request, as well as providing easy access to the digitalized system.

The newly created mobile application follows similar characteristics to the existing computer-based system. Functionalities like book searching, book borrowing, book return, book renewal, and user authentication will be included. In addition, Librarian will have access to functions like catalog editing and book return request handling. It is believed that the mobilized system can provide ease of access and allow different users to perform library transactions at the same time.

This paper is organized as follows: First, functionalities of existing LMS and technical components involved will be discussed and analyzed in section II and III. Then, design of our system will be coved in section IV. Finally, Implementation details like used methodology and resources for carrying out this project will be included in section V.

2. Review on Library Management System

In this section, existing approaches in implementing LMS will be discussed. We will cover the strength and weakness of currently developed LMS. We would get insight from previous studies and explain the selections of methodologies in this project.

2.1 From Software Architecture perspective

2.1.1 Desktop Application built using .NET framework

K. Chinmai Devi et al.(2021) proposed a system built in .Net framework using C# programming language, in connection to SQL SERVER [5]. It provides functions like creating members, adding new books, editing book catalogs, issues of books, etc. On top of that, the whole system is divided into 3 modules, namely user, book info, and admin. Different users can access different functions with the help of keyboard and mouse. The advantage of building the system using C# .NET is that the .NET platform can provide templated solutions to common problems in programming. Also, ASP.NET can be served as hosting environment and it is suitable for developing websites and Internet-distributed objects by providing a collection of supporting classes, as well as publishing mechanisms. However, the limitation of building LMS in desktop application is obvious. K.Chinmai Devi et al. [5] have mentioned that a possible enhancement is by connecting the system to a bar code scanner, so that the users do not need to manually type any book ID or member ID. Therefore, to make the whole system compact, the system is better implemented using mobile applications, as there will be a camera pre-installed in every device.

2.1.2 Desktop Application built with API for connecting to servers

Shanmugam A.P et al. have proposed a similar project [10], also using C# .NET as frontend and SQL server for the backend. Instead of using ASP .NET for database connection, Open database connectivity (ODBC) served as the application programming interface (API) between the client and server. In their previous studies, it is found that ODBC did protect data from being alternate and is considered safe [10].

IKiS is another desktop application running in kiosk, proposed by Mardiana and Meizano Ardhi Muhammad in 2017 [6]. It supports specific functions like searching for resources and printing transaction-independent documents. Different from the system proposed by K. Chinmai Devi et al., the client side is touchscreen-based and the server side is connected via API. Not only does this makes the design more compact, but also satisfies the independent principle in software engineering. It makes sure that two parts can run independently and enhance the reusability of the whole program. Such software architecture will also be followed in our project. However,

the inclusion of different hardware components has made the system occupy a large physical space. Therefore, we would adopt the mobile approach in our project.

2.1.3 Desktop Application with direct access to database

A similar project have been done by A.Thendral Mary et al. in 2017 [2]. Enhanced Library Management System is a window application built using Eclipse Neon IDE and MySQL database. The system provides similar functions like controlling user access and managing book transactions. Instead of establishing connections via API, the client application made direct connection to database server. Though this can make the connection easier to be implemented, this is considered unsafe as the database will be accessible from the Internet and making it more vulnerable to attacks. As a result, our project would provide web service for users, serving as a bridge between client application and database.

2.1.4 Web Application

Other than desktop application, another way of implementation is through web application. UG-LMIS [3] is a typical example of web-based library system. It used PHP as frontend software and MYSQL serving as backend software. The advantage of this kind of implementation is that the system can be accessed by using either computers or mobile phones. It did enhance accessibility and provide convenience for users. Nevertheless, web application is known to have limited access to hardware components on mobile phone compared to traditional native applications. It is more suitable to build the system on native app so that the internal camera can be assessed for scanning barcodes.

2.2 From Input Methods perspective

2.2.1 Mobile-based approach with NFC Scanner

Mobile based LMS have been proposed by A. Larsan Aro Brian et al. in 2014 [1]. They suggested that WiFi based Local Positioning System (LPS) and Near Field Communication (NFC) tags can be used together for locating books, as well as performing transactions like borrowing and return. Books can be borrowed out and returned without involving any help from librarians. To illustrate, users borrow books that are equipped with NFC tags by using NFC readers preinstalled in smartphones. After success borrow, the book can pass through the NFC reader at library entrance without arsing any alarm. Similar for return process, users only need to place the book inside a return box equpipped with NFC readers. The book can then be returned and overdue charges can be paid using the smartphones. This flexible design did provide a lot of convenience for readers and staff as this approach can save time for handling requests and provide users an easy access to library systems. However, comparing with traditional barcode, equipping NFC tags to millions of books in public library can be a high-cost event. Nowadays, most books in public library are equipped with barcodes. Scanning of barcodes provide a much easier and cheaper approach in term of implementation. In addition, camera are more commonly found in mobile devices, compared to NFC readers. It means that accessing barcode information is much easier than accessing information on NFC tags. NFC approach would not be beneficial to those readers without a NFC-enabled device. Therefore, we choose scanning barcode as input methods in our project.

2.2.2 Radio Frequency Identification

Radio Frequency Identification (RFID) is also another common technology used in automated system. RFID chip typically contains of two parts: integrated circuit that contains unique ID and an antenna that is acting as sender and transceiver of radio waves [7]. It is usually used with a RFID reader and a middleware software so that the reader can communicate with tags. Regarding the application in LMS, RFID tags are thin and flexible so that they can be placed inside the cover of each book, the information inside can be scanned by the RFID readers and transmitted to the middleware for further process when the library users wish to borrow or return any materials. Similar to the application of NFC, the whole process do not involve any help from library staff and it can save time for book circulation compared to traditional and manual approach. Furthermore, RFID readers allow scanning multiple books at a time, meaning that it save more time handling borrowing and return transactions. Nevertheless, privacy is one of the

concerns when implementing RFID-based system in library. Unathorised tag reading, writing, hotlisting, eavesdropping and tracking can be done by any unauthorized readers. To illustrate, security bit can be altered by readers, allowing them to leave the library without finishing check-out process. As a result, RFID used in library usually contain little information about the book, the information stored are just the same as the ID displayed on a barcode. In addition, RFID tags are 10x much more expensive than using paper-printed barcode. Therefore, we would choose to store the bookID in barcode in this project, and store other book information in separated databases.

2.3 Conclusion

After reviewing existing library management system, we have found that system proposed on different platform are having different advantages and disavantages. However, there is no system in the market that are compact and convenient to use. Therefore, we propose our system to be designed in web based, with mobile app serving as client and API serving as the bridge between client and database.

3. Review on technical components

3.1 Mobile Development platform

There are currently two popular development platform in the market: android and iOS, owned by Google and Apple respectively. On one hand, Android operates on open source ecosystem, which means that it allows tons of modifications and functionalities. On the other hand, Apple is restricted to a close ecosystem, which means that every content developed must follow regulations provided by Apple. After reviewing, there are good and bad for both approaches, the comparison is summarized as below:

	Android	iOS
Advantages	Cover more devices More flexible development	More control and stability App design takes less time
Disadvantages	Less secure More time for development and testing	More restrictions in development Support limited number of devices

Table 1 Comparison on Android and iOS

Since our project is related to book processing, covering a large number of devices is always our priority. Though creating android application takes more time as UI design guidelines are not provided like Apple does, the benefits of having less development restrictions outweight this drawback. Flexible development means encouraging innovative development, the system can support more features in long term. Therefore, we choose Android as our development platform.

3.1.1 Programming languages

In development of android application, Java and Kotlin are widely used. On one hand, Java is created in early 90s and having “C-like” syntax that makes developer easily to learn and use. On the other hand, Kotlin is introduced in 2011 and designed to be a concise and efficient alternative to Java. Their major differences are summarized as follows:

Java	Kotlin
Casting requires type checking according to operation	Support smartcast feature which automatically manage redundant cast
Attributing null to variables is allowed	Attributing null to variables stop code from compiling
Execptions need to be explicitly catch and declare	No catching or decaration of execptions is needed
Every Class, constructor, getter and setter needs to be explicitly coded	Support keyword ‘data’ for creation of object class, constructor, getting and setter

Table 2 Comparison on Java and Kotlin

From the table above, it is shown that kotlin is providing many convenient code-writing features for developers. Although those new features are considered to be time saving, time for development also depends on the supporting community. Since Java is having a longer history, it is having a larger support community and many problems and bugs have been discussed and solved before. The time needed for solving a new problem can hence be shorten. Furthermore, code in Java is more robust and straight forward, which enhances readability. Therefore, Java would be chosen for my project.

3.2 Database

SQL server owned by Microsoft and mySQL owned by Oracle are two most popular databases nowadays. They are both relational database management system (RDBMS) which can perform storage, update, retrieval of data by using SQL command. Their advantages and disadvantages are summarized as follows:

	SQL Server	MySQL
Advantages	Scalable performance On going development	Provides cross-platform support Support more programming languages Easy to use
Disadvantages	Support fewer platform Licensed and more costly	Lower scalability Syntax more complex to understand Lower stability

Table 3 Comparison on SQL Server and mySQL

According to research done by Justin Alecxander F. Ravago[4], mySQL is having stability issues and poor performance on scaling. When it comes to handling multiple operations at a given time, mySQL takes more time compared to SQL server. Since our application would cover a large number of target users and the data set involved is also large, making sure that updates of tables can be performed in shorter time is a major indictor when deciding suitable database. Meanwhile, SQL server is commonly picked between the two in business setting due to its enhanced security and data integrity. It has a higher coverage of APIs which making it more usable and flexible in development. As a result,we would choose SQL server as our database.

4. Proposed design, solution, and system

In this section, we will discuss the design of our proposed system, provide diagrams and explain how it will be tested and implemented.

4.1 Software architecture overview

Figure 1 Software architecture

Our proposed library management system is separared into two main parts, namely frontend and backend parts. Front-end, also called client-side, refers to the mobile application developed. It is the part where users interacts with. Back-end, also known as the server-side, refers to web service API and database. It is the part where the data holds and communication between database and client application begins.

4.2 Use-Cases Diagram

The following diagram concludes all the situations and functionalites that will be covered in this project. In short, the mobile application developed will separated into two interfaces, namely reader interface and staff interface. They both shared common functionalities like login and browsing book catalog. Each type of user will have access to different functions. For example, the readers are allowed to borrow and renew library materials. Meanwhile, the staff are allowed to manage book information, as well as checking reader information.

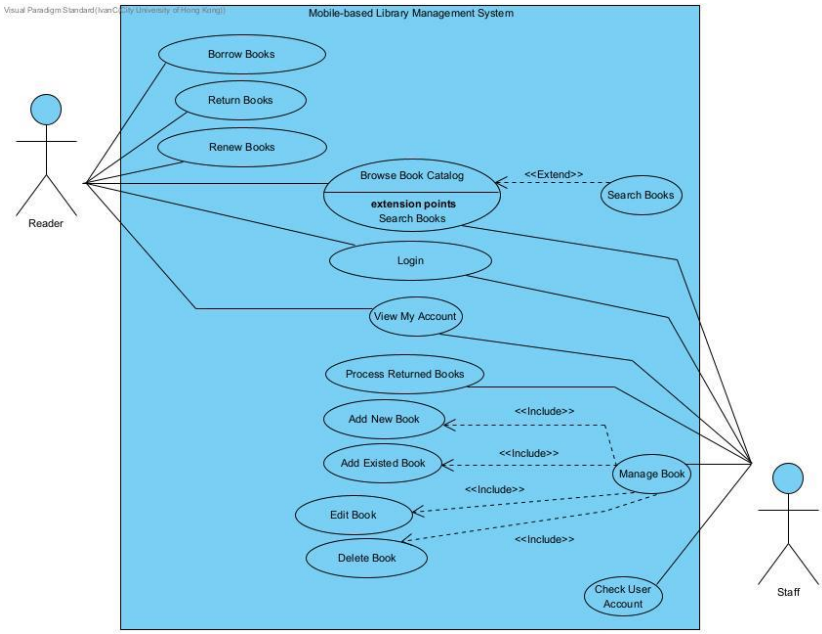


Figure 2 Use case diagram

4.3 Use-case Specification

There are a total of 12 use cases in this project. The use case specification and the interaction between user and the system are illustrated below:

Use Case Name	Borrow Book	
Actor	Reader	
Desciption	A use case for user to borrow book	
Reference ID	LMS01	
Typical course of events	Actor Action	System Response
	Step 1: User selects “Borrow or Return” function from main page	
		Step 2: The system jumps to the page for inputting barcode.
	Step 3: User press on the scan button	
		Step 4: The system opens the phone camera for scanning barcode
	Step5: User scans the barcode of the book	
		Step 6: The system checks the book copy status in database, to see if the book is available for borrowing
		Step 7: The system checks the borrowing quota of the user

		Step 8: The system update status of the book and borrowing history Step 9: The system display success message and provide corresponding borrowID and duedate
Alternative course of events	Step 6: If the book is not in available status, display Error Message "The book is invalid for borrow/return." Step 7: If the max borrowing quota of the reader is reached, display Error Message "The Reader account has reached his borrowing quota limit"	
Pre-Condition	User have logged in to the system	
Post-Condition	User successfully borrow the book	

Table 4 Usecase description of borrow book

Use Case Name	Return Book	
Actor	Reader	
Desciption	A use case for user to return book	
Reference ID	LMS02	
Typical course of events	Actor Action	System Response
	Step 1: User selects “Borrow or Return” function from main page	
		Step 2: The system jumps to the page for inputting barcode.
	Step 3: User press on the scan button	
		Step 4: The system opens the phone camera for scanning barcode
	Step5: User scans the barcode of the book	
		Step 6: The system checks the book copy status in database, to see if the book is available for return
		Step 7: The system checks the active borrowing history to see if the user have borrowed that book.
		Step 8: The system update status of the book and status of borrowing history

		Step 9: The system display success message and provide corresponding borrowID and duedate
Alternative course of events	Step 6: If the book is not in borrrowed status, display Error Message "The book is invalid for borrow/return." Step 7: If the borrowing history cannot be found, display Error Message "Cannot find borrowHistory"	
Pre-Condition	User have logged in to the system	
Post-Condition	User successfully return the book	

Table 5 Usecase description of return book

Use Case Name	Renew Book	
Actor	Reader	
Desciption	A use case for user to renew book	
Reference ID	LMS03	
Typical course of events	Actor Action	System Response
	Step 1: User selects “renew book” function from main page Step 4: The user selects the book he wants to renew Step 8: The user press “OK” to confirm the message	Step2: The system checks the list of books that the user borrow in database Step3:The system display the list of book Step 5: The system checks the renewal times of selected books Step 6: The system update the renewal times and due date in database Step 7: The system display success renewal message and show corresponding due date of the renewed book Step 9: The system refresh the renewal page.

Alternative course of events	Step 5: If the user have selected book with renewal times equal to 5, show error message “Some books have been renewal 5 times. Please select again.” Step 6: If the system fails updating record, display error message “FAIL Renewal for Book.” and the reason behind
Pre-Condition	User have logged in to the system
Post-Condition	User successfully renew the book

Table 6 Usecase description of renew book

Use Case Name	Browse Book Catalog	
Actor	Reader,Staff	
Desciption	A use case for user to view the book catalog, preforming searching and viewing deatail	
Reference ID	LMS04	
Typical course of events	Actor Action	System Response
	Step 1: User selects “browse catalog” from main page Step 4: The user enters keyword for searching Step 6: The user choose “Details” option to view full information about the book (including copies avaliable in different branches).	Step2: The system retrieve books information from database Step3:The system display the list of book Step 5: The system displays book consisting of the keyword the user typed Step 7: The system retrieve book information and copies information from database, display to users.
Alternative course of events		
Pre-Condition	User have logged in to the system	
Post-Condition	User get book information and copies information of a particular book	

Table 7 Usecase description of browse book catalog

Use Case Name	Login	
Actor	Reader,Staff	
Desciption	A use case for user to login to the system	
Reference ID	LMS05	
Typical course of events	Actor Action	System Response
	Step 1: User opens the application Step 3: User enters his account name and password and chooses “Login”	Step2: The system display the login screen to the user Step 4:The system checks the account name and password from the database Step 5: The system displays main screen to user
Alternative course of events	Step 3: If user leave either account name or password empty, show error message “Please enter Username and Password!” Step 4: If the account name and password provided by the user doesn’t match with database records, show error message “WRONG CREDENTIALS.”	
Pre-Condition	User have not logged in to the system	
Post-Condition	User successfully logged in to the system	

Table 8 Usecase description of login

Use Case Name	Add New Book	
Actor	Staff	
Desciption	A use case for user to add new book	
Reference ID	LMS07	
Typical course of events	Actor Action	System Response
	Step 1: User selects “manage resources” function from main page Step3: User chooses “add book” function Step 5: The user enters information of the new book and chooses “register new Bookid”	Step2: The system display the Book Management Screen and ask if the user wants to add or edit book Step 4: The system shows the form for entering book information Step 6: The system checks the ISBN and insert new Book to database
Alternative course of events	Step 6: If ISBN already exist in database, show error message “ISBN already exist”	
Pre-Condition	User have logged in and user wants to add a new book to the system	
Post-Condition	New book record registered in database	

Table 9 Usecase description of add new book

Use Case Name	Add Existed Book	
Actor	Staff	
Desciption	A use case for user to add copies for existed book	
Reference ID	LMS08	
Typical course of events	Actor Action	System Response
	Step 1: User selects “manage resources” function from main page Step3: User chooses “add book” function Step 5: The user selects the book title, location, enter the new barcode and pressed “add copies”	Step2: The system display the Book Management Screen and ask if the user wants to add or edit book Step 4: The system retrieves titles of existed books and list of location from the database and display it. Step 6: The system checks whether the new barcode already existed Step7: The system insert new book copies record into database.
Alternative course of events	Step 5: If the user leave fields empty or barcode length is shorter than 10, show error message. Step 6: If barcode provided is found in database, show error Message “Barcode already exist”	
Pre-Condition	User have logged in to the system and user wants to add book copies to the system	
Post-Condition	New book copies registered in database	

Table 10 Usecase description of add existed book

Use Case Name	Edit/Delete Existed Book	
Actor	Staff	
Desciption	A use case for user to edit information about existed book, or delete the existed book	
Reference ID	LMS09	
Typical course of events	Actor Action	System Response
	Step 1: User selects “manage resources” function from main page	
		Step 2: The system display the Book Management Screen and ask if the user wants to add or edit book
	Step3: User chooses “edit book” function	
		Step 4:The system provides a textbox for user to enter BarcodeID and a scan button for image input
	Step 5: User input the barcode	
		Step 6: The system retrieve book information from database using the barcode provided, and displays book information on the screen
	Step 7 : The user can modify the textbox and selects “Update This Book”	
		Step 8: The systems checks the ISBN and update the information of book copies to database

Alternative course of events	Step 7: If the users wishes to delete the book, user can select “delete book” function and status of book copies will be updated in step 8.
Pre-Condition	User have logged in to the system and user wants to edit information about existing book copies to the system
Post-Condition	Information about book copies is updated in database

Table 11Usecase description of edit book and delete book

Use Case Description		
Use Case Name	Check user account	
Actor	Staff	
Desciption	A use case for user to view the information about the reader, including personal information and records of book he borrowed.	
Reference ID	LMS10	
Typical course of events	Actor Action	System Response
	Step 1: User selects “Check Reader” function from main page Step3: User chooses the scan button for input Step 5: The user scan the reader ID	Step2: The system provides a textbox for user to enter ReaderID and a scan button for image input Step 4: The system opens the mobile camera for input Step 6:The system retrieve information about the reader (personal information and borrowing records) using the readerID provided. Step 5: The system displays reader Information
Alternative course of events	Step 5: If readerID cannot be found in database, show error message “Incorrect Input. readerID not exist”	
Pre-Condition	User have logged in to the system and user wants to check information about a reader	

Post-Condition	Information about reader is shown to user

Table 12 Usecase description of check user account

Use Case Description		
Use Case Name	Process return book	
Actor	Staff	
Desciption	This use case describe the process of handling return requests from readers	
Reference ID	LMS11	
Typical course of events	Actor Action	System Response
	Step 1: User selects “Process return” function from main page Step3: User chooses the scan button for input Step 5: The user scan the barcode on book	Step2: The system provides a textbox for user to enter ReaderID and a scan button for image input Step 4: The system opens the mobile camera for input Step 6: The system checks the status of the book Step 7: The system update the status of the book to available and display success message
Alternative course of events	Step 6: If the status of book is not “on return request”, show error message “The book is invalid for Return Processing.”	
Pre-Condition	User have logged in to the system and user wishes to put the returned book back to shelf (process return request).	
Post-Condition	The return request have been processed, status of book made available again	

Table 13 Usecase description of process return book

Use Case Description		
Use Case Name	View Account Information	
Actor	Reader,Staff	
Desciption	This use case describe the process of a user viewing information of his own account	
Reference ID	LMS12	
Typical course of events	Actor Action	System Response
	Step 1: User selects “My Account” function from main page	Step 2: The system checks the information of the log in account from database Step 4: The system display information about the user account
Alternative course of events		
Pre-Condition	User have logged in to the system and user wishes to check his own account.	
Post-Condition	The account information is displayed on screen.	

Table 14 Usecase description of view account information

4.4 Backend Design

In this section, we would explain the design of our database and the API being created.

4.4.1 Database Design & ER Diagram

The ER Diagram is constructed for building database schema, it contains six entities (Library,Book, BookCopies,Account,Reader,Staff) and their relationships are illustrated below:

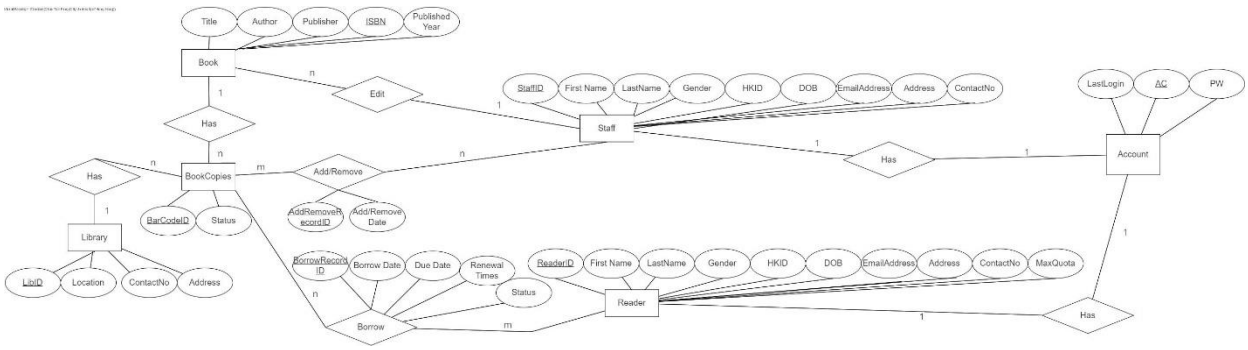


Figure 3ER diagram

The resulting schema is shown below:

Table Name	Columns(Primary keys are underlined and Foreign key are italic)
Book	Title,Author,Publisher,<u>ISBN</u>,PublishingYear,*EditStaffID*
Library	<u>LibID</u>,Location,ContactNo,Address
BookCopies	<u>BarcodeID</u>,Status,*ISBN*,*LibID* (*Status: 1 means on-shelf, 2 means borrowed-out, 3 means on return request, 8 means deleted)
Reader	<u>ReaderID</u>,FirstName,LastName,Gender,HKID,DOB,EmailAddress,Address,ContactNo,MaxQuota,*Ac*
Staff	<u>StaffID</u>,FirstName,LastName,Gender,HKID,DOB,EmailAddress,Address,ContactNo,*Ac*
Account	<u>Ac</u>,Pw,LastLogin
AddAndRemove History	<u>AddRemoveRecordID</u>,Date,*BarcodeID*,*StaffID*
BorrowHistory	<u>BorrowRecordID</u>,BorrowDate,DueDate,RenewalTimes,Status,*BarCodeID*,*ReaderID* (*Status: 1 means active record, 8 means completed record)

Table 15 Database schema

4.4.2 Stored Procedures Design

To perform different operations on different database tables, we have created 11 stored procedures in total, their functionalties and description are listed below:

Stored procedure	Details
Login	**Description:** This stored procedure is called when user logins their account. **Input:** Account, Password **Process:** It first check whether a matching can be found in table Account. If so, further check table Staff and table Reader to determine the account type. Finally, update LastLogin column in table Account **Output:** Account, LastLogin, AccountType on successful login Error Message 'WRONG CREDENTIALS' if match cannot be found
Borrow	**Description:** This stored procedure is called when user borrow a book **Input:** Account, Barcode of Book **Process:** It first check whether the barcode exist in table BookCopies. If so, it checks the borrowing record of the account from table BorrowHistory, compare the number of borrow records with max quota from table Reader. If quota doesn't exist, perform insertion to BorrowHistory table and update Status of Bookcopies to borrow-out(2). **Output:** Success State, BorrowRecordID, DueDate on successful borrow Error Message 'Reader account has reached borrowing quota limit' if quota is reached Error Message 'The Book is not available' if barcode not exist in table BookCopies
Return	**Description:** This stored procedure is called when user return a book **Input:** Account, Barcode of Book **Process:** It first check whether the borrow record exist in table BorrowHistory. If so, it checks whether the status of the book copies is borrowed out(2). Finally, update status of book copies to on-return-request(3) and update status of borrow history to completed(8). **Output:** Success State, BorrowRecordID, DueDate on successful return

	Error Message 'Cannot find borrowHistory' if borrow record cannot be found Error Message 'BookCopies Status is not Borrow out' if status of the book copies is not borrow-out (2)
ProcessReturn	**Description:** This stored procedure is called when staff process a returned book **Input:** Barcode of Book **Process:** It first check whether the barcode exist in table BookCopies. If so, update the status of the book copies to on-shelf (1). **Output:** Success State on success ErrorMessage if transaction error occurs
Renew	**Description:** This stored procedure is called when reader renew a book **Input:** Barcode of Book **Process:** It updates the borrowing history with new due date (14 days after old due date) and increase renewal times of that borrow record by 1. **Output:** Success State, new due date, title, barcode on success ErrorMessage, title, barcode if transaction error occurs
InsertBook	**Description:** This stored procedure is used for inserting a new book **Input:** Title, Author, Publisher, ISBN, Publishing Year, Staff Account **Process:** It first checks whether the ISBN already exist in table Book. If not, perform insertion to Book with the book information provided and the StaffId obtained from table Staff. **Output:** Success State on success ErrorMessage 'ISBN already exist' if ISBN already exist
InsertBookCopies	**Description:** This stored procedure is used for inserting a new copy for an existed book **Input:** ISBN, LibId, Barcode, Staff Account **Process:** It first checks whether the barcode already exist in table BookCopies. If not, perform insertion to BookCopies with the book information provided and insert a new record to table AddAndRemoveHistory with today date and staff id obtained from table Staff.

	Output: Success State on success ErrorMessage 'Barcode already exist' if barcode already exist
UpdateBook	**Description:** This stored procedure is used for updating book information and location of book copies **Input:** Barcode, Title, Author, Publisher, Publishing Year, Staff Account, Location **Process:** It first checks whether the barcode already exist in table BookCopies. If so, perform updates on table Book with the book information provided and staffId obtained from Staff table. Meanwhile, perform update on table Book Copies with the updated location. **Output:** Success State, Barcode, Title, Author, Publisher, Publishing Year, StaffId on success ErrorMessage 'Invalid Barcode. No update can be made.' if barcode cannot be found
DeleteBook	**Description:** This stored procedure is used for deleting a book copies **Input:** Barcode, Staff Account **Process:** It first checks whether the barcode already exist in table BookCopies. If so, it further checks the status of that book copies. If the status found is on-shelf (1), update status of Book Copies to 8 (deleted). Meanwhile, perform insertion to table AddAndRemoveHistory with the date of today and staffId obtained from table Staff. **Output:** Success State on success ErrorMessage 'Invalid Barcode. No delete can be made.' if barcode cannot be found
GetReaderInfo	**Description:** This stored procedure is used for obtaining information of a reader account **Input:** ReaderId **Process:** It first checks whether the reader already exist in table Reader. If so, retrieve the information of reader from table Reader. **Output:** Success State, ReaderId, First Name, Last Name, Gender, HKID, DOB, Email, Address, Contact No, Max Quota, Reader Account on success

	ErrorMessage 'readerID not exist' if readerId cannot be found
GetBookCopiesInfo	**Description:** This stored procedure is used for getting information about a book copy **Input:** Barcode **Process:** It first checks whether the barcode already exist in table BookCopies. If so, retrieve the information of book from table Book and location of book from table BookCopies. **Output:** Success State, Title, ISBN, Author, Publisher, Publishing Year, Location on success ErrorMessage 'Barcode not exist' if barcode cannot be found

Table 16 Stored procedure design

4.4.3 API Design

The API supports create, read, update and delete (CRUD) operations on database objects by using standard HTTP methods like GET, POST, PUT, DELETE. It accepts and returns JSON-encoded data. Their endpoints are summarized as below:

Method	Endpoint	Description	Related Stored Procedure
GET	/	Response with 'Express hosting on azure' to show service is running	N/A
GET	login/{ac}/{pw}	Authenticate account object	Login
PUT	borrow/{ac}/{barcode}	Perform borrow operation	Borrow
PUT	ret/{ac}/{barcode}	Perform return operation	Return
GET	borrowedbooks/{ac}	List all borrowed books	N/A
PUT	renew/{barcodeID}	Perform renewal operation	Renew
GET	books	List all books	N/A
GET	bookcopies/isbn/{isbn}	List all bookcopies with the provided ISBN	N/A
GET	acs/{ac}	List account information with the provided account name	N/A
GET	readers/{id}	List read information with the provided readerID	GetReaderInfo

PUT	processret/{barcode}	Perform return processing on a book	ProcessReturn
POST	book	Create a new book object	InsertBook
GET	libs	List all libraries	
POST	bookcopies	Create a new copy for an exised book	InsertBookCopies
GET	bookcopies/{barcode}	List book copies information by a given barcode	GetBookCopiesInfo
PUT	book	Update book information	UpdateBook
DELETE	bookcopies/{barcode}/{ac}	Delete a book copy with the provided barcode	DeleteBook

Table 17 API design

4.5 Frontend Design

There are a total of 14 activities in our android application. The control flow of the program is illustrated in the diagram below:

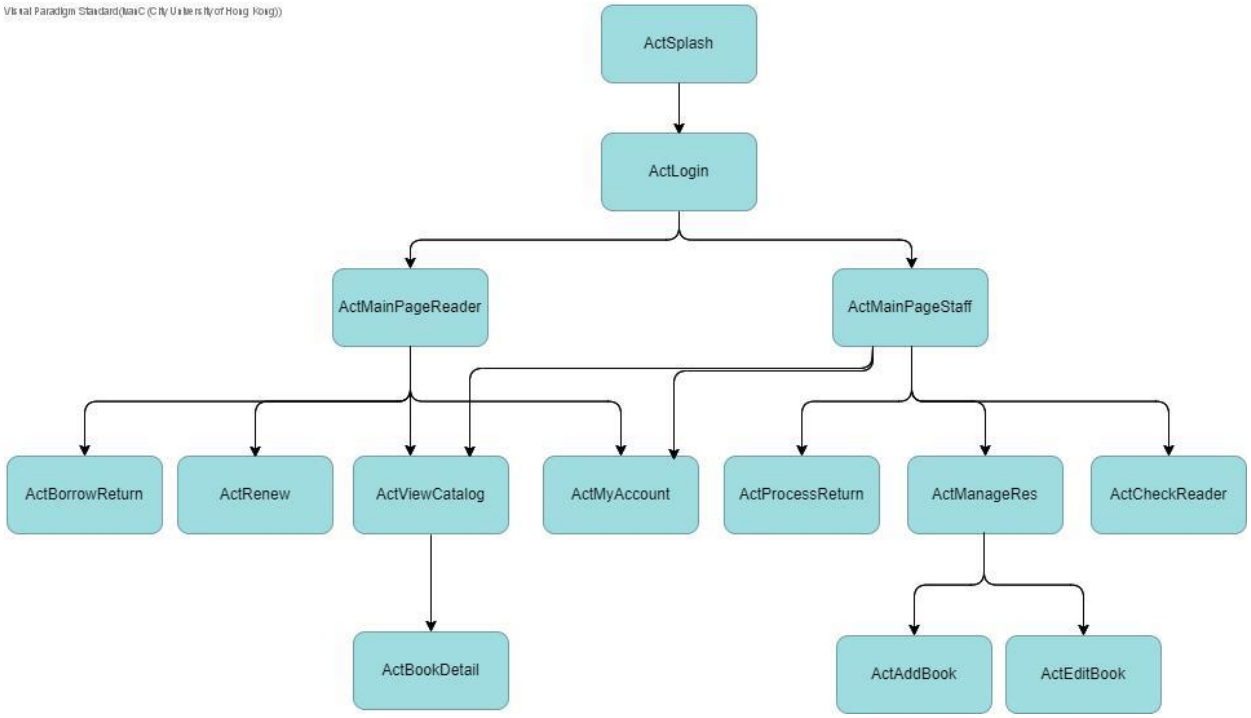


Figure 4Control flow of activities

4.5.1 User Interface & Functionalities

In the following, we would explain the functionalities of each activity.

ActSplash

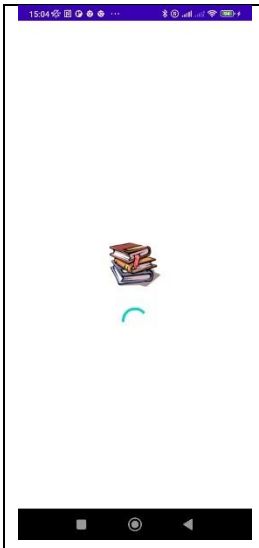


Figure 5Splash screen

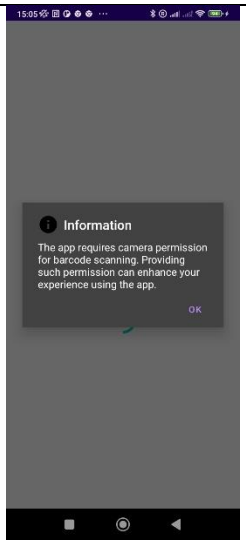


Figure 6Reminder of permission needed

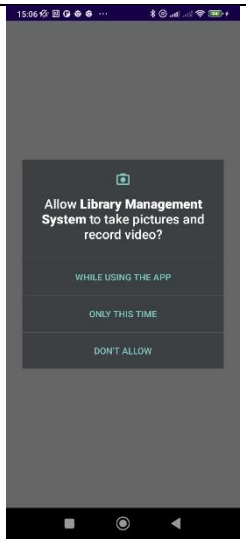


Figure 7Requesting permission

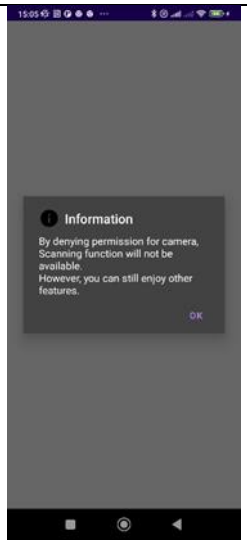


Figure 8Inform users after denying permission

This is the first activity launched when the application starts. During creation of this activity, we would initialize Retrofit client, the external libraries we used for exchanging data via web service, and ask the user for permission to use the build-in camera. We provide justification for the reason behind and remind user that allowing the permission would enhance user experience if the user denies it. After that, the application redirects to the login page.

ActLogin

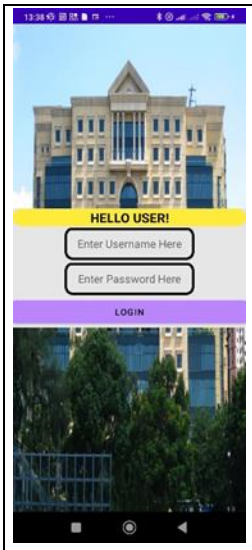


Figure 9 Login screen

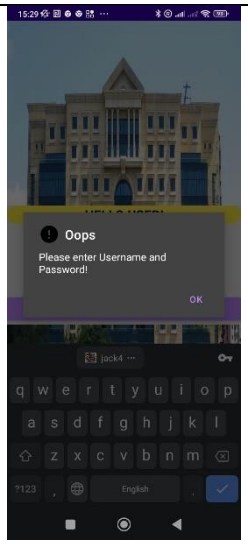


Figure 10 Login with empty fields

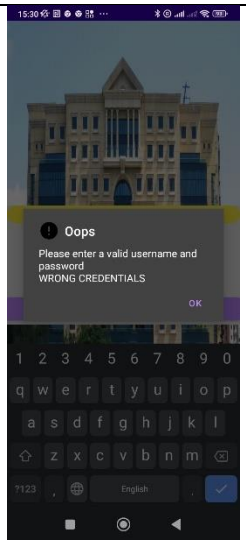


Figure 11 Login with incorrect credentials

This is the activity where the user logins. User can type their username and password to login. Upon pressing the login button, we would check for missing fields and ask the user to fill it if there are any. Otherwise, HTTP GET request would be sent out to endpoint login/{ac}/{pw} of API, with account and password providing as parameters. If the API respond with a success state, user would be redirected to main page. Otherwise, a 'WRONG CREDIENTALS' error message is shown.

ActMainPageReader & ActMainPageStaff

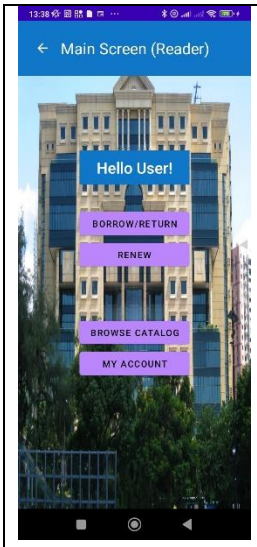


Figure 12 Main screen for reader

Figure 13 Main screen for staff

This is the activity shown after users have successfully login. If the account type is a reader, we would display the ActMainPageReader. If the account type is a staff, we would display the ActMainPageStaff. This Page allows users to select different operations available in our system.

ActBorrowReturn

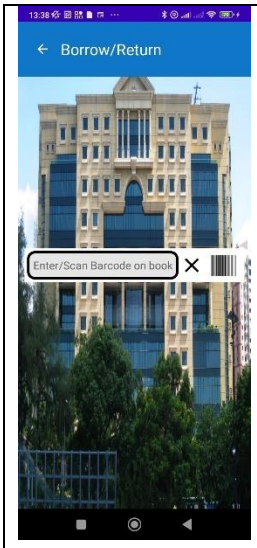


Figure 14 Borrow & return screen

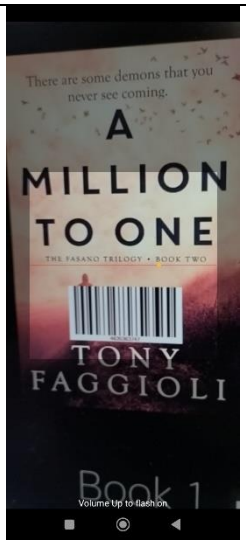


Figure 15 Scanning book barcode

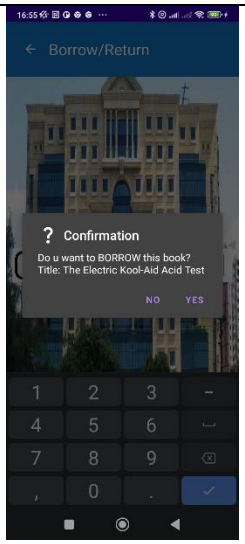


Figure 16 Confirmation of borrowing

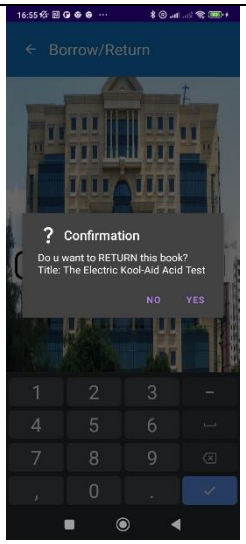


Figure 17 Confirmation of return

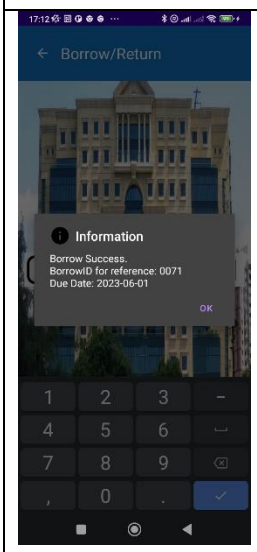


Figure 18 Information after success borrowing

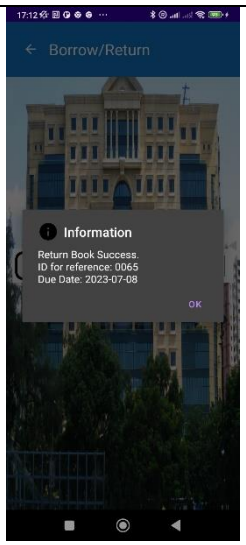


Figure 19 Information after success return

This activity is activated by choosing “Borrow/Return” from ActMainPageReader and it is available only for reader account. The user can borrow or return a book by pressing the barcode button. Our application would open the camera for barcode scanning. After the scanning result is received, HTTP request will be sent to bookcopies/{barcode} endpoint of API for obtaining the status of the book. If the status is “on-shlef”, we would ask for confirmation of borrowing. If the

status is “borrowed-out”, we would ask for confirmation of return. Borrow and return results will be shown afterward.

ActRenew

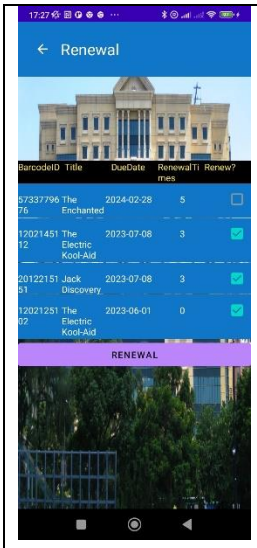


Figure 20 Renewal Screen

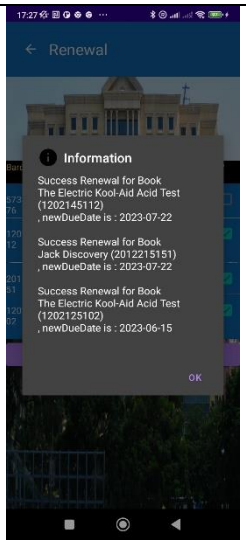


Figure 21 Information after success renewal

Figure 22 Updates made after renewal

This activity is activated when the user selects “Renew” on ActMainPageReader. Similar to Borrow/Return, this activity is only accessible from a reader account. Upon creation of the activity, login username would be passed to borrowedbooks/{ac} endpoint of API for retrieving list of borrowed books and related information. Information like book titles, barcodes, due dates and renewal times will be displayed on the listview, such that the user can choose the book they would like to renew. Upon selection and pressing renewal button, the application would check for renewal times. If there is no selected book with renewal times equal to 5, HTTP PUT request will be sent to renew/{barcodeID} endpoint of API for performing renewal on each book selected. After success renewal, success message would be displayed and ActRenewal is recreated afterward.

ActViewCatalog & ActBookDetail

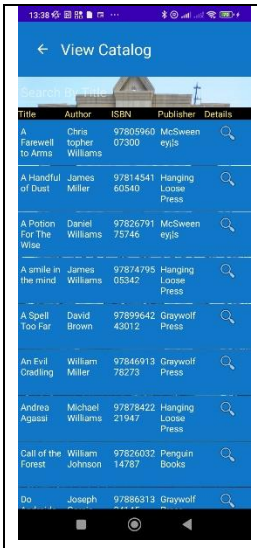


Figure 23 View catalog screen

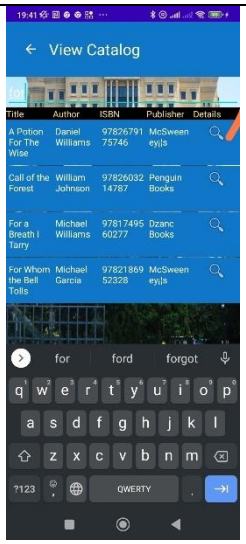


Figure 24 Catalog searching

Figure 25 View book detail

ActViewCatalog is accessible by both reader and staff users from the corresponding ActMainPage. During creation of this activity, the application sent a GET request to books endpoint of API for retrieving list of books in the library, all the books and related information would be displayed on the list view. Meanwhile, a edit text box is provided on top of the list view so that users can filter the list by typing keywords. Furthermore, users can click an item on the view and ActBookDetail will be activated for displaying additional information like status of copies in different branches.

ActMyAccount

Figure 26 View my account

ActMyAccount is accessible from both ActMainPageReader and ActMainPageStaff. When the application creates this activity, HTTP GET request will be sent to acs/{ac} endpoint of API for retrieving account information. The retrieved information is then displayed to user.

ActProcessReturn

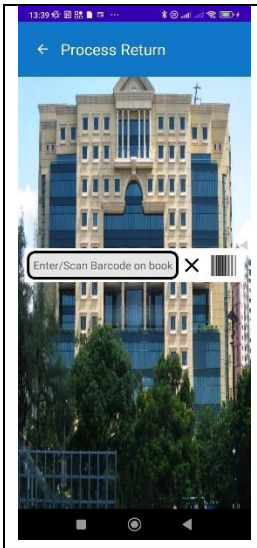


Figure 27 Process return screen

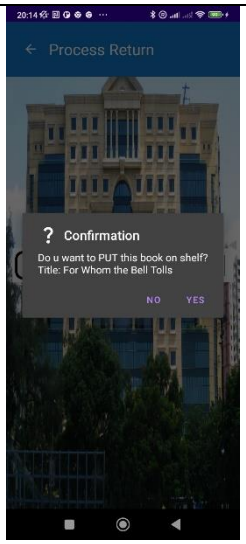


Figure 28 Confirmation for processing return

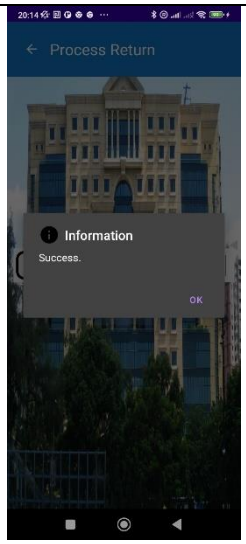


Figure 29 Information after success return processing

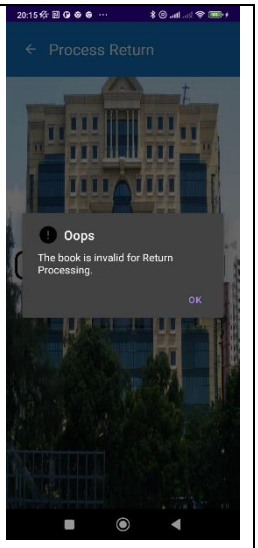


Figure 30 Error in processing return

This activity can only be accessed from ActMainPageStaff. Similar to ActBorrowReturn, user can press on scan button for inputting barcode. After the barcode is detected, the application would send a HTTP GET request to bookcopies/{barcode} endpoint of API for obtaining status of the book copy. If the status is “on return request”, the application would ask the user for confirmation. Another HTTP PUT request would be sent to processret/{barcode} of API for carrying out “process return” operation after confirmation. Alternatively, if the status is not “on return request”, the application would display an error message.

ActManageRes

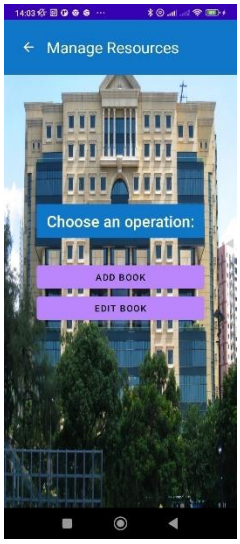


Figure 31 Manage Resources screen

This activity is only accessible from ActMainPageStaff. It is a transitional page when user chooses “Manage Resources” on ActMainPageStaff. The user can select “Add Book” or “Edit Book”, and the application would redirects the user to ActAddBook and ActEditBook respectively.

ActAddBook

Figure 32 Screen for adding books

Figure 33 nformation after adding new book

Figure 34 Spinner refreshed for adding book copies

Figure 35 Adding Existed Book

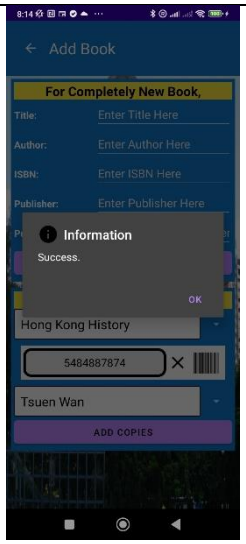


Figure 36 Success information after adding existed book

This activity is accessible from ActManageRes. The upper half is provided for user to create a new book while the lower half is provided for user to insert a new copy for existed books. When this activity is created, HTTP GET request would be sent to books and libs endpoints of API for obtaining list of book titles and list of library locations respectively. After the data is received, the spinner in the lower half would be initialized.

To add a new book, the user needs to enter title, author, ISBN, publisher, publishing year and press the “Register New Book ID” button. When the button is pressed, we capitalize the first letter of every words in field Title and Author. We also perform validation checks on ISBN and Publishing Year such that length of ISBN is 13 digits and Publishing Year is between 1900 and 2023. Finally, the provided information are encoded as JSON object and sent via HTTP POST request to book endpoint of API. Success message would be shown after creation and the spinner responsible for book titles would also be updated accordingly.

To add an existed book, the user needs to choose the book title in the first spinner and location in the second spinner. Meanwhile, an edit text box is provided for entering barcode of the new book. After all the fields have been filled, pressing the “add copies” button would send a HTTP POST request to bookcopies endpoint of API. The application would print a success message when the response body of the JSON object received contains a “success” transaction state.

ActEditBook

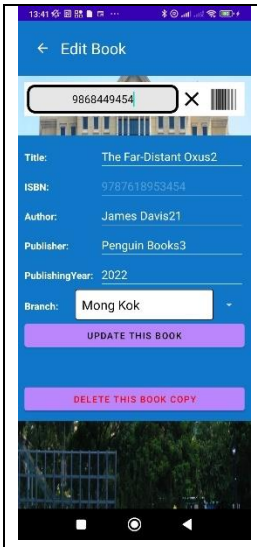


Figure 37 Edit book screen

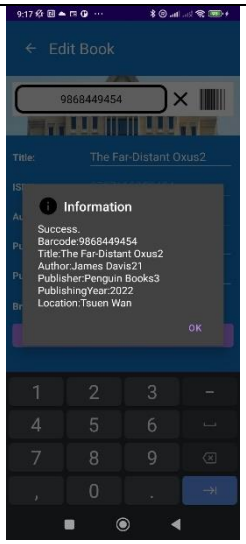


Figure 38 Information after successful update

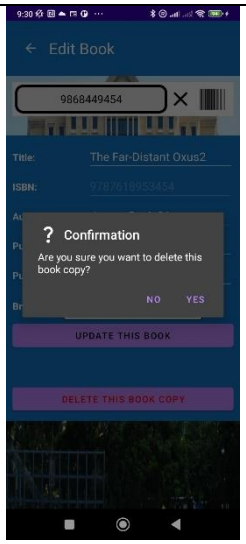


Figure 39 Confirmation on deletion

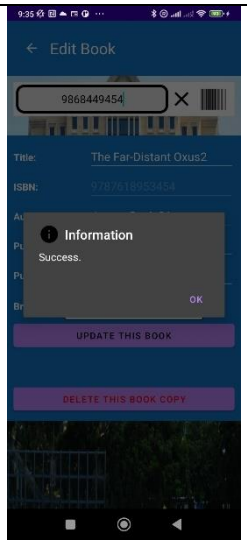


Figure 40 Information after successful deletion

This activity is activated when the user selects “Edit Book” on ActManageRes. It is created for staff users to update any book in the library. When our application creates this activity, HTTP GET request would be sent to libs endpoint of API for retrieving branch information and initializing the spinner representing branch. In order to update a book, the user needs to input the barcode of the book, either by manual input or by scanning via camera. After the edit text box is filled with 10 digits, HTTP GET request would be sent to bookcopies/{barcode} endpoint of API for retrieving information about the book copy. The retrieved information would be displayed and user can make changes on them. The input data together with the barcode will be encoded as JSON format and sent via HTTP PUT request to book endpoint of API when the “Update” button is clicked. Message and the updated results would be displayed after the response body of JSON object is received.

Apart from editing information about a book, users are also allowed to delete a book copy. A confirmation dialog would be displayed when user press the “Delete” button. A HTTP DELETE request with barcode and username support as parameters would be sent to bookcopies/{barcode}/{ac} endpoint of API after confirmation. Success message would be printed out after receiving response object with successful transaction state.

ActCheckReader

Figure 41 Screen for checking reading account

This activity is activated by choosing “Check Reader” from ActMainPageStaff. User can input reader ID he would like to query by manual typing or barcode scanning. After the edit textbox is filled with 4 digits, HTTP GET request would be sent to readers/{id} endpoint of API for retrieving information about a reader. Those information would be displayed afterward.

4.5.2 Class Diagram

Our project folder contains one single java source folder composing of two separate folders, namely Activity and WebServices. All the java classes related to activities are located in Activity folder and all the java classes related to web services objects are located in WebServices folder. The detailed class and methods are illustrated as follows:

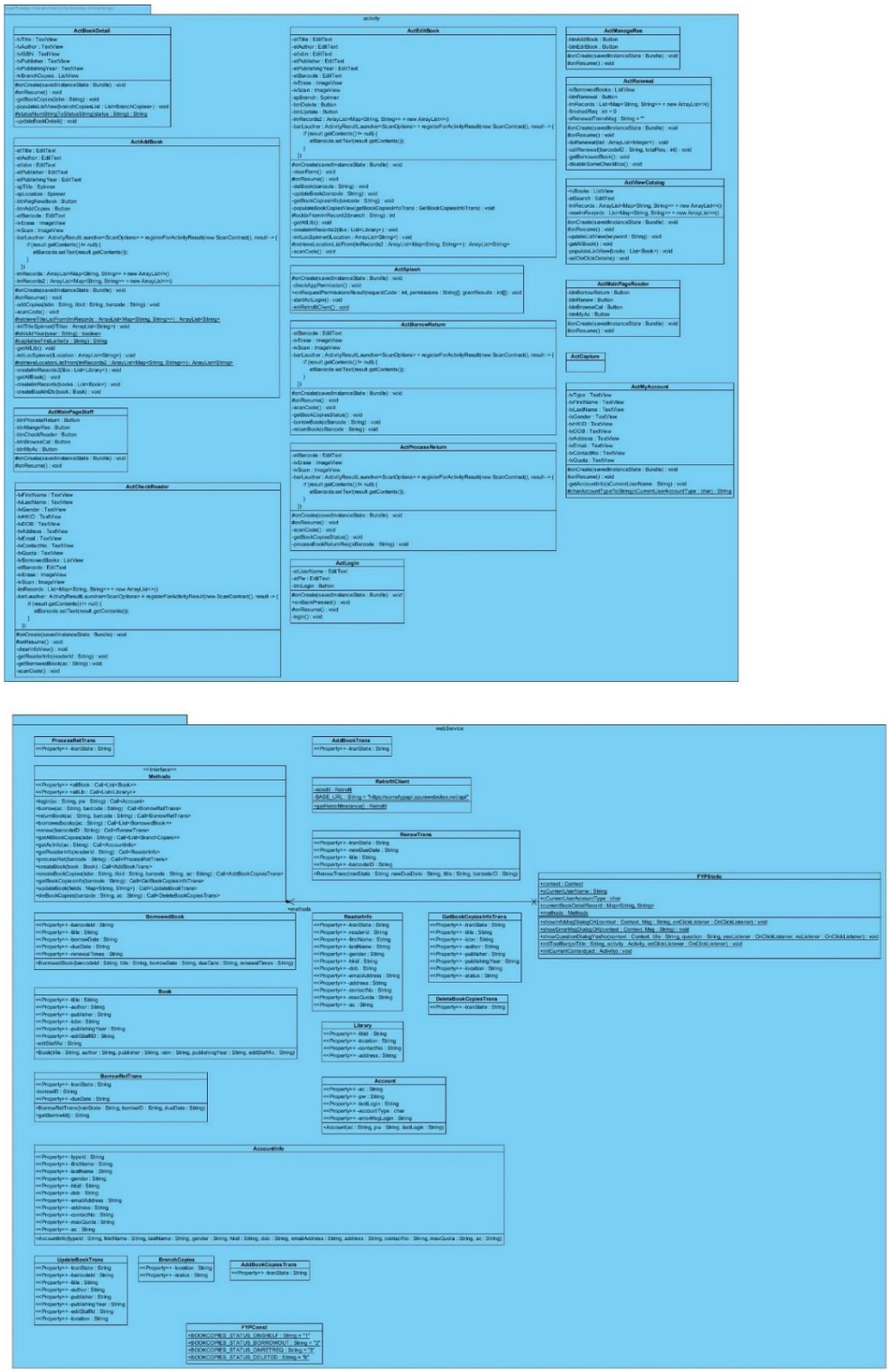


Figure 42 Class diagram

Every activity class represents a single screen in the application. Within every activity, all the layout elements are presented as attributes of that class. During on creation of a certain activity, all these attributes would be initialized, and their behaviour would be set according to the intended interactivity with the user.

In order to interact with the API service we created, we provide a class called methods in our web services folder. All the intended API calls are defined as java functions within this class. We made use of a library named Retrofit, which enables us to retrieve and upload data to our created API. On the other hand, all the JSON objects we received from web services are also presented as java class within the web services folder. For example, the class named library is a prototype of java object we are going to create when we retrieve list of libraries JSON object from the API.

5. Implementation and methodology details

Implementation of this project is subdivided into two main parts: frontend development and backend development. Methodologies and Timeline for implementations are discussed in this section.

5.1 Software development methodologies

In this project, we choose waterfall model as the development methodology. We perform every software development phase one by one such that phases does not overlap with each other.

Compared to other methodology like agile, water fall model provides clear and precise structure for development. It specifies tasks to be finished in every period of time. It is suitable for our project because our project has very clearly defined scope and the final product is much known from the beginning. Furthermore, there are no significant changes made during development and the time for this project is fixed into one year time period, impling that methodologies like spiral model which specialized in risk management and agile model which specialized in adapting to changes are not suitable.

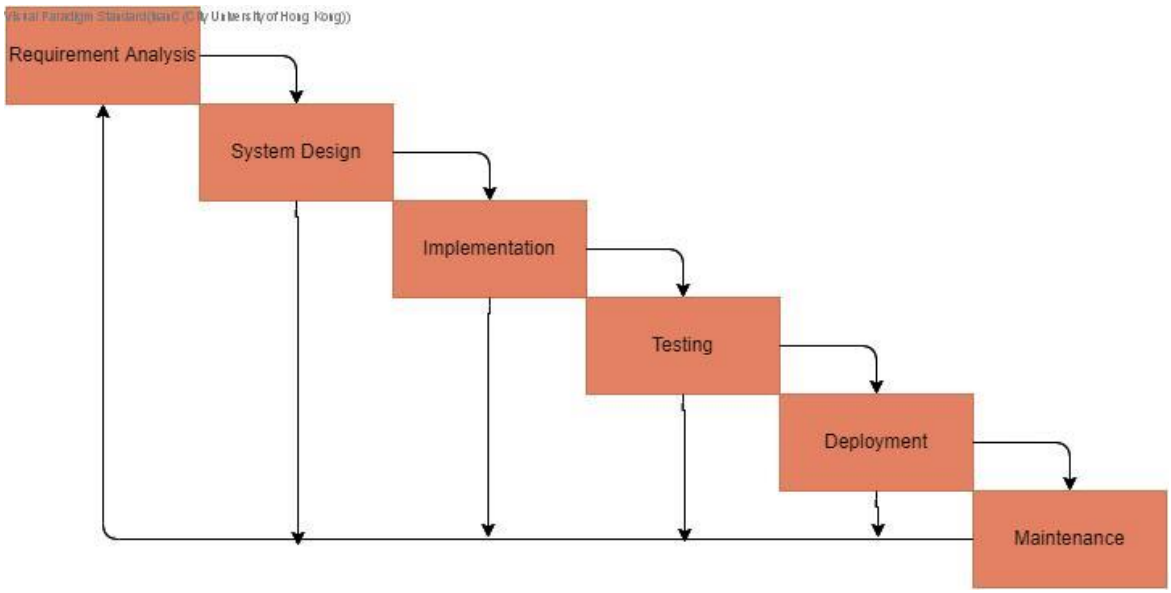


Figure 43 Waterfall model

5.2 Development Timeline

To implement the whole system, we break it down into three main parts: database, API and the android application. Database table and corresponding data are being created first. After the database is set up, stored procedures, API and android application are programmed stimultaneously. In order to make the project manageable, the project is subdivided into different smaller modules according to the functionalities. In addition, two weeks time are spent for finishing a function. All the functions together takes about 3 months time and the project is finished in late February. Testing is carried out on March and deployment is done on April. For detailed timeline and monthly log, they are included in Appendix A and B respectively.

5.3 Tools & Language Used

5.3.1 Database Setup

Our database is set up using Microsoft SQL Server Management Studio. It is the environment where we create tables, insert records, create stored procedures and perform testing and debugging. All the SQL scripts are manually typed one by one, except for the one which inserts new records. For the script used for insertion, we generate the initial value by randomizing values in Python.

5.3.2 API

For API, Visual Studio Code is chosen as the integrated development environment. We used NodeJS as the java runtime environment and used ExpressJS framework for server setup and creating Restful API. On top of that, we used java module named mssql for establishing connection with the database.

5.3.3 Android Application

For mobile application, Android Studio 4.2 is chosen as the integrated development environment. Extensible markup language (XML) is used for creating user interfaces and Java is used for controlling objects interactions and interactive components.

5.3.4 Version Control

For both API and android application, updates and backup of code are monitored using git. We created local git repository and commit all the changes when updates are made. Meanwhile, all the updates are also pushed to the online platform GitHub. Not only does this prevent any code

from losing, it also facilitates development as we can transvered our code back to any previous version.

5.4 Testing Procedure and results

5.4.1 Overview

During implementation of mobile application, Junit test cases are created for functions with a return value. For every activity, unit testing is carried out to test whether the program output matches desired results by using assertions. On the other hand, the API we developed is validated using Postman. For every endpoints we set up, requests are sent out using Postman to see if we can retrieve the desired outcome.

After all the components are implemented, integration test and system testing are carried out. Possible scenarios and expected outcomes are drafted out to see if our deliverable satisfy desired results. Details are included in following section.

5.4.2 Blackbox testing

In this project, we adopt blackbox testing technique. It means that we test against the specification of the program and assuming that tester have no prior knowledge about the system code. This method is suitable because it takes less time compared to white box testing. Also, it focuses on functionality of the program. Since our project is intended to provide easy access to library resources for users, user experience is more important compared to backend design. In order to generate test cases, we perform equivalence class partitioning and group similar inputs according to the expected outputs. For detailed test results, please refers to appendice C, D, E.

5.5 Deployment

API and database are hosted on Microsoft Azure after successful implementation and testing. It is a public cloud computing platform which supports service hosting so that users can perform different operations on database via the Internet.

6. Summary of achievements

Before working on this project, I have no knowledge about mobile application and web API development. Throughout the two semesters, I keep on researching different technical components related to frontend design and backend development. Other than that, I have also consulted some of my supervisors in my placement company, so that I can successfully carry out the project.

Overall, this project has given me chances for building up the ability to learn something individually, which is essential for my future career. Not only does this process equips myself with problem solving skills, it also equipped me with soft skills like presenting a software product and discussing a solution with others. These experiences are highly valuable as they are fundamental skills for any software developers.

7. Futher Improvement

Due to limited time constraints, lots of the functionalities of our system can be enhanced but they are not implemented.

For example, the inclusion of images in our android application. To make the interface more appealing and more user friendly, we can include images of books, accounts and libraries. Those images can be stored in a computer serving as the file server. Meanwhile, the file paths can be stored in columns of tables for retrieval.

Another improvement can be made in terms of security. In current implementation, we directly stored all the user passwords in database tables and made direct comparison when user logins. This could made password vulnerable to attackers. A better approach would be storing the hashed password and salt in the table. Whenever user logins, we attach the salt and applies one-way encryption alogithm like SHA1 on the password. We made comparison on hashed password rather than the raw password. This can enhance the security of our system.

8. References

[1] A. Larsan Aro Brian, L. Arockiam, and P. D. Sheba Kezia Malarchelvi, "AN IOT BASED SECURED SMART LIBRARY SYSTEM WITH NFC BASED BOOK TRACKING," *International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)*, vol. 11, no. 5, pp. 19–21, Nov. 2014.

[2] A.Thendral Mary, S.Ramya, Mr.S.Krishna Murthy, and Dr.A.Valarmathi, "ENHANCED LIBRARY MANAGEMENT SYSTEM," *International Journal of Creative Research Thoughts (IJCRT)* , 04-Oct-2017. [Online]. Available: https://ijcrt.org/papers/IJCRT1704024.pdf. [Accessed: 10-Oct-2022].

[3] D. G. Marcel, "Development of an online Integrated Student Management Information System: Case Study 'University of Gitwe," *International Journal of Advanced Research in Computer Science*, vol. 10, no. 5, pp. 59–67, 2019.

[4] Justin Alecxander F. Ravago Lyceum, *Comparison of MySQL and MS SQL Server*, 2019.

[5] K. Chinmai Devi, Kavitha Chaduvula, V. Rasagna, Shaik Kathija, and P. Santhoshi RupaDevi, "LIBRARY INFORMATION MANAGEMENT SYSTEM USING KIOSK,") EPRA International Journal of Research and Development (IJRD), vol. 6, no. 7, pp. 412–423, Jul. 2021.

[6] Mardiana and Meizano Ardhi Muhammad, "IKiS Self Service Kiosk for Library Service," *3rd International Conferences on Information Technology and Business (ICITB) , 7th Dec 2017*, pp. 137–142, Dec. 2017.

[7] NEERAJ KUMAR SINGH and PREETI MAHAJAN, "APPLICATION OF RFID TECHNOLOGY IN LIBRARIES," *International Journal of Library and Information Studies*, vol. 4, no. 2, pp. 1–9, 2014.

[8] N. Kurniasih, Sujito, Yulianti, A. Sudirman, N. Agustini Damayani, J. Paing, A. Dewi Kuraesin, J. Sugiono, G. S. Achmad Daengs, and F. Jati Nugroho, "The analysis on utilization of UNPAD library management system by end-users using the technology acceptance model," *Journal of Physics: Conference Series*, vol. 1175, p. 012229, 2019.

[9] S. Sharma, S. Mishra, S. Gupta, and S. Kumar, "Library Management System," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 5, pp. 889–893, 2022.

[10] Shanmugam A.P, Ramalakshmi, A, Sasthri, G, and Baalachandran, S, “Library Management System,” *JOURNAL OF XI'AN UNIVERSITY OF ARCHITECTURE & TECHNOLOGY*, vol. XII, no. XI, pp. 743–753, Dec. 2020.

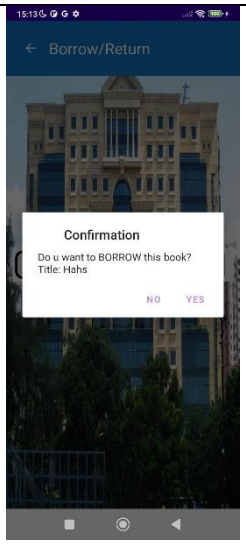
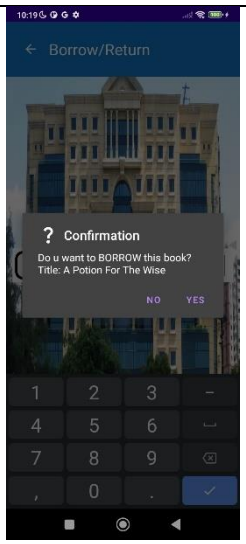
Apendices A: Project Timeline

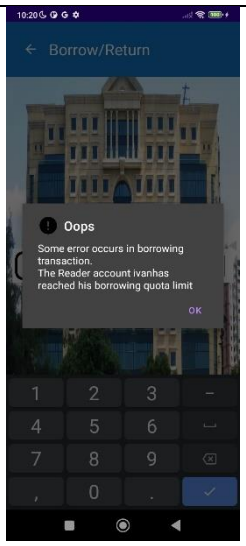
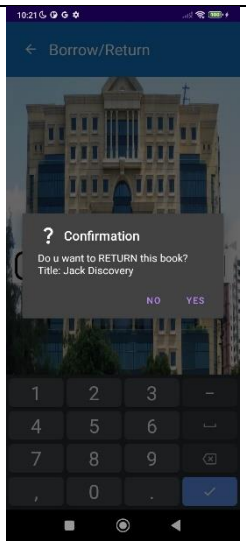
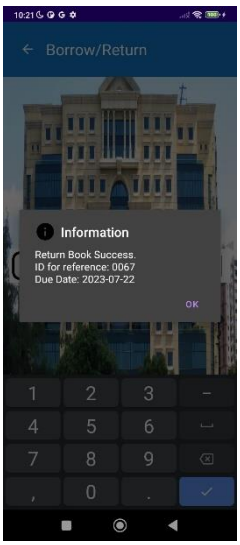
ID	Title	Start Time	End Time
1	Database Setup	11/15/2022	11/29/2022
2	Login Function	11/29/2022	12/15/2022
3	Borrow and Return Function	12/15/2022	01/05/2023
4	Renewal Function	01/05/2023	01/12/2023
5	View Catalog Function	01/12/2023	01/19/2023
6	View Account Detail Function	01/19/2023	01/26/2023
7	Prepare for Interim Report II	01/22/2023	02/05/2023
8	Add Book Function	01/26/2023	02/09/2023
9	Edit Book Function	02/09/2023	02/23/2023
10	Creating Test Cases	02/23/2023	04/23/2023
11	Deployment	04/23/2023	05/07/2023
12	Prepare For Final Report	05/07/2023	06/04/2023
13	Prepare For Final Present/Demo	06/25/2023	07/16/2023

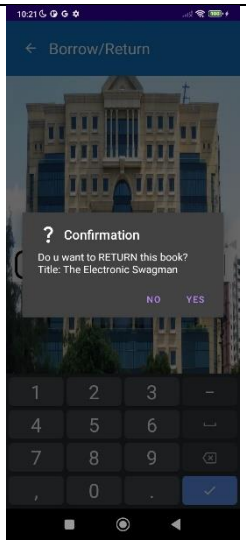
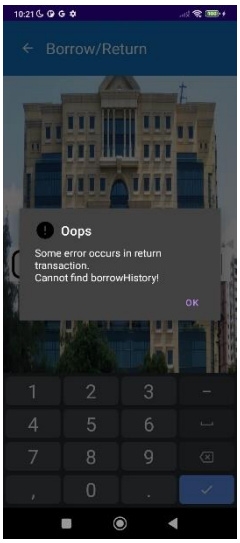
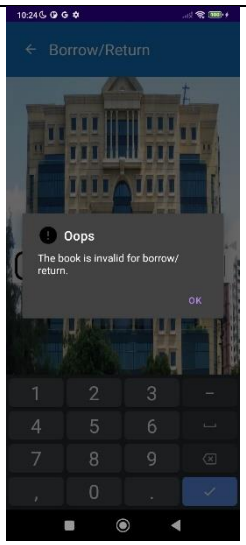
Apendices B: Monthly Log

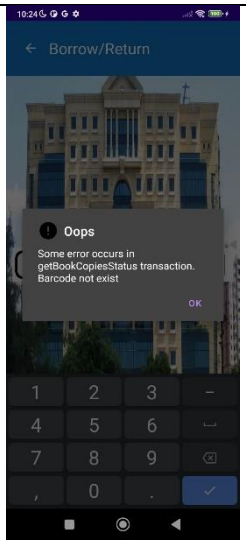
October	● Use case diagram finished ● ER diagram Database design finished ● UI design drafted
November	● Database schema ready (from ER diagram) ● Script for creating tables ready ● Study Android Application Development (different UI components)
December	● Database set up finished ● Work on Establishing connection between database and api ● Create UI and apply stylings
January	● Create basic functionalities for the application (login, borrow, renewal) ● Create some stored procedures and perform data extraction and retrieval from database
February	● Functions for Staff & Customer interface finished ● Start to prepare test cases for testing
March	● Test case generation finished ● Finetuned and ajustment to application
April	● Deployment finished

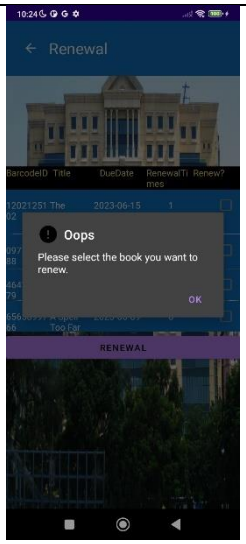
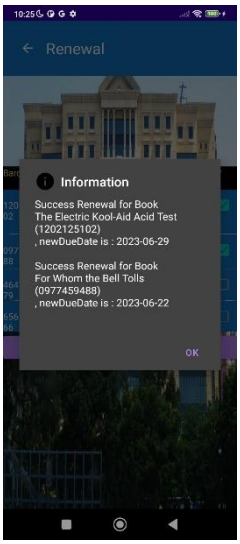
Apendices C: Testcase – Reader Interface

Items	Test case	Expected result	Actual result
Part A	Borrow and Return		
A1	Reader borrow a book by entering barcode of book copies with status 1 (on-shelf)	Ask for confirmation 'Do u want to BORROW this book?' and show book title 'Borrow Success' , borrow ID and due date are shown in dialog box after clicking Yes	 
A2	Reader borrow a book by entering barcode of book copies with status 1 (on-shelf) when borrowing quota is reached.	Ask for confirmation 'Do u want to BORROW this book?' and show book title Error message 'The reader account has reached its borrowing quota limit' is shown in dialog box after clicking Yes	

			
A3	Reader return a book by entering barcode of book copies with status 2 (borrowed-out) and the borrower is the current user	Ask for confirmation ‘Do u want to RETURN this book?’ and show book title ‘Return Success’ , return ID and due date are shown in dialog box after clicking Yes	 

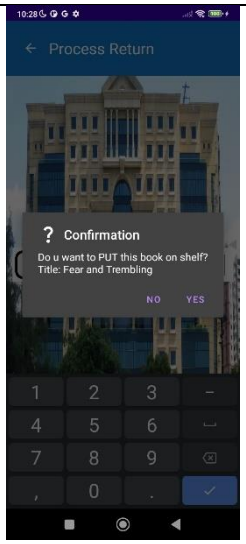
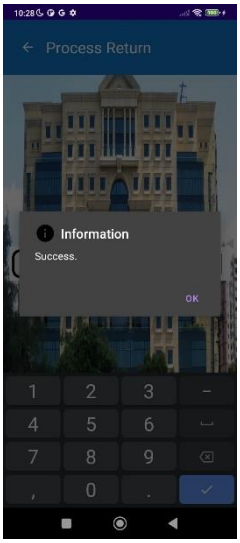
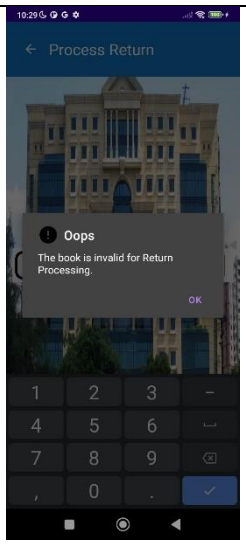
A4	Reader return a book by entering barcode of book copies with status 2 (borrowed-out) and the borrower is not the current user	Ask for confirmation ‘Do u want to RETURN this book?’ and show book title Error message ‘Cannot find borrowHistory’ is shown in dialog box after clicking Yes	 
A5	Reader enters barcode of book copies with status 3 (on-return-request)	Error message ‘The book is invalid for borrow/return’ is shown in dialog box	

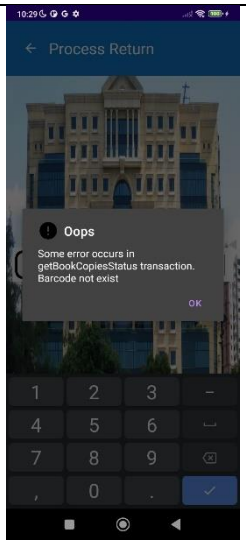
A6	Reader enters invalid barcode	Error message ‘Barcode not exist’ is shown in dialog box	

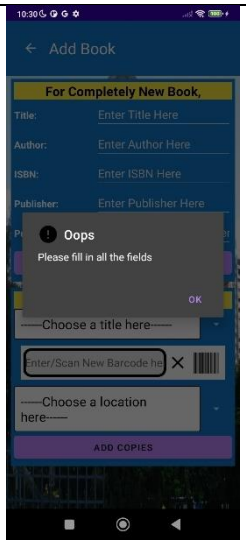
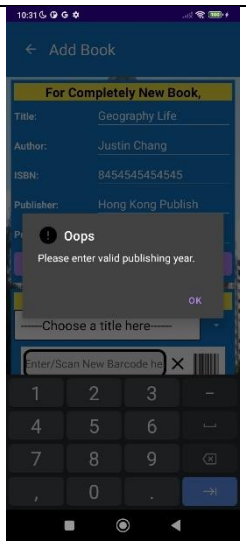
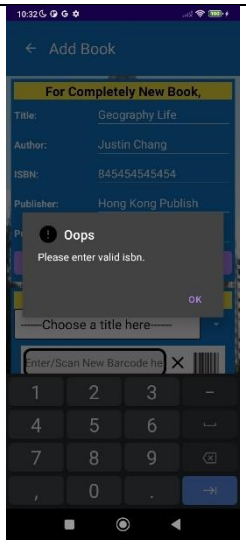
Part B	Renew		
B1	Reader renews no books	Error message ‘Please select the book you want to renew’ is shown in dialog box	
B2	Reader renews 2 books including book that have been renewal 5 times	Error message ‘Some books have been renewal 5 times. Please select again.’ is shown in toast message.	
B3	Reader renews 2 books and those books are renewal less than 5 times	Success message, book title, barcode, and new due date are displayed in a dialog box. The renewal activity refreshes afterwards.	

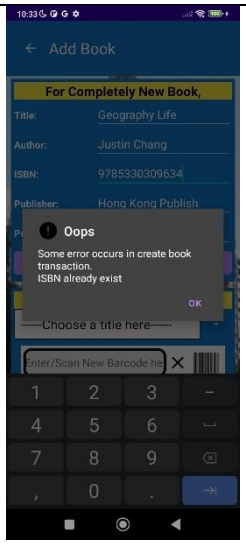
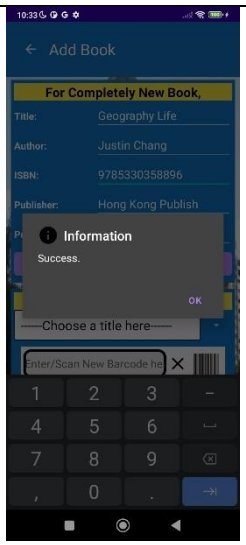
			 10:25 Renewal BarcodeID Title DueDate RenewalTimes Renew? 12021251 02 The Electric Kool-Aid 2023-06-29 2 09774594 88 For Whom the Bell 2023-06-22 1 46476979 79 Hahs 2023-06-08 0 65658997 86 A Spell Too Far 2023-06-09 0 RENEWAL

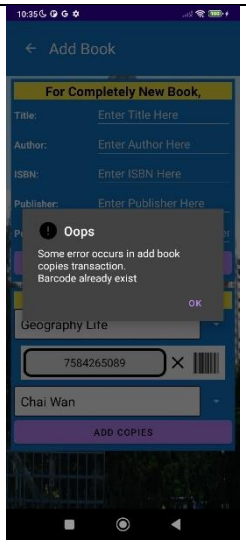
Apendices D: Testcase – Staff Interface

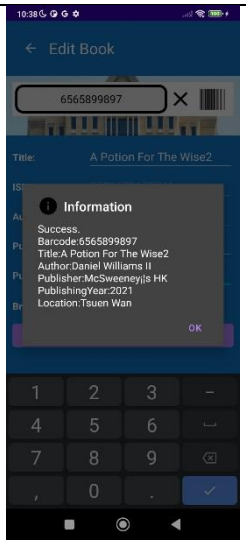
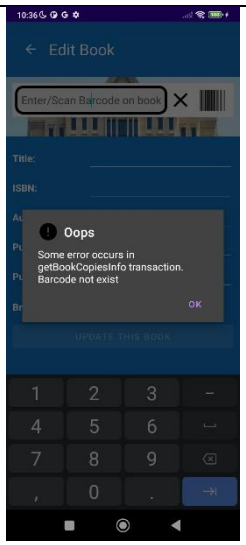
Items	Test case	Expected result	Actual result
Part A	Process return		
A1	Staff enters barcode of book copies with status 3 (on-return-request)	Ask for confirmation 'Do u want to PUT this book on shelf?' and show book title Success Message is shown after clicking Yes	 
A2	Staff enters barcode of book copies with status which is not 3 (on-return-request)	Error message 'The book is invalid for Return Processing." is shown	

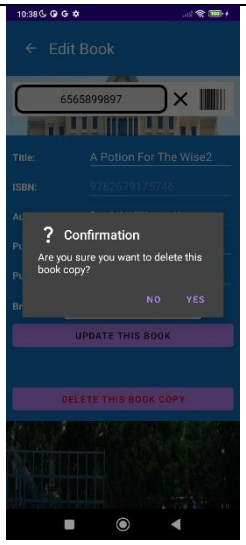
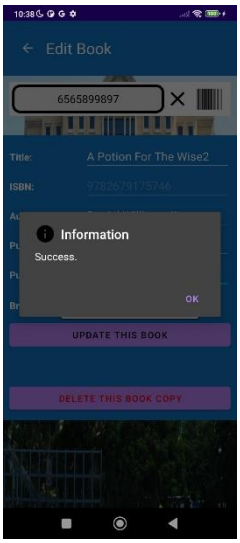
A3	Staff enters invalid barcode	Error message ‘Barcode not exist’ is shown	

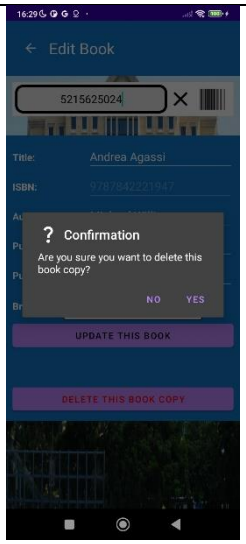
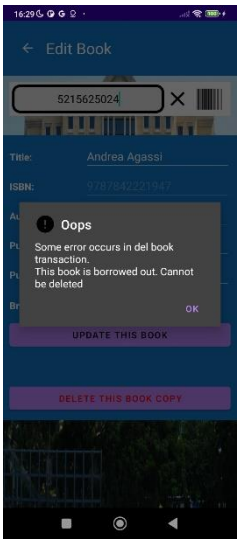
Part B	Add Book		
B1	Staff add a new book without any fields filled	Error message ‘Please fill in all the fields’ is shown	
B2	Staff add a new book with publishing year extremely small	Error message ‘Please enter valid publishing year’ is shown	
B3	Staff add a new book with ISBN length smaller than 13	Error message ‘Please enter valid isbn’ is shown	

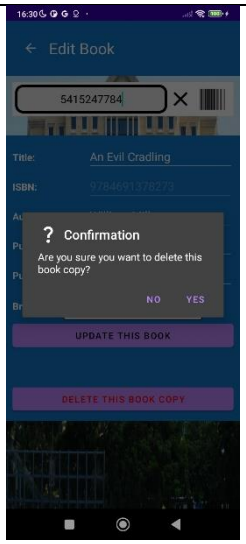
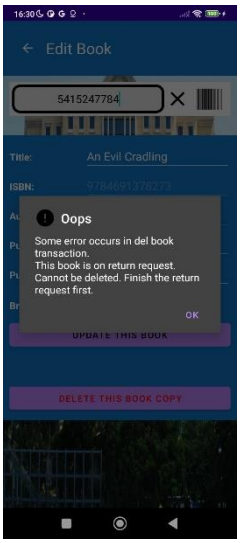
B4	Staff add a new book with ISBN already found in database	Error message ‘ISBN already exist’ is shown	
B5	Staff add a new book with all valid data	‘Success’ message is shown in a dialog box	
B6	Staff add an existed book without any fields filled	Error message ‘Missing fields found! Please fill in all the fields (title, barcode, location)’ is shown	

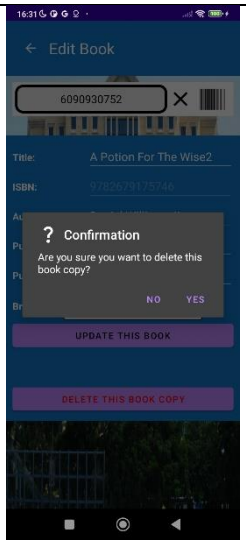
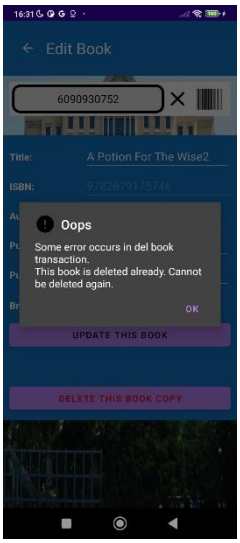
B7	Staff add an existed book with barcode already found in database	Error message ‘Barcode already exist’ is shown	
B8	Staff add an existed book with barcode length smaller than 10	Error message ‘The barcode length need to be 10 digits!’ is shown	
B9	Staff add an existed book with all valid data	‘Success’ message is shown in dialog box	

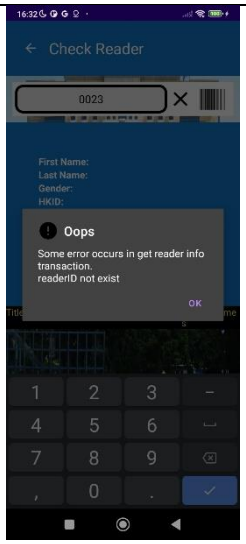
Part C	Edit Book		
C1	Staff updates title, author, publisher, publishing year and branch for a book copy	‘Suceess’ message and updated information like barcode, title, author, publisher, publishing year, location are shown in dialog box	
C2	Staff enters invalid barcode	Error message ‘Barcode not exist’ is shown in dialog box	

C3	Staff deletes a book copy with status 1	Ask for confirmation ‘Are you sure you want to delete this book copy?’ ‘Success’ message is shown in dialog box after clicking Yes	 

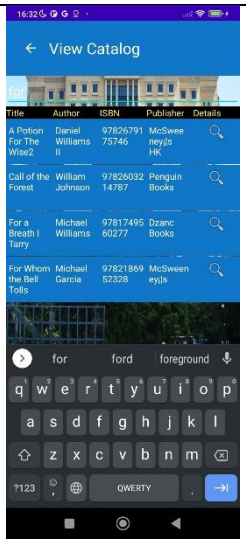
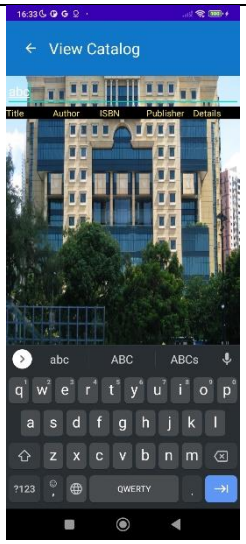
C4	Staff deletes a book copy with status 2	Ask for confirmation ‘Are you sure you want to delete this book copy?’ Error message ‘The book is borrowed out. Cannot be deleted’ is shown in dialog box after clicking Yes	 

C5	Staff deletes a book copy with status 3	Ask for confirmation ‘Are you sure you want to delete this book copy?’ Error message ‘The book is on return request. Cannot be deleted. Finish the return request first’ is shown in dialog box after clicking Yes	 

C6	Staff deletes a book copy with status 8	Ask for confirmation ‘Are you sure you want to delete this book copy?’ Error message ‘The book is deleted already. Cannot be deleted again.’ is shown in dialog box after clicking Yes	 

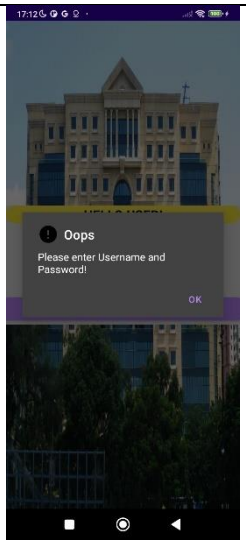
Part D	Check reader account		
D1	Staff enters a valid reader ID existed in database	Reader information and list of borrowed books are displayed	
D2	Staff enters an invalid reader ID	Error message ‘readerID not exist’ is displayed in dialog box	

Apendices E: Testcase – Common Functionalities

Items	Test case	Expected result	Actual result
Part A	Browse Catalog		
A1	User enters no keyword in searching field	All the books are displayed.	
A2	User enters keyword “for” in searching field	Only books with title containing keyword ‘for’ are displayed	
A3	User enters keyword “abc” in searching field	No books should be displayed	

A4	User views detail of a book by clicking an item on the book list	Book information, available branches and corresponding status are displayed	

Part B	View My Account		
B1	User views his own account	Account information are displayed	

Part C	Login		
C1	User leaves empty fields when login.	Error message ‘Please enter Username and Password!’ is shown in dialog box	
C2	User login using a reader account and password	User is redirected to the main screen for reader	
C3	User login using a staff account and password	User is redirected to the main screen for staff	

C4	User enters wrong username or password	Error Message ‘Please enter a valid username and password’ is shown